Contribution to the development of a methodology to build indoor overheating vulnerability maps at the city scale integrating climate change data and urban heat island

Obaidullah Yaqubi

THESE DE DOCTORAT DE

NANTES UNIVERSITE

ECOLE DOCTORALE N° 602
Sciences pour l'Ingénieur
Spécialité : Génie Civile

Par
Obaidullah YAQUBI

Contribution to the development of a methodology to build indoor overheating vulnerability maps at the city scale integrating climate change data and urban heat island

Thèse présentée et soutenue à Nantes, le 19 Septembre 2022
Unité de recherche : GeM CNRS UMR 6183

Rapporteurs avant soutenance :

Pascal Stabat	Professeur, Responsable du cycle Ingénieur en énergétique de Mines Paris Tech
Peter Riederer	Ingénieur de recherche, HDR, CSTB

Composition du Jury :

Président :	Bruno LACARRIERE	Professeur, IMT-Atlantique
Examinateurs :	Pascal STABAT	Professeur, École des Mines de Paris - Université PSL
	Peter RIEDERER	Ingénieur de recherche, HDR, CSTB
	Darren ROBINSON	Professeur, University of Sheffield, UK
	Bruno LACARRIERE	Professeur, IMT-Atlantique
	Stéphanie BONNET	Maitre de conférences Hors Classe, HDR, Nantes Université
	Marion BONHOMME	Maitre de conférences des universités, INSA Toulouse
Dir. de thèse :	Marjorie MUSY	Directrice de Recherche, CEREMA Ouest
Co-enc. de thèse :	Sihem GUERNOUTI	Chargée de recherche, CEREMA Ouest

Invitée :

Co-enc. de thèse :	Auline RODLER	Chargée de recherche, CEREMA Ouest

Table of contents

List of figures

List of tables

Résume

Le problème de la surchauffe intérieure des bâtiments devient de plus en plus un sujet d'intérêt pour la communauté scientifique ainsi que les décideurs politiques en matière d'urbanisme en raison de l'augmentation de la température moyenne de la Terre, de l'augmentation de la fréquence des événements météorologiques extrêmes, de l'effet d'îlot de chaleur urbain, et du fait que de nos jours la plupart des gens passent la majorité de leur temps à l'intérieur des bâtiments.

Le présent travail de recherche porte sur le développement d'une méthodologie pour l'évaluation de la vulnérabilité des villes à la surchauffe intérieure, visant à soutenir la prise de décision stratégique dans la planification urbaine pour les interventions politiques d'adaptation au changement climatique.

Compte tenu de la nature interconnectée des questions à traiter, ce manuscrit commence par un chapitre d'introduction détaillé présentant les concepts clés, les énoncés du problème, l'objectif de la thèse et la méthodologie globale employée.

Chaque chapitre suivant est consacré à une partie spécifique du travail de thèse. Ainsi, le deuxième chapitre de ce manuscrit porte sur la définition des typologies des bâtiments et l'identification des bâtiments représentatifs utilisés. Le troisième chapitre porte sur la prise en compte du changement climatique et les données sur les îlots de chaleur urbains dans les fichiers météo utilisés dans les simulations. Le quatrième chapitre présente les paramètres du bâtiment influençant sa performance thermique ainsi que les indices de mesure de la surchauffe intérieure. Le cinquième chapitre présente les modèles réduits ou métamodèles développés et la manière d'extrapolation des résultats de simulations des bâtiments représentatifs au reste du parc bâti à l'échelle de la ville.

Enfin, la conclusion retrace les principaux travaux et résultats développés au cours de la thèse. Elle pose ensuite les limites de ce travail de thèse dans chacun des domaines abordés et propose quelques pistes complémentaires de réflexion dans les perspectives de ce travail.

Abstract

The problem of indoor overheating is increasingly becoming a subject of interest to the scientific community as well as the policy makers in urban planning due to the rise in global average temperature, increase in the frequency of extreme weather events, urban heat island effect, and the fact that nowadays most of people spend the majority of their time indoors.

The present research demonstrates a methodology for an urban-scale indoor overheating vulnerability assessment, aimed to support strategic decision making in urban planning for climate-change adaptation policy interventions.

Given the inter-connected nature of questions needed to be handled, this manuscript starts with a detailed introduction chapter outlining the key concepts, presenting problem statements, research objective, and overall method employed.

Each subsequent chapter is dedicated to a specific part of the research effort. The second chapter of this manuscript is about building typologies definition and identification of representative buildings to be used. Third chapter is concerned with climate change and urban heat island data. Fourth chapter presents indoor overheating measurement indices and attempts to identify the most influential building parameters through a literature review. Fifth chapter covers surrogate models and how to extrapolate the simulations results of representative buildings to the rest of build stock.

Finally, the conclusion traces the main ideas developed during the thesis. It then sets out the limits of the study in each of the areas covered and develops some additional avenues for reflection in the perspectives of this work.

Acknowledgment

This research work was carried out with the support of CEREMA (Centre for Studies and Expertise on Risks, the Environment, Mobility and Urban Planning), a major French public agency for developing public expertise in the fields of urban planning, regional cohesion and ecological and energy transition for resilient and climate-neutral cities and regions.

Completion of a doctoral research project, such as this, required and greatly benefitted from the support, insights, and expertise of many people.

First of all I would like to thank members of jury for participating in the defense of this doctoral thesis, and particularly the reporters of my doctoral thesis, Pascal Stabat and Peter Riederer.

To Director and Supervisors of my thesis, thank you for accepting to supervise this research work and for being so supportive during these three years of thesis: **Marjorie Musy** for providing wise counsel and helping me stay on course throughout the research, **Sihem Guernouti** for orienting me to be more critical of the method and results and that way helping me dig deeper into the problem, **Auline Rodler** for the attentive reading of the manuscript, quick constructive feedback on the research and non-research related problems.

I would also like to thank CEREMA for offering me the opportunity to conduct this research thesis and for providing administrative, technical and financial support to accomplish all necessary tasks. I also thank my colleagues at CEREMA that made the journey of research more joyful.

Finally, I am thankful to my family, particularly my parents, for being so supportive and patient throughout all this.

Preface

Increase in global average temperature and in the frequency of extreme weather events, coupled with urban heat island effect and the fact the people spend majority of their time indoors, make the problem of indoor overheating a subject of interest to the scientific community as well to the policy makers in urban planning. In the latter, policy choices and decisions are aimed to foster benefits for all, which in turn requires an understanding of the degree of exposure and vulnerability to hazards at scales significantly larger than building. One way to provide this type of support for decision making at city scale is through development of rapid assessment tools that could link climate change data, urban heat island effect and individual performance of buildings.

With that in mind, the aim of this manuscript is to pave the way for the development of a comprehensive methodology for practitioners and policy makers in urban planning to take into account climate change scenarios, urban climate, energy transition and health in urban development policies.

Given the interdisciplinary nature of the problem tackled in this thesis, a process approach was adopted to accomplish all necessary tasks. Each chapter of the manuscript, here, is dedicated to the description of a major step of the research.

Chapter 1 sets out the manuscript by providing the context, principle elements involved in this study, objective, method of research, and a summary. The second chapter, after a critical literature review of building typologies construction methods, describes a data-driven method to aggregates residential buildings into clusters of buildings with similar characteristics, identifies one representative building from each cluster that undergoes a characterization step.

The third chapter starts with a state of the art on the following issues: global and regional climate models, future climate scenarios, climate downscaling approaches, weather processing tools, UHI calculation methods. Then it describes a workflow to generate future typical weather files from EUROCORDEX climate portal and compares them with 2003 heatwave weather data.

The fourth chapter of manuscript is divided into two parts. First part presents a literature review on the results of sensitivity analysis studies to identify what are the most influential building parameters for summer overheating and energy consumption. The second part of chapter four demonstrates various indices used in indoor overheating assessment and attempts to compare the performance with one another, where possible.

The fifth chapter aggregates three previous chapters and describes a rapid assessment tool based on surrogate modelling to extend the indoor overheating assessment study on reference buildings to the rest of build stock.

Finally, the conclusion traces the main ideas developed during the thesis. It then sets out the limits of the study in each of the areas covered and develops some additional avenues for reflection in the perspectives of this work.

Following articles were published during the research on this doctoral project.

Yaqubi, Obaidullah, Auline Rodler, Sihem Guernouti, Marjorie Musy **(**2021). 'Summer passive strategies assessment based on calibrated building model using on site measurement data'. *Proceedings to IBPSA 2021*, Bruges, Belgium. https://doi.org/10.26868/25222708.2021.30226

Yaqubi, Obaidullah, Auline Rodler, Sihem Guernouti, Marjorie Musy **(**2022). 'Creation and application of future typical weather files in the evaluation of indoor overheating in free-floating buildings'. *Building and Environment*. *216:109059* https://doi.org/10.1016/j.buildenv.2022.109059

Chapter 1 : Introduction, context, key concepts, and methodology

This chapter sets out the manuscript by introducing the context, principle elements involved in this work, objective, method of research, and a summary at the end. Following is the list of main elements covered in the chapter:

- Introduction to the subject: definition of key concepts, literature review
- Problem statements and objective
- Methodology and a summary

1.1 Background/Context

1.1.1 Climate change and heatwaves

The Intergovernmental Panel on Climate Change (IPCC) of United Nations defines climate change as "*A change in the state of the climate that can be identified (e.g., by using statistical tests) by changes in the mean and/or the variability of its properties and that persists for an extended period, typically decades or longer. Climate change may be due to natural internal processes or external forcings, or to persistent anthropogenic changes in the composition of the atmosphere or in land use"* (IPCC 2018).

Heavy downpours causing record floods, major hurricanes, unprecedented heatwaves: climate change manifests itself in a variety of ways. What differentiates climate change from natural weather variability is the long-term trends. Earth orbiting satellites, ocean buoys, and remote meteorological weather stations are the most commonly used instruments to monitor our current weather and climate information, but data collected from natural ice cores, corals, tree rings, and sediments from oceans and lakes enable scientists to extend the world's climatic records thousands of years back. Comparison of current climate data and past climatic conditions allow scientists to see long-term variations in earth's atmosphere, oceans, dry land, and glaciers.

Multiple factors, both anthropogenic and natural, influence earth's climate system. Natural factors affecting climate systems include sun's cyclical radiation intensity variations, volcanic eruptions, and variations in the concentration of naturally occurring greenhouse gases. However, recent past climate records indicate that our current climate warming, particularly the changes that have been happening since mid-20th century are much faster than ever before, and therefore cannot be explained by natural factors.

Anthropogenic causes, specifically the greenhouse gas (GHG) emissions generated from human activity are the main cause of the earth's rapidly changing climate today. Due to these activities, concentration of CO2, methane, and nitrous oxide in the atmosphere are unprecedented in the past 800`000 years. CO2 concentration alone has risen by 46% since preindustrial times (Rosbakh, Auerswald, and Poschlod 2021).

It is important mentioning that while climate change and global warming are frequently used interchangeably, global warming is just one aspect of climate change and it refers to the average global temperature rise near the surface of the earth.

In 2020, the global average temperature was approximately 14.9 ° C, which is 1.2 ° C higher than the pre-industrial (1850-1900) level (see Figure 1-1). According to World Meteorological Organization (WMO) the six years period since 2015 has also been the warmest on record for the most part of Europe, Middle East, and northern parts of Asia (WMO 2020) .

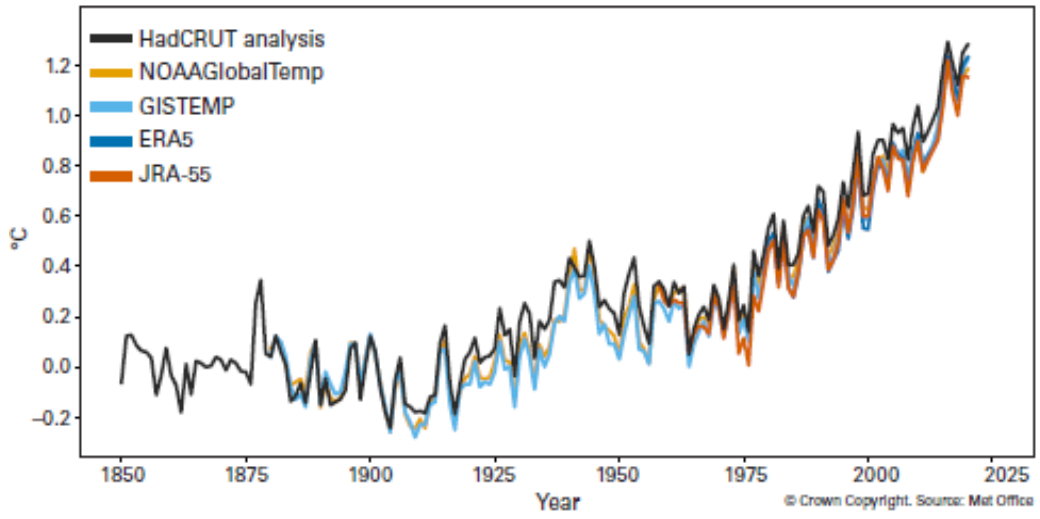


Figure 1-1: Global temperatures – change from pre-industrial. (Source: WMO)

All the key indicators and associated impact information presented in the report of WMO point to unrelenting global warming. Unfortunately, the negative trend will continue to worsen in the coming decades regardless of how successful we are in reduction of anthropogenic sources.

Looking at the current global average temperature and data of the past decades raises the question of how is the climate going to change in the coming century? The answer to this question depends entirely on how human societies develop in terms of demographics, economics, technologies, demand and supply of energy, and land usage.

To give a more accurate answer to this question, IPCC has released special report on greenhouse gas (GHG) emission level that presents a set of scenarios taking into account various driving forces and emissions in scenario literature. In this report, it has defined four reference scenarios, the RCP (representative concentration pathway), each illustrating one possible evolution profile of GHG concentrations as a function of socio-economic development scenarios, technological changes, and various climate adaption and mitigation strategies (see Figure 1-2). Scenarios presented in this report do not include probabilities of occurrence; therefore, preference for scenarios in the research or any other sector is subjective and can vary among users.

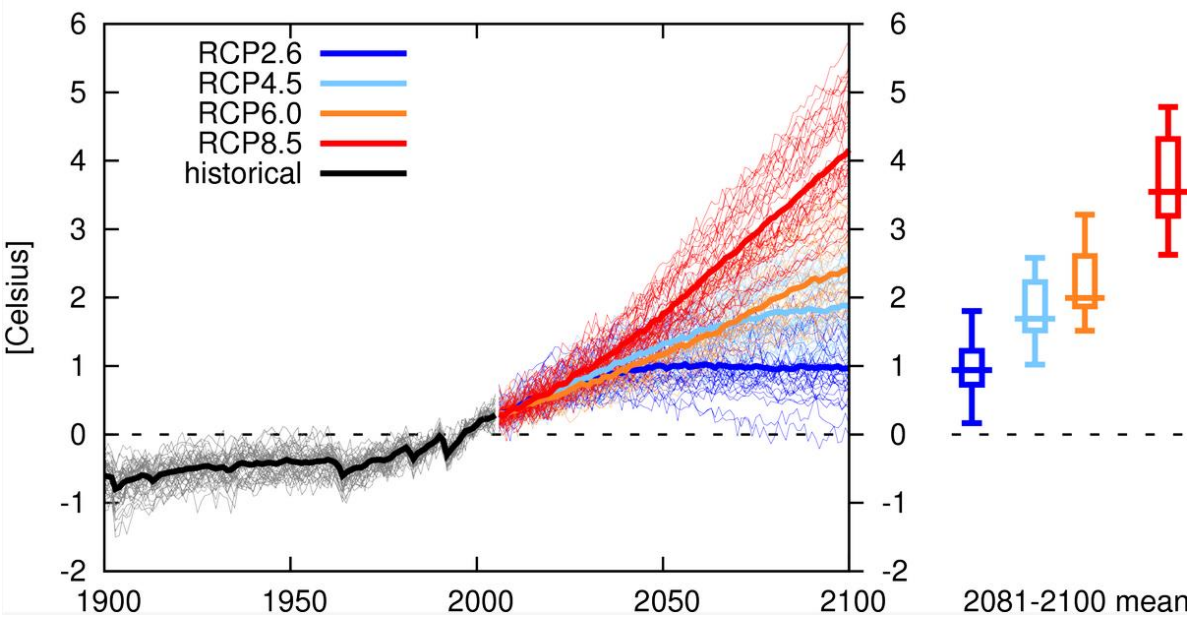


Figure 1-2 : Global average temperature projection scenarios. (source:KNMI)

Disaster and surprise events are also not included in these scenarios but previous studies have shown that rise in global average surface temperatures due to climate change is accompanied by an increase in frequency and intensity of extreme weather events such as heatwaves (Lorenzo, Díaz-Poso, and Royé 2021).

National Climate Assessment report finds that the number and the strength of heatwaves, heavy downpours, and major hurricanes have significantly increased over the last decades. These extreme changes in temperature and precipitation can disrupt and damage critical infrastructures and vitality of communities. It increases health risks associated with air quality, heatwaves, floods, wildfires and ground-level ozone pollution (Doherty et al. 2018).

Europe in particular is more affected by heatwaves and cold snaps compared to other extreme weather events like hurricanes that form in tropical and subtropical latitudes.

An example is the exceptional heatwave in summer of 2003 that resulted in at least 30,000 excess deaths in Europe, of which nearly 15,000 where in France, between August 1 and 20, 2003 (Wagner 2018).

(R. Zhang et al. 2020) presented evidence obtained through observational analyses and numerical studies, which illustrated that the rise in frequency of European heatwaves is linked to decrease of Arctic sea ice concentration and Eurasian snow cover fraction. Future projection analysis of numerical simulations by the same authors also suggests that Europe may experience more hot summers as both Arctic sea ice concentration and Eurasian snow cover fraction continue to decline.

(Ouzeau et al. 2016) simulated heatwaves using EURO-CORDEX regional multi-model and concluded that under future climate conditions, no matter what scenario considered, the frequency, duration and intensity of heatwaves increase across France and other parts of Europe.

Authors state, heatwave events could occur during a larger span of summer time and the 2003 event would be a typical event by the end of the century. Authors also state that the duration and intensity of 2003 event would be much lower than the strongest heatwaves that will occur during the last 30 years of 21st century (Ouzeau et al. 2016). Figure 1-3, below, demonstrates the historical heatwave data and projected heatwaves assuming the high-emission scenario (representative concentration pathway (RCP) 8.5) from Meteo France.

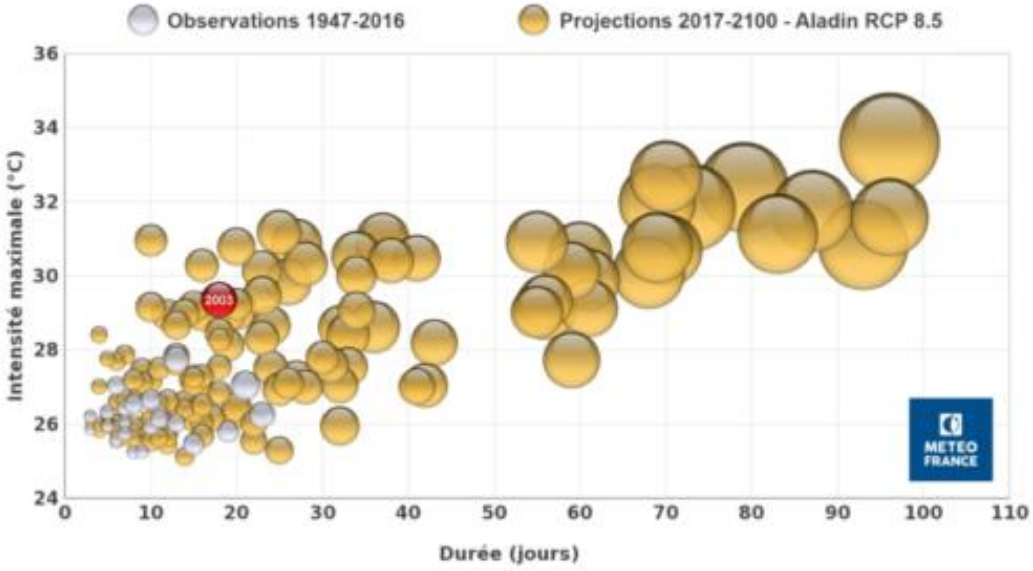


Figure 1-3: Heatwaves in France, historical records and future projection under climate scenario RCP8.5 (Source:MeteoFrance)

(I4CE 2019)

As can be seen in Figure 1-3 the intensity and duration of heatwave in 2003 is significantly smaller than what is expected to happen in the second half of this century with business as usual emission-scenario.

Report from JRC PESETA IV project that studies the biophysical and economic consequences of climate change states that since 1980, heat and cold waves have claimed the lives of nearly 90’000 individuals in Europe. If global average temperature stabilizes at 1.5 °C by 2100, each year more than 100 million Europeans will be exposed to heatwaves that is considered “intense” in our today’s definition of it. With unmitigated climate change (3°C by 2100), this number rises to 300 million each year. This number stands to 10 million per year in our current climate conditions (1981-2010).

Report also outlines that yearly fatalities caused from extreme temperatures could rise from current 2’700 deaths to around 30’000 in 2100 with a 1.5 °C global average temperature increase. This could reach to 50’000 with 2 °C and around 90’000 with 3 °C. As can be seen in Figure 1-4, this raise in the number of fatalities is most pronounced in southern Europe (France, Spain, Greece, and Italy) due to increased exposure rate and extreme heat (European Commission. Joint Research Centre. 2020). The most vulnerable group of people are those who have reduced physiological and behavioural capacity to regulate thermal conditions and those

who do not have access to technological means (e.g. air conditioning) due to financial constraints (European Commission. Joint Research Centre. 2020).

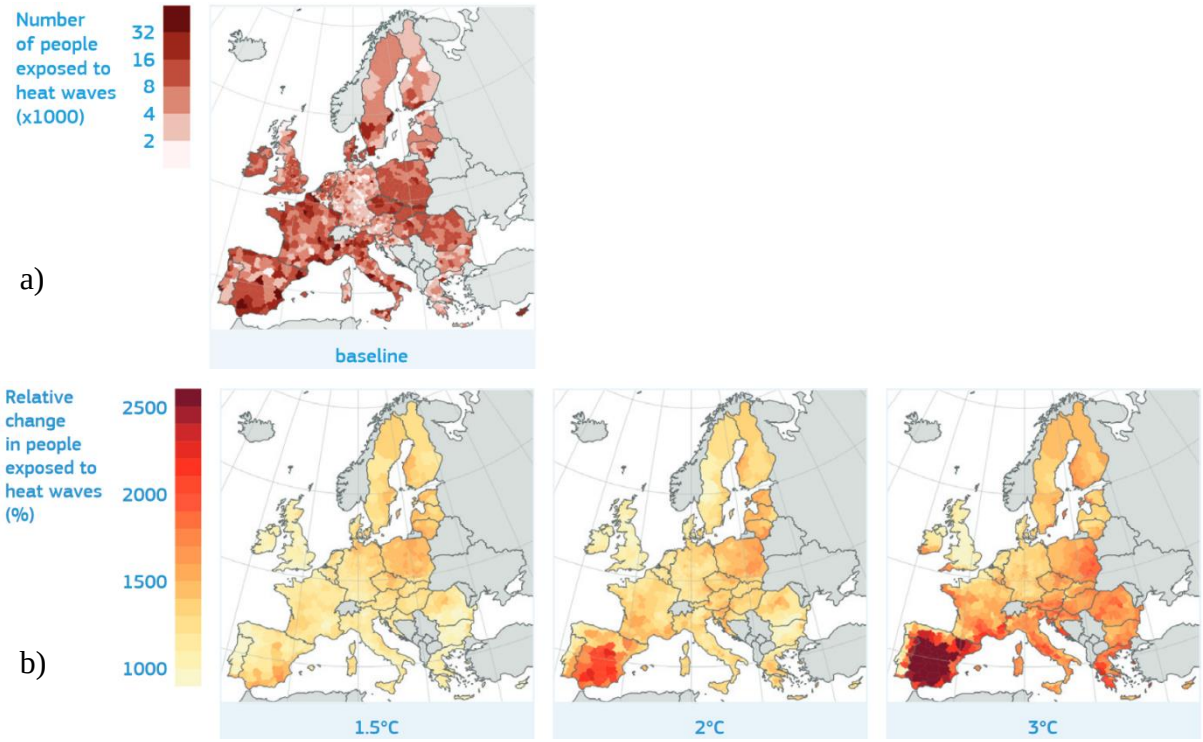


Figure 1-4: a) Number of people annually exposed to a present 50-year heatwave. b) Projected changes in human exposure to these events for 1.5°C, 2°C, and 3°C global warming (Source:JRC PESETA IV)

Extreme heat also raises the occurrence risk of other types of climate related disasters. It can exacerbate intensity and frequency of droughts, or cause wildfires. Droughts and wildfires in turn create a web of impact that span across a wide range of economic and social sector such as crop failure, power supply disruption, variation in composition and structure of vegetation, etc.

Climate change will affect food production beyond just crop production; it affects livestock, fisheries, and aquaculture. Effects of climate change on human health are not limited to mortality from exposure to extreme heat, but also include effects on human mortality and morbidity from less extreme sub-optimal temperatures, air quality, and water and vector borne diseases.

Climate change could cause more population displacement, increase the risk of conflicts over water or other natural resources. Most of these impacts listed above and those that have not been mentioned are beyond the scope of this study, which primarily focuses on over-temperature, and vulnerability of people to it.

Influences of changing frequencies and intensities of extreme heat and global warming **are believed to be exacerbated** in the urban areas by a distinct urban microclimate feature known as *urban heat island (UHI)* effect.

1.1.2 Urban heat island

As urban areas grow in size and density, significant changes take place in their landscape. Buildings, bridges, roads, parking lots, and other infrastructures built from dense materials replace open land and green areas. Surfaces that were once covered with greenery and permeable natural soils turn into impermeable and dry high-density surfaces. These changes make urban regions warmer than their peripheries, forming an "island" shape of higher temperature in the urban landscape.

During a hot summer, day surface temperature of buildings, roads, infrastructures (bridges, pedestrians, etc.) can become 10 to 30 °C hotter than the near surface air temperature, causing them to alter heat exchanges in urban settings between surface and near surface temperatures.

Urban surface materials, due to their thermal characteristics (specific heat, mass, conductivity, diffusivity and emissivity) store 15 to 30% more heat than natural materials during the day. In turn, these surfaces radiate additional heat into the atmosphere, causing a similar but not so extreme, increase in air temperature.

The difference in air temperature between rural and urban area becomes more pronounced after sunset, as urban surfaces continue to radiate heat into the surrounding environment, preventing the air temperature of the urban area from dropping during the night. Lack or reduced presence of evapotranspiration (e.g. through lack of vegetation, water surfaces) in urban areas is another major set of factors that causes an increase in the intensity of UHI. Third set of factors influencing UHI effect, which is intertwined with the previous two, is urban morphology (geometric form of structures) (Oke et al. 2017).

Figure 1-5 : Left, urban heat island explained by (Jolma architects, 2018); Right, Nantes Urban heat Island nocturnal intensity map produced by (Bernard, 2017) for a clear day of summer

A considerable body of research has been carried out on how to quantify intensity, extent and scale of UHI as well as main factors influencing the intensity of UHI effect. Literature review carried out by (Tzavali, Paravantis, and Mihalakakou 2015) concluded that the intensity of UHI effect varies depending on city size, land use, topographic factors, vegetation ratio, urbanization, industrialization, time of the day, season of the year, and prevailing

meteorological condition. On scale and type of UHI in the literature, (S. W. Kim and Brown 2021) analysed 51 case studies of UHI across the globe and classified them on conventional UHI classification types, horizontal ranges and vertical positions. A summary of literature review on UHI scale and types is presented in Table 1-1, below (S. W. Kim and Brown 2021).

Table 1-1: Scale and types of UHI studied in literature

Type of heat islands	Horizontal ranges	Vertical layers	Urban unit	Data time scale	Data collection methods
Boundary layer Heat Island (UHI_{UBL})	Macro (100 s Km <)	Urban boundary layer (UBL) (250–2500 m)	Urban region	Hours-days	Historical weather data, remote sensing
	Meso (10s – 100 s km)		A city	Hours- 1 day	
Canopy layer Heat Island (UHI_{UCL})	Local (o,5 – 10s km)	Urban canopy layer (UCL) (25–250 m)	A neighbourhood, blocks	Minutes–hours	Stationary weather stations, temporarily fixed weather stations, mobile measurements with weather station mounted on mobile platforms
	Micro (100s M – 0.5 KM)	Building canopy layer (BCL) (10S M - 100 S M)	Buildings, street canyon	Seconds–minutes	Temporarily fixed weather stations, mobile measurements with weather station mounted on mobile platforms, thermal imaging camera, remote sensing
Surface Heat Island (UHI_{Surf})		Land surface layer (LSL) (<10S M)			

(Oke et al. 2017) in their book on urban climates, in addition to the three types mentioned in Table 1-1 have included a Subsurface Heat island (UHI_{sub}) as well, depicted in Figure 1-6.

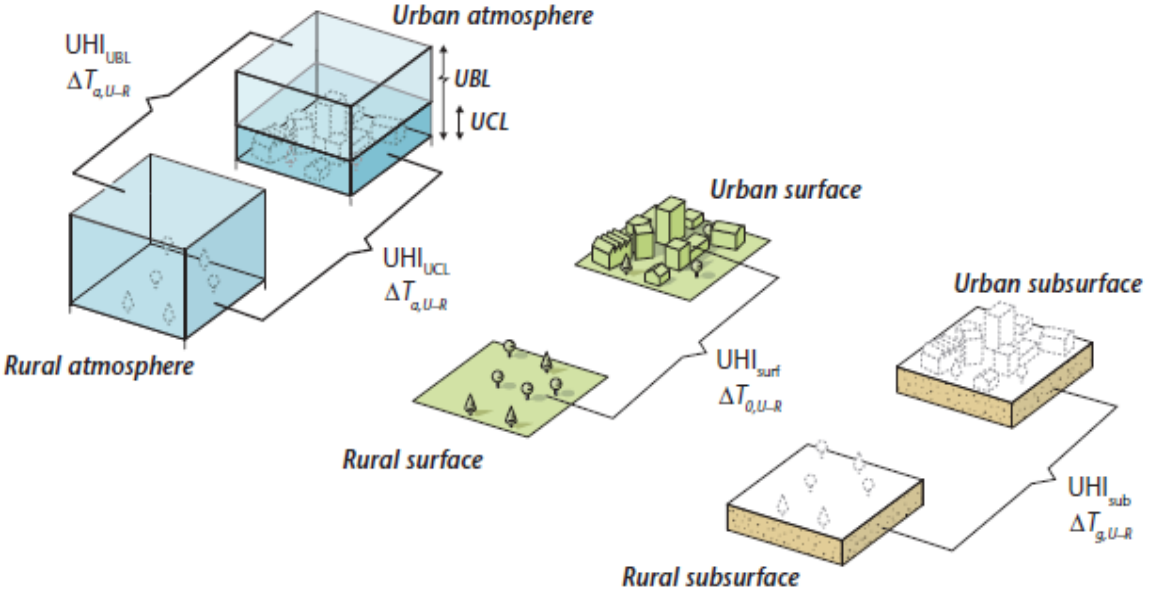


Figure 1-6 : Illustration of temperature differences in four types of UHI (source:Oke et al. 2017)

Intensity of UHI is an important indicator often used to measure severity/magnitude of UHI effect in urbanized areas. This indicator is dependent on the type and scale of UHI.

One way to estimate/calculate UHI intensity is to compare maximum and/or average air temperature of an urban area with its surrounding (rural area) as shown in Equation 1-1.

$$UHI\ intensity = \Delta T_{u-r} = T_u - T_r \qquad \textit{Equation 1-1}$$

Where: $\triangle T_{u-r}$ Difference between urban and rural air temperatures

T_u (Average/maximum) air temperature in urban area

T_r (Average/maximum) air temperature in rural area

Another approach to measure UHI intensity is with the concept of energy balance. In contrast to the first approach which is entirely based on temperature difference, the energy balance concepts calculates and analyses sizes and types of various heat fluxes generated within the studied spatial unit (Oke et al. 2017).

In simple terms, energy balance is the statement of energy conservation applicable to volumes and surfaces at all temporal and spatial dimensions. In urban heat island intensity calculation, this concept is used to assess the transfer and storage of energy within urban systems and in between atmosphere and urban system. Its applicability to all spatial scales allows it to be written for individual facets (roofs, walls, streets, green surfaces, etc.), for urban units with the urban climate (people, buildings), for a whole atmosphere-surface interface, or for specific layers of atmosphere.

In order to interpret observations accurately, communicate with unambiguity and compare outcomes, it is critical to conduct a more detailed investigation on the four types of UHI, their causation, spatial and temporal variations, as well as what are the impacts at any given situation.

The four types of UHI effect are as follows:

The **Surface Urban Heat Island effect (UHI_{Surf} or SUHI)** is mostly determined by the geometrical, thermal, and radiative properties of the surface facets. Satellite sensors are usually used to measure temperatures and they show that its magnitude reaches to its peak in clear daytime conditions mostly in the parts of city that have no or little vegetation, or where large portion of the urban surface area is made up of roof facets. In addition, its magnitude is sensitive to green surface coverage ratio of rural areas surrounding the urban agglomeration.

A more accurate assessment of the spatial variation of urban reference temperature (T_U), that could be used in Equation 1-1, from satellite sensors, requires corrections for atmospheric and surface emissivity effects, and preferably for thermal anisotropy for a 3D-surface city, which seen from a satellite sensor platform, depends on both the viewing angle and the solar geometry.

Rural reference temperature (T_r) can also be difficult to define because it is quite common to find that non-urban areas exhibit similar, if not greater, spatial variability than that of urban areas. It requires consideration for surface types, moisture level in the soil, shadows and topographic variations such as elevation, slope, proximately to water bodies (Oke et al. 2017).

Of the four, the most studied type is **Urban Canopy Layer Heat Island (UHI_{UCL})**. The magnitude of UHI_{UCL} rises and falls in response to temporal variations (time of day or season of the year) and meteorological conditions (wind speed and direction, clouds). UHI_{UCL} is observed/measured by thermometers (thermocouples) installed near surface to measure temperature (To) in urban and rural areas**,** which are then used as input to Equation 1-1. Near surface air temperature data can be collected in a weather screen or ventilated radiation shield at one or more sites that are considered to represent urban and rural climates (Stewart and Oke 2012a). This stationary approach is also called "fixed". If a fixed station has capacity to continuously monitor climate conditions, then it can capture temporal variations too. An alternative approach is to mount a thermometer on a vehicle and transverse it across a settlement and then to its non-urbanized surroundings. This approach is called "transverse". The latter allows studying both temporal and spatial variation of UHI_{UCL}.

As can be seen in Figure 1-6, UHI_{UCL} is the difference between the air temperature of near surface, below roof level, in the city and the temperature of near-surface air over its non-urbanized surroundings. Its magnitude reaches its peak after sunset when air above the urban areas cools more slowly than air above rural landscapes. Because the rate of night-time cooling, is inversely linked to the sky view factor (SVF), the magnitude of the UHI_{UCL} is greatest when buildings are tall and streets are narrow (i.e. city centre locations with limited greenery). During the day, UHI_{UCL} is frequently low or even negative (Oke et al. 2017).

Boundary layer UHI (UHI_{UBL}) is closely coupled with UHI_{UCL}, but it is above roof-level, has different magnitude, and is generated through a different process (Oke 1995). As can be seen in Figure 1-7, UHI_{UBL} forms a massive urban plume both by day and night (Junyan Yang et al. 2020).

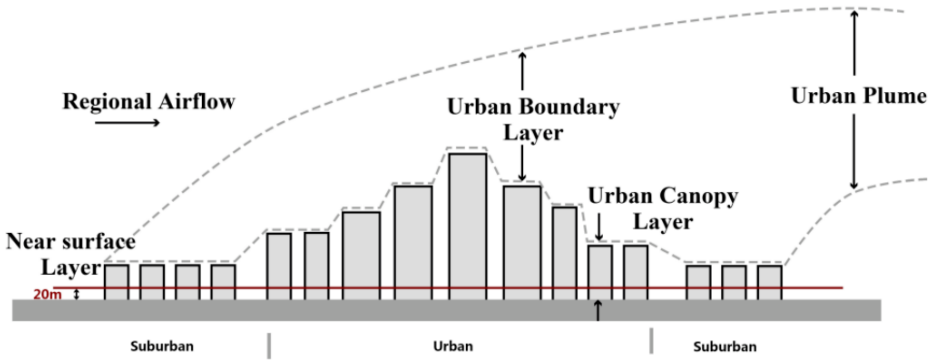


Figure 1-7: Urban climate scales and potential temperature profiles due to various UHI effects (source: Junyan Yang et al. 2020)

This giant plume is maintained by an enhanced sensible heat flux from urban area. In reality, it is a mixture of various internal boundary layers, which develop downstream from various land-uses plus the plumes of heat, water vapour, and pollutants from different sources.

The UHI_{UBL} is much less frequently monitored, as it requires very tall observation towers, aircrafts or balloons. Nowadays, UHI_{UBL} can be inferred utilizing ground-based remote sensing with profiling radiometers. Due to advection of warmer urban air downwind over rural surfaces,

it is important to consider the wind direction when finding suitable rural reference temperatures (Oke et al. 2017).

Subsurface UHI (UHI_{Sub}) is defined as the difference in ground temperature of urban area above that found at the same depth in surrounding no-urban area. It is formed because of sensible heat migration from the urban surface and urban infrastructure into the ground over an extended period. Its evidence are often obtained from thermometers mounted inside water wells or boreholes that run from a few 10s to 100 m in depth (Oke et al. 2017).

On the question of **factors causing UHI**. In 1973, a link was first proposed between urbanization-induced warming and the size of the city, as measured by the population, based on night-time air temperature (Oke 1973). With the proliferation of remote sensing measurement technologies of earth's surface temperature, similar relationships have been proposed on a global scale. Given the complexity of urban systems, it remains difficult to identify and isolate all the causes of UHI and the factors contributing to the observed differences in ΔT in the cities (Manoli et al. 2019). Nonetheless, a brief study of literature on the main causes of UHI revealed that the underlying factors influencing intensity of UHI differ by the type of UHI studied. According to (Oke et al. 2017) the main causes of UHI are presented in Table 1-2.

Table 1-2 : Potential causes of UHI

Cause	**Description of the cause**
Canopy layer heat Island (UHI_{UCL})	
Surface geometry	(a) Increased surface area ($\lambda_c=1$) (b) Closely-distanced buildings - Greater shortwave irradiance absorption due to multiples reflections of building surfaces (lower system albedo) - Small sky view factor ($\Psi_{sky}<1$) that reduces net longwave heat loss, particularly at night - Wind shelter in UCL reduces heat losses by convection and advection.
Thermal properties of surfaces	Artificial materials used in construction have a greater heat storage capacity that later release larger sensible heat
Anthropogenic heat (human activity)	The amount of anthropogenic heat released in cities because of fuel combustion and electricity consumption is substantially higher (AC, vehicles, machineries, etc.).
Urban 'greenhouse effect'	More downward longwave radiation is emitted to UCL by a warmer, more polluted, and frequently moister urban atmosphere.
Decreased evapotranspiration	Construction materials increase imperviousness of urban surfaces. Water remained on the surfaces after a rain evaporates faster than rural areas

Boundary layer UHI (UHI_{UBL})	
Polluted boundary layer	Aerosol and gaseous pollutants in urban atmosphere change radiation transmission causing a greater absorption and scattering of shortwave and a larger absorption and emission of longwave radiations.
Sensible heat flux	Larger turbulent sensible heat flux from rougher, warmer city surface. Upward mixing of warmer canopy layer air (i.e. UHI_{UCL}).
Anthropogenic heat	Heat sent upward into UBL from chimneys and factory stacks.
Entrainment	Stronger convection causes greater injection of warmer, drier air from above capping inversion, down into UBL.

λ_c = Complete, or three dimensional aspect ratio ($\lambda_c = A_C/A_T$), where A_C = Complete surface area and A_T = Plan area of total surface ; Ψ_{sky} = Sky view factor

Overall, factors affecting UHI can be summarized as follows: increased imperviousness, modified urban geometry, low albedo of urban surface materials, increased population density, greater anthropogenic heat release, and reduced presence or absence of vegetation (Mohajerani, Bakaric, and Jeffrey-Bailey 2017; Oke et al. 2017; Vujovic et al. 2021).

UHI effect has various **direct and indirect impacts** on the well-being of urban inhabitants, their sleep quality (Y. Li et al. 2020), as well as on attractiveness of public spaces in city centres, energy consumption (air conditioning), resilience of infrastructures and urban networks, preservation of flora and fauna biodiversity. In a detailed study on the perceived impact of UHI, (Aghamohammadi et al. 2022) categorized impacts of UHI in five themes: (1) public health deterioration, (2) acceleration of urban migration patterns to spend more time in cooler areas, (3) productivity reduction, (4) increase in household energy consumption, (5) and deterioration of environmental quality and natural resources.

Increased temperature in urban areas often exerts greater pressure on urban microclimate consequently causing variations in precipitation pattern, natural air circulation, water quality, and air pollution. (Wang, Guo, and Han 2021) studied the relationship between UHI intensity and found a statistically significant correlation between daytime UHI intensity and increased concentration of ground-level ozone (O_3).

Additionally, elevated urban temperature acts as a precursor for the photochemical reactions in the atmosphere enhancing urban smog (H. Li et al. 2018). Urban smog in turn triggers a wide range of medical complications such as respiratory problems, and even cardiovascular failures (Tan et al. 2010).

This phenomenon poses a threat to human health more than ever because the majority of the world's population now live in densely built cities, and warming of these areas can significantly increase morbidity and mortality, particularly during heat waves (Manoli et al. 2019).

Interaction of global **climate change** with **UHI effect** is an open question for researchers now. (Wilby 2008) investigated urban impact on climate and vice versa. The results of their study, under high emission climate scenario showed further intensification of nocturnal heat island and greater concentration of ground-level ozone, which are both most pronounced in summer.

(Sachindra et al. 2016) in a similar study concluded that the presence of urban structures can magnify the effects of global warming on cities more than less urbanized areas.

(Oke et al. 2017) argue that we cannot simply assume that global temperature increase will raise the background temperature and its impact on UHI effect is additive. Because the definition of UHI effect is based on difference between urban and adjacent rural temperatures and the intensity of both can be modulated by environmental conditions and background temperatures. There is high degree of uncertainty here, because in the future both environmental conditions and background temperatures are expected to change.

(McCarthy, Best, and Betts 2010) used a global climate model, across multiple regions around the globe that could account for varied land cover fractions, as well as an urban land-surface scheme to account for the physical presence of cities and associated expected heat flow from human activities. Their results demonstrate that climate change has the ability to affect the climate potential of urban heat islands, with an increase of up to 30% in some places but a worldwide average reduction of 6%. Meaning that the impact of global climate change on UHI will differ from region to region. Their findings also revealed that raising global CO_2 concentrations from 323 to 645 ppm increased UHI by less than 0.5 °C, far less than the heat associated with global driving, which is 3 °C for the same level of CO_2 concentration increase. The authors also state that climate change will further increase the disparity in extreme nocturnal temperature between rural and urban areas.

(Lemonsu et al. 2013) studied Parisian urban climate under changing global climate conditions, following two emission scenarios, aiming to quantify the impact of global warming on urban and surrounding urban areas. Contrary to expected outcome, their results showed that during summer, under future climate scenario, the warming trend is more pronounced in adjacent rural areas than urban neighbourhoods in Paris due to soil dryness. For that reason, a significant decrease in nocturnal UHI_{UCL} (greater than 2°C) is noted. They emphasize that the extremes in temperature are more significant in suburban areas as the effects from partial urbanization accumulates with dryness of the soil. On the flip side, urban geometry is less dense than in the city centre, reducing shadow effects and promoting air warming in the street-canyon. Furthermore, natural soils' dryness considerably reduces evaporation, which then boosts sensible heat release. That is why, in cities surrounded by dry regions, this phenomenon frequently results in the formation of “cool island” during the day (Lemonsu et al. 2013).

On the interactions of global climate change and UHI, it can be concluded that global climate change will enhance the effects of UHI, although it might not influence its magnitude considerably. Nonetheless, local mitigation and adaptation strategies of UHI will still be needed to offset the impacts of global warming on urban areas.

1.1.3 Buildings and summer heat

Comprehensive time-activity studies in Europe and US have shown that people on average spend 16 hours/day indoors. This number increases to approximately 20 hours/day for those above 64 years old (Brasche and Bischof 2005), asserting the importance of indoor air quality and indoor thermal comfort.

Average lifetime service of buildings in France are estimated at 60 years (Mauro 2013). Reinforced concrete buildings in particular could render services for more than 100 years. Service quality of buildings decline as they age due to failure of building systems, appearance of cracks in the outside walls, weathering, opening of joints, moisture accumulation in the insulation layer, etc. 75% of buildings in Europe were constructed before 1990. It indicates that some of them have already completed their lifetime service and need continuous maintenance and retrofits to function (Park et al. 2020).

Projected variations in extreme weather events and global temperature increase will further increase pressure on buildings, making them uncomfortable or even potentially dangerous to occupants' wellbeing (Green et al. 2016; Hamdy et al. 2017a; Jun Yang et al. 2019). Heatwaves in particular can cause severe overheating in buildings that are not equipped to cope with it. It could lead to several problems ranging from thermal discomfort and productivity reduction to illnesses and even death of occupants (Hamdy et al. 2017a).

This concern is particularly relevant for buildings that are also subject to UHI effect. Co-occurrence of heatwave and UHI will further aggravate the pressure on buildings and as a consequence, the risk of indoor overheating in buildings is expected to rise.

Overheating in residential buildings already in Europe and North America has been reported by (Baborska-Narożny, Stevenson, and Grudzińska 2017; Lane et al. 2014; Lee and Shaman 2017), indicating rising concerns about overheating in temperate climate regions.

The magnitude of occupant vulnerability inside the building due to overheating depends on several parameters such as duration and intensity of exposure to heat, as well as, on personal adaptation capacity of the occupant. Installation of cooling systems on already energy intensive building sector could mitigate associated risks. However, the resulting energy demand would affect global climate change. Moreover, if installed in every household these systems would dramatically increase the electricity demand for cooling at peak time and at the same time discharge hot air that will further intensify urban warming. Another phenomena that affects occupants' vulnerability to future climate conditions and heatwaves is summer **energy precariousness** (Battersby 2016) which is especially true for naturally ventilated , and poorly insulated buildings that have traditionally not relied on mechanical systems to keep occupants safe from overheating during summer. Installation of a new cooling mechanism on such buildings will put a huge financial burden on households that are already in difficulty paying for heating needs.

French law recognizes energy precariousness as the inability or difficulty of people to have access to the required energy for basic needs due to lack of financial resources or poor housing conditions. Low-income households are considered to be more vulnerable to climate variations, in large due to this issue. As the average global temperature increases and energy precariousness remains common, adaptation to higher indoor air temperature while minimizing energy needs for mechanical cooling becomes more important than ever.

On the question of exposure to summer over-temperature, the building itself plays an important role. (Petrou et al. 2019) performed a statistical study of building stock in UK to analyse indoor temperature in different types of dwellings in an effort to identify the links between factors that

increase or decrease the risk of over-temperature in the buildings. They found out a considerable correlation between the size of the building and vulnerability of occupants in it. The latter decreases with the size increasing. They also showed that buildings built before 1900 were cooler than those built after. The study also revealed multiple correlations between household indoor overheating problem and dwelling characteristics, highlighting the complex nature of it.

Another new trend in energy efficiency design is Passivehause, which is promoting continuous high insulation of the entire envelope and good standard of airtightness. Is this type of construction perceived to be more prone to over-temperature compared to "**draughty**" homes?

To answer this question, (R. Mitchell and Natarajan 2019) investigated 82 buildings in UK constructed based on Passivehause principles. According to the authors, most of the buildings as a whole passed the overheating design criteria, however a notable number of individual bedrooms experienced high temperature during the night. The degree of discomfort in their paper was analysed with chartered Institution of Building Service Engineers TM59 criteria (Chartered Institution of Building Services Engineers 2013). The authors suggested a more in-depth, room-by-room approach in assessment of over temperature for building designers. A major limitation of their study was that they used historical weather data from 2011 to 2017 that were mostly mild and cool years.

Zero Energy Hub (NZH) conducted a survey and in their report in 2015, stated that 70% of housing providers reported overheating within their wider build stock. In their survey the highest risk was identified in single-aspect-high-rise apartments (apartments that have three closed sides) facing south, located in dense urban areas.

According to NZH, there are a number of factors causing overheating in buildings, and it often arises when these factors and processes act together (see Figure 1-8).

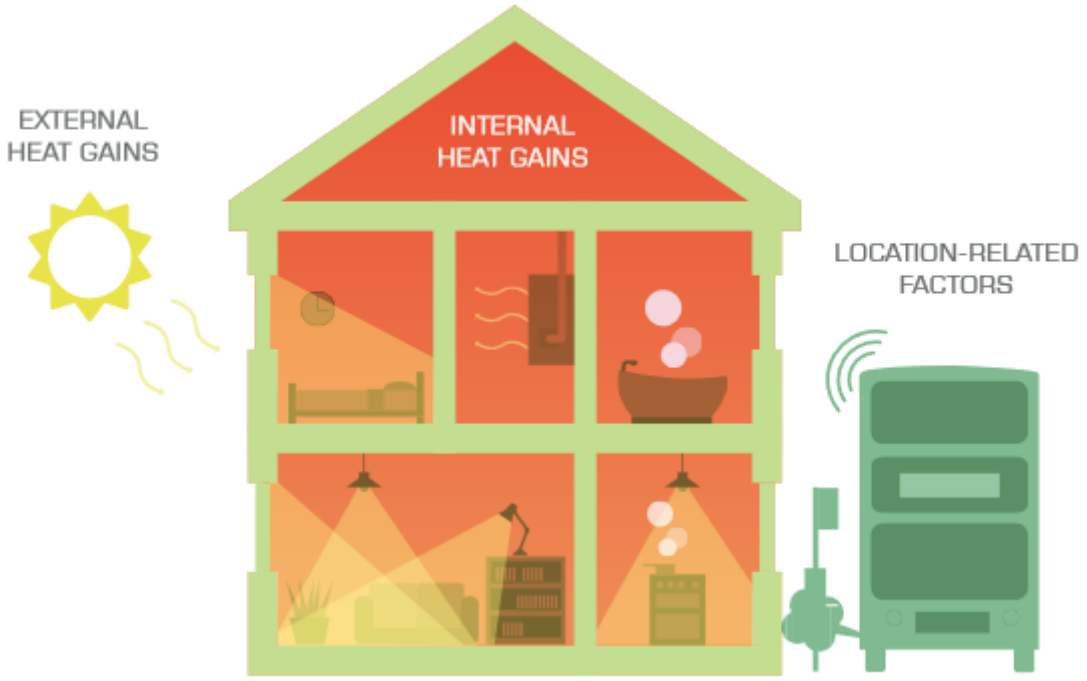


Figure 1-8 : Illustration of three main causes of overheating in buildings (Source: NZH)

(Fosas et al. 2018) through building simulation studies showed that improving insulation does not cause an increase in overheating risk, if the building is designed well with good solar shading and ventilation, particularly during the night.

On the contrary, some studies such as by (Beizaee, Lomas, and Firth 2013; Kotol et al. 2014) have suggested that overheating risk is amplified by an increase in insulation of envelope and improvement of airtightness. They also stated that buildings constructed after 1990 performed worse than those built before.

Examples of opposite conclusion also exists in the literature. (Salagnac 2007) has recommended improving insulation to better prepare buildings for future heatwaves and reducing overheating risk.

Studies that monitored indoor thermal condition in the house have shown that some houses overheat with increased insulation but because the evidence points in both directions, it has been difficult to establish a solid causality between overheating and envelope's thermal capacitance (R. Mitchell and Natarajan 2019). Factors causing overheating/discomfort in buildings are further studied in **chapter 4** of this manuscript, through case studies.

In addition to energy demand and thermal performance variations, some studies have suggested that over temperature also impacts structural integrity of buildings. (Salagnac 2007) points out to the clay soil subsidence and swelling triggered by dryness and rehydration of the environment, which then affects the foundation of concerned buildings. The author states that this phenomenon affects individual houses more than large multi-storey buildings that have deep foundations. Author suggests preparation of Cadastral plans showing location of clay soil areas as a preventative measure to reduce risks associated with this phenomenon.

1.1.4 Risk, vulnerability, hazard and exposure concepts

Before delving deeper onto the study of objective and methodology in the manuscript, it is worthwhile distinguishing different terminologies frequently encountered in impact assessment of climate change.

1.1.4.1 Risk

Risk is the possibility/potential for adverse consequences where something of value is threatened and the occurrence and degree of outcome is not certain. Risk results from the interaction of vulnerability, its exposure over time as well as to the hazard and likelihood of its occurrence (IPCC 2018), illustrated in Figure 1-9.

Climate risk is the possibility of specific climate-related impacts (climate impacts) that can affect assets, people, ecosystems, culture, etc. Examples of risks include risk of water scarcity for smallholder farmers (water scarcity as a potential consequence of climate change), the risk of food insecurity in the rural population; the risk of extinction of biodiversity species; risk of damage to transport infrastructure because of erosion, landslides, etc. In short, risk is the possibility of consequences in which the result is uncertain.

Risk assessment can address this uncertainty in a variety of ways. In disaster risk assessment, one of the approaches is probabilistic assessment, in which risk is presented as the likelihood of occurrence of hazardous event multiplied by the impact of it.

In the context of climate change, this probabilistic approach is often not applicable because most of the hazards and consequences associated with climate change, cannot be described as standard events, which is one of the requirements of the probabilistic approach. In addition, the consequences of climate change alone cannot be assessed using a probabilistic approach, as future pathways of socio-economic development, levels of greenhouse gas emissions and climate impacts remain uncertain. Instead of that, scenario approach is proposed by IPCC (IPCC 2018). For example, different climatic consequences for different scenarios of greenhouse gas emissions; different scenarios of vulnerability under different paths of socio-economic development.

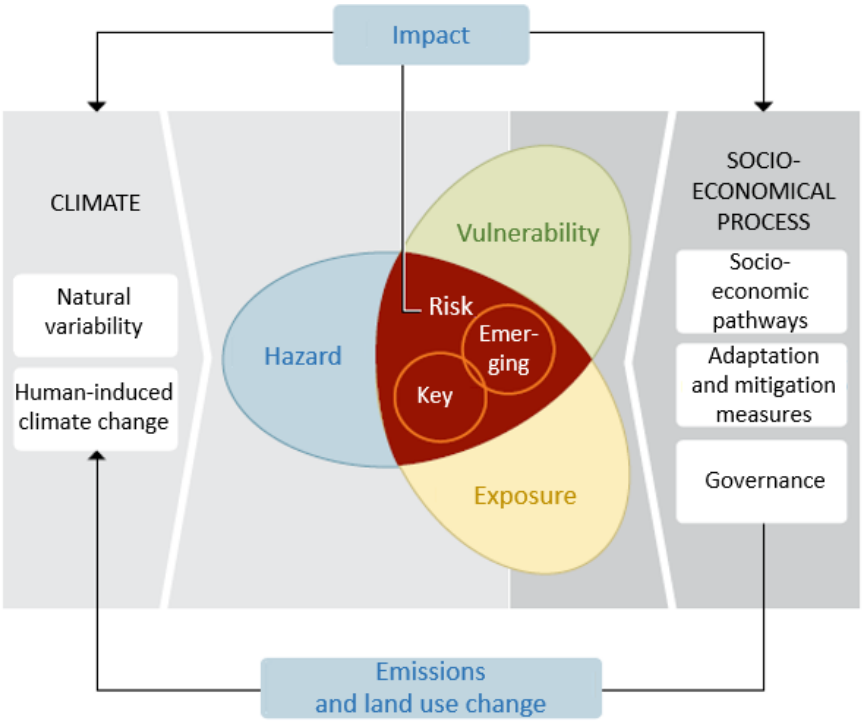


Figure 1-9 : Contributing factors of risk (Adapted from IPCC 2014 report)

1.1.4.2 Danger "hazard"

The possible occurrence of a natural or human-induced physical phenomenon or trend or physical impact that can cause death, injury and other health consequences, material and property losses, as well as damage to infrastructure, livelihoods, supply systems services, ecosystems and ecological resources (IPCC 2018). In this manuscript, we will use the term "hazard" and "danger" interchangeably to refer to climate-related physical phenomena, trends or their physical effects.

The hazard can be a phenomenon (for example, heavy rain), or direct physical impact (such as a climate induced heatwave). It does not have to be extreme weather; slowly ongoing phenomena can also be dangerous. It is also important to take into account the likelihood of a particular hazard by setting thresholds to help determine the frequency of it (for instance,

number of consecutive hours indoor temperature in a room is above 27 degrees). In here, 27 is a threshold and number of consecutive hours is the frequency of that event (hazard) taking place.

1.1.4.3 Exposure

Presence of people, livelihoods, species or ecosystems, ecological functions, services and resources, infrastructures or economic, social and cultural assets in places and conditions that could be adversely affected (IPCC 2018).

"Exposure" refers to the relevant elements of the socio-ecological system (e.g. people, livelihoods, as well as species, ecosystems, etc.) that may be adversely affected by hazards. Exposure can be expressed as absolute values, densities, percentages, etc. (e.g., population density in an area affected by a heatwave; percentage of wetlands in an area affected by pollution, etc.).

Changes in exposure over time (e.g., changes in the number of people living in drought prone areas) can significantly increase or decrease the risk.

1.1.4.4 Vulnerability

It is referred to propensity or predisposition to adverse effects. The term vulnerability covers a variety of concepts and elements, including sensitivity or susceptibility to harm and lack of ability to cope and adapt (IPCC 2018).

Vulnerability refers to those characteristics of socio-ecological system's elements that can increase or decrease the potential impacts of a specific climatic hazard. It includes two relevant concepts: sensitivity and adaptive capacity.

Sensitivity is determined by those factors that directly affect the consequences of the hazard. Sensitivity can include the ecological or physical characteristics of the system (e.g. soil type in agricultural fields, water retention capacity, building material for residential buildings), as well as social, economic and cultural characteristics (e.g. age, income).

Adaptive capacity in the context of climate risk assessment refers to the ability of communities to cope with current or future climate impacts. This does not mean the ability of ecosystems to respond to impacts, but rather the ability of society to manage ecosystems. Adaptive capacity has two key components: first, the ability to ***overcome*** problems (the ability of the population, institutions, organizations and systems to deal with the problem of unfavourable conditions, cope with them and overcome them in the short and medium term, using available professional skills, material values, beliefs, resources and opportunities. For example, existing early warning systems for a heatwave. Second, the ability of systems, institutions, people and other organisms to ***adapt*** to the potential hazard, seize opportunities, or respond to consequences. For example, the potential for introducing new agricultural practices in cities. Lack of this ability can significantly increase the vulnerability of the system and by extension, risk.

1.1.4.5 Impacts (Consequences, Outcomes)

Effects on natural and anthropogenic systems. In this manuscript, the term "impact" is used primarily to refer to the effects of extreme weather and climate events and climate change on

natural and human systems. Impacts generally refer to effects that affect people's lives, livelihoods and health, ecosystems, economy, society, culture, services and infrastructure due to the interaction of climate change or climate hazards occurring over time and the vulnerability of the affected society or system (IPCC 2018).

“Impact” is the general term for describing consequences, ranging from direct physical impacts of a hazard to indirect consequences on society, also called social impacts. Impacts are the basic building blocks of chains of causation.

1.1.4.6 Climate extreme (extreme weather event)

A value of a meteorological or climatic variable that is above (or below) a threshold value towards the higher (or lower) ends of the variable's range of observed historical values.

1.1.4.7 Mitigation

Climate mitigation measures in general consist of a set of activities that are aimed to limit the rate of greenhouse gas emissions into the atmosphere, by better controlling energy usage (energy efficiency), by substituting fossil fuels with renewable energies and by storing carbon (carbon capture). In other words, mitigation consists of putting in place sustainable development programs. In urban context, mitigation means taking steps/actions to reduce air pollution, and greenhouse gas emission rates from urban areas. Currently the most popular mitigation strategies are focused on improvement of technologies, and switching fuels. (Oke et al. 2017) argues that a more efficient urban form, transport and land-use mix also have a notable potential to moderate city contributions.

1.1.4.8 Adaptation

Adaptation to global climate change and global warming is the adaptability of natural or anthropogenic systems in response to real or expected climatic changes aimed at reducing vulnerability or/and use of favourable conditions.

The word adaptation itself evokes an ability of a functioning society to adjust. Adaptation strategies complement mitigation measures, which aim to emit less greenhouse gases and restore or protect the carbon sink capacities of ecosystems. Even if all emissions of carbon cease today, the climate inertia generated by heavy anthropogenic activities so far will continue to cause climate’s disturbance in the future. This indicates that adaptation is an essential strategy to reduce vulnerability and improve resilience.

Adaptation can be individual (changes in individual behaviour) and collective (changes in communities, companies, governments, etc.).

It is important to emphasize that the more we manage to mitigate, the less we have to adapt and vice versa.

In the context of this manuscript, adaption consists of making systems or territories less vulnerable to climate change, through actions reducing the effective impacts of climate change, or improving the response capacities of societies and the environment. For instance, installation of a heatwave alarm system and diffusing information and guidelines to people on how to

protect themselves during one. Other examples of adaption to higher temperature in the cities are, installation of misting system or planting more trees in the city to reduce temperature.

Adaption must also be seen as permanent strategies and be implemented over a long period.

As the ecosystem degrades, it could exacerbate the possible social, economic or geopolitical crises to a degree that sometimes becomes too costly to maintain the same level of services, which will consequently increase vulnerability.

In general, adaptive capacity and vulnerability are inversely proportional, a low vulnerability score implies a good adaptive capacity and vice versa. Therefore, as mentioned in section 1.1.4.4, vulnerability is both the result of exposure rate to natural hazard and its adaptive capacity.

In France, The Economic Council for Sustainable Development report has defined adaptation as "*All the changes in organization, location and techniques that societies will have to make to limit the negative impacts of climate change and maximize its beneficial effects* (Hallegatte, Lecocq, and de Perthuis 2010)."

An important part of adaption is the issue of anticipating the effects of climate change (intensity/duration) on environment and by extension on the economy, society, health and life. Because it is easier for communities to adapt to an anticipated change than to an unexpected one.

1.1.5 Integrated solutions to urban climate

Increased population density and economic activities on cities over the last few decades, combined with the challenges that global warming will undoubtedly bring, has prompted a noticeable push by civil society and governments at different levels to develop activities, innovative initiatives, and transformative actions to assist cities in dealing with the effects of climate change. Most of these initiatives have proved insufficient so far, and there is a clear need to **assist decision-makers** to think strategically on how to layer adaption interventions in the cities in a way that could lead to better resilience for different potential future scenarios (Lin et al. 2021).

Indeed, diverse adaptation (technological, nature-based, and societal) and mitigation (cool roofs, building greenery, retro-reflective materials, urban green infrastructure) measures need to be implemented to bring about the transformation required to build resilience to overheating and its impacts in the cities (Costanzo, Evola, and Marletta 2021).

Naturally, the first step in implementation of adaptation or/and mitigation solutions is to identify the most vulnerable points/regions to urban overheating to maximize the benefit on the city and people living in the cities. The term "vulnerable" was used here, because the motivation is to not only monitor and characterize exposure rate of urban environment but also to know when and where it leads negative impact on human life.

Limiting the vulnerability of people to climate change, extreme weather events, and UHI requires us to take collective actions at different scales to build resilience for multiple potential future scenarios: global scale, urban scale, and building scale.

It is encouraging to highlight that urban planning has the potential to play a vital role in the development and implementation of collective actions in urban systems. Main advantages of urban planning are the universality of the profession and the instruments available in it. Tools such as plan-making, stakeholders involvement, design standards, and development management are crucial to deliver urban adaptation/mitigation solutions at different scales (Shalaby and Aboelnaga 2017). Furthermore, cities often have stronger management mechanisms that allows all involved stakeholders to devise and apply policies and collective actions more effectively than those at national or super-national scales.

1.1.6 Decision making in urban planning

Decision-making in general is a cognitive process where a practitioner/decision-maker selects a type of action among different alternatives. This process is theoretically based on specific set of criteria, and on analysis of options and data obtained from various sources.

In urban planning, **local urban plans**, which are drawn by local authorities based on their expertise considering a wide range of criteria (social, economic, environmental), establish guidelines for spatial and physical organization of municipality and overall vision on its territory's development. They contain recommendations and town planning policies adopted by the city council/local authorities, which serve as guides in the **decision-making** process for the present and future development choices.

In other words**, local urban plans** are the principle decision-support tools aiming to ensure consistency between intervention choices in various sectoral issues (e.g. housing, commerce, transport, environmental protection, recreation, public facilities, energy) considering development potential and constraints of natural and build environment as well as expectations and concerns of citizens.

This decision-making support tool is an important element for a better management of the municipal territory. This is why we must ensure that the plan is focused on the implementation of practical solutions that takes into account the financial reality, local management resources and more importantly, in this period, potential climate variation scenarios.

In the French context, "Plan Local d'Urbanisme" (PLU) is the equivalent expression referring to a set of various documents aimed at ensuring the proper urban development of cities. For France, this document has replaced the old Land Use Plans (POS) and National Urban Planning Regulations (RNU) documents.

PLUs normally contain the following documents (le Plan Local d'Urbanisme (P.L.U) 2020):

- An introductory report detailing choices made to moderate the use of space and minimize urban sprawling based on a territorial diagnosis.
- A sustainable planning and development project (PADD) which sets out the urban planning project and defines:
    - General trends and orientations of urban planning policy, town planning, landscape, protection of natural, agricultural and forest areas, and preservation or restoration of ecological landscape;

    - General guidelines concerning housing, transport and travel, energy networks, development of digital communications, commercial equipment, economic development, and leisure activities.
- Orientations for planning and programming (OAP) guidelines, which, in compliance with the PADD, include recommendations relating to urban development, housing, transport, and changes of areas that are to be urbanized.
- Urban regulations (graphical and written) which, in compliance with PADD and OAP, delimits agricultural areas, natural and forest areas, urban areas, areas to be urbanized, and sets general rules of urbanization.
- Various annexes (accessibility to public utilities, list of subdivisions, water and sanitation networks, noise exposure plans for airports, protected areas, concrete covered zones, etc.).
  Annexes also include risk prevention plans of municipalities. These plans regulate the land use in a municipality stipulating the risk it may be subject to, such as risks associated with flooding, landslides, wildfires, avalanches, storms, marine submersions, earthquakes, volcanic eruptions, risks of ground collapse due to underground structures (catacombs, underground quarries), etc.
- Complementary related studies.

The consideration of climate and energy issues is integrated into the OAP part of PLUs, which mostly give recommendations. However, the impact of these recommendations is not assessed, either because of the absence of a standard workflow to do it or because of large scales that imply a huge work to check each application. Assessing the impact of these recommendations would require the implementation of a rapid assessment tool that would help decision makers to take into account climate change and energy issues at city scale efficiently.

In addition, vulnerability of urban dwellers to heatwaves and urban heat island effect considering future climate change scenarios are not given adequate attention in the risk prevention plans section of PLUs. Consideration of these phenomena warrants the need for creation/addition of vulnerability maps to extreme temperature under current and future climate scenarios.

Another similar instrument frequently used in urban planning is cadastral maps. An equivalent term also used interchangeably with cadastral maps is land registry. In France, the term “cadaster” is seen more in the literature and it referred to a plan covering entire French territory, delimiting the geographical limits of land holdings in the form of plots (parcels).

Cadastral maps are mostly managed and administered by the state (ministry of finance and public accounts) and are used as a base for determining certain taxes, particularly property taxes.

Cadastral maps have not been used extensively in urban climate engineering so far for two main reasons: inaccessibility of cadastral maps to public, and unavailability of urban climate data (land coverage details) at cadastral scale.

In this manuscript, an alternative approach is proposed that allows practitioners in urban climate engineering perform urban climate analysis to create heat exposure/vulnerability maps at cadastral scale relying only on publicly available (open source) data, thereby reducing the probability of exposing private and confidential data.

1.1.7 Vulnerability and Exposure Maps

Publicly accounting of climate hazard losses and understanding their economic, social, and environmental implications requires practitioners to systematically evaluate, record, and share, as appropriate, exposure and vulnerability information.

Distribution and dispersion of exposure and vulnerability information in the form of vulnerability and risk maps, freely available and accessible, are considered one of the best approaches that could lead to the reduction in climate and weather-related hazard losses.

Climate hazard on maps could be represented by past, present and future climate variabilities, extremes, and in some cases the hazard could also be a function of climate extremes in combination with other factors such as urban heat island that increases the intensity of heatwaves or land use change that raises the susceptibility to, landslides, flooding, crop failures and draught.

Today, maps that show synthetized climate data have become part of a standard package of tools to communicate climate-related risks and hazards. Vulnerability maps are frequently utilized to direct the attention of users to the geographic locations where impacts on communities are expected to be the highest and therefore an adaptive intervention is required (de Sherbinin et al. 2019).

Vulnerability maps could be used by practitioners for planning (prioritization and targeting) of adaptation interventions, in understanding underlying factors of vulnerability, in planning of emergency response actions, and in communication of risks. These maps are made across a range of scales, territories, climate hazards, and for various thematic focus.

(de Sherbinin et al. 2019) in an interdisciplinary research project, performed a comprehensive review on mapping strategies of social vulnerabilities due to climate change. Their main finding in the review was that most maps in this context are academic in nature, and therefore not geared toward decision/policy makers. They also state that some maps that claim relevance to the policies also frequently fall short of best practice. They argue that vaguely defined maps (lack specificity on the target attribute or climate hazard focus) showing “vulnerable population” are unlikely to lead to any changes in policies or implementation responses.

They suggest vulnerability maps should not be an end goal but rather an entry point for discussions by policy makers. To ensure maps meet such criteria, they suggest mappers to consider a number of issues. These include, increasing end-user collaboration (taking into account the opinion of decision/policy makers and other stakeholders before building a map); pay greater attention to map communication beyond mapping as a final product; validate the data in the map if possible; and evaluate the value of information presented in the map.

As discussed earlier, social vulnerability to climate change is a function of (1) exposure rate to temperature, precipitation changes, and increase in the intensity and frequency of extreme weather events; and (2) sensitivity and adaptive capacity of population to these changes.

All factors influencing vulnerability, e.g. exposure rate to extreme weather events, land cover and usage, population density, socio-economic status of people, and effectiveness of public/private institutions in charge, are dynamic and can vary spatially and temporally. This means that the relative contribution of each factor to vulnerability is different from one place to another and from one time to another.

In this manuscript where the focus is on vulnerability of people to over-temperature, we refer to "vulnerability map" when both the exposure rate and all factors influencing adaptive capacity/sensitivity of target social group to overheating is taken into consideration. If factors affecting adaptive capacity/sensitivity of population are assumed static or absent from the map then we refer to it as an "over-temperature exposure map".

1.2 **Problem Statements**

The following statements summarize the problems and issues discussed in previous sections and will be dealt in the subsequent chapters to various extents:

(1) Proven global climate change, urban heat island, urban microclimate and heatwaves have already brought significant shifts in the pattern of human activity and will continue doing so in the future. Movement of climate zones is one of the impacts of these variations. Some regions may gain from these shifts and some may lose. However, one thing is clear; we all have to adapt to these inevitable changes. This means there is a need to map out indoor vulnerable regions in PLUs and layer adaptive and mitigation interventions accordingly.

(2) Typical weather data used by practitioners in thermal evaluation of buildings that are generated from historical records collected from rural weather stations do not contain future climate change scenarios and UHI effect for urban areas, while they are critical for better description of vulnerability to both indoor and outdoor over-temperature.

(3) There is a lack of standard definition for overheating vulnerability in indoor environment of buildings. Indoor overheating and thermal comfort are spatially and temporally variable. This requires practitioners/researcher to use multiple indices and dimensions of study to address relevant questions.

(4) There is a lack of knowledge on buildings sensitivity to over-temperature due global warming or heatwaves.

(5) No process is available for the integration of climate change data, and UHI effect in thermal simulation of buildings at city scale.

1.3 Objective

For business, industry, legislative and government agencies, regardless of the sphere of their activities, it is the same problem: the need to take into account climate change in current and future decisions. Some sectors are expecting to benefit from climate change, but many will have to deal with increased vulnerability induced by it.

Building sector, in particular is going to be significantly affected by these variations, because they are where majority of people spend most of their time. This gives buildings and their surrounding the potential to put occupants in danger or protect them during a major external climate event, depending on how they are built and operated.

Reducing vulnerability and adaptation to future changes require an understanding of the level, timing and potential impact of climatic risks. In the last decade, access to reliable climatic data and forecasts has noticeably expanded. Thanks to this and efforts made by scientists across the world to adapt information to the needs of various sectors, it is now possible to make more efficient climate data-informed decisions at various spatial and temporal scales.

Considering the stated challenges and opportunities, this thesis positions itself, within a broader context, to develop a methodology for integration of climate change and UHI data into decision support tools in urban planning to evaluate indoor overheating risk in residential buildings at city scale.

In other words, the objective of this thesis is to pave the way for the development of a comprehensive methodology for practitioners and policy makers in Urban Planning to take into account climate change scenarios, urban climate, energy transition and health in urban development policies.

For policy makers this methodology will be used in assessment of local urban plans (PLUs) and climate change in terms of UHI and vulnerability of population to heatwaves. This objective could be achieved with creation of indoor heat exposure vulnerability maps that could be used as an input for local adaptation solutions such as reintegration of nature to the city, installation of misting systems or for mitigation measures such as buildings' refurbishment programs.

This means the output of this thesis will be in the form of an indoor overheating vulnerability map or/and in the form a rapid assessment tool (web-based application running on an API generated using this methodology). In case the data on adaptive capacity and sensitivity of occupants to overheating are not available, then the map would be called an indoor overheating exposure map (Figure 1-10).

1.4 General Methodology

Creation of indoor overheating exposure map of buildings at city scale, in the context of climate change heatwaves, and UHI requires consideration of at least the following crucial aspects:

- Appropriate method to select reference buildings and perform energy and thermal comfort assessment simulations on the reference buildings.

- Good quality data that represent climate change scenarios, heatwaves and urban heat island.

- Development of an approach to extend the application of given climate scenarios on reference buildings to the rest of buildings stock in the city.

Having the given aspects in mind, we have organized the thesis work steps as well as individual sections of this manuscript. An outline of the general workflow in generation of vulnerability/exposure map of indoor thermal conditions in residential buildings is presented in Figure 1-10.

The methodology here was derived to enable decision-makers rapidly assess indoor thermal conditions of different neighbourhoods in the city. The complexity of questions tackled in this research project required involvement and interaction of many disciplines: meteorology, building thermal and energy studies, urban planning, statistics, data science, geography, geospatial mapping/modelling. For this reason, at the start of this thesis it was decided to carry out this research on a process approach. In this approach the main objective is at the centre and it is leading for all decisions in the research (Tobi and Kampen 2018).

Success of this approach relies on strict follow up of an agenda/a conceptual framework throughout the research period. In this research project, every step of the way first the “why” and “what” of the step are answered and then the “how” of the research work. That is why, data collection, analysis and presentation may considerably be different in each chapter of this manuscript.

Nevertheless, to maintain the consistency, similar to overall process approach, we built a data analysis plan before starting to achieve certain tasks. That way, we knew that the output of data analysis files are exactly the kind of data that can be used in other steps.

Following up this approach allowed us to build and formalize the different sections of this research work in the integration stage.

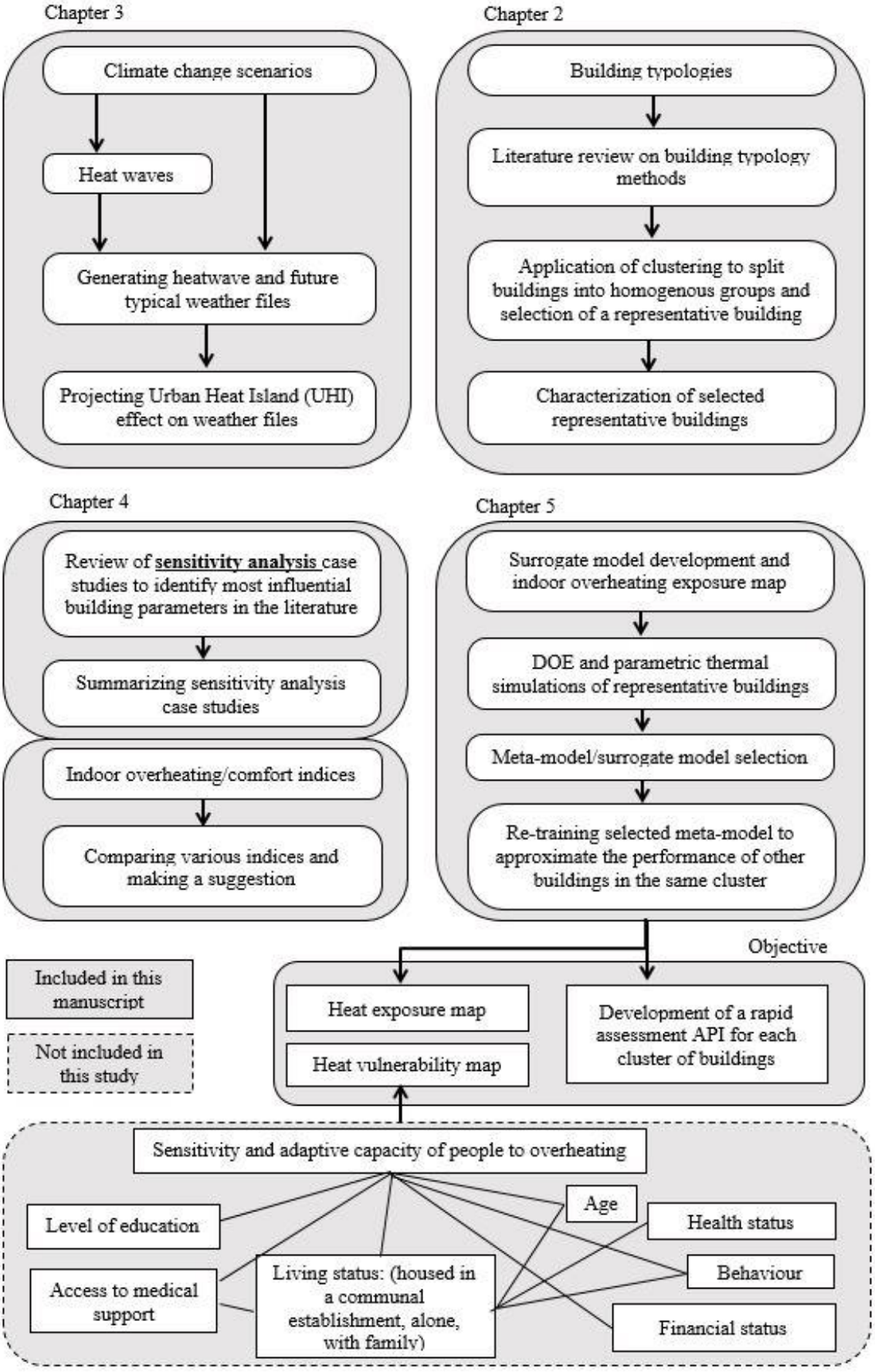


Figure 1-10 : Proposed general process in creation of overheating vulnerability map

The diagram of Figure 1-10 demonstrates the simulation-based workflow in this thesis that attempts to address all three crucial aspects mentioned in the first paragraph of general methodology.

The four boxes of the workflow demonstrate the main four chapters of this manuscript each describing a major step in preparation of indoor over-heating map of residential buildings.

A major challenge in indoor overheating assessment at city scale is the efficient representation of urban build stock in such a way that takes into consideration UHI effect as well as the buildings' characteristics relevant to the problem. Chapter 2 of this manuscript is dedicated to dealing with this challenge using the existing literature and modern data-driven tools. It starts with a critical literature review on buildings typology construction methods, it then choses cluster analysis machine learning technique to aggregate residential buildings into clusters of buildings with similar characteristics, and then identifies one representative building from each cluster, which further undergoes a characterization step.

Chapter 3 of the manuscript is devoted to the description of a workflow that could be used by building practitioners and researcher to access and use data from multiple climate models for building performance evaluation studies taking into consideration UHI effect.

Another challenge in the preparation of indoor overheating map pertains to measurement indices and identification of key building parameters influencing thermal comfort and indoor overheating in summer. Chapter 4 of the manuscript attempts to deal with this challenge by a review of literature on sensitivity analysis studies of buildings' thermal performances, and through a detailed comparative analysis of indoor overheating measurement indices.

The third major aspect of this research is the application of simulation results on identified representative buildings to the rest of buildings through an extrapolative process. As the objective of this research is to demonstrate the methodology that could be used as support in the strategic decision for climate change policy intervention, it is necessary to make sure it is efficient and can rapidly provide an initial approximation of building performance or buildings' performances, it was decided to use surrogate modelling as a way to extend the application of reference buildings to the rest of buildings. Chapter 5 of this manuscript is devoted to the description of input data, methods and techniques for surrogate model development.

In summary, the manuscript systematically illustrates the following issues:

1- It discusses why and how to identify reference buildings?
2- It describes why and how to access and use open source future and historical weather data for building performance simulations (BPS)?
3- It also presents which building parameters influence indoor thermal comfort and which indices could be used to describe summer performance of residential buildings in temperate climate regions.
4- And the last chapter demonstrates how meta-models can be used to extend the outputs of reference building models to the rest of build stock

1.5 Summary

The objective of this research is more than just to provide a scientific response in a specific discipline but the transversal study of a methodology allowing various issues to integrate and pave the way for creation of something that would eventually be used to help policymaking and lead to subsequent implementation response. Having this in mind, the organization of first and subsequent chapters follows a process approach. In the first chapter, it starts with an introduction to the main triggers that influence overheating in buildings, and then it proceeds to the explanation of various terminologies frequently encountered in impact assessment.

Following are the summary of key messages presented in this chapter I:

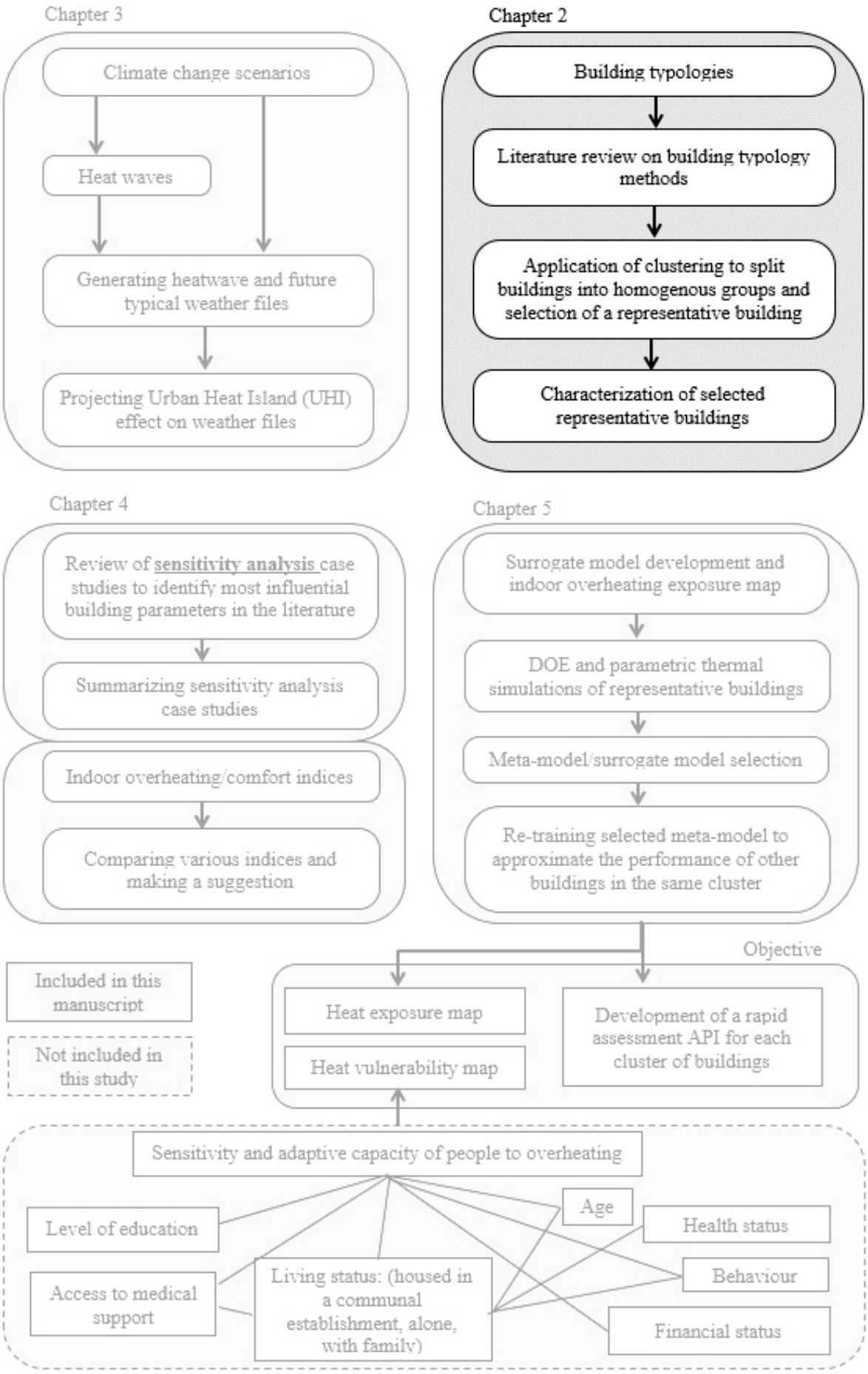
- Average global temperature was 1.2 ° C higher than the pre-industrial (1850-1900) level in 2020, and this global increase is accompanied by extreme events that further put pressure on urban infrastructure and ecosystem.

- The global warming may enhance the negative impacts of UHI effect, but it does not systematically affect its magnitude, because UHI is the difference between rural and urban temperatures and the net global temperature increase influences rural and urban areas of different places differently. Regardless of whether global warming affects UHI effect's magnitude or not; there is still a need for local adaptation and mitigation strategies to offset the impacts of over-temperature in populated areas.

- Limiting the vulnerability of people to climate change, extreme weather events, and UHI requires us to take collective actions at different scales to build resilience for multiple potential future scenarios: global scale, urban scale, and building scale.

- Urban planning, thanks to the instruments available, has the potential to play an important role in implementation of collective actions aimed at reducing the impact of over-temperature.

- Reducing vulnerability to overheating requires an understanding of the level, timing and potential effects of climate risks. Over the last decade, access to reliable climatic data has notably expanded. Thanks to this and efforts made by scientists across the world to adapt information to the needs of various sectors, it is now possible to make more efficient climate data-informed decisions at various spatial and temporal scales.


Chapter 3
Climate change scenarios
Heat waves
Generating heatwave and future typical weather files
Projecting Urban Heat Island (UHI) effect on weather files
Chapter 2
Building typologies
Literature review on building typology methods
Application of clustering to split buildings into homogenous groups and selection of a representative building
Characterization of selected representative buildings
Chapter 4
Review of sensitivity analysis case studies to identify most influential building parameters in the literature
Summarizing sensitivity analysis case studies
Indoor overheating/comfort indices
Comparing various indices and making a suggestion
Chapter 5
Surrogate model development and indoor overheating exposure map
DOE and parametric thermal simulations of representative buildings
Meta-model/surrogate model selection
Re-training selected meta-model to approximate the performance of other buildings in the same cluster
Objective
Heat exposure map
Heat vulnerability map
Development of a rapid assessment API for each cluster of buildings
Included in this manuscript
Not included in this study
Sensitivity and adaptive capacity of people to overheating
Level of education
Access to medical support
Living status: (housed in a communal establishment, alone, with family)
Age
Health status
Behaviour
Financial status

Chapter 2 : Building typologies

2.1 Background/literature review

An accurate analysis of building stock during extreme weather events requires simulation and analysis of large number of buildings taking into account geometrical attributes, thermo-physical properties and occupants' behaviour. However, in practice, simulations, analyses and calculations of large number of buildings at each individual building is not possible due to large volume of input data for each building, and time limitations (Ali et al,2019.).

A viable solution in such case is to develop a set of archetypal building types that could represent most of the building stock and perform necessary scenario analysis or simulations on them.

To this end, many researchers and authors have proposed methodologies for aggregation of building stock into types, categories, clusters, and archetypes based on multiple criteria for various objectives. For instance, assessing the impact of various potential retrofit interventions in building sector (Jorgji et al. 2019), urban energy simulation (Cerezo et al.,2018), predicting building energy consumption for heating and cooling demand (Korolija et al. 2013). As a result, the literature sources that explore building stock types vary by a wide range of methods and techniques.

The terms, build stock archetypes, types, samples, prototypes, reference buildings, exemplary buildings etc. are used interchangeably in the literature referring to a category or sample of buildings representing the larger built stock. This process itself called typification of building stock and/or aggregation of build stock to reference buildings.

In this text, build stock aggregation means reducing the number of buildings to be modelled by representing buildings with just one building model. This could mean the build stock is entirely represented with one single building or by a set of buildings each representing a category.

According to Tabula definition, the term "building typology" is referred to a systematic description of the criteria for the definition of typical buildings (archetypes) as well as to a set of exemplary buildings representing the building types (Tabula team, 2012).

To cover most of the build stock aggregation methods, this work focused on both conventional classification methods and data-mining approaches in building stock classification. Build stock aggregation methods rely on different levels of input information granularity, various calculation techniques, scale of building stock, and application objective of archetype.

Development of building stock typology relies on input data on the bases of which to categorize and model each group of reference buildings. The level of detail of the available information can vary significantly, resulting in the selection of different aggregation techniques. Each technique has its own strengths, weaknesses, capacity, and scope of application.

Depending on the chosen aggregation technique, input attributes necessary to develop building stock typology in relation to energy consumption and thermal comfort can include: general information and boundary conditions of building (year of construction, climate zone, density of urban area, etc.), urban morphology (detached, continues, high rise, etc.), engineering systems

(HVAC, heating system's efficiency, etc.), occupancy type (residential, educational, etc.), thermo-physical as well as other properties of envelope and systems (U values of walls, roof, openings, infiltration rate, etc.), building geometry (form, compactness, height, window wall ratio (WWR), orientation, number of levels, etc.), operational parameters (temperature set-point, frequency of use of appliances, etc.), and meteorological data (temperature, humidity, etc.). (Ali et al. 2019; Cerezo et al. 2017; Coma et al. 2019; Dascalaki et al. 2011; Davila, Reinhart, and Bemis n.d.; Famuyibo, Duffy, and Strachan 2012; Ghiassi and Mahdavi 2017; Guillaumet et al. n.d.; Hamdy et al. 2017b; Heine Kristensen 2018; Hidalgo et al. 2019; Korolija et al. 2013; Mata, Sasic Kalagasidis, and Johnsson 2014; Monteiro et al. 2015; Pasichnyi, Wallin, and Kordas 2019; Schoetter et al. 2019; Stewart and Oke 2012b; TABULA Project Team 2012; Tornay et al. 2017),

Depending on objective and availability of information, the granularity level of input data can vary substantially as well. For instance, residential buildings in Tabula classification are subdivided into four groups of single-family house, terraced houses, multiple family houses, and apartment houses (TABULA Project Team 2012). Hamdy and al, 2017 have gone further by dividing apartment houses into six flat typologies in relation to their location in the building (corner/middle and ground/middle/top floor)(Hamdy et al. 2017b).

The available data about building stock also varies from country to country. For instance, Geographic Information System (GIS) data are not publicly available in many countries including China (X. Li et al. 2018). Sometimes regions within a country can also have significant variance in data availability on build stock specifications. E.g., there are more data available about Ile-de-France as compared to other regions in France (Tornay et al. 2017). (Ghiassi and Mahdavi 2017) used official and crowd-sourced GIS database, statistical information and building performance standard data of Vienna city to develop the build stock archetypes. GIS has also been used by many other authors as a platform to collect and organize urban-scale building information as well as a tool to visualize and process the main data (Ali et al., 2019; Davila et al.; Ghiassi and Mahdavi, 2017; Österbring et al., 2016; Sokol et al., 2017)

Data from building energy performance certification (EPC) has also been used by authors as an input for classification of buildings into archetypes (Ali et al. 2019; Famuyibo, Duffy, and Strachan 2012) or for characterization of archetypes after the segmentation (Sokol, Cerezo Davila, and Reinhart 2017; Tornay et al. 2017).

At European scale, (Coma et al. 2019) have performed a comprehensive study about European projects concerning the residential building stock characterization information; namely, projects within the Intelligent Energy Europe Program (IEE) (TABULA, EPISCOPE), the new EU building stock Observatory, and Project Policies to Enforce the Transition to Nearly ZERO Energy Buildings in the EU (ENTRANZE)(Coma et al. 2019).

In the United states, (Moura, Smith, and Belzer 2015) have developed a long-term residential floor space time-series spanning 120 years from 1891 to 2010. The authors have collected data from the U.S. census Bureau, U.S. Department of Housing and Urban Development, and National Survey data (Moura, Smith, and Belzer 2015). In official report from U.S. Department of Energy (DOE) on Commercial Reference Building Models of the National stock, the data

from Commercial Buildings Energy Consumption Survey (CBECS) was used to develop 16 building types in 16 locations across U.S.(Deru et al. 2011).

In French territory, the French Geographical Institute 'Institut national de l'information géographique et forestière' (IGN) has compiled a 3D shapefile dataset (the IGN-BDTOPO) containing information about the footprint, use, height, and date of construction of buildings in France[1]. CSTB (*Le Centre Scientifique et Technique du Bâtiment*), and its partners have initialized a first version of the BDTOPO as part of the GO-RÉNOVE project. The services of the GO-RÉNOVE project are the first operational applications of the IGN-BDTOPO, on the theme of the renovation of the existing housing stock. French Institute on Economics and Statistics "Institut national de la statistique et des études économiques" (INSEE)[2] has gathered construction information of buildings and year of construction. Bonhomme and Tornay, 2017, have enriched the building stock database information by adding the most prevailing construction material for their reference buildings as a function of location in France (Tornay et al. 2017).

2.1.1 Building stock typologies generation techniques

Building stock aggregation methods can be broadly categorized into deterministic and data-mining/data-driven approaches (Guillaumet et al., 2018). The terminology is with reference to the role of automatic classifiers and a priori assumptions. In a deterministic approach, buildings are classified according to the existing bibliography and expert opinions. The aim of this approach is usually to develop easily interpretable results with limited available information about the entire building stock.

Data-mining approaches on the other hand are strongly dependent on the data availability and development of information technology that improved the possibilities for information gathering, management and documentation. Such a classification is therefore greatly based on computer programming. Grouping of methods for the development of building stock typologies are shown in Figure 2-1 and are discussed in details in the following sub-headings.

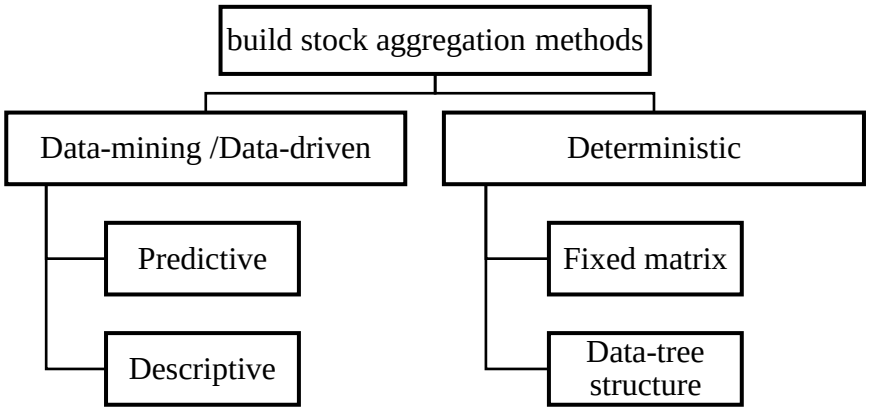


Figure 2-1 : Methods of build stock typology development

[1] http://professionnels.ign.fr/bdtopo#tab-1
[2] https://www.insee.fr/en/accueil

2.1.1.1 Deterministic approaches

In mathematics and computer sciences, deterministic is referred to a system in which no randomness is involved. This segmentation in this approach is highly based on *a priori* assumptions. In the context of build stock typology, we use it to address a classification approach in which the modeler decides the segmentation criteria primarily relying on their experience, expertise, literature, or for a specific objective. Literature review showed that most observed variables/segmentation criteria in deterministic approach are construction year, building use and in the case of residential buildings single or multiple family houses.

Typological studies that employ the same approach for categorization may use various physical or other predetermined characteristics to categorize the buildings. In general, deterministic build stock aggregation technique involves three steps:

- Classification or segmentation based on initial predetermined characteristics.
- Parametrization or characterization of the archetypes with supplementary details.
- Simulation or modelling of each archetype (Monteiro et al. 2017)

The greatest strength of deterministic approach is that additional input variable-sets can be added to the classification without compromising the typification process. For example, if residential single-family buildings are classified by date of construction into seven archetypes another archetype could be added to the list without any necessity to change the previous archetypes. Another strength is that the archetypes developed in this approach are easily interpretable, although this quality is not exclusive to this approach.

Lack of explicitly stated arguments, or reasons in support of the initial segmentation attributes of building stock classification aimed to model indoor thermal comfort or energy consumption also known as predetermined characteristics is a major limitation of this approach. For instance, categorizing building stock by year of construction even though many of them have been retrofitted over the years. Uncertainties associated with underlying aggregation assumptions about buildings can substantially increase the risks of miscalculation or error in the model. For instance, assuming that all buildings built before 1973 oil shock have lower energy performance without considering other energy related attributes, for instance, thermo-physical properties of materials used in construction or renovations performed on the buildings over the years.

Another downside of this approach is that as the number of segmentation criteria increases, the resulting archetype groups become smaller and numerous. Two types of deterministic techniques are seen more frequently in the literature: fixed matrix and data-tree format.

2.1.1.1.1 Fixed matrix

In this research effort, fixed matrix is referred to a firm arrangement of building types (real or virtual) in rows and columns in a grid or table. National building typology (TABULA) elaborated in the framework of Intelligent Energy Europe (IEE) program is a clear example of this approach in which the columns represent different sizes of residential buildings and rows different construction periods. This building typology represents building types with real exemplary buildings (Cerezo et al. 2017; Monteiro et al. 2017). TABULA building type matrix has led to numerous other European research projects and studies. An example is the

comparative study conducted by (Loga, Stein, and Diefenbach 2016) of 20 European countries residential building stocks. TABULA has also been used as a base for various energy efficiency and refurbishment assessment studies over the years (Ballarini et al. 2017; Coma et al. 2019; Dascalaki et al. 2011; Droutsa et al. 2014; Kragh and Wittchen 2014; Loga, Stein, and Diefenbach 2016; TABULA Project Team 2012). The number of archetypes in this approach equals to the product of rows and columns of the matrix.

Another example of this classification method is Local Climate Zones (LCZ). Objective of LCZ although is drastically different from Tabula. In Tabula, the focus is on building itself, but in LCZ the main objective of classification is to categorize urban areas based on their microclimatic characteristics.

(Oke et al. 2017) has used LCZ to categorize urban landscape types. LCZ is based on the assumption that, four urban climate controls (fabric, land-cover, structure, and metabolism) cluster together in the cities. For instance, core city center parts of urban areas are mostly covered with tall buildings or paved with high-density impervious materials that are often dry and have a high heat storage capacity. There is also a higher concentration of human activity that generate heat, possibly moisture, and air pollutants from air conditioners and vehicles. In the other end of spectrum are low-density houses with relatively light construction, surrounded by a greater percentage of vegetation, open space, and perhaps with less emission of heat, moisture and pollutants. Based on this idea, the author argues that different urban climatic variations can be linked to different urban landscape types. Thus, there is a spatial correlation between urban fabric, land-cover, structure, and urban metabolism (Oke et al. 2017).

This idea underlies the notion of LCZ. Application of LCZ on a real case is challenging though. Because the lines/thresholds that separates one type of LCZ from another are not distinct in the cities. Modeler are obliged to assign each cluster/neighborhood to a specific LCZ category based on their a priori understanding of locality. At small scale, it is feasible to assign a LCZ to a locality. However, manually assigning each neighborhood to a specific LCZ at the scale of a city or multiple cities can be tedious and time consuming as shown by (Leconte 2014) for Nancy.

2.1.1.1.2 Data-tree structure

In data-tree structure, the number of attributes (columns) varies unlike fixed matrix. The main segmentation criteria can further undergo division depending on variability of the parameter e.g. representativeness.

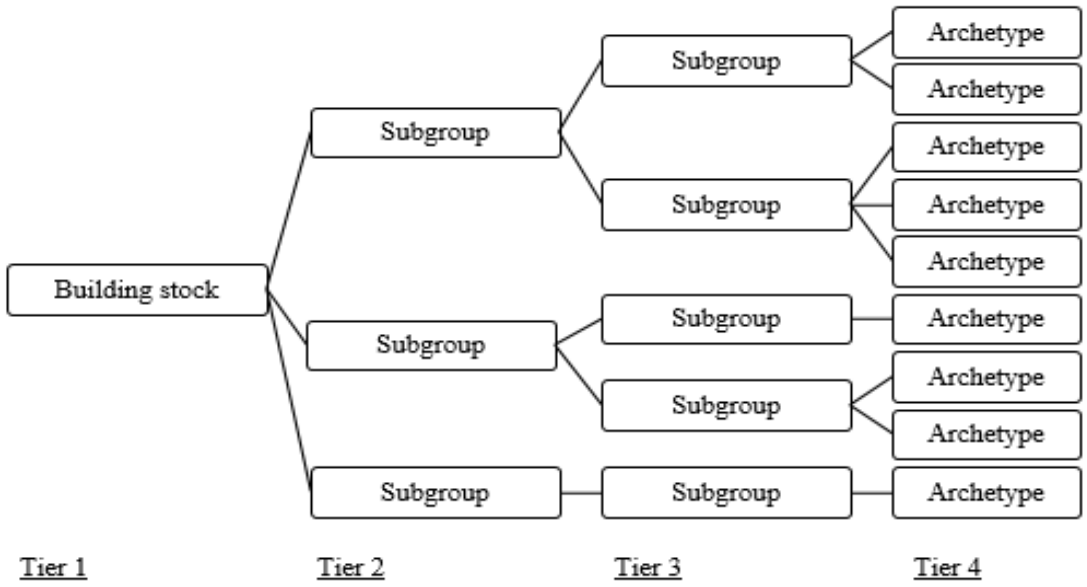


Figure 2-2 : Data-tree structure

For instance, residential buildings are divided into multifamily and single family houses. Multi-family houses represent a large portion of houses and can further be divided into houses with flat and slopped roofs (Monteiro et al. 2015). The number of leaves of the tree shape structure in this approach equals to the number of resultant archetypes.

A major shortcoming of **deterministic approach** is that the classification can be performed only on a limited number of attributes. For example, buildings in TABULA are grouped based on two attributes: type of dwelling and year of construction. Practitioners may find it difficult to build a classification table if/when they are asked to group buildings based on multiple attributes. For instance, 10 attributes. Data-driven solution on the other hand can easily handle high-dimensional data and identify homogenous groups of objects in a large dataset that contains thousands of points and have multiple attributes.

2.1.1.2 Data-mining/data-driven approaches

Data mining is an essential part of a larger framework called knowledge discovery database (KDD) but some authors do not make a distinction between data mining and KDD. In brief, data mining is the process of search and extraction for potentially valuable information also referred as knowledge in large databases (Bhojani 2016).

(Ghuman 2014) has divided the KDD process into seven steps:

- **Data cleaning**: resolving inconsistencies such as noisy data, missing values, and detecting outliers.
- **Data integration**: multiple data sources are combined into a common source.
- **Data selection**: determination of relevant data from raw dataset using engineering or statistical feature extraction methods. Engineering techniques employ expert knowledge to distinguish relevant features whereas statistical methods make use of data-mining or

statistical techniques such as Analysis of Variance (ANOVA), Chi-squared, or coefficient of correlation statistical technique.

- **Data transformation**: transforming nominal or categorical data to numeric data, standardizing data by converting all the values between 0 and 1 or between -1 and 1, reducing dimensions (attributes) using Principal Component Analysis (PCA) or Singular Value Decomposition (SVD).
- **Data mining**: application of techniques to extract patterns.
- **Pattern evaluation**: identifying pattern representing the target knowledge
- **Knowledge representation**: visual representation of the knowledge. (Ghuman 2014)

(Ali et al. 2019) used the term data pre-processing steps referring to the first four steps of KDD. Based on the literature review on building stock typology, data-mining techniques for this purpose could be divided into predictive and descriptive sub-groups (Gera and Goel 2015).

The main distinction between the two methods is based on their application; the first method aims to say something about the future results with the help of past and the second method determines what happened in the past by analyzing stored data.

It is important to note that the techniques presented here do not represent all existing data mining techniques, tools, and algorithms that are also widely used in other fields of science.

2.1.1.2.1 Predictive algorithms in data mining for building stock typology

Predictive techniques are also known as classification/ regression techniques as well. In classification, each element in the database is assigned to a class according to its similarities (Rokach and Maimon 2015).

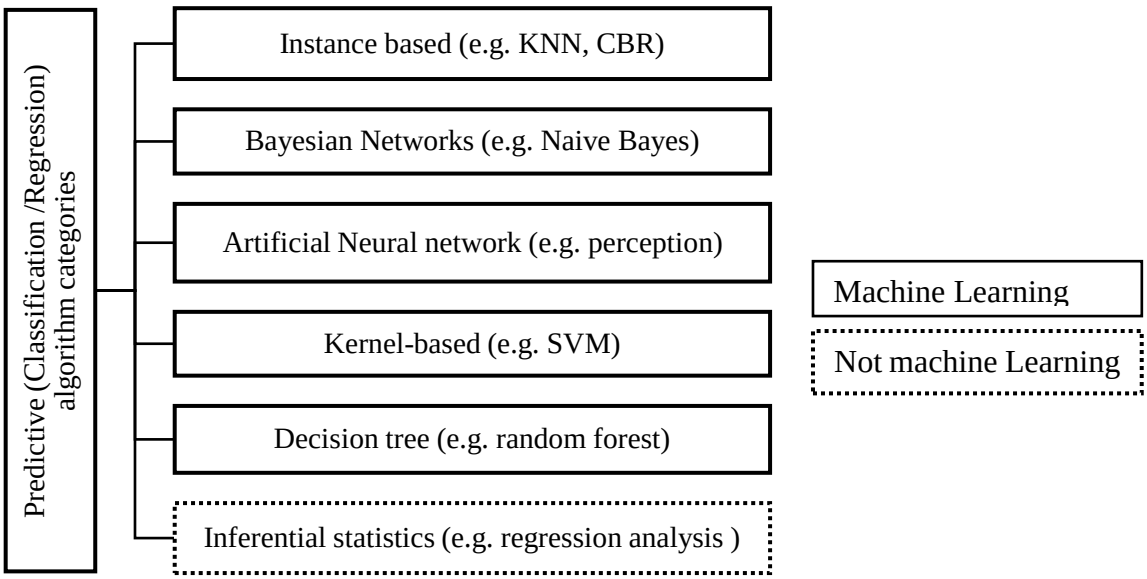


Figure 2-3 : Predictive techniques

Predictive algorithm categories are similar to supervised machine learning techniques in which there is a provision of labelled data. This means the training data you feed to the algorithm contains the desired solution. A typical task of this learning technique is classification. For building stock classification, practitioners need to label a significant number of buildings and train the model on those labeled buildings. After, that algorithm should be able to associate new entries to each class based on similarities that it identified from training data in each class.

Another task of this learning technique is prediction, which is also called regression, where the algorithm predicts a numeric value; such is price of house, energy consumption, etc. using the data model it generated/learned from previous observations.

As mentioned, application of predictive machine learning in buildings stock classification requires a significant number of labelled data. Manually labeling buildings based on their energy performance and/or thermal comfort performance is, first, tedious and time consuming, and second, prone to human error and biases.

An alternative way is to use unsupervised machine learning that has been extensively used in various fields of studies, including medical sciences and market research, for identification of homogenous objects in data sets. Their potential on building stock classification, however, has not been explored sufficiently.

2.1.1.2.2 Descriptive algorithms in data mining for building stock typology.

Descriptive algorithms in general recap and transform the data into presentable information for reporting and extracting meaning (Agyapong, Hayfron-Acquah, and Michael 2016). Sub-categories of descriptive techniques are presented in Figure 2-4.

Summarization algorithms are not always listed under machine-learning learning algorithms. However, they are an important part of descriptive statistics. They are grouped into univariate and multivariate analysis. Univariate analysis is considered to be simplest form of analysis and its dataset contains one variable. Univariate does not deal with causes, relationships, etc. its major objective is to describe the variable in a simpler format e.g. find the mean, mode, or standard deviation.

Multivariate analysis, on the other hand involves more than one variable. It can contain dependent or/and independent variables in a dataset. It also means that these variables could be analyzed simultaneously or separately. Multivariate analysis is mostly used for the following objectives: (1) data reduction (structural simplification); (2) grouping; (3) dependence evaluation; (4) prediction of one or two variables observing the changes in other variables; (5) statistical hypothesis testing.

Second category of descriptive algorithms is anomaly detection. This category is considered a step in data mining, and its objective is to find points that deviate from normal behavior of observations.

The third category of descriptive techniques, which is the focus of this study, is cluster analysis.

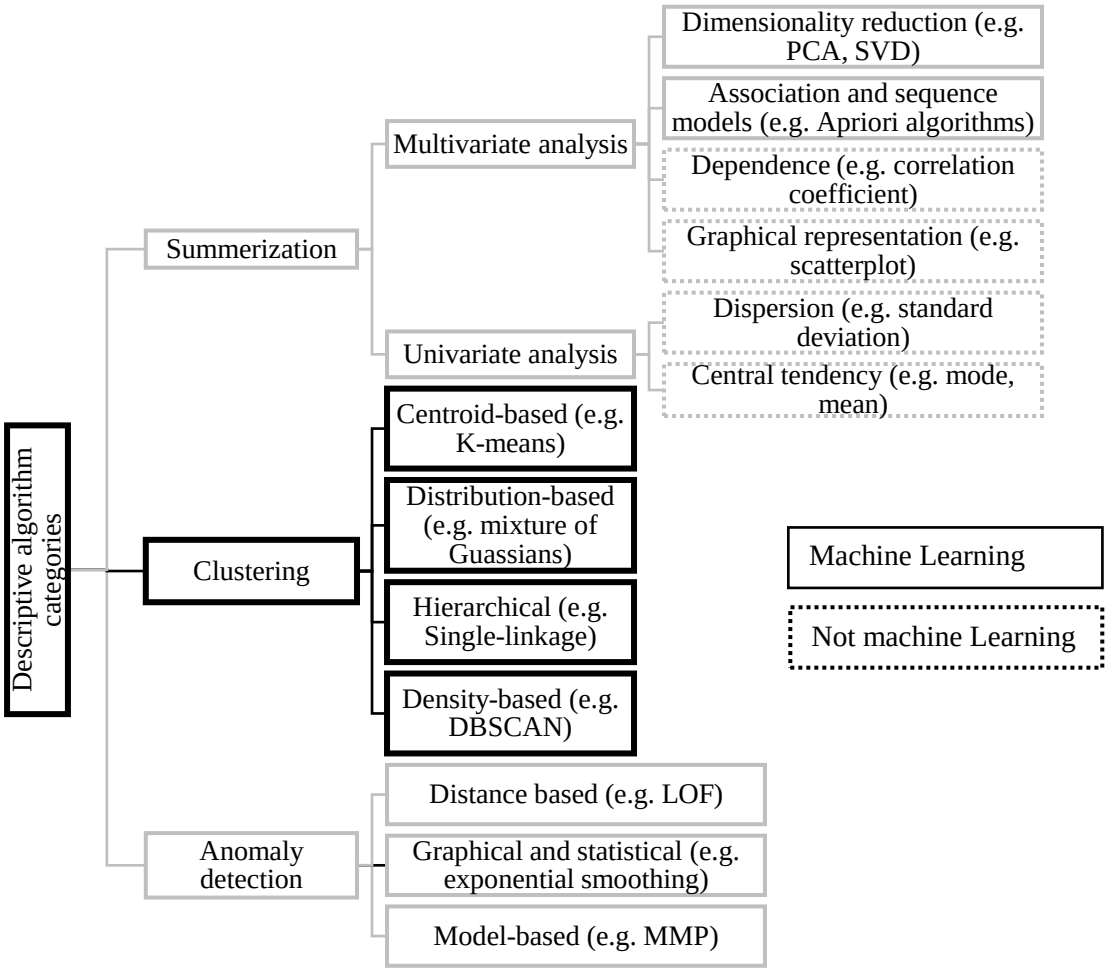


Figure 2-4 : Descriptive algorithm categories

The objective of cluster analysis is usually to group a set of objects in such a way that the objects in the cluster are more similar to each other than to objects in other clusters. We, henceforth, use the term clustering when we talk about cluster analysis.

In contrast to predictive learning techniques, for clustering which is also referred to as unsupervised machine learning, there is no need for provision of labelled data. For instance, a building practitioner wants to split a dataset of buildings into multiple homogenous groups of buildings. At no point the practitioner needs to tell the algorithm which cluster or class does this specific building belong. Clustering algorithm associates it automatically.

The buildings in our case study city (Nantes) are not labeled; therefore, our best option with data-driven method is to use ***Unsupervised Machine Learning (clustering)*** to group buildings into homogenous clusters.

The final goal is to select one representative building from each cluster as a reference (typical) building of that specific cluster.

In the data and method section, the creation of input database, preparation of input data, performance assessment indicators, clustering techniques, and extraction of reference building for a case study are described with more details.

2.2 Data and Methods

2.2.1 Input data for clustering

In line with the overall objective of this thesis that focuses on integration of urban heat island effect, climate change and heat waves in indoor thermal comfort and energy consumption assessment of buildings at city scale, the input attributes for clustering were selected.

In machine learning, *attribute* is referred to the data type, (e.g., building year of construction), while a *feature* can have several meaning, depending on the context, but here we use it when we talk about the value of attribute (e.g., building year of construction = "1964").

Numerous parameters are seen/used in scientific papers that explain urban climate and heat island effect. Parameters that were selected as attribute for clustering here, meet at least one, two, three, four, or all of the five following criteria:

(1) They have potential effects on urban heat island.

(2) They are the elements that have the potential to indicate the exposure rate of indoor thermal conditions to outdoor climate variables, such as exposure to solar radiation and temperature.

(3) They are relatively easy to calculate.

(4) They are controlled by design of local urban plans.

(5) The input information for calculations can be accessed from open source data.

Raw data used in this study to build attributes for clustering were accessed from the following openly available data sources:

BDTOPO: BDTOPO is a 2D and 3D vector spatial database containing description of landscape elements including but not limited to building footprint, building height, building year of construction, vegetation coverage, water coverage ratio, presence of trees, transportation routes, and etcetera throughout France.

Following parameters were extracted for the whole city: building footprint area, building height, vegetation coverage (NDVI), and water surface coverage.

Land data of CEREMA (données foncier): year of construction of building parcels (buildings).

MAPUCE project: reference spatial unit (RSU) boundary lines. RSU lines divide buildings into blocks/neighborhoods.

RSU lines were computed in the context of MAPUCE project for all French urban territories. Computation of these lines are based on dual of Delaunay triangulation. Characteristics of Voronoi tessellation are used to calculate new blocks/neighborhoods. Separation lines correspond to the medial axis of negative areas of cadastral parcels (Bocher et al. 2018). Each block/neighborhood is called a RSU, as shown in Figure 2-5.

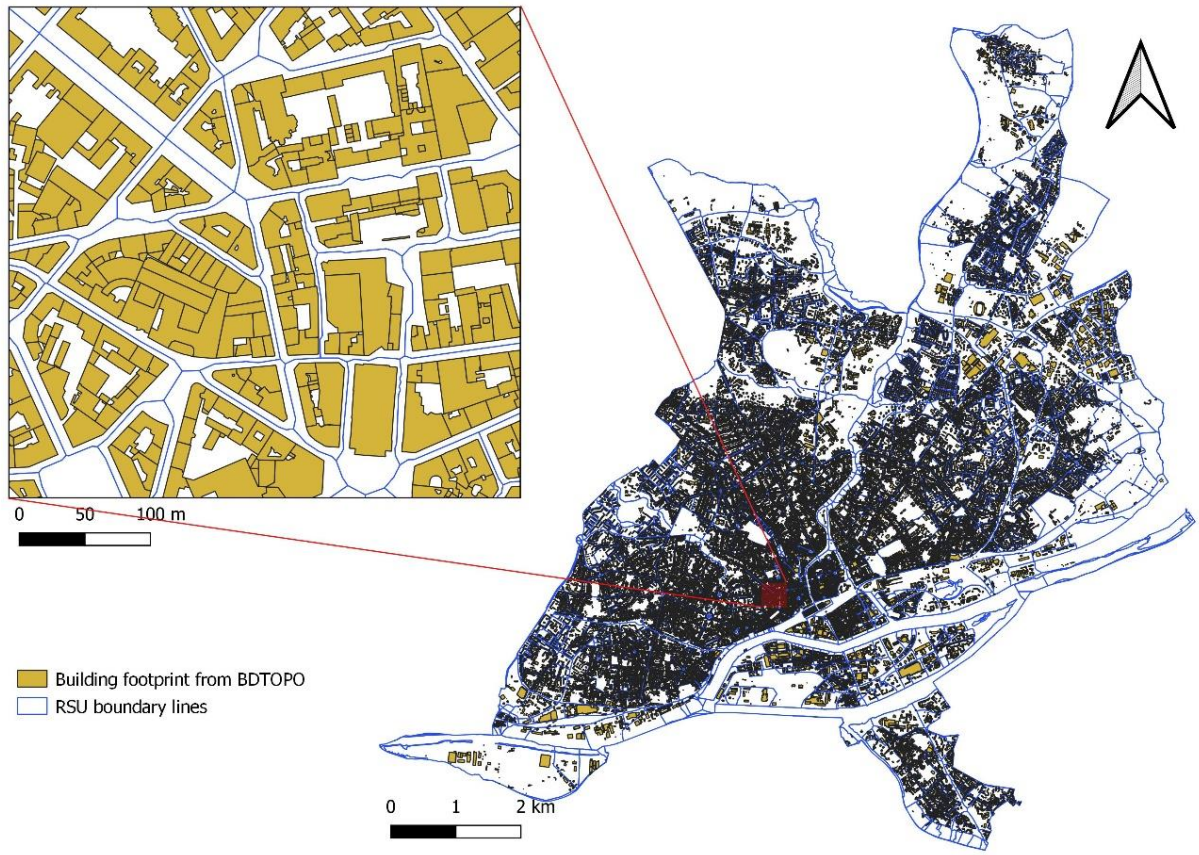


Figure 2-5 : Reference spatial unit (RSU) boundary lines for case study city

Data from year of construction and other building parameters from BDTOPO were joined into one data frame by spatially superimposing one vector map over another.

Superimposing RSU lines and BDTOPO map, also allowed us to calculate building footprint ratio, vegetation and water coverage percentage inside each RSU.

The following parameters that affect variations in urban microclimate and indoor thermal conditions were added or synthetically calculated:

- **Building height**: the z-value in BDTOPO database

- **Building volume**: it was calculated using BDTOPO data by multiplying the area of polygon to height of the polygon.

- **Free vertical area ratio to total vertical area of building**: we used python's GeoPandas package to calculate the length of shared wall between two adjacent buildings and multiply it by the height of shortest building. Since a building can have more than one joint neighbor, we first identified what are the joint neighbors of each polygon, then length of shared wall between each joint neighbor and target building were calculated and multiplied by the height of shortest polygon. At the end, all shared areas were summed and added as an attribute to a database. Shared area of building was then divided to total vertical area of

building giving us the ratio of vertical area ratio to total vertical area of building. (**Appendix 2-1)**.

- **Building net compacity**;

$$Building\ Net\ compacity = {Free\ vertical\ area}/{(Building\ Volume)^{2/3}}$$

- **Building year of construction** from données foncier of CEREMA

- **Buildings' distance from peripheries**: this distance is referred to the distance of each building from closest line that separates urban and non-urban areas. The boundary line that delineates urban and non-urban areas were calculated using **MorphLim** application[3]. This application is designed to identify morphological boundary of urban agglomerations. Urban boundary in this application is identified in three steps. First, it surrounds each built polygon with a buffer of increasing width. The width increases to a geometric logic and the number of clusters are counted at each dilation step. It then plots the process on a log-log graph, where the X-axis is the width of dilation and Y-axis is the number of clusters. In the second step, it identifies a distance threshold on dilation curve that shows a major change in its behavior. Among the maximum curvature points, the point of main curvature has the highest absolute value of curvature. From there, the point of main curvature is located on the estimated curve, which gives the distance threshold to plot urban envelope (Diameter). In the final step, the vector file showing urban envelop is exported (Tannier and Thomas 2013).

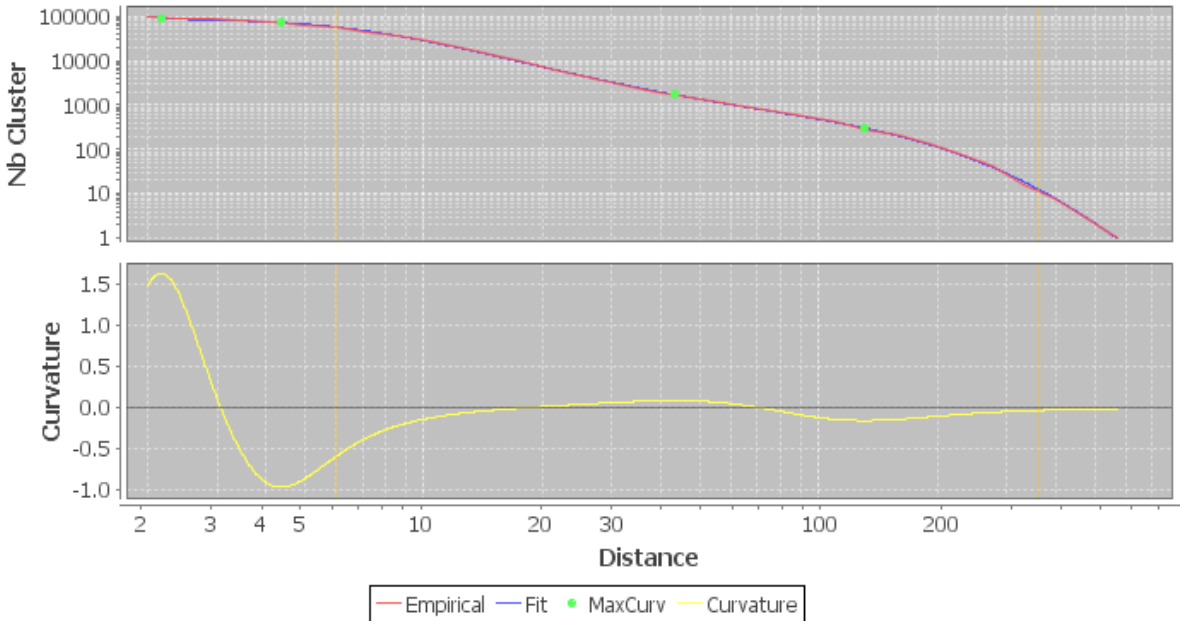


Figure 2-6: Curve fitting of MorphLim for Nantes

For this case study city (Nantes), diameter threshold (distance) = 130.365m is where the main curvature is located. Therefore, it is calculated to represent critical diameter limit of built cluster in the shape file.

Buffer radius = Diameter threshold/2 = 130.3/2 = 65.15m.

[3] https://sourcesup.renater.fr/www/morpholim/

Figure 2-7 shows the position of buildings and urban envelop boundary line in Nantes., The distance of each building from the closest line that separates urban and peripheries was measured and added as a feature (Bernard 2017).

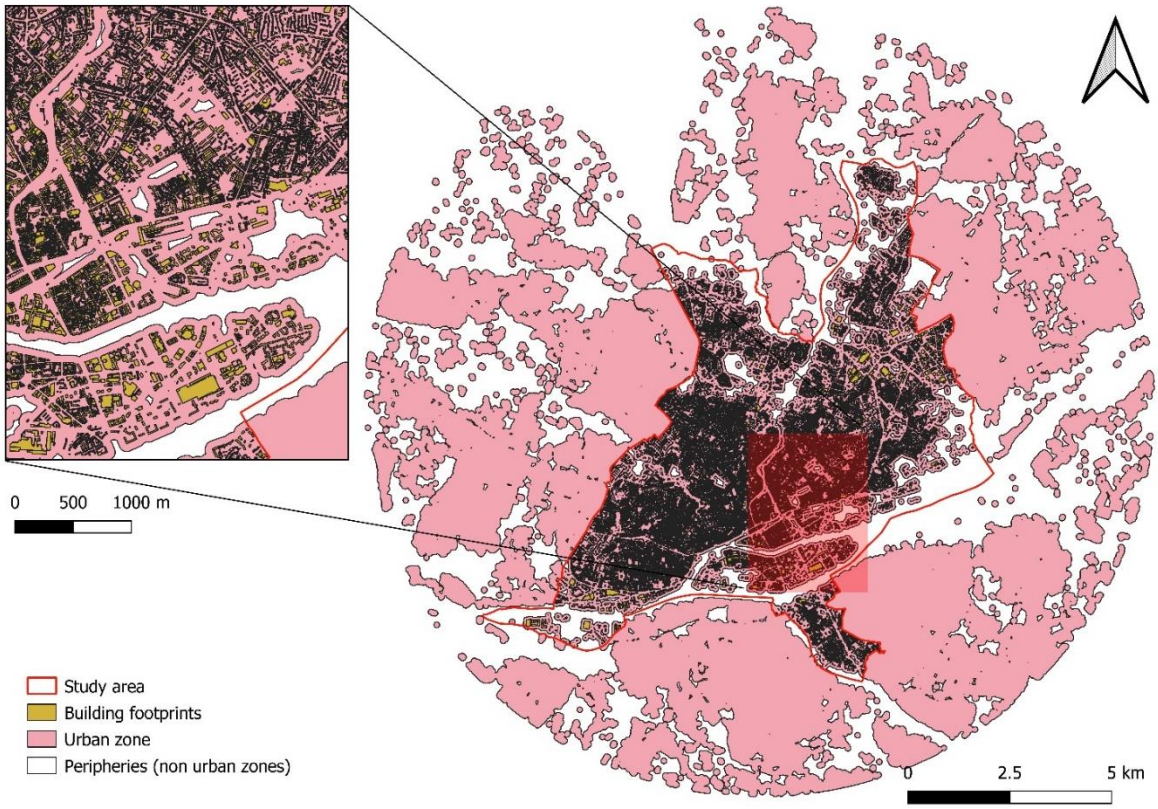


Figure 2-7 : Urban envelope boundary line of Nantes

- **Vegetation and water surface percentage in RSU**: after superimposing RSU lines and BDTOPO, the percentage of RSU area covered by vegetation and water were calculated.

- **Building footprint density in RSU**: total areas of buildings in a RSU/area of RSU

- **Façade density in RSU**:

$$Facade\ density\ in\ RSU = \frac{Free\ vertical\ areas\ of\ buildings\ in\ RSU}{(Free\ vertical\ areas + area\ of\ RSU)}$$

The map in Figure 2-8 shows façade density in Nantes.

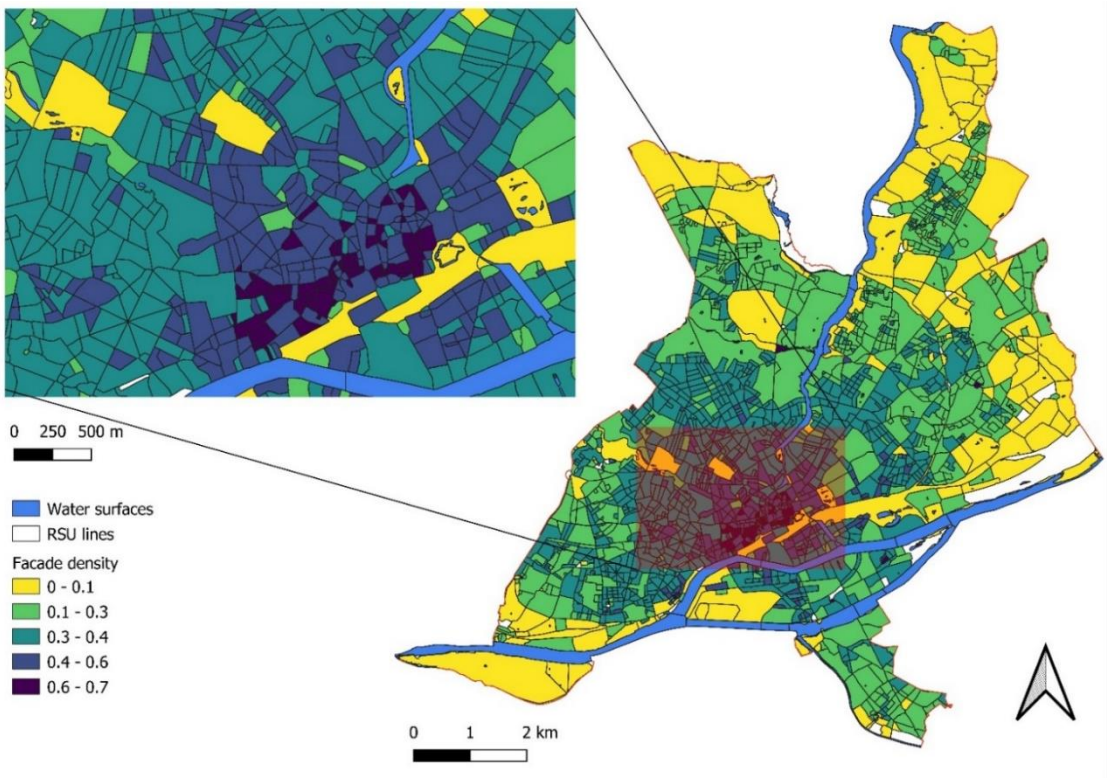


Figure 2-8 : Facade density of buildings at RSU scale in Nantes

- **Sky view factor at RSU scale**: Sky view factor was calculated using UMEP of QGIS[4].

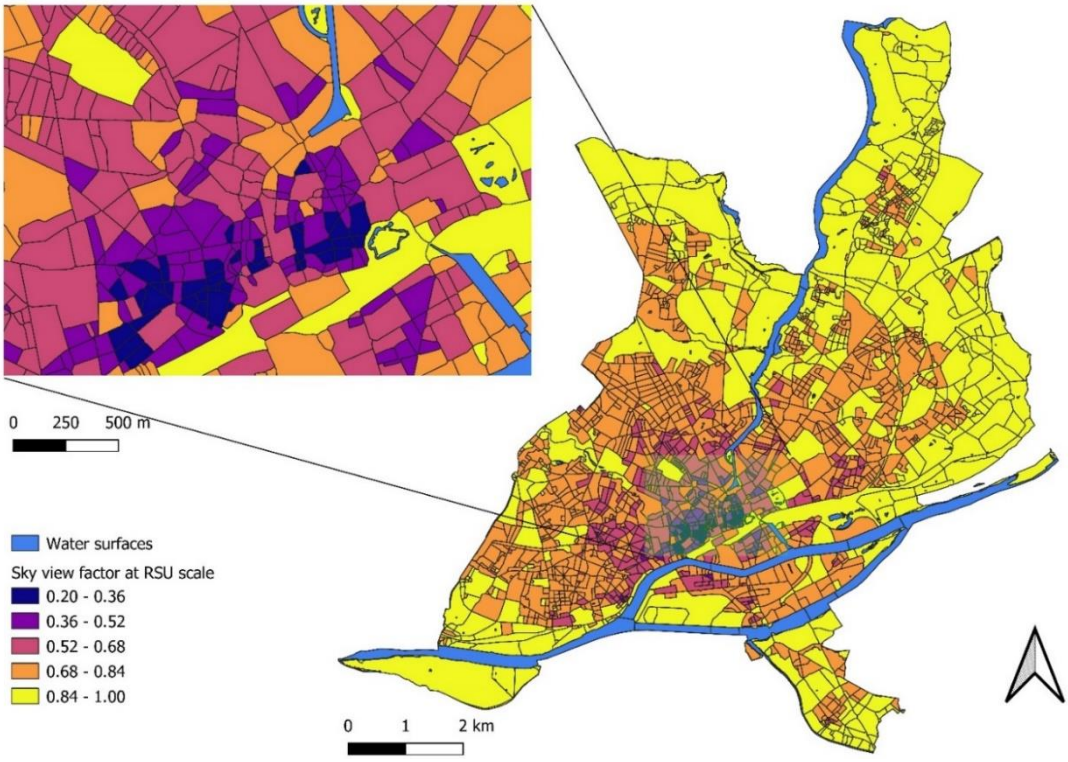


Figure 2-9 : Average sky view at RSU scale

[4] https://plugins.qgis.org/plugins/UMEP/ and (Lindberg et al. 2018)

- **Buildings' roofs Sky view factor (SVF):**

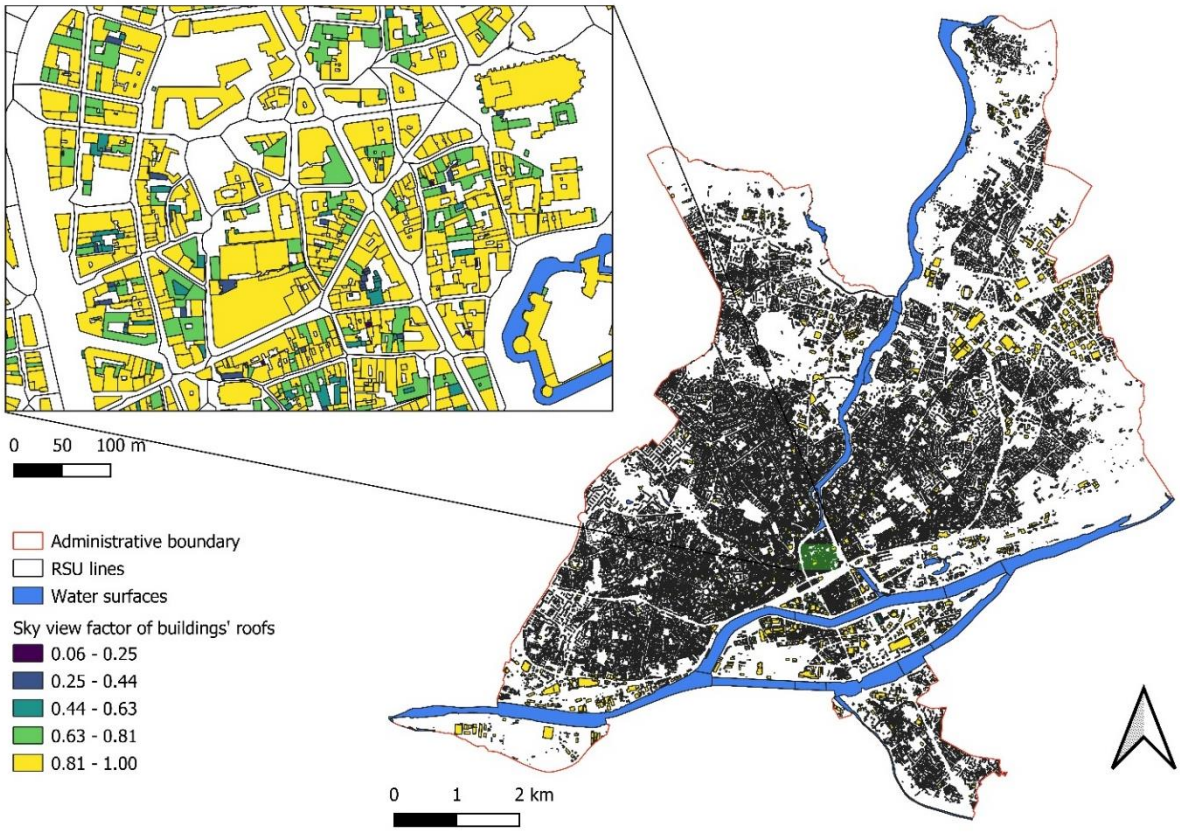


Figure 2-10 : Average roofs' sky view factor (SVF)

Appendix 2-2, shows how the output of UMEP in QGIS in the form of a raster map is transformed into a vector file, which is then used as input for clustering.

- **Building shape**: using building shape for clustering required transformation of closed polygon into a format that can be used in clustering. To do so, as can be seen in Figure 2-11,

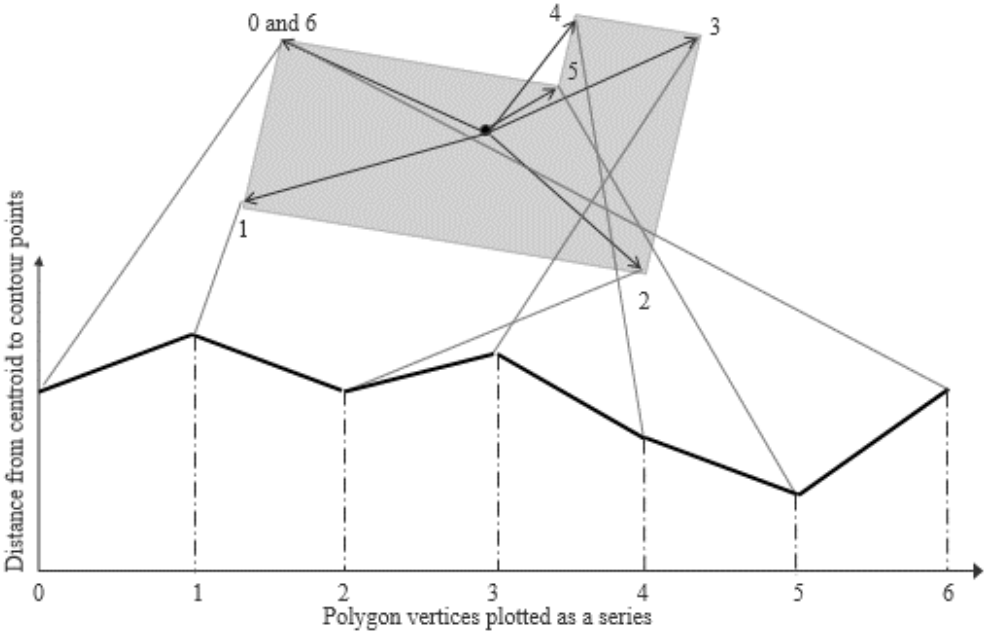


Figure 2-11 : Building polygon to series

Building polygons that have the forms of closed polygons were decomposed into a series of data by measuring the distance of each vertex of building from the center of gravity of building. Application of a clustering algorithm that measure the distance between the given series can be effective to identify/cluster buildings that have similar shapes in database. When buildings have too many vertices or when they are circular, the length of series becomes too long. Many of the unsupervised algorithms do not perform well when the number of attributes is too high. Therefore, we decided to exclude building shape from further processing.

After calculation of the given parameters and their aggregation with BDTOPO parameters and land data, a Pearson Correlation analysis was performed to see if there is a collinearity between the input parameters. The analysis showed a strong correlation between SVF at RSU scale, Building footprint density, and façade density. Therefore, SVF at RSU scale, and Building footprint density were excluded from further processing. There was also a strong correlation between area of buildings and their volume. Of the two, volume was selected for further processing. (**Appendix 2-3**)

2.2.2 Data preprocessing

Following the creation of input data, a data cleaning procedure was applied to handle missing values, detect and remove outliers.

Missing features of attributes were filled out with nearest neighbor principle or removed from dataset.

Outliers were detected by graphing the data points, and removing those that were drastically different from the bulk of data points.

Data was then normalized, and transformed before being used in data mining algorithms. The objective of data-transformation is to make data normal distribution-like, and stabilize variance.

Data columns used as input for clustering to extract reference buildings are listed below:

1- Building height
2- Building volume
3- Buildings' roofs SVF
4- Buildings' Net_compacity
5- Building year of construction
6- Free vertical area ratio to total vertical area of building
7- Buildings' distance from peripheries
8- Normalized difference vegetation index (NDVI)/vegetation percentage in RSU
9- Water surface percentage in RSU
10- Façade density of RSU

2.2.3 Framing the problem of reference building identification

The objective here is to cluster similar instances of buildings in one cluster and identify the most typical one in each cluster, for the case study city. Another aim is to minimize the influence of a priori assumptions/biases of modeler on what cluster or class does a specific building belong.

With this in mind, as mentioned in section 2.1.1.2, unsupervised clustering techniques was selected to identify cluster of buildings that share similar characteristics. Since the data was already preprocessed and it was not going to change, *offline batch learning* approach was used to run the clustering algorithms. Alternative way is an *online* approach, and it is recommended for cases when input data continuously updates or when input data is so large that computer runs out of memory to process them in one batch.

Note: for practitioners that have huge datasets that do not continuously update, they can also split batch-learning work across several servers, using MapReduce techniques.

2.2.4 Performance measurement indicators

There are several techniques to assess how well your machine-learning model works, but the majority of them focus on lowering the error between the actual and predicted values, such as relative means squared error (RMSE) or mean absolute error (MAE). This approach is more suitable to supervised machine learning, but for unsupervised machine learning, like in this study, when the ground truth is unknown, other ways of measuring are used.
Quality of clustering is assessed using some similarity or dissimilarity metrics, such as the distance amongst cluster points. If the clustering algorithm successfully identifies dissimilar and similar values, it has done a good job (Géron 2017). The **Average Silhouette Score**, **Calinski-Harabasz** indicator, and **Davies-Bouldin index** are three of the most frequently used metrics for clustering method assessment.

Average Silhouette score:
This indicator is used to compare how similar an object is to its own cluster or in other words, how dissimilar it is to other (neighboring) clusters. Results of this indicator produce a value between -1 and +1, where a score near +1 implies that the item is well matched in its cluster. A value of zero or negative means the object is very badly matched in its cluster (Pedregosa et al. 2011; Rousseeuw 1987). Both Euclidean and Manhattan distance metrics can be used to measure the degree of similarity or dissimilarity, but here we used Euclidean distance.

Calinski-Harabasz indicator:
In clustering, Calinski-Harabasz index is used as a measure of the quality of a partitioning in a dataset. This indicator is the ratio of inter-group variance to intra-group variance. Therefore,

algorithms attempt to maximize this score. The value of this indicator varies between 0 and +∞ and the value strongly depends on the number of points in the clustering (Pedregosa et al. 2011).

Davies-Bouldin index:
This indicator is used to define the average degree of similarity measure of each individual cluster with its most similar cluster, and the similarity is the ratio of inter-cluster (within-cluster) distances to intra-cluster (between-cluster) distances. This indicator varies between 0 (best separation) and +∞ (worst separation) (Davies and Bouldin 1979; Pedregosa et al. 2011).

2.2.5 Clustering algorithms

2.2.5.1 Partitioning algorithms (Centroid-based)

This group of algorithms split the "n" number of data into "k" number of clusters (groups). This separation method is preferred over the hierarchical model in pattern recognition. The following criteria are often established to satisfy the methods:

Each cluster must have at least one object.

Each data object belongs to one cluster.

The most commonly used partitioning methods are K-Means, K-Medoids, Partitioning around Medoids (PAM), and Clustering Large Applications (CLARA) (Chitra and Maheswari 2017). Here, we only briefly describe K-Means and K-Mediods.

K-Means: In this algorithm, a cluster of data points is represented by its centroid, which is obtained from the average of points in that particular cluster. The average point is therefore a fictive point and does not correspond to a real data point in the cluster.

It is also the most commonly used algorithm in scientific and industrial application.

Advantages:

- Effective in dealing with large data sets (computationally not expensive).
- Generalizes to clusters of different shapes and sizes, such as elliptical clusters.
- Frequently terminates at local optimum.
- Works only with numerical values.

Disadvantages:

- Does not perform well with noisy data or when number of attributes is more than 20.
- Requires specifying the number of "k" clusters at the beginning.

A way to overcome this latter limitation is to use Elbow technique, as in Figure 2-12, to determine the optimal number of clusters in K-Means. In this technique, K-Means clustering is run on a range of values of K (say 2 to 15), and then for each value of K the average distortion score is calculated. Distortion score is referred to the sum of the squares of the distance from each point to its specified center.

Application of elbow technique is possible for K-Means, mainly because this algorithm is not computationally expensive.

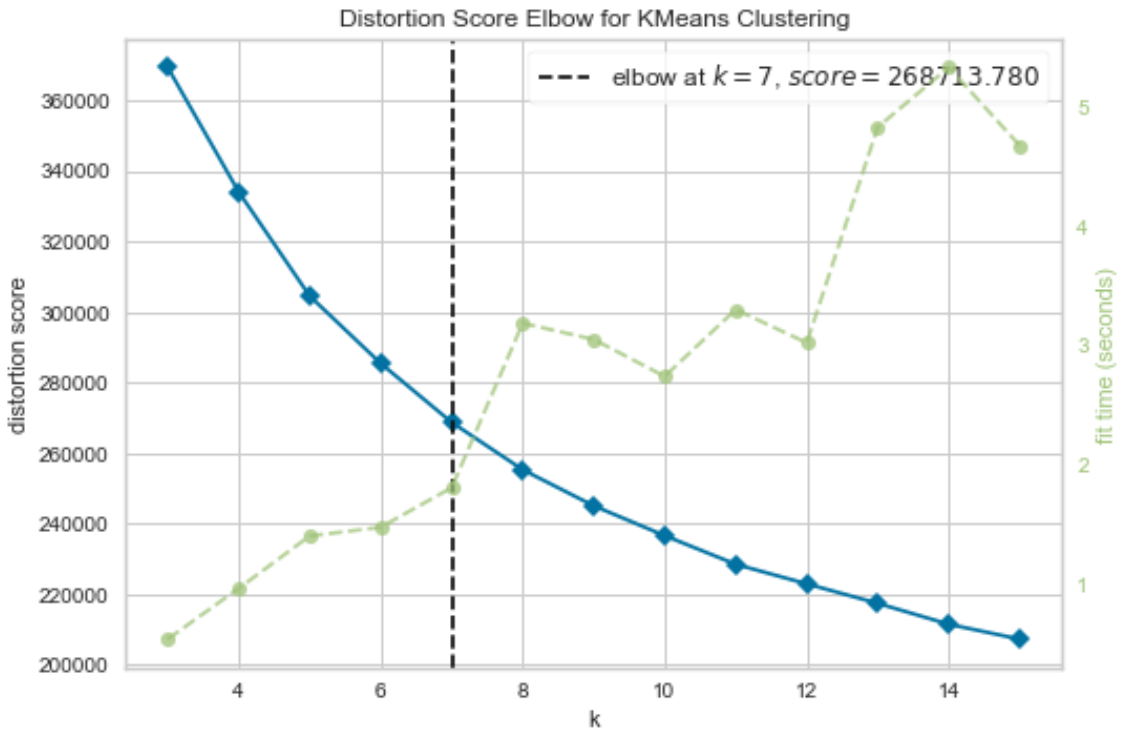


Figure 2-12 : Elbow technique to determine number of clusters

K-Medoids: this algorithm uses an actual point in the cluster to represent the cluster. The partitioning is based on minimizing the sum of dissimilarities between objects and the representative of cluster.

Advantages:

- Identifies a real data point in each cluster to represent cluster.

Disadvantages:

- Requires specifying the number of "k" clusters at the beginning.
- Computationally expensive.

The main advantage of partitioning algorithms over other types of algorithms is that it is simple to implement, and understand them. It takes significantly less time to execute them. Main drawback is that practitioners need to specify the number of clusters in advance and they generate only spherical shaped clusters.

2.2.5.2 Hierarchical clustering

This method creates a cluster from top to bottom (divisive) and/or bottom to top (agglomerative). In both of these approaches, it first requires creation of a dendrogram to visualize the suggested number of clusters. A dendrogram is a tree-like format that stores a sequence of clustered clusters.
The typical of way selecting number of clusters is by drawing horizontal lines that pass through the longest arms of dendrogram. As can be seen in Figure 2-13, we can split the data into 3, 5,

and 9 clusters using hierarchical clustering. Performance indicators show that quality of clustering is highest when number of clusters is 3.

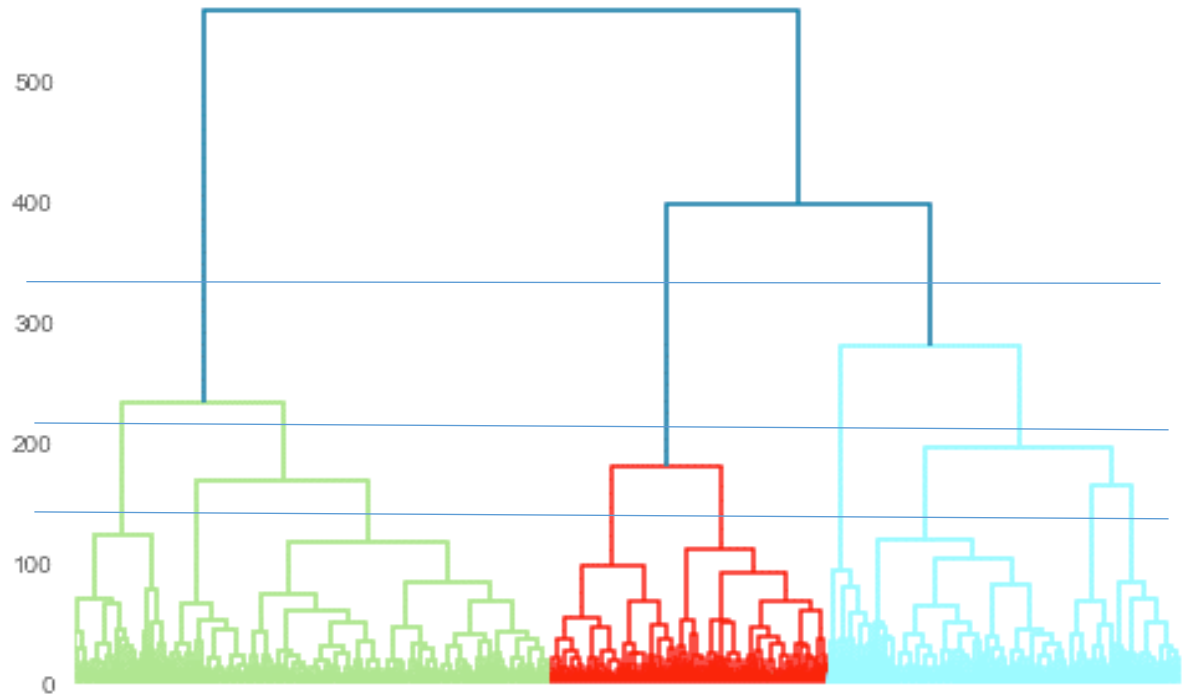


Figure 2-13 : Dendrogram of hierarchical clustering

In this study, we used **Agglomerative Nesting** and **Birch** clustering techniques of hierarchical clustering to split the building data into clusters. Other techniques of hierarchical clustering seen in literature are CHAMELEON and DIANA (Chitra and Maheswari 2017).

Key advantage of hierarchical clustering is the flexibility of level of granularity and its ability to handle any type of data (attribute type). One drawback of hierarchical clustering is that Agglomerative nesting technique is computationally more expensive than K-Means but less expensive than K-Mediods. Birch, on the other hand, is efficient at processing large volumes of data. Another major limitation of hierarchical clustering is that it is sensitive to noise and it breaks large clusters even if they are similar (Chitra and Maheswari 2017).

2.2.5.3 Density-based algorithms

In this model, clusters are defined by placing areas of higher density in the cluster. Some of density-based algorithms are Density Based Spatial Clustering of Applications with Noise (DBSCAN), Generalized-DBSCAN (GDBSCAN), Ordering Points to Identify the Clustering Structure (OPTICS), and DBCLASD.
We used DBSCAN in this study. The density based algorithm DBSCAN is the commonly known algorithm of this group of algorithms. In execution of this algorithm, the basic principle is to concentrate on two parameters: the maximum radius of the neighborhood (Eps) and the minimum number of points in a cluster (Min pts). DBSCAN model identifies clusters of varying

shapes and noise. It works by detecting patterns and estimating the spatial location and distance to the neighbor.
Main advantage of DBSCAN algorithm is that practitioners are not required to specify number of clusters and that it is able to handle noisy data. It does not perform well with high dimensional data (Chitra and Maheswari 2017).

2.2.5.4 Distribution-based clustering algorithms

Distribution-based also known as model-based relies on the assumption that data is made up of probability distributions. Unlike k-means, which is a hard clustering and captures the mean of spherical-shaped clusters. Gaussian mixture distribution can capture the means and variances of different elliptical-shaped clusters as well. If the variance is encoded as a matrix instead of just a number, this allows the distribution to be spread out more in one direction than another does. This method yields optimal number of partitions. As such, the modeler does not need to enter the initial number of partitions before analysis. In this approach of clustering, as the distance of point increases from the center, the probability of it to belong to the distribution decreases.

Appendix 2-4 presents python script used for data pre-processing, normalization, transformation and application of clustering algorithms using Scikit learning (Pedregosa et al. 2011).

2.3 Results of clustering

2.3.1 Comparison of clustering techniques

The objective is to split residential buildings of our case study city (Nantes) into homogenous clusters and select one representative building from each cluster. To achieve this objective, input parameters described in section 2.2.1 (input data for clustering) that show UHI effect and exposure of building to external environmental conditions are assembled in a databases.
Input data was then pre-processed and made ready for cluster analysis. Performance of clustering algorithms were assessed with three indicators, average silhouette score, Calinski-Harabasz, Davies-Bouldin index.

Overall, five different techniques of clustering, at least one from each category of methods, were used to group the residential buildings into homogenous clusters: KMeans from partitioning algorithms, Birch and Agglomerative Nesting from hierarchical clustering, DBSCAN from Density-based algorithms, and Gaussian Mixture Model from Distribution-based clustering. Because we did not know what is the right number of clusters for the given dataset, we run the algorithms multiple times for each technique to find the number of clusters

and technique that generate the best cluster analysis. Results are summarized in Figure 2-14, below.

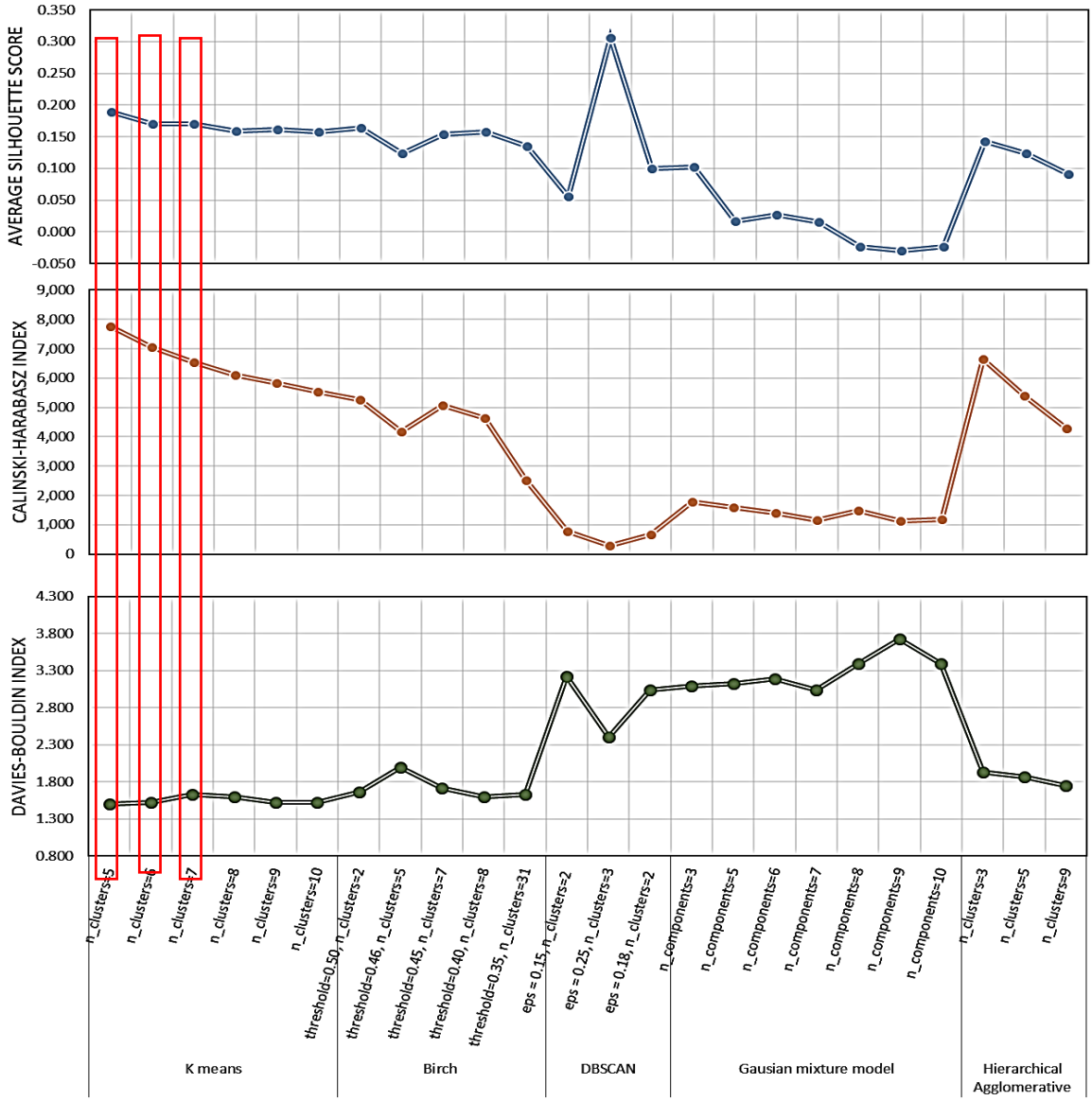


Figure 2-14: Comparative results of clustering

As can be seen in Figure 2-14, K-Means with number of clusters equal to 5, 6, and 7 performs better than other clustering techniques in all indicators. Silhouette score index is highest for DBSCAN when number of clusters are 3, but Calinski-Harabasz index and Davies-Bouldin index show that it performs poorly. Relative difference between the values of indicators with K-Means when number of clusters are 5, 6, and 7 are small, therefore we took into consideration the results obtained from Elbow method, as presented in Figure 2-12.

For the same reason, it was decided to use K-Means when number of clusters is equal to 7 as the final technique to split residential building data into clusters. As discussed, with K-Means,

the centroid is the average of all entries in that cluster. The average, obviously, is not a real entry point, so there is a need to select a real building from each cluster to represent that cluster.

2.3.2 Identification of reference buildings

To do so, using Euclidean Distance, the distance of each entry point to the cluster centroid was measured and the real entry point closest to the centroid was selected as the reference building in each cluster.
Appendix 2-5 shows the position of closest building to/from the centroid of clusters.
The index of closest building to the centroid of each cluster was then exported and the building was identified on the map (see **Appendix 2-4** for python script).
Figure 2-15 shows the position of seven reference buildings identified through the methodology described in methods section, for the case study city, Nantes.

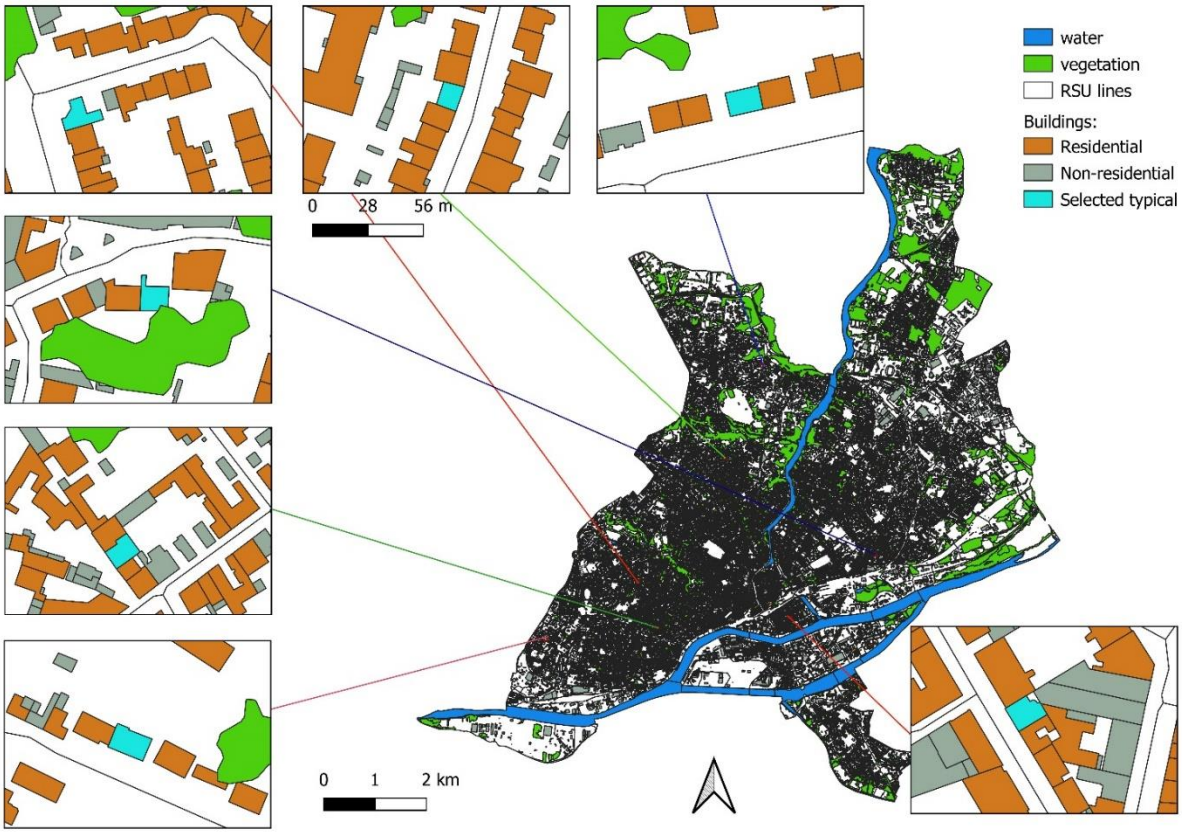


Figure 2-15 : Reference buildings identified with KMeans when n_clusters =7

As mentioned, the relative scores of indicators for K-Means when number of clusters are 5, 6, or 7 were pretty close. Same methodology could have applied on those as well. However, selection of 7 instead 5 or 6 gives the practitioners/researchers the possibility to include a wider range of variations.

Clustering here split buildings with consideration of their surrounding parameters that influence UHI effect in each (RSU). Consideration of UHI parameters for each cluster then allowed to project the influence of UHI effect on the reference building of each cluster and by extension

to all buildings in that cluster. The workflow, tool and approach used to project UHI effect on typical weather data is illustrated in chapter 3 of this manuscript.

Table 2-1 presented below demonstrates the details of closest real building to the fictive centroid of K-Means when number of clusters were 7.

Table 2-1 : Identified centroids of residential buildings in Nantes

Reference building	Building height	Building Volume	Net compacity	Roof's mean SVF	Distance from peripheries	Year of construction	Vegetation percentage in RSU	Water percentage in RSU	Façade density in RSU	Free vertical area ratio
KM7_0	6.2	594	4.7	0.96	681.24	1932	9.90	0	0.36	0.80
KM7_1	7.4	732	4.6	0.89	321.00	1900	7.55	0	0.35	0.80
KM7_2	6.9	585	3.3	0.95	710.51	1930	5.95	0	0.38	0.55
KM7_3	4.6	573	4.5	0.96	314.42	1981	5.15	0	0.32	0.86
KM7_4	6.1	431	3.3	0.92	277.93	1952	6.27	0	0.31	0.57
KM7_5	10.3	1139	3.3	0.86	319.38	1880	0.00	0	0.51	0.55
KM7_6	6.5	718	4.4	0.97	176.09	1969	27.20	1.29	0.18	0.87

Map below, shows the parts of city where there is concentration of different buildings. RSUs in this map are assigned to the cluster that accounts for the majority of buildings inside it.

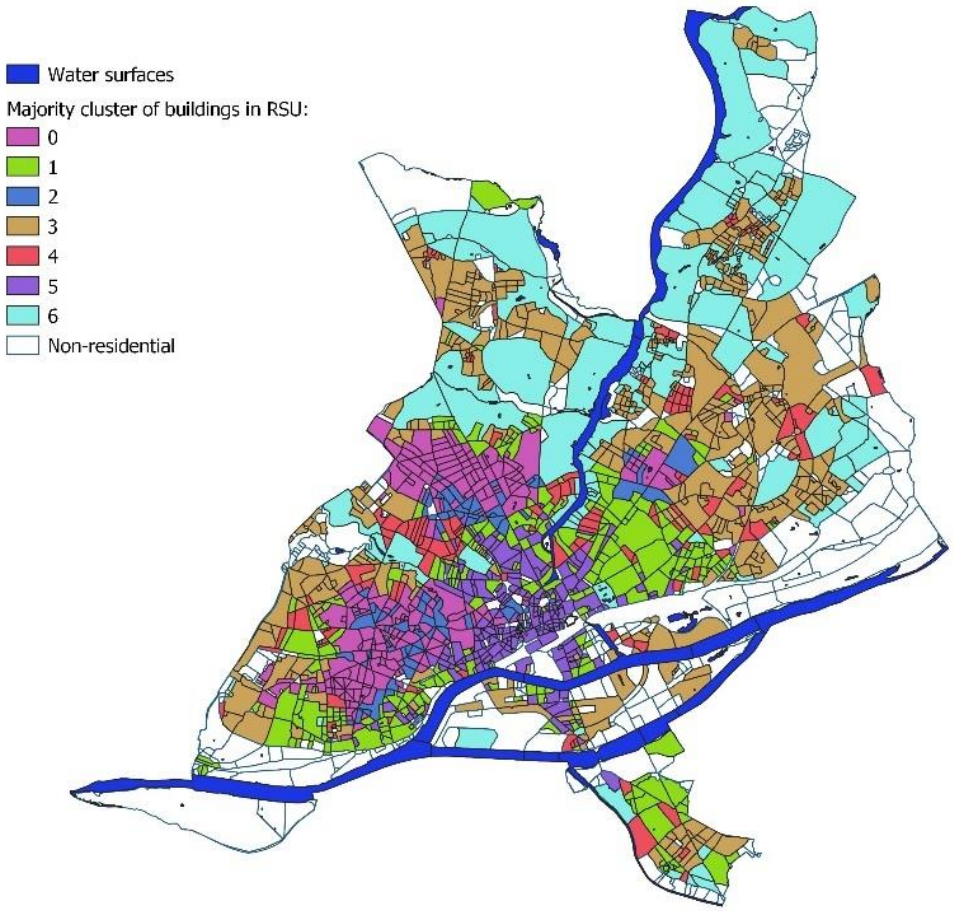


Figure 2-16 : Majority of building clusters in RSUs

2.4 Characterization of reference buildings

After identification of reference buildings using K-Means clustering. The buildings were located on Google Maps (Figure 2-17) and additional details were added to prepare them for further processing.

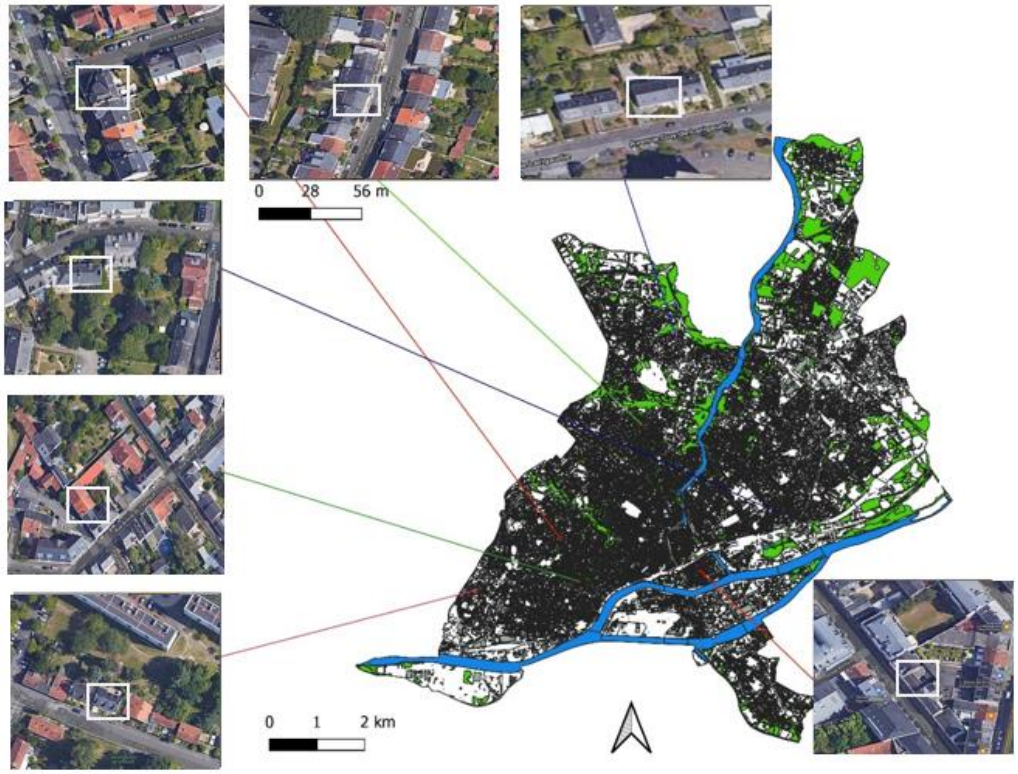


Figure 2-17 : Reference buildings identified with KMeans when n_clusters =7 on Google maps

Characterization of reference of buildings were carried out carefully considering all other buildings in the cluster. Year of construction was selected as a primary parameter to enrich the identified reference buildings. It was selected because majority of the data about thermo-physical properties of buildings are available as a function of year of construction.

Year of construction for selected closest building to centroid of cluster is not exactly the same as the year of construction in the whole cluster. Therefore, we analyzed the distribution of year

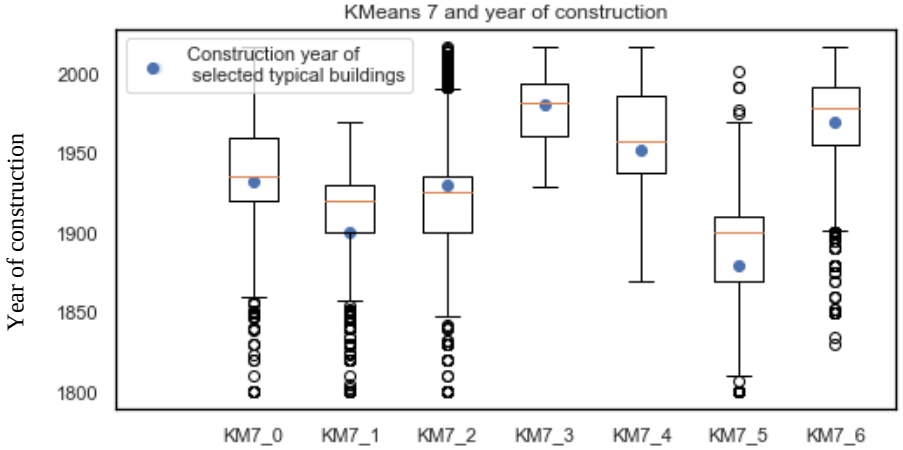


Figure 2-18 : Boxplots of year of construction for residential building clusters

of construction in each cluster. Boxplots in Figure 2-18 and Table 2-2 show statistical distribution of year of construction in each cluster in comparison to the selected closest real building to the centroid.

Table 2-2 : Year of construction in the cluster and values of edges

label	Year of construction of selected reference	Year of construction range in each cluster				
		Lower whisker	Lower quartile	median	Upper quartile	Upper whisker
KM7_0	1932	1860	1920	1935	1960	2017
KM7_1	1900	1857	1900	1920	1930	1970
KM7_2	1930	1848	1900	1926	1936	1990
KM7_3	1981	1929	1961	1982	1994	2017
KM7_4	1952	1870	1938	1957	1986	2017
KM7_5	1880	1810	1870	1900	1910	1970
KM7_6	1969	1902	1955	1978	1991	2017

In Figure 2-18, it is noticeable that clusters KM7_3, KM7_6 and KM7_4 are composed of buildings constructed in recent years and are mostly located in outskirts of the city as shown in Figure 2-16. Clusters KM7_5, KM7_1, KM7_2, and KM7_0 are composed of older buildings, which are concentrated in city center areas.

In Table 2-2, the year of construction of selected reference building is compared to the median year of construction of other buildings within the same cluster. In clusters KM7_1 and KM7_5, there is approximately 20 years difference between the year of construction of the building closest to the centroid (selected representative building) and the median building of the cluster, but as the construction techniques remain substantially the same and that in typologies such as Tabula or others these buildings are classified in the same category, year of construction was selected as a primary parameter.

2.4.1 Estimating window wall ratio (WWR) of reference building

Window size for each reference buildings with reference to year of construction was estimated from DPE data collected by ADEME [5]. First data for Nantes was filtered out and then using the following expression estimated the ratio of window openings in percentage.

WWR=(['surface_baies_orientees_nord']+['surface_baies_orientees_est_ouest']+ ['surface_baies_orientees_sud']) / (['surface_parois_verticales_opaques_deperditives'] +['surface_baies_orientees_nord'] + ['surface_baies_orientees_est_ouest'] + ['surface_baies_orientees_sud'])

After calculations, the WWR of buildings were plotted against the ranges of year of construction for Tabula, as depicted in Figure 2-19. From this figure, it is clear that WWR is mostly hovering between 5 and 20% for all buildings. For better representation and characterization of window size and location, all identified reference buildings, shown in Figure 2-17, were either visited on site or visualized through Google Street view.

[5] https://data.ademe.fr/datasets/dpe-44

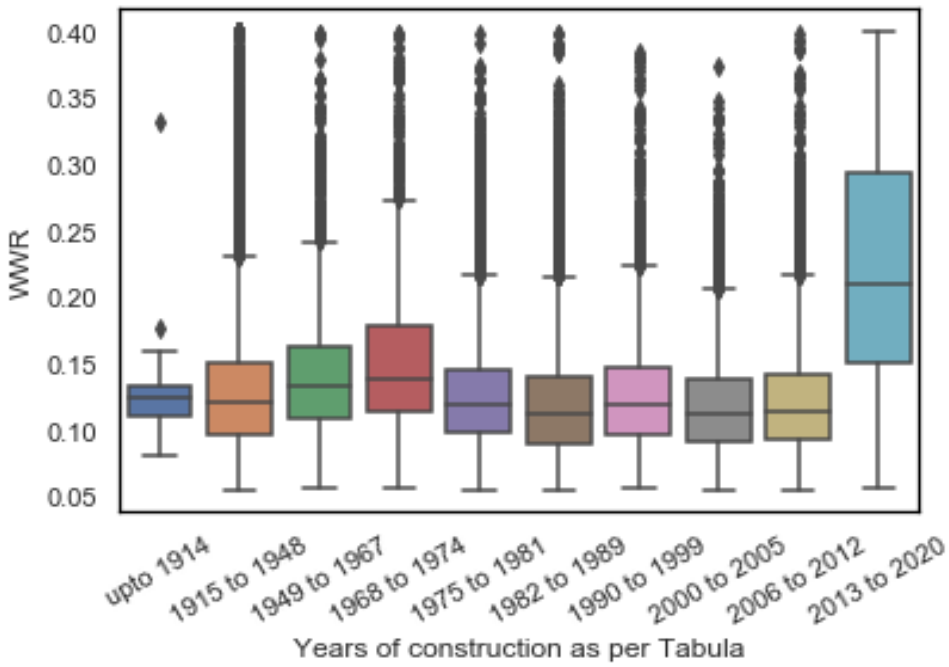


Figure 2-19 : WWR of residential buildings as a function of year of construction

2.4.2 Estimating U-value of envelop elements

A preliminary estimation of envelope's thermo-physical properties for each cluster as a function of year of construction was performed based on data presented by (Civel and Elbeze, 2016). The authors performed a description of residential buildings across France to determine their average U-value measure in [$W.m^{-2}.K^{-1}$]). A summary of their findings is presented in Table 2-3.

Table 2-3 : Initial thermo-physical properties of residential buildings' envelopes

Construction date	<1974	74-81	82-89	90-2000	2001-2005	2006-2014
U-walls	2.5	1	0.8	0.5	0.47	0.36
U-windows	4	3	3	3	2.3	2.1
U-roof	2.5	0.5	0.32	0.26	0.25	0.2
U-floor	1.2	1.2	0.74	0.5	0.36	0.27

(Civel and Elbeze 2016)

Since the data in Table 2-3 are estimated from a small sample and does not give any specific information for building constructed before 1974, it is necessary to find other sources to enrich selected typical buildings.

In the Tabula data table for France, the table below shows the percentage of buildings refurbished up until the end of 2013 for two categories of buildings: single-family houses (SFH) and Multiple-family houses (MFH).

Table 2-4 : Percentage of buildings refurbished until the end of 2013

Percentage of buildings refurbished (with improved thermal protection) [%]				
Building classes	SFH I	SFH II	MFH I	MFH II
Construction date	until 1975	1975-2000	until 1975	1975-2000
walls	37	88	19	52
roofs / upper floor ceilings	62	90	25	65
basement / cellar ceiling	12	42	10	30
windows*	35	75	23	57

Data from Table 2-4 clearly indicates that when it comes to performance evaluation of buildings, for both historical and new buildings, it is necessary to take into consideration thermo-physical properties for both a retrofitted and non-retrofitted scenario. Table 2-5 presents a summary of building thermo-physical properties for France from Tabula detailing the upper and lower limits of U-values for different elements of building envelope.

Table 2-5 : Thermo-physical properties of Tabula buildings in France

Tabula buildings For France		**Initial state U-value [$W.m^{-2}.K^{-1}$]**			**After high performing renovation [$W.m^{-2}.K^{-1}$]**		
		Roofs/ upper floor ceilings	**Exterior walls**	**windows**	**Roofs/ upper floor ceilings**	**Exterior walls**	**windows**
Single family houses	Up to 1914	1.35/1.11	1.7	4.8	0.2/0.10	0.24	1
	1915 to 1948	1.35/2.4	1.8	4.8	0.2	0.19	1
	1949 to 1967	2.42	1.8	2.6	0.1	0.19	1
	1968 to 1974	1.35	2.4	2.6	0.2	0.19	1
	1975 to 1981	0.57/0.76	0.61	2.8	0.1	0.19	1
	1982 to 1989	0.32/1.35	0.42	2.6	0.1	0.24	1
	1990 to 1999	0.23	0.36	2.6	0.1	0.19	1
	2000 to 2005	0.19/0.26	0.33	1.8	0.14	0.19	1
	2006 to 2012	0.24	0.34	1.6	0.37	0.16	1
	2013 to 2020	0.17	0.21	1.4	0.13	0.12	0.8
Individual semi-detached houses	Up to 1914	1.35/1.11	1.7	4.8	0.2/0.1	0.24	1
	1915 to 1948	2.42/3.6	2.1	4.8	0.1	0.24	1
	1949 to 1967	1.45	2.6	2.6	0.1	0.19	1
	1968 to 1974	3	2.4	2.6	0.23	0.19	1
	1975 to 1981	0.49	0.61/2.4	1.4	0.1	0.19	1
	1982 to 1989	0.32	0.47	2.6	0.1	0.19	1
	1990 to 1999	0.23	0.36	1.8	0.1	0.19	1
	2000 to 2005	0.19	0.33	1.6	-	0.19	1
	2006 to 2012	0.19	0.29	1.6	0.37	0.15	1
	2013 to 2020	0.12	0.25	1.4	0.1	0.12	0.8

Collective houses (lodgments<10)	Up to 1914	1.35/1.30	1.7	2.6	0.23/0.20	0.24	1
	1915 to 1948	2.42	2.3	2.8	0.1	0.24	1
	1949 to 1967	2.42	3	4.8	0.1	0.19	1
	1968 to 1974	0.76	0.78	5.6	0.23	0.19	1
	1975 to 1981	0.49	0.61	2.8	0.1	0.19	1
	1982 to 1989	0.62	0.36	2.8	0.23	0.19	1
	1990 to 1999	0.43	0.33	2.6	0.23/0.18	0.19	1
	2000 to 2005	0.28	0.33	1.6	0.23	0.19	1
	2006 to 2012	0.19	0.3	1.6	0.23	0.19	1
	2013 to 2020	0.28/0.15	0.18/0.19	1.4	0.12/0.1	0.11/0.12	0.8
Collective houses (lodgments>10)	Up to 1914	2.4/1.3	1.8	4.8	0.2/0.1	0.23	1
	1915 to 1948	3.2	1.7	2.8	0.23	0.24	1
	1949 to 1967	3.2	3	2.6	0.23	0.27	1
	1968 to 1974	3.2	0.78	2.6	0.23	0.19	1
	1975 to 1981	0.76	0.79	2.8	0.23	0.19	1
	1982 to 1989	0.62	0.78	2.6	0.23	0.19	1
	1990 to 1999	0.43	0.36	3.3	0.23	0.19	1
	2000 to 2005	0.28/0.24	0.33	1.6	0.23/0.14	-	1
	2006 to 2012	0.28	0.3	1.6	0.23	0.15	1
	2013 to 2020	0.12	0.24/0.33	1.4	0.1	0.11	0.8

2.4.3 Dividing buildings into zones

A review by (Shin and Haberl 2019), concludes that thermal zoning has a considerable impact on accuracy of thermal and energetic assessment of buildings at design phase or/and evaluation phases. The authors in their review of thermal zoning methods, discuss three major ways of dividing a buildings into zones: (1) zoning of building based on guidelines, such as ASHRAE Standard 90.1-2016, and CIBSE Applications Manual AM11, IBPSA guidelines on zoning; (2) automatic zoning system of buildings with the help of software, such as Autodesk, which uses the method proposed by (Smith, Bernhardt, and Jezyk 2011), or EASL optimizer proposed by (Yi 2016); (3) thermal zoning based on designer's subjective judgement.

Most of the main strategies of zoning analyzed in the review by (Shin and Haberl 2019) are based on conventional zoning strategy of one core zone and perimeter thermal zoning. (Garreau 2021; Garreau et al. 2021) also use this conventional strategy in her thesis and research article to create building zones.

In this study however, selection and identification of reference buildings using unsupervised machine learning allowed us to find them on the site and determine thermal zones based on the position of windows in each building. The approach we have adopted is more realistic and based on the actual layout of each reference building.

2.4.4 Occupancy

Density of occupants was set according to EN 16798-1 to 42.5 m^2/person and 28.3 m^2/person in single-family houses and multi-family houses respectively. Occupancy usage schedule for all reference buildings were assumed the same, and is presented in Figure 2-20.

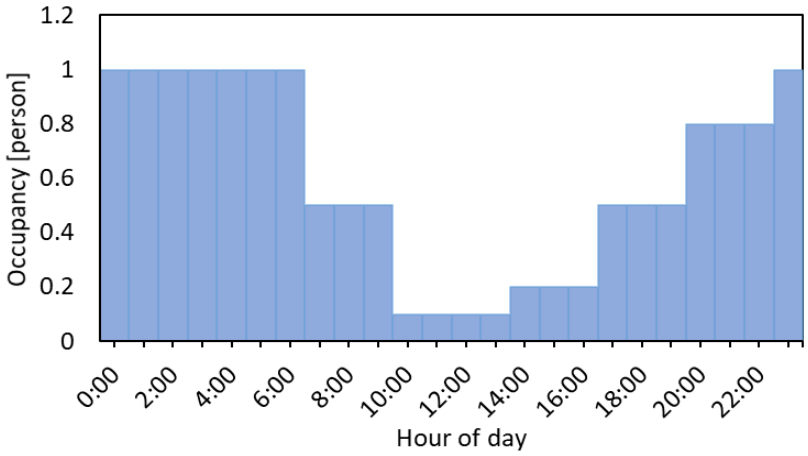


Figure 2-20 : Occupancy usage schedule according EN 16798-1

2.4.5 External shading

Windows of reference buildings are equipped with external manual blinders/shutters allowing occupants to control direct solar radiation intake into the zones. In summer months, during the day, occupants are expected to frequently use external shadings as a passive approach to control solar gains. We did not find a concrete source in literature to show when and how occupants operate window shutters in Nantes, our case study city. Therefore, after a discussion it was

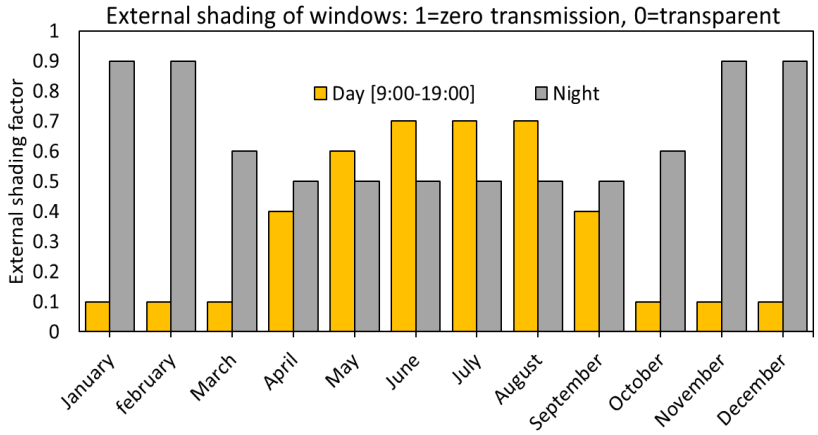


Figure 2-21 : Modelled external shading schedule of occupants

decided to develop theoretical scenarios of how a typical user would use windows during summer and winter months. Decisions to open or close an external window shading in winter months are assumed to be driven by safety concerns and outdoor environmental conditions (temperature, rain, etc.). For that, it was assumed that during the day, in winter months, occupants would fully open external shading and close it during the night (see Figure 2-21).

In summer months, the decisions on how to use windows are assumed to be driven, in addition to those mentioned above, by solar protection. Orientation of window plays an important role on how a user would operate/control external shading. Typically, a user would keep the external shading of a window oriented east during the night and early morning to limit direct solar irradiance into building before they wake up. Similarly, a user would keep external shading of a window oriented west in the afternoon and evening. Windows oriented south are exposed to direct solar irradiance in the middle of day; therefore, users would probably use external shading to limit exposure to direct sunlight at that time of the day. Figure 2-22 shows how an occupant would operate the external shading of windows. The values plotted in this graph are based on the experience and discussion of the author with other experts in this field. No reliable data to prove this assumption has been found by the author.

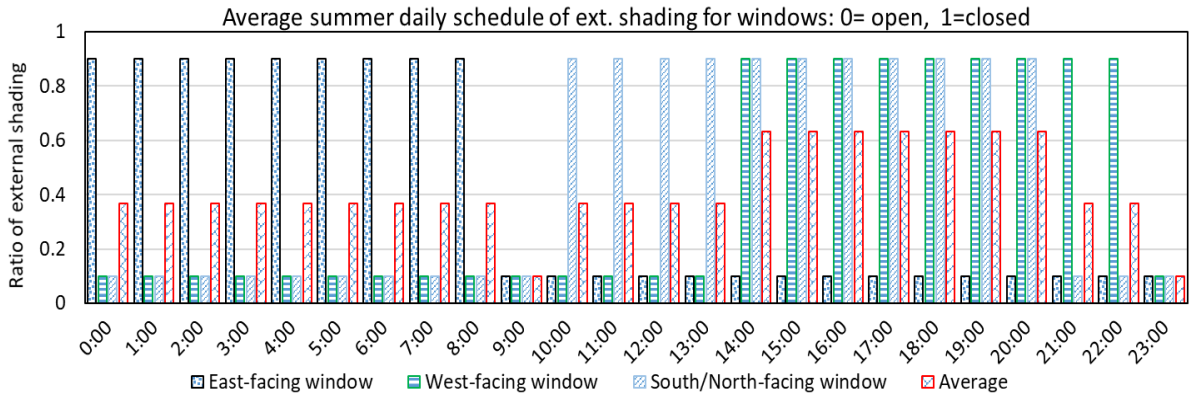


Figure 2-22 : Daily schedule of external shading operation of windows during summer

2.4.6 Air inflow rate

Occupants are assumed to be conscious of the local environment and would open and close windows as can be seen in Figure 2-23.The figure presents an estimation of air inflow volume into the building zones as a function of season, number and status of windows. Air inflow into a naturally ventilated building is highly correlated to the status and openness ratio of windows. As can be seen in Figure 2-23, when there is no opening in the zone or when windows are closed like in winter, a minimum of 0.7 [ACH] which is approximately 0.4 [$m^3h^{-1}m^{-2}$] enters to the zone but when there is one window and two windows in a zone the average air change rate increases to 1.3[ACH] and 1.9 [ACH], respectively.

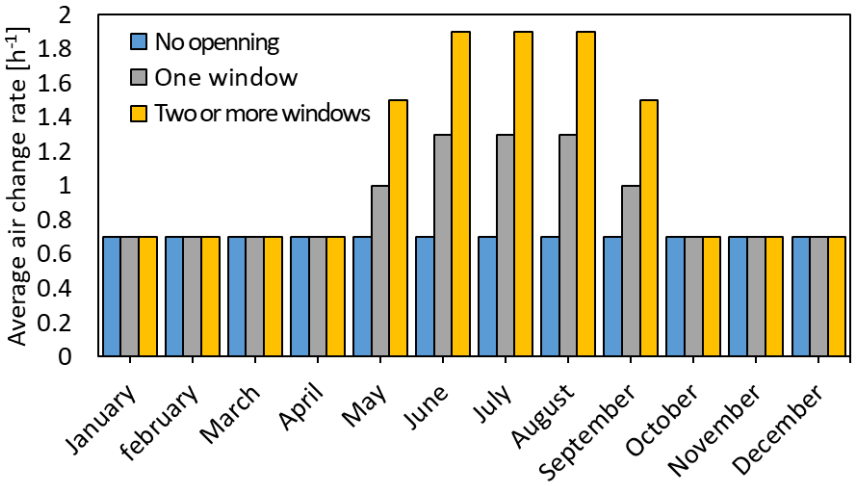


Figure 2-23 : Average air change rate in different thermal zones

Appendix 2-6 further illustrates how the values presented in Figure 2-23, were estimated/calculated.

Both, external shading and window operation for air inflow are passive strategies that are proved to have considerable impact on indoor overheating control. They are usually controlled by the decision and ability of users to operate them. Therefore, the author thinks it is more realistic and practical to model these two parameters as a function of occupant profiles. To do so, three types of occupant profiles are described that could represent behavior of user across the city:

1- Highly conscious of external environment
2- Medium conscious user
3- Poor conscious user

An adaptive user is an occupant who is considered to be highly conscious of his/her environment and he/she operates windows and external shading when there is not direct solar radiation into the zone to improve comfort. **A medium conscious user** is referred to an occupant that operates windows for air inflow but does not use external shading to regulate solar intake into the zone, due to absence of external shading equipment or some other reasons. **A poor conscious user** is referred to someone who operates neither windows nor external shading to regulate air inflow and/or solar intake, due to absence of necessary tools, health conditions or external constraints, such as noise. Separate profiles were created for each user type and were used in building simulations.

2.4.7 3D models of buildings and thermal assessment

After characterization, a 3D model of each building was created and imported to TRNYS v.17 for thermal and energetic evaluations. Models of identified reference buildings are presented in Figure 2-24.

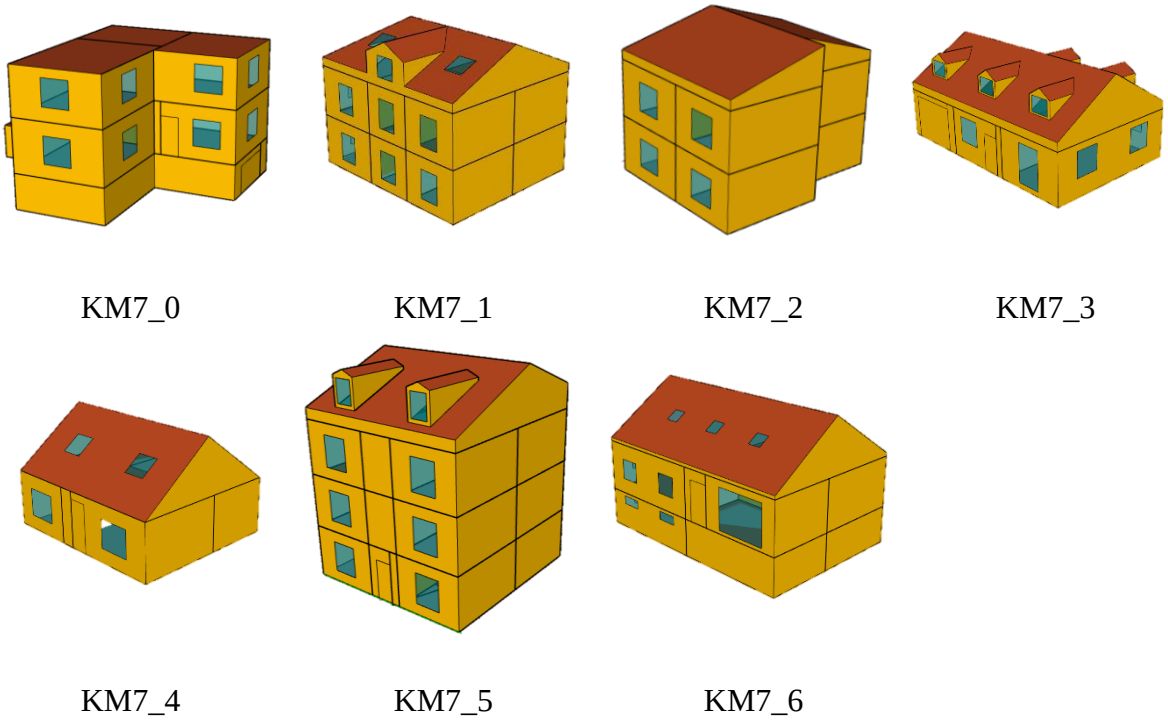


Figure 2-24 :3D models of selected reference buildings

Perhaps a better way to present reference buildings identified with this method is like in Figure 2-15, because these buildings were selected based on their UHI effect parameters as well as their individual characteristics.

2.5 Summary

As was stated in chapter 1, the objective of this research work is to pave the way for the development of methodology that would allow practitioners to build indoor overheating vulnerability map at the city scale. However, in practice, simulations or any sort of study at such a scale requires analysis of large number of buildings taking into consideration many characteristics. Detailed dynamic simulations tools are expensive to be used at city scale for indoor overheating evaluation tasks. A viable solution in such case is to develop a set of archetypal building types that could represent most of the building stock and perform necessary scenario analysis or simulations on them. This chapter of the manuscript is dedicated to literature review and description of cluster analysis to identify clusters of buildings and represent each one with one real building.

Following is a summary of the tasks and results covered in this chapter:

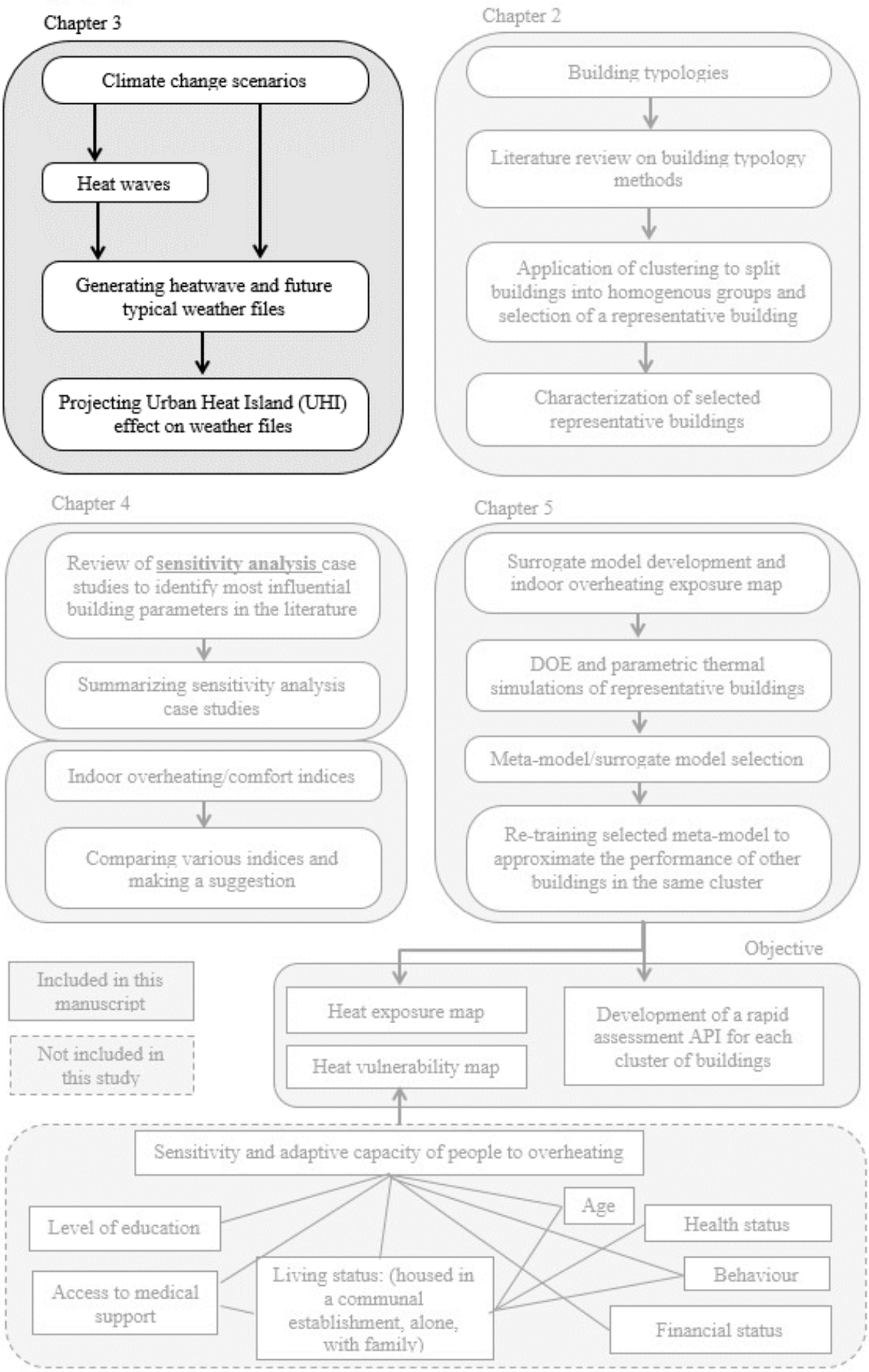
Main issues covered:

- A detailed literature review on building archetypes development methods was carried out and their advantages, disadvantages were discussed.
- To minimize the biases and a priori assumptions of the modeler, it was decided to use cluster analysis (unsupervised machine learning) to group residential buildings in the case study city, Nantes, into different groups.
- Input parameters for cluster analysis were chosen in line with objective of the research thesis in such a way that takes also into consideration the urban environment (possible UHI effect).
- Additional information necessary for characterization of reference buildings were investigated.

Results:

- There is no better or worse way to group buildings into clusters or groups, it depends on the objective and available input information. Practitioners/researchers need to test multiple methods for each case to identify which one suits their case better.
- K-Means clustering technique was chosen, after a comparative analysis of various methods for cluster analysis.
- Centroid of K-Means is calculated from the average parameters of all elements in the cluster and it is a fictive point. Therefore, the closest real entry (building) to the fictive centroid was identified and selected as the representative building of the cluster.

Overall, this chapter described the workflow for identification of representative buildings, discussed the characterization stage and made the representative buildings ready for parametric simulations. The next step is to prepare future weather files, and project the influence of UHI effect on the weather file of each representative building.


Chapter 3
Climate change scenarios
Heat waves
Generating heatwave and future typical weather files
Projecting Urban Heat Island (UHI) effect on weather files
Chapter 2
Building typologies
Literature review on building typology methods
Application of clustering to split buildings into homogenous groups and selection of a representative building
Characterization of selected representative buildings
Chapter 4
Review of sensitivity analysis case studies to identify most influential building parameters in the literature
Summarizing sensitivity analysis case studies
Indoor overheating/comfort indices
Comparing various indices and making a suggestion
Chapter 5
Surrogate model development and indoor overheating exposure map
DOE and parametric thermal simulations of representative buildings
Meta-model/surrogate model selection
Re-training selected meta-model to approximate the performance of other buildings in the same cluster
Objective
Heat exposure map
Heat vulnerability map
Development of a rapid assessment API for each cluster of buildings
Included in this manuscript
Not included in this study
Sensitivity and adaptive capacity of people to overheating
Level of education
Access to medical support
Living status: (housed in a communal establishment, alone, with family)
Age
Health status
Behaviour
Financial status

Chapter 3 : Climate change data, heatwave, urban heat island weather data

3.1 State of the art

3.1.1 Future climate data

The effects of climate change are already noticeable and it requires addressing thermal discomfort of buildings associated with it in a way that meets current and future scenario needs. In addition, currently, buildings are designed with typical climate conditions that are based on historical climate records. They naturally do not perform well under extreme weather conditions. This point highlights the need to direct the research towards creation of future and extreme weather files to help practitioners to design buildings that not only meet typical heating demand requirements, but also are able to perform well under extreme conditions, including heatwaves (Hosseini, Bigtashi, and Lee 2021). One of the best ways to make sure buildings meet such requirements is to use building performance simulation (BPS) tools at different stages of development.

BPS is also called as building simulation or building energy and thermal modelling in the literature. It refers to the use of computer software, mainly to predict/evaluate the energy demand/consumption of buildings. Many of these simulation tools are now capable of modelling thermal comfort, daylight performance, indoor air quality, environmental footprint, and many more of building indicators (Altan, Padovani, and Hashemi 2016). These tools are able to simulate buildings at different stages of development: concept design, schematic design, design development, construction documents. They can also help practitioner to evaluate building performance of existing buildings.

Three methods of thermal and energetic simulations are commonly used nowadays for building performance evaluations: static BPS, semi-dynamic BPS, and dynamic BPS.

Static BPS: this method is used in simplified programs to evaluate how buildings perform in a stationary regime using a limited number of building factors. This method only partially simulates building performance. Mainly because it does not consider periodic changes of temperature. Accuracy of input data has a fundamental importance to obtain results. This approach is popular in building energy certification. An example of static thermal simulation tool is DOCETpro 2010, and CadSoft Thermix for Energy performance certificate (DPE) calculations.

Semi-dynamic: this method of BPS employs dynamic simulation to account for thermal inertia but it requires simplified input data to represent climatic conditions and building description. An example of Semi-dynamic simulation tool is Sketch Design Software.

Dynamic BPS: this method analyses in details thermo-physical properties of envelop, thermal inertia, periodic variability of outside weather conditions, solar radiation, natural ventilation, heat gains from occupants, etc. An example of dynamic BPS is Trnsys type 56.

Of the three, dynamic method is considered more appropriate in building thermal comfort and energy demand assessment. This method requires at least hourly weather data that contains temperature, humidity, radiation, wind, atmospheric pressure, etc. Depending on the objective of dynamic BPS, two distinctive weather data types could be used: synthetized and observed data. The latter is often used in performance monitoring phase and is collected from weather stations or by in-situ measurements. The former is more frequently used in the design phase, and is synthetically generated from climate normals. WMO defines climate normals as a period that covers at least 30 years of data.

For evaluation of buildings' climate resilience, future weather data are required. These data are based on future emission scenarios and projections produced using climate models.

Emission scenarios are used as input for General Circulations Models that also referred as Global Climate Models (GCMs). Researchers have utilized GCMs to examine the impact of climate change in a variety of sectors. For future climate scenarios, GCMs mathematically simulate atmospheric, oceanic, and biotic interactions and integrate them with radiative forcing scenarios. The models are made up of grid cells created by latitude and longitudinal lines, which are used to generate meteorological data. Although these models aid in the consideration of climate change, their output data cannot be utilized directly for building energy modelling (Hosseini, Bigtashi, and Lee 2021).

GCMs cover entire surface of the globe and their spatial resolution is coarse, typically between 150 to 600 km (P.Tootkaboni et al. 2021). Application of given GCMs for build thermal evaluation requires downscaling to a finer spatial and temporal resolution to consider regional and local scale estimates of climate variability and change. As can be seen in Figure 3-1, there are two main approaches to downscale GCMs: Dynamical downscaling (DDS) and empirical-statistical downscaling (ESD).

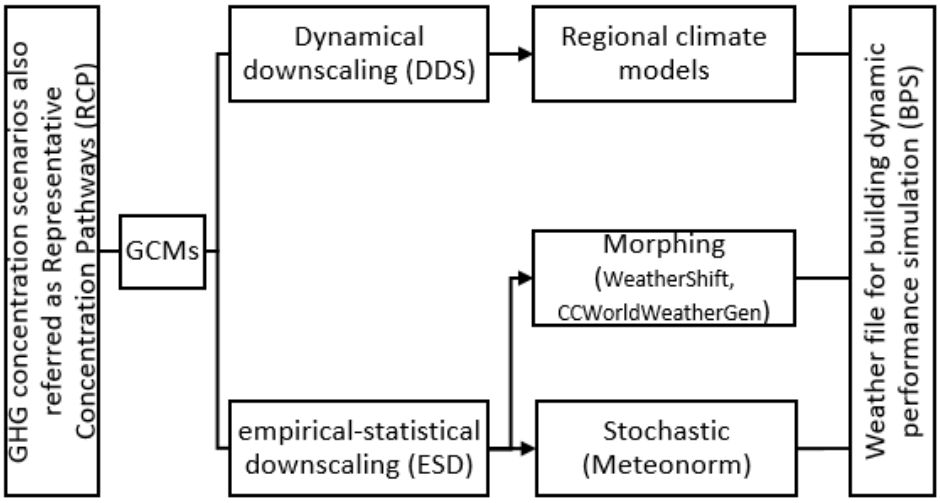


Figure 3-1 : Main GCM to RCM downscaling approaches

DDS and ESD stand on two distinctive philosophies: DDS relies on climate data that are based on our knowledge of physical processes (solving equations for humidity, temperature, local wind, etc.) and ESD makes use of information obtained from the statistical analysis (e.g. regression relationships) of previous observed climate data.

(Erlandsen et al. 2020; Moazami et al. 2019) in their studies have also discussed a hybrid approach in which the results of dynamically downscaled GCMs, also referred to as Regional climate model (RCM), stored at a coarse resolution undergo further downscaling using statistical approach.

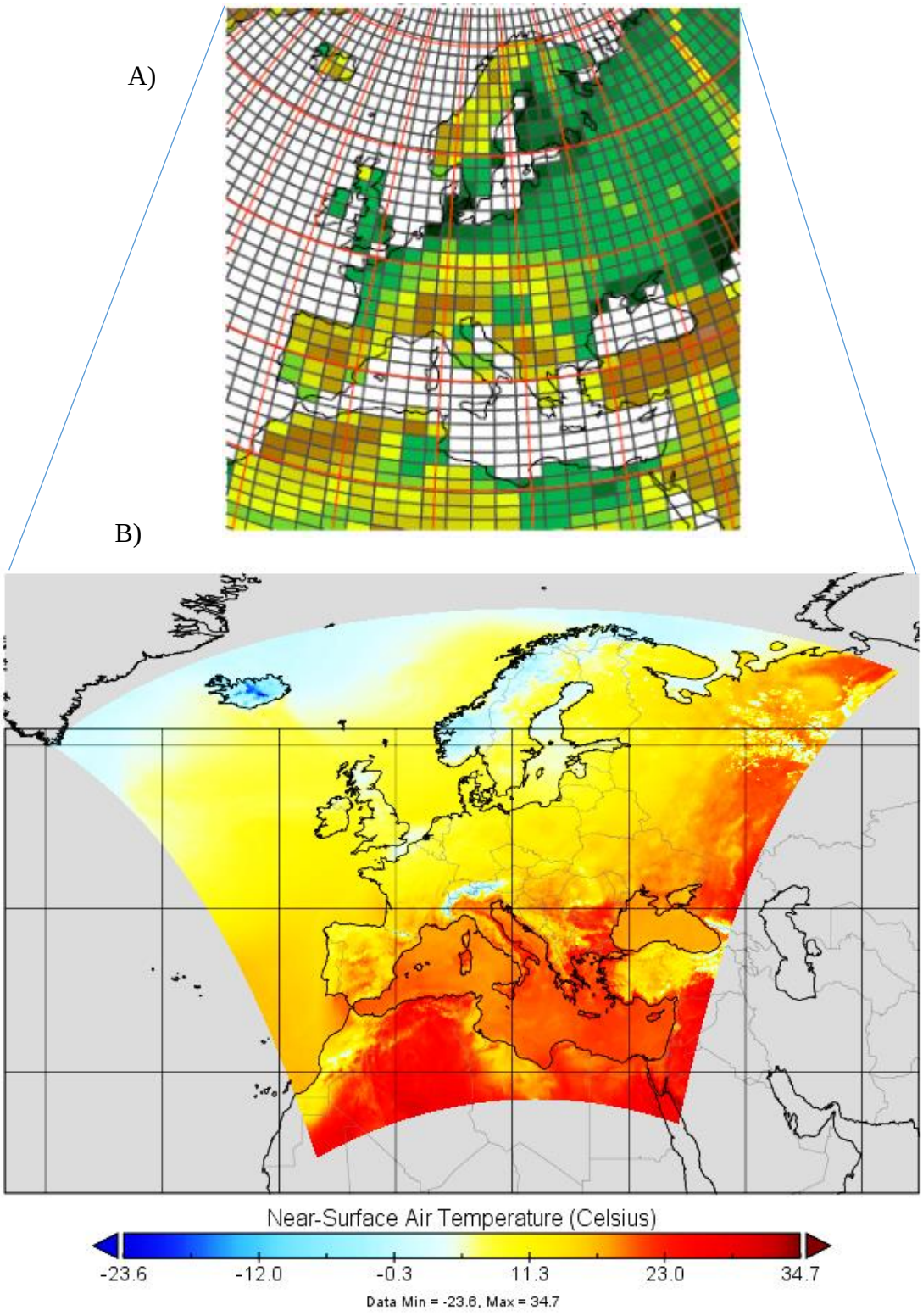


Figure 3-2 : A), IPSL-CM6A-LR at 150 km resolution (elevation); B), SMHI RCM dynamically downscaled from IPSL GCM to a resolution of 12.5 km over Europe in CORDEX.

Figure 3-2 shows schematic representation of the DDS technique. As can be seen in this figure, GCMs do not capture variations in vegetation, complex topographies and littoral zones that are located inside the rectangles, which are important aspects of the physical response that governs the signal of regional/local climate change.

In contrast to GCMs, RCMs account for vegetation, complex topographies, and littoral zone pretty well. RCMs are also able to provide a detailed description of extreme weather conditions, including statistical data on extreme weather events (Rummukainen 2010).

Of course, the DDS has certain drawbacks too: it is computationally expensive, and highly sensitive to boundary conditions. Boundary conditions of RCMs are provided by the GCM that they downscale from. This means if the GCM contains errors, RCM will contain them too.

Furthermore, RCMs (like GCMs) contain semi-empirical settings like for convection: the assumption here is that these semi-empirical settings will be valid for future climate scenarios as well. Another constraint of RCM is the spatial resolution of models. For current generation of RCMs, it is around 1 km and this resolution requires phenomenal processing power for calculations.

RCMs generally need to be validated against observational datasets to assess the ability of the model to reproduce current climatic conditions. This then makes it possible to define its deficiencies, resulting from the various modelling assumptions and their related uncertainties.

One of the major sources of uncertainty in RCMs comes from the large number of physical processes parameterized in the climate model and many of these parameters are uncertain. Since uncertain parameters are responsible for a large part of the modelling errors, the parameter uncertainty is usually contained by calibration or tuning methods (bias correction). This tuning process is one of the aspects of work that requires highly specialized technical skills to be able to ensure efficient implementation and operation of the RCMs.

Theory of bias correction for DDS method (Figure 3-3):

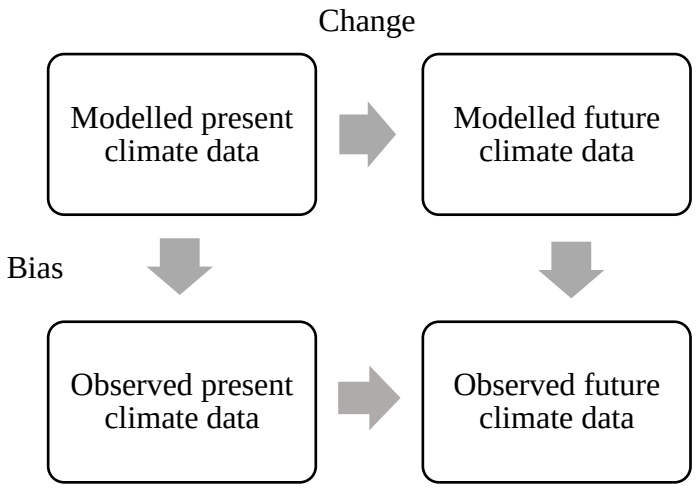


Figure 3-3 : Schematic representation of bias adjustment theory

Two assumptions of this theory are: (1) bias of current and future climate model is equal, (2) climate model results for present climate is correct.

It is also important to note that Bias function (difference between present modelled and observed climate data) can be a linear function or a scale transfer function that has mean, variance and shape (Dierickx 2019).

For practitioners, if the objective of use of weather data is the relative change in the future compared to present, bias is not a big problem. It is however important, if practitioners are dealing with water availability in the future. Bias in relative humidity level can also have a huge impact in future climate information prediction.

In this work however, we did not bias adjust our climate models because in the literature review, we came across arguments that showed that local bias correction is still an open question for researchers.

Two articles by *Machard, Anaïs et al.* (Machard et al. 2020) , and *Maraun, Douglas. on 'Bias Correcting Climate Change Simulations – critical review* (Maraun 2016) argue that current local bias correction techniques are resource intensive (require weather data collection over an extended period) and they rely on the assumption that bias correction factors (relative or absolute value) will remain the same in the future for all climate variables (temperature, humidity ratio, GHI, wind speed, etc.). However, there is no evidence to support this assumption, and it only adds a supplementary uncertainty in the climate model.

Figure 3-4 shows schematic representation of the **ESD technique**. As can be seen in this figure, ESD is a two-step process.

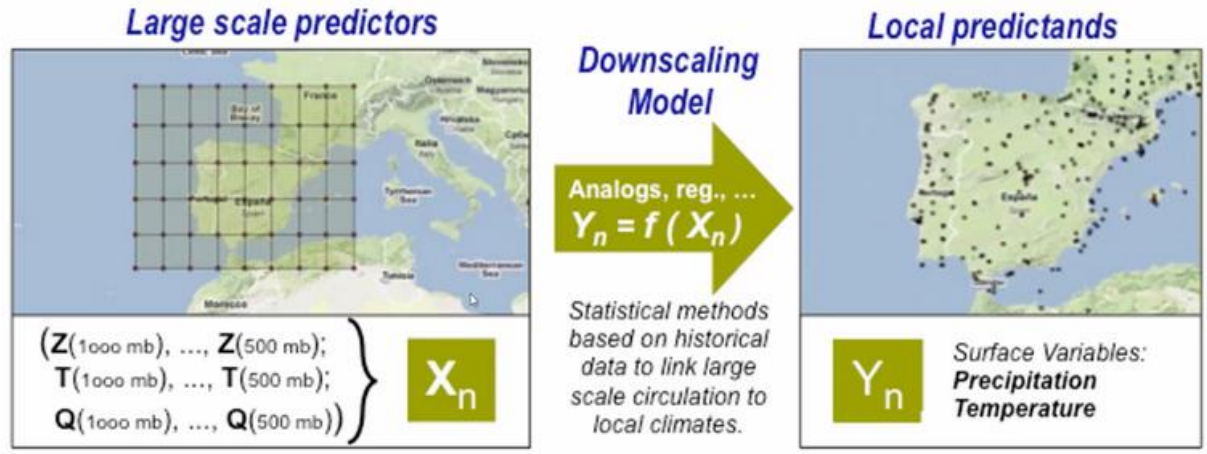


Figure 3-4 : Schematic representation of statistical downscaling (Dierickx 2019)

First step is to derive statistical relationship between observed small-scale variables and greater global climate model scale variables. The second step applies the derived statistical relationships to estimate future climate at the smaller scale based on larger variables from future climate GCMs. Unlike DDS that downscales all climate parameters, ESD only downscales primary climate variables and that is one of the reasons why it is computationally not as expensive.

There are two major downscaling approaches in ESD: Stochastic downscaling and morphing method (P.Tootkaboni et al. 2021).

A well-known tool that uses **stochastic ESD** approach is Meteonorm. In this approach, synthetic weather time series data is created using empirically derived statistics. It is

computationally cheap, but it requires large amount of "training" datasets to give the model proper statistics and handle unknown coefficients of the model. (Belcher, Hacker, and Powell 2005) argue that the weather series it generates may not be always consistent meteorologically.

Major tools using **morphing ESD** approach are CCWorldWeatherGen and WeatherShift. In this approach, the key assumption is that present weather data is a reliable baseline. It then adjusts present design weather data by the changes to climate forecasts of GCMs and RCMs. This adjustment is done in three ways: shifting, scaling, or combination of both (Machard et al. 2020).

Several studies have been conducted on the relative difference of DDS and ESD on climate impact assessments of buildings. (P.Tootkaboni et al. 2021) in their paper on comparative analysis of different future weather data for building energy simulations, compared weather files from three tools that are based on ESD (WeatherShift, Meteonorm, and CCWorldWeatherGen) with one DDS future typical meteorological year. Their results shows that all ESD weather files have relatively similar operation in predicting thermal comfort and energy consumption in buildings in comparison to DDS weather file. Their paper also states that ESD method, regardless of how it is used can provide sufficient information to perform comparative analysis on long-term variations in energy consumption of buildings, but existing inconsistency within the method can lead to significant prediction error. Under such conditions, they found DDS method more reliable when the objective of the study is to investigate and communicate resilience of buildings to future climate conditions. (Ramon et al. 2019) in their paper state that ESD method is more suited to investigate average energy performance in future climate realization but less suited to assess extreme conditions. DDS, on the other hand, can be used for both average and extreme assessment purposes. (Moazami et al. 2019) in their study on the impact of future weather data types on building energy performance concluded that weather files generated using DDS that take into account both typical and extreme climatic conditions are most reliable to evaluate energy robustness in the context of future climate uncertainties.

3.1.2 Heatwave weather data

One of the modern trends in climate research is the study of abnormal (extremely hot or extremely cold) weather events. Such extreme surface temperature variations are formed fairly in short time intervals that are in the form of waves. It increases, reaches to a peak and then starts to descend creating the shape of a wave. These heat or cold waves are characterized by a peak temperature, duration, intensity, and spatial dimension. All these characteristics listed above play an important role in vulnerability of people exposed to heatwave.

Integration of heatwave weather data in BPS is not usual among practitioners so far. This is mainly because of two reasons:

First, absence of sufficient data from historical heatwaves in many locations across the world. Second, absence of one globally accepted definition for heatwave or in other words, presence of multiple definitions for it (M. Zhang et al. 2022).

This is due to the fact that the impact of a heat wave is influenced by a variety of factors in different regions: environment, socio-demographic features of community, and population

adaptation capacity. A reasonable definition of heatwave at either local or regional level is essential for analysing its health implications and implementing effective heat warning systems at various levels.

Classical definition of it, however, states that a heat wave is commonly assessed in comparison to a specific location's average weather and normal temperature ranges of a season. The problem with this classical definition is that the baseline is not specified. As the global temperature increases, the baseline also shifts and the intensity of past extreme weather events would seem smaller over time.

There is also no consent on the type of heatwave data that BPS practitioners need to use: some suggest using historical measured heatwaves, others prefer modelled heatwave data from various climate models, but majority do not use it at all (Machard et al. 2020).

3.1.2.1 Heatwave tools and indices

Based on classical definition, an R package developed by (W. Schlegel and J. Smit 2018) provides a comprehensive analysis that detects, and visualizes marine heatwaves. (M. Zhang et al. 2022) argue that the R package is not efficient when applied to large gridded data.

Another tool that allows extracting and processing heatwaves data of any location efficiently is Global Heat Wave and Warm-spell Data Record and Analysis (GHWR) MATLAB toolbox. This tool contains various definitions for heatwaves, and is also capable of detecting/extracting multiple characteristics of heatwave (Raei et al. 2018).

(M. Zhang et al. 2022) in their paper presented a Google Earth Engine-based toolkit named *heat wave tracker* (HWT). This tool is based on cloud computation engine, and it can be used to dynamically visualize, extract, and process complex heatwave from multi-climate datasets. This tool has been used to exploit climate datasets and develop nine heatwave datasets across Australia. The tool works well at continental scale and according to the authors, it is applicable anywhere in the world.

According to the (M. Zhang et al. 2022), outputs of these tools are extremely sensitive to the initial definition of heatwave index adopted. In the following, a brief list of definitions/indices for heatwaves is illustrated.

3.1.2.1.1 Constant temperature threshold approach

A heatwave, according to this definition, occurs when temperatures consistently remains above a certain threshold for a specific period of time (Hertel et al. 2009; Wagner 2018). Extended period of time is referred to 2, 3, or more, consecutive days when maximum, minimum or average temperature is above a specific threshold.

3.1.2.1.2 Probability distribution function (PDF)-derived temperature threshold

Temperature thresholds of heatwaves with this approach are set in accordance to local climate conditions. For instance, temperature events during which the observations exceed the 90^{th} or 95^{th} percentile threshold for multiple consecutive days is defined as heatwave threshold temperature (Raei et al. 2018).

3.1.2.1.3 Average temperature + 5 degree C

As the name suggests, this approach is based on spatially localized temperature threshold that separates heatwave days from typical ones. It only considers long-term average daily temperature to define threshold. In other words, it is when temperature is +5 degrees above the long-term average. (Raei et al. 2018) argues that this approach is an outdated way of measuring heatwaves.

3.1.2.1.4 Upper tail percentile

In this technic, heatwave temperature threshold is both geographically and temporally localized. Thresholds are defined as an upper tail (85th, 95th) percentile thresholds of PDF generated from long-term daily temperatures over a window of 15 (21) days, centered around the calendar day of interest for each calendar day (Machard et al. 2020; Raei et al. 2018; Stefanon, D'Andrea, and Drobinski 2012).

3.1.2.1.5 Summer-derived threshold approach

This approach is theoretically similar to PDF-derived temperature threshold method but in contrast to that uses only long-term summer daily temperatures. According to (Raei et al. 2018) this definition of heatwave is more appropriate for studies that concentrate on extreme heat and its impact on mortality, morbidity of people or yield level of agricultural products.

3.1.2.1.6 Excess heat factor (EHF)

This method uses the *"excess heat"* and *"heat stress"* measures to build the index. *"Excess heat"* is used to characterize the local long-term temperature anomalies in comparison to typical climate of the region, and *"heat stress"* illustrates the short-term excessive temperature variation to account for thermal acclimation (Perkins, Alexander, and Nairn 2012; Raei et al. 2018). This index can effectively describe the impacts of heatwaves on human mortality and morbidity.

3.1.2.1.7 Standardized heat index (SHI)

This method was proposed by (Raei et al. 2018) with the introduction of heatwave detection toolbox, GHWR. This approach is a probabilistic and generalized index that illustrates the heatwaves as a function of their occurrence probability for long-term weather data records. It is both spatially and temporally localized. According to the author, this index can be used to detect cold-spells (cold waves) as well as warm-spells (heatwaves) in the climate datasets. This index also allows practitioners/researchers to select or tune thresholds so that it becomes more suitable for their studies.

In this study, a temperature-based index was selected to measure the characteristics of heatwave in weather files, because it was previously employed by (Wagner 2018) for France. The details of the index is described in section 3.2.4 of this chapter and a python script was developed to calculate the intensity, duration, and cumulative intensity of heatwave in each weather file.

3.1.3 Integration of UHI effect on weather data

Majority of tools for BPS simulations use weather data in Typical Meteorological Year (TMY2), EnergyPlus weather (EPW), DOE or other common formats. The weather information

contained in these weather files can come from various sources: could be synthetically generated for one point using ESD or DDS approaches, or obtained from weather stations.

One common point in these weather files is that they are extracted for a specific location, and under normal conditions, the urban climate condition (UHI effect) is not taken into account.

As discussed on chapter I, urban environmental conditions that influence UHI intensity can have significant impact on summer and winter indoor temperatures of buildings in cities. Therefore, projecting the dynamics of UHI effect on BPS is essential for appropriate indoor vulnerability and thermal comfort assessment. One way to solve this is by chaining an Urban Climate model (UCM) with building simulation tool. Following paragraphs will briefly describe UCMs that can project the UHI effect on weather files efficiently.

Various urban climatic models have been developed by researchers that capture the physical interaction of urban elements (e.g., buildings, trees, vegetation, human activities that generate heat). Each one of these UCMs are designed for a different spatial and temporal scale.

(Lauzet et al. 2019) performed a comprehensive review on techniques and tools that are used by practitioners/researchers to take into account local climate in BPS. The authors classified UCM models with two criteria: spatial scale and horizontal resolution of the mesh elements used in the model.

- City-scale models
    - MESO-NH & Town Energy Balance Model (TEB)
    - Weather Research and Forecast (WRF) model, and Building Effect Parametrization (BEP)
- District-scale models
    - Parametric models
        - Urban Weather Generator (UWG) & Canyon Air Temperature (CAT)
        - Canopy Interface Model (CIM)
    - Explicit models
        - ENVI-met
        - SOLENE-Microclimat
- Street Models
    - Town Energy Balance (TEB) for (canyon)
    - Zonal models

In a similar review, (Jänicke, Milošević, and Manavvi 2021) listed more models such as ADMS Temperature, Humidity model, advanced SkyHelios model, CFD models (ANSYS FLUENT, OpenFOAM), RayMan, SOLWEIG, TownScope, and UMEP that are used to model micro and meso-scale climate conditions. The authors argued that newer models are increasingly focusing on multi-scale models that are also able to bridge the gap between micro- and meso-scale models. These new models have the potential to produce better spatially consistent results. Current popular examples of these models include UMEP, and Integrated Multi-scale Environmental Urban Model (IMEUM). Another example of integrated multi-scale model is described by (Wong et al. 2021), where they coupled UCM into WRF model and chained the outputs of WRF to OpenFOAM which was further linked to EnergyPLus.

Among the listed tools and techniques above, The Urban Weather Generator (UWG) parametrically simulates the UHI effect on rural weather station data for a single point (Bueno et al. 2013). UWG is very helpful to take into consideration boundary conditions of building for BPS, where local/urban air temperature observations are not available.

Preferred technique to project the urban climate condition effect on BPS is highly influenced by the tool at the disposal of modeler, scale of study, temporal resolution, as well as the relevance to policy and decision-making.

(Allacker et al. 2019) presents a case on trade-off between data granularity of models and relevance to urban-scale decision/policy-making. According to the authors, data granularity is high at small-scale simulations and low at large scale. Relevance to decision/policy-making at city scale on the other hand, is low at small-scale simulations and high at large-scale simulations (Figure 3-5).

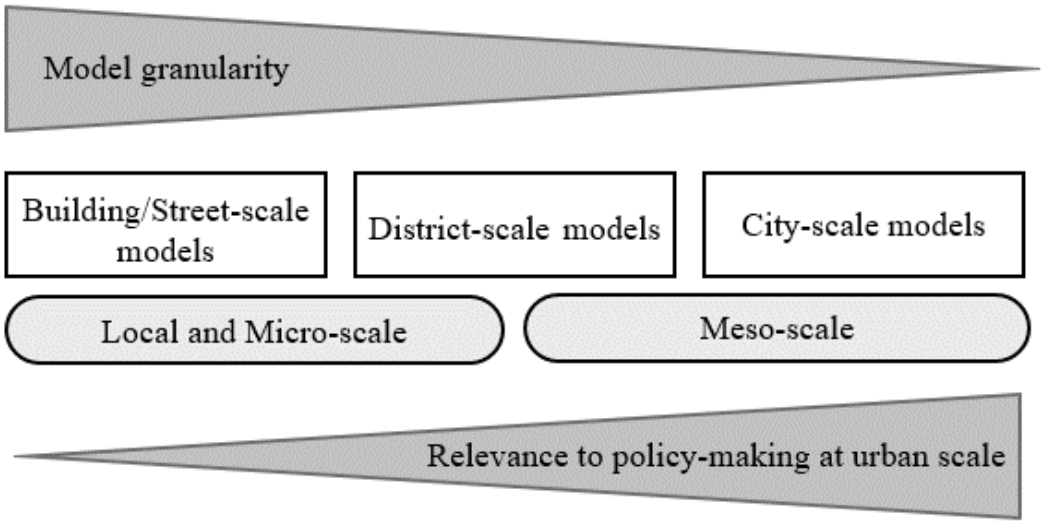


Figure 3-5 :Trade-off between model granularity and relevance to policy/decision-making

The knowledge and information coming from dynamic simulation models that are simulated at building scale can be very helpful at making decision at the level of a single building, but more difficult when the scope is larger, as it happens in this study that attempts to build a map that shows indoor vulnerability of people to overheating at city-scale.

In addition, urban planning policies and decisions aim to foster benefits for all at meso-scale/urban-scale (e.g., promoting energy consumption, reducing overheating vulnerability of building stock, cutting on GHG emissions), whereas, intervention resulted from those decisions are generally taken at micro-scale (e.g., improving insulation).

Developing an easy to use and interpret method to link urban climate, and BPS can be immensely helpful in such a context (Perera et al. 2018).

Considering all the issues discussed, in this study, the author decided to use Urban Weather Generator (UWG) method to project the influence of UHI effect on typical future weather data and observed weather data. It was selected because it provides a good balance between the granularity of study and relevance to policy making. It is computationally cheap and does not require extensive data collection from the site. Recent studies by (Boccalatte et al. 2020;

Martinez et al. 2021; Palme et al. 2017; Parker 2021) have also used UWG to demonstrate its potential to project UHI effect on weather files.

UWG is a methodology and software tool that estimates hourly urban canopy air temperature and humidity ratio based on urban morphological parameters and urban land use. It can be used alone or in conjunction with other existing programs to account for the impact of UHIs.

UWG model contains four interacting components: Rural Station Model (RSM) which estimates sensible heat fluxes; Vertical Diffusion Model (VDM) that calculates vertical air temperature profiles at a rural weather station; Urban Boundary Layer (UBL) that accounts for vertical histograms of the air temperature above the urban coverage; and Urban Canopy and Building Energy Model (UC-BEM) that allows to take into consideration temperature and humidity ratio of the air in the urban canyon (Kamal et al. 2021). UWG has also previously been validated in several studies for Basel, Singapore, Toulouse, Rome, Barcelona, and Abu Dhabi (Bande et al. 2019).

3.2 Data and Methods

A large number of impact, vulnerability, and adaptation studies are being conducted across the world, and in particular in Europe. Several common data processing tools and procedures are necessary for researchers/practitioners to move from the GCM data to the impact/vulnerability model data. Selection of climate data is one of several inputs necessary for a typical impact/vulnerability assessment, regardless of the industry.

Researchers/practitioners can access climate data, most of the time free, from online platforms where climate science specialists provide the raw data of GCMs and RCMs. Major data providers are as follows:

Table 3-1 : Names and links to climate data providers' platforms

	Name and link to data provider platforms	**Remark**
1	Climate4impact: developed within the European projects IS-ENES, IS-ENES2 and CLIPC. https://climate4impact.eu/impactportal/data/esgfsearch.jsp#	Open source /part of data is bias adjusted
2	IPCC data : http://ipcc-data.org/sim/gcm_monthly/AR5/Reference-Archive.html Users need to register here before being able to access data : DKRZ long term archive : The German Climate Computing Center https://cera-www.dkrz.de/WDCC/ui/cerasearch/	Open source for non-commercial use.
3	Earth System Grid Federation (ESGF): an international effort led by the Department of Energy (DOE), and co-funded by National Aeronautics and Space Administration (NASA), National Oceanic and Atmospheric Administration (NOAA), National Science Foundation (NSF), and	Open source for non-

	international laboratories such as the Max Planck Institute for Meteorology (MPI-M) German Climate Computing Centre (DKRZ), the Australian National University (ANU) National Computational Infrastructure (NCI), Institut Pierre-Simon Laplace (IPSL), and the British Atmospheric Data Center (BADC). CMIP5: https://esgf-node.llnl.gov/projects/esgf-llnl/	commercial use
4	HadGHCND - gridded daily temperatures observations : Designed for the analysis of climate extremes and for climate model evaluation https://www.metoffice.gov.uk/hadobs/hadghcnd/	Only climate variable available is temperature
5	Climate Data Store (e.g., The Coordinated Regional Downscaling EXperiment (CORDEX)): contains a catalogues and modelled and historical GCMs and RCMs. https://cds.climate.copernicus.eu/cdsapp#!/dataset/projections-cordex-domains-single-levels?tab=form	Freely available
6	Meteo France Archives https://donneespubliques.meteofrance.fr/?fond=rubrique&id_rubrique=26 Données Publique −► Observations In situ −► Données SYNOP essentielles OMM −► Téléchargement de données archivées	Historical weather data from weather stations / open source

Table 3-1 demonstrates a list of major raw data providers that BPS practitioners can use for impact and vulnerability assessment. Data downloaded from these platforms need to be processed before being ready to be used in BPS studies.

Other ways of accessing data for BPS are through software such as Meteonorm, CCWorldWeatherGen, WeatherShift, and EnergyPLus weather file platform.

In the present study, availability of open source RCMs (dynamically downscaled GCM) data from EURO-CORDEX presented an opportunity to systematically compare three future DDS climate models and one future ESD model, assuming the high-emission scenario [representative concentration pathway (RCP) 8.5], with observed weather data of 2003. The latter was accessed from Meteo France archives and transformed into EnergyPlus (.epw) file format that can be directly used in BPS.

The python scrip used to transform downloaded historical weather data from Meteo France archives into (.epw) weather file format is presented in **Appendix 3-1**.

3.2.1 Extracting yearly weather data

Coordinated Regional climate Downscaling Experiment CORDEX (www.cordex.org) is an international coordinated effort supported by World Climate Research Programme's Working Group on Regional Climate. As a part of CORDEX, EURO-CORDEX is today the main reference framework for regional downscaling research of climate data. The main goals of this

program are: (1) to evaluate and improve different RCMs, (2) to better understand regional and local climate phenomena through downscaling, (3) to generate coordinated RCM projections at global scale, (4) and to enable users of regional climate data to exchange knowledge (Daniela et al. 2020).

EURO-CORDEX maintains a consistent database of multi-year historical and projected data that can be used for climate adaptation studies in various sectors. The data for Europe is available on a horizontal grid resolution of 0.11° x 0.11°, equivalent of 12.5 km (Jacob et al. 2014). All necessary components to generate weather files for building simulations can be downloaded at 3h, 6h, daily, monthly and seasonal temporal resolution. For this study, we downloaded 3h time-step data.

In this study, EURO-CORDEX regional climate projection data were accessed via Climate Data Store (CDS) portal supported by Copernicus Climate Change Service (C3S) initiative that provides information about past, present, and future climate in Europe and the rest of the world[6]. CDS portal allows to access climate variables of GCMs and RCMs in various combinations and different horizontal and temporal resolutions. Raw data is available in NetCDF (Network Common Data Form) file format that is commonly used within the climate modelling community to share array-oriented scientific data. Climate data in this format are stored in multi-dimensions and users can view/access geographical coordinates (latitude, longitude), time, level, climate variable (temperature, relative humidity, etc.). Practitioners are cautioned to check for bias-adjustment of climate data before using them in BPS. In this study, raw data were downloaded from CDS portal and they are not bias adjusted.

Figure 3-6 depicts near-surface air temperature of an RCM in EURO-CORDEX region as well as the position of our case study city (Nantes) in a NetCDF file.

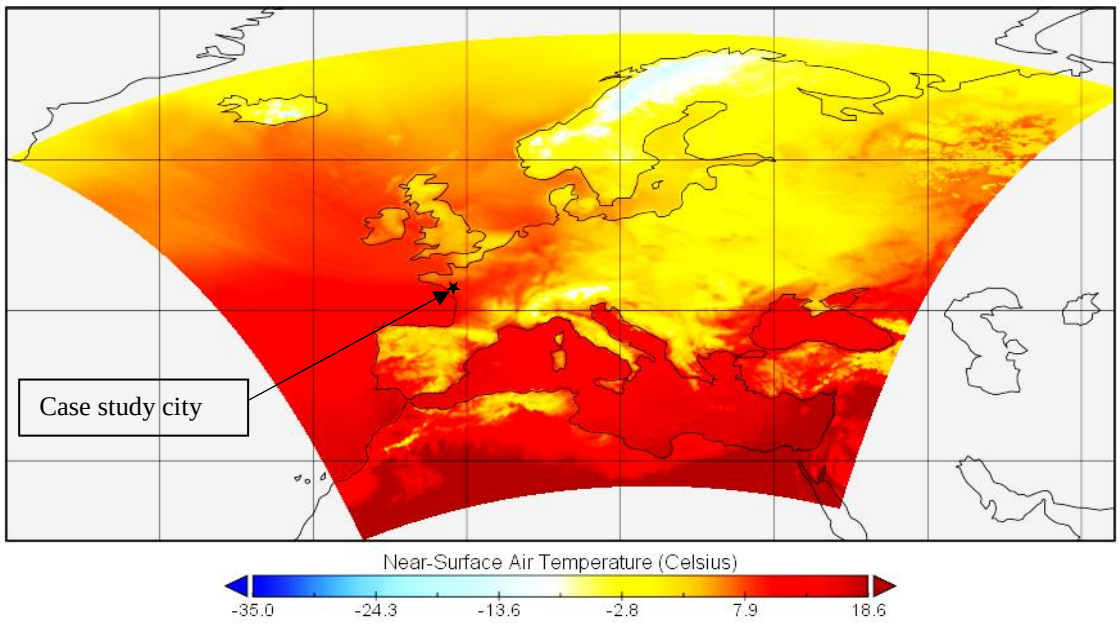


Figure 3-6: Near surface air temperature NetCDF data file visualized in Panoply

[6] https://cds.climate.copernicus.eu/cdsapp#!/dataset/projections-cordex-domains-single-levels?tab=form

Raw NetCDF data that contain RCMs in CORDEX domains were projected to a Rotated grid pole coordinate system.

Data of Nantes were extracted using python script that identified the closest point of the data grid in NetCDF file to the assigned latitude and longitude coordinates.

The script first transforms regular latitude and longitude of target location into rotated latitude and longitude then using the new rotated latitude-longitude identifies the closest grid point in NetcDF file. **Appendix 3-2** shows the script that transforms lat/lon to rlat/rlon and vice versa. **Appendix 3-3** explains the script that identifies the closest point in NectCDF and extracts climate variables of that point for one year.

Six climate variables (dry-bulb temperature [K], relative humidity [%], global solar radiation [W/m^2], cloud cover [%], atmospheric pressure [Pa], and wind speed [m/s]) as suggested by (Machard et al. 2020) were downloaded for thirty years (2040 to 2070) of the following global-regional climate models:

Table 3-2 : Dynamically downscaled climate models

	Institution	Global climate model (GCM)	Regional climate model (RCM)	GCM_RCM names used
1	CNRM	CNRM-CERFACS-CM5 (France)	CNRM-ALADIN63 (France)	CNRM_ALADIN
2	SMHI	IPSL-CM5A-MR (France)	SMHI-RCA4 (Sweden)	ISPL_SMHI
3	GERICS	MPI-M-MPI-ESM-LR (Germany)	GERICS-REMO2015 (Germany)	MPI_REMO

The climate models in Table 3-2 were chosen based on the availability of completed simulations with all six climate variables for RCP8.5 scenario experiments at 3h time-step interval. More combination of GCM and RCM are possible in CDS portal but, here, we only used each GCM and RCM once.

Dry-bulb temperature was first converted from Kelvin to Celsius. Polynomial interpolation (n=7) was used to convert 3h time-step data to 1h time-step for dry-bulb temperature and global solar radiation and linear interpolation for the rest of variables.

Dew point temperature (T_d) was estimated from dry bulb temperature and relative humidity using August–Roche–Magnus formula for dew point temperature approximation (Thiis et al. 2017).

$$T_d = \frac{b[ln\left(\frac{RH}{100}\right) + \frac{a \cdot T}{b+T}]}{a - ln\left(\frac{RH}{100}\right) - \frac{a \cdot T}{b+T}} \qquad \textit{Equation 3-1}$$

Where: RH Relative humidity [%]

T Dry bulb temperature [°C]

a = 17.27, b = 237.7 °C, for T ≤ 60 °C and an error of ±0.4 °C.

Sunrise and sunset time for the given location was calculated using Python Astral package, which is based on equations from Astronomical Algorithms, by Jean Meeus. Interpolated global

solar radiation data that were before sunrise and after sunset were set to zero. Solar zenith angle, direct normal irradiance and diffuse horizontal radiation were calculated following the methodology described by (Machard et al. 2020). Practitioners can also use alternative techniques available in *pvlib Python Package* such as DISC, and Perez approximation to calculate solar angle, direct and diffuse irradiance from global horizontal irradiance (F. Holmgren, W. Hansen, and A. Mikofski 2018; Nou et al. 2016). Following the steps described above, for each year of each climate model, a yearly weather file was generated. In the next step, 30 years of weather data for each climate model were assembled to generate future typical weather file.

3.2.2 Assembling typical weather files

We used EN ISO 15927-4_2005 standard created by European Committee for Standardization, proposing a method for constructing reference year of hourly weather data to generate typical future weather file. In this method, dry-bulb temperature, relative humidity, and global horizontal radiation climate variables are the key parameters in selection of "best" months to form reference year, with wind speed as a secondary (ISO 15927-4 2005).

Following the ISO 15927-4 method for each climate model, we first merged 30 years of hourly weather data and calculated daily means. Then for each calendar month, cumulative distribution function (CDF) of daily means of every year and of multiple-year were calculated.

For each calendar month, Finkelstein-Schafer statistic (FS) was calculated and individual months from multiple-year dataset were ranked in ascending order.

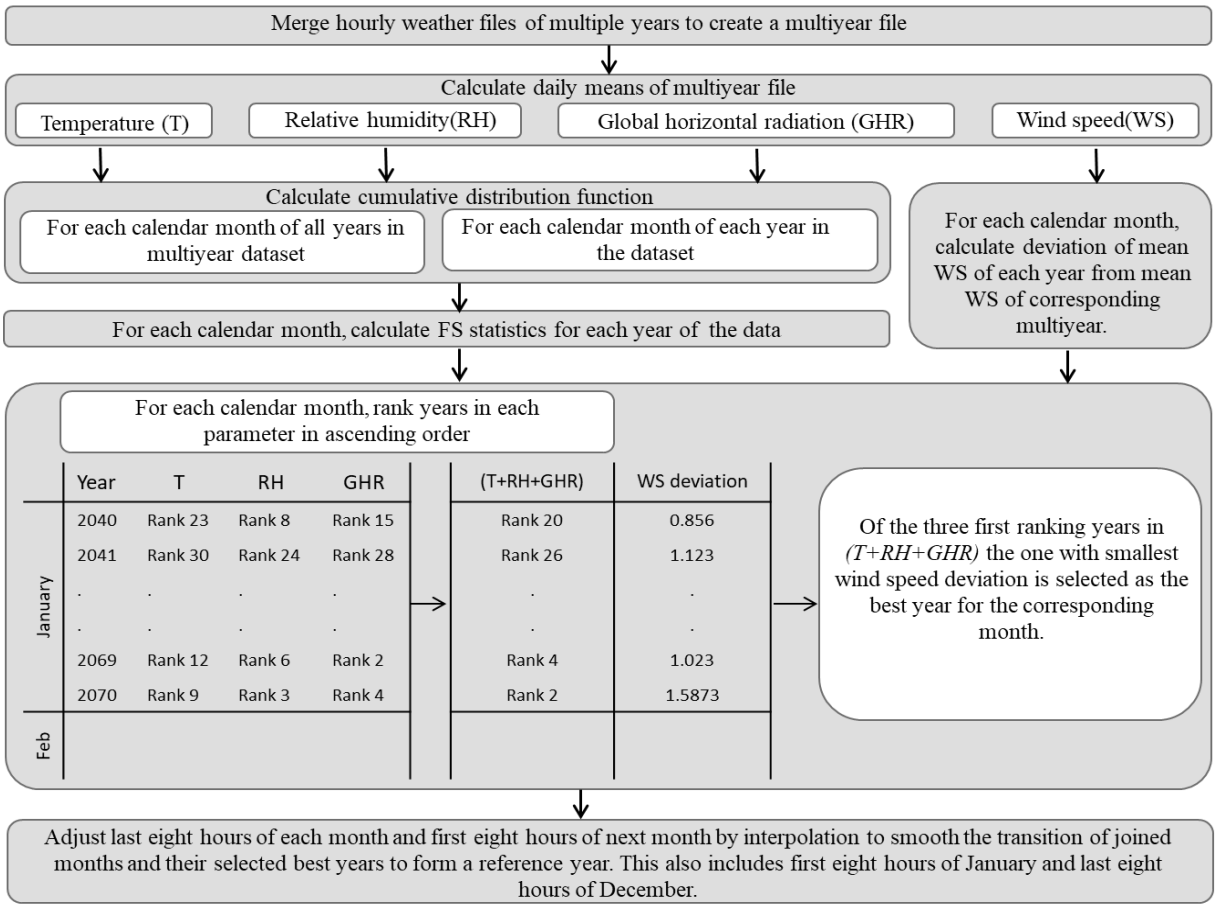


Figure 3-7 : Workflow to assemble typical weather file

For each calendar month and each year, separate ranks were added for each of the three key parameters. ISO 15927-4 gives equal weighting to three key climate parameters. Therefore, ranks of key parameters were only added to one another and ranked in ascending order. Of the three months with lowest total ranking for three key parameters, the one with the smallest wind speed deviation was selected as the "best" month to be included in the reference year. The procedure is also presented in Figure 3-7.

Appendix 3-4 illustrates the calculation procedure and script used in creation of typical meteorological year from 30 years of data.

As an example, Figure 3-8 shows cumulative distribution function plots of dry-bulb temperature, relative humidity, global horizontal irradiance and mean wind speed deviation plot for the calendar month of July in IPSL-SMHI model. In this example, from the three key parameters, July of 2050, July of 2060 and July of 2061 were candidates of best month for reference year. Among them July of 2061 had the smallest wind speed deviation from July in multiyear dataset. Therefore, it was selected as the "best" month for reference year.

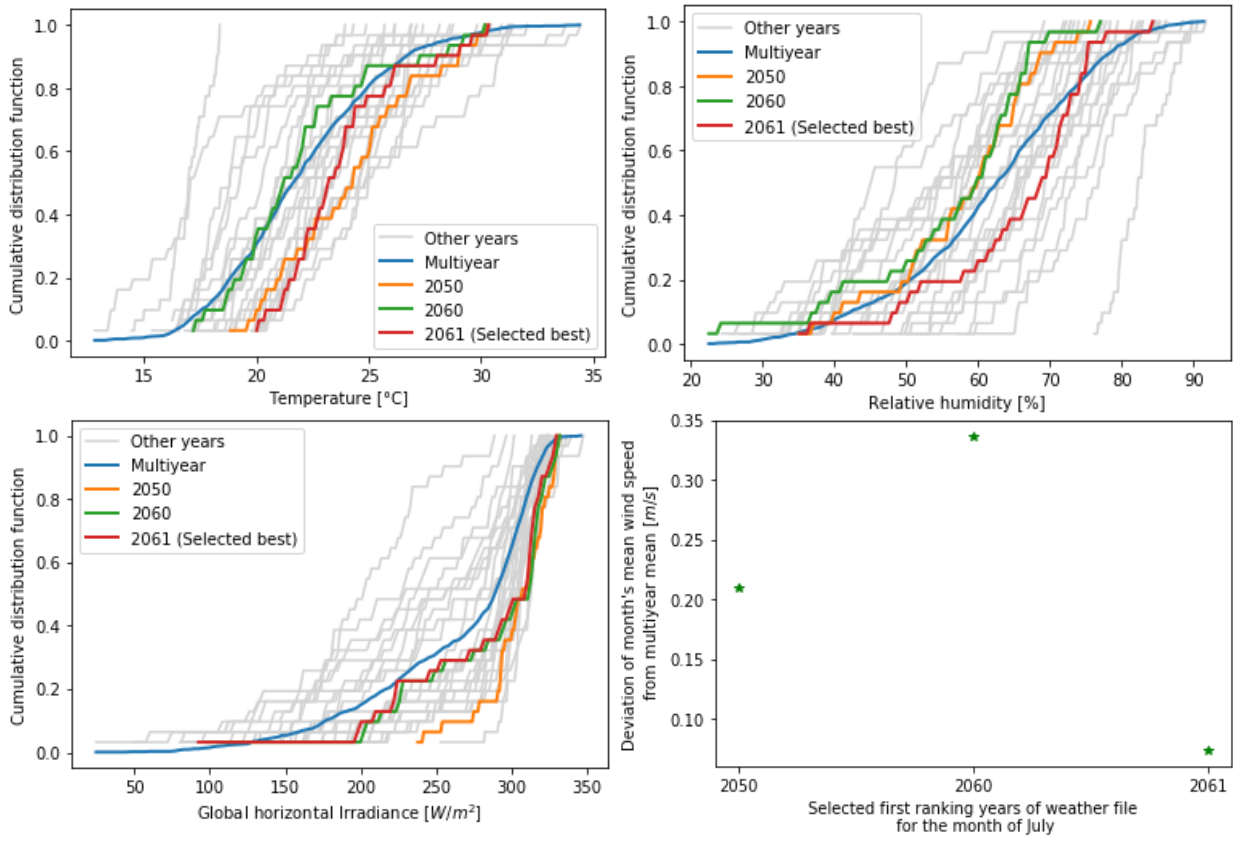


Figure 3-8: CDF plots of three key parameters and plot of wind speed deviation for July calendar month in IPSL_SMHI climate model

Using the method described, three future typical weather files (2040 to 2070) for the case study city from dynamically downscaled climate models were assembled.

For comparative analysis presented below, observed weather data of 2003, and one ESD future weather scenario (meteonorm RCP 8.5 2050) was also downloaded from MeteoFrance archives and Meteonorm v.8 software respectively, for our case study city, Nantes.

3.2.3 Projecting UHI effect using UWG

As mentioned earlier, we used spatially broad modelling approach using UWG to project the influence of UHI effect on weather files. For this, to maintain consitency with our previous works, we used python to read epw file, project the effect of UHI and export a modified version of weather file.

In the proposed methodology in this manuscript, we first mapped building clusters with building and urban morphological parameters. They were also used as input for UWG numerical model.

The input parameters of UWG model were taken from the cluster of buildings that are decribed in chapter 2 of this manuscript. **Appendix 3-5** demonstrates the brief procedure to modify weather files with UWG in python.

3.2.4 Comparative analysis of weather files

Statistical distribution of monthly dry bulb temperature, relative humidity, and global horizontal irradiance of each DDS climate model, as well as Meteonorm RCP 8.5 2050, and measured weather data of 2003 were compared. Additionally, heating degree-days (HDD) and cooling degree-days (CDD) indices in each weather file were calculated using the methodology and base temperatures recommended by EUROSTAT to form a common and comparable basis in comparison. These two indicators are commonly used to give a rough estimation of heating and cooling energy demand. Calculation of HDD and CDD both rely on base temperature, which depends in principle on various factors related to building and surrounding. In this study, base temperature in HDD calculation was set to constant value of 15°C and in CDD calculation to a 24°C (Bhatnagar, Mathur, and Garg 2018).

In HDD calculation:

If $T_m \leq 15°C$ then $HDD = \sum_i (18°C - T^i_m)$,

Else HDD = 0,

Where T^i_m is the mean air temperature of day i.

Similarly in calculation of CDD,

If $T_m \geq 24°C$ then $CDD = \sum_i T^i_m - 21°C)$,

else CDD = 0,

Weather files were also investigated with a temperature-based index to check the frequency and intensity of heatwaves in them. This temperature-based heatwave index was developed for France after the exceptional heatwave in summer of 2003. This index relies on the heatwave and health alert system (Sacs) piloted by *Santé Publique* France. Its objective is to anticipate heat waves that are likely to have a major health impact. Every day, in each metropolitan department, the level of risk is assessed by MeteoFrance comparing forecasts of meteorological indicators with departmental alert thresholds. Thresholds are shown in Figure 3-9, below.

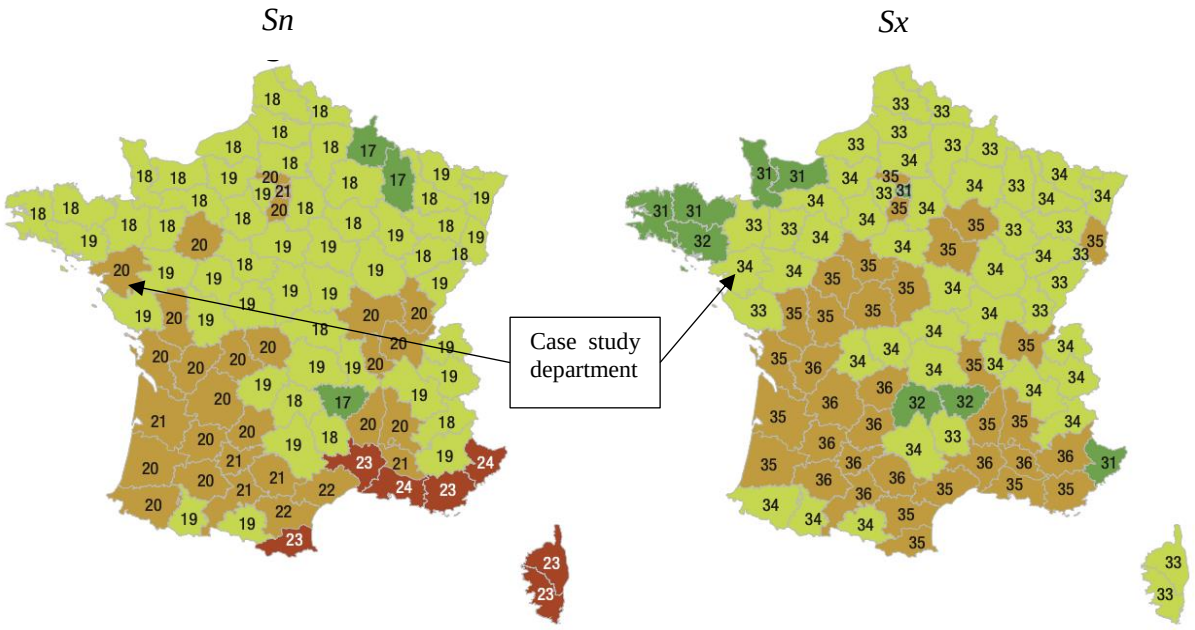


Figure 3-9: Heatwave meteorological indicator thresholds, ***Sn****: threshold for the three-day average of minimum temperatures in [°C].* ***Sx****: threshold for the three-day average of maximum temperatures in [°C].*

These thresholds are defined on the bases of a historical analysis and with aim to anticipate heat waves that are likely to be associated with excess mortality of at least 50% in 4 major cities and 100% in smaller cities. In this method, "*A heat wave is defined as a period when the minimum and maximum temperatures, averaged over three days, simultaneously reach or exceed departmental alert thresholds*". The onset of a heat wave corresponds on the first day on which the meteorological indicators of Sacs (average over three consecutive days of minimum and maximum temperatures) reaches or exceeded alert thresholds (Figure 3-9). The end of a heatwave is the last day of meeting or exceeding these thresholds (Wagner, 2006). The Sacs thresholds have seen certain evolutions over the years and the threshold presented in Figure 3-9 are those of 2016.

Selected weather files for the case study city were measured against the heatwave meteorological thresholds of case study city department. To do so, maximum daily temperatures (day_max) and minimum daily temperature (day_min) of weather files were extracted from hourly data. A python script was written to detect and measure the heatwaves parameters such as start date, duration, peak temperature, intensity of maximum temperatures, and intensity of minimum temperatures.

Appendix 3-6 illustrates calculation of heatwave presence in .epw weather files with a python script using this temperature-based index**.** The script to calculate HDD and CDD is in Appendix 3-6.

3.3 Results and discussions

3.3.1 Comparative analysis of weather files

The objective of this sub-section is to analyse the differences in various climate models at monthly scale with different climate variables. Monthly statistical distribution of dry bulb temperature, global horizontal irradiance, and relative humidity of three DDS models, one ESD (meteonorm 2050), and 2003 observed weather data are shown in Figure 3-10.

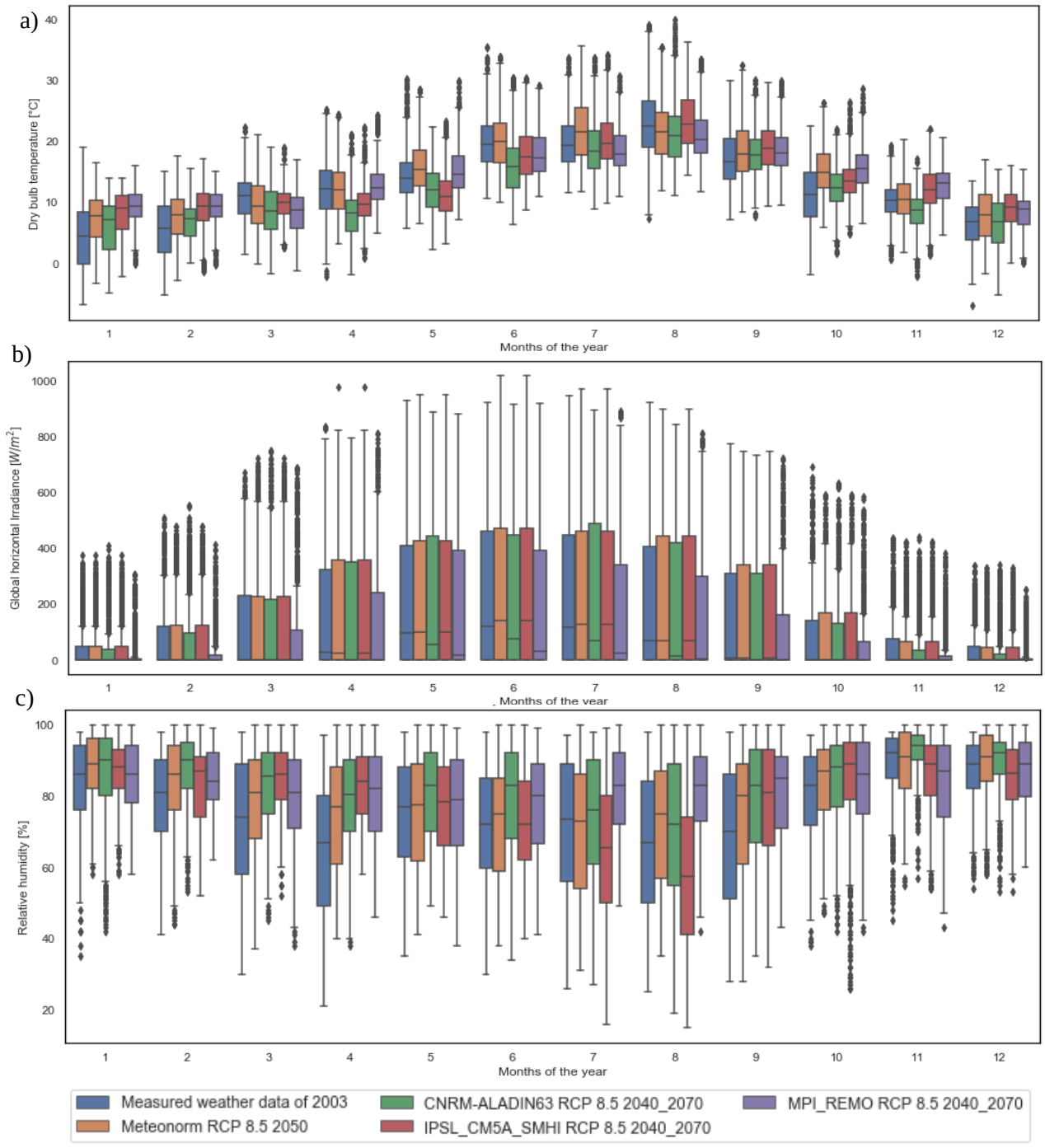


Figure 3-10: Monthly statistical distribution of: a) dry bulb temperature, b) global horizontal irradiance, c) relative humidity

Observed weather of 2003 was selected for comparison because it was the severest heatwave recorded in France up until June 28, 2019 when a temperature of 45.9°C was recorded during another heatwave in a weather station in France, exceeding previous record temperature of 2003 by almost 2°C. In contrast to 2003 heatwave, number of excessive mortality was considerably lower mainly because the duration of 2019 heatwave was shorter and there were better heat-response plans in place (D. Mitchell et al. 2019).

Moving back to comparative analysis, boxplots in Figure 3-10 show significant variations in monthly mean values among the selected weather data files. This indicates a great difference in predictions from one climate model to another. All three dynamically downscaled weather files for 2040-2070 and Meteonorm weather file for 2050 show a consistent higher mean monthly value of dry bulb temperature and relative humidity compared to observed weather data of 2003. However, the differences in global horizontal irradiance seems insignificant and does not provide enough evidence to notice an upward or downward trend.

Zooming in into summer months we notice that mean monthly dry bulb temperature in July, August, and September of IPSL-SMHI climate model is closest to mean monthly temperature of 2003 measured weather data. For the same summer months, mean relative humidity of Meteonorm 2050 is closest to 2003 measured weather file.

Among dynamically downscaled weather files, CNRM-ALADIN and MPI-REMO predict higher relative humidity but lower global horizontal irradiance and dry bulb temperature during summer months. IPSL-SMHI, on the other hand, predicts higher temperature and global horizontal irradiance but lower relative humidity for the same summer months. Comparing the length of whiskers and size of interquartile ranges in boxplots for dry bulb temperature and relative humidity variables show that Meteonorm 2050, MPI-REMO, and IPSL-SMHI weather files have relatively smaller dispersion compared to measured weather data of 2003 and CNRM-ALADIN weather files. This could indicate that measured weather of 2003 and CNRM-ALADIN contain more severe temperature anomalies compared to other climate models investigated here.

Variations in weather files are also reflected on the number of HDD and CDD, presented in Table 3-3.

Table 3-3: Number of Heating Degree Days (HDD) and Cooling Degree Days (CDD) in different climate models for the case study city

	Observed data	**Statistically downscaled**	**Dynamically downscaled FTWY**		
	Measured weather data of 2003	**Meteonorm RCP 8.5 2050**	**CNRM-ALADIN RCP 8.5 2040-2070**	**IPSL_SMHI RCP 8.5 2040-2070**	**MPI_REMO RCP 8.5 2040-2070**
HDD	2106	1741	2365	1873	1595
CDD	103	86	62	67	26

The main purpose of HDD and CDD is to demonstrate heating or cooling energy demand of buildings. The spike in CDD for observed weather data of 2003 is most likely due to heatwave data for the month of August in the weather file. Spike in HDD in CNRM-ALADIN, on the other hand, is probably due to a cold snap in March and April, shown in (Figure 3-10,a). Application of heatwave presence assessment index, described in section 2.3, also detects heatwave that will likely to have major health impact only in CNRM-ALADIN typical future weather data and measured weather file of 2003 (see Figure 3-11).

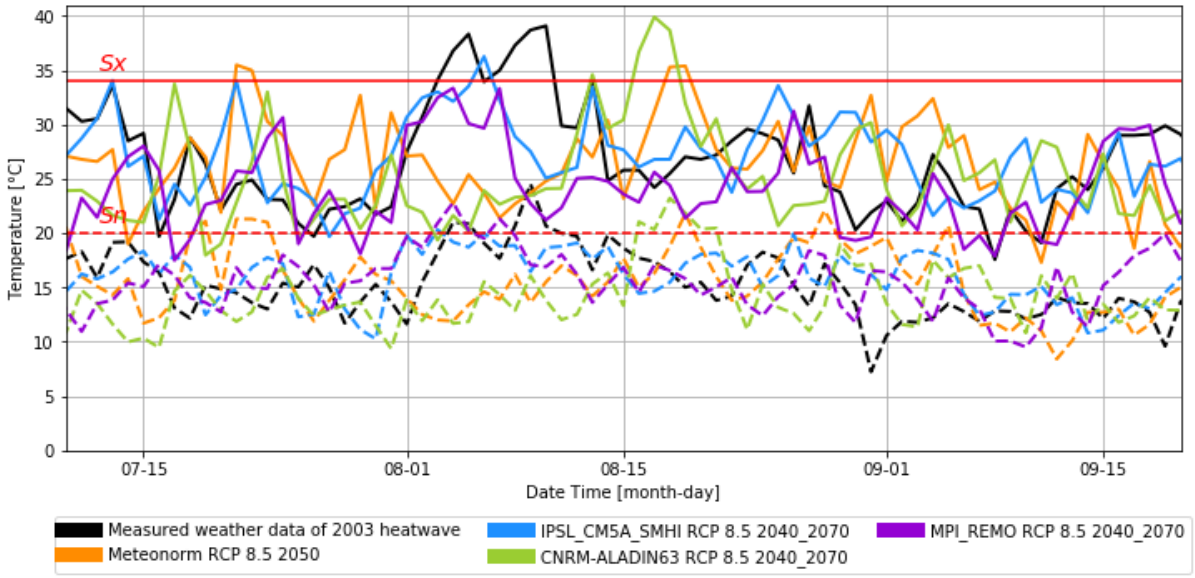


Figure 3-11: Maximum and minimum daily temperatures of weather files plotted alongside heatwave meteorological indicator thresholds

In Figure 3-11, red straight lines show the position of heatwave thresholds, colours represent weather files, solid lines demonstrate maximum daily temperatures, and dashed lines indicate minimum daily temperatures.

Duration of heatwave detected in CNRM-ALADIN weather file is 3 days and peak temperature is 39.9 degrees. In 2003 measured weather data, the duration of heatwave is 9 days and peak temperature reaches up to 39.1 degrees.

Among the selected weather files, for our case study city, measured weather data of 2003 and CNRM-ALADIN can both be used in the study of heatwave impact on indoor thermal comfort during summer months. However, if the study period also includes winter months, the 2003 measured weather file may not be suitable. That is mainly because it underestimates temperature increase due to climate change compared to other weather files in winter months. Due to the absence of a concrete reference for future weather conditions, it is difficult to declare one climate model is better than others; nonetheless, considering monthly statistical dispersion and presence of anomalies it is better, under current conditions**,** to use measured weather data, such as weather data of 2003 for indoor thermal comfort evaluation buildings during heatwaves.

For typical future weather scenarios, weather data generated from ESD and DDS could both be used. In this study, due to time limitations we decided to evaluate the performance of our selected reference buildings with only IPSL-SMHI typical future weather file of 2040 to 2070.

3.3.2 Urban heat island effect on weather data

IPSL-SMHI FTWY was chosen for further evaluation of this methodology. This weather file and 2003 measured weather file were modified for urban heat island effect, individually for each representative building, using UWG model before being used in Trnsys v.17 for multi-zone dynamic building simulations. Input of UWG model were urban morphological parameters of each building cluster.

Output of UWG model were new weather files for each building cluster containing urban microclimate data in terms of temperature and relative humidity. The output files were then compared to the baseline weather file to evaluate the efficacy of UWG methodology for each cluster of buildings. In this comparison, mean monthly temperature of baseline weather file was subtracted from mean monthly temperature of modified weather file. Results of this comparison are presented in Table 3-4. Differences between baseline and modified weather files show the intensity of UHI effect projected by UWG model for each cluster of building as a function of their urban morphological characteristics on two weather files.

Table 3-4: Monthly mean UHI effect projected by UWG model on weather file of each cluster (°C)

IPSL-SMHI future weather file (2040 -2070)							
	KM7_0	KM7_1	KM7_2	KM7_3	KM7_4	KM7_5	KM7_6
January	0.28	0.34	0.33	0.32	0.25	0.32	0.24
February	0.26	0.29	0.28	0.27	0.22	0.26	0.20
March	0.35	0.34	0.34	0.33	0.28	0.32	0.27
April	0.71	0.66	0.66	0.66	0.60	0.65	0.57
May	0.92	0.82	0.84	0.82	0.77	0.82	0.75
June	1.02	0.92	0.94	0.93	0.87	0.91	0.81
July	0.98	0.88	0.91	0.87	0.81	0.88	0.78
August	0.97	0.93	0.95	0.91	0.83	0.90	0.79
September	1.14	1.13	1.13	1.10	1.01	1.06	0.99
October	1.04	1.05	1.03	1.03	0.94	0.97	0.94
November	0.32	0.38	0.37	0.36	0.28	0.33	0.28
December	0.33	0.39	0.38	0.37	0.30	0.36	0.29
2003 measured weather data							
	KM7_0	KM7_1	KM7_2	KM7_3	KM7_4	KM7_5	KM7_6
January	0.58	0.56	0.53	0.48	0.43	0.67	0.46
February	0.72	0.68	0.65	0.60	0.55	0.78	0.58
March	0.87	0.81	0.79	0.74	0.69	0.84	0.73
April	1.00	0.90	0.88	0.84	0.79	0.94	0.83
May	1.09	0.94	0.93	0.88	0.84	1.01	0.89
June	1.05	0.90	0.89	0.84	0.79	0.97	0.86
July	1.13	0.97	0.96	0.91	0.86	1.04	0.92
August	1.88	1.79	1.76	1.68	1.61	1.88	1.67
September	1.71	1.64	1.61	1.54	1.47	1.64	1.52
October	0.81	0.73	0.70	0.66	0.60	0.79	0.64
November	0.42	0.41	0.38	0.33	0.27	0.44	0.30
December	0.42	0.41	0.38	0.33	0.28	0.48	0.30

As can be seen in Table 3-4, temperature difference between modified and baseline weather files for all buildings is consistently higher in summer compared to winter months. UHI effect projected on KM7_0, KM7_1, and KM7_5 is higher than on KM7_6, and KM7_4 throughout the year, mostly likely because they are located in more densely built areas and has lower ratio of vegetation and greenery. Additionally, UHI effect projected on two weather files by UWG for the same urban morphological parameters varies considerably. To investigate this variation, we compared climate variables of the two weather files and the difference in the magnitude of UHI effect between the two but did not notice any clear correlation between model performance and climate variables. However, temperature, relative humidity, wind speed, global horizontal irradiance, etc. in the two weather files were not identical and the difference between UHI effect projected on two weather files for the same urban morphological parameters indicates that the magnitude of UHI effect projected also depends on the choice of weather station. This is in line with findings reported by (Bueno, Nakano, and Norford 2015).

3.4 Conclusions

We generated three FTWYs using this methodology, compared them with future weather file of Meteonorm 2050, and measured weather data of 2003. Comparative analysis of future weather files showed a difference not only between the weather files that were generated by DDS or ESD approaches but also between different DDS files themselves. This difference was demonstrated by statistical distribution plots of monthly air temperature, relative humidity and global horizontal irradiance in each weather file. The objective of comparison between future weather files and 2003 observed weather data was to check if given future files may or may not include heatwaves that would be reliable enough for BPS assessment. It was concluded that the heatwaves included in future typical weather files do not represent the intensity, duration and severity of heatwave that has already happened in 2003, let alone heatwaves that could happen in the future.

Another aim of this study was to provide insights for practitioners in BPS on how to generate future and present ready-to-use weather files using open source RCMs and, second, through a comparative analysis show their potential in evaluation of building's resilience to heatwaves and climate change, considering urban heat island effect. The method described here does not illustrate in details the uncertainties associated with emission scenarios, climate projections, climate models, and bias adjustment of climate models as they were addressed previously by (Hosseini, Bigtashi, and Lee 2021; Machard et al. 2020).

One major limitation of this work is that bias adjustment was not implemented in the script through which the yearly weather files were generated.

Another limitation of this study was that we used urban morphological parameters of closest building to cluster centroid to project the influence of UHI effect on that cluster. An alternative approach could have been using the urban morphological parameters of cluster centroid itself.

In this study, vegetation ratio as one of the necessary inputs to project the UHI effect using UWG was calculated from BDTOPO data. In the future studies, other sources of data, such as satellite images could be used to generate higher resolution data of the urban vegetation coverage ratio.

3.5 Summary

Climate data plays an important role in proper impact assessment of climate change on all sectors. For BPS, access to reliable and precise climate data describing outdoor conditions is an essential element. The objective of this chapter was to present a workflow for BPS practitioners to access and process open source future climate data.

To achieve the objective, this chapter investigates the following elements:

- It presents a detailed literature review on downscaling of climate data from GCMs to RCMs.

- It describes tools and indices used in heatwave presence evaluation from weather data.

- It illustrates the trade-off between data granularity and relevance to the policy at the city scale, which in turn affects the choice of tools and methods for weather data and BPS.

- In the data and methods section, it provides a list of climate data providers.

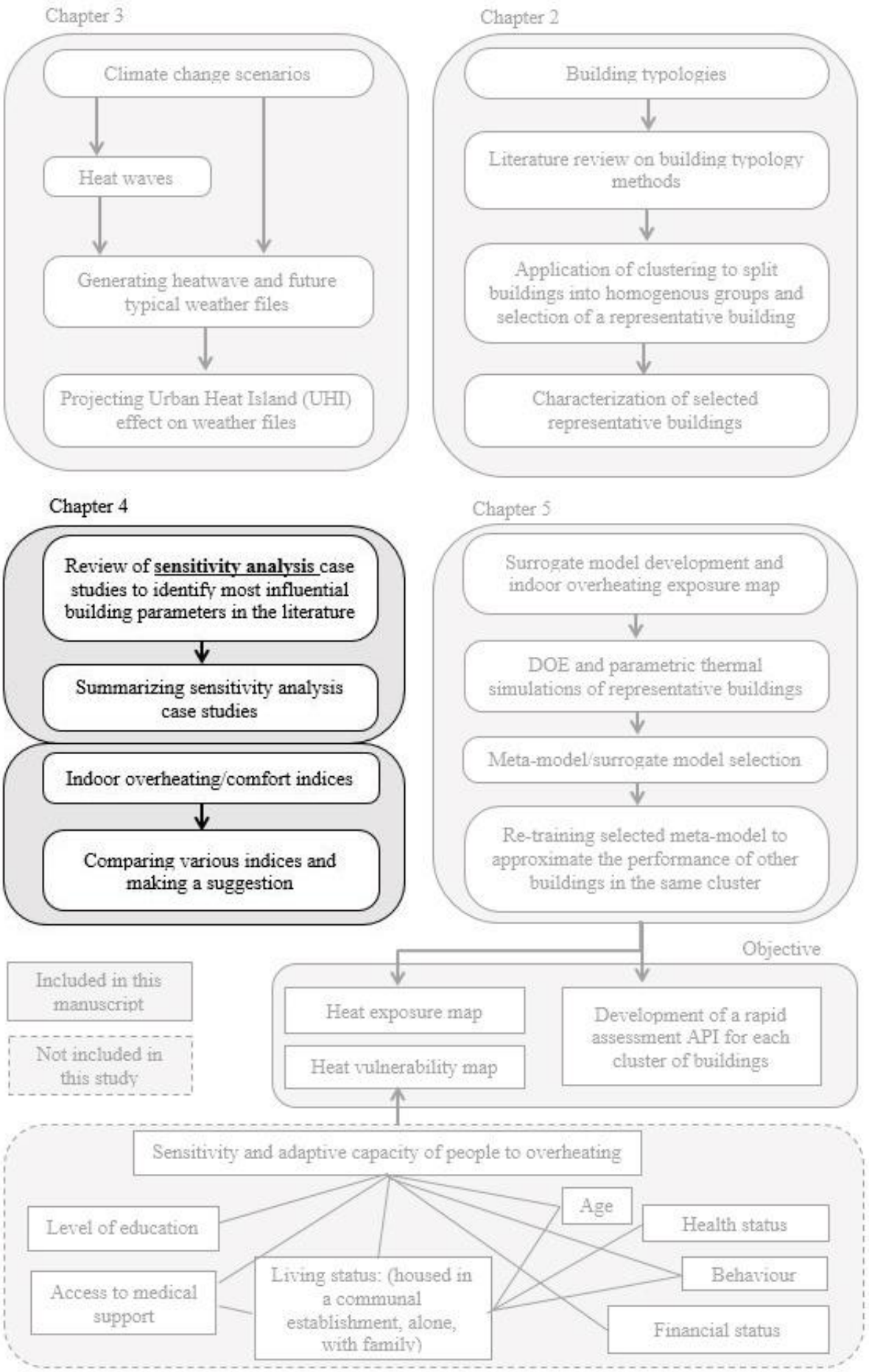
- Among the data providers, this chapter uses high-resolution RCM data from EUROCORDEX to generate typical future weather data for BPS practitioners. The workflow described in this section allows practitioners/researchers to access open source climate data of various climate models across Europe, and North Africa.

- Medium future typical weather data (2040-2070) of three climate models are compared to 2003 heatwave weather data for the case study city, Nantes, and the presence of heatwave in the weather files is investigated, employing a temperature-based index.

- The influence of UHI effect for each representative building (identified in chapter 2) is projected on the typical future weather file, using UWG tool.

In line with the overall objective of this thesis, which is to evaluate indoor performance of buildings at city scale taking into consideration future climate and UHI effect, this chapter presented a workflow to help researchers/practitioners access and use future and historical climate data, measure the intensity of heatwaves in them, and project the influence of UHI effect using UWG.

The next chapter illustrates what thermo-physical parameters of building influences its thermal and energetic performance the most, and what indices are available and suitable for indoor overheating measurement.


Chapter 3
Climate change scenarios
Heat waves
Generating heatwave and future typical weather files
Projecting Urban Heat Island (UHI) effect on weather files
Chapter 2
Building typologies
Literature review on building typology methods
Application of clustering to split buildings into homogenous groups and selection of a representative building
Characterization of selected representative buildings
Chapter 4
Review of sensitivity analysis case studies to identify most influential building parameters in the literature
Summarizing sensitivity analysis case studies
Indoor overheating/comfort indices
Comparing various indices and making a suggestion
Chapter 5
Surrogate model development and indoor overheating exposure map
DOE and parametric thermal simulations of representative buildings
Meta-model/surrogate model selection
Re-training selected meta-model to approximate the performance of other buildings in the same cluster
Objective
Heat exposure map
Heat vulnerability map
Development of a rapid assessment API for each cluster of buildings
Included in this manuscript
Not included in this study
Sensitivity and adaptive capacity of people to overheating
Level of education
Access to medical support
Living status: (housed in a communal establishment, alone, with family)
Age
Health status
Behaviour
Financial status

Chapter 4 : Building performance parameters and measurement indices

4.1 Part One: Factors of overheating and energy demand in buildings: a literature review

With climate change, extreme weather events and the amount of time spent indoors, the need for comprehensive building performance analysis, in thermal comfort as well as in energy consumption, has never been as strong as today (Pang et al. 2020). Building system and subsystems responsible for energy consumption and thermal comfort are highly non-linear and include different parameters. Sensitivity analysis as a powerful approach in identification of influencing parameters has been receiving significant attention in the last decades. A recently published literature review on the role of sensitivity analysis on building performance was done by (Pang et al. 2020). This work provides a comprehensive review on application of sensitivity analysis on a wide spectrum of building related topics. The emphasis is on the case studies where they rigorously investigate methods for sensitivity analysis, uncertainty and sampling analysis. It also includes the tools used for sensitivity analysis, building simulations; and categorization of input and output parameters of sensitivity analysis. However, this paper does not demonstrate the results of sensitivity analyses from the case studies. To tackle this question, we decided to conduct a literature review of papers that presented sensitivity analyses results for energy consumption and indoor thermal comfort in residential buildings. Figure 4-1, below shows the overall objective of this section of the thesis.

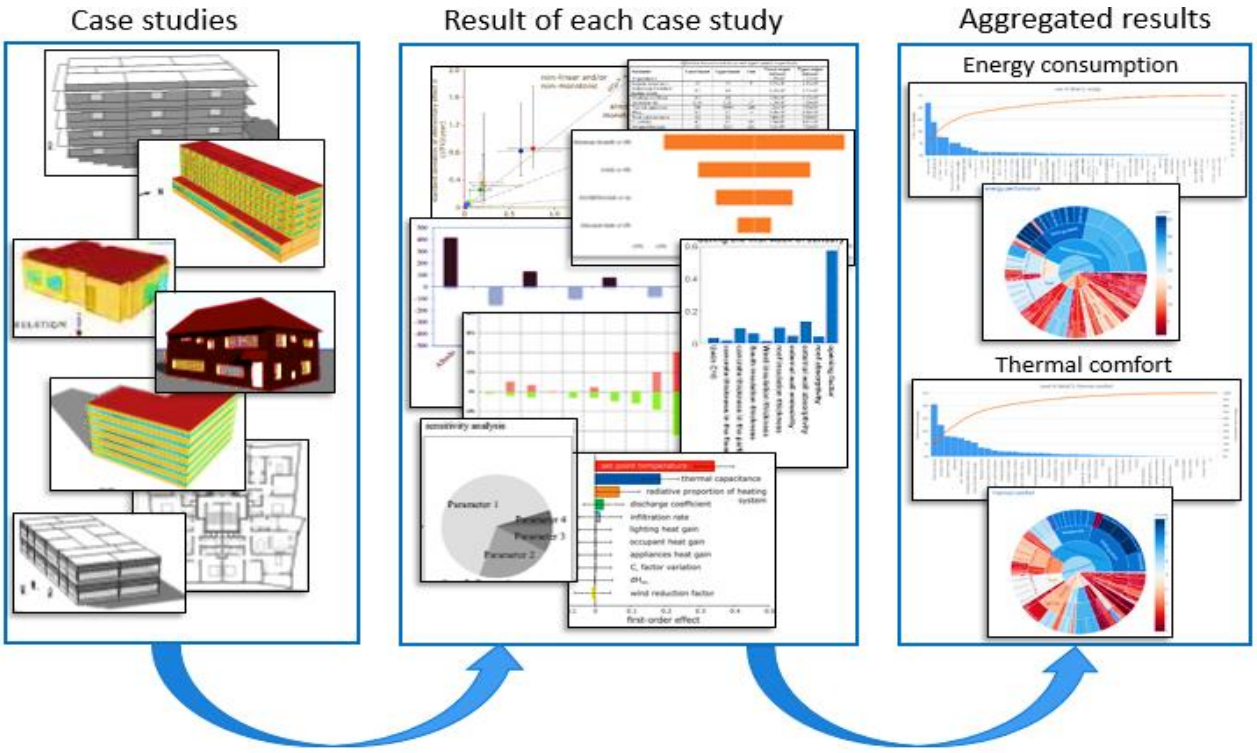


Figure 4-1: Flowchart in aggregation of building sensitivity analyses results

Before moving forwards with that, a differentiation between sensitivity analysis and uncertainty analysis must be made. In uncertainty analysis, the objective is to discover variation of an output depending on the uncertainty of input parameter or parameters. In sensitivity analysis, the objective is to measure the effect of individual independent variables on a particular dependent variable given a set of assumptions (Ray, Mukherjee, and Mandal 2015).

$$S = \frac{dx/x}{dp/p} \qquad \textit{Equation 4-1}$$

Where:	S	Sensitivity
	X	State
	P	Parameter
	d*x and* d*p*	Change of values of state variables, parameters.

Having said that, the goal of this part is to identify, through aggregation of sensitivity analysis studies, which input parameters contribute the most to variability of outputs in buildings (heating energy consumption, cooling energy consumption, thermal comfort, etc.) and rank their relevance and relative importance based on what is found in the literature. Review on the results of sensitivity analysis is therefore both qualitative and quantitative.

Key words used in search of papers were "sensitivity analysis", "buildings", "energy consumption/thermal comfort". Only papers published after 2009 and those that had clearly ranked the relative influence of independent variables on dependent were included in the study. At the end, results of sensitivity analysis from 72 papers were included for further processing.

Sensitivity analysis results in the selected papers were presented in many formats such as scatter plot, Tornado plot, bar plot, box plot, pie chart, spider plot, and numerical value. All the given formats allowed identifying the rank (relative importance) and total number of input parameters in relation to output as can be seen in Figure 4-1.

Papers included in this study employed different methods to conduct sensitivity analysis and some of them used more than one method for that purpose, as shown in Figure 4-2.

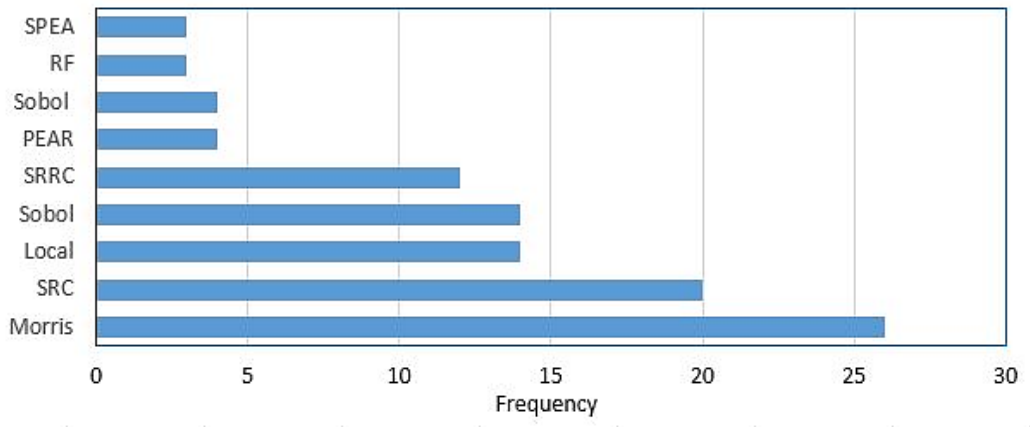


Figure 4-2: Sensitivity analysis methods seen in the selected papers

In Figure 4-2, horizontal axis indicates the frequency of times a specific method was observed in a paper and vertical axis shows sensitivity analysis method.

Papers that illustrated building energy and thermal comfort sensitivity analysis varied extensively in terms of number of initial parameters taken into account, building size and use, location, final ranking of parameters in terms of importance, and objective of sensitivity analysis. This extensive variation on a wide spectrum of elements rendered the process of cross comparison between the papers complex. Nonetheless, we decided to use an average method to consider frequency of times a parameter was listed in the papers, rank of parameter and total of number of parameters in each paper.

4.1.1 Method

Figure 4-3, demonstrates the method utilized to aggregate the results of sensitivity analysis studies in the literature. It starts with creation of a database containing the information about

a)

rank of parameter in the paper

	parameter 1	parameter 2	parameter ..	parameter ..	parameter n	Total number of parameters in the paper
paper 1	2	1	4	3		**4**
paper 2	2	3	1			**3**
paper ..	1		2		3	**3**
paper n	5	3	2	1	4	**5**
Number of papers studying the parameter	***4***	***3***	***4***	***3***	***2***	

b)

	parameter 1	parameter 2	parameter ..	parameter ..	parameter n	total number of parameters
paper 1	=2/(4*4)	1	4	3		**4**
paper 2	2	3	1			**3**
paper ..	1		2		3	**3**
paper n	5	3	2	1	4	**5**
Number of papers studying the parameter	***4***	***3***	***4***	***3***	***2***	

c)

The smaller the number the higher the impact

	parameter 1	parameter 2	parameter ..	parameter ..	parameter n	total number of parameters
paper 1	0.125	0.083	0.250	0.250		4
paper 2	0.125	0.250	0.063	0.083		4
paper ..	0.083		0.167		0.5	3
paper n	0.250	0.200	0.100	0.067	0.4	5
Number of papers studying the parameter	4	3	4	3	2	
	6.86	5.63	6.91	7.50	2.22	

=1/average

*Figure 4-3 : Method used to summarize case studies from literature. **a**): Rank of each parameter in each paper; **b**) dividing rank in each to the product of number of parameter and papers studying that parameter; **c**) shows average rank of each parameter*

the paper, parameters studied, and rank of each parameter in the results for all papers selected in the study.

In this method, rank of each parameter in a paper is divided to the product of total number of parameters to the number of times same parameter has been observed in all papers. Parameters that rank higher (1st, 2nd, 3rd... etc.) when divided to the product generate smaller numbers so the smaller the number the higher the impact. To create a general impact factor inverse average of number was calculated, as also presented in Figure 4-3 and in the equation below.

$$Average\ rank\ ratio\ of\ a\ parameter = \frac{1}{\frac{\sum_{i=0}^{n} \frac{R}{(P * T)}}{P}} \qquad \textit{Equation 4-2}$$

Where:
- R — rank of a specific parameter in each paper
- P — number of paper studying the parameter
- T — total number of parameter in that paper

4.1.2 Results

Aggregated results from literature review were analysed separately for thermal comfort/indoor overheating, heating energy consumption, cooling energy consumption and energy consumption (heating + cooling) in four levels of detail. An example of the level of details is given in Figure 4-4. The window wall ratio (WWR) in each orientation is at the level of detail 4. The average WWR of the building is at the level of detail 3 while window characteristics is at level of detail 2 and finally, envelope properties at level of detail 1.

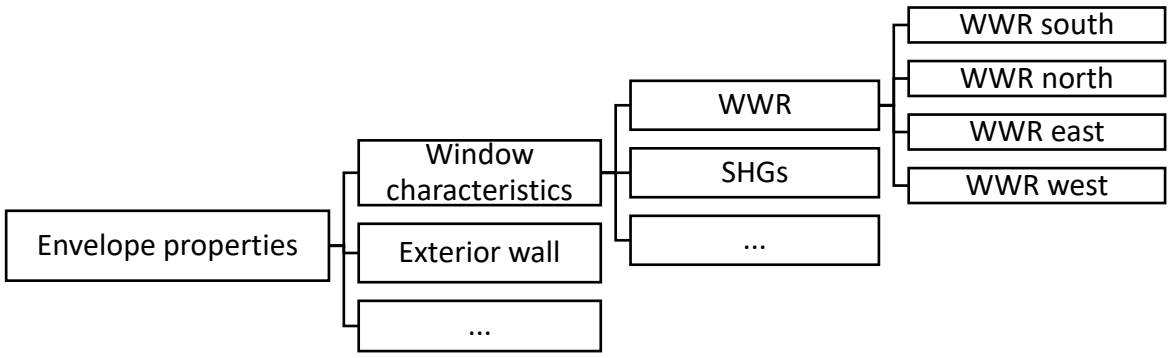


Figure 4-4: an example of level of details

Level four is the most and level one is the least detailed characterization of building.

Aggregated results of sensitivity analysis in various levels of details are shown in Figure 4-5. In this figure, vertical axis represents the aggregated rank of parameters and horizontal axis illustrates number of papers for level of detail 4 and average number of papers for the other levels of details.

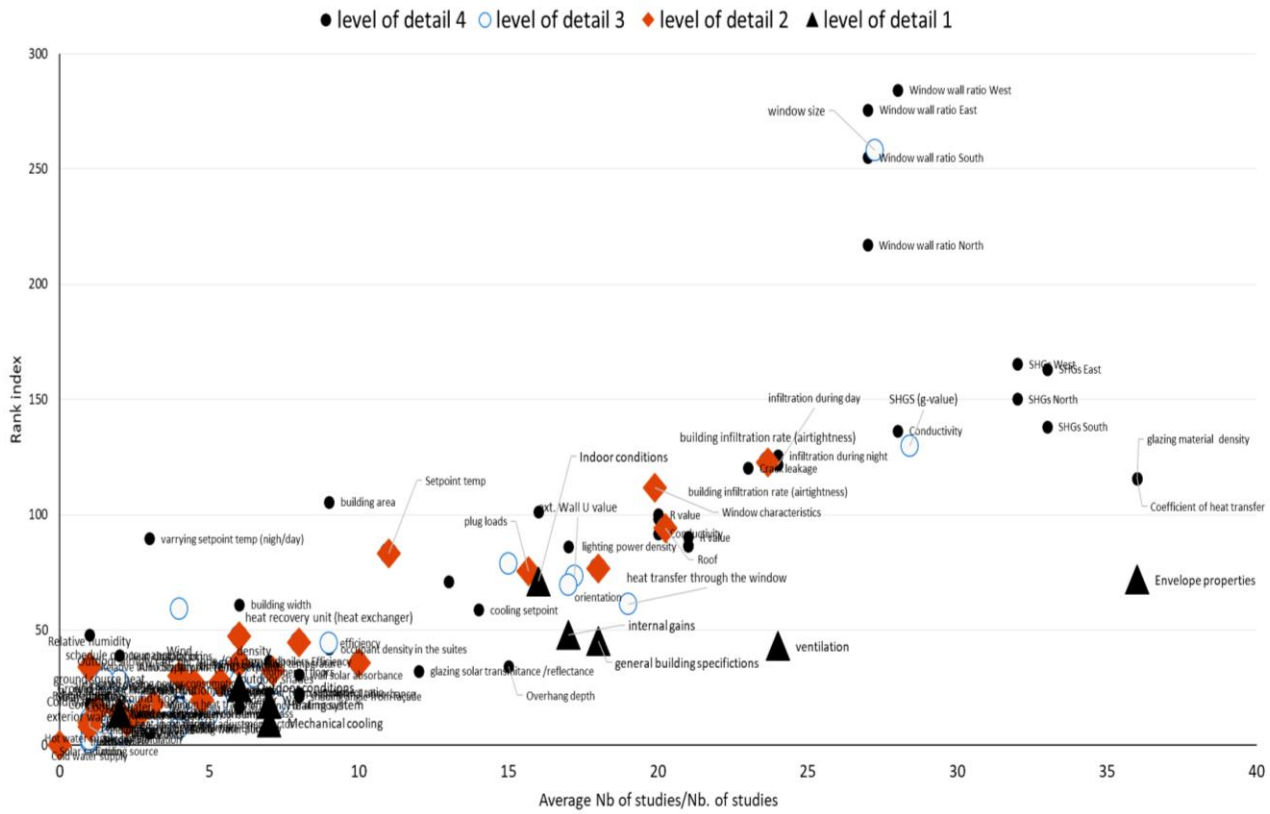


Figure 4-5: Relative rank and number of studies of each building parameter with different levels of detail for energy consumption (heating +cooling)

Considering the context of the thesis and the scale of study, results from third level of details are for the moment deemed best in characterization of reference buildings. These results are briefly presented in the following paragraphs.

In thermal comfort assessment, solar heat gains through the windows (SHGs) also referred as g-value in some papers, was identified as the most influential parameter and was closely followed by WWR. The next five influential parameters were roof U-value, lighting power density, U-value of exterior wall, infiltration rate (airtightness), and heat transfer through the window. These parameters are also shown in Figure 4-6, below. It is also important to mention that in thermal comfort case studies various indices were used as an indicator of comfort (PMV, SET, indoor operative temperature).

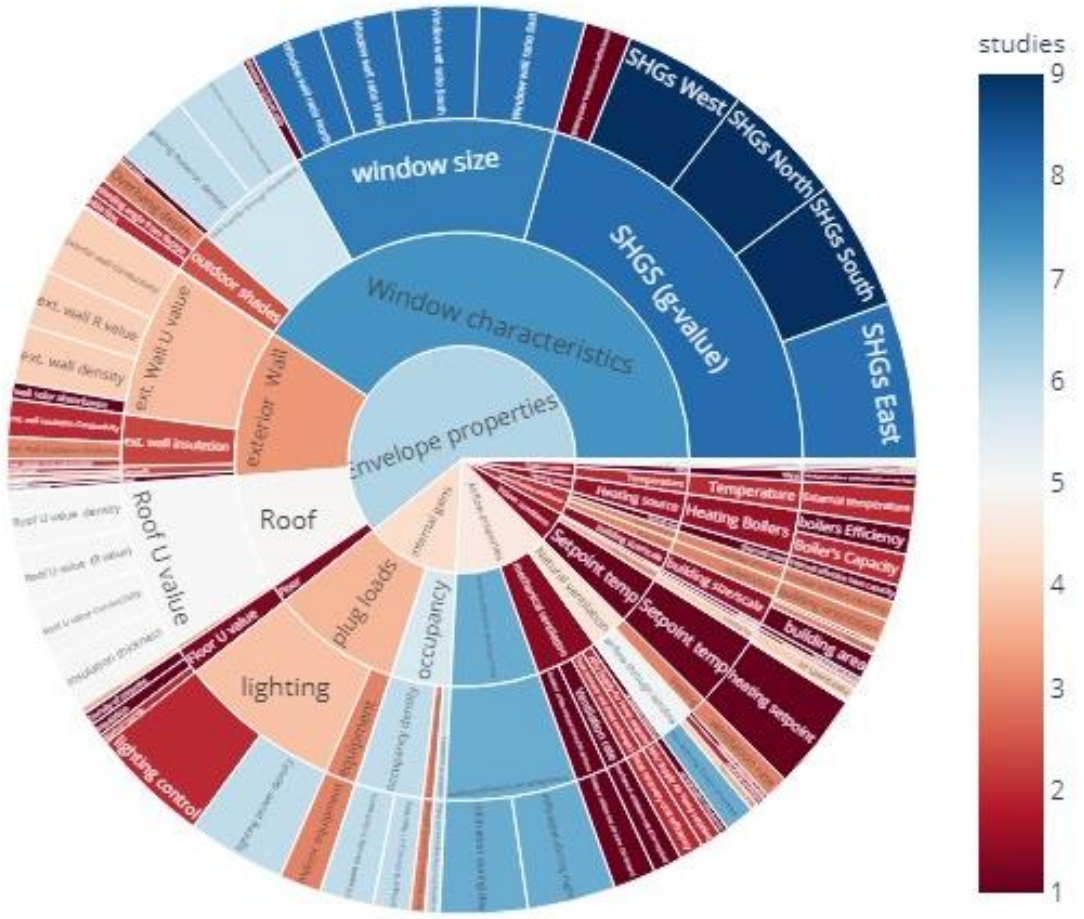


Figure 4-6 : Rank, and levels of detail of parameters influencing comfort and overheating in buildings

In Figure 4-6, the small circle in the middle shows parameters with level of detail one, and subsequent donut shape graphs show levels of details two, three, and four. Colour shows the number of papers that studied the parameter and the width of the segment indicates the relative influence of the parameter.

In energy consumption for heating purpose, WWR was identified as the most influential parameter, followed by solar heat gains (SHGs) (Figure 4-7). The next five influential parameters were building infiltration rate (airtightness), exterior wall U-value, roof U-value, heat transfer through the window and set point temperature in order of influence (Figure 4-7).

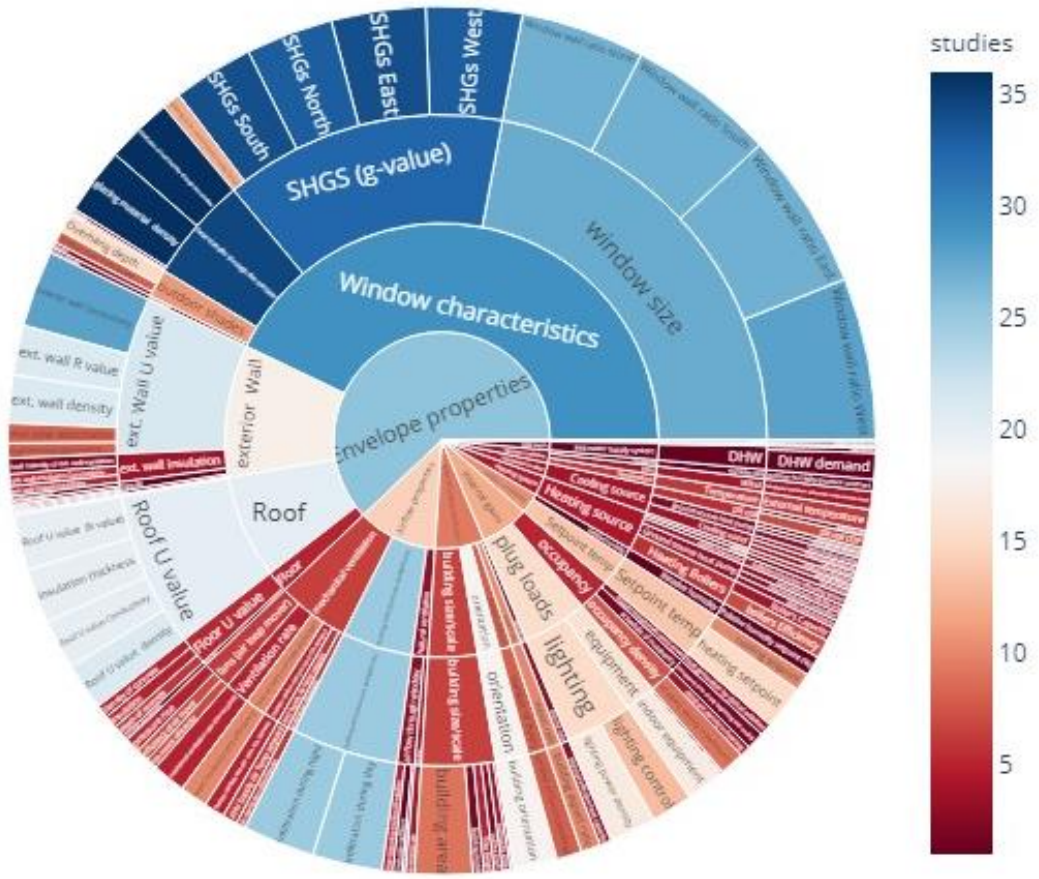


Figure 4-7 : Rank, and levels of detail of parameters influencing energy demand for heating and cooling in buildings

Most influential parameters identified in cooling energy consumption was WWR and was closely followed by SHGs. Exterior wall U-value, roof U-value, building size/scale, lighting power density, and efficiency of heat recovery unit were the next five influential parameters listed in order of rank.

Appendix 4-3, visualizes the results obtained from this literature review in more details.

4.1.3 Discussion points

Method employed in aggregation of parameters from different sources is not a standard method and was devised to answer the needs for this specific case. Alternative approaches may also exist that could possibly change the outcomes.

Selected papers only focused on building parameters. Urban morphological and microclimate parameters that also influence energy consumption and thermal comfort were not taken into consideration.

Sensitivity analysis results collected in these papers were conducted in multiple geographical locations with a wide range of climatic conditions, which in turn can also introduce inaccuracy in the study.

Some papers ranked the importance of parameter by giving it an index (percentage/ratio), some just presented ranks on a histogram and some named them in order of importance. In this brief review, only the order of influence/affect (first, second, third, etc.) of results and total number of parameters were taken into account.

The main gaol of this qualitative and quantitative literature review on sensitivity analysis case studies was to identify which building parameters contribute the most to variability of thermal indicator (heating energy consumption, cooling energy consumption, comfort). The results of this literature on building parameters influencing indoor comfort and overheating are used as a source in the selection of inputs for parametric simulations of selected representative buildings.

Based on the literature review in this section, input parameters deemed important to be considered for parametric simulations of free-floating residential buildings are as follows:

- Window size: this parameter varies for buildings that are in the same cluster and literature review showed it is very influential on both summer performance and energy consumption.

- Solar gains through the window: this parameter is closely related to widow size but unlike window size, it can be regulated by the status of external shading (window shutter) operated by occupants. Therefore, for parametric simulations the status of external window shutter and status of window for natural air inflow rate are modelled as a function of occupant type.

- Thermo-physical properties of envelop elements (external wall, external roof, adjacent walls, and type of window glazing)

- General characteristics of building: building size/scale.

4.2 Part Two: Overheating indices

4.2.1 Indoor overheating assessment indices

Researchers from different fields of science, including but not limited to agronomists, physiologists, epidemiologists, bio-meteorologists, over the years have proposed numerous methods to describe heat stress and overheating, but still there is no consensus on how to evaluate it through simulation or measurement (Lomas and Porritt 2017). Therefore, there is a need to specify what we mean by overheating or heat stress in buildings. Moreover, indoor overheating is dynamic and can vary spatially and temporally, meaning that indoor overheating can be different from one zone to another zone in a building and, from one time to another. Similar to thermal comfort, indoor overheating is subjective and perceived differently in various climates and with various adaptive measures.

A literature review on indoor overheating indices by (Epstein and Moran 2006) collected and analysed more than 40 different heat stress indices. Authors argue that too much emphasis has been placed on academic accuracy of many of these indices at the expense of practicality. They recommend using simple and easy to use indices, which although may lack the integration of many environmental variables, together with appropriate regulations that take into account the influences of clothing, indoor wind speed, and acclimation.

A summary of indices collected by Epstein and Moran in their review is in **Appendix 4-1** of this manuscript.

Most of those were designed for specific purposes, but among them, the following direct indices that rely on measurement of environmental factors gained, were discussed in more details by the authors and concluded to be more practical.

1- Effective temperature (ET) / Corrected Effective temperature (CET)
2- Wet-bulb globe temperature (WBGT)
3- Discomfort Index (DI)

Their literature review covered indices that were in use or proposed until 2005. Since then thermal indices have evolved and a number of new indices have been introduced. The most important change in that front is the adoption of adaptive comfort indices by ANSI/ASHRAE Standard 55 and ISSO 74 (Dutch Guidelines) in 2004 and 2005. European standard adopted it in 2007 in EN 15251. Adaptive indices were slightly modified and updated in ANSI/ASHRAE Standard 55 in 2017, and EN 16978-1 was introduced in 2019 to replace EN 15251.

In addition to adaptive indices other heat stress indices, which have gained popularity and are not mentioned in the list of indices illustrated in the literature review of Epstein et al., are as follows:

- Human temperature regulation two-node model to predict local skin temperatures of individual body parts, which is used as an indicator, by Gagge et al. originally proposed in 1986, and it has been integrated into *pythermalcomfort package* developed by Tartarini and Schiavan (Tartarini and Schiavon 2020).
- Cooling effect index (CE) described in ASHRAE 55 (ASHRAE-55 2017)

- Universal thermal comfort index (UTCI) for indoor and outdoor thermal conditions. This index was proposed by International Society of Biometeorology (ISB) Commission.
- Heat stress index derived from temperature and humidity ratio.

In the present manuscript, having the main objective of the thesis in mind, from the list of indices in **Appendix 4-1** and those presented above, **one** widely used thermal comfort index and **five** common indoor overheating indices were chosen to describe how buildings perform in summer. In the following sections, first, each index is briefly described and then through a case study, indices are compared to each other.

1- PMV/PPD index

Predicted Mean Score (PMV) – an indicator that predicts the average value of temperature sensitivity of a large group of people based on the balance of human body temperature on a seven-points scale.

The PPD is a measure that sets the predicted percentage of people who are dissatisfied with the ambient environment, who are too warm or too cold. People who are dissatisfied with the temperature of the environment are those people who will evaluate the environment as "hot", "warm", "cool" or "cold" on a seven-point scale of temperature sensitivity as presented in Table 4-1.

Table 4-1 : 7-Point Temperature Sensitivity

Scale score	Human feeling
+3	Hot
+2	Warm
+1	Slightly warm
0	Neutral
-1	Slightly cool
-2	Cool
-3	Cold

The relation between PPD and PMV is calculated with the following equation.

$$\mathrm{PPD} = 100 - 95 \cdot EXP(-0.03353 \cdot PMV^4 - 0.2179 \cdot PMV^2) \qquad \textit{Equation 4-3}$$

In accordance with ISO 7730:2005, the environmental condition is considered comfortable (neutral) if the PMV value is ±0.5 points, which corresponds to the PPD value ≤ 10%.

Input parameters in calculation of PMV are the following six factors:

- Air temperature [°C]
- Mean radiant temperature [°C]
- Air speed [m/s]
- Relative humidity of air [%]
- Metabolic rate [Вт/m^2]
- Thermal insulation of a set of clothes [clo]

Using indoor air temperature, indoor humidity ratio and the rest of parameters mentioned above, hourly PMV value in each thermal zone of buildings can be calculated within the Trnsys v.17 simulation environment.

2- EN 16798-1 adaptive index

Adaptive comfort index is based on the idea that connection and control over the immediate environment allow occupants to adapt to a wider range of thermal conditions. With reference to this principle, EN 16798-1 norm for adaptive comfort is related to exponentially decaying weighted mean outdoor temperature (T_{RM}).

$$T_{RM} = (1-\alpha)\,[T_{N-1} + \alpha\, T_{N-2} + \alpha^2\, T_{N-3} + \alpha^3\, T_{N-4} + \alpha^4\, T_{N-5} + \ldots]\ (°C) \qquad \textit{Equation 4-4}$$

Where: T_{RM} outdoor running mean temperature
T_{N-n} yesterday's mean daily temperature
α constant between 0 and 1

In calculation of weighted mean outdoor temperature, α constant controls the speed of changes in running mean outdoor temperature. Based on Smart Controls and Thermal Comfort (SCATs) project recommended value for it is 0.8 (McCartney and Fergus Nicol 2002). EN 16798-1 norm for adaptive thermal comfort index illustrates indoor thermal comfort from operative temperatures in three categories: category I, II, III.

Category I is considered more suitable for high demanding comfort level, category II for typical situations and category III for low demanding levels.

Following equations calculate maximum and minimum allowable indoor operative temperature limits (T_{MAX} & T_{MIN}) when 10 °C < T_{RM} < 30 °C.

Category I	Upper limit:	T_{MAX} (°C) =0.33*T_{RM} + 18.8 + 2	*Equation* 4-5
	Lower limit:	T_{MIN} (°C) =0.33*T_{RM} + 18.8 - 3	
Category II	Upper limit:	T_{MAX} (°C) =0.33*T_{RM} + 18.8 + 3	
	Lower limit:	T_{MIN} (°C) =0.33*T_{RM} + 18.8 - 4	
Category III	Upper limit:	T_{MAX} (°C) =0.33*T_{RM} + 18.8 + 4	
	Lower limit:	T_{MIN} (°C) =0.33*T_{RM} + 18.8 - 5	

Optimal operative temperature in EN 16798-1 equals to:

$$T_{COMFORT}\,(°C) = 0.33*T_{RM} + 18.8 \qquad \textit{Equation 4-6}$$

In calculation of comfort and overheating with this adaptive index, ***degree-hours*** and/or ***percentage-of-hours*** indoor operative temperature inside each thermal zone of building exceeding the upper/lower boundary limits in category I, II, and III could be calculated.

Figure 4-8 shows the relative position of temperature thresholds calculated with EN16798-1 equations.

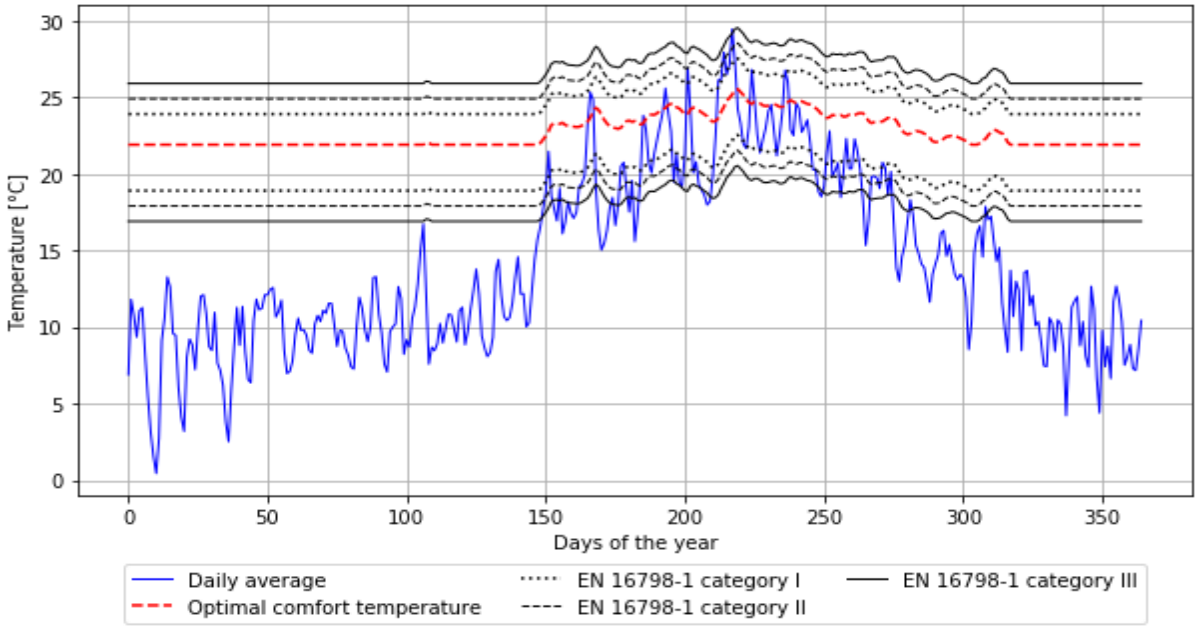


Figure 4-8: EN 16789-1 thresholds with reference to outdoor temperature from IPSL-SMHI (2040-2070) climate model

3- Givoni Bioclimatic index

Adoption of adaptive index was a major shift in the way thermal comfort and overheating was measured. A major element of concern with adaptive comfort has been the lack of relative humidity signal, and indoor air speed in preparation of the comfort range.

Givoni diagram could potentially address this major concern as it assess summer thermal comfort of occupants subjected to various indoor air velocities with consideration of humidity.

Another reason this index is selected, is that installation of a ceiling or a portable fan is the first response of occupants to extreme temperature in naturally ventilated houses when passive strategies fail to provide expected comfort.

Temperature, humidity ratio and indoor air velocity are the main parameters involved in comfort evaluation with this index.

In Givoni bioclimatic index, as depicted in Figure 4-9, thermal comfort polygons are presented on a psychometric chart (Malet-Damour 2012).

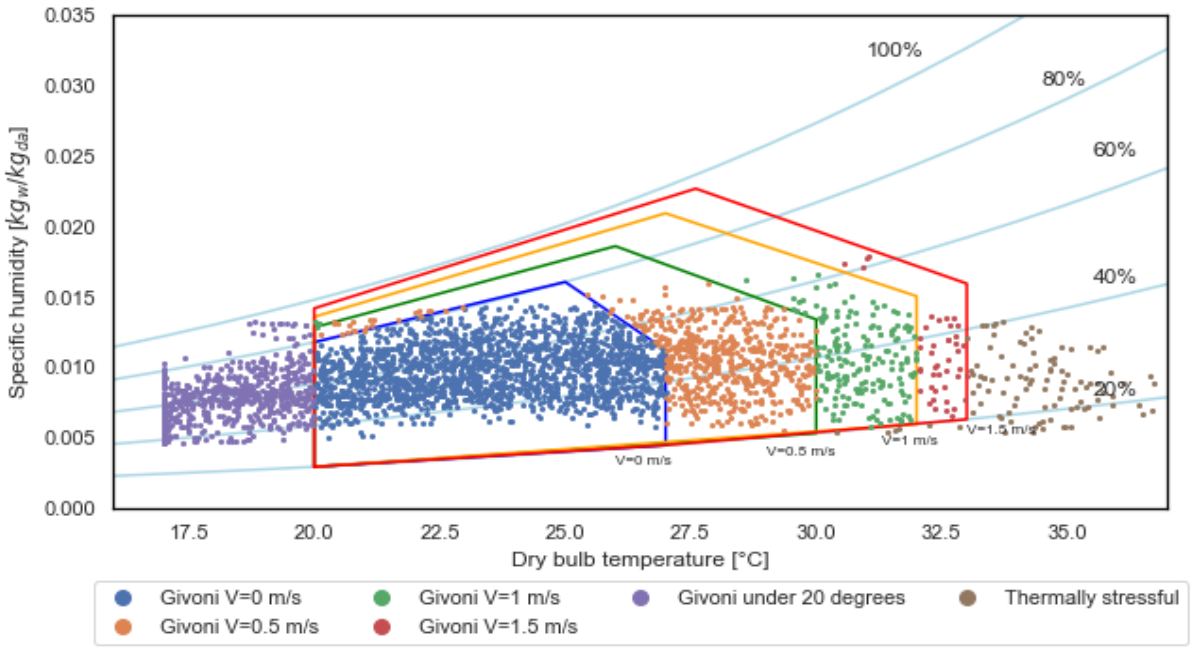


Figure 4-9 : Psychometric chart and Givoni bioclimatic design polygons

The lines of polygons, shown above, determine the limits of effectiveness of design strategies for indoor thermal comfort in relation to indoor wind speed for an occupant engaged in a sedentary activity wearing summer clothing (0.5 clo). Psychometric chart lines, on which thermal comfort polygons are drawn, are related to the air pressure. The latter is estimated using barometric formula as a function of height above sea level. Average height of our case study city from sea level is around 30 meters.

$$p = 101325\ (1 - 2.25577 * 10^{-5}\ h)^{5.25588} \qquad \textit{Equation 4-7}$$

Where: h Average height above sea level [m]

p Air pressure [Pa]

Ideal comfort region with this index is the polygon blue, where air temperature is between 20 to 27°C, relative humidity is between 20 to 80%, and indoor air velocity is 0 m/s. Second polygon (green) is called natural ventilation comfort zone, where indoor air temperature is between 20 to 30°C, relative humidity is between 20 to 90%, and air velocity is up to 0.5 m/s. Air velocity in the third polygon (orange) is 1m/s and it is induced both by natural ventilation and ceiling fans. Indoor air temperature in the third polygon is between 20 to 32°C and relative humidity is between 20 to 94%. We added the fourth polygon (red), where air speed is 1.5 m/s, for extreme cases, because it has been used and considered acceptable in some hot and warm countries (Nicol 1974). (Kumar et al. 2019) in their paper argue that marginal temperature gain of air velocity increase from 1 m/s to 1.5m/s is less than 1 °C, therefore, maximum temperature of polygon four is 1 degree higher than polygon three. (**Appendix 4-2** presents the Python script to plot or modify polygon boundaries of Givoni index)

4- Fixed temperature thresholds

Fixed temperature thresholds are probably the simplest way to measure indoor or even outdoor overheating, especially when reporting the indoor/outdoor thermal conditions in a specific time. For example, using statements such as, temperature is over 28 °C in the living room or it was above 29 °C in the living yesterday afternoon. Even when time duration is longer than an hour or day, practitioners need to specify time for this index to convey a meaning about the temperature condition in a specific location. In comparative cases, measures such as number of hours or percentage of hours temperature, in a specific location, is/was over 28 °C, during the 5 summer months, could also be used. In here, 28 °C is used as an example; any constant temperature could be used. Building KM7_5 was simulated in TRNSYS and indoor overheating performance of building, as measured by the number of hours above various fixed temperature thresholds, was evaluated. Figure 4-10 shows number of hours above the selected thresholds.

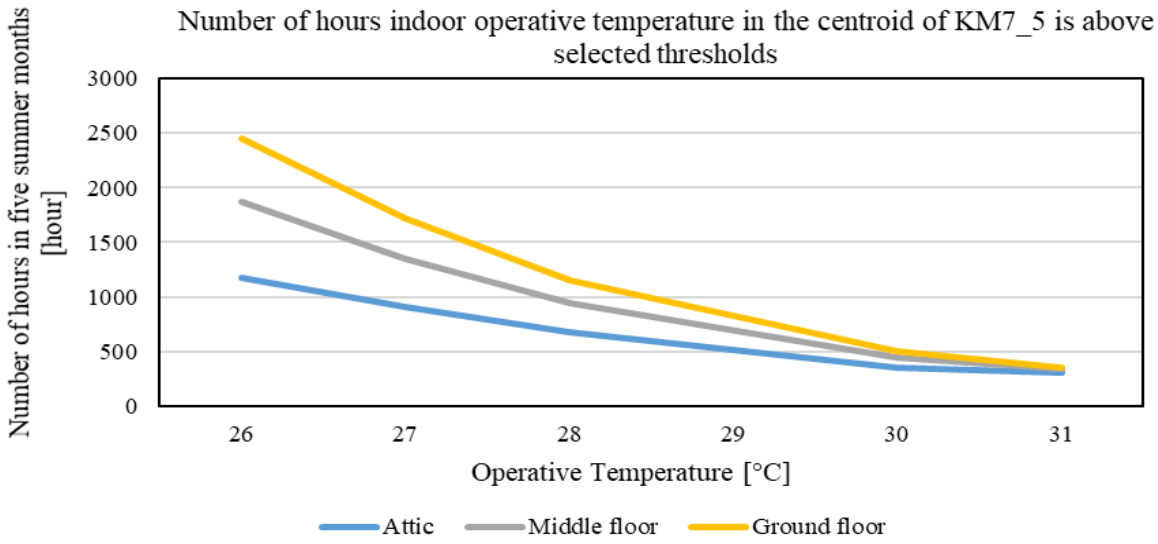


Figure 4-10 : Number of hours indoor operative temperature was above the fixed temperature thresholds

The Figure 4-10 shows the number of hours temperature is above the specified threshold in each storey of KM7_5 building. According to the graph above, if a practitioner decides to use fixed temperature of 27°C as a measure of discomfort/overheating threshold then Attic has performed better than middle floor and ground floor, over the five summer months. Higher number of hours when temperature is above 26 °C and 27 °C shows that ground floor and middle floor have larger potential to store heat, therefore, keeping temperature stable.

Significant difference on the performance of attic, middle floor and ground floor also points out to the problem of using the fixed temperature as an index for overheating.

Another way to use fixed temperature threshold is by calculating maximum number of consecutive hours it is above a specific threshold. The idea of using consecutive hours above a threshold was also analysed and discussed in the thesis of Nicolas Lauzet (Lauzet 2019). In this study, the constant temperature of 27 °C as an index of indoor over-temperature was chosen. It was selected with reference to the night-time temperature threshold of 26 °C proposed by

Chartered Institution of Building Services Engineers (CIBSE). According to CIBSE, night-time indoor temperature should not exceed 26 °C more than 1% of annual hours for occupants to sleep well. Considering the adaptive capacity of occupants, difference between the climate of UK and France, mild relative humidity of our case study city, and the fact that in this study it is used both during day and night, it was decided to use 27 °C instead of 26°C as the threshold value to measure comfort along with other indices.

Using maximum consecutive hours above a threshold allows depicting cumulative intensity of overheating. However, showing intense temperature increase over a short span of time requires consideration for peak indoor operative temperature as well. Having said that, the following graph, Figure 4-11, shows maximum consecutive hours operative temperature is above 27°C and peak indoor operative temperature.

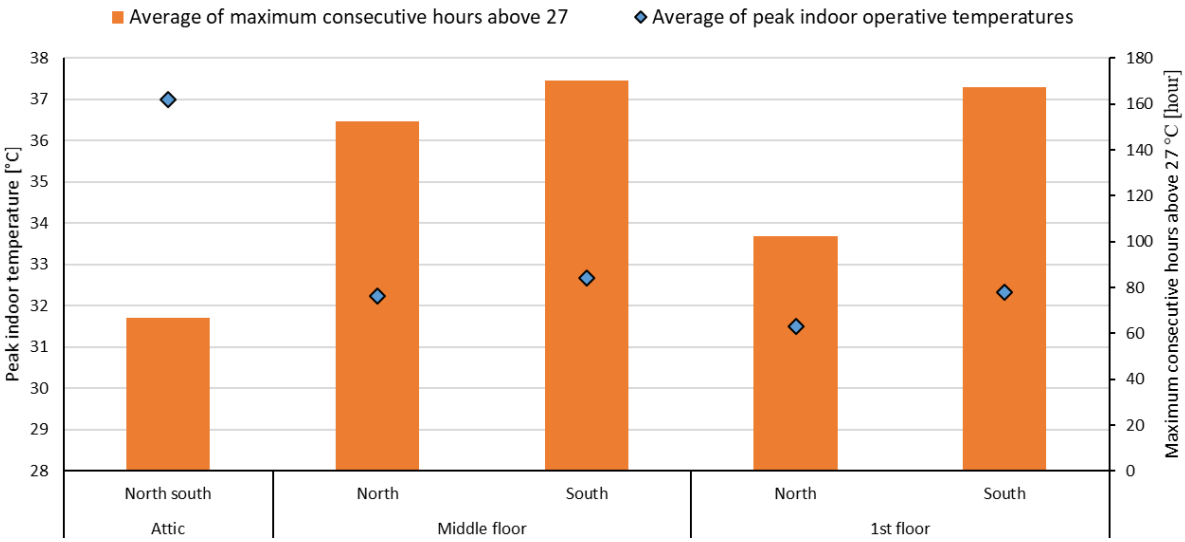


Figure 4-11 : Peak indoor operative temperature and maximum consecutive hours above 27°C in each floor of building KM7_5

The Figure 4-10 and Figure 4-11 show drastic difference on the way indoor temperature of the same building is explained. It is clear from Figure 4-11 that peak temperature is considerably higher in attic compared to the peak temperature in the thermal zones of other stories. Additionally, maximum consecutive hours show a considerable difference with regard to the orientation of the zone. (Orientation here means the orientation of main façade of the zone).

Absence of indoor air speed and relative humidity are probably the main shortcoming of performance analysis using this index.

5- Heat stress index

Heat stress index (HI), first proposed by (Steadman 1979), is also called as apparent thermal comfort temperature, and is used as an index to predict the risk of physiological heat stress in an individual taking into account temperature, relative humidity and type of activity. The main objective of this index is to explain how an individual would feel under certain environmental conditions. This index, assumes that relative humidity has a considerable impact on how an

individual would feel because it influences the capacity of human body to regulate internal heat through perspiration (sweating). Body feels hotter when relative humidity is high and perspiration evaporates slower. Therefore, HI is high where relative humidity is high. At low relative humidity, HI is less or equal to air temperature.

This index is measured in the shade, and assumes a wind speed of 2.5 m/s and normal atmospheric pressure.

The equation below has been derived from multiple regression analysis performed by (Steadman 1979) to calculate heat index.

$$HI = -8.784695 + 1.61139411 \cdot T + 2.338549 \cdot RH - 0.14611605 \cdot T \cdot RH - 1.2308094 \cdot 10^{-2} \cdot T^2 - 1.6424828 \cdot 10^{-2} \cdot RH^2 + 2.211732 \cdot 10^{-3} \cdot T^2 \cdot RH + 7.2546 \cdot 10^{-4} \cdot T \cdot RH^2 - 3.582 \cdot 10^{-6} \cdot T^2 \cdot RH^2$$

Equation 4-8

Where: RH Relative humidity [%]

T Dry bulb temperature [°C]

HI is designed for temperature above 20°C and they are categorized according to the following table to describe possible physiological effects caused by heat in people.

Table 4-2 : Heat index range and possible effect of within each range

Index range (HI)	**Category**	**Possible effects for people in high risk**
≤ 27	No hazard	-
27-32	Caution	Possible fatigue with prolonged exposure
32-41	Extreme Caution	Sunstroke, muscle cramps, and heat exhaustion possible with prolonged exposure
41-54	Danger	Sunstroke, muscle cramps, and heat exhaustion likely. Heatstroke possible with prolonged exposure.
≥54	Extreme danger	Heat stroke or sunstroke likely

6- Discomfort index

Discomfort index has been used, as stated by (Epstein and Moran 2006), as a substitute to Wet Bulb Globe Temperature (WBGT). Mainly because, direct onsite measurement of WBGT is not practical for buildings. The equation bellow, is used to calculate discomfort index.

$$DI = 0.5 \cdot T_w + 0.5 \cdot T_a$$

Equation 4-9

Where: Tw Wet Bulb Globe Temperature [°C]

Ta Dry bulb temperature [°C]

Tw is called in some resources as aspirated (psychometric) wet-bulb temperature. This measure is not available from weather station data. Therefore, the equation below, proposed by (Stull 2011), is used to estimate it as a function of air temperature (Ta) and relative humidity (RH %).

$$T_w = T_a \cdot atan\left[0.151977(RH + 8.313659)^{\frac{1}{2}}\right] + atan(T_a + RH) - atan(RH - 1.676331) + 0.00391838(RH)^{\frac{3}{2}} atan(0.023101 \cdot RH) - 4.686035$$

Equation 4-10

Where: RH Relative humidity [%]

Ta Dry bulb temperature [°C]

Various weighting factor of Tw in Equation 4-9 has been used by authors, from 0.5 to 0.85. Literature review by (Epstein and Moran 2006) states the coefficient of determination (R^2) of DI with WBGT is above 0.95 for all weighting factors. This index assumes occupants are wearing light summer clothing.

Similar to HI, specific thresholds are proposed for DI to describe possible physiological effects of over temperature on human body.

Table 4-3 : Discomfort index (DI) ranges

Range	Category
< 21	No discomfort
21-24	Less than 50% feel discomfort
24-27	More than 50% feel discomfort
27-29	Most population feel discomfort
29-32	Everyone suffer sever discomfort
> 32	Medical support required

4.2.2 How related indices are to each other?

In order to show the correlation between illustrated indices, indoor thermal condition in the attic of KM7_5 was calculated for all indices over a period of 5 months including summer, and the outcomes are plotted against each other.

In PMV calculation, indoor air speed was assumed 0.1 m/s, occupant performing a light physical activity like sitting, and clothing level was assigned to the simulation according to the Figure 4-12, below. Assigning clothing level in this manner enables to take into account the influence of blanket that occupants use at night.

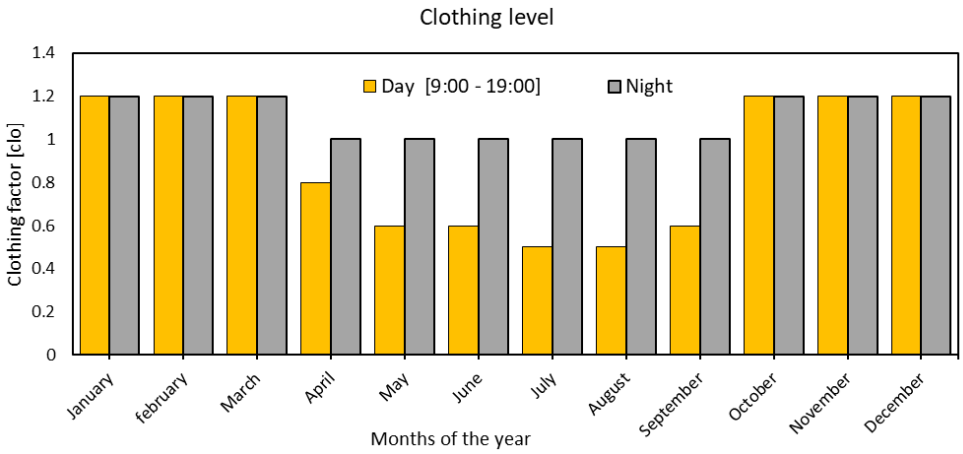


Figure 4-12 : Clothing level of occupant used in calculation of PMV index

Figure 4-13 shows the sensation of occupants according to PMV, Givoni Bioclimatic, HI, and DI for attic of KM7_5 for the 5 summer months of the year. These indices were selected for comparison because they have defined thresholds that categorize sensation of occupants unlike EN16798-1 or consecutive hours above a specific temperature, which are continuous measures of thermal sensation.

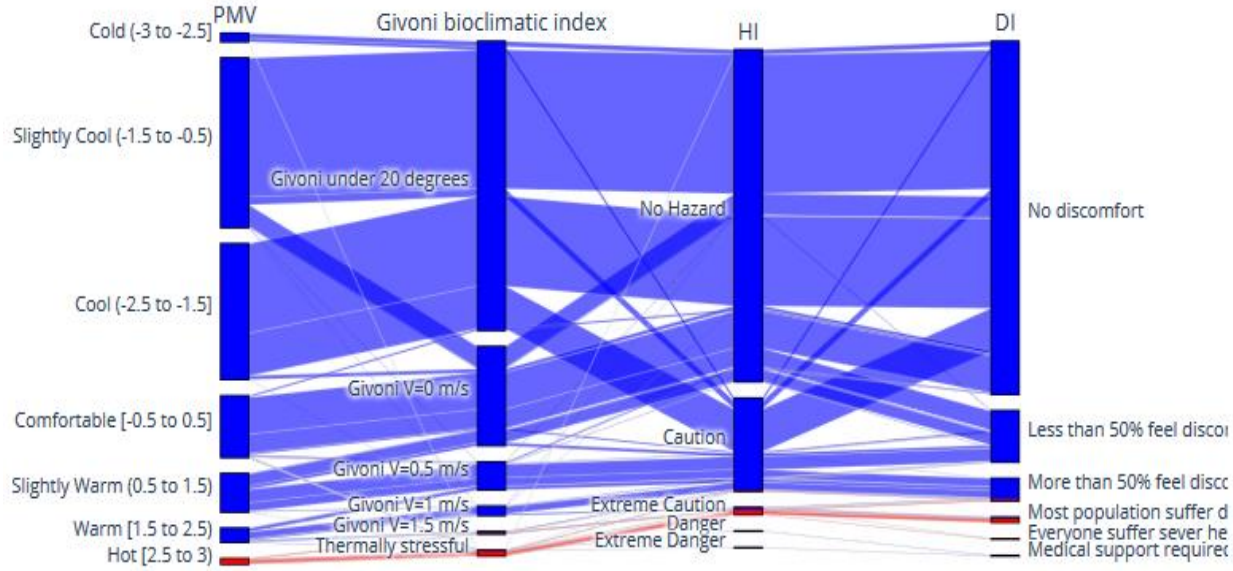


Figure 4-13 : Comparing indices for Attic of KM7_5

The graph shows that the hours in “Hot [2.5-3)” sensation scale are listed as “thermally stressful” region in Bioclimatic Index, “Extreme caution” region in HI, and “Most population suffer” range in DI. The graph shows that the last sensation range with PMV and GIVONI index correspond to third sensation range with HI and DI.

This could mean that DI, and HI have more divisions (categories) for temperatures that are extremely high. Meaning, DI and HI could better quantify the sensation of people under extreme conditions compared to GIVONI and PMV.

To better demonstrate this difference, the hourly values of PMV and GIVONI are plotted against the DI and HI indices in the following figures:

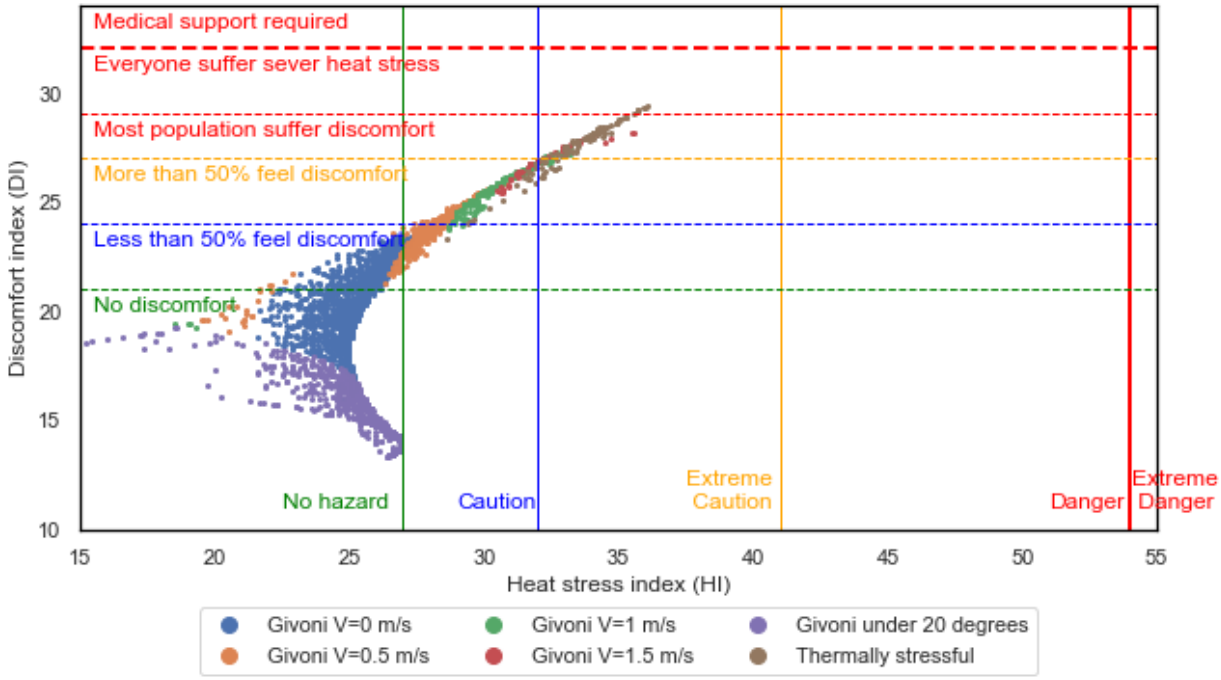


Figure 4-15 : GIVONI Bioclimatic ranges compared to DI and HI ranges

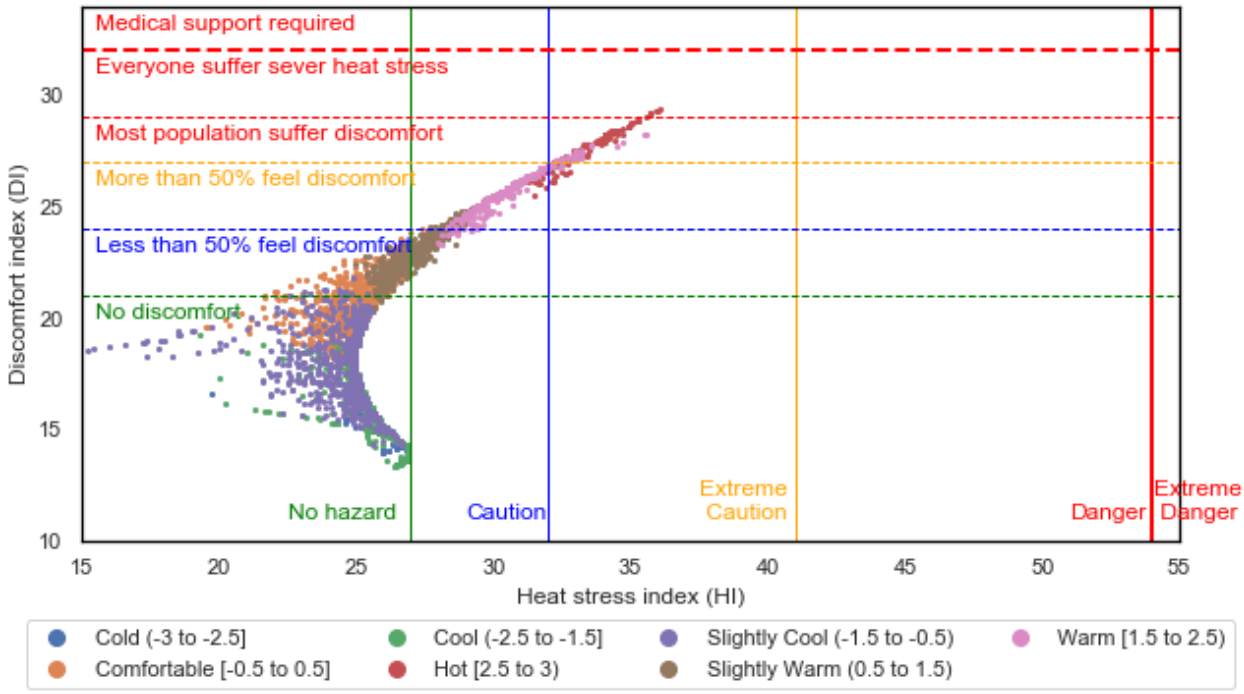


Figure 4-14 : PMV sensation ranges compared to ranges with DI, and HI

Figure 4-14 and Figure 4-15 also illustrate that DI and HI are strongly correlated for index values above 30 in HI and above 25 with DI.

Figure 4-16 and Figure 4-17 further demonstrate how indices describe the relation between an individual and his/her environment for the Attic of KM7_5.

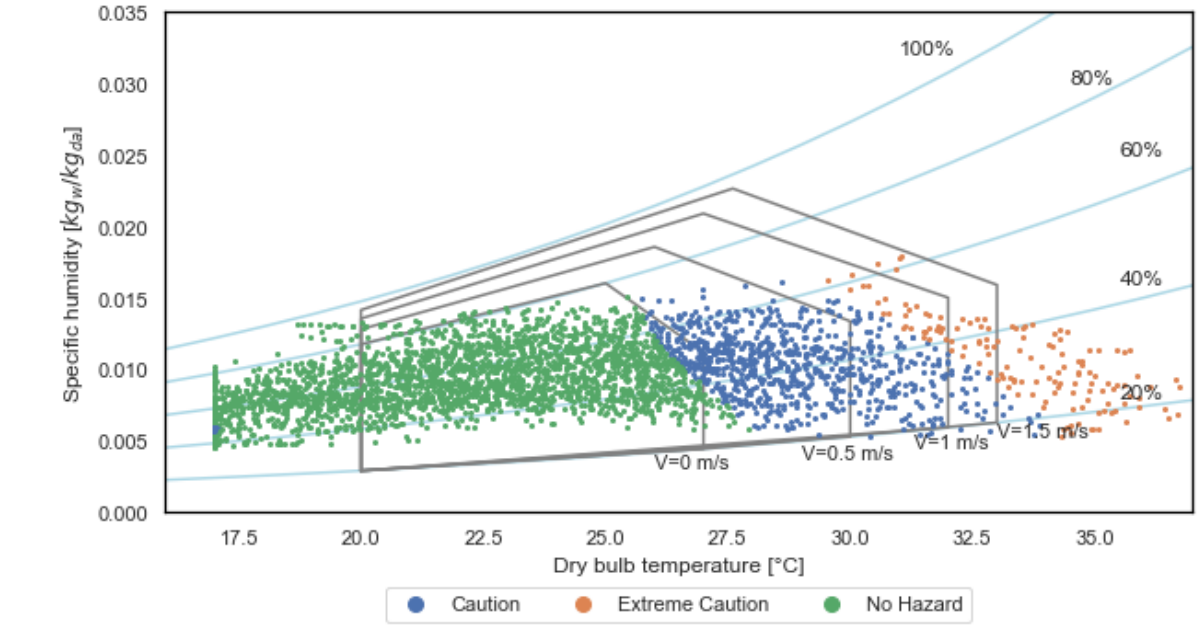


Figure 4-17 : HI ranges compared to GIVONI ranges

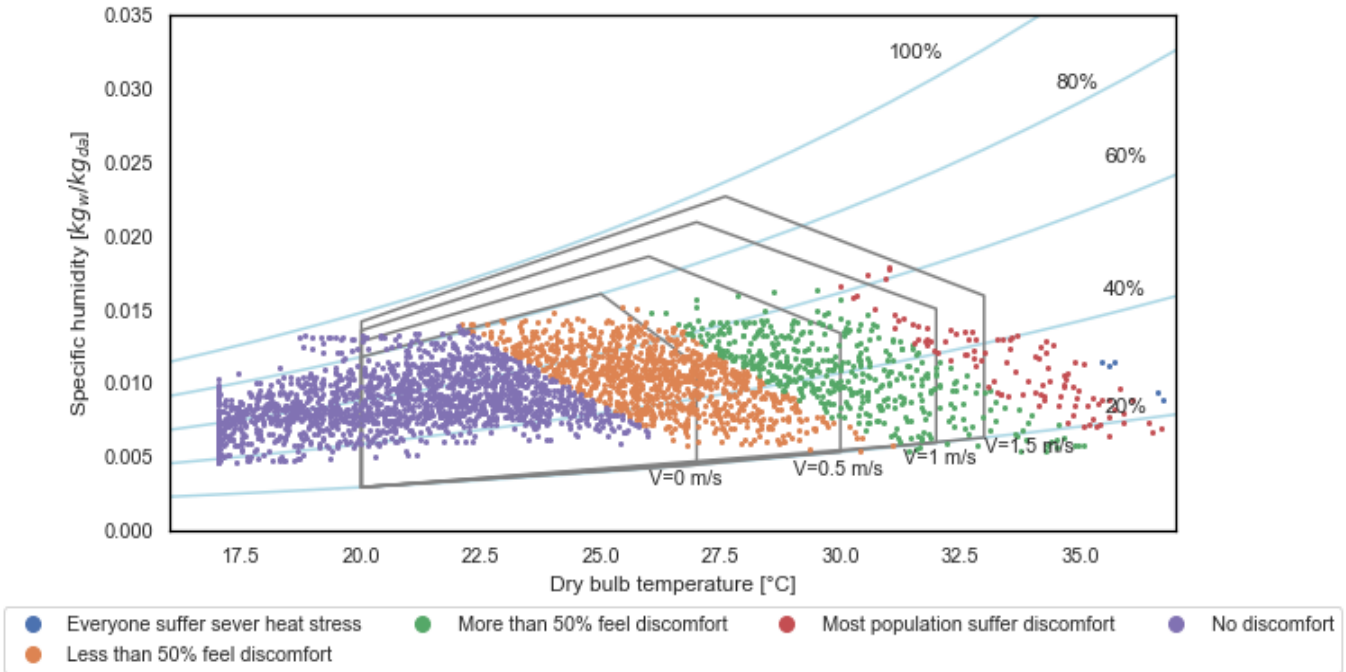


Figure 4-16 : DI ranges compared to GIVONI ranges

From given indices we notice that some indices give complementary results and highlight certain aspect of occupants' sensation that are ignored or given less attention by other indices. For instance, peak operative temperature in a thermal zone and maximum consecutive hours above a certain threshold complement each other. Peak operative temperature show how

quickly a zone gets heated up and consecutive hours above shows how long does this specific zone hold the heat. Givoni index on the other hand, takes into consideration the effect of relative humidity and indoor air speed to create a picture of discomfort.

4.2.3 How to know/define if a building is vulnerable to overheating?

The indices presented in section 4.2.1 provide valuable information on how to describe the occupants' perception of indoor thermal condition, but the question of vulnerability to overheating also depends on factors such as adaptive capacity of occupants, their health conditions, and many more that were discussed in details in chapter one of this manuscript.

Here the focus is on the status of indoor environmental conditions, assuming a standard occupant. On this premise, one can infer that the degree of vulnerability is influenced only by environmental conditions and the measurement index.

Some standards have defined thresholds, above which they categorize a building (a thermal zone) to be overheated. In the following, some of the most relevant ones for free-floating buildings are briefly described.

1- CIBSE

CIBSE TM59 uses a pass/fail approach to test if a building (or a thermal zone of building) meets the criteria or not. Requirements are different for bedrooms, kitchen, and living room for naturally ventilated buildings. With TM59, the condition of approval is fulfilled when a building passes both (A, and B) of the following criteria.

Table 4-4 : CIBSE TM59 and TM52 criteria

Criteria	**Room (zone)**	**Compliance condition of criteria**
A	Living rooms, kitchen and bedroom	Number of hours that operative temperature is greater than maximum allowable operative temperature according to adaptive comfort index (category II for normal expectations), shall not be more than 3% of *occupied hours* during the period of May to September (5 summer months).
B	Only bedrooms	Operative temperature in a bedroom between 22:00 and 7:00 [10 pm to 7 am] should not be above 26°C more than 1% of annual hours (1% of annual hours equals to 32.85 hours).
CIBSE TM52 also has a third criteria that deals with the severity of overheating within any one day, but is not present in TM59		
C	Living rooms, kitchen and bedroom	Indoor operative temperature shall not be greater than **6°C** from maximum allowable operative temperature within any given day.

It is also important to mention that the passing condition with TM52 is, if a room (thermal zone) meets at least two of the three criteria, it passes. (Tm59 design methodology for the assessment of overheating risk in homes. 2017), (Chartered Institution of Building Services Engineers 2013).

2- RT2012

According to RT2012 (French building thermal regulation for new buildings until 2022), the indoor temperature in summer, called TIC, for five consecutive days must be lower than conventional reference temperature. This reference temperature is most of the time 26 °C.

3- RE2020

‘‘Réglementation Environnementale”2020 (RE2020) came into effect on 1st January 2022 in France for some of the new buildings and will gradually come into effect for all buildings on a timeline, replacing RT2012. With RE2020 government attempts to follow the following three key objectives: (1) prioritize energy sobriety; (2) reduce carbon impact of the construction; (3) guarantee comfort during extremely hot summer temperatures (under climate change and/or heatwaves).

Two principle elements in RE2020 for summer are: (1) it takes into account typical climate (année classique) and heatwave (année caniculaire) weather data in building design simulation, and (2) it encourages the usage of passive strategies relying on bioclimatic design of buildings, to avoid or delay the installation of mechanical cooling system.

RE2020 uses Degree-Hour index to evaluate discomfort in new buildings. The reference temperature for it is the maximum allowable operative temperature in category II of EN 15251 adaptive comfort, which is to be calculated based on a conventional heatwave scenario. Adaptive limit is only applicable during the day [7h-22h] and not at night-time.

RE 2020 defines 2 thresholds that the temperature inside the building must not exceed to avoid any discomfort: (1) at night, the temperature threshold of 26°C; (2) during the day, an adaptive temperature threshold between 26° and 28°C. Beyond these thresholds, each degree of the building is considered uncomfortable for the occupant. During the day, this threshold is constant but it is not necessarily identical to that of the previous day. It varies from day to day to take into account the capacity of the human body to adapt to high temperatures after a succession of hot days, within the limit of + 2°C compared to the consensual threshold of 26 °C.

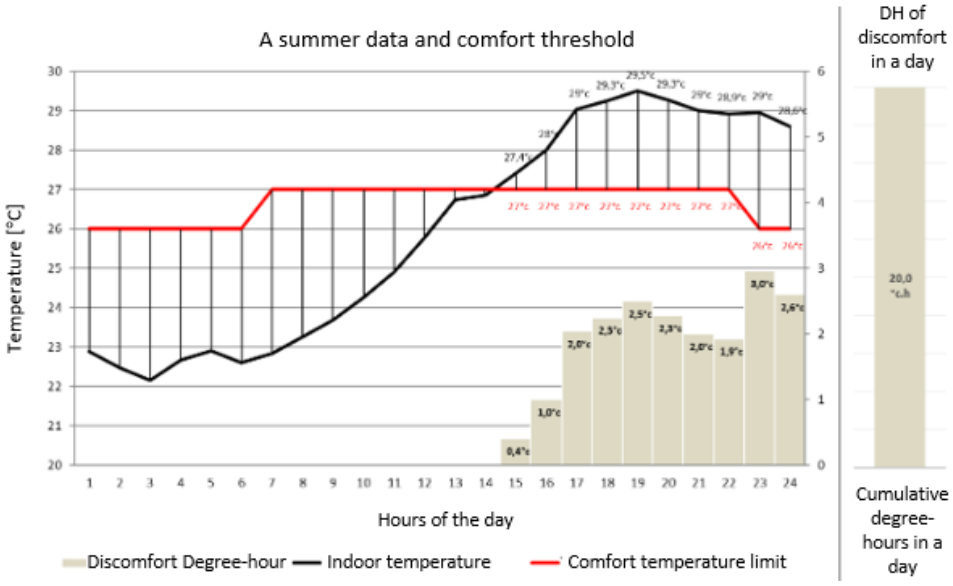


Figure 4-18 : Example of degree-hour calculation without consideration for occupied hours (source: modified from CEREMA publications)

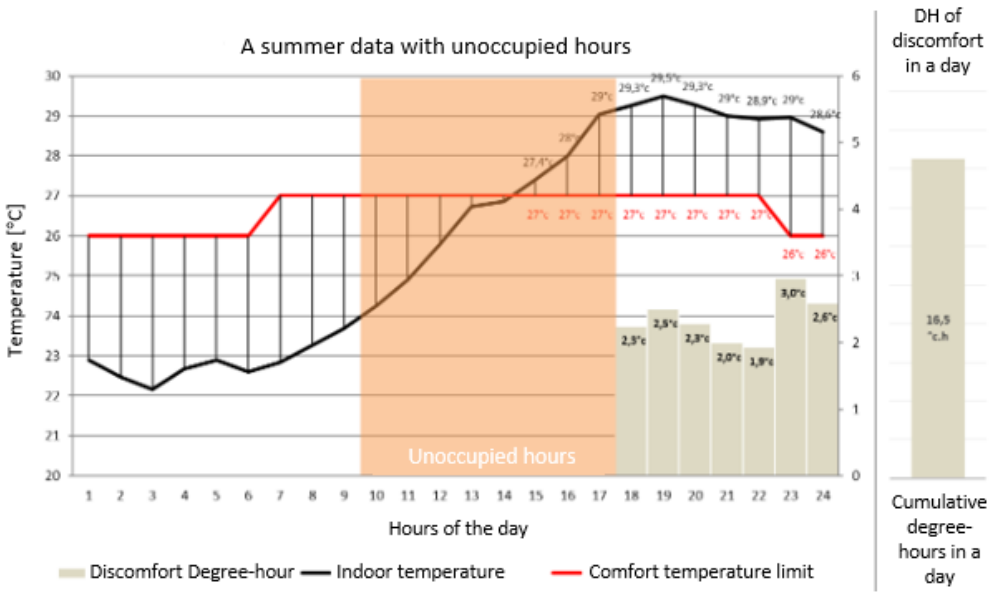


Figure 4-19 : Example of degree-hour calculation with consideration for occupied hours (source: modified from CEREMA publications)

RE2020 characterizes buildings in three situations: situation "confortable", situation "risque d'inconfort", and situation "non réglementaire". To define these situations, it presents two thresholds based on degree-hours [hour-°C]:

(1) A lower boundary threshold of **350** degree-hour [hour-°C], below which a building is regulatory.

(2) An upper boundary threshold that can vary depending on the category and exterior constraints.

Exterior constraint could be something like the presence of noise in the area that prevent cooling by window or extremely hot climate.

Category 2 (RE2020 definition): Air-conditioned building + Residential use + BR2 or BR3[7] + H2d or H3 (hot climate regions as depicted in Figure 4-20) + Altitude [0; 800m]

Category 1 = no external constraint, i.e., whatever is not in Category 2 according to the definition of RE2020

Upper boundary threshold is also different for collective and individual house as presented in Table 4-5 and Table 4-6.

Figure 4-20: Climate regions in France

[7] BR3, BR2, and BR1 are levels of noise:
BR1 is a weak exposure to noise, BR2 is a medium exposure that prevents users to open windows in summer, and BR3 is a level of noise for which it is obligatory to install acoustic insulation.

*Table 4-5: RT2020 summer comfort in **Individual houses** (attached and detached)*

	Category I (without exterior constraint)	Category II (with exterior constraint)
Upper boundary threshold [degree hour] [hour – °C]	1250	1850

*Table 4-6 : RT2020 summer comfort in **collective houses** (attached and detached)*

	Upper boundary thresholds degree-hours [hour – °C]		
	Category 1, except parts of air-conditioned buildings in zones H2d and H3	Category 1 air-conditioned houses: For zones H2d and H3	Category 2
Average Area ≤ 20m^2	1250	1600	2600
20 m² < Average Area ≤ 60 m²	1250	1700-5 * Area	2850 - 12.5 * Area
Average Area > 60 m²	1250	1400	2100

If degree-hour is in between the lower and upper boundary comfort thresholds, the building still complies with the regulatory requirement of RT2020, but it is encouraged to use passive and/or bioclimatic solutions in the summer period.
Source: (Publications of CEREMA)

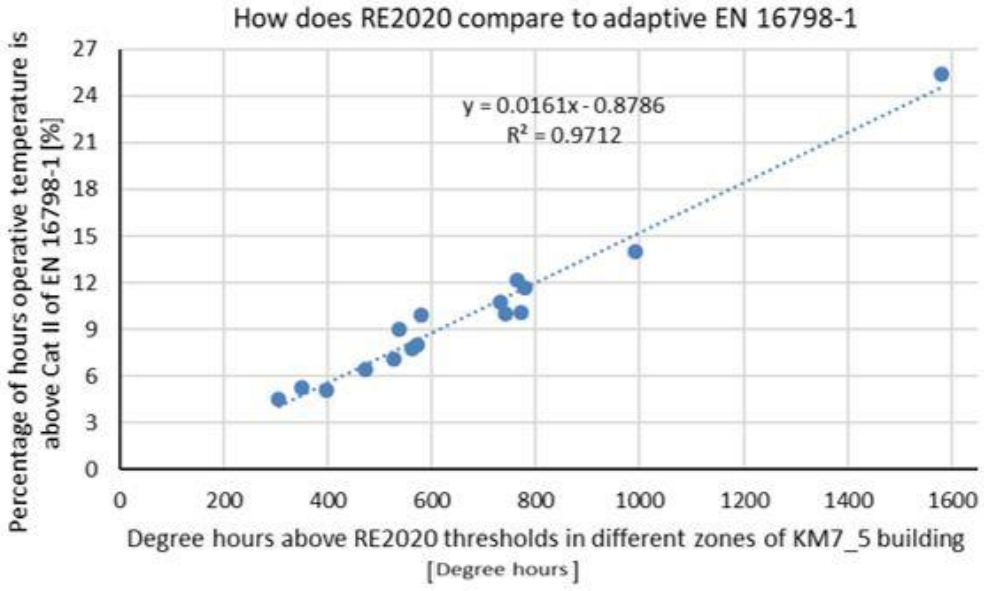


Figure 4-21 : Degree-hour in RE2020 compared to percentage of hours above category II and EN 16798-1 in 5 summer months

The 3 to 6% range in percentage of hours operative temperature is above maximum allowable operative in EN 16798-1 threshold correspond to 200 to 400 degree-hours of RE2020 in different zones of building. This shows underlying similarity between RE2020 350 degree-hours threshold and 3% threshold of TM59.

A major shortcoming of TM59 and RE2020, is that they both only concentrate on temperature. Relative humidity and indoor air speed are not being considered in overheating assessment.

HI and DI, indices were designed for extremely hot climate regions, they may not be able to describe the building in oceanic climate region.

Vast amount of indices, standards, and data from building simulations can be overwhelming for decision-makers and other stakeholders, especially if it is implemented at the scale of city. Therefore, there is a clear need to list critical attributes of an index or indices that could describe indoor thermal conditions of a thermal zone and the whole building.

Based on comparative analysis of indices, their individual behaviour, it is noticeable that many of the indices provide complementary results and some of them describe certain aspects of building that are ignored of given less attention in other indices.

This means, practitioners in BPS could create a better picture of indoor performance of buildings at city scale if they use more than one index to describe it.

Given the influence of multiple factors on vulnerability of occupants, a suggestion of this manuscript is that the indices should be able to describe the following characteristics indoor conditions to present a realistic picture of indoor thermal conditions:

- Intensity of indoor operative temperature (maximum operative temperature);
- Duration of over temperature (maximum consecutive hours/days over a threshold);
- Adaptive capacity of occupants including their absence and/or presence;
- Take into account the influence of indoor air speed, and relative humidity;

Having the global objective of this manuscript in mind, 6 indoor overheating indices, described below, were selected to be used in estimation of the degree of discomfort in residential buildings at city scale.

- Percentage of hours indoor thermal condition of a zone is outside the Givoni index's red polygon (v=1.5 m/s) to depict scenarios where having a simple fan or relying on passive strategies are not sufficient.
- Peak indoor operative temperature to show the intensity of overheating
- Percentage of hours above Category ii of EN 16798-1 to take into consideration adaptive capacity of occupants
- Maximum number of consecutive hours temperature is above 27 degrees to show duration of over temperature
- Degree-hours according to RE2020 to take into account adaptive capacity of occupants and time of exposure.
- RE2020 situation categorical outcomes (comfortable, at risk, non-regulatory) to give categorical label to each thermal zone with regard to their summer performance.

4.3 Summary

This chapter is divided into two parts. The goal of first part is to identify, through aggregation of sensitivity analysis studies, which input parameters contribute the most to variability of outputs in buildings (energy consumption and summer performance) and rank their relevance and relative importance based on what is found in the literature.

For that, specific keywords were used to identify the papers and from those articles, a table containing the required information was built. After that, aggregation was carried out through a simple method that takes into consideration the individual rank of parameter, total number of parameters in each sensitivity analysis, and the number of papers studying the parameter. Using the illustrated method an aggregated list of parameters for energy consumption and indoor thermal conditions at different levels of details were prepared and described in the results.

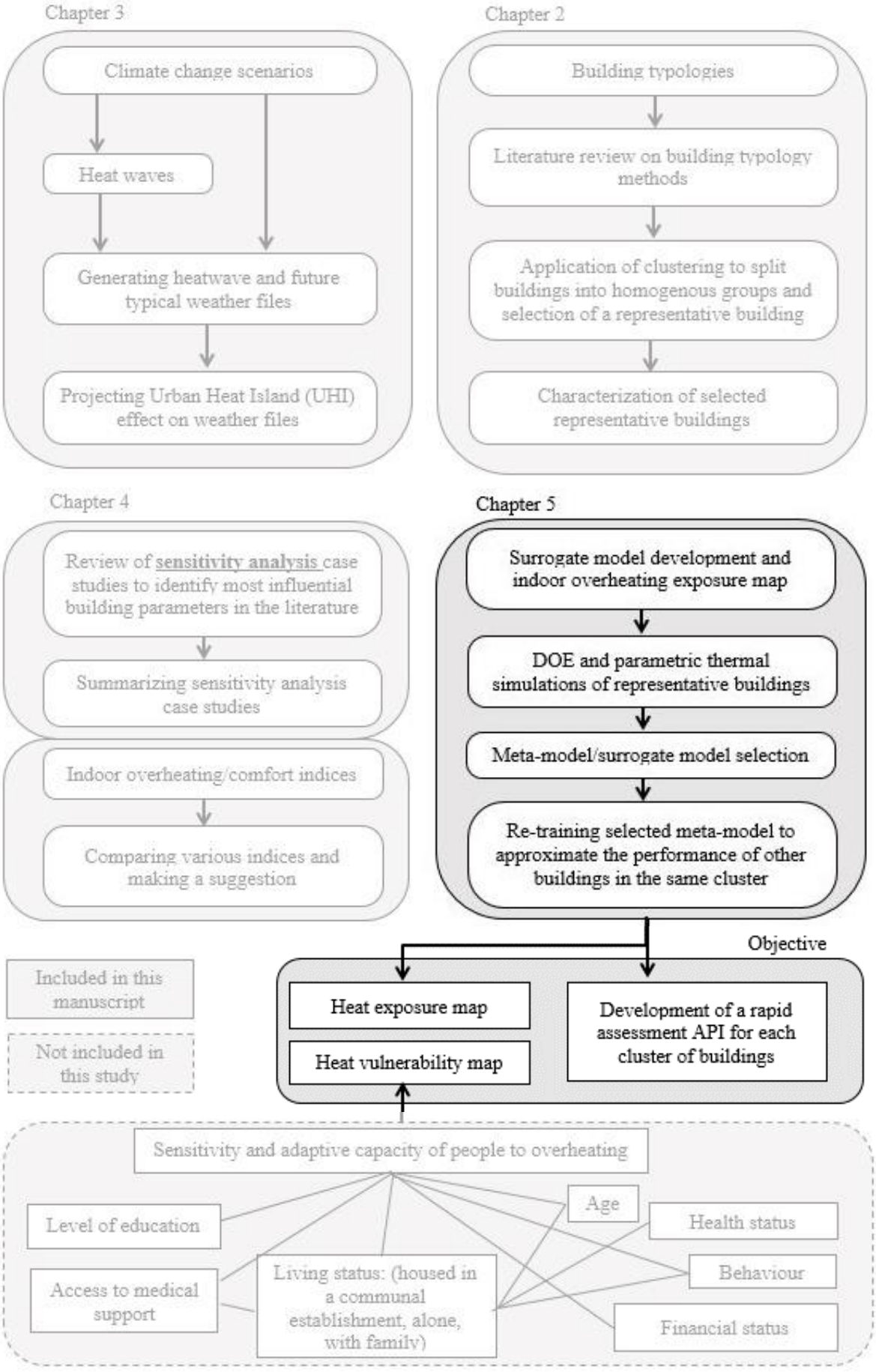
The goal of the second part of this chapter is to specify what is indoor overheating and how to measure it. This part starts with a literature review and then describes in details a few indices frequently used in assessment of indoor thermal conditions.

As an example, indoor thermal conditions of attic of cluster KM7_5 was calculated, for 5 summer months, with all those indices and the outputs are compared to one another to observe how each index describes indoor thermal conditions of the same thermal zone and which one provides what type of information for practitioners.

At the end, it was decided to use 6 indices that provide complementary results, to approximate indoor thermal conditions of buildings at city scale.

Overall, the objective of this thesis is to create indoor overheating map at city scale, taking into consideration climate change data and UHI effect. To that end, the second chapter of this manuscript, illustrated the preparation of a build stock database and from it, identified 7 representative buildings. Third chapter provided a workflow to access climate data, create future typical weather files, and project the UHI effect on the weather file of each representative building. This chapter, through a literature review helped identify what other building parameters need to be considered for summer performance evaluation of buildings and what indices do practitioners/researchers can use to better describe indoor thermal conditions.

The next chapter will describe the method to extend the application of dynamic thermal building simulations on typical/reference residential buildings to the rest of build stock for the case study city, Nantes, and will prepare a citywide urban heat exposure/vulnerability map. For that, it performs a parametric simulation on each representative building (by varying building parameters identified in the first part of this chapter) and calculating overheating indices (identified in the second part of this chapter) in each simulation. From parametric simulations, it creates a new database for each cluster of buildings and then trains a multi-output surrogate model capable of rapidly approximating indoor thermal conditions of other buildings within the same cluster.


Chapter 3
Climate change scenarios
Heat waves
Generating heatwave and future typical weather files
Projecting Urban Heat Island (UHI) effect on weather files
Chapter 2
Building typologies
Literature review on building typology methods
Application of clustering to split buildings into homogenous groups and selection of a representative building
Characterization of selected representative buildings
Chapter 4
Review of sensitivity analysis case studies to identify most influential building parameters in the literature
Summarizing sensitivity analysis case studies
Indoor overheating/comfort indices
Comparing various indices and making a suggestion
Chapter 5
Surrogate model development and indoor overheating exposure map
DOE and parametric thermal simulations of representative buildings
Meta-model/surrogate model selection
Re-training selected meta-model to approximate the performance of other buildings in the same cluster
Objective
Heat exposure map
Heat vulnerability map
Development of a rapid assessment API for each cluster of buildings
Included in this manuscript
Not included in this study
Sensitivity and adaptive capacity of people to overheating
Level of education
Access to medical support
Living status: (housed in a communal establishment, alone, with family)
Age
Health status
Behaviour
Financial status

Chapter 5 : Heat vulnerability maps

The principal objectives of this chapter is to describe the method to extend the application of dynamic thermal building simulations on typical/reference residential buildings to the rest of build stock for the case study city, Nantes, and prepare a citywide urban heat exposure/vulnerability map.

The problem from a first look seems straightforward that should be generalized well; however, it raises multiple practical difficulties due to missing building parameters belonging to the same cluster but significantly different from selected centroid, expected degree of accuracy, and applicability.

As discussed in chapter 2, the clusters were built on building and urban heat island effect parameters. Input parameters of clustering did not take into account thermo-physical characteristics of buildings, as were illustrated in chapter 4 – part-I that influence indoor overheating.

In the characterization stage of chapter 2, after identification of cluster centroids, a number of missing input parameters for building were attributed to cluster centroid as a function of year of construction, and assumptions were made on the behaviour of occupants. However, within each cluster, these inputs could vary considerably. Therefore, it is necessary to know how the centroid building is going to perform with varying range of input parameters that are used in characterization stage in order to extrapolate the results from cluster centroid to the rest of buildings within the same cluster. The term centroid of cluster is used interchangeably with representative or typical building referring to the closest real building to fictive centroid of each cluster, identified in chapter 2 of this manuscript.

In this study, as described in chapter 2, the individual performance of centroid buildings alone may not tell us much about the rest of the built stock, they become more useful if we build a number of assumptions into the surrogate, based on our expertise and experiences. In this problem, the primary task is to convey the behaviour of the modelled centroid building as accurately as possible to the rest of buildings within the same cluster, while remaining computationally effective.

Surrogate models, also referred to as “meta-models”, “emulators”, and/or “approximation models” in the literature provide an appealing data-driven approach that would allow solving this problem in an efficient manner. In the building sector, these meta-models are argued to be effective low-cost tools as a replacement of the computationally expensive models for building performance assessment.

Surrogate models are divided into three groups: data-fit models, reduced-order models, and hierarchical models (Allaire and Willcox 2010; Eldred, Giunta, and Collis 2004). Data-fit models are created from the relationships of input and output simulation data of high-fidelity models using regression, interpolation, or machine learning techniques. The key limitation of data-fit models is the “curse of dimensionality”, requiring practitioners to carefully implement a design of experiment to balance the cost of computation and model performance. Reduced-order models are commonly used for systems, which are based on partial differential equations

or large number of ordinary differential equations. They are not suited for systems where governing equations are unknown or are empirically based, as their derivation relies on the knowledge of governing equations (Allaire and Willcox 2010; Berger et al. 2017). Hierarchical surrogate models, also referred to as variable-fidelity models, use simplified mathematical models and models with simplified physics (Allaire and Willcox 2010). In this thesis work, the relationships between selected input parameters and simulations outputs are unknown; therefore, the surrogate modelling strategies that perform better in black-box situations, such as data-fit methods will be used. In the following, the terms surrogate modelling or meta-modelling are used interchangeably referring to the data-fit group of surrogate models.

A huge number of methods for constructing such models have been developed; a detailed review of the most significant of them is illustrated in the book of (Forrester, Sóbester, and Keane 2008).

Engineering problems of this nature that require the construction of a black-box-type meta-model to emulate the response of an expensive simulation tool in an efficient way come in various formats. According to (Forrester, Sóbester, and Keane 2008) they all can be distilled down to following structure:

Given an expensive simulation/experiment function y=f(x) and a set of data samples with relevant input and outputs within the design domain$(x \in D)$*, we seek to build an approximation function* $\hat{y}=f(\hat{x})$.

In this formulation, an important aspect of a surrogate/meta-model is to be accurate. The objective is to build the model process with the minimum number of expensive simulations. The degree of accuracy can be measured through absolute terms such as Relative Mean Squared Error (RMSE), Mean Squared Error (MSE) and/or in percentage terms such as R^2. In this study, in the following, both R^2 and MSE will be used. The specific threshold of acceptable degree of error is different from one problem to another. The process is also presented in the Figure 5-1.

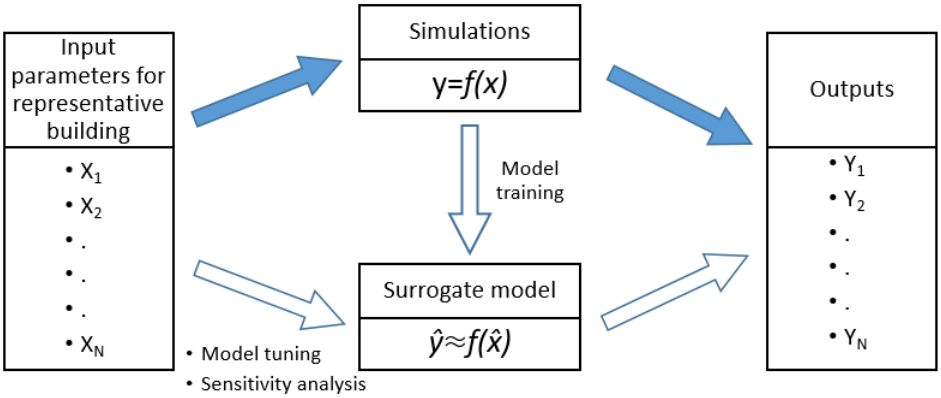


Figure 5-1 : Formulation of surrogate models

As can be seen in Figure 5-1, there are three key steps in the surrogate model formulation: (1) selection of input and outputs of surrogate model, identified in chapter 4 of this manuscript, (2) determining the minimum number of samples necessary to train the surrogate model, and (3) surrogate model selection.

Identifying the minimum number of samples requires the discovery of promising regions and exploitation of those regions to build the surrogate model. The choice of samples is referred to as the experimental design by (Herten et al. 2017). The choice of samples to be evaluated in construction of a surrogate model are affected by the goal of the process and model requirements.

In the following section of this chapter, first input and output data of surrogate model are illustrated then some of the common techniques for experiment design are described. Following that, surrogate model is selected through a comparative analysis, which is then used for approximation of other buildings within the same cluster.

5.1 Data and Methods

5.1.1 Input and output data of surrogate model

The goal of the process in this problem is to predict the performance of buildings that are in the same cluster using a surrogate model. The latter is built on the basis of simulations performed on the representative building by varying its input data to take into account the variance in the cluster. Input parameters of model were selected on the bases of study in the first-section of chapter 4 and characterization section of chapter 2. Table 5-1 illustrates the input parameters of cluster that could vary significantly for each building of the cluster.

Table 5-1: Varying parameters of cluster KM7_5 centroid

Building type: KM7_5	**Surface Area of building [m2]**	**U-value of exterior wall [W/m²K]**	**U-value of exterior roof [W/m²K]**	**U-value of intermediate wall [W/m²K]**	**Type of window**	**Adaptive capacity of user = External shading status & Window status (air inflow= infiltration + natural ventilation)**	**Window to wall ratio (window to area ratio) [%]**	**Principle orientation [degrees]**
Type	**Continuous**	**Continuous**	**Continuous**	**Continuous**	**Categorical**	**Categorical**	**Categorical**	**Categorical**
Level	**3**	**3**	**3**	**3**	**2**	**3**	**2**	**2**
Min [-1]	70	0.24	0.2	0.44	[1] Simple	[1] Not adaptive user	[W1] Small, R<20%	NS [O1]
mean [0]	95	1.25	0.792	1.158	[2] Double	[2] Intermediate user	[W2] Large, R>20%	EW [O2]
Max [1]	120	3	2.42	2.081		[3] Highly adaptive		

Output data of surrogate model were selected based on the study of overheating indices in part two of Chapter 4. The Table 5-2 presents the list of output from surrogate model.

Table 5-2 : Outputs of surrogate model

Indicator of performance for five	Percentage of hours above Cat	Percentage of hours in thermally	Peak indoor	Maximum number of consecutive	Degree-hours	RE2020 situation

summer months (May to September)	ii of EN 16798	stressful region of Givoni index	operative temperature	hours temperature is above 27 degrees	according to RE2020	
Unit of measurement	[%]	[%]	[°C]	[hours]	[degree-hour]	1,2,3
Type of data	Continuous	Continuous	Continuous	Continuous	Continuous	Categorical

5.1.2 Design of Experiment (DoE)

The DoE method makes it possible to develop prediction models for a given system by establishing a polynomial-type relationship between the input and output variables with the fewest possible combinations. Several types of experimental designs have been proposed in the literature. In the following, some of the key types of DoE are described. The choice of an experiment matrix is the fundamental problem of the design of experiments in order to obtain the best precision with a minimum number of combinations. The number of combinations is calculated based on the number of factors, levels, and type of data. In this case, in Table 5-1 of input parameters, number of factors equal to 8, 4 of which are continuous factors that have three levels, 1 is a categorical factor that has 3 levels, and 2 other categorical factors have 2 levels.

5.1.2.1 Types of DoE

a. Full factorial experiment design

This is probably the easiest type of DoE to build. In this case, the matrix of experiments contain all the possible combinations of the factors. If we consider m number of factors at n levels, the size of the matrix is therefore n^m. In the case of multiple factors at different levels, the combination number is $n_1^{m_1} * n_2^{m_2} * \dots * n_k^{m_k}$.

In this study the total number of runs for the parameters of each cluster in Table 5-1 equals to $3^5 * 2^3 = 1944$. In this combination, continuous variables have 3 levels of variations and categorical variables have 3 and 2 levels of variations.

b. Fractional factorial design

In this DoE approach, as its name suggests, only a fraction of full factorial design is used. In other words, it is a reduced version of full factorial design.

In this approach all the interactions of order three and higher are taken into consideration. (Montgomery 2017) also argues that a system often times is dominated by effect of parameters themselves and two-factor interactions. Knowing this factor we can choose to ignore the importance of three-factor or higher order interactions.

In the literature, several types of fractional factorial plans have been developed. For instance, Taguchi tables are fractional plans based on orthogonal Hadamard matrices (Dean et al. 2015). Fractional factorial designs allow practitioners to reduce the number of runs without sacrificing significant loss of information. They are, however, not recommended in places where input factors have more than two levels, or when test spaces are constrained with unusual run size or

restrictions. A computer-generated optimal design is considered more effective in handling such types of scenarios.

c. Response Surface method (RSM)

The previous two methods are more suited for first and second order models. For scenarios where a higher order of interaction is expected, RSM plans are used. Three most widely used techniques within RSM are: composite plans, Box-Behnken and Doehlert plans (Dean et al. 2015).

d. D-optimal DoE

D-optimal designs were developed to address constrained problems. They are also used to reduce the number of combinations as much as possible in the case of an unconstrained problem (Goupy 2013). D-optimal selects the best points to run the experiment. This is done by maximizing the determinant of the matrix $[X][X]$ or minimizing $[X][X]^{-1}$ and hence satisfying the D-optimality criterion. D-optimal is generated by iterative search algorithms, and most of state-of-the art generators are part of commercial software such as Minitab[8] or JMP (SAS)[9]. D-optimal is flexible and can be applied to cases where conventional DOE protocols do not apply.

The D-optimal algorithms generated in computers works in the following manner. First, the user determines the response (Y) and independent variables (input factors) of an approximate mathematical model (in this case building simulation in Trnsys). Then, computer generates a set of possible candidate points based on the level and number of factors. From these candidate set of points, a subset is selected that maximizes the determinant of $[X][X]$ matrix. In computer, the D-optimal experiment design starts with the selection of a random set of points. Points inside and out of the randomly selected design are exchanged iteratively until no exchange can be found that would increase the determinant of $[X][X]$ matrix.

5.1.2.2 Selected method for DoE

Taking into consideration the description presented in the previous section and the detailed comparative analysis presented by (Romani, Draoui, and Allard 2021), D-optimal was recommended to be more suited for the problem dealt in this chapter. The specific reasons substantiating this choice are listed as follows:

a. A significant number of input parameters are more than two levels, as presented in Table 5-1.
b. The design space for the input parameters are constrained. Meaning the values in the table show the maximum, minimum and median values of the building feature.
c. Some of the input parameters are categorical values.

Given the stated restrictions, a full factorial experiment design would naturally give the best results, but it requires too many runs. Therefore, the alternative is D-optimal method in DoE. In full factorial the number of simulations for each representative building was 1944 but using D-optimal it dropped to 72 runs.

[8] Minitab is a command- and menu-driven software package for statistical analysis.
[9] JMP is a software program used for statistical analysis.

5.1.3 Function approximation models

In this problem, where the goal is to train a surrogate model to approximate the indoor thermal performance of other buildings within the same cluster, the nature of function is not known a priori and the problem requires us to have multiple outputs. Therefore, there is no solid information to tell us which surrogate model would work better. Having said this, several possible surrogate model types can be considered: Decision Trees, Random Forests, polynomial regression, multi-variate adaptive regression splines (MARS), kriging, RBFs, support vector machines (SVMs), among others. Figure below illustrates how surrogates model for this problem are tested and compared to one another.

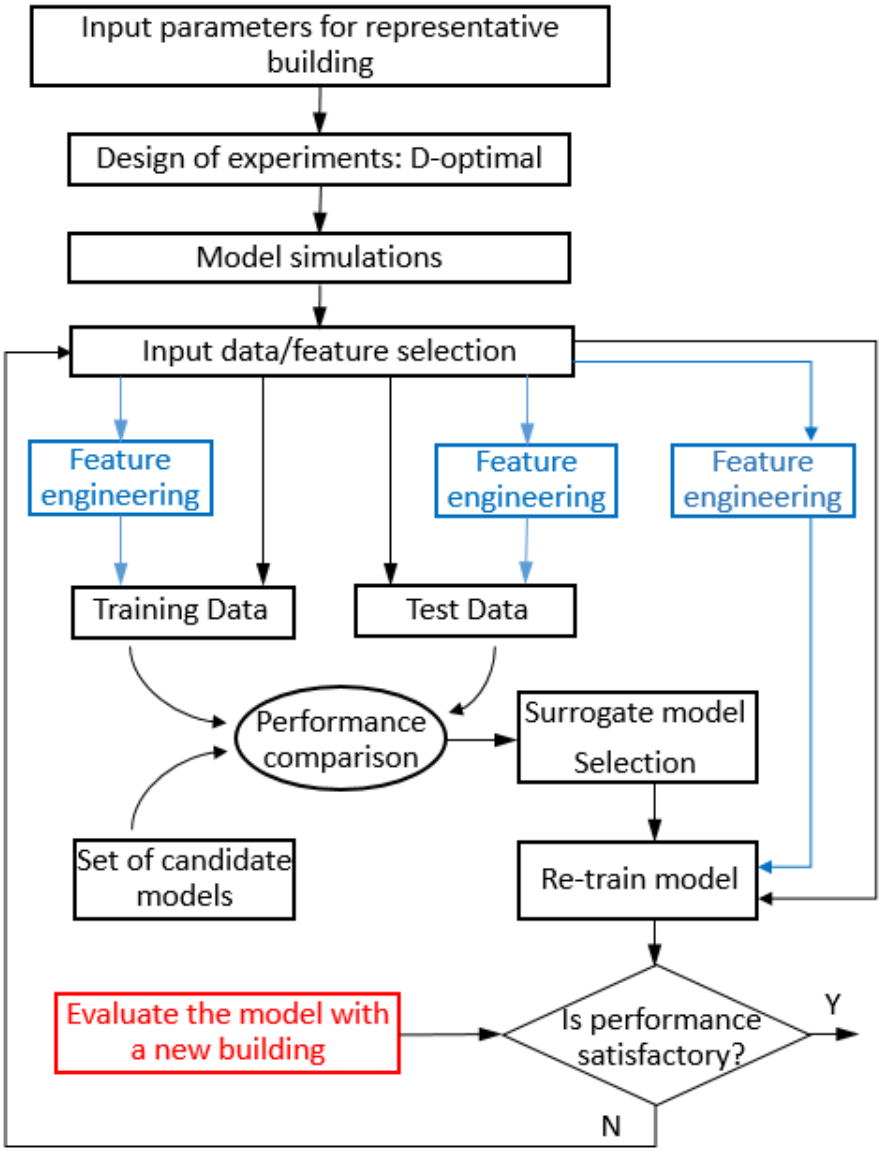


Figure 5-2 : *Schematic description of surrogate model construction and deployment for the studied problem*

In Figure 5-2, blue lines indicate non-linear transformation/manipulation of input features to include the combined influence of parameters. For instance, parameters X_1, and X_2, are transformed to X_1X_2 and are added to original features.

Choosing the right approximation function requires consideration for many factors such as level of practicality, interpretability, degree of accuracy, replicability and logic that inspired the development and deployment of the technique. In the following, practical and technical details of some of the approximation techniques, which are seen to be used in the literature, are presented. Fundamental equations and theories upon which these approximation techniques are built are not discussed in this manuscript, but references are provided for further research.

5.1.3.1 Polynomial and simple functions for approximation

(S. H. Kim and Fani 2019) called the polynomial and simple functions as non-interpolating models and according to the authors this category include functions such as linear models, quadratic, polynomial, and generalized regression. The models listed above minimize the sum of errors between sampled data points and predetermined functional form. The authors argue that the non-interpolating models lead to construction of simple and easily interpretable functions, but they may not be flexible to capture highly nonlinear correlations between input variables and target variable. The widely used technique in this category is probably the second-order polynomial regression. When least square method is used to measure the uncertainty, then the training sample size must be greater than the number of coefficients.

$$N>(n+1)\ (n+2)/2 \qquad \textit{Equation 5-1}$$

Where: N Is the number of runs (samples)

n Number of factors

Higher order polynomial regression are rarely used in surrogate model building for the following three reasons:

- The proper polynomial order is difficult to determine for problems where the nature of underlying function is not known a priori.
- Number of coefficients increases dramatically for high dimensional problems
- Higher order polynomials could also cause over-fitting.

(Cheng et al. 2020)

The mathematical forms of main polynomial equations to build surrogate models are as follows:

- **Linear model**

$$Y = a_0 + \sum_{i=1}^{n} a_i \cdot X_i \qquad \textit{Equation 5-2}$$

- **Quadratic**

$$Y = a_0 + \sum_{i=1}^{n} a_{ii} \cdot X_i^2 \qquad \textit{Equation 5-3}$$

- **Linear with interaction**

$$Y = a_0 + \sum_{i=1}^{n} a_i \cdot X_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} a_{ij} \cdot X_i \cdot X_j \qquad \textit{Equation 5-4}$$

- **Full quadratic**

$$Y = a_0 + \sum_{i=1}^{n} a_i \cdot X_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} a_{ij} \cdot X_i \cdot X_j + \sum_{i=1}^{n} a_{ii} \cdot X_i^2 \qquad \textit{Equation 5-5}$$

In these equations, Y is the response,

$X_i, and\ X_j$ are the input parameters

a_0, a_i, a_{ij} are coefficients of the model

System of equations for polynomial regression are also written in the form of the following matrix:

$$[Y] = [X]\,[A] \qquad \textit{Equation 5-6}$$

Where: A Vector of coefficients

X Vector of matrix calculation

Y Vector of response

5.1.3.2 SVM Machine learning technique for approximation/prediction

SVM stands for Support Vector Machine and it is a predictive analysis data-classification algorithm. Means its primary use is for classification but it has been used in regression tasks as well. SVM works by first mapping the data into a high dimensional feature space so that data points could be categorized. The mapping is called kernelling and the mathematical function used to transform is known as kernel function. Following is four types of kernel functions:

- Linear
- Polynomial
- Radial basis function (RBF)
- Sigmoid

The SVM algorithm is designed in such a way that it looks for points on the graph that are closest to the split line. These points are called support vectors. Then, the algorithm calculates the distance between the support vectors and the separating plane. This distance is called the gap. The main goal of the algorithm is to maximize the gap distance. The best hyperplane is the hyperplane for which this gap is as large as possible.

5.1.3.3 Decision Tree

A decision tree (also called a classification tree or regression tree) is a decision support tool used in machine learning, data analysis, and statistics. The structure of a tree is "leaves" and "branches". On the edges ("branches") of the decision tree, the features on which the objective function depends are written. The values of the objective function are written in the "leaves", and the other nodes are the features by which the cases differ. To classify a new case, one must go down the tree to a leaf and return the corresponding value.

5.1.3.4 Random Forest

Random forest (RF) is a machine learning algorithm which consists of (ensemble) of trees. It is applied for classification, regression and clustering. The main idea of RF is to use a large ensemble of decision trees, each of which by itself generates a very low quality classification, but due to their large number, the quality of overall classification improves significantly.

In a RF algorithm, the more trees, the better the quality of the regression/classification, but RF setup and operation times also increase proportionately with it. It is also important to note that often with an increase in the *n_estimators* (number of trees for RF), the quality on the training set increases (it can even reach 100%), and the quality on the test does not change anymore. That is the point where a practitioner decides the number of trees (Pedregosa et al. 2011).

5.1.3.5 Gradient Boosting Regression

Boosting is an ensemble technique, similar to Random Forest, in which the predictors are built sequentially rather than independently. This technique uses the idea that the next model will learn from the mistakes of the previous one.

Gradient boosting is used for **classification** and **regression** problems it builds a prediction model in the form of an ensemble of weak predictive models, usually decision trees.

The goal of this supervised learning algorithm is to determine the loss function and minimize it. Let us look into the math behind gradient boosting. Let, for example, take the mean square error (MSE) as the loss function:

$$L\left(Y_i, Y_i^p\right) = MSE = \sum (Y_i - Y_i^p)^2 \qquad \textit{Equation 5-7}$$

Where Y_i is the *i-th* target value, Y_i^p is the *i-th* predicted value and $L\left(Y_i, Y_i^p\right)$ is the loss function. The goal is to plot the predictions in such a way that the MSE is minimal. Using gradient descent and updating predictions based on (learning rate,α), we look for values where the MSE is minimum.

$$Y_i^p = Y_i^p + \alpha * \delta \sum \left(Y_i - Y_i^p\right)^2 / \delta Y_i^p \qquad \textit{Equation 5-8}$$

This equation becomes (derivative of loss with respect to predicted),

$$Y_i^p = Y_i^p - \alpha * 2 * \sum (Y_i - Y_i^p) \qquad \textit{Equation 5-9}$$

Where α is learning rate and $\sum(Y_i - Y_i^p)$ is the sum of residuals.

Therefore, the predictions are updated so that the sum of the residuals tends to zero and the predicted values are close to the real ones.

5.1.3.6 Multinomial Logistic regression

Logistic regression is one of the statistical classification methods using Fisher's linear discriminant. Unlike conventional regression, the logistic regression method does not predict the value of a numeric variable based on a sample of initial values. Instead, the value of the function is the probability that the given original value belongs to a particular class. For example, in this study, instead of predicting the numeric value of RE2020 for each zone or cluster, the categories of RE2020, as described in section 2 of chapter 4, is predicted. The logistic regression predicts the probability of a certain floor in the building belonging to a class. The response of logistic regression is always between the interval [0, 1].

In RE2020 situations, there are three possible situation types: comfortable, at risk, and non-regulatory. Then the result of regression is the probabilities of each building floor/thermal zone for the *i* instance illustrated as, $P(Y_i$=comfortable), $P(Y_i$=at risk), and $P(Y_i$=non-regulatory).

So, as mentioned, in a logit regression model, the predicted values of the dependent variable or response variable cannot be less than (or equal to) 0, or greater than (or equal to) 1, regardless of the values of the independent variables; therefore, this model is often used to analyse binary dependent or response variables.

Berkson, 1944, first used the term logit and the logistic function showing the probability of outcome is defined as follows:

$$\mathrm{y} = \frac{\exp(\beta_{0k} + \beta_{1k}\, x_1 + \cdots + \beta_{pk}\, x_p)}{1 + \exp(\beta_{0k} + \beta_{1k}\, x_1 + \cdots + \beta_{pk}\, x_p)} \qquad \textit{Equation 5-10}$$

The term logit comes from the fact that this model can be easily linearized using a logit transformation. Assuming that the binary dependent variable y is a continuous probability p ranging from 0 to 1, then we can transform this probability p as follows:

$$p' = \log \{p/(1-p)\}$$

This transformation is called the logit or logistic transformation.

Multinomial logistic regression is an extension of binary logistic regression and both operate on the assumption that there is no correlation between independent variables.

5.1.4 Framework in the construction of surrogate model

The goal of this chapter is to illustrate the method to predict the performance of buildings that are in the same cluster using surrogate modelling. In this section, input and output data, DOE construction techniques, and various surrogate models were described. In the following paragraphs, the process of surrogate model construction is demonstrated.

After selection of input and output data of surrogate model, a set of samples using D-optimal was selected and parametric simulations on the representative buildings were carried out. Results of parametric simulations for each cluster were collected to create a new database, which was then used as training and testing data to select appropriate surrogate modelling technique. For that, data for KM7_5, as also shown in Figure 5-2, was split into 70% training and 30% test data and were used to train and test all of the candidate surrogate models. In the following, the results and procedure of how they were obtained are illustrated in more details.

5.2 Results and discussions

To simplify the description of methodology and in particular sensitivity analysis results, in the following only the data and procedure implanted the centroids of Cluster KM7_5 (city center) and KM7_6 (peripheries) will be described in more details.

5.2.1 Feature importance analysis

The centroid building of KM7_5 is a multi-family 4-storey building consisting of a ground floor, two middle floors and attic. Building belonging to cluster KM7_5 are concentrated in city center and are located in densely built areas. The centroid of building KM7_6 is a 3-storey single family house consisting of a ground floor, a middle floor and attic. Buildings of cluster km7_6 are located in peripheries of the case study city, as shown in Figure 5-3.

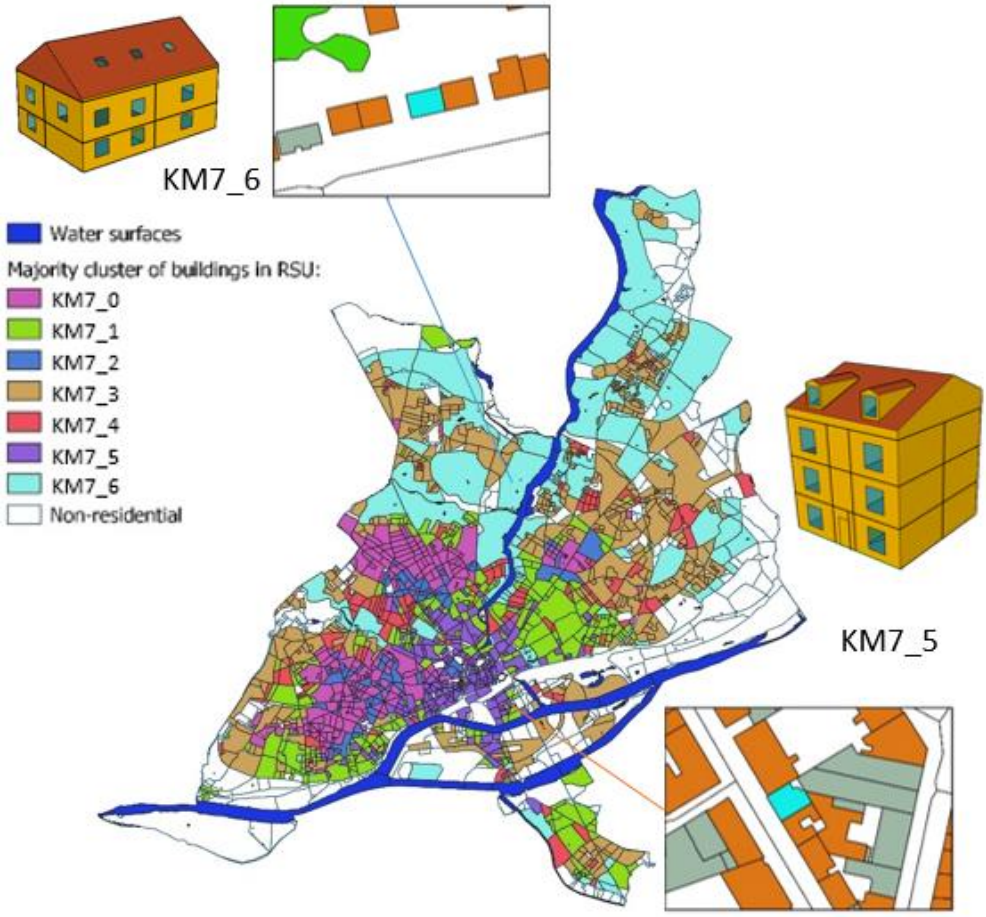


Figure 5-3 : KM7_5 and KM7_6 buildings in the case study city

Division of buildings into zones, location, other parameters of base case scenario are presented in chapter 2 of this manuscript. After creation of D-optimal DOE, building parameters demonstrated in Table 5-1 were varied and simulated in TRNSYS V.17.

In each simulation, output parameters illustrated in Table 5-2 and energy consumption information were calculated for each zone of the building. In the following, in line with overall objective of the research, only parameters related to summer overheating were processed. Appendix 5-1 shows the variations of the KM7_5 cluster centroid resulting from the D-optimal experiment design.

After completion of simulations, to measure if input parameters are inter-correlated, a Pearson correlation analysis was performed on the input parameters of each cluster and results showed relatively weak inter-correlation between a small number of input parameters, meaning that they can be used for regression analysis and construction of surrogate model.

After that, a feature importance analysis was carried on the results of parametric simulations for each cluster centroid, where the vertical position of zone was also included as a variable. Results of feature importance analysis for KM7_5 is presented in Figure 5-4.

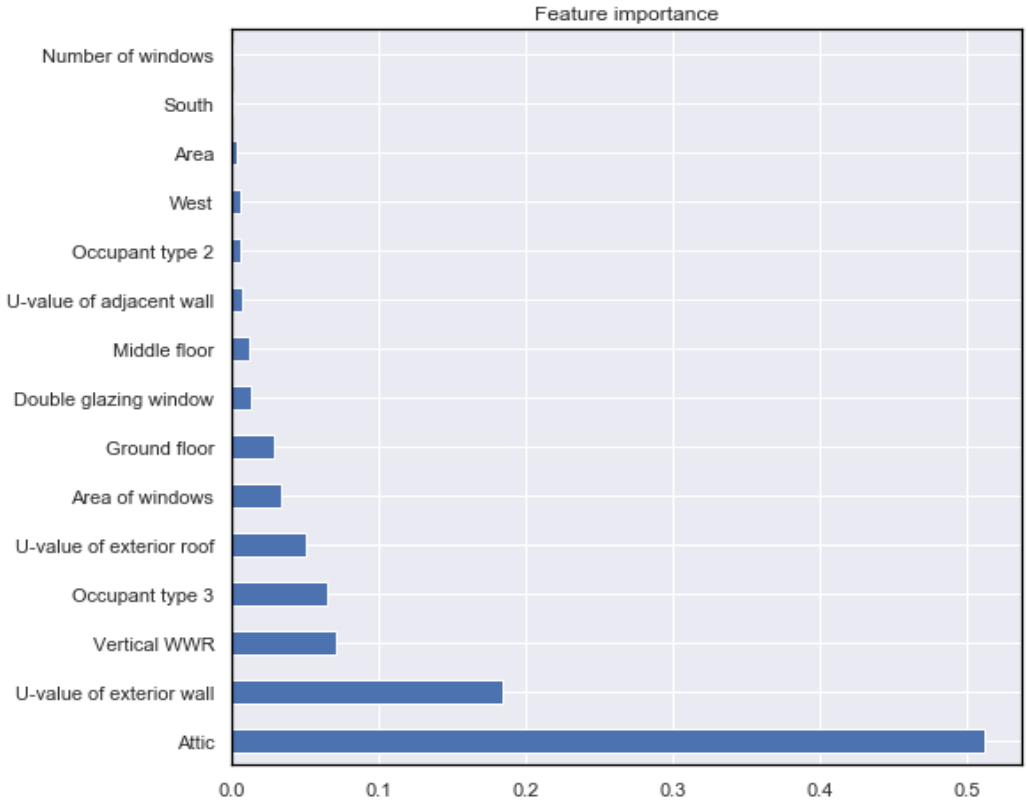


Figure 5-4 : Feature importance analysis on the centroid building of KM7_5

Results in Figure 5-4 show that regardless of how different other features are in a building, indoor thermal conditions during summer inside a building is highly influenced by the location of thermal zone within a building. In the overall variations of output parameters for the whole building, attic accounts for more than 50% of those variations.

This simply means that for indoor overheating analysis and construction of a surrogate model, it is necessary to study attic separately from other floors. In other words, even if other parameters remain the same, attic would on average be warmer than other floors. Having this in mind, in the following influence of parameters, tuning, training and validation of surrogate models are carried out separately for attic and other floors of the building (ground floor and middle floor).

5.2.1.1 Feature importance analysis for ground and middle floors

Input data from attic was filtered out from data and simulation results. Then a new feature importance analysis was carried out on the input parameters assuming their influence does not change when they interact with each other. Results are presented in Figure 5-5.

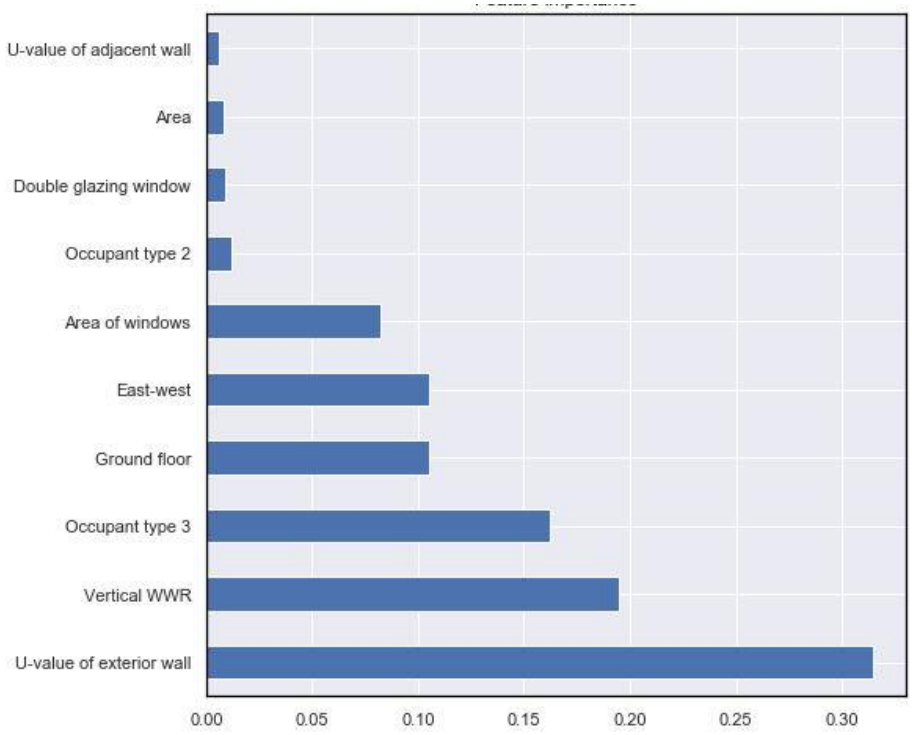


Figure 5-5 : Feature importance analysis results on ground and middle floor of KM7_5 assuming input parameters do not interact each other

For KM7_5 in the ground and middle floors, U-value of exterior wall, vertical window to wall ratio, occupant type (occupant behavior), net area of windows, and type of window glazing play the most significant role. For KM7_6, size of window, type of occupant, u-value of exterior wall, principal orientation, and area are among the most influential parameters, as presented in Figure 5-6.

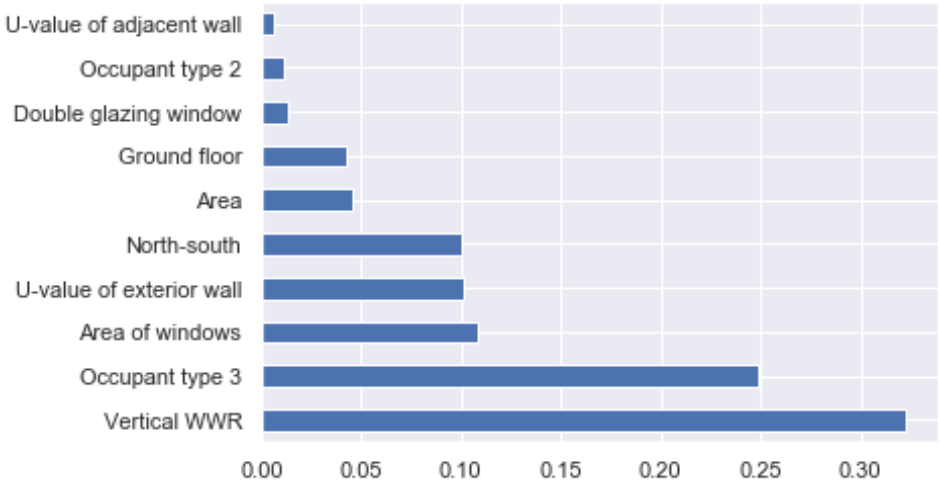


Figure 5-6 : Feature importance analysis results on ground and middle floor of KM7_6 assuming input parameters do not interact each other

In both buildings the first 5 parameters are almost the same but their order is different. This means that type of building can change the importance of one building parameter over another for its summer performance. For instance in this case, indoor summer performance of building located in densely built areas due to their limited exposure to solar irradiance, are more influenced by the thermo-physical properties of envelope elements, while building that are more exposed to external environment (sparsely built areas) are more influenced by the size of window and orientation. Occupant behavior, on the other hand, has a strong influence in both of them.

Results of Figure 5-5, and Figure 5-6 showed the results of feature importance analysis of individual parameters; however, there is no indication of how their importance would change on the output parameters if they combine and act together. To measure that, a new feature matrix consisting of all polynomial combinations of input features with degree one was created. Number of input parameters increased to 55 parameters and a feature importance analysis was carried out on them.

Bar charts in Figure 5-7 show the top 10 important parameters after combination of input parameters for KM7_5, and KM7_6. In this graph, the top 8 parameters of Figure 5-5 and their combinations make up the top 10 parameters in Figure 5-7. The result shows that combination of input parameters did not dramatically change the order of influence in parameters for summer discomfort.

The combination of parameters as mentioned earlier, are only for degree one non-linear transformation. For instance, parameters X_1, and X_2, were used to generate parameter X_1X_2 and were added to original features. Higher degree such as X_1^2 and $X_2 \cdot X_1^2$ were not generated because they dramatically increase the number of attributes and coefficients. It has been also argued that higher degrees of feature transformation lead to overfitting of the model.

Additionally, feature transformation by combination is often used only for multinomial regression analysis or in creation of surrogate models that are based on linear with interaction or/and full quadratic polynomial equations, described in section 5.1.3.1. Many other machine-learning techniques do not require the features to be combined in order to capture non-linear correlations between dependent and independent variables.

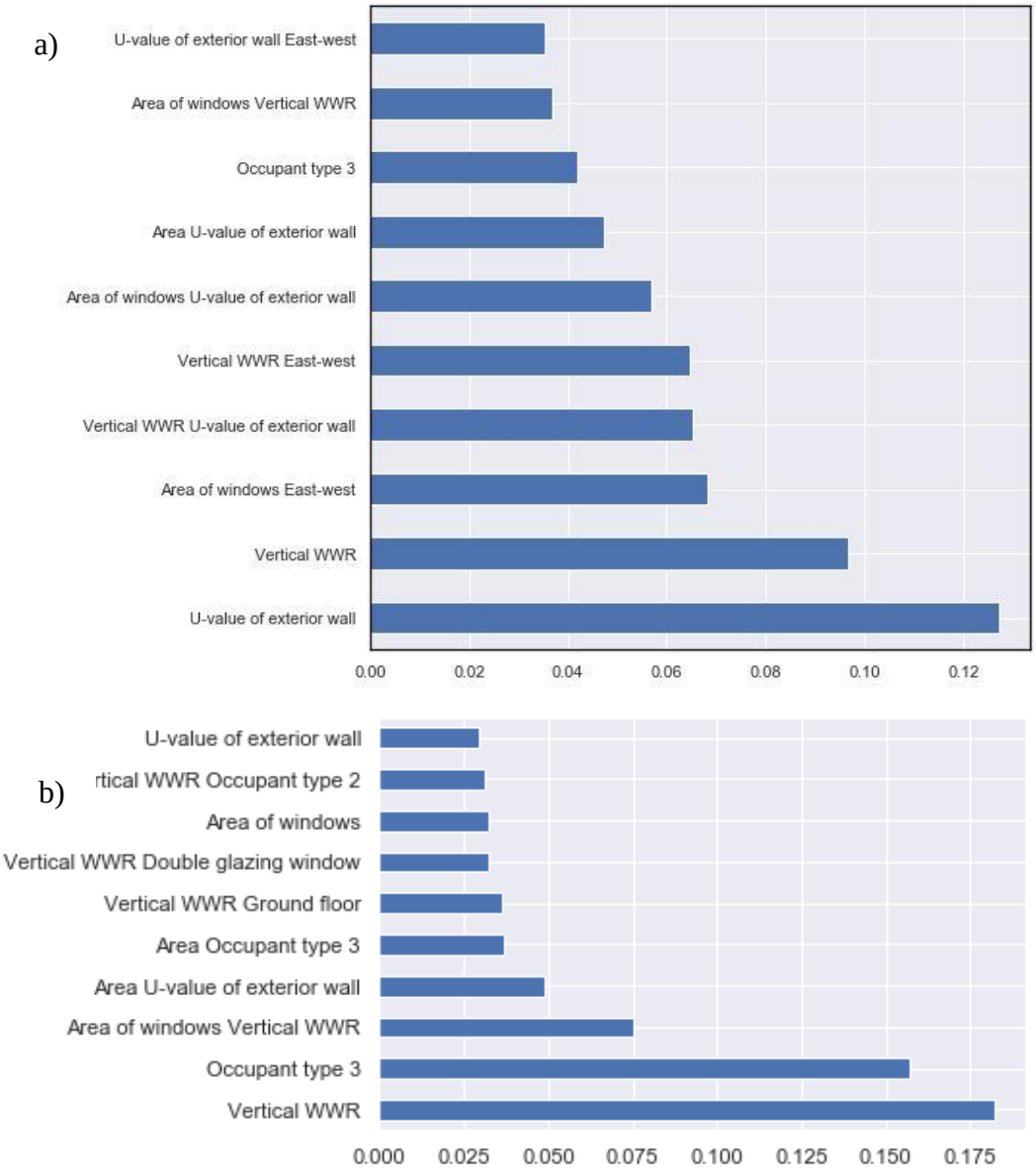


Figure 5-7 : Feature importance analysis results on combined input parameters for a): KM7_5 and b): KM7_6

5.2.2 Surrogate Model Selection for ground and middle floors

In the problem of surrogate model development, the key requirements are to maximize model accuracy and minimize its complexity. For multinomial regression, a superset of features was created by combining the input parameters of Table 5-1. For the rest of machine-learning techniques, only the input parameters were used to train and test their performance. As shown in Figure 5-2 the process of surrogate model has an important stage that deals with model selection. The selection of surrogate model requires consideration for physics-based relationships between dependent and independent parameters, and the expectation of practitioners from the underlying response. In this case, there are two underlying expected response types: (1) a multi-output regression-type surrogate model that generates five continuous dependent variables described in Table 5-2, and (2) a classification-type surrogate model that predicts the RE2020 situation type categorical dependent variable.

Since insights into the physics that describe the correlation between dependent and independent variables are not obvious due to complex dynamic simulations and post processing of data. It is necessary to find a balance between model complexity and performance accuracy.

For performance accuracy measurement, input data was split into 0.7 and 0.3 ratios, where 0.7 was used for training and 0.3 to test the performance of models. In the measurement of accuracy, for multi-output regression-type surrogate model mean squared error (MSE) was calculated to measure the difference between simulated and predicted values as demonstrated in Table 5-3. Additionally, average MSE and average RMSE for all five indices were also calculated.

Table 5-3 :Mean squared error on test data for middle and ground floor of KM7_5

Means Squared Error compared to test data (MSE) Index	Simple Multiple Linear Regression	Multiple Linear regression with interaction of input parameters	Decision Tree	Support Vector Machine (SVM)	Gradient Boosting Regression	Random Forest Regression
Percentage of hours above Cat-II EN 16798	15.71	15.85	25.46	17.78	7.63	9.366
Givoni thermally stressful	0.043	0.037	0.033	0.046	0.01	0.011
Maximum Consecutive hours above 27	4475.35	5024.17	7908.8	6472.33	1062.177	1147.20
Peak operative temperature	0.34	0.38	0.549	0.36	0.22	0.261
RE2020 degree hour	126497.97	123612.7	184443.8	223180.06	62006.5	75723.3
Average MSE	26197.88	25730.6	38475.74	45934.11	12615.31	15376.03
RMSE	161.8	160.4	196.152	214	112.3	124

As can be seen in Table 5-3, Surrogate model based on Ensemble Gradient Boosting Regressor performs better than other techniques at predicting indoor characteristics of middle and ground floor. MSE and RMSE are both relative terms and do not always provide all the information need. That is why the strength of relationship between simulated and predicted values of test data for each dependent variable was also calculated with R^2 score for the best performing regressor.

Table 5-4 : Coefficients of determination (R^2 score) of multi-output regressor surrogate model.

Index	**R2_score of Gradient Boosting Regressor**
Percentage of hours above Cat-II EN 16798	0.804
Givoni thermally stressful	0.866
Maximum Consecutive hours above 27	0.868
Peak operative temperature	0.764
RE2020 degree hour	0.790

Figure 5-8 plots the predicted values of surrogate models and test data for each dependent variable of a distribution plot.

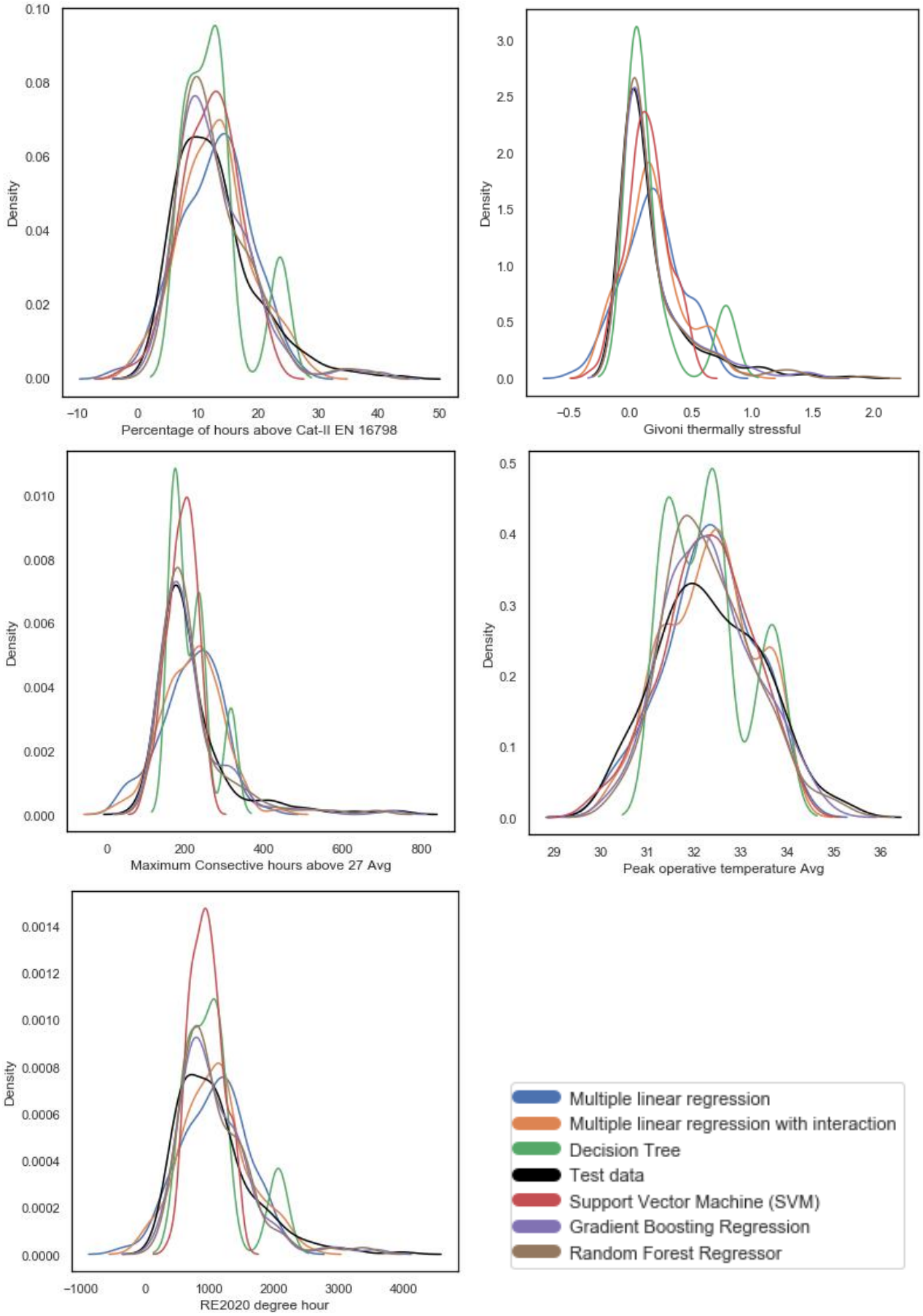


Figure 5-8 : Distribution plots showing performance of each regression-type surrogate model for each dependent variable of ground and middle floors for KM7_5 cluster

From figures that show distribution of predicted values compared to test data and the degree of error measured and presented Table 5-3, Gradient Boosting Regression performs better at predicting indoor overheating indices for all selected output variables that are continuous.

As for classification-type surrogate model, where the objective is generate a categorical type output, accuracy is measured with different indices. In this study, a confusion matrix was employed that summarized the number of correct and incorrect predictions broken down by each class. It provides insight into the number and type of errors. For prediction of **classification-type surrogate model**, in this study two techniques were used: (1) multinomial logistic regression and (2) Gradient Boosting Classifier. In both of these techniques, the output is RE2020 situation type. Results are presented as follows:
Multinomial Logistic regression. Overall accuracy = 89%

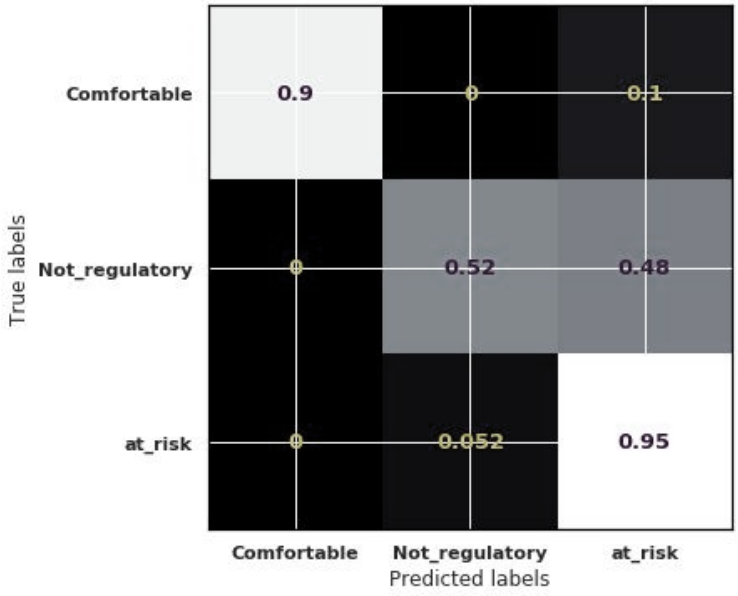


Figure 5-9 : Confusion matrix for classification with Multinomial logistic regression [%]

Gradient Boosting Classifier. Overall accuracy = 84.7%

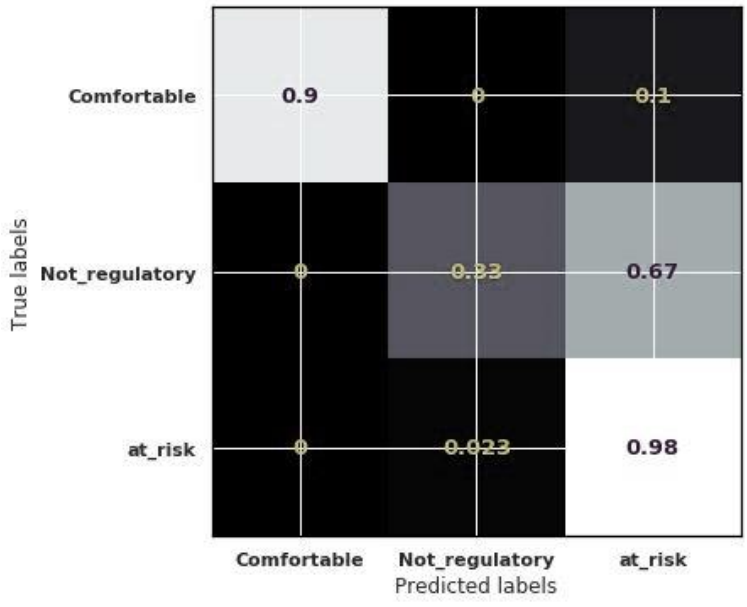


Figure 5-10 : Confusion matrix for classification with Gradient Boosting Classifier [%]

In classification of categorical variables, multinomial logistic regression performed better for this problem. It is however, important to mention that Gradient Boosting Classifier due to its ability to be tuned more and more, has the potential to outperform Multinomial Logistic Regression if the practitioners gives enough time to tune all hyper-parameters. Finally, Multinomial Logistic Regression was selected for surrogate model construction when the expected output is a categorical variable.

5.2.3 Surrogate model's performance with a new building from the same cluster

The next step after selection and training of the surrogate model is validation. For that, a random building from the same cluster was selected and simulated using TRNSYS v17. Trained surrogate model was then used to approximate the performance of this new building.
Results of simulations, predictions and ranges of answers for ground floor, and middle floors of KM7_5 cluster are presented in Figure 5-11, Figure 5-12 and Table 5-5 below.

In Figure 5-11, 5-12 the blue lines indicate simulated values and red line show surrogate model approximation for the same floor of building. The boxplots illustrate the range of variations in the performance of building centroid as input parameters were varied in parametric simulations.

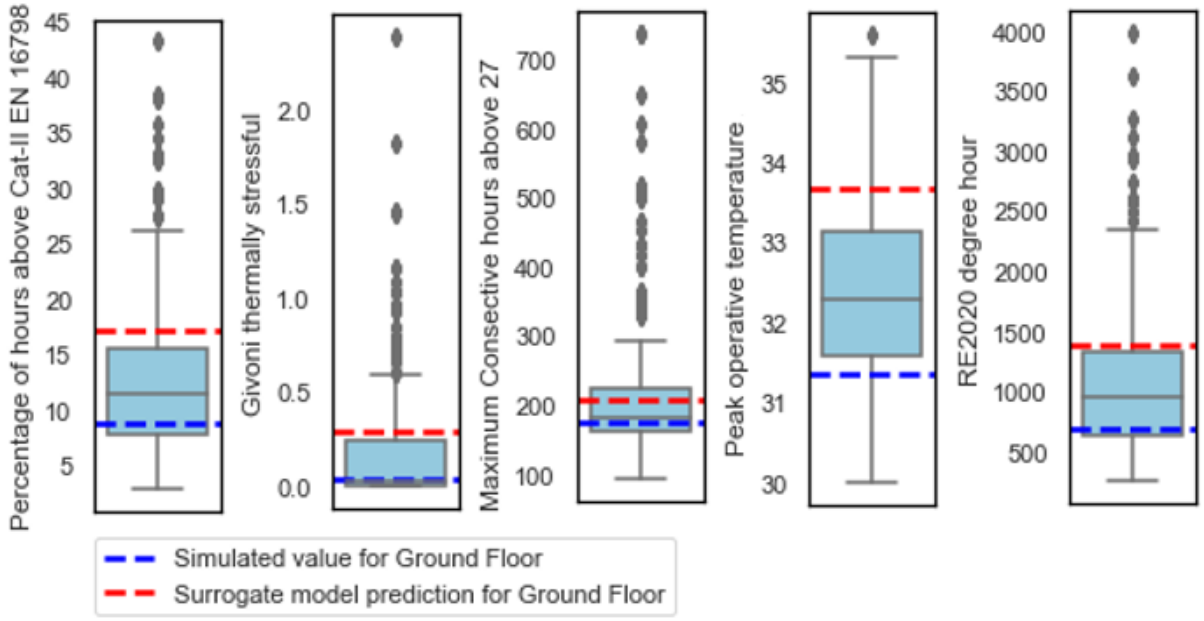


Figure 5-11: Simulated and predicted values for ground floor of the test randomly selected building in cluster KM7_5

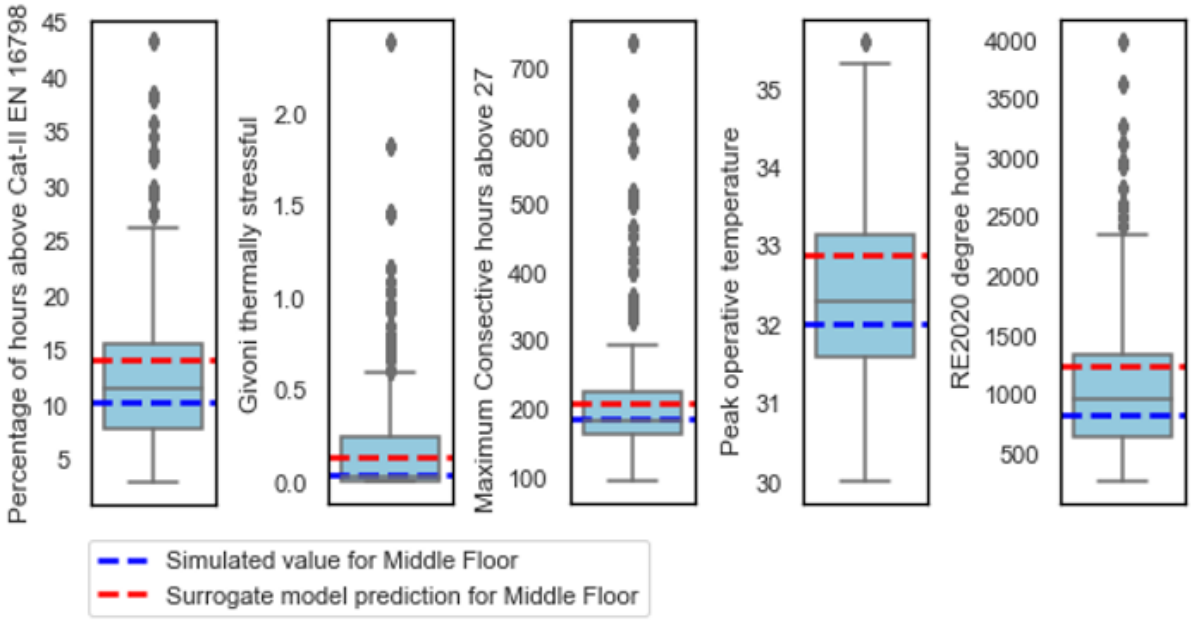


Figure 5-12 : Simulated and predicted values for middle floor of the test randomly selected building in cluster KM7_5

In Table 5-5 the term predicted is referring to value that surrogate model predicted for the floor.

Table 5-5 : Simulated and predicted values for the validation building in cluster KM7_5

Floor	Measurement index	Simulated	Predicted	difference
Ground floor	Percentage of hours above Cat-II EN 16798	8.9	17.24	8.3
	Givoni thermally stressful	0.036	0.29	0.25
	Maximum Consecutive hours above 27	177.37	208.8	31.5
	Peak operative temperature	31.36	33.6	2.2
	RE2020 degree hour	691.33	1376.7	685
Middle floor	Percentage of hours above Cat-II EN 16798	10.24	14.2	3.96
	Givoni thermally stressful	0.036	0.129	0.09
	Maximum Consecutive hours above 27	185.15	209.96	24.7
	Peak operative temperature	32.0	32.89	0.89
	RE2020 degree hour	824.43	1232.07	408

Comparison in the simulated and approximated values shown in Table 5-5 and Figure 5-11, 5-12 indicate a small difference between predicted and simulated values of the new building for ground floor and middle floor of the building.

Implementation of Multinomial logistic regression for ground and middle floors predicted the RE2020 situation of zones in each floor with 100% accuracy, as presented in Figure 5-13. Result of surrogate model on validation building indicate that surrogate model of KM7_5 performs well for ground and middle floors.

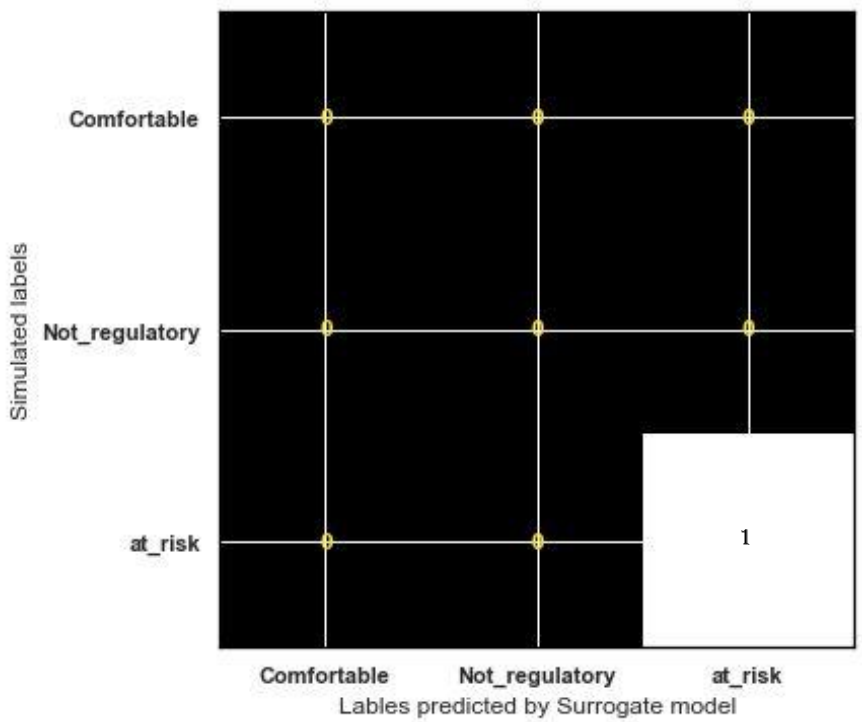


Figure 5-13 : Confusion matrix of predicted and simulated RE2020 situation of the validation model

The results of surrogate model validation for KM7_6 is presented in appendix 5-2 of this manuscript. Outputs of surrogate model for ground floor and middle floor were similar to what was observed for ground and middle floor of KM7_5.

5.2.4 Surrogate model for attic

As illustrated in data selection of this chapter, attic of centroid for cluster KM7-5 showed significantly different behaviour, and influence of parameters varied considerably compared to ground and middle floors. Therefore, a separate analysis was carried out for it.

In the meta-model parameters related to orientation (north, south, east, and west) are absent from attic. This is because in parametric simulations attic was considered as a single zone and simulations were performed for two major orientations: North-South and East-West. U-value of interior is also absent because attic was considered as one thermal zone.

5.2.5 Surrogate model selection for Attics

Results of parametric simulations for attic, similar to ground and middle floor was split into training and testing set. Various surrogate model construction functions were trained and tested on the selected data. Comparative results of their performance are presented in Figure 5-14 and Table 5-6.

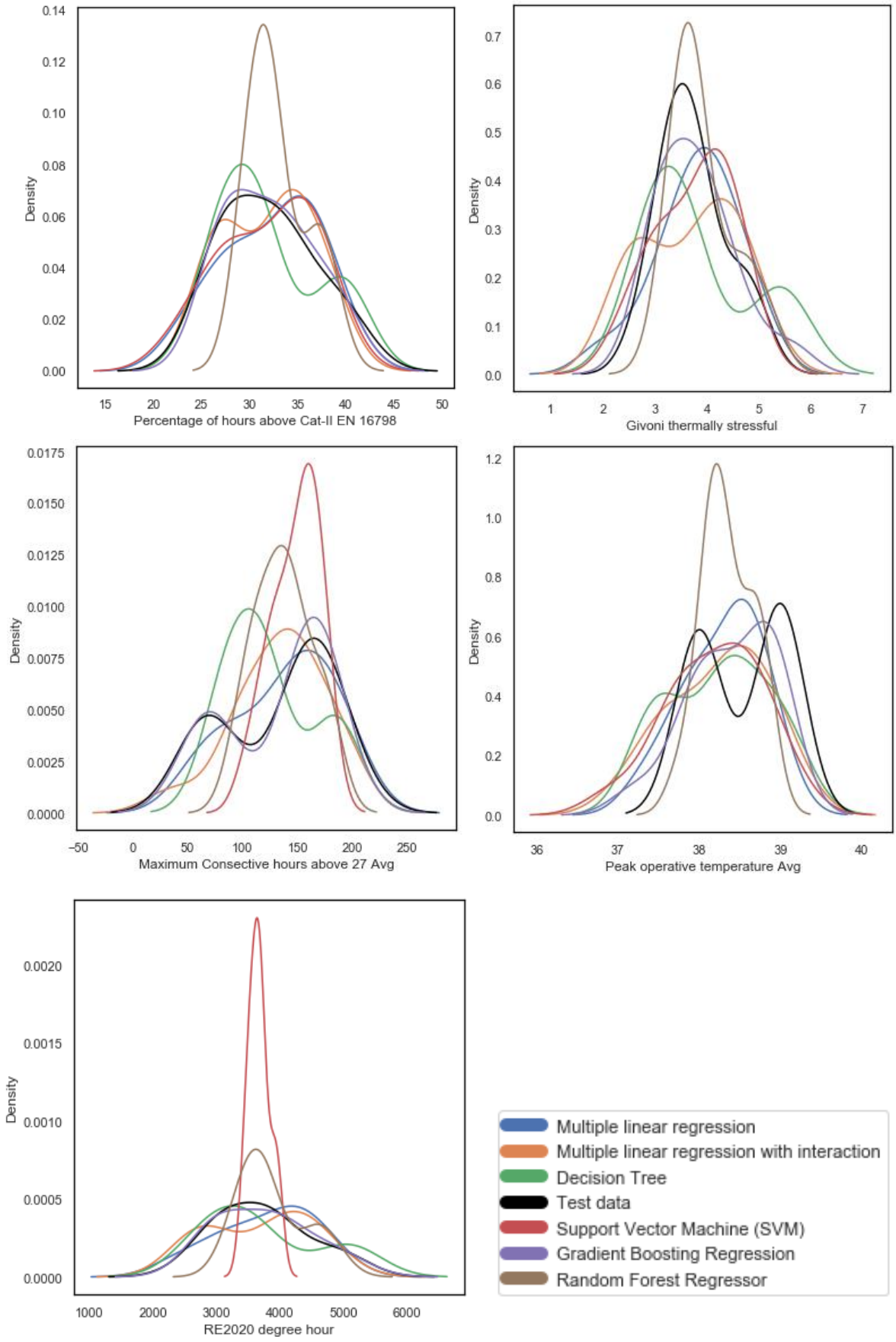


Figure 5-14 : Distribution plots showing performance of each regression-type surrogate model for each dependent variable of KM7-5 attic

Figure 5-14 shows that SVM and Random Forest supervised learning methods have over fitted training data and do not perform well on test data. Linear regression methods under fit the model and therefore do not perform well on test data.

Table 5-6 : Mean squared error on test data for attic

Means Squared Error compared to test data (MSE) Index	Simple Multiple Linear Regression	Multiple Linear regression with interaction of input parameters	Decision Tree	Support Vector Machine (SVM)	Gradient Boosting Regression	Random Forest Regression
Percentage of hours above Cat-II EN 16798	2.68	1.3338	4.727	2.666	0.4568	7.47
Givoni thermally stressful	0.19	0.22	0.29	0.1445	0.061	0.158
Maximum Consecutive hours above 27	298.7	437.23	807.8	1068	61.08	825.97
Peak operative temperature	0.268	0.383	0.445	0.408	0.18	0.23
RE2020 degree hour	97482	73598	128909	526114.4	8504	165847
Average MSE	19556	14807.46	25944	105437	1713	33336
RMSE	139	121.7	161.07	324.7	41.39	182.58

Figure 5-14 and results presented in Table 5-6 demonstrate that gradient boosting performs considerably better than other surrogate model functions in predicting the performance of test data. To measure how close predicted and simulated values are on test data R^2 score for Gradient Boosting Regressor was calculated.

Table 5-7 : Coefficients of determination (R^2 score) of multi-output regressor surrogate model

Index	R2_score for Gradient Boosting Regressor
Percentage of hours above Cat-II EN 16798	0.976
Givoni thermally stressful	0.92
Maximum Consecutive hours above 27	0.97
Peak operative temperature	0.30
RE2020 degree hour	0.983

R^2, and RMSE show a good performance in prediction of indices, using Gradient Boosting regression, on test data except for Peak Indoor Operative Temperature. Inability of surrogate functions to capture variations of peak indoor operative temperature for attic could be because the range of variations in peak indoor temperature of attic in the training data is small compared to other indices (between 37 and 40 °C). Small range in the training data, in turn, means that input parameters that were chosen in parametric simulations do not influence peak indoor operative temperature of the attic significantly and other factors may be influencing it.

Simulation and prediction results of meta-model on validation building is presented in Table 5-8 and Figure 5-16.

Table 5-8 : Simulated and predicted values for the attic of validation building in cluster KM7_5

Floor	Measurement index	Simulated	Predicted	difference
Attic	Percentage of hours above Cat-II EN 16798	13.08	34.496	21.4
	Givoni thermally stressful	0.402	4.34	3.94
	Maximum Consecutive hours above 27	159.98	165.035	5
	Peak operative temperature	34.21	39.33	5.12
	RE2020 degree hour	1055.026	4179	3124

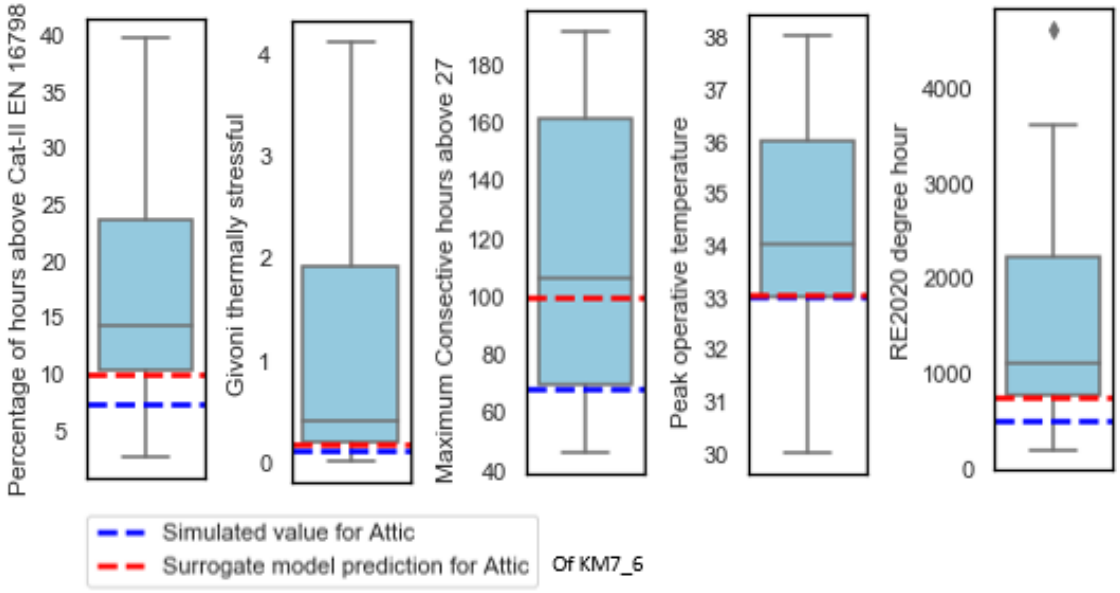


Figure 5-15 : Simulated and predicted values for Attic of the validation building in cluster KM7_6

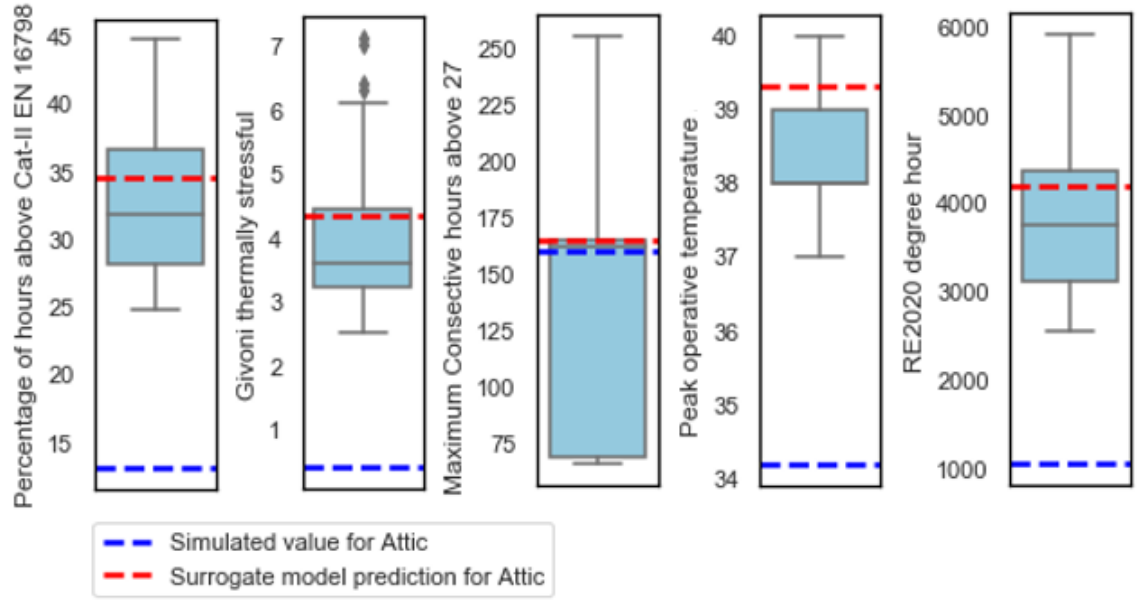


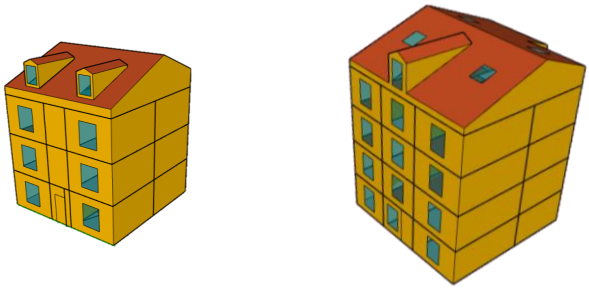
Figure 5-16 : Simulated and predicted values for Attic of the validation building in cluster KM7_5

Application of trained surrogate model to predict indoor thermal indices in the attic of the validation building that was selected from the same cluster further demonstrates that input parameters that allowed surrogate model to predict indoor thermal indices of ground floor and middle floors do not produce satisfactory results for attic. This could be because the indoor thermal conditions of attic vary extensively with outdoor environmental conditions. Meaning, an increase in outdoor temperature can rapidly change indoor temperature of attics. Insulation, thermal inertia, and type of ventilation are also factors that could be affecting this rapid change.

In addition, unlike the KM7_6 cluster, there is a significant difference in the shape of attic as well in the number and types of openings between the cluster centroid KM7_5 and the validation building in the same cluster, as presented in Figure 5-17, below.

Figure 5-15 shows that surrogate model for attic of KM7_6, which is able to approximate the indoor performance of attic for that specific cluster because the shape and type of windows for centroid building and validation building are similar.

a)



b)

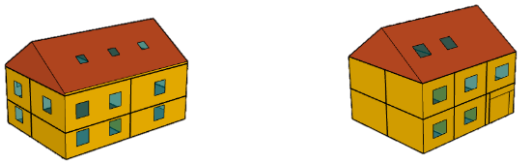


Figure 5-17 :
*a) **Left**; cluster centroid building used for parametric simulations, **right**; validation building selected from cluster KM7_5*
*b) **Left**; cluster centroid building used for parametric simulations, **right**; validation building selected from cluster KM7_6*

The difference in shape and properties of attic is pervasive within the buildings of the same cluster, making it complicated for the surrogate model, described earlier, to correctly predict indoor thermal conditions of attics of the rest of buildings based on those parameters.

5.2.6 Implementation of surrogate model on the rest of buildings in cluster KM7-5

Satisfactory performance of surrogate model to predict indoor overheating indices of validation building using the Gradient Boosting Regression for the ground and middle floors of validation buildings indicates that surrogate model could be deployed to estimate indoor thermal condition of the rest of buildings in the clusters to approximate the performances' of ground and middle floors of buildings.

5.2.6.1 Technical implementation process

Implementation of trained surrogate model is closely linked to the database used for cluster analysis, presented in chapter 2, because the database contains "building year of construction", and a cluster number that had been assigned to each building during cluster analysis.

Building year of construction was used as a primary feature linking most of the assumptions and buildings in the database. A python script was developed to implement the trained surrogate model for each building using the assumptions, and that way approximate 6 outputs indicators for each building. The process is also shown in the Figure 5-18, below.

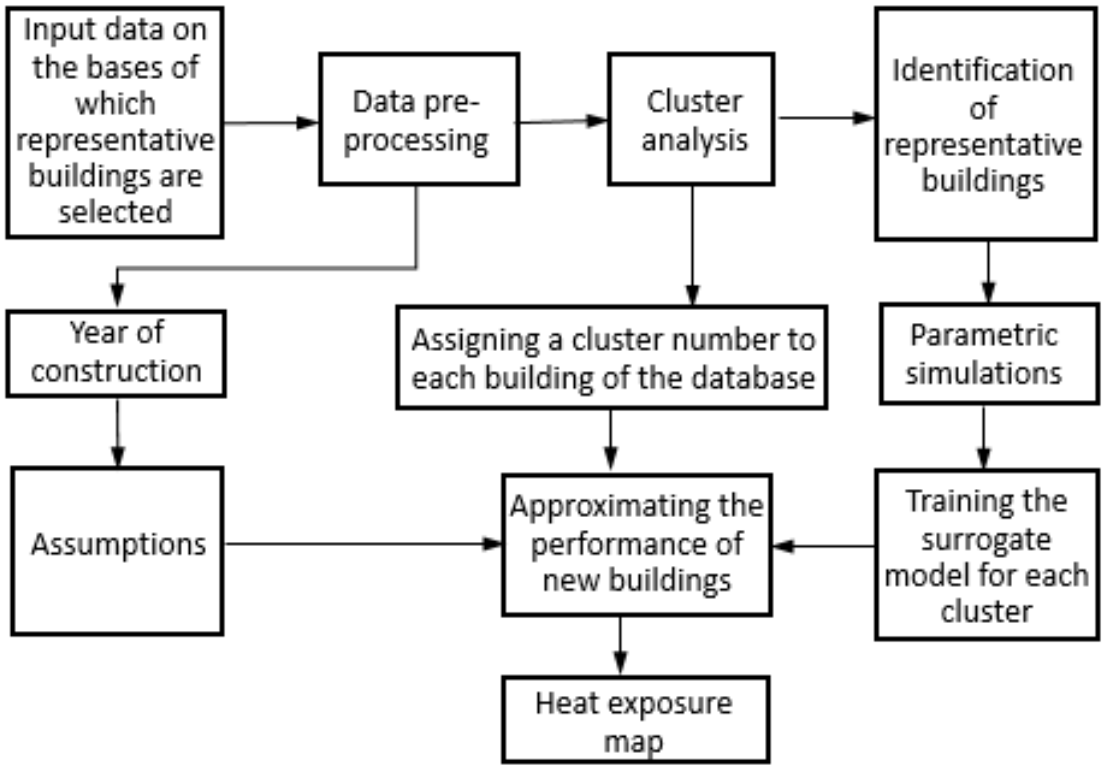


Figure 5-18 : Implementation of surrogate model process

5.2.6.2 Numerical implementation

Scikit-learn (Pedregosa et al. 2011), a free Python library dedicated to machine learning, is employed for data preprocessing, cluster analysis, training and testing of machine-learning based surrogate models.

Python Geopandas package (Jordahl et al. 2020) is another free library widely used in this thesis to read, and write vector-based spatial data format including ESRI shapefile, GeoJSON files and more. The library enables users to transform geospatial data into matrices and tables. This library also provides a high-level interface with matplotlib for making maps, in this case indoor overheating maps, shown below.

The Python library Numpy, and pandas have also been used extensively to manipulate matrices and to build numerical algorithm. Matplotlib, Seaborn, and Plotly libraries have been coupled with the Numpy arrays for the visualization of many parts of the data.

In addition to technical and numerical implementation, the application of surrogate model from cluster centroid to the rest of buildings is conditioned on the assumptions described in the characterization section of chapter 2.

Here is a brief list of those assumptions:

- The variance and range of training data used in the training section is applicable to all buildings within the cluster.
- U-values of exterior wall and U-value of roof for buildings in the cluster can be described as a function of year of construction, presented in Table 2-5.
- Window size for each building is estimated as a function of year of construction based on the data depicted in Figure 2-19.
- Occupants control natural air inflow rate and external shading of windows and according to the assumption presented in Chapter 2; there are three types of occupants: good user, intermediate user, and poor user.
- The performance of the floor is the sum of the performance of all thermal zones that are in the same floor. Because in a floor zones can be oriented in multiple directions at the same time, in the feature importance analysis for the floor the importance of orientation is significantly smaller compared to a number of other features, as shown in Figure 5-5.
- Type of window glazing and U-value of envelope elements can also be different whether a building is retrofitted or not retrofitted.

Following this process and assumptions, indoor performance of buildings for KM7_5, and KM7_6 were calculated and visualized on maps, presented below.

The assumptions illustrated earlier, can vary from building to building and from user to user, because so, the indoor overheating exposure map can also be different in accordance to the assumptions. Indoor overheating exposure map is then visualized for each index and each floor. Maps, below, show buildings of cluster KM7_5 that are concentrated in Nantes, city center.

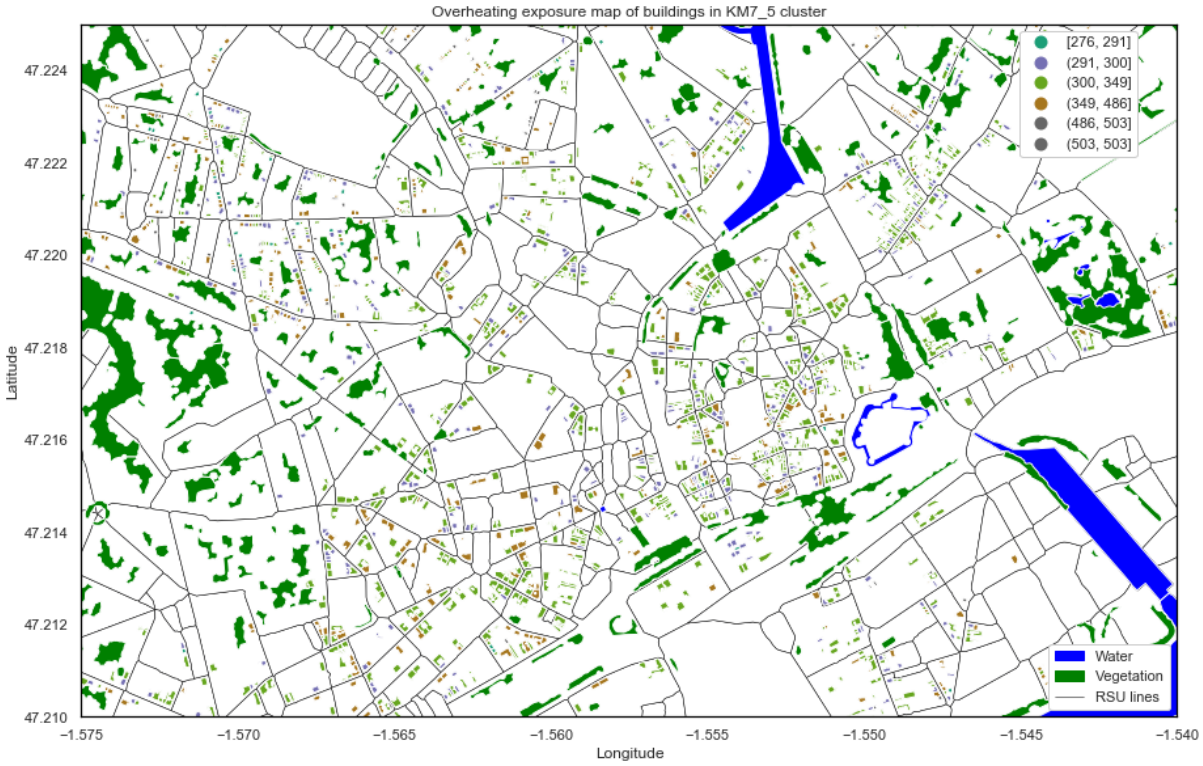


Figure 5-19 : Maximum consecutive hours above 27 degrees for the ground floor of KM7-5 cluster

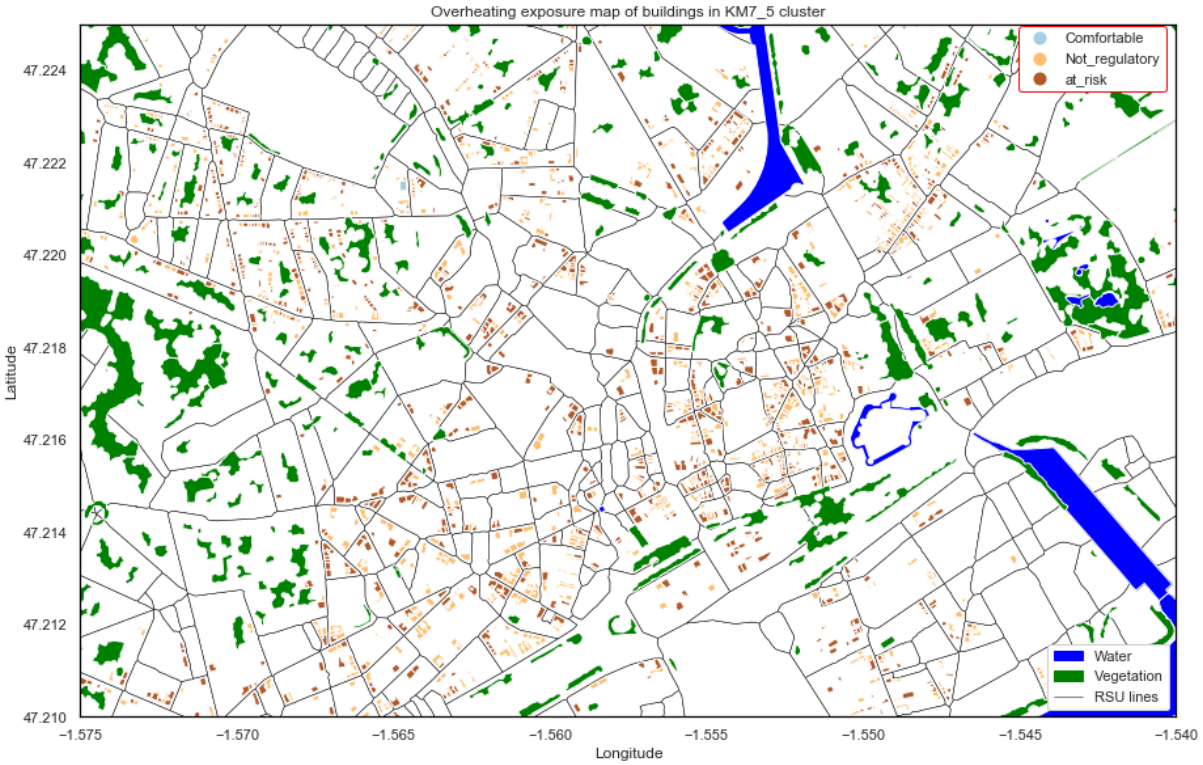


Figure 5-20 : RE2020 situation of ground floor calculated using multinomial logistic regression

In the maps illustrated in Figure 5-19 and Figure 5-20 the key assumptions are that occupant living in the **ground floor** of all buildings in KM7-5 are type 1 (poor user/non-adaptive user) and the buildings are retrofitted (have double-glazing window and U-value of envelop elements have been improved). Figure 5-21 and Figure 5-22, below, show exposure maps of **middle floor** based on the same assumptions.

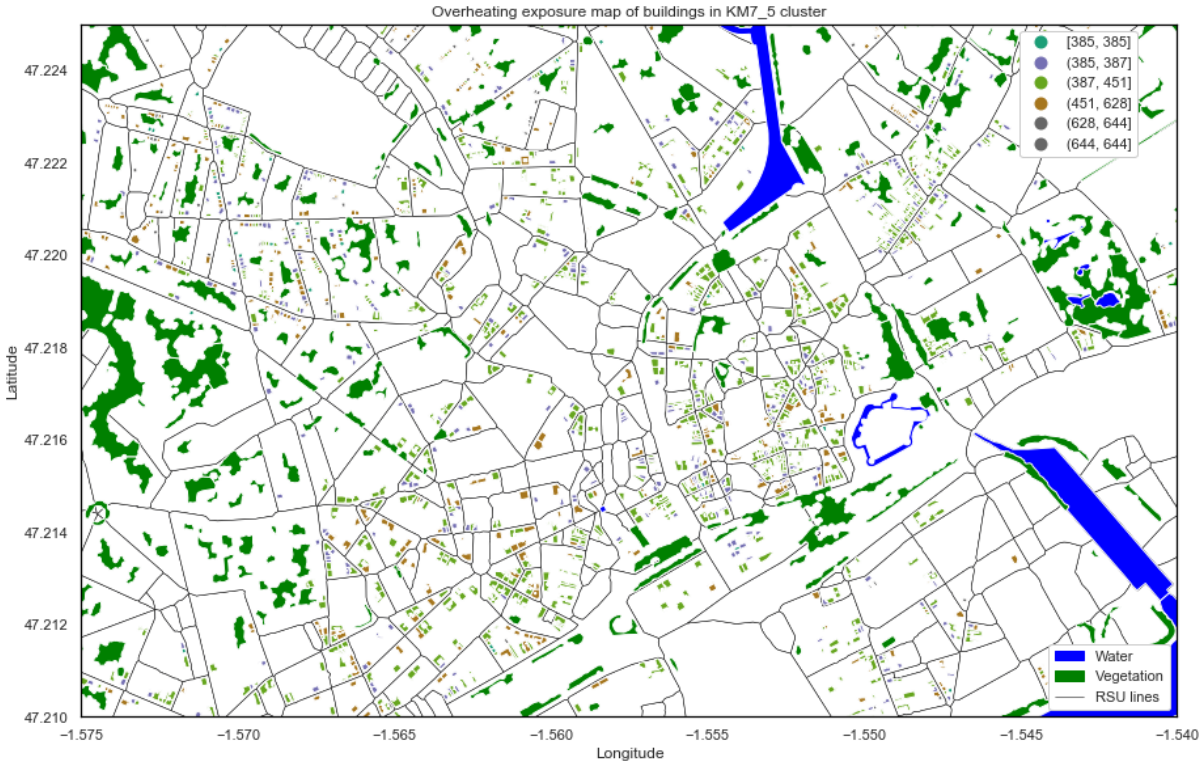


Figure 5-21 : Maximum consecutive hours above 27 degrees for the middle floor of KM7-5 cluster

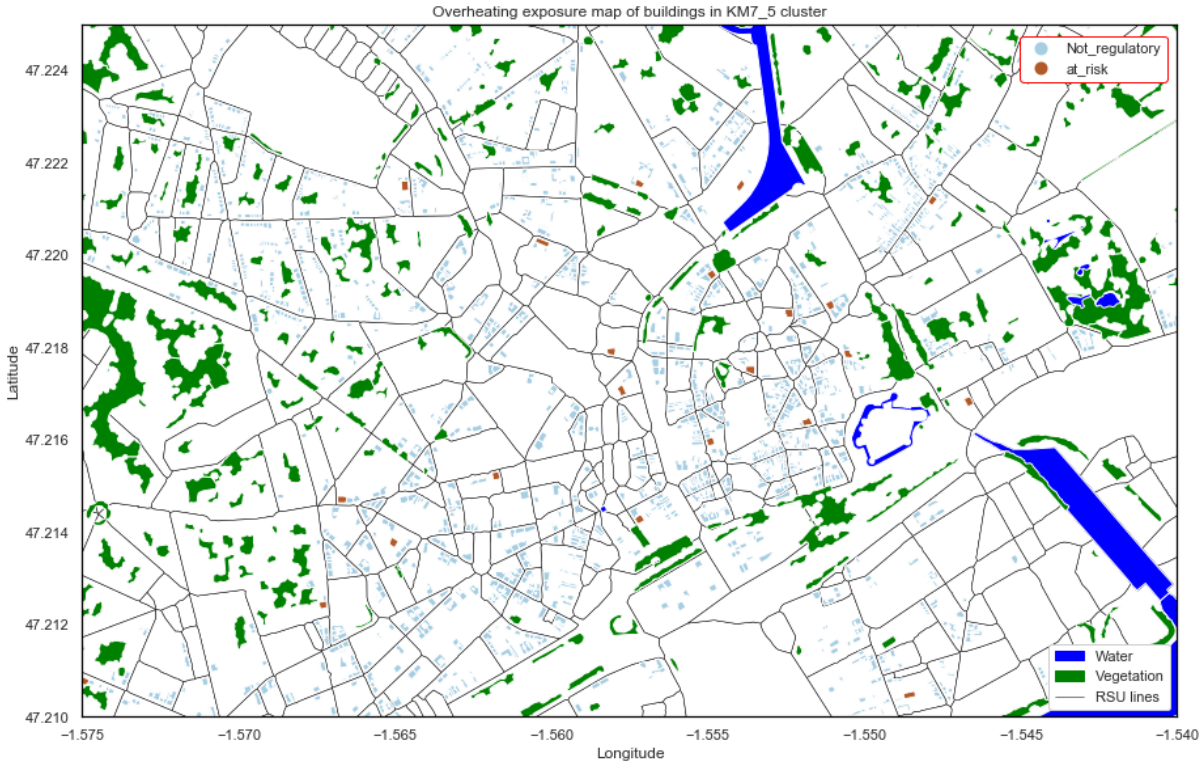


Figure 5-22 : RE2020 situation of middle floor calculated using multinomial logistic regression

Changing the type of user from **type 1** (poor user/not adaptive) to **type 3** (adaptive user) for middle floor, changes the indoor thermal conditions to as follows.

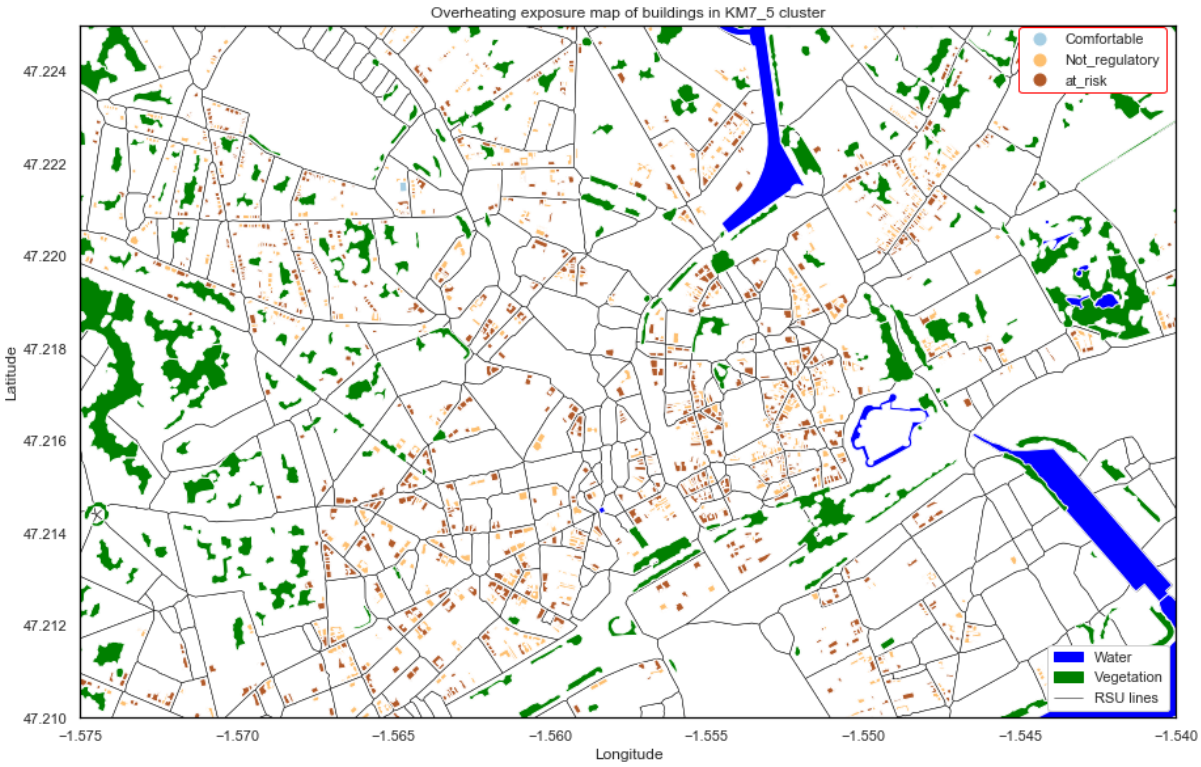


Figure 5-23 : RE2020 situation of middle floor calculated using multinomial logistic regression for occupant type 3

Significant change in the RE2020 situation, just by changes in the occupant type indicates that in order to draw an accurate picture of indoor overheating exposure of residential buildings it is essential to collect not only thermo-physical characteristic data of buildings but also gather more information about occupants and their capacity to adapt to extreme weather scenarios.

It is also important to note that indoor overheating exposure maps presented in this chapter are based on RCP 8.5 climate scenario, medium future (2040-2070) typical weather file from IPSL-SMHI climate model.

5.3 Conclusion

Indoor vulnerability assessment of buildings at city scale requires to identify efficient and easily interpretable method that could link a set of reference buildings (archetypes) to the rest of built stock in the city.

The research in the second chapter of this manuscript proposed an approach to identify and characterize residential reference buildings and this chapter suggested the application of surrogate modelling as an approach to extend the behavior simulated in each reference buildings to the rest of built stock that are within the same cluster.

To do so, first an experimental design based on D-optimal was constructed and parameters of building centroid (representative building) were simulated in accordance to D-optimal table.

Results of simulations for each reference building were collected and used to create a new database. Various surrogate modelling techniques based on regression and supervised machine learning were trained and tested on the database and the best performing one was identified.

When output of surrogate model is one or multiple continuous variables, ensemble gradient boosting regression technique performs significantly better than other techniques and when the output of surrogate model is a categorical variable (RE2020 situation), both multinomial logistic regression and ensemble gradient boosting classifier generate satisfactory results. However, to move forward with the process, multinomial logistic regression was selected.

Based on previous literature on indoor overheating and thermal comfort assessment that recommended studying building at zone scale, in this study parametric simulation and characterization of reference of building was carried out at scale of a thermal zone and performance of all zones within each floor was averaged, considering the area of zone, to the whole floor.

The result further indicate that with input parameters described in previous sections of this chapter, indoor thermal condition of ground floor and middle floor of the build stock can be estimated using a surrogate model, but assessing of attics' of building stock requires further research on parameters that were not taken into consideration in this study.

For ground and middle floors of buildings in summer overheating assessment, window wall ratio, U-value of exterior wall, occupant type (occupant behavior) and its vertical position (ground floor or middle floor), size of window, and type of glazing are the most important parameters. These parameters, however, are not sufficient to estimate summer overheating of the attic.

For attic, considering the difference that was noticed between reference building and other buildings within the same cluster, type of roof itself, number and type attic windows (inclined, vertical) also have significant influence but due to absence of data on these parameters they were not considered neither in the initial clustering of buildings and nor in the characterization stage.

As described in chapter 2 of this manuscript, occupants influence summer indoor conditions by manually opening and closing of external shadings, and window openings. These two passive strategies have been proven to significantly influence indoor comfort (our research article presented at IBPSA 2021 conference and a copy it is added in appendix 2-7) and are a major source of uncertainty in summer performance modelling of buildings.

The uncertainty with occupant behavior can drastically alter the way indoor heat exposure map is built, and what buildings are vulnerable and what are not to overheating. In addition, there is no reliable data on how occupants inside a building act during summer. Therefor a suggestion of this manuscript is to present vulnerability map in a scenario approach. In one scenario, an indoor overheating exposure map is presented for occupants that are unable or unwilling to manually operate external shades and windows. In another scenario, occupants are assumed to be well aware of the external environmental conditions and operate windows and external shades accordingly. This scenario-based approach could also be used to determine what buildings are vulnerable under what assumption.

Conclusions, Limitations and Prospects

The objective of this PhD thesis was to develop a methodology for integration of climate change and urban heat island data into decision support tools in urban planning to evaluate indoor overheating risk in residential buildings at city scale. For policy makers and other stakeholders this methodology would be used in assessment of local urban plans (PLUs) and climate change in terms of UHI and vulnerability of population to heatwaves.

This objective was partially achieved with creation of indoor heat exposure maps (ground floors and middle floors) that could be used as an input for local adaptation solutions such as reintegration of nature to the city, installation of misting systems or for mitigation measures such as buildings' refurbishment programs.

The developed methodology is based on the typological approach, dynamic thermal simulation of buildings and surrogate modelling to estimate the overheating of buildings at the city scale. Key novelty and strength of this methodology is its flexibility and interpretability. Thus, its could be used easily by practitioners. In other words, practitioners, depending on their needs, could modify, add, or remove one or more step to create a vulnerability map. Generated maps can then be used as an entry point for discussions in policymaking on how to layer adaptive responses to climate change and urban warming.

Synthesis of the results obtained

The first chapter of the manuscript is dedicated to the literature review of issues related to climate change, heatwaves, urban heat island and key concepts frequently encountered in climate impact studies. The literature review allowed to identify the problems with integration of global warming and other related challenges in decision support tools for urban planning. It was noted that almost all private and public sector organizations as well individuals need to be able to adapt to overheating due to climate change. Adaptation and implementation of policies that are aimed to reduce climate induced risk first require the development of a method that would enable practitioners to identify most vulnerable locations in a city and then layer the adaptive policy responses accordingly. To that end, the objective as stated earlier is the creation of indoor overheating exposure map.

Creation of indoor overheating exposure map at city scale, taking into consideration global warming, heatwaves and UHI requires consideration for the following crucial aspects:

- **Good quality data that represent climate change scenarios, heatwaves and urban heat island;**

The problem of access and processing of future climate data for building performance was studied and discussed in chapter 3 of this manuscript. A workflow was illustrated to generate typical future weather data for building simulations from high resolution dynamically downscaled regional climate models of EURO-CORDEX. Dynamically downscaled future typical weather data from Euro-Cordex were then compared with statistically downscaled weather data of meteonorm and historical heatwave data of Nantes. A simplified procedure to access and use historical weather from MeteoFrance was also presented. The chapter also

contains the process to project the influence of urban heat island on the weather data that are then used for building performance simulations.

- **Appropriate method to select reference buildings and perform energy and thermal comfort assessment simulations on the reference buildings;**

Identification and characterization of reference buildings were described in chapter 2 of the manuscript. First, the procedure to access, pre-process and calculate new parameters that allow practitioners to include the influence of urban heat island as well as building parameters were described. The raw data to build initial database was accessed from BD-TOPO Données foncières of CEREMA and DPE. Parameters such, façade density, sky-view factor, distance from peripheries and building shape elements were calculated for our case study city, Nantes.

After creation of building stock data, an unsupervised machine learning was chosen to divide residential building into clusters that share similar characteristics. Multiple clustering techniques were deployed and the quality of clustering was measured. Among the techniques, K-means clustering was identified as the one performing better than other techniques. Within each cluster, the real building closest to the fictive centroid of K-Means was chosen as the reference building of the cluster. Reference buildings were then characterized and additional parameters were added to it for parametric simulations.

- **Indices to measure overheating;**

The literature review on indoor overheating indices showed that researchers have developed numerous indices to describe overheating, but still there is no consensus on which one to use in the evaluation of indoor thermal conditions. Results of study on indices revealed that HI and DI indices were designed for extremely hot climate regions and may not be able to describe overheating of the buildings that are located in oceanic climate region.

Some indices such as those used by TM59 and RE2020 only concentrate on temperature and do not consider relative humidity and indoor air speed in the assessment.

Based on comparative analysis of indices, their individual behaviour, it is noticeable that many of the indices provide complementary results and some of them describe certain aspects of building that are ignored of given less attention in other indices.

Practitioners in BPS could create a better picture of indoor thermal performance of buildings at city scale if they use more than one index to describe it.

Given the influence of multiple factors on thermal vulnerability of occupants, a suggestion of this manuscript was that the indices should be able to describe the following characteristics of indoor conditions to present a realistic picture of indoor thermal conditions:

- Intensity of indoor operative temperature (maximum operative temperature);
- Duration of over temperature (maximum consecutive hours over a threshold);
- Adaptive capacity of occupants, considering their absence and/or presence (RE2020);
- Take into account the influence of indoor air speed, and relative humidity;

- **Development of an approach to extend the application of parametric simulation results on representative buildings to the rest of the built stock in the city.**

Chapter 5 of this manuscript is dedicated to the illustration of surrogate model development.

The aim was to develop a surrogate model for each cluster of buildings stock of Nantes city. Thus, following the identification and characterization of reference buildings, a parametric simulation was carried out on centroid (representative building) of KM7-5 and KM7_6 clusters and surrogate models for these two clusters were developed.

In the city-scale energy and thermal performance studies of buildings practitioners/researchers often use a simplified thermal zoning configurations (one zone per floor or the whole building as a zone), but in this study a more complex zoning configuration was used, based on the characteristics of identified representative building. Ground floor and middle floor were divided into multiple zones and attic was assumed as a single zone.

This decision to divide floor into zones were made because indoor overheating, similar to thermal comfort, is spatially as well temporally variable and multiple zones in each floor are better able to describe indoor thermal conditions than taking the whole floor or the whole building as a single zone.

The results of simulations from zones were then averaged to calculate indoor thermal indices of the floor.

D-optimal design of experiment was used to optimize the number of variations for parametric simulations. In each run of simulation, input parameter variations and corresponding calculated output parameters of simulations were collected to build a new database for each cluster of buildings. These database or matrices are used to construct the surrogate models.

The first step in the development of surrogate model is the selection of the method.

For the selection of surrogate models for clusters KM7-5 and KM7-6, data of attic were filtered out from middle and ground floors and then the data was split into training (0.7) and test (0.3) parts. Six different regression-based polynomial and supervised machine learning techniques (Simple Multiple Linear Regression, Multiple Linear regression with interaction of input parameters, Decision Tree, Support Vector Machine, Gradient Boosting Regression, Random Forest Regression) were used to train and test the performance of each model for continuous output variables. For categorical output variable (RE2020 situations) two classification-based techniques (Multinomial Logistic regression, Gradient Boosting Classifier) were used.

After comparison of results, for continuous indices, ensemble gradient boosting regressor performed better than other techniques in prediction of test data. For categorical output variable, both technique generated similar results with multinomial logistic regression performing slightly better than ensemble gradient boosting classifier.

Using the techniques described above, indoor thermal indices for a new building randomly selected from cluster KM7_5 and KM7_6 were estimated. The results of estimation with BES and surrogate modelling simulations were then compared. Surrogate models estimated all indices for ground floor and middle floor with a good degree of accuracy. However, the

surrogate model did not estimate all indoor summer indices for attic of new building satisfactorily due to one or both of the following reasons:

1- Input parameters that influence indoor overheating are different for attic from ground and middle floors, therefore it is necessary to collect more data on attic and its properties, in order to construct a surrogate model capable of predicting overheating in attics;
2- Attic is more exposed to outdoor environment, and its indoor thermal conditions are more influenced by the surrounding environment than by thermo-physical properties used as input for surrogate model.

Limitations and Prospects

This thesis attempted to path the way for researcher and practitioners to develop a robust methodology to integrate data from climate change, heatwaves and urban heat island in decision support tools for urban planning, by creation of indoor overheating exposure map.

Nantes city was used as a case study city and using available open-source data on climate, buildings, and urban parameters, an indoor overheating exposure map was created.

Some of the data, techniques and methods applied in this manuscript, looked from a critical point of view, deserve deeper research. In the following, major limitations of each step taken in the creation of overheating exposure map are discussed.

Chapter 1:

Scope and method of the study

In this thesis, a process-based approach was used to organize tasks and achieve the final objective. Main advantages of process-based approach is that it is scalable (can be implemented by a team of practitioners simultaneously) and over time the process can evolve and be improved. However, application of this approach on a research problem can be time consuming, especially when it involves dealing with various subjects. Researcher/practitioner has to find a balance between relevance to the question and level of detail in each step of the way, which is not always clear. For instance, in the third chapter of this manuscript we used one fixed threshold approach to measure heatwaves; however, there are multiple other approaches that could have been used.

In the future studies, depending on the availability of resources a more detailed study could be carried on these steps, to see if other approaches better describe the problem.

Absence of data on social vulnerability at city scale at a fine resolution

At the beginning of research effort, the objective was to build a vulnerability map taking into consideration both exposure rate to overheating and adaptive capacity of occupants, but absence of data on socio-economic status of city-habitants at a fine resolution, which play a significant role on the degree of vulnerability according to the literature review, led us to build a heat exposure map. In the future, following this methodology, socio-economic data could be collected to build a social vulnerability map, then after determining the relative influence of

social vulnerability and exposure rate they could be superposed one over another to create an overheating vulnerability map.

Chapter 2:

Identification of representative buildings

In the second chapter of this manuscript, a method was proposed to identify representative buildings and then each identified representative building was characterized.

The chapter starts by a literature review of approaches to divide buildings into groups. In this chapter it was decided to use unsupervised machine learning to divide buildings into clusters and from each cluster, select one representative building. This approach was chosen to minimize the familiarity and experience bias of practitioners and compare all buildings to one another. The latter is an advantage of machine learning techniques because a human observer can compare a limited number of objects to one another, but machines can compare thousands and even more objects to one another. The quality of representative buildings identified with this approach, on the other hand, is highly influenced by the type and properties of input data.

In this manuscript, input parameters were selected in such a way to include the influence of UHI effect, year of construction and building morphological parameters. UHI effect parameters for each cluster were averaged in each cluster and the weather file for the cluster was modified using UWG to project the influence UHI.

Combination of UHI and building parameters in this study increased the level of variations and therefore, the degree of dissimilarity between clusters. The values of three indices, which are used to measure the quality of clusters, are relatively low as compared to studies in other domains.

In the future studies, UHI parameters could be used alone to identify clusters of RSUs (neighborhoods) that are similar to one another and within each cluster of neighborhoods a new building typology could be constructed, using only building parameters.

Another issue that needs to be taken into consideration in future building typology construction is urban expansion and/or increase in the density of buildings in urban areas.

Smaller number of representative buildings

Clustering was carried out on a various number of clusters. The degree of dissimilarity, as measured by three indices, was slightly larger when number of clusters was 7 compared to number of clusters equal to 6 and 5. Selecting number of clusters to 5 or 6 could reduce the number of representative buildings but still represent residential buildings. In the future, a smaller number of representative buildings could be selected for the sake practicality; if/when, the degree of dissimilarity between them is not significant.

Quality of data

The quality of some of the data used as input for clustering could be improved. That green coverage ratio of city data of BD-Topo was used, but some research has shown that the resolution of the parameters could be improved if more up-to-date satellite images are used.

Characterization assumptions

Some of the assumptions on occupant behavior and building thermo-physicals as a function of year of construction can evolve over time. Some buildings that were not retrofitted are now retrofitted. The assumptions about schedules on windows status that influence natural air-inflow rate and external shading were based on limited literature sources and on our study on one building. This is a small sample, and in the future further research is required to ensure the assumptions are in line with reality

Chapter 3:

The third chapter of this manuscript is about climate data for future, historical weather data, heatwaves, and projection of UHI influence on weather data. The focus was more on practicality and how to make them accessible to Building Performance Evaluation (BPE) practitioners. The steps and details described in this chapter have some limitations, which are listed briefly in the following:

- The raw data of all necessary climate variables used to construct typical weather data for the future are not bias-adjusted, therefore only a comparative assessment can be done with them;
- The thresholds used in heatwave detection, are fixed absolute thresholds for maximum and minimum daily temperatures and are based on the worst historical heatwave event. In future studies, for France or any other locations around the world, other methods of heatwaves measurement could also be used;
- Comparative analysis of weather files in this study was focused on medium future and on typical weather files. Further study is suitable to perform a comparative analysis of observed heatwave weather data of 2003 with near, and far future weather files, and with artificially generated extreme hot years;
- Further research is suggested in the creation of Extreme Hot Year weather files from EURO-CORDEX to be compared with historical extreme weather years;
- In this study, future typical weather files of three climate models were compared to one another and historical weather data of 2003 for our case study, Nantes. Multiple other combination of GCMs and RCMs are possible on the EUROCORDEX platform and practitioners/researchers could further explore their potential for BPS evaluation.

Chapter 4:

The chapter of manuscript has two parts. First part summarizes a literature of review of cases studies that performed sensitivity analysis on thermal comfort and energy consumption of buildings. An study was carried out to summarize and aggregate the results of sensitivity analyses in various papers and to identify the most influential parameters. A few limitation of method used in aggregation of results are presented as follows:

- Method employed in aggregation of parameters from different sources is not a standard method and was devised to answer the needs for this specific case. Alternative approaches may also exist that could possibly change the outcomes;

- Selected papers only focused on building parameters. Urban morphological and microclimate parameters that also influence energy consumption and thermal comfort were not taken into consideration;
- Sensitivity analysis results collected in these papers were conducted in multiple geographical locations with a wide range of climatic conditions, which in turn can also introduce inconsistency in the study;
- Some papers ranked the importance of parameter by giving it an index (percentage/ratio), some just presented ranks on a histogram and some named them in order of importance. In this brief review only the order of influence/affect (first, second, third, etc.) of results and total number of parameters were taken into account.
- Some important parameters were not studied in any paper and therefore they were not considered in parametric simulations. It is the case, for instance, most of the attic characteristics such as horizontal glazing, shape of the roof, that were at the end of this work, underlined as critical factors influencing heat exposure in this part of the buildings.

The second part of this chapter is performs a detailed study of indices used in indoor overheating. It starts with a literature review and by explanation of major techniques to used by researcher and other building actors.

Due to time limitations and the fact that comparing indices was not the core objective of this manuscript, a small number of indices were compared to one another. Further research on the combination of different indices to describe indoor overheating could be conducted.

Additionally, some of the indices that were employed in this study did not have standard thresholds, above or below which a building or thermal zone could be declared as “overheated” or comfortable. For instance, maximum consecutive hours about 27 °C. In this index, the duration of exposure to over-temperature is expressed in hours, but no reliable information was found to indicate above what number of hours a person with feel thermally distressed. Without thresholds, only a comparative analysis of thermal zones could be done, using this index. Determining the threshold of thermal distress due to long term exposure to over-temperature could be an area of future research.

Chapter 5:

The main objective of the research effort was to develop a new fast and efficient method to estimate how residential buildings perform under climate change and heatwaves taking into consideration urban heat island.

In this manuscript, it was decided to build surrogate model to extend the results obtained from cluster centroid (representative building) to the rest of built stock. The decision was made with three factors in mind: practicality, accuracy and interpretability of the results.

Assessed from a practicality point of view, the method proposed here, as shown in chapter 5, can estimate overheating for ground and middle floors at city scale. However, number of parametric simulations increases with number of clusters and parameters such as area and window size that requires changing the geometrical model can make the simulation process tedious. In the future research, a suggestion of this manuscript is to find ways to streamline the parametric simulations of areas and window sizes and other parameters.

A major of limitation/shortcoming of this research effort was that the surrogate model did not perform as well as it was expected for attic. A suggestion to overcome this shortcoming is to perform a detailed sensitivity analysis on attic (last floor) of residential buildings to identify what parameters influence overheating and how they could be added to the parameters that impact overheating of other floors in a building.

An important area of further research within this methodology is the introduction of renewable energy production and passive cooling techniques as a way to offset the risk associated with overheating in residential buildings at city-scale.

Overall, the methodology and workflow presented in this manuscript laid the path for practitioners and researchers in urban planning to take into account data from climate change, historical heatwaves and urban heat island in decision support tools, applied to map vulnerability of city habitant. In each chapter, the focus was on the practicality and interpretability of the approaches.

References

Aghamohammadi, Nasrin et al. 2022. 'Perceived Impacts of Urban Heat Island Phenomenon in a Tropical Metropolitan City: Perspectives from Stakeholder Dialogue Sessions'. *Science of The Total Environment* 806: 150331.

Agyapong, Kwame Boakye, Dr J B Hayfron-Acquah, and Dr Michael. 2016. 'An Overview of Data Mining Models (Descriptive and Predictive)'. 4(5): 8.

'Ali et al. - GIS-Based Residential Building Energy Modeling at .Pdf'.

Ali, Usman et al. 2019. 'A Data-Driven Approach for Multi-Scale Building Archetypes Development'. *Energy and Buildings* 202: 109364.

Ali, Usman, Mohammad Haris Shamsi, Cathal Hoare, and James ODonnell. 'GIS-Based Residential Building Energy Modeling at District Scale'. : 9.

Allacker, Karen et al. 2019. 'Energy Simulation and LCA for Macro-Scale Analysis of Eco-Innovations in the Housing Stock'. *The International Journal of Life Cycle Assessment* 24(6): 989–1008.

Allaire, D., and K. Willcox. 2010. 'Surrogate Modeling for Uncertainty Assessment with Application to Aviation Environmental System Models'. *AIAA Journal* 48(8): 1791–1803.

Altan, Hasim, Roberto Padovani, and Arman Hashemi. 2016. 'Building Performance and Simulation'. In *ZEMCH: Toward the Delivery of Zero Energy Mass Custom Homes*, Springer Tracts in Civil Engineering, ed. Masa Noguchi. Cham: Springer International Publishing, 311–38. http://link.springer.com/10.1007/978-3-319-31967-4_11 (December 16, 2021).

ASHRAE-55, ASHRAE. 2017. *ANSI/ASHRAE Standard 55-2017. Thermal Environmental Conditions for Human Occupancy; Conditions That Provide Thermal Comfort; Page 14. ISSN 1041-2336*. . Standard.

Baborska-Narożny, Magdalena, Fionn Stevenson, and Magdalena Grudzińska. 2017. 'Overheating in Retrofitted Flats: Occupant Practices, Learning and Interventions'. *Building Research & Information* 45(1–2): 40–59.

Ballarini, Ilaria et al. 2017. 'Energy Refurbishment of the Italian Residential Building Stock: Energy and Cost Analysis through the Application of the Building Typology'. *Energy Policy* 105: 148–60.

Bande, Lindita et al. 2019. 'Validation of UWG and ENVI-Met Models in an Abu Dhabi District, Based on Site Measurements'. *Sustainability* 11(16): 4378.

Battersby, Stephen. 2016. *Clay's Handbook of Environmental Health.* New York: Taylor and Francis. http://public.ebookcentral.proquest.com/choice/publicfullrecord.aspx?p=4578981 (November 3, 2021).

Beizaee, A., K.J. Lomas, and S.K. Firth. 2013. 'National Survey of Summertime Temperatures and Overheating Risk in English Homes'. *Building and Environment* 65: 1–17.

Belcher, Se, Jn Hacker, and Ds Powell. 2005. 'Constructing Design Weather Data for Future Climates'. *Building Services Engineering Research and Technology* 26(1): 49–61.

Berger, Julien et al. 2017. 'Review of Reduced Order Models for Heat and Moisture Transfer in Building Physics with Emphasis in PGD Approaches'. *Archives of Computational Methods in Engineering* 24(3): 655–67.

Bernard, Jérémy. 2017. 'Geographic and meterological maps of spatial and temporal variation of air temperature in urban zones'.

Bhatnagar, Mayank, Jyotirmay Mathur, and Vishal Garg. 2018. 'Determining Base Temperature for Heating and Cooling Degree-Days for India'. *Journal of Building Engineering* 18: 270–80.

Bhojani, Shital H. 2016. 'Commerce Computer Science'. 5(5): 4.

Boccalatte, A., M. Fossa, L. Gaillard, and C. Menezo. 2020. 'Microclimate and Urban Morphology Effects on Building Energy Demand in Different European Cities'. *Energy and Buildings* 224: 110129.

Bocher, Erwan, Gwendall Petit, Jérémy Bernard, and Sylvain Palominos. 2018. 'A Geoprocessing Framework to Compute Urban Indicators: The MApUCE Tools Chain'. *Urban Climate* 24: 153–74.

Brasche, Sabine, and Wolfgang Bischof. 2005. 'Daily Time Spent Indoors in German Homes – Baseline Data for the Assessment of Indoor Exposure of German Occupants'. *International Journal of Hygiene and Environmental Health* 208(4): 247–53.

Bueno, Bruno, Aiko Nakano, and Leslie Norford. 2015. 'Urban Weather Generator: A Method to Predict Neighborhood-Specific Urban Temperatures for Use in Building Energy Simulations'. In *12th Symposium on the Urban Environment,* , 6. http://www.meteo.fr/icuc9/LongAbstracts/udc5-4-6531531_a.pdf.

Bueno, Bruno, Leslie Norford, Julia Hidalgo, and Grégoire Pigeon. 2013. 'The Urban Weather Generator'. *Journal of Building Performance Simulation* 6(4): 269–81.

Cerezo, Carlos et al. 2017. 'Comparison of Four Building Archetype Characterization Methods in Urban Building Energy Modeling (UBEM): A Residential Case Study in Kuwait City'. *Energy and Buildings* 154: 321–34.

Cerezo, Carlos, Julia Sokol, Christoph Reinhart, and Adil Al-Mumin. 'THREE METHODS FOR CHARACTERIZING BUILDING ARCHETYPES IN URBAN ENERGY SIMULATION. A CASE STUDY IN KUWAIT CITY.' : 8.

Chartered Institution of Building Services Engineers. 2013. *The Limits of Thermal Comfort Avoiding Overheating in European Buildings ; CIBSE TM52: 2013*. London: CIBSE.

Cheng, Kai, Zhenzhou Lu, Chunyan Ling, and Suting Zhou. 2020. 'Surrogate-Assisted Global Sensitivity Analysis: An Overview'. : 28.

Chitra, K, and Dr D Maheswari. 2017. 'A Comparative Study of Various Clustering Algorithms in Data Mining'. : 7.

Civel, Edouard, and Jeremy Elbeze. 2016. 'Energy Efficiency in French Homes : How Much Does It Cost ?' *Climate Economics Chair of Paris-Dauphine University*: 30.

Coma, Julià et al. 2019. 'Comparative Analysis of Energy Demand and CO2 Emissions on Different Typologies of Residential Buildings in Europe'. *Energies* 12(12): 2436.

Costanzo, Vincenzo, Gianpiero Evola, and Luigi Marletta. 2021. *Urban Heat Stress and Mitigation Solutions: An Engineering Perspective*. 1st ed. London: Routledge. https://www.taylorfrancis.com/books/9781003045922 (November 23, 2021).

Daniela, Jacob et al. 2020. 'Regional Climate Downscaling over Europe: Perspectives from the EURO-CORDEX Community'. *Regional Environmental Change* 20(2): 51–66.

Dascalaki, Elena G., Kalliopi G. Droutsa, Constantinos A. Balaras, and Simon Kontoyiannidis. 2011. 'Building Typologies as a Tool for Assessing the Energy Performance of Residential Buildings – A Case Study for the Hellenic Building Stock'. *Energy and Buildings* 43(12): 3400–3409.

Davies, David L., and Donald W. Bouldin. 1979. 'A Cluster Separation Measure'. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1(2): 224–27.

Davila, Carlos Cerezo, Christoph Reinhart, and Jamie Bemis. 'Modeling Boston: A Workflow for the Generation of Complete Urban Building Energy Demand Models from Existing Urban Geospatial Datasets'. : 28.

Dean, Angela M., Max Morris, John Stufken, and Derek Bingham, eds. 2015. *Handbook of Design and Analysis of Experiments*. Boca Raton London New York: CRC Press, a Chapman & Hall book.

Deru, M. et al. 2011. *U.S. Department of Energy Commercial Reference Building Models of the National Building Stock*. http://www.osti.gov/servlets/purl/1009264-pitlfN/ (December 13, 2019).

Dierickx, Florian. 2019. *Copernicus Climate Change Programme: User LearningService Content*. https://bookdown.org/floriandierickx/bookdown-demo/bookdown-demo.pdf.

Doherty, Sarah et al. 2018. *National Climate Assessment Report. Chapter 2: Our Changing Climate*. https://nca2018.globalchange.gov/chapter/2/ (August 4, 2021).

Droutsa, K.G., S. Kontoyiannidis, E.G. Dascalaki, and C.A. Balaras. 2014. 'Ranking Cost Effective Energy Conservation Measures for Heating in Hellenic Residential Buildings'. *Energy and Buildings* 70: 318–32.

Eldred, Michael, Anthony Giunta, and S. Collis. 2004. 'Second-Order Corrections for Surrogate-Based Optimization with Model Hierarchies'. In *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, New York: American Institute of Aeronautics and Astronautics. http://arc.aiaa.org/doi/10.2514/6.2004-4457 (May 23, 2022).

Epstein, Yoram, and Daniel S. Moran. 2006. 'Thermal Comfort and the Heat Stress Indices'. *Industrial Health* 44(3): 388–98.

Erlandsen, Helene Birkelund et al. 2020. 'A Hybrid Downscaling Approach for Future Temperature and Precipitation Change'. *Journal of Applied Meteorology and Climatology* 59(11): 1793–1807.

European Commission. Joint Research Centre. 2020. *Climate Change Impacts and Adaptation in Europe: JRC PESETA IV Final Report.* LU: Publications Office. https://data.europa.eu/doi/10.2760/171121 (October 18, 2021).

F. Holmgren, William, Clifford W. Hansen, and Mark A. Mikofski. 2018. 'Pvlib Python: A Python Package for Modeling Solar Energy Systems'. *Journal of Open Source Software* 3(29): 884.

Famuyibo, Adesoji Albert, Aidan Duffy, and Paul Strachan. 2012. 'Developing Archetypes for Domestic Dwellings—An Irish Case Study'. *Energy and Buildings* 50: 150–57.

Forrester, Alexander I. J., András Sóbester, and A. J. Keane. 2008. *Engineering Design via Surrogate Modelling: A Practical Guide*. Chichester, West Sussex, England ; Hoboken, NJ: J. Wiley.

Fosas, Daniel et al. 2018. 'Mitigation versus Adaptation: Does Insulating Dwellings Increase Overheating Risk?' *Building and Environment* 143: 740–59.

Garreau, Enora. 2021. 'Development of a parsimony analysis methodology for urban energy simulation/Développement d'une méthodologie d'analyse de la parcimonie pour la simulation énergétique urbaine'. Universite Paris PSL. https://pastel.archives-ouvertes.fr/tel-03403586.

———. 2021. 'Solar Shading and Multi-Zone Thermal Simulation: Parsimonious Modelling at Urban Scale'. *Energy and Buildings* 249: 111176.

Gera, Mansi, and Shivani Goel. 2015. 'Data Mining - Techniques, Methods and Algorithms: A Review on Tools and Their Validity'. *International Journal of Computer Applications* 113(18): 22–29.

Géron, Aurélien. 2017. *Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. First edition. Beijing ; Boston: O'Reilly Media.

Ghiassi, Neda, and Ardeshir Mahdavi. 2017. 'Reductive Bottom-up Urban Energy Computing Supported by Multivariate Cluster Analysis'. *Energy and Buildings* 144: 372–86.

Ghuman, Sukhdev Singh. 2014. 'A Review of Data Mining Techniques'. : 6.

Goupy, Jacques. 2013. *Introduction aux plans d'expériences: avec applications*. 5e éd. Paris: 'l'Usine nouvelle' Dunod.

Green, Helen K. et al. 2016. 'Mortality during the 2013 Heatwave in England – How Did It Compare to Previous Heatwaves? A Retrospective Observational Study'. *Environmental Research* 147: 343–49.

Guillaumet, M P, P Borges, M Rosas-Casals, and O Travesset-Baro. 'BUILDING ARCHETYPES IN URBAN ENERGY MODELS. A COMPARATIVE CASE STUDY OF DETERMINISTIC AND STATISTICAL METHODS IN ANDORRA.' : 10.

Hallegatte, Stéphane, Franck Lecocq, and Christian de Perthuis. 2010. *Economy of adaptation to climate change / Economie de l'adaptation au changement climatique*. Conseil Economique pour le Développement Durable. https://archive.wikiwix.com/cache/index2.php?url=http%3A%2F%2Fwww.ladocumentationfrancaise.fr%2Fdocfra%2Frapport_telechargement%2Fvar%2Fstorage%2Frapports-publics%2F104000108.pdf.

Hamdy, Mohamed, Salvatore Carlucci, Pieter-Jan Hoes, and Jan L.M. Hensen. 2017a. 'The Impact of Climate Change on the Overheating Risk in Dwellings—A Dutch Case Study'. *Building and Environment* 122: 307–23.

———. 2017b. 'The Impact of Climate Change on the Overheating Risk in Dwellings—A Dutch Case Study'. *Building and Environment* 122: 307–23.

Heine Kristensen, Martin. 2018. 'Urban Building Energy Modelling for Retrofit Analysis under Uncertainty'. Ph.D. Aarhus University. https://ebooks.au.dk/index.php/aul/catalog/book/301 (December 9, 2019).

Hertel, Sabine, Alain Le Tertre, Karl-Heinz Jöckel, and Barbara Hoffmann. 2009. 'Quantification of the Heat Wave Effect on Cause-Specific Mortality in Essen, Germany'. *European Journal of Epidemiology* 24(8): 407–14.

Herten, Joachim van der, Tom Van Steenkiste, Ivo Couckuyt, and Tom Dhaene. 2017. 'Surrogate Modelling with Sequential Design for Expensive Simulation Applications'. In *Computer Simulation*, ed. Dragan Cvetkovic. InTech. http://www.intechopen.com/books/computer-simulation/surrogate-modelling-with-sequential-design-for-expensive-simulation-applications (March 23, 2022).

Hidalgo, Julia et al. 2019. 'Comparison between Local Climate Zones Maps Derived from Administrative Datasets and Satellite Observations'. *Urban Climate* 27: 64–89.

Hosseini, Mirata, Anahita Bigtashi, and Bruno Lee. 2021. 'Generating Future Weather Files under Climate Change Scenarios to Support Building Energy Simulation – A Machine Learning Approach'. *Energy and Buildings* 230: 110543.

I4CE. 2019. 'Heat Wave, Stop Being Surprised',. *Institute for climate economics*. https://www.i4ce.org/heat-wave-stop-being-surprised-lets-invest-better/.

IPCC. 2018. *Annex I: Glossary [Matthews, J.B.R. (Ed.)]. Global Warming of 1.5°C. An IPCC Special Report on the Impacts of Global Warming of 1.5°C above Pre-Industrial Levels and Related Global Greenhouse Gas Emission Pathways, in the Context of Strengthening the Global Response to the Threat of Climate Change, Sustainable Development, and Efforts to Eradicate Poverty*. [Masson-Delmotte, V., P. Zhai, H.-O. Pörtner, D. Roberts, J. Skea, P.R. Shukla, A. Pirani, W. Moufouma-Okia, C. Péan, R. Pidcock, S. Connors, J.B.R. Matthews, Y. Chen, X. Zhou, M.I. Gomis, E. Lonnoy, T. Maycock, M. Tignor, and T. Waterfield (eds.)].

ISO 15927-4. 2005. '[EN ISO 15927-4_2005] -- Hygrothermal Performance of Buildings. Calculation and Presentation of Climatic Data. Hourly Data for Assessing the Annual Energy Use for Heating and .Pdf'.

Jacob, Daniela et al. 2014. 'EURO-CORDEX: New High-Resolution Climate Change Projections for European Impact Research'. *Regional Environmental Change* 14(2): 563–78.

Jänicke, Britta, Dragan Milošević, and Suneja Manavvi. 2021. 'Review of User-Friendly Models to Improve the Urban Micro-Climate'. *Atmosphere* 12(10): 1291.

Jordahl, Kelsey et al. 2020. *Geopandas/Geopandas: V0.8.1*. Zenodo. https://zenodo.org/record/3946761 (July 16, 2022).

Jorgji, O et al. 2019. 'Analysing the Impact of Retrofitting and New Construction through Probabilistic Life Cycle Assessment. A Method Applied to the Environmental-Economic Payoff Value of an Intervention Case in the Albanian Building Sector'. *IOP Conference Series: Earth and Environmental Science* 323: 012184.

Kamal, Athar et al. 2021. 'Impact of Urban Morphology on Urban Microclimate and Building Energy Loads'. *Energy and Buildings*: 111499.

Kim, Se Woong, and Robert D. Brown. 2021. 'Urban Heat Island (UHI) Intensity and Magnitude Estimations: A Systematic Literature Review'. *Science of The Total Environment* 779: 146389.

Kim, Sun Hye, and Boukouvala Fani. 2019. 'Machine Learning-Based Surrogate Modeling for Data-Driven Optimization: A Comparison of Subset Selection for Regression Techniques'. *Optimization letters*: 22.

Korolija, Ivan, Ljiljana Marjanovic-Halburd, Yi Zhang, and Victor I. Hanby. 2013. 'UK Office Buildings Archetypal Model as Methodological Approach in Development of Regression Models for Predicting Building Energy Consumption from Heating and Cooling Demands'. *Energy and Buildings* 60: 152–62.

Kotol, Martin, Carsten Rode, Geo Clausen, and Toke Rammer Nielsen. 2014. 'Indoor Environment in Bedrooms in 79 Greenlandic Households'. *Building and Environment* 81: 29–36.

Kragh, J., and K.B. Wittchen. 2014. 'Development of Two Danish Building Typologies for Residential Buildings'. *Energy and Buildings* 68: 79–86.

Kumar, Sanjay et al. 2019. 'Comparative Study of Thermal Comfort and Adaptive Actions for Modern and Traditional Multi-Storey Naturally Ventilated Hostel Buildings during Monsoon Season in India'. *Journal of Building Engineering* 23: 90–106.

Lane, Kathryn et al. 2014. 'Extreme Heat Awareness and Protective Behaviors in New York City'. *Journal of Urban Health* 91(3): 403–14.

Lauzet, Nicolas et al. 2019. 'How Building Energy Models Take the Local Climate into Account in an Urban Context – A Review'. *Renewable and Sustainable Energy Reviews* 116: 109390.

———. 2019. ‘Taking into account climate change and urban overheating in the design of energy frugal buildings with focus on thermal comfort of occupants: methodological propositions / Prise en compte cumulee du chnagement climatique et des surchauffes urbaines en phase amont de conception frugale des batiments centree sur le confort des occupants: Des propositions methodologiques’. Universite Bretagne Sud.

Leconte, Francois. 2014. ‘Characterization of urban heat island based on climatic zoning and mobile measurements : Case study of Nancy / Caractérisation des îlots de chaleur urbain par zonage climatique et mesures mobiles : cas de Nancy’. Université de Lorraine. http://www.theses.fr/2014LORR0255.

Lee, W. Victoria, and Jeffrey Shaman. 2017. ‘Heat-Coping Strategies and Bedroom Thermal Satisfaction in New York City’. *Science of The Total Environment* 574: 1217–31.

Lemonsu, Aude et al. 2013. ‘Evolution of the Parisian Urban Climate under a Global Changing Climate’. *Climatic Change* 116(3–4): 679–92.

Li, Huidong et al. 2018. ‘Interaction between Urban Heat Island and Urban Pollution Island during Summer in Berlin’. *Science of The Total Environment* 636: 818–28.

Li, Xinyi et al. 2018. ‘Developing Urban Residential Reference Buildings Using Clustering Analysis of Satellite Images’. *Energy and Buildings* 169: 417–29.

Li, Yunfei, Sebastian Schubert, Jürgen P. Kropp, and Diego Rybski. 2020. ‘On the Influence of Density and Morphology on the Urban Heat Island Intensity’. *Nature Communications* 11(1): 2647.

Lin, Brenda B et al. 2021. ‘Integrating Solutions to Adapt Cities for Climate Change’. *The Lancet Planetary Health* 5(7): e479–86.

Lindberg, Fredrik et al. 2018. ‘Urban Multi-Scale Environmental Predictor (UMEP): An Integrated Tool for City-Based Climate Services’. *Environmental Modelling & Software* 99: 70–87.

Loga, Tobias, Britta Stein, and Nikolaus Diefenbach. 2016. ‘TABULA Building Typologies in 20 European Countries—Making Energy-Related Features of Residential Building Stocks Comparable’. *Energy and Buildings* 132: 4–12.

Lomas, Kevin J., and Stephen M. Porritt. 2017. ‘Overheating in Buildings: Lessons from Research’. *Building Research & Information* 45(1–2): 1–18.

Lorenzo, Nieves, Alejandro Díaz-Poso, and Dominic Royé. 2021. ‘Heatwave Intensity on the Iberian Peninsula: Future Climate Projections’. *Atmospheric Research* 258: 105655.

Machard, Anaïs et al. 2020. ‘A Methodology for Assembling Future Weather Files Including Heatwaves for Building Thermal Simulations from the European Coordinated Regional Downscaling Experiment (EURO-CORDEX) Climate Data’. *Energies* 13(13): 3424.

Malet-Damour, Bruno. 2012. ‘Impact of the Climate on the Design of Low-Energy Buildings for Australia and Reunion Island’. In *12th Conference of International Building Performance Simulation Association*, Sydney, 2867–73. hal-00768497.

Manoli, Gabriele et al. 2019. ‘Magnitude of Urban Heat Islands Largely Explained by Climate and Population’. *Nature* 573(7772): 55–60.

Maraun, Douglas. 2016. ‘Bias Correcting Climate Change Simulations - a Critical Review’. *Current Climate Change Reports* 2(4): 211–20.

Martinez, S. et al. 2021. ‘A Practical Approach to the Evaluation of Local Urban Overheating–A Coastal City Case-Study’. *Energy and Buildings* 253: 111522.

Mata, É., A. Sasic Kalagasidis, and F. Johnsson. 2014. ‘Building-Stock Aggregation through Archetype Buildings: France, Germany, Spain and the UK’. *Building and Environment* 81: 270–82.

Mauro, Lea. 2013. ‘Value of Land and Dwellings : Treatment in French National Accounts’. http://www.oecd.org/sdd/na/Presentation-OECD-VA.pdf.

McCarthy, Mark P., Martin J. Best, and Richard A. Betts. 2010. ‘Climate Change in Cities Due to Global Warming and Urban Effects: CLIMATE CHANGE IN CITIES’. *Geophysical Research Letters* 37(9): n/a-n/a.

McCartney, Kathryn J, and J Fergus Nicol. 2002. ‘Developing an Adaptive Control Algorithm for Europe’. *Energy and Buildings* 34(6): 623–35.

Mitchell, Dann, Kai Kornhuber, Chris Huntingford, and Peter Uhe. 2019. ‘The Day the 2003 European Heatwave Record Was Broken’. *The Lancet Planetary Health* 3(7): e290–92.

Mitchell, Rachel, and Sukumar Natarajan. 2019. ‘Overheating Risk in Passivhaus Dwellings’. *Building Services Engineering Research and Technology* 40(4): 446–69.

Moazami, Amin, Vahid M. Nik, Salvatore Carlucci, and Stig Geving. 2019. ‘Impacts of Future Weather Data Typology on Building Energy Performance – Investigating Long-Term Patterns of Climate Change and Extreme Weather Conditions’. *Applied Energy* 238: 696–720.

Mohajerani, Abbas, Jason Bakaric, and Tristan Jeffrey-Bailey. 2017. ‘The Urban Heat Island Effect, Its Causes, and Mitigation, with Reference to the Thermal Properties of Asphalt Concrete’. *Journal of Environmental Management* 197: 522–38.

Monteiro, Claudia Sousa et al. 2017. ‘The Use of Multi-Detail Building Archetypes in Urban Energy Modelling’. *Energy Procedia* 111: 817–25.

Monteiro, Claudia Sousa, Carlos Cerezo, André Pina, and Paulo Ferrão. 2015. ‘A METHOD FOR THE GENERATION OF MULTI-DETAIL BUILDING ARCHETYPE DEFINITIONS: APPLICATION TO THE CITY OF LISBON’. : 6.

Montgomery, Douglas C. 2017. *Design and Analysis of Experiments*. Ninth edition. Hoboken, NJ: John Wiley & Sons, Inc.

Moujalled, Bassam, Valérie Leprince, and Adeline Melois. ‘French Database of Building Airtightness, Statistical Analyses of about 215,000 Measurements: Impacts of Buildings Characteristics and Seasonal Variations’. : 11.

Moura, Maria Cecilia P., Steven J. Smith, and David B. Belzer. 2015. '120 Years of U.S. Residential Housing Stock and Floor Space' ed. Wei-Xing Zhou. *PLOS ONE* 10(8): e0134135.

Nicol, J.F. 1974. 'An Analysis of Some Observations of Thermal Comfort in Roorkee, India and Baghdad, Iraq'. *Annals of Human Biology* 1(4): 411–26.

Nou, Julien, Rémi Chauvin, Stéphane Thil, and Stéphane Grieu. 2016. 'A New Approach to the Real-Time Assessment of the Clear-Sky Direct Normal Irradiance'. *Applied Mathematical Modelling* 40(15–16): 7245–64.

Oke, T. R. 1995. 'The Heat Island of the Urban Boundary Layer: Characteristics, Causes and Effects'. In *Wind Climate in Cities*, eds. Jack E. Cermak, Alan G. Davenport, Erich J. Plate, and Domingos X. Viegas. Dordrecht: Springer Netherlands, 81–107. http://link.springer.com/10.1007/978-94-017-3686-2_5 (November 7, 2021).

Oke, T. R., G. Mills, A. Christen, and J. A. Voogt. 2017. *Urban Climates*. Cambridge: Cambridge University Press. http://ebooks.cambridge.org/ref/id/CBO9781139016476 (November 2, 2021).

Oke, T.R. 1973. 'City Size and the Urban Heat Island'. *Atmospheric Environment (1967)* 7(8): 769–79.

Österbring, Magnus et al. 2016. 'A Differentiated Description of Building-Stocks for a Georeferenced Urban Bottom-up Building-Stock Model'. *Energy and Buildings* 120: 78–84.

Ouzeau, G. et al. 2016. 'Heat Waves Analysis over France in Present and Future Climate: Application of a New Method on the EURO-CORDEX Ensemble'. *Climate Services* 4: 1–12.

Palme, M. et al. 2017. 'From Urban Climate to Energy Consumption. Enhancing Building Performance Simulation by Including the Urban Heat Island Effect'. *Energy and Buildings* 145: 107–20.

Pang, Zhihong, Zheng O'Neill, Yanfei Li, and Fuxin Niu. 2020. 'The Role of Sensitivity Analysis in the Building Performance Analysis: A Critical Review'. *Energy and Buildings* 209: 109659.

Park, Ji Hun et al. 2020. 'Impact of a Passive Retrofit Shading System on Educational Building to Improve Thermal Comfort and Energy Consumption'. *Energy and Buildings* 216: 109930.

Parker, James. 2021. 'The Leeds Urban Heat Island and Its Implications for Energy Use and Thermal Comfort'. *Energy and Buildings* 235: 110636.

Pasichnyi, Oleksii, Jörgen Wallin, and Olga Kordas. 2019. 'Data-Driven Building Archetypes for Urban Building Energy Modelling'. *Energy* 181: 360–77.

Pedregosa, F. et al. 2011. 'Scikit-Learn: Machine Learning in Python'. *Journal of Machine Learning Research* 12: 2825–30.

Perera, A.T.D., Silvia Coccolo, Jean-Louis Scartezzini, and Dasaraden Mauree. 2018. 'Quantifying the Impact of Urban Climate by Extending the Boundaries of Urban Energy System Modeling'. *Applied Energy* 222: 847–60.

Perkins, S. E., L. V. Alexander, and J. R. Nairn. 2012. 'Increasing Frequency, Intensity and Duration of Observed Global Heatwaves and Warm Spells'. *Geophysical Research Letters* 39(20): 2012GL053361.

Petrou, Giorgos et al. 2019. 'The Summer Indoor Temperatures of the English Housing Stock: Exploring the Influence of Dwelling and Household Characteristics'. *Building Services Engineering Research and Technology* 40(4): 492–511.

P.Tootkaboni, Mamak, Ilaria Ballarini, Michele Zinzi, and Vincenzo Corrado. 2021. 'A Comparative Analysis of Different Future Weather Data for Building Energy Performance Simulation'. *Climate* 9(2): 37.

Raei, Ehsan et al. 2018. 'GHWR, a Multi-Method Global Heatwave and Warm-Spell Record and Toolbox'. *Scientific Data* 5(1): 180206.

Ramon, Delphine et al. 2019. 'Future Weather Data for Dynamic Building Energy Simulations: Overview of Available Data and Presentation of Newly Derived Data for Belgium'. In *Energy Sustainability in Built and Urban Environments*, Energy, Environment, and Sustainability, eds. Emilia Motoasca, Avinash Kumar Agarwal, and Hilde Breesch. Singapore: Springer Singapore, 111–38. http://link.springer.com/10.1007/978-981-13-3284-5_6 (May 20, 2021).

Ray, Santanu, Joyita Mukherjee, and Sudipto Mandal. 2015. 'Modelling Nitrogen and Carbon Cycles in Hooghly Estuary along with Adjacent Mangrove Ecosystem'. In *Developments in Environmental Modelling*, Elsevier, 289–320. https://linkinghub.elsevier.com/retrieve/pii/B9780444635365000132 (July 13, 2022).

Rokach, Lior, and Oded Z. Maimon. 2015. *Data Mining with Decision Trees: Theory and Applications*. 2nd edition. New Jersey London Singapore Beijing: World Scientific.

Romani, Zaid, Abdeslam Draoui, and Francis Allard. 2021. 'Metamodeling and Multicriteria Analysis for Sustainable and Passive Residential Building Refurbishment: A Case Study of French Housing Stock'. *Building Simulation*. https://link.springer.com/10.1007/s12273-021-0806-7 (October 8, 2021).

Rosbakh, Sergey, Karl Auerswald, and Peter Poschlod. 2021. 'Rising CO2 Concentrations Reduce Nitrogen Availability in Alpine Grasslands'. *Ecological Indicators* 129: 107990.

Rousseeuw, Peter J. 1987. 'Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis'. *Journal of Computational and Applied Mathematics* 20: 53–65.

Rummukainen, Markku. 2010. 'State-of-the-art with Regional Climate Models'. *WIREs Climate Change* 1(1): 82–96.

Sachindra, D. A., A. W. M. Ng, S. Muthukumaran, and B. J. C. Perera. 2016. 'Impact of Climate Change on Urban Heat Island Effect and Extreme Temperatures: A Case-study'. *Quarterly Journal of the Royal Meteorological Society* 142(694): 172–86.

Salagnac, Jean-Luc. 2007. 'Lessons from the 2003 Heat Wave: A French Perspective'. *Building Research & Information* 35(4): 450–57.

Schoetter, Robert et al. 2019. 'Caractérisation du tissu urbain français pour la modélisation du climat urbain et de son interaction avec la consommation énergétique dans les bâtiments'. *La Météorologie*: 10.

Shalaby, Heidi, and Somaya Aboelnaga. 2017. 'Climate Change Impacts on Urban Planning in the Cities'. *SSRN Electronic Journal*. https://www.ssrn.com/abstract=3162375 (October 13, 2021).

de Sherbinin, Alex et al. 2019. 'Climate Vulnerability Mapping: A Systematic Review and Future Prospects'. *WIREs Climate Change* 10(5). https://onlinelibrary.wiley.com/doi/10.1002/wcc.600 (December 4, 2021).

Shin, Minjae, and Jeff S. Haberl. 2019. 'Thermal Zoning for Building HVAC Design and Energy Simulation: A Literature Review'. *Energy and Buildings* 203: 109429.

Smith, Lillian, Kyle Bernhardt, and Matthew Jezyk. 2011. 'Automated Energy Model Creation for Conceptual Design'. In *2011 Spring Simulation Multi-Conference, SpringSim '11, Boston, MA, USA, April 03-07, 2011. Volume 8: Proceedings of the 2011 Symposium on Simulation for Architecture and Urban Design (SimAUD)*, ed. Ramtin Attar. SCS/ACM, 13–20. http://dl.acm.org/citation.cfm?id=2048538.

Sokol, Julia, Carlos Cerezo Davila, and Christoph F. Reinhart. 2017. 'Validation of a Bayesian-Based Method for Defining Residential Archetypes in Urban Building Energy Models'. *Energy and Buildings* 134: 11–24.

Steadman, R. G. 1979. 'The Assessment of Sultriness. Part I: A Temperature-Humidity Index Based on Human Physiology and Clothing Science'. *Journal of Applied Meteorology (1962-1982)* 18(7): 861–73.

Stefanon, Marc, Fabio D'Andrea, and Philippe Drobinski. 2012. 'Heatwave Classification over Europe and the Mediterranean Region'. *Environmental Research Letters* 7(1): 014023.

Stewart, I. D., and T. R. Oke. 2012a. 'Local Climate Zones for Urban Temperature Studies'. *Bulletin of the American Meteorological Society* 93(12): 1879–1900.

———. 2012b. 'Local Climate Zones for Urban Temperature Studies'. *Bulletin of the American Meteorological Society* 93(12): 1879–1900.

Stull, Roland. 2011. 'Wet-Bulb Temperature from Relative Humidity and Air Temperature'. *Journal of Applied Meteorology and Climatology* 50(11): 2267–69.

TABULA Project Team. 2012. *Application of Building Typologies for Modellingthe Energy Balance of the Residential Building Stock*. Darmstadt: IWU.

'TABULA_FinalReport.Pdf'.

Tan, Jianguo et al. 2010. 'The Urban Heat Island and Its Impact on Heat Waves and Human Health in Shanghai'. *International Journal of Biometeorology* 54(1): 75–84.

Tannier, Cécile, and Isabelle Thomas. 2013. 'Defining and Characterizing Urban Boundaries: A Fractal Analysis of Theoretical Cities and Belgian Cities'. *Computers, Environment and Urban Systems* 41: 234–48.

Tartarini, Federico, and Stefano Schiavon. 2020. 'Pythermalcomfort: A Python Package for Thermal Comfort Research'. *SoftwareX* 12: 100578.

'The guide in Urban planning / Le guide du PLU (in French)'. 2020. https://www.correze.gouv.fr/Politiques-publiques/Amenagement-du-territoire-construction-et-logement/Urbanisme/le-Plan-Local-d-Urbanisme-P.L.U.

Thiis, Thomas K. et al. 2017. 'Monitoring and Simulation of Diurnal Surface Conditions of a Wooden Façade'. *Procedia Environmental Sciences* 38: 331–39.

Tm59 Design Methodology for the Assessment of Overheating Risk in Homes. 2017. Place of publication not identified: CHARTERED INST OF BUILDI.

Tobi, Hilde, and Jarl K. Kampen. 2018. 'Research Design: The Methodology for Interdisciplinary Research Framework'. *Quality & Quantity* 52(3): 1209–25.

Tornay, Nathalie et al. 2017. 'GENIUS: A Methodology to Define a Detailed Description of Buildings for Urban Climate and Building Energy Consumption Simulations'. *Urban Climate* 20: 75–93.

Tzavali, Anna, John P Paravantis, and Giouli Mihalakakou. 2015. 'URBAN HEAT ISLAND INTENSITY: A LITERATURE REVIEW'. *Fresenius Environmental Bulletin* 24(12): 21.

Vujovic, Svetlana et al. 2021. 'Urban Heat Island: Causes, Consequences, and Mitigation Measures with Emphasis on Reflective and Permeable Pavements'. *CivilEng* 2(2): 459–84.

W. Schlegel, Robert, and Albertus J. Smit. 2018. 'HeatwaveR: A Central Algorithm for the Detection of Heatwaves and Cold-Spells'. *Journal of Open Source Software* 3(27): 821.

Wagner, Vérène. 2018. 'Evolution of heatwaves and associated mortality in France, 2004-2014/Évolution des vagues de chaleur et de la mortalité associée en France, 2004-2014 (in French)'. : 6.

Wang, Yuanyuan, Zhongyang Guo, and Ji Han. 2021. 'The Relationship between Urban Heat Island and Air Pollutants and Them with Influencing Factors in the Yangtze River Delta, China'. *Ecological Indicators* 129: 107976.

Wilby, Robert L. 2008. 'Constructing Climate Change Scenarios of Urban Heat Island Intensity and Air Quality'. *Environment and Planning B: Planning and Design* 35(5): 902–19.

WMO. 2020. *State of the Global Climate 2020*. World Meteorological Organization, 2021. https://library.wmo.int/doc_num.php?explnum_id=10618.

Wong, Nyuk Hien et al. 2021. 'An Integrated Multiscale Urban Microclimate Model for the Urban Thermal Environment'. *Urban Climate* 35: 100730.

Yang, Jun et al. 2019. ‘Heatwave and Mortality in 31 Major Chinese Cities: Definition, Vulnerability and Implications’. *Science of The Total Environment* 649: 695–702.

Yang, Junyan et al. 2020. ‘Impacts of Urban Form on Thermal Environment Near the Surface Region at Pedestrian Height: A Case Study Based on High-Density Built-Up Areas of Nanjing City in China’. *Sustainability* 12(5): 1737.

Yi, Hwang. 2016. ‘User-Driven Automation for Optimal Thermal-Zone Layout during Space Programming Phases’. *Architectural Science Review* 59(4): 279–306.

Zhang, Mingxi et al. 2022. ‘Heat Wave Tracker: A Multi-Method, Multi-Source Heat Wave Measurement Toolkit Based on Google Earth Engine’. *Environmental Modelling & Software* 147: 105255.

Zhang, Ruonan et al. 2020. ‘Increased European Heat Waves in Recent Decades in Response to Shrinking Arctic Sea Ice and Eurasian Snow Cover’. *npj Climate and Atmospheric Science* 3(1): 7.

Appendices

Appendix 2-1

- Measuring length of shared wall between two polygons using Geopandas.
- Measuring area of shared wall between two joint buildings.

In [1]:
```
import geopandas as gpd
import numpy as np
import pandas as pd
from shapely.geometry import polygon
from shapely.geometry import MultiLineString
import itertools
from geopandas import GeoDataFrame, overlay
```

In [3]:
```
# Read shapefile
df=gpd.read_file(r'D:\Donnees Bat. Cerema\BDTOPO_CEREMA_2\BDTOPO\Buildings in BDTOPO.shp')
```

Identifying the indices of joint neighbors to each polygon

In [134…
```
df = df.reset_index()
```

In [136…
```
df=df.rename(columns={"index":"UID"})
```

In [137…
```
df.UID = df.UID.astype(str)
```

In [139…
```
df["NEIGHBORS"] = None

for index, row in df.iterrows():

    neighbors = df[~df.geometry.disjoint(row.geometry)].UID.tolist()
    neighbors = [ X for X in neighbors if row.UID != X ]

    df.at[index, "NEIGHBORS"] = ", ".join(neighbors)

df.head(1)
```

Out[139…

	UID	ID	USAGE1	NB_LOGTS	NB_ETAGES	MAT_MURS	MAT_TOITS	HAUTEUR	Z_MIN_SOL	Z_MIN_TOIT	Z_MA
0	0	BATIMENT0000000303978565	IndiffÃ? Â©renciÃ? Â©	NaN	NaN	None	None	2.3	12.2	NaN	

In [140…
```
# Save as new shapefile to be openned in QGIS
df.to_file(driver = 'ESRI Shapefile', filename= r'D:\Donnees Bat. Cerema\BDTOPO_CEREMA_2\BDTOPO\BDTOPO neighbors.
```

In [ ]:
```
# Saving the file in excel to verify the output and for each joint neighbor add a new column
```

In [198…
```
df.to_excel(r'D:\Donnees Bat. Cerema\BDTOPO_CEREMA_2\BDTOPO\BDTOPO neighbors.xlsx', index = 0)
```

Calculating area of shared wall between target polygon and joint neighbors

In [3]:
```
# Importing the excel sheet that contains the parsed indix numbers in columns
df2=pd.read_excel(r'D:\Donnees Bat. Cerema\BDTOPO_CEREMA_2\BDTOPO\BDTOPO neighbors.xlsx')
```

In [31]:
```
df2=df2.drop(['HAUTEUR','Unique_ID','area1','T_perimi','Count','geometry','NEIGHBORS'],axis=1)
df2.head(1)
```

Out[31]:

	UID	neighbor1	n_height1	neighbor2	n_height2	neighbor3	n_height3	neighbor4	n_height4	neighbor5	...	neighbor14	neighbor15	neighl
0	0	9.0	3.2	10.0	2.6	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	

1 rows × 30 columns

In [34]:
```
# Index of each neighbor and corresponding height of it
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 79789 entries, 0 to 79788
Data columns (total 30 columns):
UID           79789 non-null int64
neighbor1     64152 non-null float64
n_height1     64152 non-null float64
neighbor2     35480 non-null float64
n_height2     35480 non-null float64
neighbor3     12370 non-null float64
n_height3     12370 non-null float64
neighbor4     4074 non-null float64
n_height4     4074 non-null float64
neighbor5     1379 non-null float64
n_height5     1379 non-null float64
neighbor6     538 non-null float64
n_height6     0 non-null float64
neighbor7     235 non-null float64
neighbor8     113 non-null float64
neighbor9     64 non-null float64
neighbor10    36 non-null float64
neighbor11    22 non-null float64
neighbor12    16 non-null float64
neighbor13    12 non-null float64
neighbor14    9 non-null float64
neighbor15    5 non-null float64
neighbor16    5 non-null float64
neighbor17    3 non-null float64
neighbor18    2 non-null float64
neighbor19    2 non-null float64
neighbor20    2 non-null float64
neighbor21    2 non-null float64
neighbor22    2 non-null float64
neighbor23    1 non-null float64
dtypes: float64(29), int64(1)
memory usage: 18.3 MB
```

In [130…

```
#
df2.to_excel(r'D:\Donnees Bat. Cerema\BDTOPO_CEREMA_2\BDTOPO\BDTOPO neighbors.xlsx', index = 0)
```

Reprojected to orginal/french coordinate system to measure distances in meters

In [164…

```
# importing shapefile
df1=gpd.read_file(r'D:\Donnees Bat. Cerema\BDTOPO_CEREMA_2\BDTOPO\BDTOPO neighbors2.shp')
df1.head(1)
```

Out[164…

	UID	ID	USAGE1	NB_LOGTS	NB_ETAGES	MAT_MURS	MAT_TOITS	HAUTEUR	Z_MIN_SOL	Z_MIN_TOIT	Z_MA
0	0	BATIMENT0000000303978565	IndiffÃ? Â©renciÃ? Â©	NaN	NaN	None	None	2.3	12.2	NaN	

In [165…

```
df1['UID']=df1['UID'].astype(str).astype(float)
```

In [166…

```
# merging df1 to df2
df1=pd.merge(df1,df2, on=["UID"], how="right")
df1.head(2)
```

Out[166…

	UID	ID	USAGE1	NB_LOGTS	NB_ETAGES	MAT_MURS	MAT_TOITS	HAUTEUR	Z_MIN_SOL	Z_MIN_TOIT	...	n
0	0.0	BATIMENT0000000303978565	IndiffÃ? Â©renciÃ? Â©	NaN	NaN	None	None	2.3	12.2	NaN	...	
1	1.0	BATIMENT0000000303977720	IndiffÃ? Â©renciÃ? Â©	NaN	NaN	None	None	2.6	10.8	13.4	...	

2 rows × 49 columns

In [168…

```
print(df1.crs)
```

```
PROJCS["NTF (Paris) / Lambert Centre France",GEOGCS["NTF (Paris)",DATUM["Nouvelle_Triangulation_Francaise_Paris",
SPHEROID["Clarke 1880 (IGN)",6378249.2,293.466021293627,AUTHORITY["EPSG","7011"]],AUTHORITY["EPSG","6275"]],PRIME
M["Paris",2.10350625299998],UNIT["Grad",0.0157079632679489]],PROJECTION["Lambert_Conformal_Conic_1SP"],PARAMETER[
```

```
"latitude_of_origin",52],PARAMETER["central_meridian",0],PARAMETER["scale_factor",0.99987742],PARAMETER["false_easting",600000],PARAMETER["false_northing",200000],UNIT["metre",1,AUTHORITY["EPSG","9001"]],AXIS["Easting",EAST],AXIS["Northing",NORTH]]
```

In [170…

```
df1['UID']=df1['UID'].astype(np.int64)
df1['neighbor1']=df1['neighbor1'].fillna(-1).astype(np.int64)
```

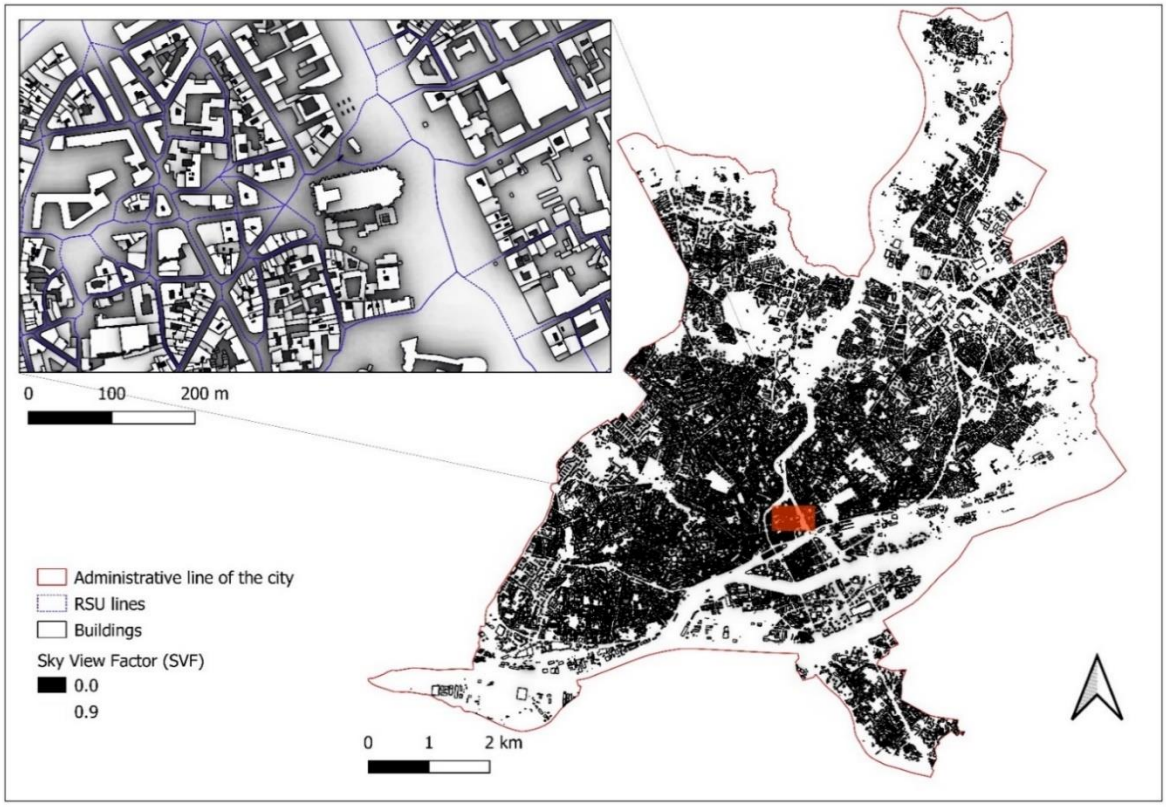
In [171…

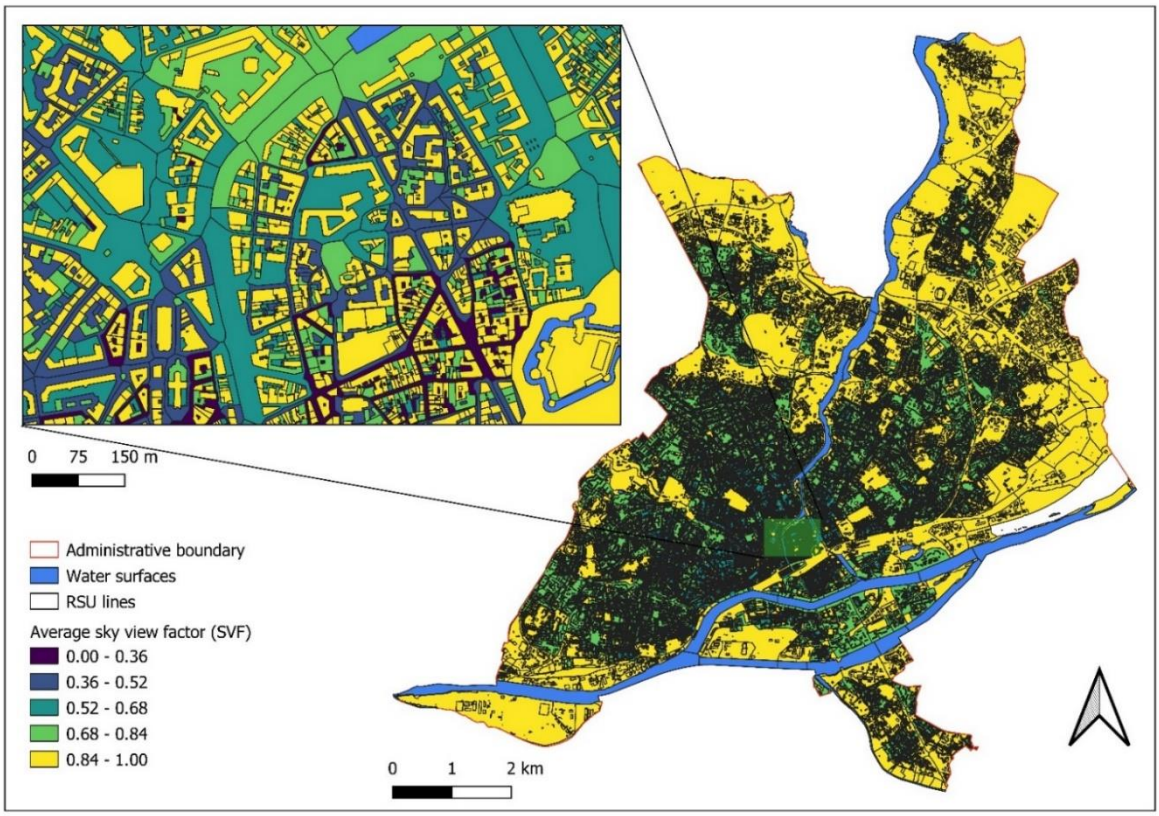
```
df1['Unique_ID']=df1['Unique_ID'].astype(np.int64)
df1['neighbor2']=df1['neighbor2'].fillna(-1).astype(np.int64)
df1['neighbor3']=df1['neighbor3'].fillna(-1).astype(np.int64)
df1['neighbor4']=df1['neighbor4'].fillna(-1).astype(np.int64)
df1['neighbor5']=df1['neighbor5'].fillna(-1).astype(np.int64)
df1['neighbor6']=df1['neighbor6'].fillna(-1).astype(np.int64)
df1['neighbor7']=df1['neighbor7'].fillna(-1).astype(np.int64)
df1['neighbor8']=df1['neighbor8'].fillna(-1).astype(np.int64)
df1['neighbor9']=df1['neighbor9'].fillna(-1).astype(np.int64)
df1['neighbor10']=df1['neighbor10'].fillna(-1).astype(np.int64)
df1['neighbor11']=df1['neighbor11'].fillna(-1).astype(np.int64)
df1['neighbor12']=df1['neighbor12'].fillna(-1).astype(np.int64)
df1['neighbor13']=df1['neighbor13'].fillna(-1).astype(np.int64)
df1['neighbor14']=df1['neighbor14'].fillna(-1).astype(np.int64)
df1['neighbor15']=df1['neighbor15'].fillna(-1).astype(np.int64)
df1['neighbor16']=df1['neighbor16'].fillna(-1).astype(np.int64)
df1['neighbor17']=df1['neighbor17'].fillna(-1).astype(np.int64)
df1['neighbor18']=df1['neighbor18'].fillna(-1).astype(np.int64)
df1['neighbor19']=df1['neighbor19'].fillna(-1).astype(np.int64)
df1['neighbor20']=df1['neighbor20'].fillna(-1).astype(np.int64)
df1['neighbor21']=df1['neighbor21'].fillna(-1).astype(np.int64)
df1['neighbor22']=df1['neighbor22'].fillna(-1).astype(np.int64)
df1['neighbor23']=df1['neighbor23'].fillna(-1).astype(np.int64)
```

In [173…

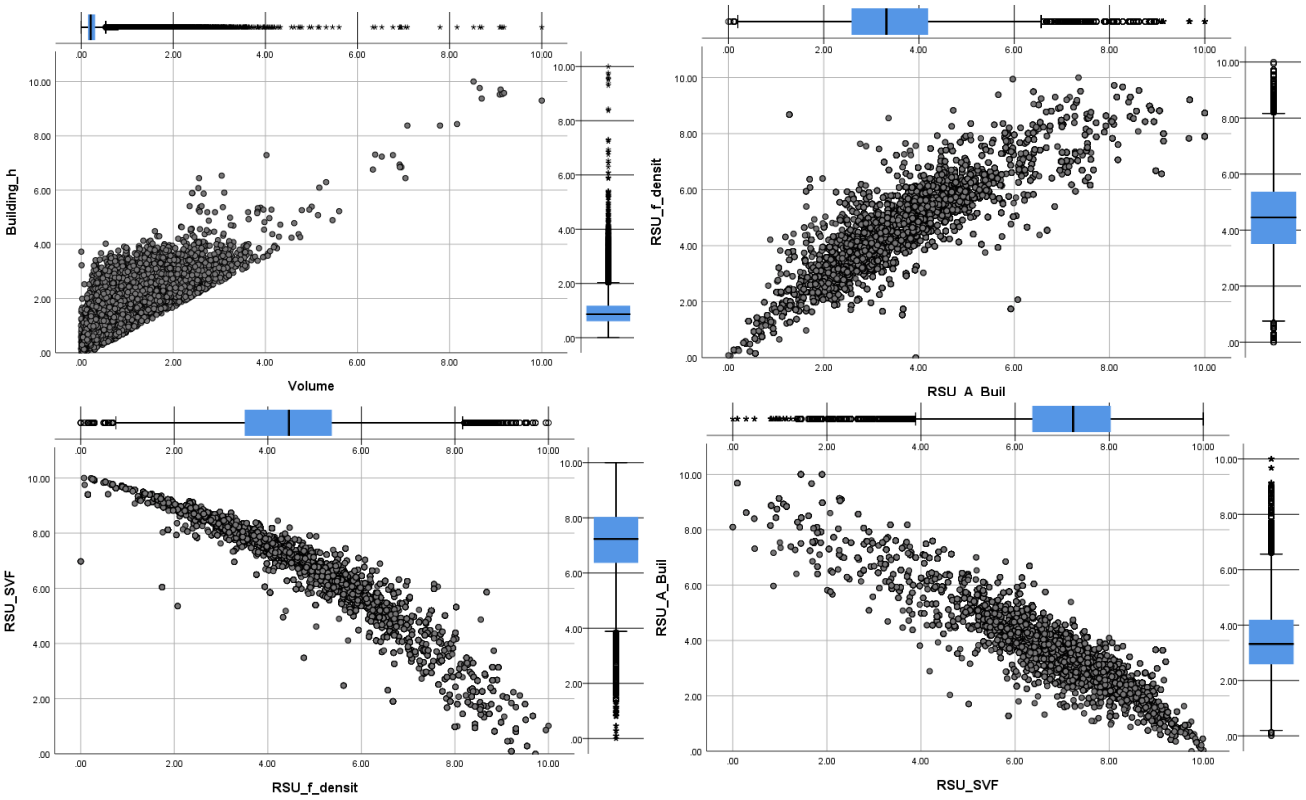
```
# Calculate length of common walls
N1_length=[]
N1_height=[]
N1_area=[]
for i, row in df1.iterrows():

    if df1.neighbor1[i] ==-1:

        X=0
        Y=0
        Z=0

    else:

        X=df1.geometry[i].intersection(df1.geometry[df1.neighbor1[i]]).length
        Y=df1.HAUTEUR[df1.neighbor1[i]]
        Z=X*min(Y,df1.HAUTEUR[i])

    N1_length.append(X)
    N1_height.append(Y)
    N1_area.append(Z)

df1["N1_length"] = N1_length
df1["N1_height"] = N1_height
df1["N1_area"] = N1_area
```

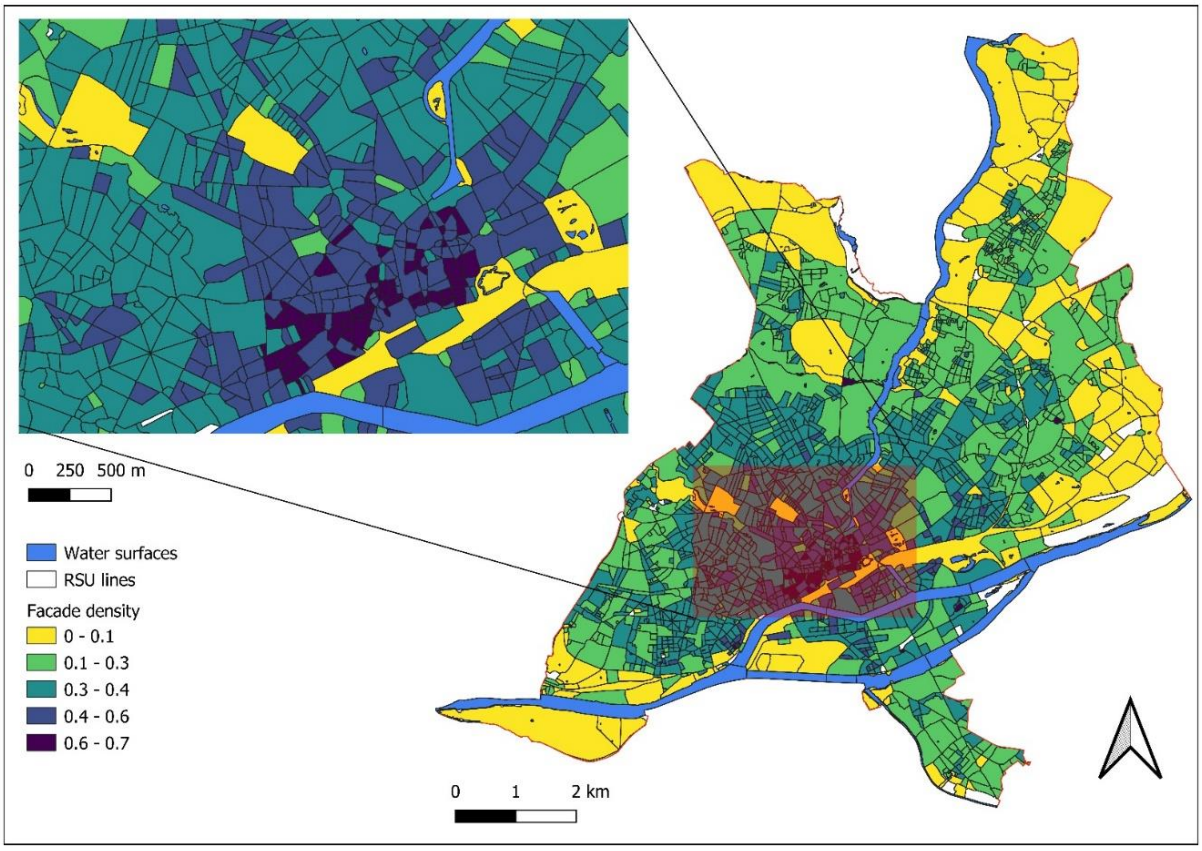
Appendix 2-2: Sky view factor map of Nantes city





Appendix 2-3: façade density map





Appendix 2-4:

- Data pre-treatment for clustering
- Cluster analysis with Scikit-learn

Clustering with urban morphological parameters

Pre-processing : data cleaning

In [1]:
```
import geopandas as gpd
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set()
import plotly.express as px
import math
```

In [2]:
```
%config Completer.use_jedi = False
```

In [3]:
```
df=gpd.read_file(r'D:\Donnees Bat. Cerema\BDTOPO_CEREMA_2\Nantes database\Bld_RSU_combined.shp')
df.head(1)
```

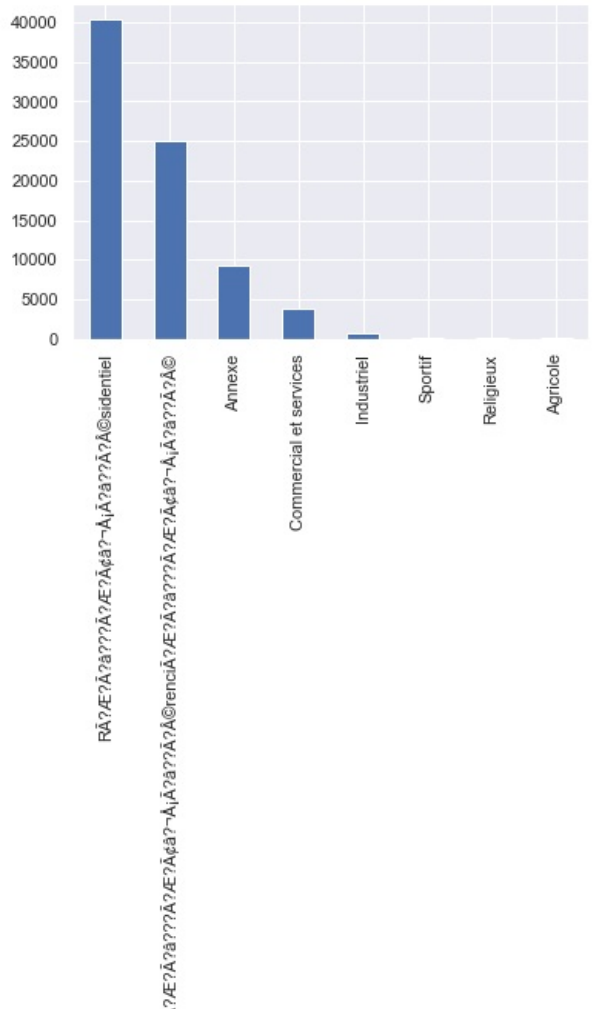
Out[3]:

	index	UID	ID	USAGE1	Building_h	Unique_ID	T_perimi	NEIGHBORS	indices	Total_area	...	W_Water_ar
0	51664	51664	BATIMENT0000000302944746	RÃ?Æ?Ã?â???Ã?Æ?Ã¢â?¬Å¡Ã?â??Ã?Â©sidentiel	7.6	51665	35	51664, 51666	51663, 51665	344.774418	...	0.0

1 rows × 41 columns

In [8]:
```
# Frequency plot of buildings in BDTOPO database
fig, ax = plt.subplots()
df['USAGE1'].value_counts().plot(ax=ax, kind='bar')
```

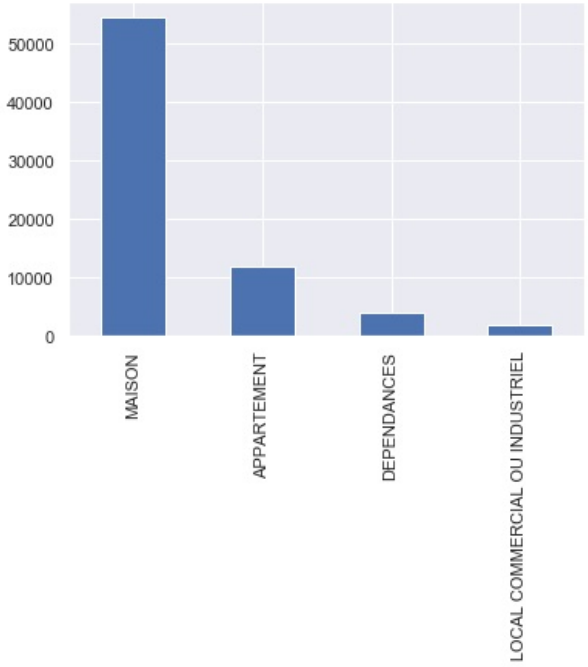
Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1d32a04acd0>
```



IndiffÃ

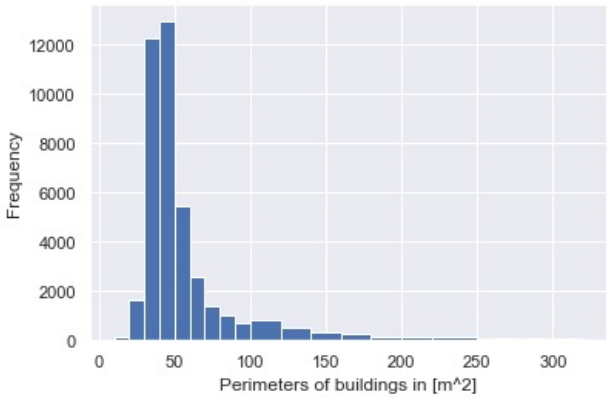
In [9]:
```
# Frequency plot of buildings in BDTOPO database
fig, ax = plt.subplots()
df['Cerema_dte'].value_counts().plot(ax=ax, kind='bar')
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x1d3447499a0>



In [10]:
```
# removing buildings that are NOT residential (RÃƒÆ'Ã†â€™?ÃƒÆ'Ã¢â‚¬Å¡Ã‚Â©sidentiel)
df = df.drop(df[df['USAGE1']!='RÃ?Æ?Ã?â???Ã?Æ?Ã¢â?¬Å¡Ã?â??Ã?Â©sidentiel'].index) #
```

In [12]:
```
# histogram to analyse individual building parameters and thier range
bin_list = [10,20,20,30,40,50,60,70,80,90,100,120,140,160,180,200,220,250,280,300,320]
plt.hist(df.T_perimi, bins=bin_list) #bins_list
plt.xlabel('Perimeters of buildings in [m^2]')
plt.ylabel('Frequency')
plt.rcParams["figure.figsize"]=7,4
plt.rcParams["figure.facecolor"] = 'w'
```



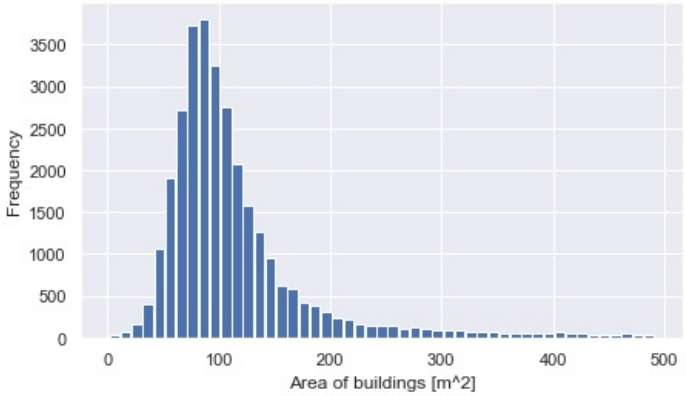
In [14]:
```
# Drop if any row in the columns Cerema_jan (year of construction) and Cerema_sto (habitable area) are empty
df = df.dropna(subset=['Cerema_jan', 'Cerema_sto'])
# Transforming Cerema_jan from and object column to int in order to plot frequency plot
df['year_con'] = df['Cerema_jan'].astype(str).astype(np.int64)
```

In [15]:
```
# Remove year of construction of it is less than 1800
df = df.drop(df[df['year_con'] < 1800].index) #
df = df.drop(df[df['A_building'] > 500].index) # area larger than 500 m2 is an outlier
```

```
df = df.drop(df[df['W_f_densit'] < 0].index) # facade density smaller the zero indicates an error
df = df.drop(df[df['Net_compac'] > 8].index) # net compacity above 8 and below 1 was determined to be outliers
df = df.drop(df[df['FreeAreaRa'] > 1].index) #  outliers
```

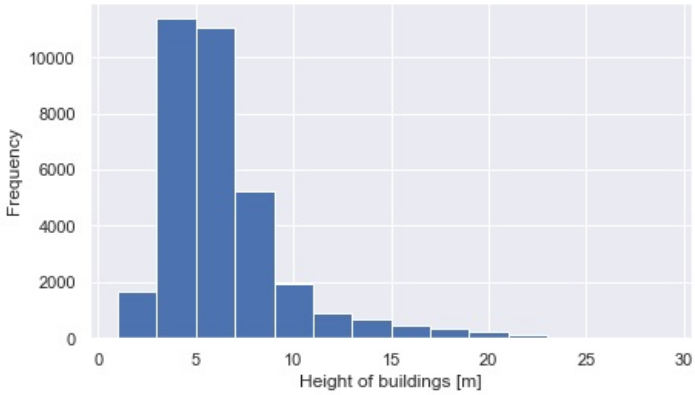
In [17]:

```
# histogram to analyse individual building parameters and thier range
numbers = range(0,500)
bin_list = [number for number in numbers if (number %10 in (1,2)) ]
plt.hist(df.A_building, bins=bin_list) #bins_list
plt.xlabel('Area of buildings [m^2]')
plt.ylabel('Frequency')
plt.rcParams["figure.figsize"]=7,4
plt.rcParams["figure.facecolor"] = 'w'
```



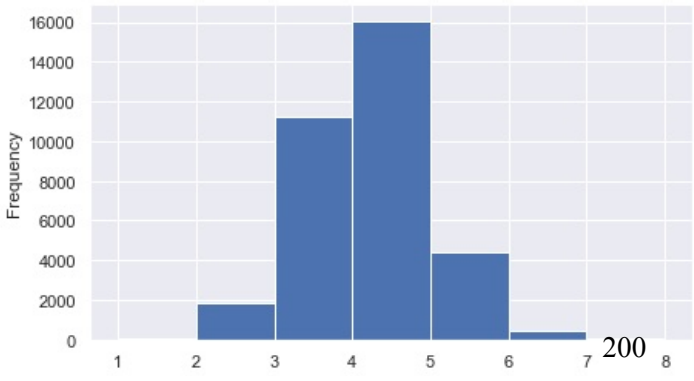
In [18]:

```
# histogram heights
numbers = range(0,30)
bin_list = [number for number in numbers if (number %2 in (1,2)) ]
plt.hist(df.Building_h, bins=bin_list) #bins_list
plt.xlabel('Height of buildings [m]')
plt.ylabel('Frequency')
plt.rcParams["figure.figsize"]=7,4
plt.rcParams["figure.facecolor"] = 'w'
```



In [19]:

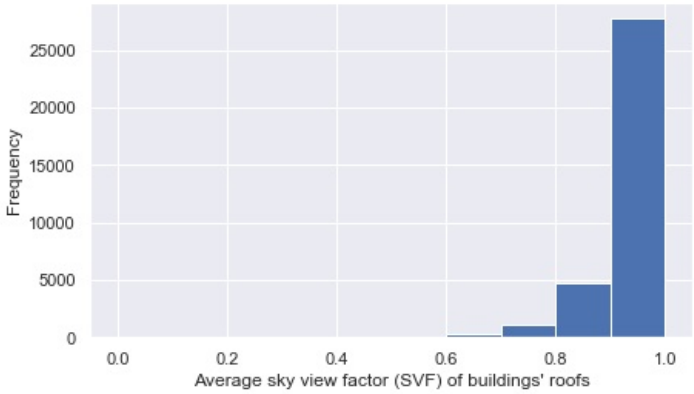
```
# histogram to analyse individual building parameters and thier range
bin_list = [1,2,3,4,5,6,7,8]
plt.hist(df.Net_compac, bins=bin_list) #bins_list
plt.xlabel('Net compacity of builings [free exterior area/volume^2/3]')
plt.ylabel('Frequency')
plt.rcParams["figure.figsize"]=7,4
plt.rcParams["figure.facecolor"] = 'w'
```



Net compacity of builings [free exterior area/volume^2/3]

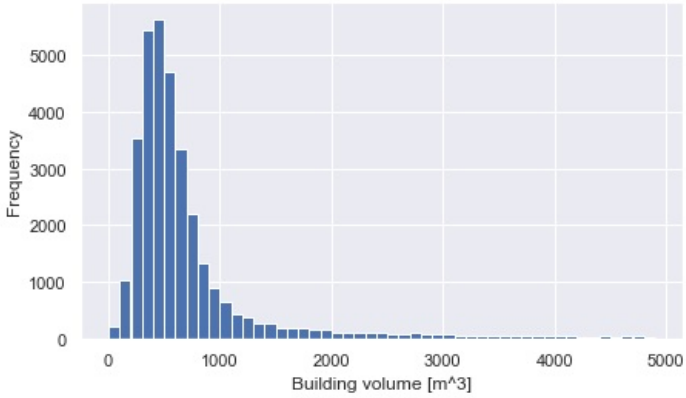
In [20]:

```
# histogram to analyse individual building parameters and thier range
numbers = range(0,30)
bin_list = [0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]
plt.hist(df.SVF_mean, bins=bin_list) #bins_list
plt.xlabel("Average sky view factor (SVF) of buildings' roofs")
plt.ylabel('Frequency')
plt.rcParams["figure.figsize"]=7,4
plt.rcParams["figure.facecolor"] = 'w'
```



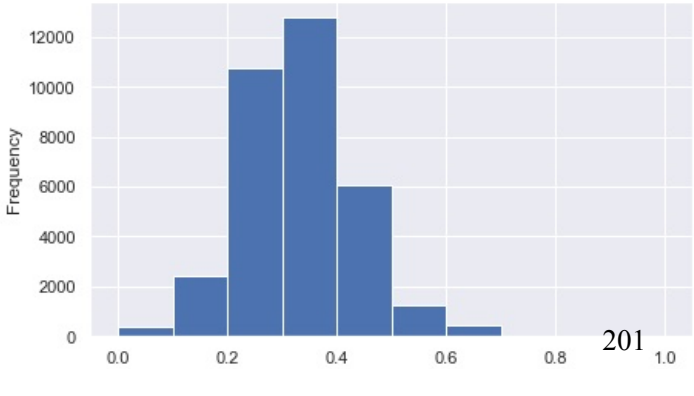
In [21]:

```
# Volume
numbers = range(0,5000)
bin_list = [number for number in numbers if (number %100 in (1,2)) ]
plt.hist(df.Volume, bins=bin_list) #bins_list
plt.xlabel('Building volume [m^3]')
plt.ylabel('Frequency')
plt.rcParams["figure.figsize"]=7,4
plt.rcParams["figure.facecolor"] = 'w'
```

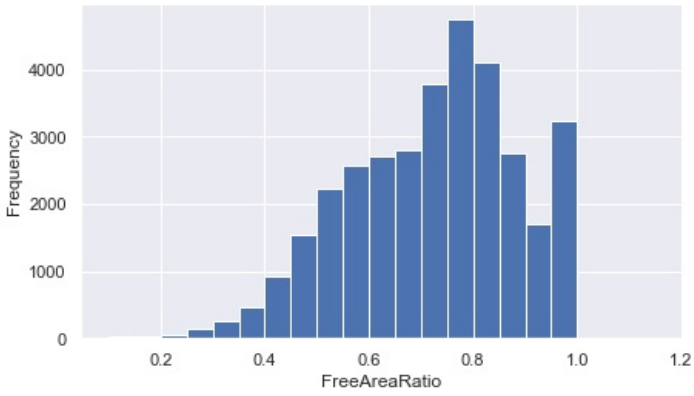


In [22]:

```
# facade density
numbers = range(0,1)
bin_list = [0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]
plt.hist(df.W_f_densit, bins=bin_list) #bins_list
plt.xlabel("Facade density of RSUs")
plt.ylabel('Frequency')
plt.rcParams["figure.figsize"]=7,4
plt.rcParams["figure.facecolor"] = 'w'
```



In [23]:
```
# histogram to analyse individual building parameters and thier range
#numbers = range(0,1.1)
bin_list = [0.1,0.15,0.2,0.25,0.3,0.35,0.4,0.45,0.5,0.55,0.6,0.65,0.7,0.75,0.8,0.85,0.9,0.95,1.0,1.05,1.1,1.15]
plt.hist(df.FreeAreaRa, bins=bin_list) #bins_list
plt.xlabel('FreeAreaRatio')
plt.ylabel('Frequency')
plt.rcParams["figure.figsize"]=7,4
plt.rcParams["figure.facecolor"] = 'w'
```



In [24]:
```
# Rearranging columns
colmn = df.columns.tolist()
print (colmn)
```

```
['index', 'UID', 'ID', 'USAGE1', 'Building_h', 'Unique_ID', 'T_perimi', 'NEIGHBORS', 'indices', 'Total_area', 'To
tal_free', 'NumNeigh', 'A_building', 'A_free_ver', 'Volume', 'Net_compac', 'SVF_mean', 'SVF_min', 'SVF_max', 'dis
tance', 'Cerema_npi', 'Cerema_dte', 'Cerema_jan', 'Cerema_nbe', 'Cerema_sto', 'build_dens', 'RSU_ID', 'W_A_Buildi
', 'W_A_Buil_1', 'W_Vegetati', 'W_Vegeta_1', 'W_Water_ar', 'W_Water_pc', 'W_A_free_v', 'W_A_free_1', 'W_f_densit'
, 'un_SVF_mea', 'un_SVF_min', 'un_SVF_max', 'FreeAreaRa', 'geometry', 'year_con']
```

In [25]:
```
# Column names to string
df.columns = df.columns.map(str)
```

In [26]:
```
# Rearranging columns
new_columns = ['UID','ID', 'USAGE1','Unique_ID', 'T_perimi', 'NEIGHBORS', 'indices', 'Total_area', 'Total_free',
df = df[new_columns]
df.head(2)
```

Out[26]:

	UID	ID	USAGE1	Unique_ID	T_perimi	NEIGHBORS	indices	Total_area	Total_free	NumNeigh	...	Volu
0	51664	BATIMENT0000000302944746	RÃ?Æ?Ã?â???Ã?Æ?Ãçâ?¬Å¡Ã?â??Ã?Â©sidentiel	51665	35	51664, 51666	51663, 51665	344.774418	219.099459	2	...	587.3003
11	51621	BATIMENT0000000302926068	RÃ?Æ?Ã?â???Ã?Æ?Ãçâ?¬Å¡Ã?â??Ã?Â©sidentiel	51622	39	51557, 51621	51556, 51620	319.118976	232.048138	2	...	513.0170

2 rows × 39 columns

In [27]:
```
df.info()
```

```
<class 'geopandas.geodataframe.GeoDataFrame'>
Int64Index: 34053 entries, 0 to 79476
Data columns (total 39 columns):
UID           34053 non-null int64
ID            34053 non-null object
USAGE1        34053 non-null object
Unique_ID     34053 non-null int64
T_perimi      34053 non-null int64
```

```
NEIGHBORS     31602 non-null object
indices       31602 non-null object
Total_area    34053 non-null float64
Total_free    34053 non-null float64
NumNeigh      34053 non-null int64
A_building    34053 non-null float64
A_free_ver    34053 non-null float64
Cerema_npi    34053 non-null float64
Cerema_dte    34053 non-null object
Cerema_jan    34053 non-null object
Cerema_nbe    34053 non-null float64
Cerema_sto    34053 non-null float64
build_dens    34053 non-null float64
RSU_ID        34053 non-null float64
W_A_Buildi    34053 non-null float64
W_Vegetati    34053 non-null float64
W_Water_ar    34053 non-null float64
W_A_free_v    34053 non-null float64
W_A_free_1    34053 non-null float64
un_SVF_min    34053 non-null float64
un_SVF_max    34053 non-null float64
un_SVF_mea    34053 non-null float64
W_A_Buil_1    34053 non-null float64
Building_h    34053 non-null float64
Volume        34053 non-null float64
Net_compac    34053 non-null float64
SVF_mean      34053 non-null float64
distance      34053 non-null float64
year_con      34053 non-null int64
W_Vegeta_1    34053 non-null float64
W_Water_pc    34053 non-null float64
W_f_densit    34053 non-null float64
FreeAreaRa    34053 non-null float64
geometry      34053 non-null geometry
dtypes: float64(27), geometry(1), int64(5), object(6)
memory usage: 10.4+ MB
```

Transforming data for clustering

In [28]:

```
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import seaborn as sns
sns.set()
# Normalizing data
from sklearn import preprocessing

# Hierarchical clustering
from sklearn.cluster import AgglomerativeClustering
from sklearn.mixture import GaussianMixture
import scipy.cluster.hierarchy as sch

from sklearn import metrics
from sklearn.metrics import pairwise_distances
from sklearn.metrics import davies_bouldin_score
```

In [29]:

```
from pandas import DataFrame
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
```

In [30]:

```
# Reindex data to save the indices and export with same numbers into QGIS and visualize
df = df.reset_index()
```

In [32]:

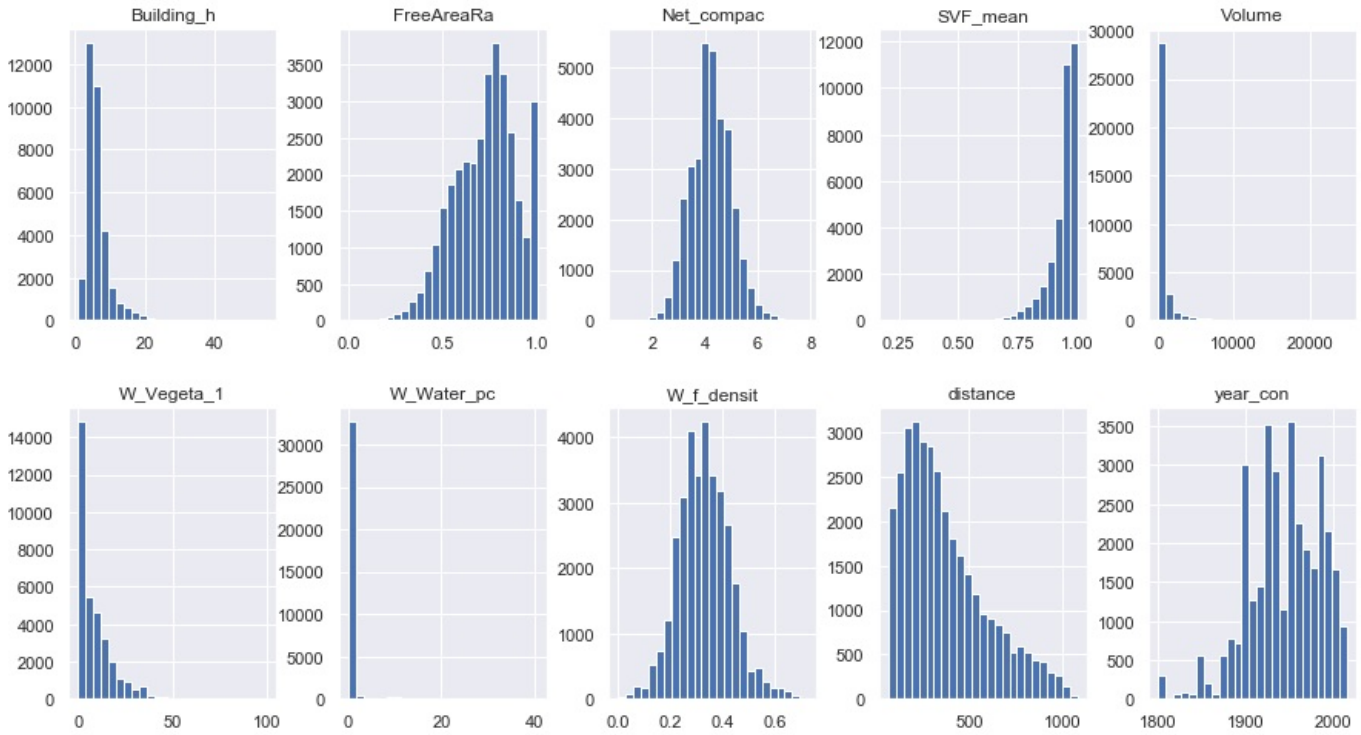
```
data = df.iloc[:, 29:39]
data.head(3)
```

Out[32]:

	Building_h	Volume	Net_compac	SVF_mean	distance	year_con	W_Vegeta_1	W_Water_pc	W_f_densit	FreeAreaRa
0	7.6	587.300363	3.124169	0.941814	480.307457	1972	0.000000	0.0	0.467	0.533170
1	5.9	513.017003	3.620959	0.985431	492.025716	1989	9.322614	0.0	0.302	0.630578
2	5.3	675.622273	4.647524	0.955571	482.764956	1989	9.322614	0.0	0.302	0.804912

In [33]:

In [33]:
```
# Checking distribution of parameters on original data
data.hist(figsize=(15,8),layout=(2,5),bins=25)
pyplot.show()
```



In [34]:
```
from numpy import exp
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import PowerTransformer
```

Yeo-johnson Power Transformation

In [35]:
```
data.columns
```

Out[35]:
```
Index(['Building_h', 'Volume', 'Net_compac', 'SVF_mean', 'distance',
       'year_con', 'W_Vegeta_1', 'W_Water_pc', 'W_f_densit', 'FreeAreaRa'],
      dtype='object')
```

In [36]:
```
pt = PowerTransformer(method='yeo-johnson',standardize=False)
columns = ['Building_h', 'Volume', 'SVF_mean', 'distance',
       'W_Vegeta_1', 'W_Water_pc', 'FreeAreaRa']
mat = pt.fit_transform(data[columns])
mat[:5]
```

Out[36]:
```
array([[ 1.25019484e+00,  3.44519242e+00,  2.01203769e+06,
         1.08403174e+01,  0.00000000e+00, -0.00000000e+00,
         7.59915068e-01],
       [ 1.17999267e+00,  3.41107260e+00,  3.65137778e+06,
         1.09082620e+01,  2.53852940e+00, -0.00000000e+00,
         9.51745982e-01],
       [ 1.14837012e+00,  3.47950609e+00,  2.43159457e+06,
         1.08546796e+01,  2.53852940e+00, -0.00000000e+00,
         1.33959043e+00],
       [ 1.10495865e+00,  3.21039386e+00,  1.78281585e+06,
         1.09489900e+01,  2.53852940e+00, -0.00000000e+00,
         1.35076784e+00],
       [ 1.09806123e+00,  3.16327417e+00,  1.37277633e+06,
         1.09808407e+01,  2.53852940e+00, -0.00000000e+00,
         1.13665356e+00]])
```

In [37]:
```
# Of the given parameters Building heigth, volume, SVF , Water ratio, Vegetation,and distance are
T_columns = [f'T_{c}' for c in columns]
T_columns
```

Out[37]:
```
['T_Building_h',
```

```
 'T_Volume',
 'T_SVF_mean',
 'T_distance',
 'T_W_Vegeta_1',
 'T_W_Water_pc',
 'T_FreeAreaRa']
```

In [38]:
```
# Checking distribution of parameters on original data
data = pd.concat([data,pd.DataFrame(mat,columns=T_columns)],axis='columns')
data.head(4)
```

Out[38]:

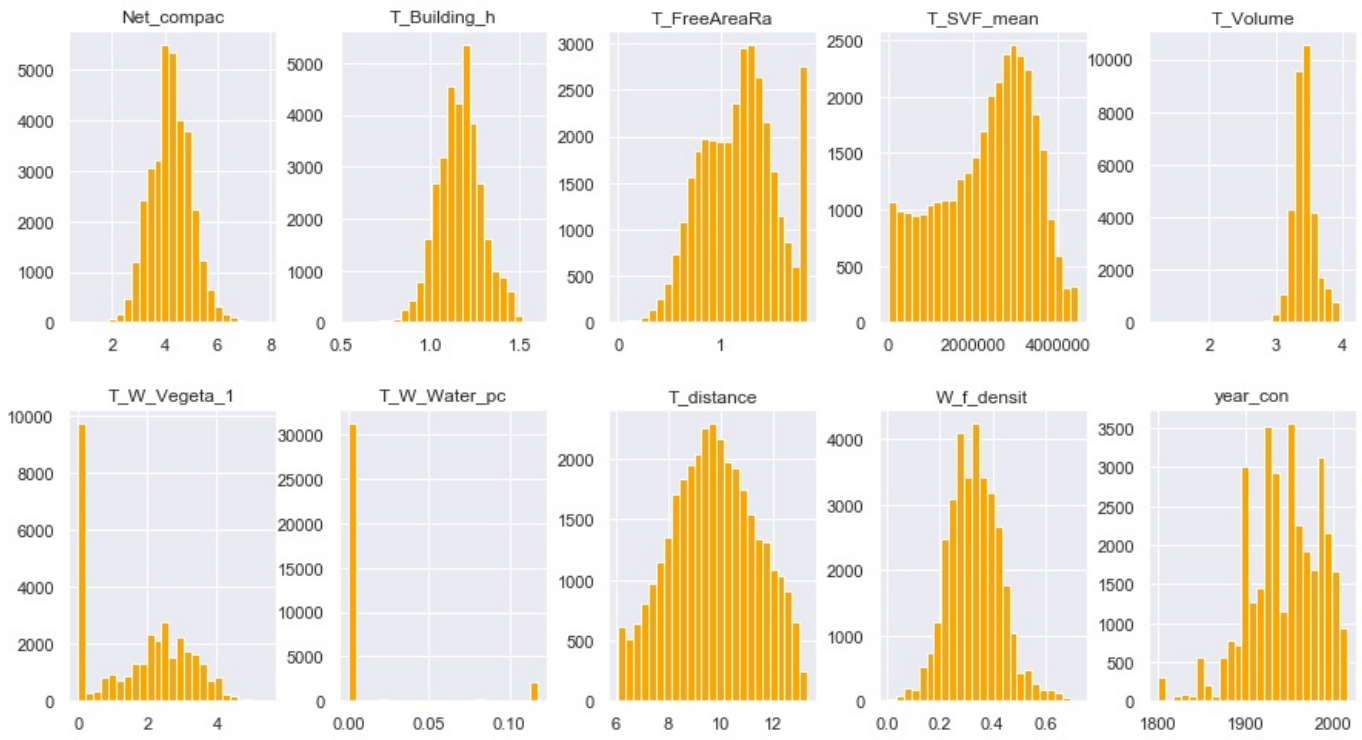
	Building_h	Volume	Net_compac	SVF_mean	distance	year_con	W_Vegeta_1	W_Water_pc	W_f_densit	FreeAreaRa	T_Building_h	T_
0	7.6	587.300363	3.124169	0.941814	480.307457	1972	0.000000	0.0	0.467	0.533170	1.250195	3
1	5.9	513.017003	3.620959	0.985431	492.025716	1989	9.322614	0.0	0.302	0.630578	1.179993	3
2	5.3	675.622273	4.647524	0.955571	482.764956	1989	9.322614	0.0	0.302	0.804912	1.148370	3
3	4.6	248.607793	4.192632	0.933079	499.163087	1936	9.322614	0.0	0.302	0.809574	1.104959	3

In [40]:
```
# Rearranging columns
new_columns = ['T_Building_h', 'T_Volume','Net_compac', 'T_SVF_mean', 'T_distance','W_f_densit', 'T_W_Vegeta_1',
       'T_W_Water_pc','year_con', 'T_FreeAreaRa']
data = data[new_columns]
data.head(2)
```

Out[40]:

	T_Building_h	T_Volume	Net_compac	T_SVF_mean	T_distance	W_f_densit	T_W_Vegeta_1	T_W_Water_pc	year_con	T_FreeAreaRa
0	1.250195	3.445192	3.124169	2.012038e+06	10.840317	0.467	0.000000	-0.0	1972	0.759915
1	1.179993	3.411073	3.620959	3.651378e+06	10.908262	0.302	2.538529	-0.0	1989	0.951746

In [41]:
```
# Checking distribution of transfomed parameters
data.hist(figsize=(15,8),layout=(2,5),bins=25,color='orange')
pyplot.show()
```



In [42]:
```
# Removing indix and name to normalize the numerical values
min_max_scaler = MinMaxScaler()
data_scaled = min_max_scaler.fit_transform(data)
data_temp = pd.DataFrame(data_scaled, index=data.index)
data_temp.head(3)
```

Out[42]:

	0	1	2	3	4	5

6	7	8	9

0	0.671664	0.775913	0.336509	0.452890	0.658869	0.643950	0.000000	0.0	0.792627	0.412034
1	0.605353	0.763932	0.405943	0.821891	0.668314	0.414465	0.464988	0.0	0.870968	0.516047
2	0.575484	0.787961	0.549423	0.547329	0.660866	0.414465	0.464988	0.0	0.870968	0.726340

In [43]:
```
# joining normalized data to original dataframe
df= df.join(data_temp,rsuffix='N_')
```

In [45]:
```
# Column names to string
df.columns = df.columns.map(str)
df = df.rename(columns={"index":"OrigIndex"})
```

In [46]:
```
# Export a sample of data to excel to verify joinning has been implemented without error
df.to_excel(r'D:\Donnees Bat. Cerema\BDTOPO_CEREMA_2\Nantes database\UrbanParamTransformed.xlsx', index = 0)
```

Clustering

K-means

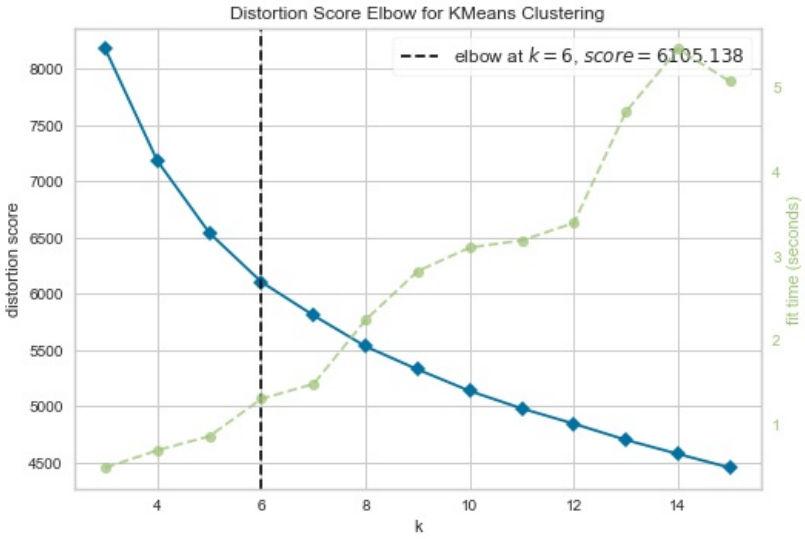
In [47]:
```
from yellowbrick.cluster import SilhouetteVisualizer
from yellowbrick.cluster.elbow import kelbow_visualizer
from yellowbrick.datasets.loaders import load_nfl
from yellowbrick.cluster import InterclusterDistance
from sklearn.linear_model import Lasso
```

In [49]:
```
# Filtering out normalized data for clustering = raw expression matrix data with 10 dimensions
x = df.iloc[:, 40:50]
x.head(2)
```

Out[49]:

	0	1	2	3	4	5	6	7	8	9
0	0.671664	0.775913	0.336509	0.452890	0.658869	0.643950	0.000000	0.0	0.792627	0.412034
1	0.605353	0.763932	0.405943	0.821891	0.668314	0.414465	0.464988	0.0	0.870968	0.516047

In [50]:
```
# K-elbow graph to determine the recommended number of clusters (k):
kelbow_visualizer(KMeans(random_state=42), x, k=(3,16))
```




```
C:\Users\obaidullah.yaqubi\AppData\Roaming\Python\Python38\site-packages\sklearn\base.py:209: FutureWarning: From version 0.24, get_params will raise an AttributeError if a parameter cannot be retrieved as an instance attribute. Previously it would return None.
  warnings.warn('From version 0.24, get_params will raise an '
```

Out[50]:
```
KElbowVisualizer(ax=<matplotlib.axes._subplots.AxesSubplot object at 0x000001D33B970310>,
                 k=None, model=None)
```

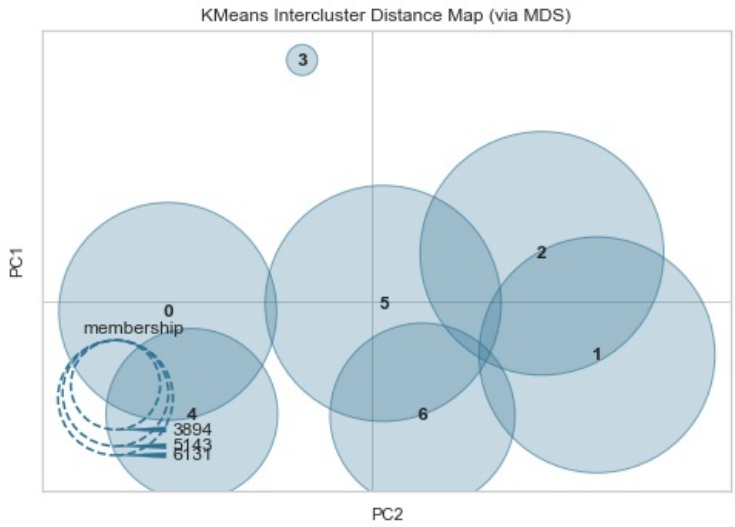
In [52]:
```
# Finding the index of row closest to the centroid
from scipy.spatial.distance import cdist
```

In [53]:
```
from matplotlib.ticker import  FormatStrFormatter
```

K-Mean when K=7

In [55]:
```
# Measuring the closest by selecting the minimum euclidean distance of centroid to each instance
model = KMeans(n_clusters=7)
clustAssign = model.fit_predict(x.as_matrix())
min_dist = np.min(cdist(x.as_matrix(), model.cluster_centers_,'euclidean'), axis=1)
y = pd.DataFrame(min_dist, index=x.index, columns =['KM7_Dist'])
Z = pd.DataFrame(clustAssign, index=x.index, columns = ['KM7'])
PAP = pd.concat([y,Z], axis=1)

visualizer = InterclusterDistance(model)
visualizer.fit(x)
visualizer.show()

df=df.join(PAP)
grouped = PAP.groupby(['KM7'])
grouped.idxmin()
```

```
<ipython-input-55-9095ebd8a8b6>:4: FutureWarning: Method .as_matrix will be removed in a future version. Use .val
ues instead.
  clustAssign = model.fit_predict(x.as_matrix())
<ipython-input-55-9095ebd8a8b6>:5: FutureWarning: Method .as_matrix will be removed in a future version. Use .val
ues instead.
  min_dist = np.min(cdist(x.as_matrix(), model.cluster_centers_,'euclidean'), axis=1)
```



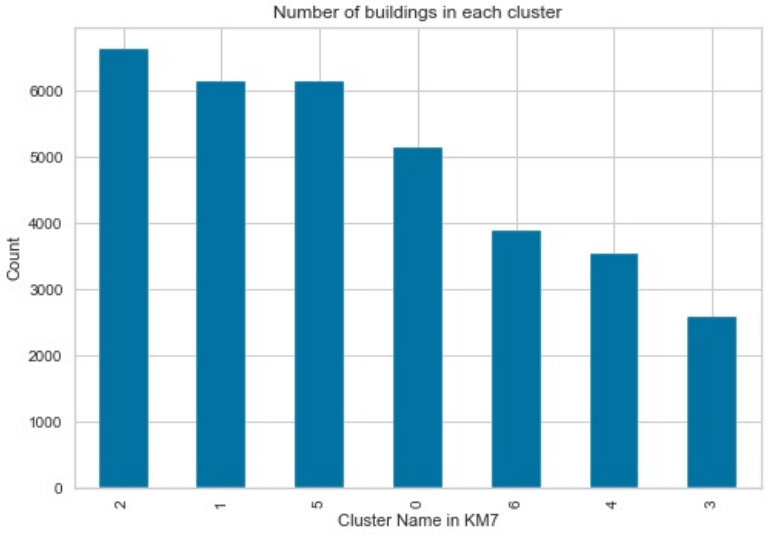
Out[55]:

KM7	KM7_Dist
0	1796
1	19688
2	2317
3	12241
4	29265
5	8571
6	5304

In [56]:
```
# Plot frequency plot of each cluster with labels
S = df['KM7']
fig, ax = plt.subplots()
S.value_counts().plot(ax=ax, kind='bar')
ax.set_xlabel("Cluster Name in KM7")
plt.title("Number of buildings in each cluster")
ax.set_ylabel("Count")
plt.rcParams['axes.edgecolor']= 'black'
```

```
average_silhouette = silhouette_score(x,S)
Calinski = metrics.calinski_harabasz_score(x,S)
Davies = davies_bouldin_score(x,S)
print("Average Silhouette score = "+str(average_silhouette))
print("Calinski score = "+ str(Calinski))
print("Average Davies score = "+ str(Davies))
```

```
Average Silhouette score = 0.17369165423960134
Calinski score = 6424.648435942717
Average Davies score = 1.5735142237019613
```



In [57]:

```
# Exporting centroid information of KMeans 7 for further processing
Gp = pd.DataFrame(grouped.idxmin())
Test1 = pd.DataFrame(df, index=Gp.KM7_Dist)
Test1.rename(columns={'KM7_Dist':'KM7_D'}, inplace=True)
Test1 = Test1.reset_index()
Test2 = pd.DataFrame(model.cluster_centers_)
Test2 = Test2.add_suffix('_Centroid')
Test1=Test1.merge(Test2,left_index=True,right_index=True)
Test1.head(2)
```

Out[57]:

	KM7_Dist	OrigIndex	UID	ID	USAGE1	Unique_ID	T_perimi	NEIGHBORS	indices	Total_area	...	0_Centroid
0	1796	3719	55430	BATIMENT0000000302945900	RÃ?Æ?Ã?â???Ã?Æ?Ãçâ?¬Å¡Ã?â??Ã?Â©sidentiel	55431	37	55430, 55433	55429, 55432	246.612713	...	0.570729
1	19688	45667	16696	BATIMENT0000000304000993	RÃ?Æ?Ã?â???Ã?Æ?Ãçâ?¬Å¡Ã?â??Ã?Â©sidentiel	16697	40	16696	16695	319.045484	...	0.569461

2 rows × 63 columns

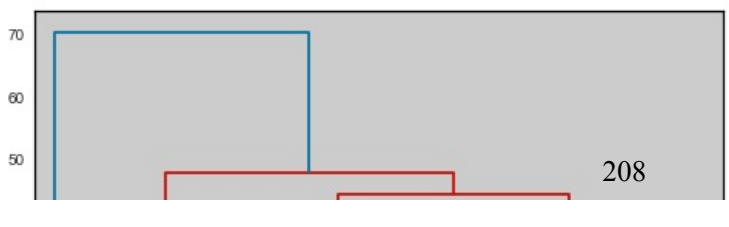
In [59]:

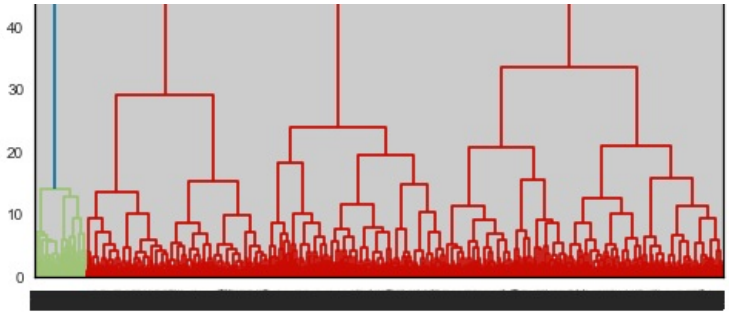
```
# Export a sample of data to excel to
Test1.to_excel(r'D:\Donnees Bat. Cerema\BDTOPO_CEREMA_2\Nantes database\TransformedKM7Centers.xlsx', index = 0)
```

Hierarchical clustering (agglomerative clustering)

In [75]:

```
# Plot dendrogram to visualize suggested number of clusters
dendro = sch.dendrogram(sch.linkage(x, method='ward', metric='euclidean'))
```

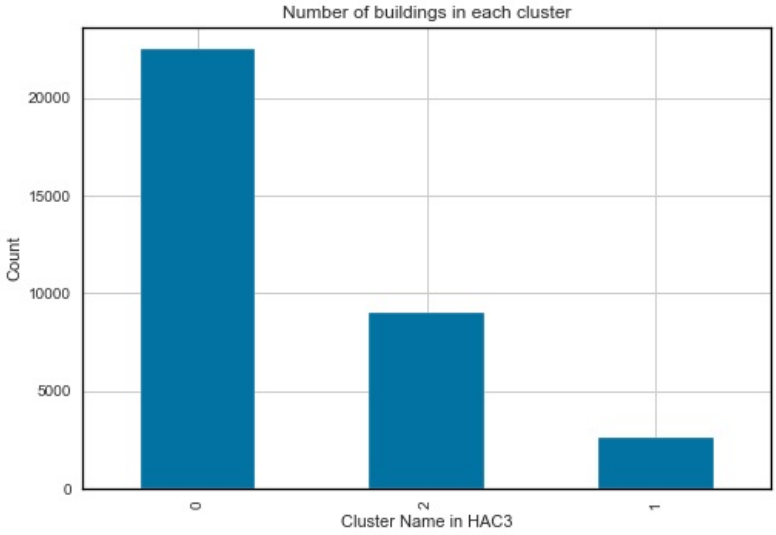




In [76]:
```
# Hierarchical clustering
Hclustering = AgglomerativeClustering(n_clusters=3, affinity = "euclidean", linkage = "ward")
Hclustering.fit(x)
df['HAC3'] = Hclustering.labels_
```

In [78]:
```
# Plot frequency plot of each cluster with labels
S = df['HAC3']
###
fig, ax = plt.subplots()
S.value_counts().plot(ax=ax, kind='bar')
ax.set_xlabel("Cluster Name in HAC3")
plt.title("Number of buildings in each cluster")
ax.set_ylabel("Count")
plt.rcParams['axes.edgecolor']= 'black'

average_silhouette = silhouette_score(x,S)
Calinski = metrics.calinski_harabasz_score(x,S)
Davies = davies_bouldin_score(x,S)
print("Average Silhouette score = "+str(average_silhouette))
print("Calinski score = "+ str(Calinski))
print("Average Davies score = "+ str(Davies))
```

```
Average Silhouette score = 0.12787249185391344
Calinski score = 7029.822169962535
Average Davies score = 1.792159450119397
```



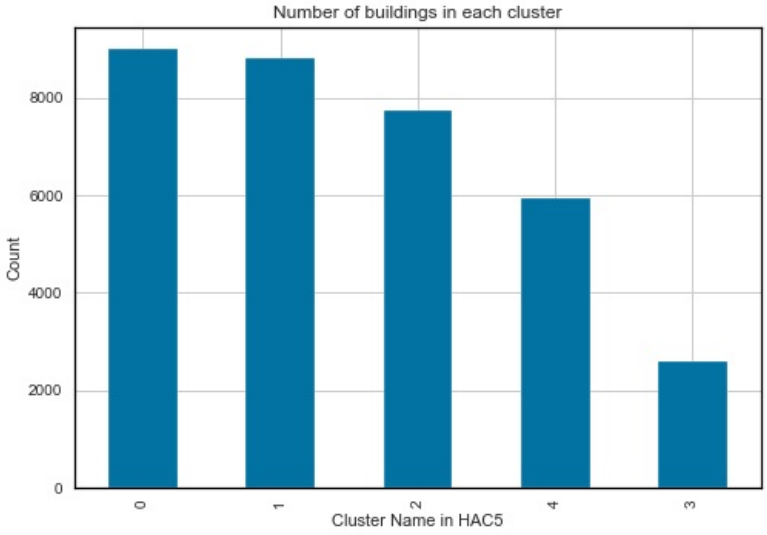
In [79]:
```
# In order to find the centroid of Hierarchical clustering
# first take an average array of values in each cluster and then find the closest item in the database for that c
AC = df.iloc[:,40:50][(df["HAC3"]==1)]
C = np.mean(AC, axis=0)
```

In [80]:
```
# Hierarchical clustering
Hclustering = AgglomerativeClustering(n_clusters=5, affinity = "euclidean", linkage = "ward")
Hclustering.fit(x)
df['HAC5'] = Hclustering.labels_
```

In [81]:
```
# Plot frequency plot of each cluster with labels
S = df['HAC5']
fig, ax = plt.subplots()
```

```
S.value_counts().plot(ax=ax, kind='bar')
#
ax.set_xlabel("Cluster Name in HAC5")
plt.title("Number of buildings in each cluster")
ax.set_ylabel("Count")
plt.rcParams['axes.edgecolor']= 'black'

average_silhouette = silhouette_score(x,S)
Calinski = metrics.calinski_harabasz_score(x,S)
Davies = davies_bouldin_score(x,S)
print("Average Silhouette score = "+str(average_silhouette))
print("Calinski score = "+ str(Calinski))
print("Average Davies score = "+ str(Davies))
```

```
Average Silhouette score = 0.12700956935904165
Calinski score = 6103.924018271532
Average Davies score = 1.838803860224349
```



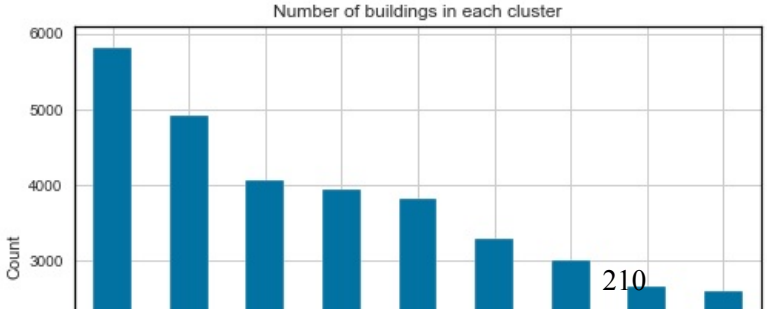
In [82]:

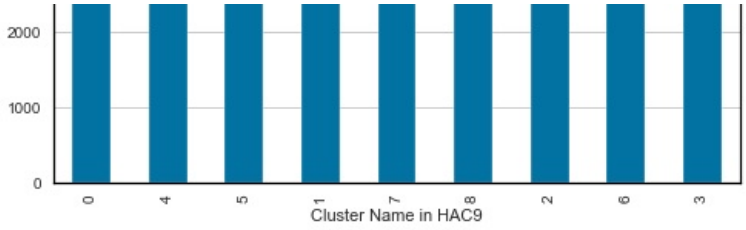
```
# Hierarchical clustering
Hclustering = AgglomerativeClustering(n_clusters=9, affinity = "euclidean", linkage = "ward")
Hclustering.fit(x)
df['HAC9'] = Hclustering.labels_
```

In [83]:

```
# Plot frequency plot of each cluster with labels
S = df['HAC9']
###
fig, ax = plt.subplots()
S.value_counts().plot(ax=ax, kind='bar')
#plt.gca().xaxis.set_major_formatter(FormatStrFormatter('Cluster Number'))
ax.set_xlabel("Cluster Name in HAC9")
plt.title("Number of buildings in each cluster")
ax.set_ylabel("Count")
plt.rcParams['axes.edgecolor']= 'black'

average_silhouette = silhouette_score(x,S)
Calinski = metrics.calinski_harabasz_score(x,S)
Davies = davies_bouldin_score(x,S)
print("Average Silhouette score = "+str(average_silhouette))
print("Calinski score = "+ str(Calinski))
print("Average Davies score = "+ str(Davies))
```

```
Average Silhouette score = 0.10799882269909711
Calinski score = 4435.296106113964
Average Davies score = 2.021605919038311
```





Density based clsutering (DBSCAN)

In [84]:
```
from sklearn_extra.cluster import KMedoids
from sklearn.cluster import DBSCAN
```

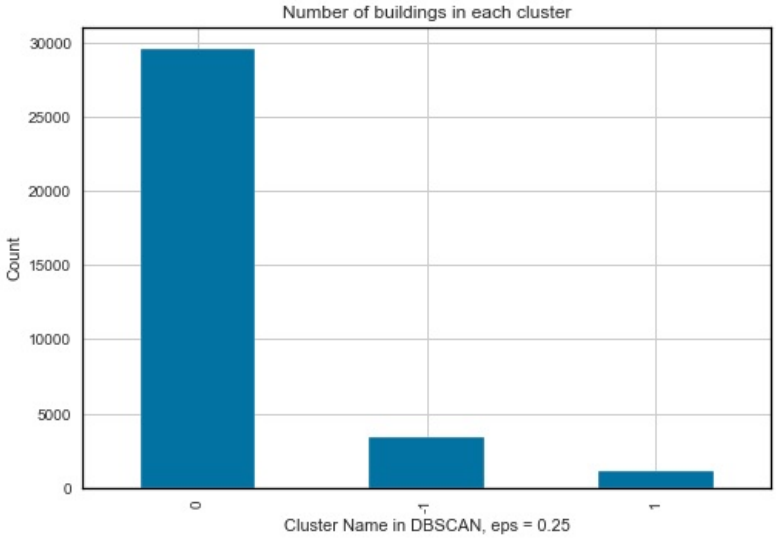
In [89]:
```
## DBSCAN clustering
cluster_DBS = DBSCAN(eps=0.25,min_samples=50).fit(x)
df['DBSCAN_005']=cluster_DBS.labels_
# print(cluster_DBS.cluster_centers_)
## Number of clusters
print(len(set(df['DBSCAN_005']))-(1 if -1 in df['DBSCAN_005'] else 0))
## Number of outliers which are indicated as (-1) in DBSCAN
list(df['DBSCAN_005']).count(-1)
```

```
3
```

Out[89]: 3395

In [90]:
```
S = df['DBSCAN_005']
###
fig, ax = plt.subplots()
S.value_counts().plot(ax=ax, kind='bar')
#plt.gca().xaxis.set_major_formatter(FormatStrFormatter('Cluster Number'))
ax.set_xlabel("Cluster Name in DBSCAN, eps = 0.25")
plt.title("Number of buildings in each cluster")
ax.set_ylabel("Count")
plt.rcParams['axes.edgecolor']= 'black'

average_silhouette = silhouette_score(x,S)
Calinski = metrics.calinski_harabasz_score(x,S)
Davies = davies_bouldin_score(x,S)
print("Average Silhouette score = "+str(average_silhouette))
print("Calinski score = "+ str(Calinski))
print("Average Davies score = "+ str(Davies))
```

```
Average Silhouette score = 0.27955932678531437
Calinski score = 3095.388785159038
Average Davies score = 2.418854662948036
```



Spectral clustering

In [93]:
```
from sklearn.cluster import SpectralClustering
```

In [94]:
```
clustering = SpectralClustering(n_clusters=7, assign_labels="discretize", random_state=0).fit(x)
df['Spec7']=clustering.labels_
```

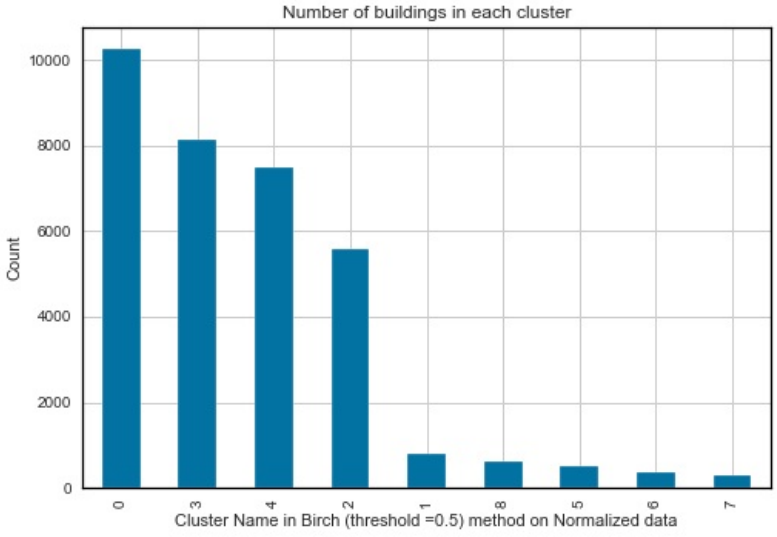
This technique requires strong processing power. 8GB ram is not sufficient

Birch clustering

In [95]:
```
from sklearn.cluster import Birch
# it can identify the number of clusters or the number can be assigned manually
```

In [99]:
```
bclust=Birch(threshold=0.5,n_clusters=None)
labels_b = bclust.fit_predict(x)
df['Birch05'] = labels_b

S = df['Birch05']
###
fig, ax = plt.subplots()
S.value_counts().plot(ax=ax, kind='bar')
#plt.gca().xaxis.set_major_formatter(FormatStrFormatter('Cluster Number'))
ax.set_xlabel("Cluster Name in Birch (threshold =0.5) method on Normalized data")
plt.title("Number of buildings in each cluster")
ax.set_ylabel("Count")
plt.rcParams['axes.edgecolor']= 'black'

average_silhouette = silhouette_score(x,S)
Calinski = metrics.calinski_harabasz_score(x,S)
Davies = davies_bouldin_score(x,S)
print("Average Silhouette score = "+str(average_silhouette))
print("Calinski score = "+ str(Calinski))
print("Average Davies score = "+ str(Davies))
```

```
Average Silhouette score = 0.15611501563321556
Calinski score = 4034.6334870124115
Average Davies score = 1.8389278961399687
```

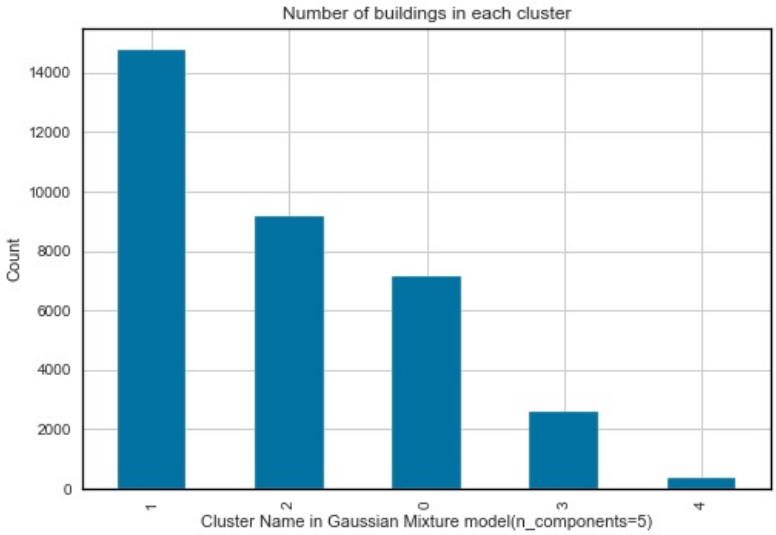


Gausian Mixture clusering

In [104…
```
# GMM with expectation minimization
EM = GaussianMixture(n_components=5)
EM.fit(x)
cluster = EM.predict(x)
df['GM5']=cluster

S = df['GM5']
###
fig, ax = plt.subplots()
S.value_counts().plot(ax=ax, kind='bar')
#plt.gca().xaxis.set_major_formatter(FormatStrFormatter('Cluster Number'))
ax.set_xlabel("Cluster Name in Gaussian Mixture model(n_components=5)")
plt.title("Number of buildings in each cluster")
```

```
ax.set_ylabel("Count")
plt.rcParams['axes.edgecolor']= 'black'

average_silhouette = silhouette_score(x,S)
Calinski = metrics.calinski_harabasz_score(x,S)
Davies = davies_bouldin_score(x,S)
print("Average Silhouette score = "+str(average_silhouette))
print("Calinski score = "+ str(Calinski))
print("Average Davies score = "+ str(Davies))
```

```
Average Silhouette score = 0.043594242550450 52
Calinski score = 4004.9206189491892
Average Davies score = 3.0649277622602584
```



Saving file back to shapefile

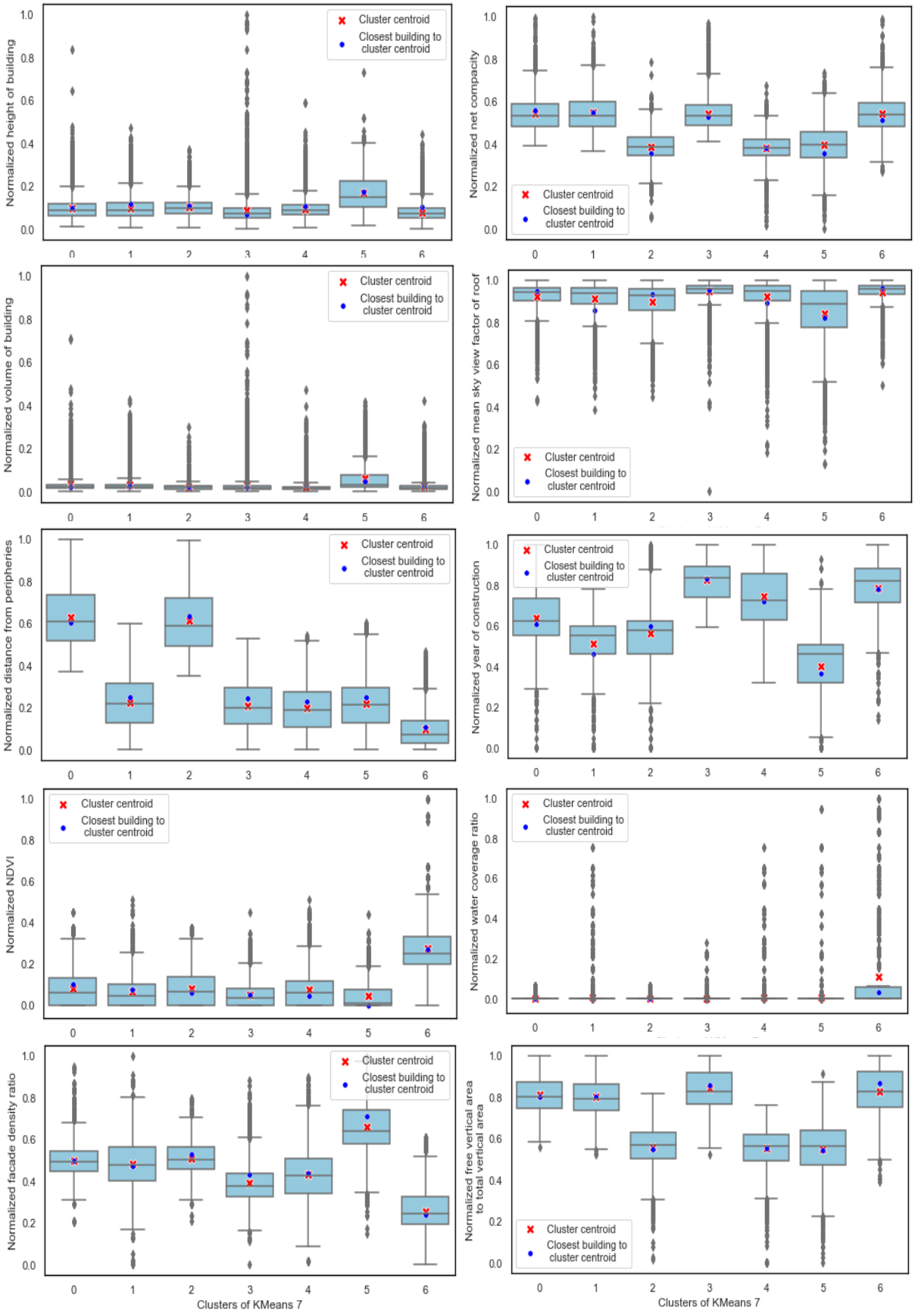
In [111…

```
# Export a sample of data to excel to verify joinning has been implemented with error
df.to_excel(r'D:\Donnees Bat. Cerema\BDTOPO_CEREMA_2\Nantes database\UrbanParamTransformed.xlsx', index = 0)
```

In [113…

```
# Save as new shapefile to be openned in QGIS
df.to_file(driver = 'ESRI Shapefile', filename= r'D:\Donnees Bat. Cerema\BDTOPO_CEREMA_2\Nantes database\Bld_RSU_
```

Appendix 2-5:


Normalized height of building
Normalized net compacity
Normalized volume of building
Normalized mean sky view factor of roof
Normalized distance from peripheries
Normalized year of construction
Normalized NDVI
Normalized water coverage ratio
Normalized facade density ratio
Normalized free vertical area to total vertical area
Cluster centroid
Closest building to cluster centroid
Clusters of KMeans 7

Appendix 2-6:

Most of simulations tools are able to estimate the volume of air that enters or exits a mechanically ventilated building. Calculation of air inflow rate (infiltration rate + air inflow through the window) in naturally ventilated buildings is, however, a more challenging task.

Airtightness of buildings are usually determined with pressure test. Nevertheless, those tests are carried out when windows and doors of buildings are firmly closed.

(Moujalled, Leprince, and Melois n.d.) presented a comprehensive study that involved testing airtightness of more than 200 000 buildings in France at 4 Pa. Graph presented below show the results of their study.

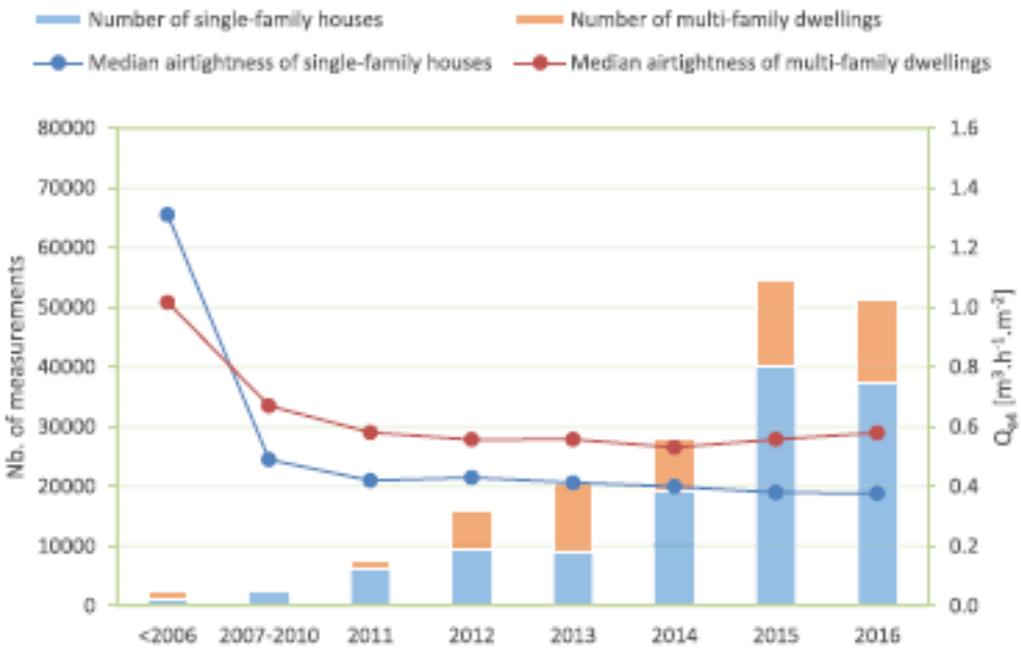


Figure 0-1 : Number of building airtightness tests and their results according to the year of construction of the building in France (source: Moujalled et al. 2019)

Air leakage in the graph is calculated in ($m^3h^{-1}m^{-2}$), but many software use Air change per hour (ACH) as an indicator to measure airtightness.

In ($m^3h^{-1}m^{-2}$) the volume of air in (m3) per (hour) through the loss surface is calculated.

Air loss surface according to RT2012, is the surface area of a zone (horizontal and vertical) except the basement floor.

Conversion of airtightness values from ($m^3h^{-1}m^{-2}$) to (ACH) and vice versa should theoretically be done case by case, because there is not solid relationship between surface area of rooms and

their volume. Since we already know that the height of majority of floor height are around 3 meters than it is possible to establish *an approximate* relationship between the two.

- To test this theory, imagine a building of 3m by 3m by 3m meters.

Volume = 27 m^3

Loss surface = 3m (3m x 4) + 2 x (3m x 3m) = 54 m^2.

Loss surface (if basement floor) = 3m (3m x 4) + (3m x 3m) = 45 m^2.

A value of 0.37 $m^3h^{-1}m^{-2}$ equals to 0.37 x 54 = 21 m^3h^{-1}. This value is then divided by the volume and equals to $21m^3h^{-1}/27m^3$ = 0.74 (ACH).

A value of 0.5 $m^3h^{-1}m^{-2}$ for the same cube equals to 1 ACH.

A value of 0.7 $m^3h^{-1}m^{-2}$ for the same cube equals to 1.4 ACH.

A value of 1 $m^3h^{-1}m^{-2}$ for the same cube equals to 2 ACH.

A value of 1.2 $m^3h^{-1}m^{-2}$ for the same cube equals to 2.4 ACH.

- Now imagine a building of 4m by 4m by 3m meters

Volume = 48 m^3

Loss surface = 3m (4m x 4) + 2 x (4m x 4m) = 80 m^2.

Loss surface (if basement floor) = 3m (4m x 4) + (4m x 4m) = 64 m^2.

A value of 0.37 $m^3h^{-1}m^{-2}$ equals to 0.37 x 80 = 29.6 m^3h^{-1}. This value is then divided by the volume and equals to $29.6m^3h^{-1}/48m^3$ = 0.616 (ACH).

A value of 0.5 $m^3h^{-1}m^{-2}$ for the same cube equals to 0.83 ACH.

A value of 0.7 $m^3h^{-1}m^{-2}$ for the same cube equals to 1.17 ACH.

A value of 1 $m^3h^{-1}m^{-2}$ for the same cube equals to 1.67 ACH.

A value of 1.2 $m^3h^{-1}m^{-2}$ for the same cube equals to 2 ACH.

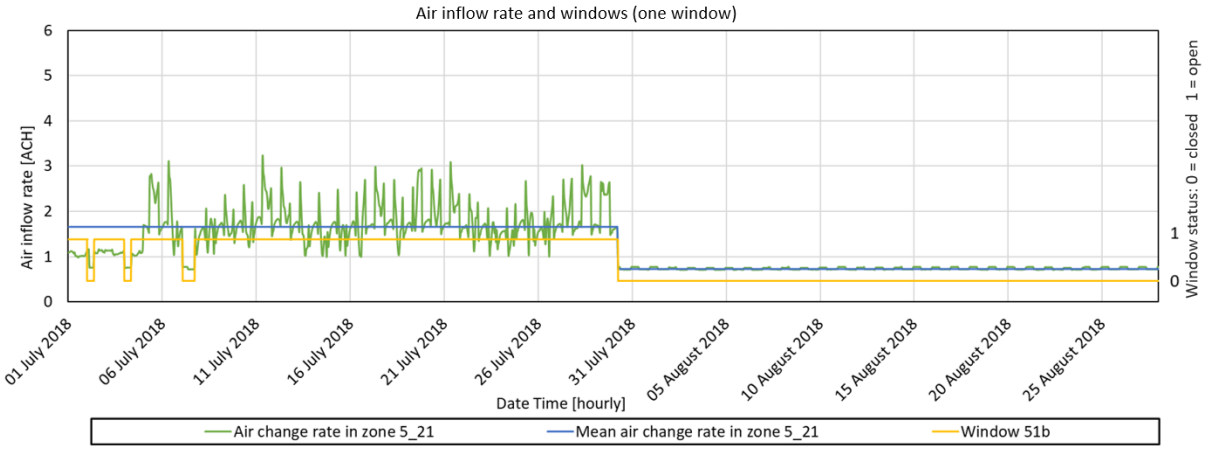
We notice that shape of air zone makes the comparison between the two units of measurement difficult, but the equation ACH= (surface leakage rate*constant/ (area * height)) should work with constant equal to 58, as a rule of thumb.

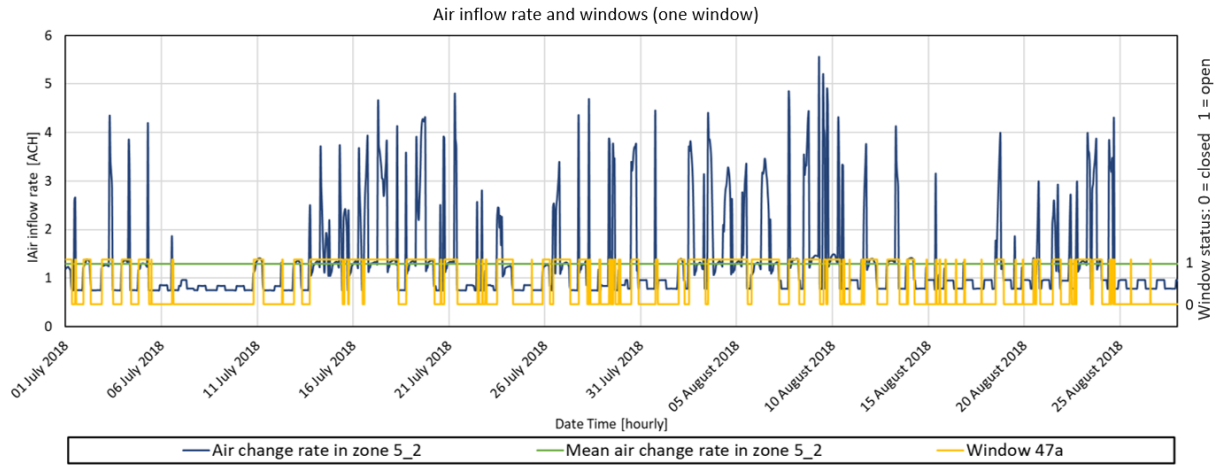
Coming back to subject of measuring air inflow rate in naturally ventilated buildings through the windows and other orifices.

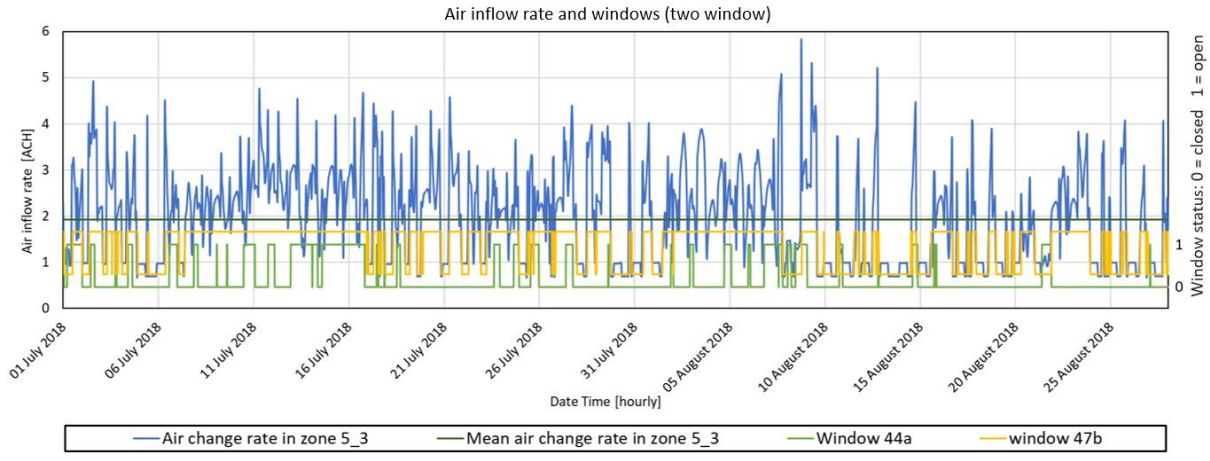
A model co-simulated with Contam and Trnsys was calibrated with onsite data to measure the influence of window openness ratio and external shading. (**IBPSA conference paper in Appendix 2-7**).

After calibration, we measured hourly air inflow rate into different zones of apartments. Results are presented as follows:

Effects of **window openning** on **indoor air change** rate calculated using CONTAM-TRNSYS co-simulation tool:







From the graphs, you can notice that average air change rate when window is regularly opened and closed is different with number of individual windows. For one window it is approximately 1.3 [ACH] and for two windows it is close to 2 [ACH]. When it is closed, it drops to 0.7 [ACH].

Results are comparable with findings of https://www.nature.com/articles/7500229

Appendix 2-7:

- Article on passive strategies presented in IBPSA 2021

BS 2021 1-3 SEPT BRUGES

INTERNATIONAL BUILDING PERFORMANCE SIMULATION ASSOCIATION

Summer Passive Strategies Assessment Based on Calibrated Building Model Using on Site Measurement Data

Obaidullah Yaqubi[1,2,3], Auline Rodler[1,2], Sihem Guernouti[1,2,3], Marjorie Musy[1,2,3]
[1]Equipe de recherche BPE, CeremaOuest, Nantes, France
[2]Institut de Recherche en Sciences et Techniques de la Ville (IRSTV), Nantes, France
[3]CNRS UMR 6183, GeM, Université de Nantes, Nantes, France

Abstract

Existing state of the art often view passive strategies as a mean for energy efficiency, while their effect on summer thermal comfort in temperate climatic conditions has been given lesser attention. Assessing these strategies require practitioners to provide the right combination of tools; proper measurement indices; and perform simulations as close as possible to real-life circumstances. This paper studies the issue with reference to an existing apartment building in real settings. Results here are in line with previous findings that recommend a synergy of passive summer strategies to reduce the impacts of over temperature in buildings.

Key Innovations

- Contam-Trnsys coupled building model was calibrated with on situ data
- Typical Weather file was modified with data collected from the site
- Hourly thermal comfort of a real building over two months of summer was gauged with adaptive and static thermal comfort indices
- Impact of window openness ratio and window blinders were assessed to see if they can reduce thermal discomfort in overheated periods.

Practical Implications

Practitioners can use the methodology proposed here to measure and optimize the effect of passive strategies in reduction of thermal discomfort due to over temperature. Using multiple indices to measure thermal comfort allows practitioners to take into account cultural norms as well as occupants' thermal expectations and preferences in their decisions.

Introduction

With the changes in worldwide climate conditions, extreme weather events will become more frequent and severe that would lead to substantial impacts. Heatwaves in particular can cause severe overheating in buildings causing several problems ranging from thermal discomfort and productivity reduction to illnesses and even death of occupants (Gamero-Salinas et al. (2020); Ozarisoy and Elsharkawy (2019)). Occupants in such circumstances contribute decidedly in energy consumption and indoor thermal comfort. For instance, changing set point temperature in mechanically ventilated buildings; or by manually adjusting windows, shutters and etc in mixed mode and naturally ventilated buildings. Their contribution is also recognized as a significant factor in uncertainty of building performance modelling.
Modern validated whole building energy modelling tools such as Trnsys, DOE-2, EnergyPlus and Pleiade Comfie provide valuable insights into energy use and thermal comfort of occupants in the buildings. They are capable of addressing a broad spectrum of energy and temperature related issues in various combinations. However, none of them alone offer sufficient capabilities to model whole building subsystems and interaction of occupants in those subsystems in isolation without making oversimplified assumptions. One approach to overcome this limitation is to use collaborative-simulation (co-simulation) as an integrated method to simulation. It is based on the idea that distinct interacting building subsystems are best simulated in specialized mature tools, dedicated to the given subsystem (Taveres-Cachat and Goia (2020); Dols et al. (2016)).
To cover given issues, this study first modifies typical weather data file with measured data; calibrates co-simulation model of coupled Contam and Trnsys with observed temperature data using Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Coefficient of Variation of the Root Mean Square Error (CV RMSE) statistical performance indicators employed by Taveres-Cachat and Goia (2020); then it proceeds with selection of thermal comfort measurement indices; and at the end assesses how openness ratio of windows and window shutters affect indoor thermal comfort of occupants during an overheated period.

Site Introduction

Observational study was carried out on a five storey apartment building located in northeast of Nantes (47.2184 °N, 1.5536 °W). Continuous measurement of data from window status, outside air temperature

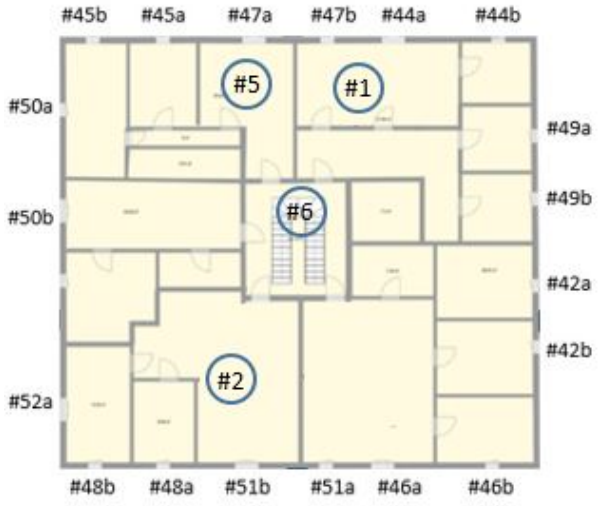


Figure 1: location plan of data collection sensors: circled numbers show indoor air temperature sensors in the living rooms and stairwell

and inside air temperature from three apartments and stairwell in the fifth storey of the building were recorded at an interval of 10 minutes. Readers can have more details about the building data collected in Rodler et al. (2018) paper. Specification of data collection equipment for the given items are described in Table 1. In this apartment building, there were two types of windows with manual roller blinds/shutters. Window type 1 was $2.05m^2$ and type 2 was $3.01m^2$. Frame, glazing and other properties of the both window types were identical. Both window type 1 and type 2 had $1600mm^2$ trickling vent.Both window types were single side hung casement windows with two fully operable windows which opened inside. Windows were equipped with roller blinds to control solar gains. Discharge coefficient of windows was estimated 0.5. Window sensors recorded the state of window with boolean signal (0,1), where 1 indicates window is open and 0 when it is closed. The sensor did not read any information about the openness ratio and the state of shutters. Figure 1 shows detailed plan of indoor air temperature reading points $(1, 2, 5, 6)$ and windows' sensors $(45b, 45a, 47a, 47b, 44a, 44b, 49a, 49b, 50a, 50b, 52a, 48b, 48a, 51b)$. Indoor temperature sensors were installed in the living room of apartments away from direct exposure to sunlight.

Local authority of housing provided information about the U value of composite exterior wall and double glazing windows, which were 0.380 $[W/m^2K]$ and 2.89 $[W/m^2K]$, respectively. Information concerning U value of roof was not available in the local authority for this building. All apartments had mechanical exhaust-only ventilation in the kitchen and bathroom. Quantity of air exhausting via each mechanical fan was given in the range of 40 to 100 $m^3/hour$. There were two types of doors in each apartment: bedroom doors were 2 by 0.8 m and discharge coefficient was estimated to approximately 0.75; bathroom doors were 2 by 0.75 m with a similar discharge coefficient. Bedroom doors were assumed to be open during the day from $8h$ to $22h$ and closed at night. Under-door crack in bedroom doors' were measured to be $4mm$ by $0.8m$. Bathroom doors were considered to be closed during day and night, but in contrast to bedroom doors, they had a larger under-door orifice of $10mm$ by $0.75m$ and an over-door crack of $4mm$ by $0.75m$ allowing living room air to be extracted through the extraction vent in the bathroom.

Co-simulation tools

Trnsys package is a combined dynamic modeling software that allows the evaluation and assessment of thermal and electrical energy systems. The package consists of graphical front-end interfaces to intuitively create simulations; for multi-zone buildings it is type56. Trnsys type56 is a non-geometrical balance model with one air node in a zone illustrating thermal capacity of air volume in the zone. This thermal capacity is separate from the volume of zone, which is an additional input. Trnsys type56 automatically generates inputs of multi-zone building such as view factors, sunlit factors and distribution factors from geometric information. Transient heat conduction through envelope elements in type56 are calculated using conductive heat transfer function method developed by Mitalas and Stephenson. Windows thermally in Trnsys are viewed as an external wall with no thermal mass; partially transparent to solar radiation but opaque to long wave heat gains. Longwave heat gains are regarded to only occur at the surfaces. Incident shortwave radiation is calculated by surface modulus using solar absorptance coefficient of material (Type56-Manual (2017); Khalifa et al. (2015)). Convective heat fluxes to the air node is calculated as a summation of infiltration gains; ventilation gains;internal convective gains (by equipment, people, lighting, etc); and gains due to convective air flow between air zones' boundary conditions. The user can define manually air mass flow into a zone in type56; but type56 alone does not automatically calculate air mass flow to adjacent zones. Calculation and definition of air mass flow rate exchanges between adjacent zones in type56 can be a tedious task for user if done manually. Contam on the other hand can automatically calculate air mass flow rate between zones, knowing the geometry and status of airflow paths between zones.

Contam like Trnsys has been in practical use for many years. It is used in a variety of applications, most notably in assessment of ventilation systems, analyses of smoke management systems, contaminant transport, etc. It allows the user to define various air paths such as, stairwells, ducts, orifices, cracks, doors, windows etc. Airflow calculations in Contam are based on non-linear airflow-vs-pressure relationships. Con-

Table 1: Specification of data collection sensors

Measurement type	Quantity	Sensor reference	Accuracy
Outside air temperature	2	AMR WM-us ambient sensor	$\pm 0.1\,^{o}C$ *to* $\pm\ 0.5\,^{o}C$
Inside air temperature	5	AMR WM-us ambient sensor	$\pm 0.1\,^{o}C$ *to* $\pm\ 0.5\,^{o}C$
Window opening sensor	13	AMR wM-Bus 0-10V, mA on/off, ARF8041AA	-

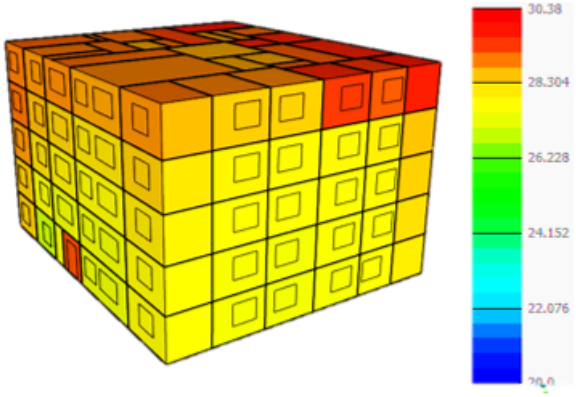


Figure 2: Modelled indoor air temperature of whole building during the overheating period on 05 August at 16h00

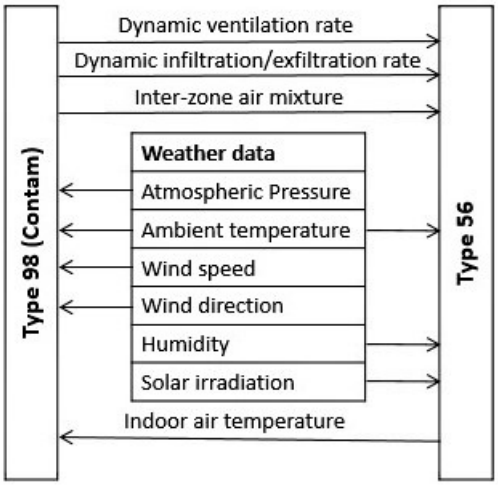


Figure 3: Schematic representation of data exchange between Type 98 (Contam) and Type 56

tam assumes all airflow through the building envelope openings and between zones (large, small, intentional or accidental airflow paths) are governed by Bernoulli hydro-static equation.

$$\Delta P = \left(P_1 + \frac{\rho V_1^2}{2}\right) - \left(P_2 + \frac{\rho V_2^2}{2}\right) + \rho g(z_1 - z_2) \quad (1)$$

Where:

ΔP = total pressure drop between points 1 and 2
P_1, P_2 = entry and exit static pressures
V_1, V_2 = entry and exit velocities
ρ = air density
g = acceleration of gravity ($9.81 m/s^2$)
z_1, z_2 = entry and exit elevations.

Coupling between the heat transfer (type 56) of Trnsys and airflow calculations (type 98) is accomplished via the quasi-dynamic method. In other words, it refers to the coupling between simultaneously running processes where data is exchanged only once within each time-step like in ping-pong. This coupling allows type56 multi-zone building heat transfer model in Trnsys to share input and output with Contam representation of multi-zone building (Khalifa et al. (2015); Dols et al. (2016)). The exchange of input and outputs for airflow and temperature data is shown in figure 3. Development of type 98 by National Institute of Standards and Technology (NIST) has facilitated automatic and standardized coupling between Contam and Trnsys. The difficult task of describing the thermal zones of a building, walls, internal gain, orientation for solar radiation, etc. is performed automatically by following detailed coupling process described in Contam documentation by Dols and Polidoro (2015). This automatic process helps create building description, minimizing modeler's input error. Building's indoor air temperature color-scaled 3D model in Trnsys3D is shown in figure 2.

Method

As can be seen in the figure 2 the building consists of five storeys connected through a central stairwell. All five storeys have similar floor plans; window sizes; and wall thermo-physical properties. Each piece in the apartment i.e. bedroom, living room, and bathroom was treated as a separate zone in the simulation tool connected to other pieces via doors and airflow paths (cracks) under and over the doors. The time step of the simulation was set to 15 min but the output of simulation was collected at hourly intervals. Temperature and humidity ratio of typical weather data file for the given location was modified with measured ones for July and August and used in simulations. Heat gains from occupancy were calculated by dividing the average load per person to average area per person. Load gains from lighting and equipment were calculated from ASHRAE tables for the number of appliances and light bulbs in the apartments (figure 4). Data collection was carried out in a non-intrusive manner in the occupied apartments of the building. As a result, a limited set of on board measured data was collected. Considering case specific limitations and possibilities, RMSE,and MAE statistical indica-

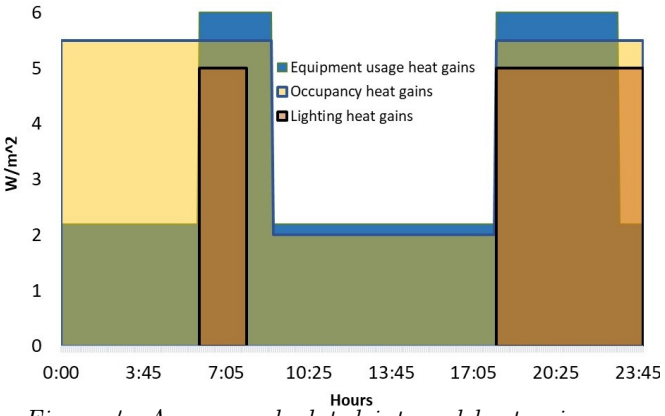


Figure 4: Average calculated internal heat gain profiles for apartments

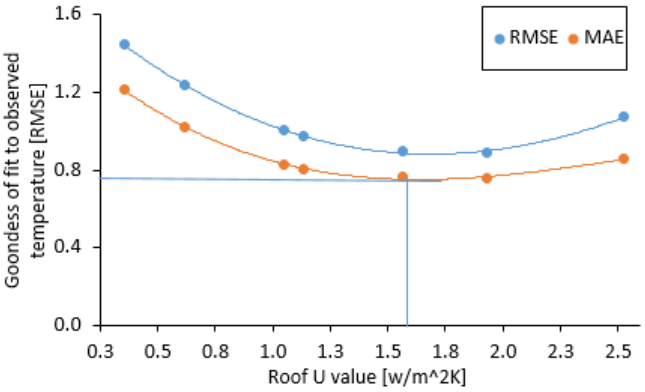


Figure 5: Roof U value calibration

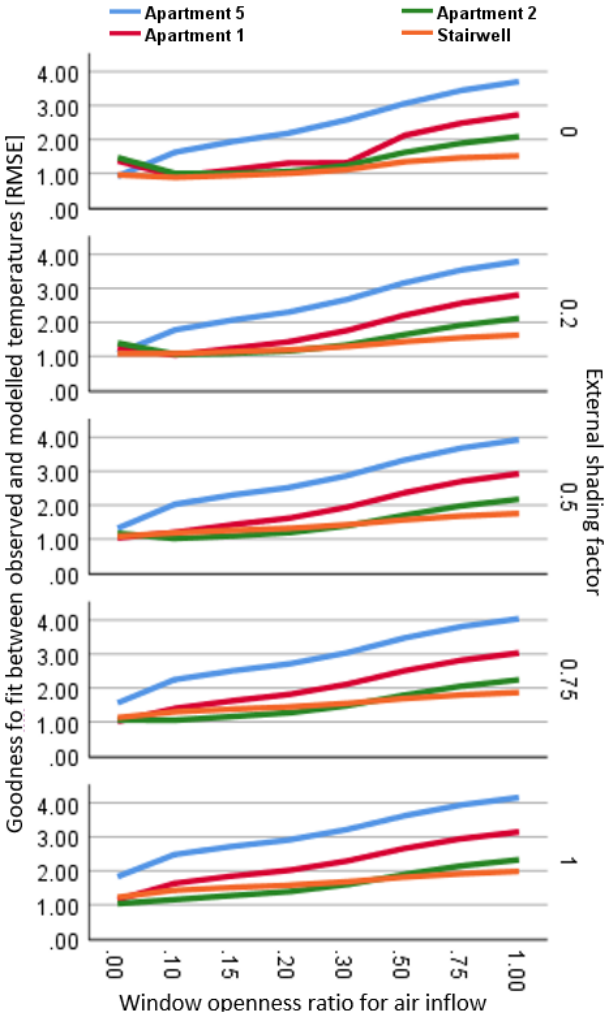


Figure 6: Calibration of window openness ratio for air inflow and external shading factor

tors were selected to evaluate the goodness of fitting parameter to actual physical parameter of building in calibration stage. Multiple simulations were iterated manually for each uncertain building parameter and the value with the lowest RMSE was used to determine the missing physical parameter. The following series of steps where taken to tune the missing physical parameters of building:

First, roof U value was tuned by comparing the co-simulation temperature with measured one in the stairwell. Mainly because the stairwell is less affected by direct solar radiation and airflow through the windows. After running multiple simulations, the U value of roof equals to $1.563\,[W/m^2K]$ was found to fit best with observed temperature (figure 5).

Second, quantity of air exhausting via each mechanical air extraction fan was tuned by fitting modelled indoor temperature with observed one in apartment 2 during vacation time when windows and doors were closed. Air extraction rate at $60m^3/h$ from kitchen air vent and $40m^3/h$ from bathroom was found to fit best the observed temperature.

Finally, calibration of window openness ratio and shading factor is more complicated as it can vary depending on the behavior of occupants, different sizes of windows, orientation, time of the day and etc.In addition, data collection sensors in windows had returned a boolean value that only showed the status of window as opened or closed. In order to calibrate the two parameters, window openness ratio and shading factor were equally changed in all windows and the correlation between goodness of fit to observed data and modelled data was different in each apartment and stairwell as shown in figure 6. The values with smallest RMSE in apartments and stairwell were selected as calibrated values.

Moving forwards with the assessment of passive summer strategies on the calibrated co-simulation model. Three widely used comfort indices were employed to measure and compare summer comfort: predicted mean vote (PMV); upper limit values of EN 16798 adaptive thermal comfort; and upper limit values of ASHRAE 55 adaptive thermal comfort. PMV index rate were calculated using *pythermalcomfort* package developed by Tartarini and Schiavon (2020). This package calculates clothing level as function of outdoor temperature at 6AM in accordance to ASHRAE 55 2017. Indoor average air speed was assumed $0.1m/s$ and metabolic rate 1.2 representing a light

physical activity and nearly stationary air velocity.
In EN 16798 and ASHRAE 55 adaptive thermal comfort measurement approaches, the sum of degree hours (DH) exceeding maximum allowable operative temperature was analyzed.
In both EN 16798 and ASHRAE 55 maximum allowable operative temperature (T_{MAX}) is related to exponentially decaying weighted mean outdoor temperature (T_{RM}) (ASHRAE (2017); Chirico and Magnavita (2019)).

$$T_{RM} = (1-\alpha)[T_{N-1} + \alpha T_{N-2} + \alpha^2 T_{N-3} + \alpha^3 T_{N-4} + \alpha^4 T_{N-5} + ...](^oC) \quad (2)$$

Where:
T_N = Mean Daily temperature of previous day
α = Constant between 0 and 1.
α controls the speed at which running mean outdoor temperature changes. Recommended value for it is between 0.6 and 0.9 corresponding to slow and fast response, respectively. Adaptive comfort theory suggests 0.9 for climates where synoptic-scale (day to day) temperature variations are minor but SCAT (Smart Controls and Thermal Comfort) project which was undertaken in European Union has recommended 0.8 for the region where this study has been carried out. According to ASHRAE 55 in weighted average calculation, prevailing outdoor mean temperature shall be based no fewer than 7 sequential days prior to the target day. In this study mean temperature of 30 previous sequential days were used to calculate mean weighted running temperature. EN 16798-1:2019 which replaces EN 15251 illustrates indoor thermal comfort at category I, II, III level and sometimes level IV. Compliance to level I upper and lower boundary limit is best, and III and IV is considered worst. In ASHRAE 55, two sets of operative temperature acceptability limits are proposed; 90% acceptability is considered for cases when higher standard of thermal comfort is desired; 80% acceptability limit is for typical situations. In this study we concentrate on upper boundary limits because the focus is on summer comfort.
Following equations determine the maximum allowable operative temperatures (T_{MAX}) for EN 16798 and ASHRAE 55: (Gamero-Salinas et al. (2020); Chirico and Magnavita (2019); ASHRAE (2017))

$$T_{MAX}(Category\ I)(^oC) = 0.31T_{RM} + 20.8 \quad (3)$$

$$T_{MAX}(Category\ II)(^oC) = 0.31T_{RM} + 21.8 \quad (4)$$

$$T_{MAX}(Category\ III)(^oC) = 0.31T_{RM} + 22.8 \quad (5)$$

$$T_{COMFORT}\ (^oC) = 0.31T_{RM} + 18.8 \quad (6)$$

$$T_{MAX}(ASHRAE\ 80\%)(^oC) = 0.31T_{RM} + 21.3 \quad (7)$$

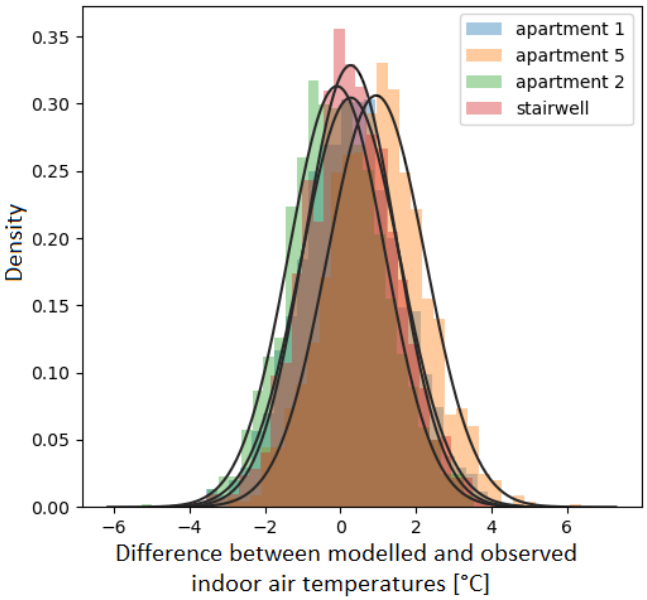


Figure 7: Frequency distribution of temperature differences between modelled and observed indoor air temperatures after calibration

$$T_{MAX}(ASHRAE\ 90\%)(^oC) = 0.31T_{RM} + 20.3 \quad (8)$$

Discussions and result analysis

As can be seen in figure 6, openness ratio of windows in the apartments ,when shading factor is in the range of 0 to 20%, does not significantly affect the indoor temperature in the stairwell because the doors that connect apartments to stairwell are most of the time closed. Nonetheless, within this shading range, best fit between observed and modelled temperatures in apartment 1 was achieved when openness ratio was between 10 to 15% and for apartment 2 between 15 to 20%. As for apartment 5, modelled temperature best fits the observed temperatures when the window was fully closed. This could mean that the occupant may have opened the window but closed the shading at the same time reducing the air inflow through the window. It is also important to note that not all occupants behave in a similar pattern. The openness ratio and shading factor can be different for each window in each apartment at different times of the day.
As shown in table 2, MAE and RMSE both in apartment 1, apartment 2, apartment 5 and stairwell are below 1 or just above it indicating a relatively good fit to observed measurements. This goodness of fit is further confirmed in figure 7 where normal frequency distribution of differences between modelled and observed indoor air temperatures are centered around zero, and in figure 8 where time series data of modelled and observed indoor air temperatures in apartment 1 is presented.

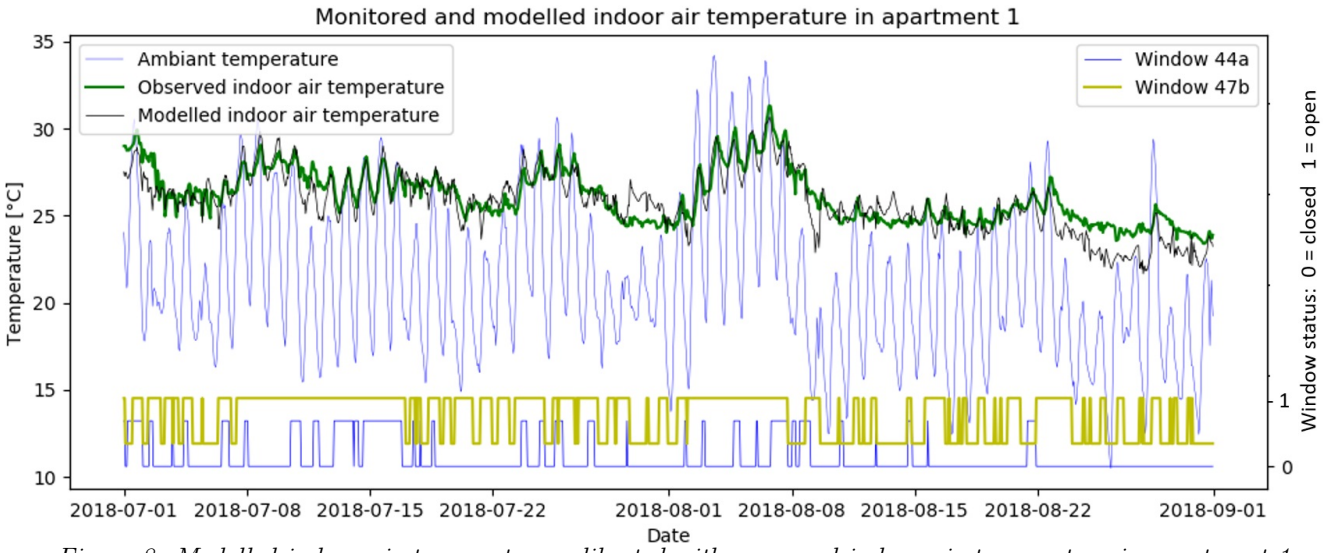


Figure 8: Modelled indoor air temperature calibrated with measured indoor air temperature in apartment 1

Returning to the subject of summer passive strategies assessment, the impact of window openness ratio and window shutters were studied. Three theoretical scenarios in addition to base case scenario, which is the calibrated model, were co-simulated and gauged against the described thermal comfort indices.

In scenario I, window shutters were set to cover 90% of total area of window during the day from 09:00 to 21:00 with base case window openness ratio. In scenario II, openness ratio of window was increased to 100% but window shutters were assumed to be fully open during day and night. In scenario III, window openness ratio was set to 100% and window shutters covered 90% during the day.

For demonstration, modelled indoor air temperature in base case scenario of apartment 1 is depicted in Figure 9 with upper boundary limits of EN 16798 adaptive thermal comfort indices, which are presented in grey, ASHRAE 55 in red, optimal comfort temperature with dashed line, and one lower boundary limit for category II of EN 16798.

Results of analyses in calibrated scenario showed that indoor air temperature did not exceed the maximum allowable operative temperature limit of category III of EN 16798 in any apartment. However, number of hours that indoor temperature exceeded category II of EN 16798 was 3 and 2% of total hours in apartment 1 and 5 respectively. These numbers reached to 9%, 6%, 13% and 11% of total hours in category I of EN 16798 in apartment 1, apartment 2, apartment 5, and stairwell respectively. Similarly, number of hours that indoor operative temperatures exceeded upper limit of 80% acceptability in ASHRAE 55 were 9%, 6%, 12% and 10% of total hours in apartment 1, apartment 2, apartment 5, and stairwell. With 90% acceptability in ASHRAE 55 this number jumped to 22%, 20%, 30% and 44% of total number of hours in the above-mentioned apartments respectively. Number of hours here indicate the frequency of times when operative indoor temperature exceeded the upper limit but does not indicate the intensity of it. To consider both, the exceeding temperature differences between modelled indoor operative temperature and upper limit of thermal comfort index, in every hour was summed up (figure 10a).

Further, analysis of calibrated scenario with PMV comfort index demonstrated that occupants felt comfortable less than 50% of the time in apartment1, apartment2, and apartment5; except in stairwell where this percentage was approximately 60%. largest percentage of discomfort here was due to temperature drop at night. only 2.3%, 3.5%, 2.8%, and 2.1% of discomfort in apartment1, apartment2, apartment5, and stairwell respectively, were due to over temperature in the day. As can be seen in figure 10a all three apartments and stairwell display most im-

Table 2: Goodness of fit between modelled and observed temperatures after calibrations

Modelled and monitored site	MAE $[^oC]$	RMSE $[^oC]$	Mean indoor temperature $[^oC]$	CV RMSE	Limit value by ASHRAE Guideline 14-2014
Apartment 1	0.72	0.91	26.12	3.5%	
Apartment 2	0.66	0.78	25.99	3%	CV RMSE ≤ 30 %
Apartment 5	1.06	1.29	27.2	4.75%	
Stairwell	0.62	0.78	26.91	2.9%	

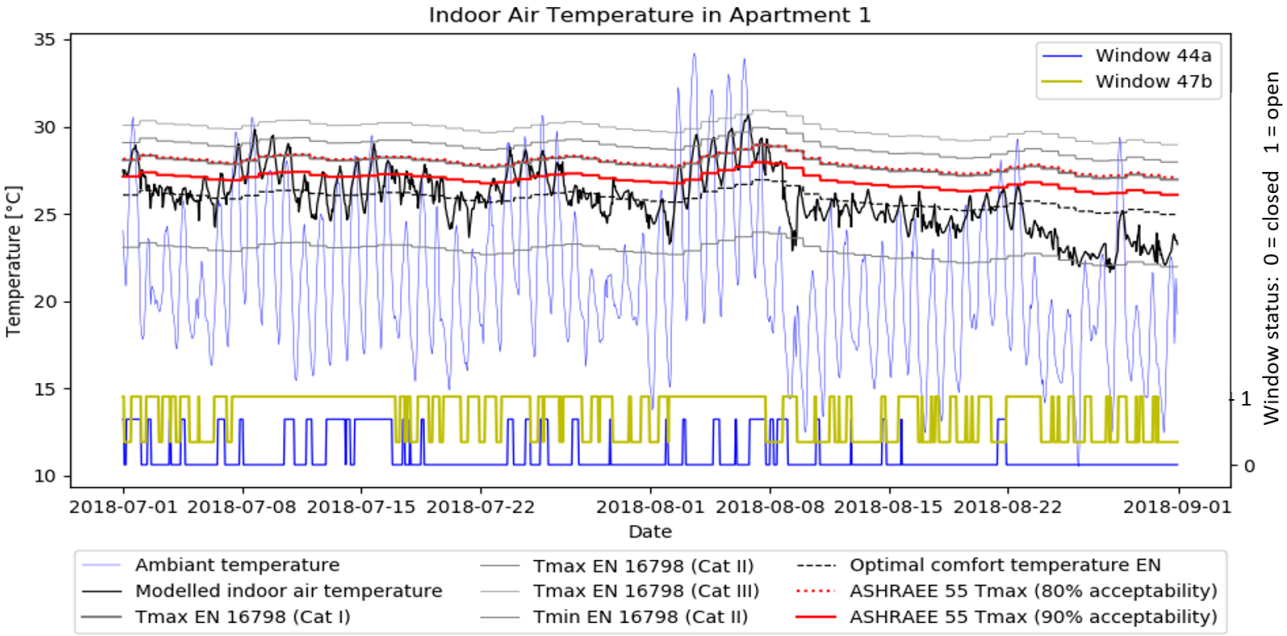


Figure 9: Calibrated model time series data with adaptive indoor thermal comfort indices in apartment 1

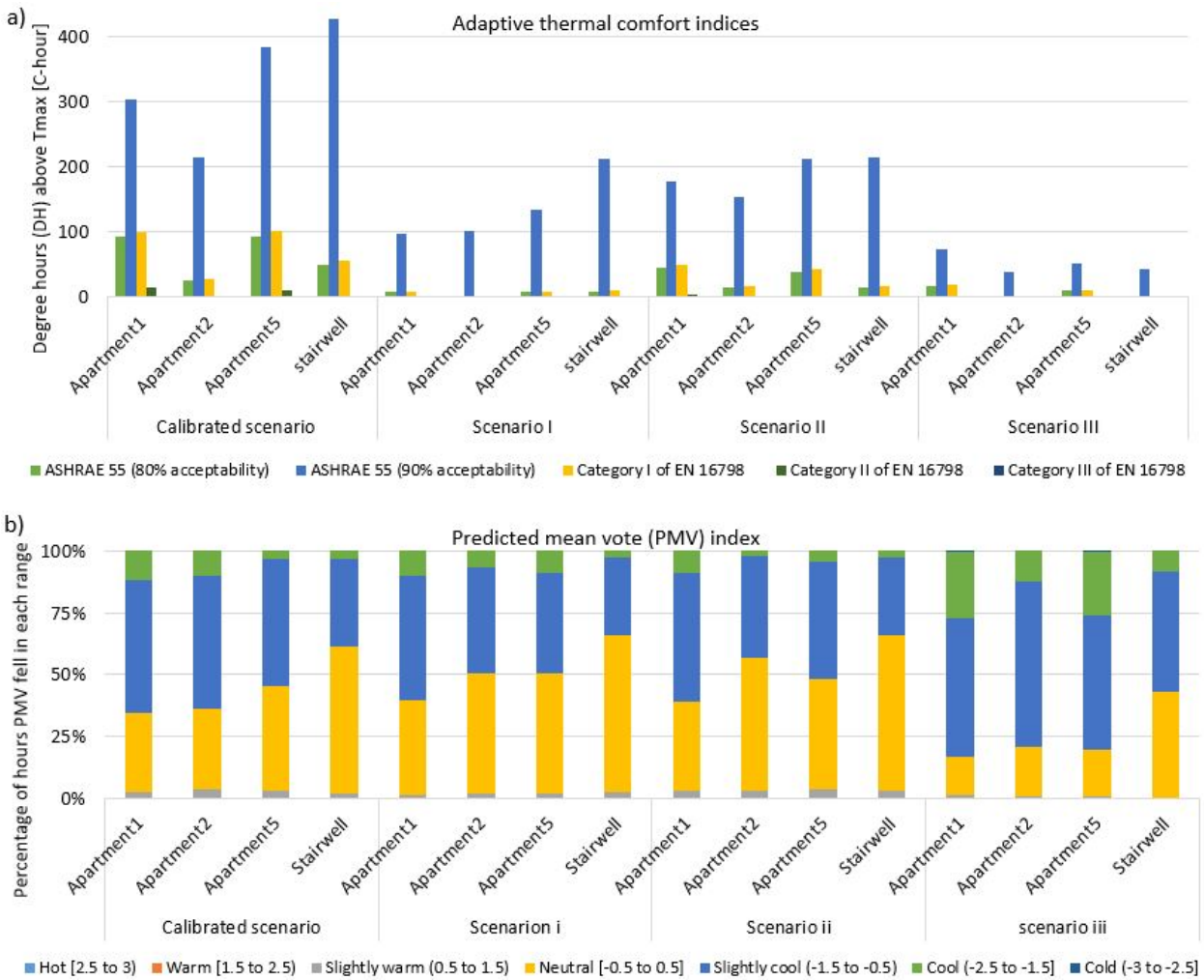


Figure 10: a, sum of degree hours when operative temperature exceeded the upper limit boundary of adaptive comfort indices in EN 16798 and ASHRAE 55; b, percentage of hours PMV index rate fell in each range

provement in scenario i and iii if measured by adaptive comfort indices; but if measured with PMV index, scenario i and ii provide better overall comfort to occupants (figure 10b). However, percentage of discomfort due to over temperature in the day increases by approximately 1% in all apartments and stairwell in scenario ii when measured by PMV index compared to base case scenario . This puts scenario ii at a worse position than base case scenario in terms of summer diurnal over temperature performance.

Conclusions

Passive strategies in the context of buildings are well known in engineering as effective adaptive techniques to achieve an objective with little to no additional input. In the present study, the effect of window shutters/blinds that reduce solar gains and window openness ratios that control natural airflow to the apartments were investigated by dynamic co-simulations of various scenarios. Results of dynamic co-simulation and comparison of scenarios with base case scenario show that excessive passive ventilation achieved by means of increase in window openness ratio brings down the peak indoor air temperature during the day but it can also cause discomfort at night. Similarly, limiting solar gains by window shutters can significantly reduce peak indoor air temperature during the day and in the meantime does not cause notable temperature drop at night; however, it can drastically limit natural visual comfort. Study on this apartment building ,which showed evidence of overheating in calibrated scenario, demonstrates that risks associated with indoor over temperature in it could have been considerably reduced with passive strategies without mechanical cooling. As an important outcome of this study for existing buildings, a good synergy between passive natural ventilation and passive measures that limit solar gains should be found with dynamic co-simulations to optimize thermal comfort. It is also necessary to mention that even with best synergies there are limits to what can be achieved by passive methods; for instance, in high demanding comfort level such as ASHRAE 55 with 90% acceptability.
Further work on this should delve into the role of different residential building typologies on summer thermal comfort; future weather scenarios with and without heatwaves; and geographical location of building in the city to take into account urban heat island effect as well.

Acknowledgement

The authors acknowledge the support of CEREMA (Centre d'études et d'expertise sur les risques, l'environnement, la mobilité et l'aménagement) and the support by the CPER 2015-2020 and the action S2EPdL in the framework of the urban observatory of Nantes that funded in-situ measurements.

References

(2017). *ANSI/ASHRAE Standard 55-2017. Thermal Environmental Conditions for Human Occupancy; Conditions That Provide Thermal Comfort; Page 14. ISSN 1041-2336*.

Chirico, F. and N. Magnavita (2019, April). New and Old Indices for Evaluating Heat Stress in an Indoor Environment: Some Considerations. Comment on Kownacki, L.; Gao, C.; Kuklane, K.; Wierzbicka, A. Heat Stress in Indoor Environments of Scandinavian Urban Areas: A Literature Review. Int. J. Environ. Res. Public Health 2019, 16 (4), 560. doi:10.3390/ijerph16040560. *International Journal of Environmental Research and Public Health 16*(8), 1444.

Dols, W. S., S. J. Emmerich, and B. J. Polidoro (2016, March). Using Coupled Energy, Airflow and IAQ Software (TRNSYS/CONTAM) to Evaluate Building Ventilation Strategies. *Building services engineering research & technology : BSER & T 37*(2), 163–175.

National Institute of Standards and Technology (2015, September). *CONTAM User Guide and Program Documentation Version 3.2*.

Gamero-Salinas, J. C., A. Monge-Barrio, and A. Sánchez-Ostiz (2020, March). Overheating risk assessment of different dwellings during the hottest season of a warm tropical climate. *Building and Environment 171*, 106664.

Khalifa, I., L. Gharbi Ernez, E. Znouda, and C. Bouden (2015, July). Coupling TRNSYS 17 and CONTAM: simulation of a naturally ventilated double-skin façade. *Advances in Building Energy Research 9*(2), 293–304.

Ozarisoy, B. and H. Elsharkawy (2019). Assessing overheating risk and thermal comfort in state-of-the-art prototype houses that combat exacerbated climate change in uk. *Energy and Buildings 187*, 201–217.

Rodler, A., S. Guernouti, M. Musy, and J. Bouyer (2018, June). Thermal behaviour of a building in its environment: Modelling, experimentation, and comparison. *Energy and Buildings 168*, 19–34.

Tartarini, F. and S. Schiavon (2020, July). pythermalcomfort: A Python package for thermal comfort research. *SoftwareX 12*, 100578.

Taveres-Cachat, E. and F. Goia (2020). Co-simulation and validation of the performance of a highly flexible parametric model of an external shading system. *Building and Environment 182*, 107111.

(2017). *TRNSYS - Multizone Building modeling with Type56 and TRNBuild Manual*.

Appendix 3-1:

- Accessing and post-processing historical weather data from MeteoFrance and transforming it into EPW file format

In [1]:
```
import urllib
import zlib
import os
import pandas as pd
import numpy as np
from datetime import date, datetime, timedelta
%config Completer.use_jedi = False
```

In [2]:
```
pd.set_option("display.max_columns", None)
```

In [3]:
```
import gzip
```

One year of monthly weather data was downloaded in the folder "ObservedData" for 2003 from Meteo France Archives of in situ observations.

Finding Your Weather staion

In [4]:
```
## list of french weather stations file is in github
link=r'D:\Weather_files\Weather data for South of France\EssentialWeatherStationsList.csv'
```

In [5]:
```
FrenchWeatherStations = pd.read_csv(link,sep=';')
FrenchWeatherStations.tail(4)
```

Out[5]:

	ID	Nom	Latitude	Longitude	Altitude
58	81405	CAYENNE-MATOURY	4.822333	-52.365333	4
59	81408	SAINT GEORGES	3.890667	-51.804667	6
60	81415	MARIPASOULA	3.640167	-54.028333	106
61	89642	DUMONT D'URVILLE	-66.663167	140.001000	43

In [2]:
```
FrenchWeatherStations.head(5)
```

In [7]:
```
from math import sin, cos, sqrt, atan2, radians

def calculate_distance(lat1, lon1, lat2, lon2):
    # approximate radius of earth in km
    R = 6373.0

    lat1 = radians(lat1)
    lon1 = radians(lon1)
    lat2 = radians(lat2)
    lon2 = radians(lon2)

    dlon = lon2 - lon1
    dlat = lat2 - lat1

    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))

    distance = R * c
    return distance
```

In [134…
```
#Coordinates of target location
location_latitude=43.12358
location_longitude=6.136299

distances=[]

for i in range(len(FrenchWeatherStations)):
    lats = FrenchWeatherStations['Latitude'][i]
    lons = FrenchWeatherStations['Longitude'][i]
    distance = calculate_distance(location_latitude,location_longitude,lats,lons)
    distances.append(distance)

min_value=min(distances)
index_min=distances.index(min_value)
ID_min=FrenchWeatherStations['ID'][index_min]
Name=FrenchWeatherStations['Nom'][index_min]
```

```
print('Name of closest weather station is {}'.format(Name))
print('ID of closest weather station is {} and its distance is {:.2f} km'.format(ID_min,min_value))
```

```
Name of closest weather station is CAP CEPET
ID of closest weather station is 7661 and its distance is 16.62 km
```

Decompressing and filtering for ID_min .gz downloaded grided weatherfiles

In [135…

```
## Observed weather data from meteoFrance archives
##
cwd = os.path.abspath(r'D:\Weather_files\ObservedData2003')
files = os.listdir(cwd)
```

In [1]:

```
df = pd.DataFrame()
for file in files:
    if file.endswith('.gz'):
        f=gzip.open('D:/Weather_files/ObservedData2003/'+str(file), 'rb')
        df_W=pd.read_csv(f,delimiter=';')
        df_output = df_W[df_W.numer_sta == ID_min] # weather station ID found earlier
        df_output.loc[:,"date"] = pd.to_datetime(df_output["date"], format = "%Y%m%d%H%M%S")
        df_output.index = df_output["date"]
        df_output=df_output.drop(["date", "numer_sta"], axis = 1)
        df_output.to_csv('D:/Weather_files/ObservedData2003/'+str(file)+".csv")
        df = df.append(df_output)
```

Out[1]:

```
'\ndf = pd.DataFrame()\nfor file in files:\n    if file.endswith(\'.gz\'):\n        f=gzip.open(\'D:/Weather_file
s/ObservedData2003/\'+str(file), \'rb\')\n        df_W=pd.read_csv(f,delimiter=\';\')\n        df_output = df_W[d
f_W.numer_sta == ID_min] # weather station ID found earlier\n        df_output.loc[:,"date"] = pd.to_datetime(df_
output["date"], format = "%Y%m%d%H%M%S")\n        df_output.index = df_output["date"]\n        df_output=df_outpu
t.drop(["date", "numer_sta"], axis = 1)\n        df_output.to_csv(\'D:/Weather_files/ObservedData2003/\'+str(file
)+".csv")\n        df = df.append(df_output)\n'
```

Transforming three hour weather data to hourly and creating an epw weather file from the given data

In [137…

```
#df.head()
```

Parameters of weather file from MeteoFrance archives

In [138…

```
#'date'       :Datetime
#'numer_sta'  :number of weather station (Nantes = 7222)
#'pmer'     :PRESSURE AT SEA LEVEL (Pression au niveau mer)[pa]
#'tend '    :presssure variations in every 3 hours(Variation de pression en 3 heures)[pa]
#'cod_tend ':Type de tendance barométrique [code (0200)]
#'dd'       :WIND DIRECTION (Direction du vent moyen 10 mn [degrees])
#'ff'       :WIND SPEED (Vitesse du vent moyen 10 mn [m/s])
#'t'        :DRY BULB TEMPERATURE (Température [K])
#'td'       :DEW POINT TEMPERATURE (Point de rosée [K])
#'u',        :REATIVE HUMIDITY (Humidité [%])
#'vv',       :HORIZONTAL VISBILITY (Visibilité horizontale [m])
#'ww',       :Temps présent [code (4677)]
#'w1',       :Temps passé 1 [code (4561)]
#'w2',       :Temps passé 2 [code (4561)]
#'n',        :TOTAL CLOUD COVER (Nebulosité totale [%])
#'nbas',     :OPAQUE SKY COVER measure (1/10) (Nébulosité des nuages de l'étage inférieur [octa] measured (1/8))
#'hbas',     :Hauteur de la base des nuages de l'étage inférieur [m]
#'cl',       :Type des nuages de l'étage inférieur
#'cm',       :Type des nuages de l'étage moyen
#'ch',       :Type des nuages de l'étage supérieur
#'pres',     :ATMOSPHERIC STATION PRESSURE (Pression station [pa])
#'niv_bar',  :Niveau barométrique [pa]
#'geop',     :Géopotentiel [m2/s2]
#'tend24',   :Variation de pression en 24 heures [pa]
#'tn12',     :Température minimale sur 12 heures [K]
#'tn24',     :Température minimale sur 24 heures [K]
#'tx12',     :Température maximale sur 12 heures [K]
#'tx24',     :Température maximale sur 24 heures [K]
#'tminsol',  :Température minimale du sol sur 12 heures [K]
#'sw',       :Méthode mesure tw
#'tw',       :Wet thermometer temperature (Température du thermomètre mouillé [K])
#'raf10',    :Rafales sur les 10 dernières minutes [m/s]
#'rafper',   :Rafales sur une période [m/s]
```

```
#'per',      :Période de mesure de la rafale [min]
#'etat_sol', :Etat du sol
#'ht_neige', :Hauteur totale de la couche de neige, glace, autre au sol [m]
#'ssfrai',   :Hauteur de la neige fraîche [m]
#'perssfrai',:Periode de mesure de la neige fraiche [1/10 hour]
#'rr1',      :LIQUID PRECIPITATION DEPTH (Précipitations dans les 1 dernières heures [mm])
#'rr3',      : =
#'rr6',      : =
#'rr12',     : =
#'rr24',     : =
#'phenspe1', :Phénomène spécial 1 [code (3778)]
#'phenspe2', : =
#'phenspe3', : =
#'phenspe4', : =
#'nnuage1',  :Nébulosité cche nuageuse N [octa]
#'ctype1',   :Type nuage N [code (0500)]
#'hnuage1',  :Hauteur de base N [m]
#'nnuage2',  : =
#'ctype2',   : =
#'hnuage2',  : =
#'nnuage3',  : =
#'ctype3',   : =
#'hnuage3',  : =
#'nnuage4',  : =
#'ctype4',   : =
#'hnuage4',  : =
#'Unnamed: 59':
```

In [139…

```
## replace mq to NaN
df = df.replace('mq', np.nan, regex=True)
```

In [140…

```
df=df.resample('H').last()# resampling doesn't go past the last index.
                          #Parameters of weather file from MeteoFrance archives
```

In [141…

```
df['t'] = df.t.astype(float)
df['td'] = df.td.astype(float)
df['u'] = df.u.astype(float)
df['pres'] = df.pres.astype(float)
df['n'] = df.n.astype(float)
df['dd'] = df.dd.astype(float)
df['ff'] = df.ff.astype(float)
```

In [142…

```
##Interpolation of 3hr data to hourly weather data
#dry bulb temperature
df['t'].interpolate(method='polynomial', order=7,limit_direction='both',limit=2, inplace=True)
# dew point temperature
df['td'].interpolate(method='polynomial', order=7,limit_direction='both',limit=2, inplace=True)
# relative humidity
df['u'].interpolate(method='linear', limit_direction='both',limit=2, inplace=True)
# Atmospheric pressure
df['pres'].interpolate(method='linear', limit_direction='both',limit=2, inplace=True)
# Cloud cover /total /opaque SKY cover
df['n'].interpolate(method='linear', limit_direction='both',limit=2, inplace=True)
## Wind direction
df['dd'].interpolate(method='linear', limit_direction='both',limit=2, inplace=True)
## Wind speed
df['ff'].interpolate(method='linear', limit_direction='both',limit=2, inplace=True)
```

In [143…

```
## year, month, day, hour to dataframe from index
df['Year']=df.index.year
df['Month']=df.index.month
df['Day']=df.index.day
df['Hour']=df.index.hour
```

In [144…

```
# TOtal cloud cover is measured in 1/10
df['TotalCloudCover']=round(df['n']//10,0)
# Opaque cloud cover is measured in 1/9
df['OpaqueCloudCover']=round(df['n']//11.2,0)
```

In [145…

```
#df.head()
```

In [146…

```
df.columns
```

Out[146

```
Index(['pmer', 'tend', 'cod_tend', 'dd', 'ff', 't', 'td', 'u', 'vv', 'ww',
```

```
       'w1', 'w2', 'n', 'nbas', 'hbas', 'cl', 'cm', 'ch', 'pres', 'niv_bar',
       'geop', 'tend24', 'tn12', 'tn24', 'tx12', 'tx24', 'tminsol', 'sw', 'tw',
       'raf10', 'rafper', 'per', 'etat_sol', 'ht_neige', 'ssfrai', 'perssfrai',
       'rr1', 'rr3', 'rr6', 'rr12', 'rr24', 'phenspe1', 'phenspe2', 'phenspe3',
       'phenspe4', 'nnuage1', 'ctype1', 'hnuage1', 'nnuage2', 'ctype2',
       'hnuage2', 'nnuage3', 'ctype3', 'hnuage3', 'nnuage4', 'ctype4',
       'hnuage4', 'Unnamed: 59', 'Year', 'Month', 'Day', 'Hour',
       'TotalCloudCover', 'OpaqueCloudCover'],
      dtype='object')
```

In [147…

```
##Rearranging columns and selecting only a few that we use in modification of weather file
Columns=['Year', 'Month', 'Day', 'Hour','dd', 'ff', 't', 'td','u','n',
         'pres','TotalCloudCover', 'OpaqueCloudCover']
df=df[Columns]
#  year is not a leap year, if you dealing with a leap year, in order to make it fit with typical
#  weather file, the leap day needs to be removed. (typical weather years are not leap years)
# Adding two more hours to the dataframe
df.reset_index(inplace=True)
```

In [148…

```
df2=pd.DataFrame({'Day':[31.0,31.5],
                  'Hour':[22.0,23.0],
                  'Month':[12.0,12.0]})
df=df.append(df2)
```

```
<ipython-input-148-742a748fe238>:4: FutureWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  df=df.append(df2)
```

In [149…

```
df.reset_index(drop=True,inplace=True)
df.tail()
```

Out[149…

	date	Year	Month	Day	Hour	dd	ff	t	td	u	n	pres	TotalCloudCover	OpaqueCloudCover
8755	2003-12-31 19:00:00	2003.0	12.0	31.0	19.0	310.0	3.1	281.024433	273.464855	69.666667	40.0	99540.0	4.0	3.0
8756	2003-12-31 20:00:00	2003.0	12.0	31.0	20.0	320.0	3.1	281.568181	273.479513	72.333333	40.0	99590.0	4.0	3.0
8757	2003-12-31 21:00:00	2003.0	12.0	31.0	21.0	330.0	3.1	278.650000	274.550000	75.000000	40.0	99640.0	4.0	3.0
8758	NaT	NaN	12.0	31.0	22.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8759	NaT	NaN	12.0	31.5	23.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Writing data from dataframe to EPW weather file for 2003

In [150…

```
df2=df
```

In [151…

```
## missing data are filled with automatic closest neighbor
df2=df2.fillna(method="ffill")
```

In [152…

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8760 entries, 0 to 8759
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   date              8760 non-null   datetime64[ns]
 1   Year              8760 non-null   float64
 2   Month             8760 non-null   float64
 3   Day               8760 non-null   float64
 4   Hour              8760 non-null   float64
 5   dd                8760 non-null   float64
 6   ff                8760 non-null   float64
 7   t                 8760 non-null   float64
 8   td                8760 non-null   float64
```

```
 9   u                 8760 non-null   float64
 10  n                 8759 non-null   float64
 11  pres              8760 non-null   float64
 12  TotalCloudCover   8759 non-null   float64
 13  OpaqueCloudCover  8759 non-null   float64
dtypes: datetime64[ns](1), float64(13)
memory usage: 958.2 KB
```

In [153…

```
df2.head(2)
```

Out[153…

	date	Year	Month	Day	Hour	dd	ff	t	td	u	n	pres	TotalCloudCover	OpaqueCloudCove
0	2003-01-01 00:00:00	2003.0	1.0	1.0	0.0	300.0	10.800000	284.550000	276.550000	58.000000	NaN	99420.0	NaN	Nal
1	2003-01-01 01:00:00	2003.0	1.0	1.0	1.0	300.0	9.933333	284.529095	274.429522	57.666667	0.0	99480.0	0.0	0.(

In [154…

```
#df2['TotalCloudCover'] = df2.TotalCloudCover.astype(int)
#df2['OpaqueCloudCover']=df2.OpaqueCloudCover.astype(int)
df2[['Day','Hour','Month','Year']] = df2[['Day','Hour','Month','Year']].astype(int)
df2['WindDirection']=df2['dd'].astype(int)
df2['DryBulbTemp']=df2['t']-273.15
df2['DewPointTemp']=df2['td']-273.15
```

EnergyPLus typical weather file will be overwritten with new data (temperature, RH, pressure, wind speed,..etc)

In [155…

```
## Writing the weather data on epw weather template
from ladybug.epw import EPW
from pandas import DataFrame
from collections import OrderedDict
%config Completer.use_jedi = False
```

In [156…

```
## Checking epw weather file data
epwFile=EPW(r'D:\Weather_files\Weather data for South of France\Hyeres-interpolated.epw')
epwDataList=epwFile.to_dict()['data_collections']
epwDataDict = OrderedDict()

for dataColumns in epwDataList:
    dataName=dataColumns['header']['data_type']['name']
    epwDataDict[dataName]=dataColumns['values']

epwDataFrame = DataFrame(epwDataDict)
epwDataFrame.head(3)
```

Out[156…

	Year	Month	Day	Hour	Minute	Uncertainty Flags	Dry Bulb Temperature	Dew Point Temperature	Relative Humidity	Atmospheric Station Pressure	Extraterrestrial Horizontal Radiation	Extraterrestrial Direct Normal Radiation	Horizontal Infrared Radiation Intensity
0	2005	12	31	24	60	*?*?*?*?*?*?*?*?*?*?*?*?*?*?*?*?*?*?	7.4	3.5	76	101102	0	0	285
1	2005	1	1	1	60	*?*?*?*?*?*?*?*?*?*?*?*?*?*?*?*?*?*?	14.1	7.7	65	101102	0	0	311
2	2005	1	1	2	60	*?*?*?*?*?*?*?*?*?*?*?*?*?*?*?*?*?*?	13.4	7.7	68	101102	0	0	310

In [157…

```
# Name and indices of columns in a typical epw weather file
#field_number: index value between 0 to 34 for different available epw fields.
#            0 Year
#            1 Month
#            2 Day
#            3 Hour
#            4 Minute
```

```
#            6 Dry Bulb Temperature
#            7 Dew Point Temperature
#            8 Relative Humidity
#            9 Atmospheric Station Pressure
#            10 Extraterrestrial Horizontal Radiation
#            11 Extraterrestrial Direct Normal Radiation
#            12 Horizontal Infrared Radiation Intensity
#            13 Global Horizontal Radiation
#            14 Direct Normal Radiation
#            15 Diffuse Horizontal Radiation
#            16 Global Horizontal Illuminance
#            17 Direct Normal Illuminance
#            18 Diffuse Horizontal Illuminance
#            19 Zenith Luminance
#            20 Wind Direction
#            21 Wind Speed
#            22 Total Sky Cover
#            23 Opaque Sky Cover
#            24 Visibility
#            25 Ceiling Height
#            26 Present Weather Observation
#            27 Present Weather Codes
#            28 Precipitable Water
#            29 Aerosol Optical Depth
#            30 Snow Depth
#            31 Days Since Last Snowfall
#            32 Albedo
#            33 Liquid Precipitation Depth
#            34 Liquid Precipitation Quantity
```

In [158…

```
def changeEPWData(oldEpwFilePath,newEpwFilePath,dataIndex,dataList):
    with open(oldEpwFilePath) as oldStream,open(newEpwFilePath,"w") as newStream:
        numCount=0
        for idx,lines in enumerate(oldStream):
            if lines.strip():
                try:
                    lineSplit=lines.strip().split(",")
                    dataTest=float(lineSplit[0])
                    lineSplit[dataIndex]=str(dataList[numCount])
                    data=",".join(lineSplit)
                    newStream.write(data+"\n")
                    numCount+=1
                except ValueError:
                    newStream.write(lines.strip()+"\n")
            else:
                newStream.write(lines)
    return newEpwFilePath
```

In [159…

```
#Writing column by column of weather file to make sure the structure of EPW file doesn't change
#a user can decide to modify one climate variable or multiples climate variables
```

In [160…

```
# In this script a temporary folder was created to save modified versions of EPW weather file.
# With every new climate variable, previously modified file from temporary folder is selected and
# further overwritten until all climate variables are overwritten.
```

In [161…

```
### Replacing year
changeEPWData(r'D:\Weather_files\Weather data for South of France\Hyeres-interpolated.epw',
              r'D:\Weather_files\ObservedData2003\Temporary\year.epw',
              dataIndex=0,
              dataList=df2.Year)

#changeEPWData(r'D:\Weather_files\ObservedData\Temporary\year.epw',
#              r'D:\Weather_files\ObservedData\Temporary\month.epw',
#              dataIndex=1,
#              dataList=df2.Month)
#changeEPWData(r'D:\Weather_files\ObservedData\Temporary\month.epw',
#              r'D:\Weather_files\ObservedData\Temporary\day.epw',
#              dataIndex=2,
#              dataList=df2.Day)
#changeEPWData(r'D:\Weather_files\ObservedData\Temporary\day.epw',
#              r'D:\Weather_files\ObservedData\Temporary\hour.epw',
#              dataIndex=3,
#              dataList=df2.Hour)
# Substituting DBT
changeEPWData(r'D:\Weather_files\ObservedData2003\Temporary\year.epw',
              r'D:\Weather_files\ObservedData2003\Temporary\DryBTemp.epw',
              dataIndex=6,
              dataList=df2.DryBulbTemp)
```

```python
# Substituting DewPoint
changeEPWData(r'D:\Weather_files\ObservedData2003\Temporary\DryBTemp.epw',
              r'D:\Weather_files\ObservedData2003\Temporary\DewPTemp.epw',
              dataIndex=7,
              dataList=df2.DewPointTemp)
#Substitutinh RH
changeEPWData(r'D:\Weather_files\ObservedData2003\Temporary\DewPTemp.epw',
              r'D:\Weather_files\ObservedData2003\Temporary\RH.epw',
              dataIndex=8,
              dataList=df2.u)
#Substituting Atm pressure
changeEPWData(r'D:\Weather_files\ObservedData2003\Temporary\RH.epw',
              r'D:\Weather_files\ObservedData2003\Temporary\AtmPressure.epw',
              dataIndex=9,
              dataList=df2.pres)

## Replacing wind direction
changeEPWData(r'D:\Weather_files\ObservedData2003\Temporary\AtmPressure.epw',
              r'D:\Weather_files\ObservedData2003\Temporary\WindDirection.epw',
              dataIndex=20,
              dataList=df2.WindDirection)

## Replacing wind speed
changeEPWData(r'D:\Weather_files\ObservedData2003\Temporary\WindDirection.epw',
              r'D:\Weather_files\Weather data for South of France\Hyeres_2003.epw',
              dataIndex=21,
              dataList=df2.ff)
```

Out[161… 'D:\\Weather_files\\Weather data for South of France\\Hyeres_2003.epw'

Verifying how it looks

In [162…

```python
# Checking how it looks now
epwFile=EPW(r'D:\Weather_files\Weather data for South of France\Hyeres_2003.epw')
epwDataList=epwFile.to_dict()['data_collections']
epwDataDict = OrderedDict()

for dataColumns in epwDataList:
    dataName=dataColumns['header']['data_type']['name']
    epwDataDict[dataName]=dataColumns['values']

epwDataFrame1 = DataFrame(epwDataDict)
epwDataFrame1.head()
```

Out[162…

	Year	Month	Day	Hour	Minute	Uncertainty Flags	Dry Bulb Temperature	Dew Point Temperature	Relative Humidity	Atmospheric Station Pressure	Extraterrestrial Horizontal Radiation	Extraterrestrial Direct Normal Radiation	Horizontal Infrared Radiation Intensity
0	2003	12	31	24	60	*?	5.500000	1.400000	75	99420	0	0	285
1	2003	1	1	1	60	*?	11.400000	3.400000	58	99480	0	0	311
2	2003	1	1	2	60	*?	11.379095	1.279522	58	99540	0	0	310
3	2003	1	1	3	60	*?	11.527669	2.032684	57	99600	0	0	307
4	2003	1	1	4	60	*?	11.400000	3.200000	57	99627	0	0	306

In [ ]:

Loading [MathJax]/extensions/Safe.js

Appendix 3-2:

- Transforming rlat/rlon to lat/lon and vice versa in python to process netCDF files

In [28]:
```
from math import *
import geopy.distance
import pyproj
from pyproj import Transformer
%config Completer.use_jedi = False
```

Transforming lat/lon to rlat/rlon and vice versa

Properties of NetCDF coordinates downloaded from CORDEX domain.

In [ ]:
```
## Grid north pole latitude and longitude to grid south pole latitude and longitude
##|S1 rotated_pole()
##    grid_mapping_name: rotated_latitude_longitude
##    grid_north_pole_latitude: 39.25
##    grid_north_pole_longitude: -162.0
```

From properties it can be seen that the position of the rotated_pole is given in north_Pole geographic coordinates. We first transform grid_north_pole_latitude/longitude to grid_south_pole_latitude/longitude.

In [ ]:
```
# SP_lat = - NP_lat     # Grid_South_pole_latitude for this case becomes -39.25
# SP_Lon = NP_lon -180  # Grid South Pole longitude for this case becomes -162.0 - 180 = -342 = 18
```

In [ ]:
```
##The function below transforms lat/lon to rlat/rlon and vice versa
#if option =1, it transforms lat/lon to rlat/rlon # else, it transforms rlat/rlon to lat/lon
```

In [29]:
```
def rotated_grid_transform(grid_in, option, SP_coor):
    lon = grid_in[0]
    lat = grid_in[1];

    lon = (lon*pi)/180; # Convert degrees to radians
    lat = (lat*pi)/180;
    SP_lon = SP_coor[0];
    SP_lat = SP_coor[1];

    theta = 90+SP_lat; # Rotation around y-axis
    phi = SP_lon; # Rotation around z-axis

    theta = (theta*pi)/180;
    phi = (phi*pi)/180; # Convert degrees to radians

    x = cos(lon)*cos(lat); # Convert from spherical to cartesian coordinates
    y = sin(lon)*cos(lat);
    z = sin(lat);

    if option == 1: # Regular -> Rotated
        x_new = cos(theta)*cos(phi)*x + cos(theta)*sin(phi)*y + sin(theta)*z;
        y_new = -sin(phi)*x + cos(phi)*y;
        z_new = -sin(theta)*cos(phi)*x - sin(theta)*sin(phi)*y + cos(theta)*z;

    else:  # Rotated -> Regular
        phi = -phi;
        theta = -theta;
        x_new = cos(theta)*cos(phi)*x + sin(phi)*y + sin(theta)*cos(phi)*z;
        y_new = -cos(theta)*sin(phi)*x + cos(phi)*y - sin(theta)*sin(phi)*z;
        z_new = -sin(theta)*x + cos(theta)*z;

    lon_new = atan2(y_new,x_new); # Convert cartesian back to spherical coordinates
    lat_new = asin(z_new);
    lon_new = (lon_new*180)/pi; # Convert radians back to degrees
    lat_new = (lat_new*180)/pi;

    print (lon_new,lat_new)
```

In [34]:
```
# Geographic coordinates of Nantes (-1.553621,47.21725)
rotated_grid_transform((-1.553621,47.21725), 1, (-342,-39.25))
```

```
-13.146332621887227 -1.7927858546021267
```

In [5]:
```
# verifying transformation
rotated_grid_transform((-13.146041795940194,-1.791703198608463), 2, (-342,-39.25))
```

```
-1.5536210000000008 47.218371
```

Appendix 3-3:

- Extracting yearly weather data from netCDF file format into csv for any given location

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
%config Completer.use_jedi = False
import xarray as xr
import warnings
from netCDF4 import Dataset
# creating a series of datatime
from datetime import date, datetime, timedelta
# plotting time series data
import seaborn as sns

from pandas import datetime
from matplotlib import pyplot
```

In [2]:
```
from astral import LocationInfo
from astral.sun import sun
```

Write the name of the year and change the name of folder in the directory of netCDF files

In [3]:
```
year = 2055
year2=year+1
```

Determinig the the closest point to Nantes coordinates

In [4]:
```
## Visualize netCDF file of temperature
tas1 = Dataset(r'F:\Donnees_PC\Weather_files\CORDEX_weather_data\MPI_REMO\MPI_REMO2055\tas_EUR-11_MPI-M-MPI-ESM-l
print("temperature data "+" : " + str(tas1.variables.keys()))
```

```
temperature data  : dict_keys(['time', 'rlat', 'rlon', 'rotated_latitude_longitude', 'lat', 'lon', 'lat_vertices'
, 'lon_vertices', 'height', 'tas'])
```

In [5]:
```
# Storing the coordinates (lon, lat) into variables
lats = tas1.variables['rlat'][:]
lons = tas1.variables['rlon'][:]
# Assigned location coordinates
lat_Nantes =  -1.7927858546021267    # 47.210517 ; latitude coordinates of Centre ville in rotated pole grid
lon_Nantes =  -13.146332621887227    #-1.546285  ; longitude coordinates of Centre ville in rotated pole grid
# Selecting the nearest grid point to assigned location
# storing the squared differences of
sq_diff_lat = (lats-lat_Nantes)**2
sq_diff_lon = (lons-lon_Nantes)**2

## Verifying the shape of coordinates
print("Total number of grid points along y axis(lat)"+" = " + str(len(sq_diff_lat)))
print("Total number of grid points along x axis(lon)"+" = " + str(len(sq_diff_lon)))

## Identifying the index of minimum values for lat and lon sq_differences
min_index_lat = sq_diff_lat.argmin()
min_index_lon = sq_diff_lon.argmin()

print("Index of closest data point along y axis(lat) to assigned location"+" = " + str(min_index_lat))
print("Index of closest data point along x axis(lon) to assigned location"+" = " + str(min_index_lon))
```

```
Total number of grid points along y axis(lat) = 412
Total number of grid points along x axis(lon) = 424
Index of closest data point along y axis(lat) to assigned location = 196
Index of closest data point along x axis(lon) to assigned location = 138
```

Creating a date-time dataframe

In [6]:
```
# creating a DataFrame of year, month, day and hours
def datetime_range(start,end,delta):
    current=start
    if not isinstance(delta,timedelta):
        delta = timedelta(**delta)
    while current < end:
        yield current
        current +=delta
```

In [7]:
```
# Dataframe of time and dates: each subsequent row difference is 3 hours
start = datetime(year,0o1,0o1)
end = datetime(year2,0o1,0o1)

df = pd.DataFrame({'DateTime':datetime_range(start,end,{'days':0,'hours':3})})

# Removing leap day data if year is a leap year
df = df[~((df['DateTime'].dt.month == 2) & (df['DateTime'].dt.day == 29))]
# reindex in place
df= df.reset_index()
## Drop index column
df=df.drop(['index'], axis=1)
# Visualize beginning and end of dataframe
print(df.head(2))
print(df.tail(1))
```

```
             DateTime
0 2055-01-01 00:00:00
1 2055-01-01 03:00:00
                DateTime
2919 2055-12-31 21:00:00
```

Reading NetCDF files for 6 climate variables of the closest grid point to the assigned location

In [8]:
```
# Visualize netCDF file of temperature
tas1 = Dataset(r'F:\Donnees_PC\Weather_files\CORDEX_weather_data\MPI_REMO\MPI_REMO2055\tas_EUR-11_MPI-M-MPI-ESM-L
print("temperature data "+" : " + str(tas1.variables.keys()))

# Visualize and load humidity ratio data "%"
hurs1 = Dataset(r'F:\Donnees_PC\Weather_files\CORDEX_weather_data\MPI_REMO\MPI_REMO2055\hurs_EUR-11_MPI-M-MPI-ESM
print("humidity ratio data "+" : " + str(hurs1.variables.keys()))

# Visualize and load Surface Downwelling Shortwave Radiation :units = "W m-2"; data"
rsds1 = Dataset(r'F:\Donnees_PC\Weather_files\CORDEX_weather_data\MPI_REMO\MPI_REMO2055\rsds_EUR-11_MPI-M-MPI-ESM
print("Surface Downwelling Shortwave Radiation data "+" : " + str(rsds1.variables.keys()))

# Visualize and load Near surface wind speed data :units = "m s-1"; data"
sfcWind1 = Dataset(r'F:\Donnees_PC\Weather_files\CORDEX_weather_data\MPI_REMO\MPI_REMO2055\sfcWind_EUR-11_MPI-M-M
print("Near surface wind speed data "+" : " + str(sfcWind1.variables.keys()))

# Visualize and load total cloud cover fraction :units "%"
clt1 = Dataset(r'F:\Donnees_PC\Weather_files\CORDEX_weather_data\MPI_REMO\MPI_REMO2055\clt_EUR-11_MPI-M-MPI-ESM-L
print("total cloud cover fraction data "+" : " + str(clt1.variables.keys()))

# Visualize and load atmospheric pressure :units "%"
ps1 = Dataset(r'F:\Donnees_PC\Weather_files\CORDEX_weather_data\MPI_REMO\MPI_REMO2055\ps_EUR-11_MPI-M-MPI-ESM-LR_
print("surface_air_pressure data Pa "+" : " + str(ps1.variables.keys()))
```

```
temperature data  : dict_keys(['time', 'rlat', 'rlon', 'rotated_latitude_longitude', 'lat', 'lon', 'lat_vertices'
, 'lon_vertices', 'height', 'tas'])
humidity ratio data  : dict_keys(['time', 'rlat', 'rlon', 'rotated_latitude_longitude', 'lat', 'lon', 'lat_vertic
es', 'lon_vertices', 'height', 'hurs'])
Surface Downwelling Shortwave Radiation data  : dict_keys(['time', 'time_bnds', 'rlat', 'rlon', 'rotated_latitude
_longitude', 'lat', 'lon', 'lat_vertices', 'lon_vertices', 'rsds'])
Near surface wind speed data  : dict_keys(['time', 'rlat', 'rlon', 'rotated_latitude_longitude', 'lat', 'lon', 'l
at_vertices', 'lon_vertices', 'height', 'sfcWind'])
total cloud cover fraction data  : dict_keys(['time', 'time_bnds', 'rlat', 'rlon', 'rotated_latitude_longitude',
'lat', 'lon', 'lat_vertices', 'lon_vertices', 'clt'])
surface_air_pressure data Pa  : dict_keys(['time', 'rlat', 'rlon', 'rotated_latitude_longitude', 'lat', 'lon', 'l
at_vertices', 'lon_vertices', 'ps'])
```

Appending climate variable values of the closest grid point to the Date-time dataframe

In [9]:
```
temp = tas1.variables['tas']
humidity = hurs1.variables['hurs']
radiation = rsds1.variables['rsds']
Wind = sfcWind1.variables['sfcWind']
Cloud_fract = clt1.variables['clt']
s_pressure = ps1.variables['ps']

# Calculating values of the indices that were previouosly found with arg_min
temperat =[]
humidityR =[]
radiationSh =[]
WindS =[]
Cloud =[]
```

```
S_pressure =[]

for index, row in df.iterrows():

    a = temp[index,min_index_lat,min_index_lon]-273.15
    b = humidity[index,min_index_lat,min_index_lon]
    c = radiation[index,min_index_lat,min_index_lon]
    d = Wind[index,min_index_lat,min_index_lon]
    e = Cloud_fract[index,min_index_lat,min_index_lon]
    f = s_pressure[index,min_index_lat,min_index_lon]

    temperat.append(a)
    humidityR.append(b)
    radiationSh.append(c)
    WindS.append(d)
    Cloud.append(e)
    S_pressure.append(f)

df["DB_Temp"] = temperat
df["RelativeHumidity"] = humidityR
df["GHRadiation"] = radiationSh
df["WindSpeed"] = WindS
df["CloudCover"] = Cloud
df["AtmosPressure"] = S_pressure

# Turn objects to float
df["DB_Temp"] = df.DB_Temp.astype(float)
df["RelativeHumidity"] = df.RelativeHumidity.astype(float)
df["GHRadiation"] = df.GHRadiation.astype(float)
df["WindSpeed"] = df.WindSpeed.astype(float)
df["CloudCover"] = df.CloudCover.astype(float)
df["AtmosPressure"] = df.AtmosPressure.astype(float)
```

Interpolation of 3hr data points to hourly and calculation of remaining fields of weather file.

In [10]:
```
df=df.set_index('DateTime').resample('H').last()
```

In [11]:
```
df['DB_Temp'].interpolate(method='polynomial', order=7, inplace=True)
df['RelativeHumidity'].interpolate(method='linear', limit_direction='both',limit=2, inplace=True)
df['AtmosPressure'].interpolate(method='linear', limit_direction='both',limit=2, inplace=True)
df['CloudCover'].interpolate(method='linear', limit_direction='both',limit=2, inplace=True)
df['GHRadiation'].interpolate(method='polynomial', order=7, inplace=True)
df['WindSpeed'].interpolate(method='linear', limit_direction='both',limit=2, inplace=True)
```

Estimating dew point temperature

In [12]:
```
# August–Roche–Magnus formula for dew point temperature approximation:
def get_dew_point_c(DryBulbTemp, Rel_humidity):
    """Compute the dew point in degrees Celsius
    :param DryBulbTemp: current ambient temperature in degrees Celsius
    :type DryBulbTemp: float
    :param Rel_humidity: relative humidity in %
    :type Rel_humidity: float
    :return: the dew point in degrees Celsius
    :rtype: float
    Set of values originates from the 1974 Psychrometry and Psychrometric Charts, as presented by Paroscientific,
     A = 17.27, B = 237.7 °C; for 0 °C ≤ T ≤ 60 °C (error ±0.4 °C).
    """
    A = 17.27
    B = 237.7
    alpha = ((A * DryBulbTemp) / (B + DryBulbTemp)) + np.log(Rel_humidity/100.0)
    return (B * alpha) / (A - alpha)
```

In [13]:
```
df['DewPointTemperature'] = round(get_dew_point_c(df['DB_Temp'],df['RelativeHumidity']),2)
```

Calculating Sunrise and sunset time for the assigned location

In [14]:
```
# Setting timezone for Nantes city to calculate sunrise and sunset time. Nantes follows Paris timezone
city = LocationInfo("Nantes","France","Europe/Paris",47.2105, -1.54628)
tz=city.timezone
print(tz)
print(city.latitude)
```

```
Europe/Paris
47.2105
```

In [15]:
```
import datetime
```

In [16]:
```
df=df.reset_index()
```

In [17]:
```
# Splitting year, month, day to calculate sunrise and sundown time in each day and add them as a column
df['year']=df['DateTime'].dt.year
df['month']=df['DateTime'].dt.month
df['day']=df['DateTime'].dt.day
```

In [18]:
```
sr = []
st = []
for i, row in df.iterrows():

    sries=sun(city.observer, date=datetime.date(df.year[i],df.month[i],df.day[i]),tzinfo=city.timezone)["sunrise"
    sriet=sun(city.observer, date=datetime.date(df.year[i],df.month[i],df.day[i]),tzinfo=city.timezone)["sunset"]

    sr.append(sries)
    st.append(sriet)

df["sunrise"]=sr
df["sunset"]=st
#print(s["sunset"])
```

In [19]:
```
sunsr=df.sunrise[12].tz_localize(None)
sunst=df.sunset[12].tz_localize(None)
hourtime=df.DateTime[12].tz_localize(None)

sunsr=sunsr.strftime("%m/%d/%Y, %H:%M:%S")
sunst=sunst.strftime("%m/%d/%Y, %H:%M:%S")
hourtime=hourtime.strftime("%m/%d/%Y, %H:%M:%S")

sunrisen = datetime.datetime.strptime(sunsr, '%m/%d/%Y, %H:%M:%S')
sundown = datetime.datetime.strptime(sunst, '%m/%d/%Y, %H:%M:%S')
mytime= datetime.datetime.strptime(hourtime, '%m/%d/%Y, %H:%M:%S')

if mytime<sundown and mytime>sunrisen:
    print(df.GHRadiation[12])
else:
    print(0)
```

```
211.78707885742188
```

In [20]:
```
# Replacing the interpolated values of GHI that are between sunset and sunrise to zero
NewGHI=[] #

for i, row in df.iterrows():
    sunsr=df.sunrise[i].tz_localize(None)
    sunst=df.sunset[i].tz_localize(None)
    hourtime=df.DateTime[i].tz_localize(None)

    sunsr=sunsr.strftime("%m/%d/%Y, %H:%M:%S")
    sunst=sunst.strftime("%m/%d/%Y, %H:%M:%S")
    hourtime=hourtime.strftime("%m/%d/%Y, %H:%M:%S")

    sunrisen = datetime.datetime.strptime(sunsr, '%m/%d/%Y, %H:%M:%S')
    sundown = datetime.datetime.strptime(sunst, '%m/%d/%Y, %H:%M:%S')
    mytime= datetime.datetime.strptime(hourtime, '%m/%d/%Y, %H:%M:%S')

    if mytime<sundown and mytime>sunrisen and df.GHRadiation[i]>0:
        n=df.GHRadiation[i]
    else:
        n=0

    NewGHI.append(n)
df["GHI"] = NewGHI
```

The precision of sunset/sunrise and global horizontal calculation could be improved if calculations are done at shorter time intervals. Here, time is calculated hourly.

In [21]:
```
# Drop unnecessary columns for now
df=df.drop(columns=['GHRadiation','day','sunrise','sunset'])
```

Calcuating DNI using DISC approximation

In [22]:
```
## Day of year: ordinal days of the year
newDays=[]
for i, row in df.iterrows():
    DS=df.DateTime[i].tz_localize(None)
    DS=DS.strftime("%m/%d/%Y, %H:%M:%S")
    doy= datetime.datetime.strptime(DS, '%m/%d/%Y, %H:%M:%S').timetuple().tm_yday
    newDays.append(doy)
df['OrdinalDay']= newDays
#tm_hour
```

In [24]:
```
import pvlib as pvl
import pytz
```

In [25]:
```
## a loop to calculate Solar zenith angle
ZenAngle=[]
for i, row in df.iterrows():
    Zt=df.DateTime[i].tz_localize(None)
    g=pvl.solarposition.get_solarposition(time=Zt,
                                          latitude=47.2105,# latitude of Nantes
                                          longitude=-1.54628, # Longitude of Nantes
                                          altitude=None,
                                          pressure=df.AtmosPressure[i],
                                          method='nrel_numpy',
                                          temperature=df.DB_Temp[i]).zenith.values[0]
    ZenAngle.append(round(g,2))
df['SolarZenith']= ZenAngle
```

In [26]:
```
import math
```

In [27]:
```
# Calculating Direct normal irradiance using zenith angle, ordinal days, atmospheric pressure, and airmass
dni=[]
for i, row in df.iterrows():
    p = pvl.irradiance.disc(ghi=df.GHI[i],
                        solar_zenith=df.SolarZenith[i],
                        datetime_or_doy=df.OrdinalDay[i],
                        pressure=df.AtmosPressure[i],
                        min_cos_zenith=0.065,
                        max_zenith=87,
                        max_airmass=(1/(math.cos(df.SolarZenith[i])+0.50572*(96.07995-df.SolarZenith[i])
                                        **(-1.6364))))['dni']
    # Accounting for cloud cover
    S1=p.tolist()*(1-(df.CloudCover[i])/100)
    dni.append(round(S1,2))

df['DNI']= dni
```

```
<ipython-input-27-f20766d7981d>:10: RuntimeWarning: invalid value encountered in double_scalars
  max_airmass=(1/(math.cos(df.SolarZenith[i])+0.50572*(96.07995-df.SolarZenith[i])**(-1.6364))))['dni']
C:\Users\obaidullah.yaqubi\AppData\Roaming\Python\Python38\site-packages\pvlib\irradiance.py:1418: RuntimeWarning
: overflow encountered in exp
  delta_kn = a + b * np.exp(c*am)
```

In [28]:
```
# Seeing how does the date-time dataframe looks like
df.iloc[5000:5004]
```

Out[28]:

	DateTime	DB_Temp	RelativeHumidity	WindSpeed	CloudCover	AtmosPressure	DewPointTemperature	year	month	GHI	Ordinal
5000	2055-07-28 08:00:00	28.545783	64.471621	2.575628	13.973272	102531.992188	21.18	2055	7	642.757086	
5001	2055-07-28 09:00:00	29.380029	55.186306	2.959718	15.382727	102502.679688	19.44	2055	7	760.595886	
5002	2055-07-28 10:00:00	29.719242	55.577503	3.178748	19.121150	102483.351562	19.86	2055	7	821.576213	
5003	2055-07-28 11:00:00	29.773220	55.968700	3.397777	22.859574	102464.023438	20.03	2055	7	821.657616	

In [29]:
```
# Verifying how does calculated DNI looks like in the file
p = pvl.irradiance.disc(ghi=df.GHI[5002],
                        solar_zenith=df.SolarZenith[5002],
                        datetime_or_doy=df.OrdinalDay[5002],
                        pressure=df.AtmosPressure[5002],
                        min_cos_zenith=0.065,
                        max_zenith=87,
                        max_airmass=(1/(math.cos(df.SolarZenith[5002])+0.50572*(96.07995-df.SolarZenith[5002])
                                        **(-1.6364))))['dni']
print(round(p.tolist(),2))
# Accounting for cloud cover
S1=p.tolist()*(1-(df.CloudCover[5002])/100)
round(S1,2)
```

```
923.01
```

Out[29]:
```
746.52
```

In [30]:
```
colmn = df.columns.tolist()
print (colmn)
```

```
['DateTime', 'DB_Temp', 'RelativeHumidity', 'WindSpeed', 'CloudCover', 'AtmosPressure', 'DewPointTemperature', 'y
ear', 'month', 'GHI', 'OrdinalDay', 'SolarZenith', 'DNI']
```

In [31]:
```
new_columns = ['DateTime','year','month','DB_Temp','DewPointTemperature','RelativeHumidity',
               'WindSpeed', 'CloudCover','AtmosPressure','GHI','DNI','SolarZenith']
df = df[new_columns]
df.head(2)
```

Out[31]:

	DateTime	year	month	DB_Temp	DewPointTemperature	RelativeHumidity	WindSpeed	CloudCover	AtmosPressure	GHI	DNI	SolarZenith
0	2055-01-01 00:00:00	2055	1	11.928796	10.76	92.573906	4.578446	96.912216	103194.500000	0.0	0.0	155.73
1	2055-01-01 01:00:00	2055	1	8.906792	7.46	90.627139	4.251208	74.319516	103196.804688	0.0	0.0	153.76

Calculating diffuse horizontal irradiance

In [36]:
```
dhi=[]
for i, row in df.iterrows():
    h = df.GHI[i]-(df.DNI[i]*math.cos(df.SolarZenith[i]))
    dhi.append(round(h,2))

df['DHI']= dhi
```

In [35]:
```
# Removing leap day data if year is a leap year
df = df[~((df['DateTime'].dt.month == 2) & (df['DateTime'].dt.day == 29))]
```

Saving the data into an excel sheet to later be used in creation of Typical weather files

In [ ]:
```
df.to_excel(r'D:\Weather_files\CORDEX_weather_data\MPI_REMO\MPI-ESM-LR_rcp85_r3i1p1_GERICS-REMO2015_v1_'+str(yea
```

Appendix 3-4:

- Creating a typical weather file from 30 years of yearly weather data

In [1]:
```
import os
import pandas as pd
from datetime import date, datetime, timedelta
```

Loading 30 year data in csv/xls format exported from yearly data

In [2]:
```
 # Before running...data for 22h and 23h was added to the last excel file to make sure it has the same length as
cwd = os.path.abspath(r'D:\Weather_files\CORDEX_weather_data\MPI_REMO')
files = os.listdir(cwd)
```

In [3]:
```
df = pd.DataFrame()
for file in files:
    if file.endswith('.xlsx'):
        df = df.append(pd.read_excel('D:/Weather_files/CORDEX_weather_data/MPI_REMO/'+file), ignore_index=True)
df.head(1)
```

Out[3]:

	DateTime	year	month	DB_Temp	DewPointTemperature	RelativeHumidity	WindSpeed	CloudCover	AtmosPressure	GHI	DNI	SolarZenith
0	2040-01-01	2040	1	5.669122	3.03	83.110458	4.562614	87.410179	100824.1875	0.0	0.0	155.76

In [4]:
```
# Because of three hour intervals and previous interpolations that covered the inner rows
# two hours of data at 22h and 23h at the end of each year is missing. ffill is used to fill 2 nan cells for a ye
df=df.set_index('DateTime').resample('H').last()
df = df.fillna(method='ffill')
# Removing leap day data if year is a leap year
df = df[~((df.index.month == 2) & (df.index.day == 29))]
# Changing data type
df = df.astype({"year":'int', "month":'int'})
```

In [53]:
```
# Seeing if joining datafiles have been done properly
df.iloc[8757:8762]
```

Out[53]:

DateTime	year	month	DB_Temp	DewPointTemperature	RelativeHumidity	WindSpeed	CloudCover	AtmosPressure	GHI	DNI	SolarZenith	DH
2040-12-31 21:00:00	2040	12	8.466852	8.14	97.828827	2.687051	18.505352	101886.265625	0.0	0.0	135.23	0.
2040-12-31 22:00:00	2040	12	8.466852	8.14	97.828827	2.687051	18.505352	101886.265625	0.0	0.0	135.23	0.
2040-12-31 23:00:00	2040	12	8.466852	8.14	97.828827	2.687051	18.505352	101886.265625	0.0	0.0	135.23	0.
2041-01-01 00:00:00	2041	1	10.870782	9.81	93.144684	4.460026	72.389999	101691.000000	0.0	0.0	155.70	0.
2041-01-01 01:00:00	2041	1	14.041664	12.60	91.031720	4.644883	75.561045	101631.875000	0.0	0.0	153.75	0.

In [6]:
```
# Average value of each variable
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 271558 entries, 2040-01-01 00:00:00 to 2070-12-31 21:00:00
Data columns (total 12 columns):
year                  271558 non-null int32
month                 271558 non-null int32
DB_Temp               271558 non-null float64
DewPointTemperature   271558 non-null float64
RelativeHumidity      271558 non-null float64
WindSpeed             271558 non-null float64
CloudCover            271558 non-null float64
AtmosPressure         271558 non-null float64
GHI                   271558 non-null float64
DNI                   271558 non-null float64
SolarZenith           271558 non-null float64
DHI                   271558 non-null float64
```

```
dtypes: float64(10), int32(2)
memory usage: 24.9 MB
```

Calculating daily means

In [7]:
```
# For each climatic parameter p : (dry bulb temperature, humidity ratio, GHI) perform the following operatios
# Resampling to calcualte daily means (p^-) of the given parameters (DBT,DPT,WSP,GHI)
df_daily = df.resample('D').mean()
```

In [8]:
```
new_columns = ['DB_Temp','DewPointTemperature','RelativeHumidity', 'WindSpeed','GHI','DNI']
df_daily = df_daily[new_columns]
df_daily.head(2)
```

Out[8]:

DateTime	DB_Temp	DewPointTemperature	RelativeHumidity	WindSpeed	GHI	DNI
2040-01-01	9.025214	6.305	83.100938	6.126465	7.808098	0.372917
2040-01-02	8.077459	4.975	80.829152	5.217163	15.170530	2.211250

In [9]:
```
df_daily.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 11323 entries, 2040-01-01 to 2070-12-31
Freq: D
Data columns (total 6 columns):
DB_Temp                11315 non-null float64
DewPointTemperature    11315 non-null float64
RelativeHumidity       11315 non-null float64
WindSpeed              11315 non-null float64
GHI                    11315 non-null float64
DNI                    11315 non-null float64
dtypes: float64(6)
memory usage: 619.2 KB
```

In [10]:
```
import numpy as np
import scipy
import matplotlib.pyplot as plt
import seaborn as sns
```

In [11]:
```
# Months dictionary
monthDict = {1:'Jan',2:'Feb',3:'Mar',4:'Apr',5:'May',6:'Jun',
             7:'Jul',8:'Aug',9:'Sep',10:'Oct',11:'Nov',12:'Dec'}
```

In [12]:
```
# A Dataframe to write out results
AllYears = np.arange(df_daily.index[0].year,df_daily.index[len(df_daily.index)-1].year+1,1).tolist()
All_df = pd.DataFrame(index=AllYears)
```

Calculating FS statistics for DB Temp, RH, and GHI, plus mean wind speed deviation for each year to multiyear.

In [13]:
```
for Month in [1,2,3,4,5,6,7,8,9,10,11,12]:
    tempe = []
    RelHu = []
    HGIrr = []
    WindS = []
    for Year in AllYears:
        #### Temperature
        y = df_daily[(df_daily.index.month == Month)].DB_Temp.values
        y = y[~np.isnan(y)]
        h, b = np.histogram(y, bins=100, density=True)
        cdf = np.cumsum(h*np.diff(b))
        #
        y1 = df_daily[((df_daily.index.month == Month) & (df_daily.index.year == Year))].DB_Temp.values
        y1 = y1[~np.isnan(y1)]
        h1, b1 = np.histogram(y1, bins=100, density=True)
        cdf1 = np.cumsum(h1*np.diff(b1))
        ## Finkelstien-Schafar Statistics
        ts = np.sum(abs(cdf - cdf1))
```

```
        tempe.append(ts)

        #### Relative humidity
        x = df_daily[(df_daily.index.month == Month)].RelativeHumidity.values
        x = x[~np.isnan(x)]
        j, a = np.histogram(x, bins=100, density=True)
        cdfRH = np.cumsum(j*np.diff(a))
        #
        x1 = df_daily[((df_daily.index.month == Month) & (df_daily.index.year == Year))].RelativeHumidity.values
        x1 = x1[~np.isnan(x1)]
        j1, a1 = np.histogram(x1, bins=100, density=True)
        cdfRH1 = np.cumsum(j1*np.diff(a1))
        ## Finkelstien-Schafar Statistics
        rs = np.sum(abs(cdfRH - cdfRH1))
        RelHu.append(rs)

        ####Global horizontal irradiance
        z = df_daily[(df_daily.index.month == Month)].GHI.values
        z = z[~np.isnan(z)]
        k, s = np.histogram(z, bins=100, density=True)
        cdfGHI = np.cumsum(k*np.diff(s))
        #
        z1 = df_daily[((df_daily.index.month == Month) & (df_daily.index.year == Year))].GHI.values
        z1 = z1[~np.isnan(z1)]
        k1, s1 = np.histogram(z1, bins=100, density=True)
        cdfGHI1 = np.cumsum(k1*np.diff(s1))
        ## Finkelstien-Schafar Statistics
        hs = np.sum(abs(cdfGHI - cdfGHI1))
        HGIrr.append(hs)

        #### Deviation of month's mean wind speed for a year from multiyear mean
        w = df_daily[(df_daily.index.month == Month)].WindSpeed.values
        w = w[~np.isnan(w)]
        WS = sum(w)/len(w)
        #
        w1 = df_daily[((df_daily.index.month == Month) & (df_daily.index.year == Year))].WindSpeed.values
        w1 = w1[~np.isnan(w1)]
        WS1 = sum(w1)/len(w1)
        ## Deviation of means
        ws = abs(WS - WS1)
        WindS.append(ws)
        ###

    All_df['DBT_'+str(monthDict[Month])]=tempe
    All_df['RH_'+str(monthDict[Month])]=RelHu
    All_df['GHI_'+str(monthDict[Month])]=HGIrr
    All_df['WS_'+str(monthDict[Month])]=WindS
```

In [14]:

```
All_df.head(2)
```

Out[14]:

	DBT_Jan	RH_Jan	GHI_Jan	WS_Jan	DBT_Feb	RH_Feb	GHI_Feb	WS_Feb	DBT_Mar	RH_Mar	...	GHI_Oct	WS_Oct	DBT_
2040	11.308012	21.705515	9.312175	0.225798	10.269585	23.408986	7.342166	0.320906	6.113424	14.323621	...	10.753382	0.108384	11.365
2041	11.310094	25.964620	4.329865	0.223447	16.417051	13.667051	9.213134	0.104302	10.490114	15.064516	...	10.876171	0.210872	25.659

2 rows × 48 columns

Ranking and selecting the best year for each month

In [15]:

```
##Ranking dry bulb temperature
All_df['R_DBT_Jan']=All_df['DBT_Jan'].rank(ascending=True)
All_df['R_DBT_Feb']=All_df['DBT_Feb'].rank(ascending=True)
All_df['R_DBT_Mar']=All_df['DBT_Mar'].rank(ascending=True)
All_df['R_DBT_Apr']=All_df['DBT_Apr'].rank(ascending=True)
All_df['R_DBT_May']=All_df['DBT_May'].rank(ascending=True)
All_df['R_DBT_Jun']=All_df['DBT_Jun'].rank(ascending=True)
All_df['R_DBT_Jul']=All_df['DBT_Jul'].rank(ascending=True)
All_df['R_DBT_Aug']=All_df['DBT_Aug'].rank(ascending=True)
All_df['R_DBT_Sep']=All_df['DBT_Sep'].rank(ascending=True)
All_df['R_DBT_Oct']=All_df['DBT_Oct'].rank(ascending=True)
All_df['R_DBT_Nov']=All_df['DBT_Nov'].rank(ascending=True)
All_df['R_DBT_Dec']=All_df['DBT_Dec'].rank(ascending=True)
##ranking relative humidity
All_df['R_RH_Jan']=All_df['RH_Jan'].rank(ascending=True)
All_df['R_RH_Feb']=All_df['RH_Feb'].rank(ascending=True)
All_df['R_RH_Mar']=All_df['RH_Mar'].rank(ascending=True)
All_df['R_RH_Apr']=All_df['RH_Apr'].rank(ascending=True)
All_df['R_RH_May']=All_df['RH_May'].rank(ascending=True)
```

```
All_df['R_RH_Jun']=All_df['RH_Jun'].rank(ascending=True)
All_df['R_RH_Jul']=All_df['RH_Jul'].rank(ascending=True)
All_df['R_RH_Aug']=All_df['RH_Aug'].rank(ascending=True)
All_df['R_RH_Sep']=All_df['RH_Sep'].rank(ascending=True)
All_df['R_RH_Oct']=All_df['RH_Oct'].rank(ascending=True)
All_df['R_RH_Nov']=All_df['RH_Nov'].rank(ascending=True)
All_df['R_RH_Dec']=All_df['RH_Dec'].rank(ascending=True)
##ranking global horizontal irradiance
All_df['R_GHI_Jan']=All_df['GHI_Jan'].rank(ascending=True)
All_df['R_GHI_Feb']=All_df['GHI_Feb'].rank(ascending=True)
All_df['R_GHI_Mar']=All_df['GHI_Mar'].rank(ascending=True)
All_df['R_GHI_Apr']=All_df['GHI_Apr'].rank(ascending=True)
All_df['R_GHI_May']=All_df['GHI_May'].rank(ascending=True)
All_df['R_GHI_Jun']=All_df['GHI_Jun'].rank(ascending=True)
All_df['R_GHI_Jul']=All_df['GHI_Jul'].rank(ascending=True)
All_df['R_GHI_Aug']=All_df['GHI_Aug'].rank(ascending=True)
All_df['R_GHI_Sep']=All_df['GHI_Sep'].rank(ascending=True)
All_df['R_GHI_Oct']=All_df['GHI_Oct'].rank(ascending=True)
All_df['R_GHI_Nov']=All_df['GHI_Nov'].rank(ascending=True)
All_df['R_GHI_Dec']=All_df['GHI_Dec'].rank(ascending=True)
```

In [16]:

```
### Sum the FS value of three primary variables (DBT,RH,GHI) for each month

All_df['January']=All_df['R_DBT_Jan']+ All_df['R_RH_Jan']+ All_df['R_GHI_Jan']
All_df['February']=All_df['R_DBT_Feb']+ All_df['R_RH_Feb']+ All_df['R_GHI_Feb']
All_df['March']=All_df['R_DBT_Mar']+ All_df['R_RH_Mar']+ All_df['R_GHI_Mar']
All_df['April']=All_df['R_DBT_Apr']+ All_df['R_RH_Apr']+ All_df['R_GHI_Apr']
All_df['MAY']=All_df['R_DBT_May']+ All_df['R_RH_May']+ All_df['R_GHI_May']
All_df['June']=All_df['R_DBT_Jun']+ All_df['R_RH_Jun']+ All_df['R_GHI_Jun']
All_df['July']=All_df['R_DBT_Jul']+ All_df['R_RH_Jul']+ All_df['R_GHI_Jul']
All_df['August']=All_df['R_DBT_Aug']+ All_df['R_RH_Aug']+ All_df['R_GHI_Aug']
All_df['September']=All_df['R_DBT_Sep']+ All_df['R_RH_Sep']+ All_df['R_GHI_Sep']
All_df['October']=All_df['R_DBT_Oct']+ All_df['R_RH_Oct']+ All_df['R_GHI_Oct']
All_df['November']=All_df['R_DBT_Nov']+ All_df['R_RH_Nov']+ All_df['R_GHI_Nov']
All_df['December']=All_df['R_DBT_Dec']+ All_df['R_RH_Dec']+ All_df['R_GHI_Dec']

## Rank order the sum total of variables to identify three lowest sizes
All_df['Jan']=All_df['January'].rank(method='first',ascending=True)
All_df['Feb']=All_df['February'].rank(method='first',ascending=True)
All_df['Mar']=All_df['March'].rank(method='first',ascending=True)
All_df['Apr']=All_df['April'].rank(method='first',ascending=True)
All_df['May']=All_df['MAY'].rank(method='first',ascending=True)
All_df['Jun']=All_df['June'].rank(method='first',ascending=True)
All_df['Jul']=All_df['July'].rank(method='first',ascending=True)
All_df['Aug']=All_df['August'].rank(method='first',ascending=True)
All_df['Sep']=All_df['September'].rank(method='first',ascending=True)
All_df['Oct']=All_df['October'].rank(method='first',ascending=True)
All_df['Nov']=All_df['November'].rank(method='first',ascending=True)
All_df['Dec']=All_df['December'].rank(method='first',ascending=True)
```

In [17]:

```
#colmn = All_df.columns.tolist()
#print (colmn)
```

In [18]:

```
all_columns = ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep',
               'Oct','Nov','Dec','WS_Jan','WS_Feb','WS_Mar','WS_Apr',
               'WS_May','WS_Jun','WS_Jul','WS_Aug','WS_Sep',
               'WS_Oct','WS_Nov','WS_Dec']
df_MS = All_df[all_columns]
```

In [19]:

```
### Selecting year of TMY for eaxh month
January = df_MS.index[df_MS['WS_Jan']==min(df_MS['WS_Jan']
                                           .loc[df_MS.index.isin(df_MS.index[df_MS['Jan']<4]
                                                                 .tolist())])].values[0]
February = df_MS.index[df_MS['WS_Feb']==min(df_MS['WS_Feb']
                                            .loc[df_MS.index.isin(df_MS.index[df_MS['Feb']<4]
                                                                  .tolist())])].values[0]
March = df_MS.index[df_MS['WS_Mar']==min(df_MS['WS_Mar']
                                         .loc[df_MS.index.isin(df_MS.index[df_MS['Mar']<4]
                                                               .tolist())])].values[0]
April = df_MS.index[df_MS['WS_Apr']==min(df_MS['WS_Apr']
                                         .loc[df_MS.index.isin(df_MS.index[df_MS['Apr']<4]
                                                               .tolist())])].values[0]
May = df_MS.index[df_MS['WS_May']==min(df_MS['WS_May']
                                       .loc[df_MS.index.isin(df_MS.index[df_MS['May']<4]
                                                             .tolist())])].values[0]
June = df_MS.index[df_MS['WS_Jun']==min(df_MS['WS_Jun']
                                        .loc[df_MS.index.isin(df_MS.index[df_MS['Jun']<4]
                                                              .tolist())])].values[0]
```

```
July = df_MS.index[df_MS['WS_Jul']==min(df_MS['WS_Jul']
                                        .loc[df_MS.index.isin(df_MS.index[df_MS['Jul']<4]
                                                              .tolist())])].values[0]
August = df_MS.index[df_MS['WS_Aug']==min(df_MS['WS_Aug']
                                          .loc[df_MS.index.isin(df_MS.index[df_MS['Aug']<4]
                                                                .tolist())])].values[0]
September = df_MS.index[df_MS['WS_Sep']==min(df_MS['WS_Sep']
                                             .loc[df_MS.index.isin(df_MS.index[df_MS['Sep']<4]
                                                                   .tolist())])].values[0]
October = df_MS.index[df_MS['WS_Oct']==min(df_MS['WS_Oct']
                                           .loc[df_MS.index.isin(df_MS.index[df_MS['Oct']<4]
                                                                 .tolist())])].values[0]
November = df_MS.index[df_MS['WS_Nov']==min(df_MS['WS_Nov']
                                            .loc[df_MS.index.isin(df_MS.index[df_MS['Nov']<4]
                                                                  .tolist())])].values[0]
December = df_MS.index[df_MS['WS_Dec']==min(df_MS['WS_Dec']
                                            .loc[df_MS.index.isin(df_MS.index[df_MS['Dec']<4]
                                                                  .tolist())])].values[0]
```

In [20]:

```
print('January = '+str(January))
print('February = '+str(February))
print('March = '+str(March))
print('April = '+str(April))
print('May = '+str(May))
print('June = '+str(June))
print('July = '+str(July))
print('August = '+str(August))
print('September = '+str(September))
print('October = '+str(October))
print('November = '+str(November))
print('December = '+str(December))
```

```
January = 2064
February = 2068
March = 2047
April = 2051
May = 2067
June = 2042
July = 2045
August = 2060
September = 2047
October = 2048
November = 2040
December = 2065
```

In [21]:

```
import warnings
warnings.filterwarnings("ignore")
```

Removing first and last 8 hours of data of each month and interpolating them linearly to smooth out transition

In [22]:

```
####
df1 = df[((df.index.month == 1) & (df.index.year == January))]
# last 7 hours of the month
df1['DB_Temp'].iloc[-7:]=np.nan
df1['CloudCover'].iloc[-7:]=np.nan
df1['WindSpeed'].iloc[-7:]=np.nan
df1['RelativeHumidity'].iloc[-7:]=np.nan
df1['DewPointTemperature'].iloc[-7:]=np.nan
######
df2 = df[((df.index.month == 2) & (df.index.year == February))]
# first 8 hours of the month
df2['DB_Temp'].iloc[:7]=np.nan
df2['CloudCover'].iloc[:7]=np.nan
df2['WindSpeed'].iloc[:7]=np.nan
df2['RelativeHumidity'].iloc[:7]=np.nan
df2['DewPointTemperature'].iloc[:7]=np.nan
# last 8 hours of the month
df2['DB_Temp'].iloc[-7:]=np.nan
df2['CloudCover'].iloc[-7:]=np.nan
df2['WindSpeed'].iloc[-7:]=np.nan
df2['RelativeHumidity'].iloc[-7:]=np.nan
df2['DewPointTemperature'].iloc[-7:]=np.nan
######
df3 = df[((df.index.month == 3) & (df.index.year == March))]
# first 8 hours of the month
df3['DB_Temp'].iloc[:7]=np.nan
df3['CloudCover'].iloc[:7]=np.nan
```

```
df3['WindSpeed'].iloc[:7]=np.nan
df3['RelativeHumidity'].iloc[:7]=np.nan
df3['DewPointTemperature'].iloc[:7]=np.nan
# last 8 hours of the month
df3['DB_Temp'].iloc[-7:]=np.nan
df3['CloudCover'].iloc[-7:]=np.nan
df3['WindSpeed'].iloc[-7:]=np.nan
df3['RelativeHumidity'].iloc[-7:]=np.nan
df3['DewPointTemperature'].iloc[-7:]=np.nan
######
df4 = df[((df.index.month == 4) & (df.index.year == April))]
# first 8 hours of the month
df4['DB_Temp'].iloc[:7]=np.nan
df4['CloudCover'].iloc[:7]=np.nan
df4['WindSpeed'].iloc[:7]=np.nan
df4['RelativeHumidity'].iloc[:7]=np.nan
df4['DewPointTemperature'].iloc[:7]=np.nan
# last 8 hours of the month
df4['DB_Temp'].iloc[-7:]=np.nan
df4['CloudCover'].iloc[-7:]=np.nan
df4['WindSpeed'].iloc[-7:]=np.nan
df4['RelativeHumidity'].iloc[-7:]=np.nan
df4['DewPointTemperature'].iloc[-7:]=np.nan
######
df5 = df[((df.index.month == 5) & (df.index.year == May))]
# first 8 hours of the month
df5['DB_Temp'].iloc[:7]=np.nan
df5['CloudCover'].iloc[:7]=np.nan
df5['WindSpeed'].iloc[:7]=np.nan
df5['RelativeHumidity'].iloc[:7]=np.nan
df5['DewPointTemperature'].iloc[:7]=np.nan
# last 8 hours of the month
df5['DB_Temp'].iloc[-7:]=np.nan
df5['CloudCover'].iloc[-7:]=np.nan
df5['WindSpeed'].iloc[-7:]=np.nan
df5['RelativeHumidity'].iloc[-7:]=np.nan
df5['DewPointTemperature'].iloc[-7:]=np.nan
######
df6 = df[((df.index.month == 6) & (df.index.year == June))]
# first 8 hours of the month
df6['DB_Temp'].iloc[:7]=np.nan
df6['CloudCover'].iloc[:7]=np.nan
df6['WindSpeed'].iloc[:7]=np.nan
df6['RelativeHumidity'].iloc[:7]=np.nan
df6['DewPointTemperature'].iloc[:7]=np.nan
# last 8 hours of the month
df6['DB_Temp'].iloc[-7:]=np.nan
df6['CloudCover'].iloc[-7:]=np.nan
df6['WindSpeed'].iloc[-7:]=np.nan
df6['RelativeHumidity'].iloc[-7:]=np.nan
df6['DewPointTemperature'].iloc[-7:]=np.nan
######
df7 = df[((df.index.month == 7) & (df.index.year == July))]
# first 8 hours of the month
df7['DB_Temp'].iloc[:7]=np.nan
df7['CloudCover'].iloc[:7]=np.nan
df7['WindSpeed'].iloc[:7]=np.nan
df7['RelativeHumidity'].iloc[:7]=np.nan
df7['DewPointTemperature'].iloc[:7]=np.nan
# last 8 hours of the month
df7['DB_Temp'].iloc[-7:]=np.nan
df7['CloudCover'].iloc[-7:]=np.nan
df7['WindSpeed'].iloc[-7:]=np.nan
df7['RelativeHumidity'].iloc[-7:]=np.nan
df7['DewPointTemperature'].iloc[-7:]=np.nan
######
df8 = df[((df.index.month == 8) & (df.index.year == August))]
# first 8 hours of the month
df8['DB_Temp'].iloc[:7]=np.nan
df8['CloudCover'].iloc[:7]=np.nan
df8['WindSpeed'].iloc[:7]=np.nan
df8['RelativeHumidity'].iloc[:7]=np.nan
df8['DewPointTemperature'].iloc[:7]=np.nan
# last 8 hours of the month
df8['DB_Temp'].iloc[-7:]=np.nan
df8['CloudCover'].iloc[-7:]=np.nan
df8['WindSpeed'].iloc[-7:]=np.nan
df8['RelativeHumidity'].iloc[-7:]=np.nan
df8['DewPointTemperature'].iloc[-7:]=np.nan
######
df9 = df[((df.index.month == 9) & (df.index.year == September))]
# first 8 hours of the month
df9['DB_Temp'].iloc[:7]=np.nan
```

```
df9['CloudCover'].iloc[:7]=np.nan
df9['WindSpeed'].iloc[:7]=np.nan
df9['RelativeHumidity'].iloc[:7]=np.nan
df9['DewPointTemperature'].iloc[:7]=np.nan
# last 8 hours of the month
df9['DB_Temp'].iloc[-7:]=np.nan
df9['CloudCover'].iloc[-7:]=np.nan
df9['WindSpeed'].iloc[-7:]=np.nan
df9['RelativeHumidity'].iloc[-7:]=np.nan
df9['DewPointTemperature'].iloc[-7:]=np.nan
######
df10 = df[((df.index.month == 10) & (df.index.year == October))]
# first 8 hours of the month
df10['DB_Temp'].iloc[:7]=np.nan
df10['CloudCover'].iloc[:7]=np.nan
df10['WindSpeed'].iloc[:7]=np.nan
df10['RelativeHumidity'].iloc[:7]=np.nan
df10['DewPointTemperature'].iloc[:7]=np.nan
# last 8 hours of the month
df10['DB_Temp'].iloc[-7:]=np.nan
df10['CloudCover'].iloc[-7:]=np.nan
df10['WindSpeed'].iloc[-7:]=np.nan
df10['RelativeHumidity'].iloc[-7:]=np.nan
df10['DewPointTemperature'].iloc[-7:]=np.nan
######
df11 = df[((df.index.month == 11) & (df.index.year == November))]
# first 8 hours of the month
df11['DB_Temp'].iloc[:7]=np.nan
df11['CloudCover'].iloc[:7]=np.nan
df11['WindSpeed'].iloc[:7]=np.nan
df11['RelativeHumidity'].iloc[:7]=np.nan
df11['DewPointTemperature'].iloc[:7]=np.nan
# last 8 hours of the month
df11['DB_Temp'].iloc[-7:]=np.nan
df11['CloudCover'].iloc[-7:]=np.nan
df11['WindSpeed'].iloc[-7:]=np.nan
df11['RelativeHumidity'].iloc[-7:]=np.nan
df11['DewPointTemperature'].iloc[-7:]=np.nan
######
df12 = df[((df.index.month == 12) & (df.index.year == December))]
# first 8 hours of the month
df12['DB_Temp'].iloc[:7]=np.nan
df12['CloudCover'].iloc[:7]=np.nan
df12['WindSpeed'].iloc[:7]=np.nan
df12['RelativeHumidity'].iloc[:7]=np.nan
df12['DewPointTemperature'].iloc[:7]=np.nan

Frames = [df1,df2,df3,df4,df5,df6,df7,df8,df9,df10,df11,df12]
result = pd.concat(Frames)
result = result.interpolate()
```

In [54]:

```
result.iloc[4446:4448]
```

Out[54]:

DateTime	year	month	DB_Temp	DewPointTemperature	RelativeHumidity	WindSpeed	CloudCover	AtmosPressure	GHI	DNI	SolarZ
2045-07-05 06:00:00	2045	7	16.187585	13.62	84.758759	2.632964	74.367569	102383.687500	63.283680	3.14	7
2045-07-05 07:00:00	2045	7	16.964542	13.35	79.273621	2.928851	68.954421	102389.856771	140.531202	18.89	6

In [24]:

```
result['DHI'] = result['DHI'].apply(lambda x : x if x > 0 else 0)
```

In [26]:

```
result.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 8760 entries, 2064-01-01 00:00:00 to 2065-12-31 23:00:00
Data columns (total 12 columns):
year                  8760 non-null int32
month                 8760 non-null int32
DB_Temp               8760 non-null float64
DewPointTemperature   8760 non-null float64
RelativeHumidity      8760 non-null float64
WindSpeed             8760 non-null float64
CloudCover            8760 non-null float64
```

```
AtmosPressure          8760 non-null float64
GHI                    8760 non-null float64
DNI                    8760 non-null float64
SolarZenith            8760 non-null float64
DHI                    8760 non-null float64
dtypes: float64(10), int32(2)
memory usage: 821.2 KB
```

In [56]:
```
result.head(3)
```

Out[56]:

DateTime	year	month	DB_Temp	DewPointTemperature	RelativeHumidity	WindSpeed	CloudCover	AtmosPressure	GHI	DNI	SolarZenith	DH
2064-01-01 00:00:00	2064	1	12.984369	12.08	94.240639	7.898614	100.0	99949.273438	0.0	0.0	155.75	0.
2064-01-01 01:00:00	2064	1	12.483222	11.62	94.461810	7.677654	100.0	99883.575521	0.0	0.0	153.77	0.
2064-01-01 02:00:00	2064	1	12.673921	11.84	94.682981	7.456694	100.0	99817.877604	0.0	0.0	147.22	0.

In [55]:
```
result.tail(3)
```

Out[55]:

DateTime	year	month	DB_Temp	DewPointTemperature	RelativeHumidity	WindSpeed	CloudCover	AtmosPressure	GHI	DNI	SolarZenith	DH
2065-12-31 21:00:00	2065	12	10.496912	10.17	97.849716	1.399294	88.309677	103469.96875	0.0	0.0	135.25	0.
2065-12-31 22:00:00	2065	12	10.496912	10.17	97.849716	1.399294	88.309677	103469.96875	0.0	0.0	135.25	0.
2065-12-31 23:00:00	2065	12	10.496912	10.17	97.849716	1.399294	88.309677	103469.96875	0.0	0.0	135.25	0.

Overwriting calculated data to an existing epw file and saving it

In [29]:
```
## Writing the weather data on epw weather template downloaded from EnergyPlus weather data platform
from ladybug.epw import EPW
from pandas import DataFrame
from collections import OrderedDict
%config Completer.use_jedi = False
```

In [30]:
```
#field_number: a value between 0 to 34 for different available epw fields.
#             0 Year
#             1 Month
#             2 Day
#             3 Hour
#             4 Minute

#             6 Dry Bulb Temperature
#             7 Dew Point Temperature
#             8 Relative Humidity
#             9 Atmospheric Station Pressure
#             10 Extraterrestrial Horizontal Radiation
#             11 Extraterrestrial Direct Normal Radiation
#             12 Horizontal Infrared Radiation Intensity
#             13 Global Horizontal Radiation
#             14 Direct Normal Radiation
#             15 Diffuse Horizontal Radiation
#             16 Global Horizontal Illuminance
#             17 Direct Normal Illuminance
#             18 Diffuse Horizontal Illuminance
#             19 Zenith Luminance
#             20 Wind Direction
#             21 Wind Speed
#             22 Total Sky Cover
#             23 Opaque Sky Cover
```

```
#            24 Visibility
#            25 Ceiling Height
#            26 Present Weather Observation
#            27 Present Weather Codes
#            28 Precipitable Water
#            29 Aerosol Optical Depth
#            30 Snow Depth
#            31 Days Since Last Snowfall
#            32 Albedo
#            33 Liquid Precipitation Depth
#            34 Liquid Precipitation Quantity
```

In [31]:
```
def changeEPWData(oldEpwFilePath,newEpwFilePath,dataIndex,dataList):
    with open(oldEpwFilePath) as oldStream,open(newEpwFilePath,"w") as newStream:
        numCount=0
        for idx,lines in enumerate(oldStream):
            if lines.strip():
                try:
                    lineSplit=lines.strip().split(",")
                    dataTest=float(lineSplit[0])
                    lineSplit[dataIndex]=str(dataList[numCount])
                    data=",".join(lineSplit)
                    newStream.write(data+"\n")
                    numCount+=1
                except ValueError:
                    newStream.write(lines.strip()+"\n")
            else:
                newStream.write(lines)
    return newEpwFilePath
```

Everytime a set of values are overwritten on an epw with this function, it has to be saved then on the saved version the second climate variable needs to be overwritten. For the moment, important points are the first and last file. (when i get time I will improve the function to enable simultaneous overwriting of all varibles).

In [32]:
```
# Writing Dry bulb temperature on epw file.
changeEPWData(r'D:\Weather_files\Nantes-RCP8-5_2050.epw',
              r'D:\Weather_files\CORDEX_weather_data\MPI_REMO\Temporary\CNRM2040_2070T.epw',
              dataIndex=6,
              dataList=result.DB_Temp)
```

Out[32]: 'D:\\Weather_files\\CORDEX_weather_data\\MPI_REMO\\Temporary\\CNRM2040_2070T.epw'

In [33]:
```
# Overwriting year
changeEPWData(r'D:\Weather_files\CORDEX_weather_data\MPI_REMO\Temporary\CNRM2040_2070T.epw',
              r'D:\Weather_files\CORDEX_weather_data\MPI_REMO\Temporary\CNRM2040_2070Y.epw',
              dataIndex=0,
              dataList=result.year)
```

Out[33]: 'D:\\Weather_files\\CORDEX_weather_data\\MPI_REMO\\Temporary\\CNRM2040_2070Y.epw'

In [34]:
```
# Overwriting dew point temperature
changeEPWData(r'D:\Weather_files\CORDEX_weather_data\MPI_REMO\Temporary\CNRM2040_2070Y.epw',
              r'D:\Weather_files\CORDEX_weather_data\MPI_REMO\Temporary\CNRM2040_2070DP.epw',
              dataIndex=7,
              dataList=result.DewPointTemperature)
```

Out[34]: 'D:\\Weather_files\\CORDEX_weather_data\\MPI_REMO\\Temporary\\CNRM2040_2070DP.epw'

In [35]:
```
# Overwriting RH
changeEPWData(r'D:\Weather_files\CORDEX_weather_data\MPI_REMO\Temporary\CNRM2040_2070DP.epw',
              r'D:\Weather_files\CORDEX_weather_data\MPI_REMO\Temporary\CNRM2040_2070RH.epw',
              dataIndex=8,
              dataList=result.RelativeHumidity)
```

Out[35]: 'D:\\Weather_files\\CORDEX_weather_data\\MPI_REMO\\Temporary\\CNRM2040_2070RH.epw'

In [36]:
```
# Overwriting Atmospheric pressure
changeEPWData(r'D:\Weather_files\CORDEX_weather_data\MPI_REMO\Temporary\CNRM2040_2070RH.epw',
              r'D:\Weather_files\CORDEX_weather_data\MPI_REMO\Temporary\CNRM2040_2070AP.epw',
```

```
                dataIndex=9,
                dataList=result.AtmosPressure)
```

Out[36]: 'D:\\Weather_files\\CORDEX_weather_data\\MPI_REMO\\Temporary\\CNRM2040_2070AP.epw'

In [37]:
```
# Overwriting GHI
changeEPWData(r'D:\Weather_files\CORDEX_weather_data\MPI_REMO\Temporary\CNRM2040_2070AP.epw',
                r'D:\Weather_files\CORDEX_weather_data\MPI_REMO\Temporary\CNRM2040_2070GHI.epw',
                dataIndex=13,
                dataList=result.GHI)
```

Out[37]: 'D:\\Weather_files\\CORDEX_weather_data\\MPI_REMO\\Temporary\\CNRM2040_2070GHI.epw'

In [38]:
```
# Overwriting DNI
changeEPWData(r'D:\Weather_files\CORDEX_weather_data\MPI_REMO\Temporary\CNRM2040_2070GHI.epw',
                r'D:\Weather_files\CORDEX_weather_data\MPI_REMO\Temporary\CNRM2040_2070DNI.epw',
                dataIndex=14,
                dataList=result.DNI)
```

Out[38]: 'D:\\Weather_files\\CORDEX_weather_data\\MPI_REMO\\Temporary\\CNRM2040_2070DNI.epw'

In [39]:
```
# Overwriting DHI
changeEPWData(r'D:\Weather_files\CORDEX_weather_data\MPI_REMO\Temporary\CNRM2040_2070DNI.epw',
                r'D:\Weather_files\CORDEX_weather_data\MPI_REMO\Temporary\CNRM2040_2070DHI.epw',
                dataIndex=15,
                dataList=result.DHI)
```

Out[39]: 'D:\\Weather_files\\CORDEX_weather_data\\MPI_REMO\\Temporary\\CNRM2040_2070DHI.epw'

In [40]:
```
# Overwriting wind speed
changeEPWData(r'D:\Weather_files\CORDEX_weather_data\MPI_REMO\Temporary\CNRM2040_2070DHI.epw',
                r'D:\Weather_files\MPI_REMO_deleteMe.epw',
                dataIndex=21,
                dataList=result.WindSpeed)
```

Out[40]: 'D:\\Weather_files\\MPI_REMO_deleteMe.epw'

In [47]:
```
## Export result as csv
#result.to_csv(r'D:\Weather_files\CORDEX_weather_data\CSV_IPSL2040_2070.csv',index = 0)
```

Appendix 3-5:

Projecting Urban Heat Island Effect on .epw weather files

In [24]:
```
from uwg import UWG
from ladybug.epw import EPW
```

UWG model Simulation

In [26]:
```
# Define the .epw, .uwg paths to create an uwg object.
epw_path = "D:/Weather_files/Observed_Nantes_2003.epw"
# Initialize the UWG model by passing parameters as arguments, or relying on defaults KM7_3
model = UWG.from_param_args(epw_path=epw_path,
                            bld=[('midriseapartment', 'pre80', 0.2),('custombuilding', 'pre80', 0.8)],
                            bldheight=6.5,
                            blddensity=0.6,
                            vertohor=0.178,
                            grasscover=0.25,
                            treecover=0.15,
                            zone='4A',
                            month=1,
                            day=1,
                            nday=365)
model.generate()
model.simulate()
# Write the simulation result to a file. Wil be saved in the same directory by adding UWG at the end of file name
model.write_epw()
```

```
Simulating new temperature and humidity values for 2 days from 1/1.
Simulating Day 1
Simulating Day 2
New climate file is generated at D:/Weather_files\Observed_Nantes_2003_UWG.epw.
```

Parameters that can/must be added to the model

In [ ]:
```
# UWG object constants
        * epw_path --   Full path of the rural .epw file that will be morphed.
        * new_epw_path -Full path of the file name of the morphed .epw file.
        * refBEM --     Reference BEMDef matrix defined by built type, era, and zone.
        * refSchedule --Reference SchDef matrix defined by built type, era, and zone.
        * month --      Number (1-12) representing simulation start month.
        * day --        Number (1-31) representing simulation start day.
        * nday --       Number of days to simulate.
        * dtsim --      Simlation time step in seconds.
        * dtweather --  Number for weather data time-step in seconds.
        * autosize --   Boolean to set HVAC autosize.
        * sensocc --    Sensible heat from occupant [W].
        * latfocc --    Latent heat fraction from occupant.
        * radfocc --    Radiant heat fraction from occupant.
        * radfequip --  Radiant heat fraction from equipment.
        * radflight --  Radiant heat fraction from electric light.
        * h_ubl1 --     Daytime urban boundary layer height in meters.
        * h_ubl2 --     Nighttime urban boundary layer height in meters.
        * h_ref --      Inversion height in meters.
        * h_temp --     Temperature height in meters.
        * h_wind --     Wind height in meters.
        * c_circ --     Wind scaling coefficient.
        * c_exch --     Exchange velocity coefficient.
        * maxday --     Maximum heat flux threshold for daytime conditions [W/m2].
        * maxnight ---- Maximum heat flux threshold for nighttime conditions [W/m2].
        * windmin --    Minimum wind speed in m/s.
        * h_obs --      Rural average obstacle height in meters.
        * bldheight --  Average Urban building height in meters.
        * h_mix --      Fraction of HVAC waste heat released to street canyon.
        * blddensity -- Building footprint density as fraction of urban area.
        * vertohor --   Vertical-to-horizontal urban area ratio.
        * charlength -- Urban characteristic length in meters.
        * albroad --    Urban road albedo.
        * droad --      Thickness of urban road pavement thickness in meters.
        * sensanth --   Street level anthropogenic sensible heat [W/m2].
        * zone -- Index representing an ASHRAE climate zone.
        * grasscover -- Fraction of urban ground covered in grass only.
        * treecover --  Fraction of urban ground covered in trees.
        * vegstart --   Month in which vegetation starts to evapotranspire.
        * vegend --     Month in which vegetation stops evapotranspiration.
        * albveg --     Vegetation albedo.
        * rurvegcover --Fraction of rural ground covered by vegetation.
        * latgrss --    Fraction of latent heat absorbed by urban grass.
        * lattree --    Fraction latent heat absorbed by urban trees.
        * schtraffic -- Schedule of fractional anthropogenic heat load.
```

```
        * kroad --      Road pavement conductivity [W/m-K].
        * croad --      Road pavement volumetric heat capacity [J/m^3K].
        * bld --        Matrix of numbers representing fraction of urban building stock.
        * albroof --    Average building roof albedo.
        * vegroof --    Fraction of roof covered in grass/shrubs.
        * glzr --       Building glazing ratio.
        * albwall --    Building albedo.
        * shgc --       Building glazing Solar Heat Gain Coefficient (SHGC).
        * flr_h --      Building floor height in meters.
        * ref_bem_vector -- List of custom BEMDef objects to override the refBEM.
        * ref_sch_vector -- List of custom SchDef objects to override the refSchedule.
    OPTIONAL_PARAMETER_SET = {'shgc', 'albroof',
                              'glzr', 'vegroof', 'albwall', 'flr_h'}
```

Default values

In [ ]:

```
# if there is no default value written in front of it, it means it is obligatory
'''
cls,
bld=DEFAULT_BLD,
bldheight,
blddensity,
vertohor,
grasscover,
treecover,
zone,
month=1,
day=1,
nday=31,
dtsim=300,
dtweather=3600,
autosize=False,
h_mix=1,
sensocc=100,
latfocc=0.3,
radfocc=0.2,
radfequip=0.5,
radflight=0.7,
charlength=1000,
albroad=0.1,
droad=0.5,
kroad=1,
croad=1600000,
rurvegcover=0.9,
vegstart=4,
vegend=10,
albveg=0.25,
latgrss=0.4,
lattree=0.6,
sensanth=20,
h_ubl1=1000,
h_ubl2=80,
h_ref=150,
h_temp=2,
h_wind=10,
c_circ=1.2,
c_exch=1,
maxday=150,
maxnight=20,
windmin=1,
h_obs=0.1,
schtraffic=DEFAULT_SCHTRAFFIC,
epw_path=None,
new_epw_dir=None,
new_epw_name=None,
ref_bem_vector=None,
ref_sch_vector=None):

    DEFAULT_BLD = (('largeoffice', 'pst80', 0.4),
                   ('midriseapartment', 'pst80', 0.6))
    DEFAULT_SCHTRAFFIC = (
        (0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.7, 0.9, 0.9, 0.6, 0.6, 0.6, 0.6, 0.6, 0.7, 0.8,
         0.9, 0.9, 0.8, 0.8, 0.7, 0.3, 0.2, 0.2),  # Weekday
        (0.2, 0.2, 0.2, 0.2, 0.2, 0.3, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.6, 0.7,
         0.7, 0.7, 0.7, 0.5, 0.4, 0.3, 0.2, 0.2),  # Saturday
        (0.2, 0.2, 0.2, 0.2, 0.2, 0.3, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4,
         0.4, 0.4, 0.4, 0.4, 0.3, 0.3, 0.2, 0.2))  # Sunday
'''

'''
bld:
```

```
* 'fullservicerestaurant'
* 'hospital'
* 'largehotel'
* 'largeoffice'
* 'medoffice'
* 'midriseapartment'
* 'outpatient'
* 'primaryschool'
* 'quickservicerestaurant'
* 'secondaryschool'
* 'smallhotel'
* 'smalloffice'
* 'standaloneretail'
* 'stripmall'
* 'supermarket'
* 'warehouse'
Choose from the following built eras:
* 'pre80'
* 'pst80'
* 'new'
Custom building types can also be referenced in this property. For example, a
built stock consisting of 40% post-1980's large office, 30% new midrise
apartment, and 30% of a pre-1980s custom building type (defined by the user)
is referenced as follows:
.. code-block:: python
        bld = [('largeoffice', 'pst80', 0.4),
                ('midriseapartment', 'new', 0.3),
                ('custombuilding', 'pre80', 0.3)]
'''
```

Comparing results

In [1]:

```
## Writing the weather data on epw weather template downloaded from EnergyPlus weather data platform
from ladybug.epw import EPW
from pandas import DataFrame
from collections import OrderedDict
%config Completer.use_jedi = False
import pandas as pd
import numpy as np
import scipy
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
# PLotting Boxplots
epwFile9=EPW(r'D:\Weather_files\Observed_Nantes_2003.epw')
epwDataList=epwFile9.to_dict()['data_collections']
epwDataDict = OrderedDict()

for dataColumns in epwDataList:
    dataName=dataColumns['header']['data_type']['name']
    epwDataDict[dataName]=dataColumns['values']

epwDataFrame9 = DataFrame(epwDataDict)
epwDataFrame9.head(2)
```

Out[2]:

	Year	Month	Day	Hour	Minute	Uncertainty Flags	Dry Bulb Temperature	Dew Point Temperature	Relative Humidity	Atmospheric Station Pressure	...	Ceiling Height	Present Weather Observation	Present Weather Codes	Preci
0	2005	12	31	24	60	*?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?	-6.971197	-3.34356	98	100940	...	99999	9	999999999	
1	2005	1	1	1	60	*?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?	10.600000	9.70000	94	100870	...	99999	9	999999999	

2 rows × 35 columns

In [3]:

```
# PLotting Boxplots
epwFile7=EPW(r'D:\Weather_files\Observed_Nantes_2003_UWG_KM7_3.epw')
epwDataList=epwFile7.to_dict()['data_collections']
epwDataDict = OrderedDict()
```

```
for dataColumns in epwDataList:
    dataName=dataColumns['header']['data_type']['name']
    epwDataDict[dataName]=dataColumns['values']

epwDataFrame7 = DataFrame(epwDataDict)
epwDataFrame7.head(2)
```

Out[3]:

	Year	Month	Day	Hour	Minute	Uncertainty Flags	Dry Bulb Temperature	Dew Point Temperature	Relative Humidity	Atmospheric Station Pressure	...	Ceiling Height	Present Weather Observation	Present Weather Codes	Preci
0	2005	12	31	24	60	*?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?	0.9	-7.5	54	100940	...	99999	9	999999999	
1	2005	1	1	1	60	*?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?	10.7	9.7	93	100870	...	99999	9	999999999	

2 rows × 35 columns

In [4]:

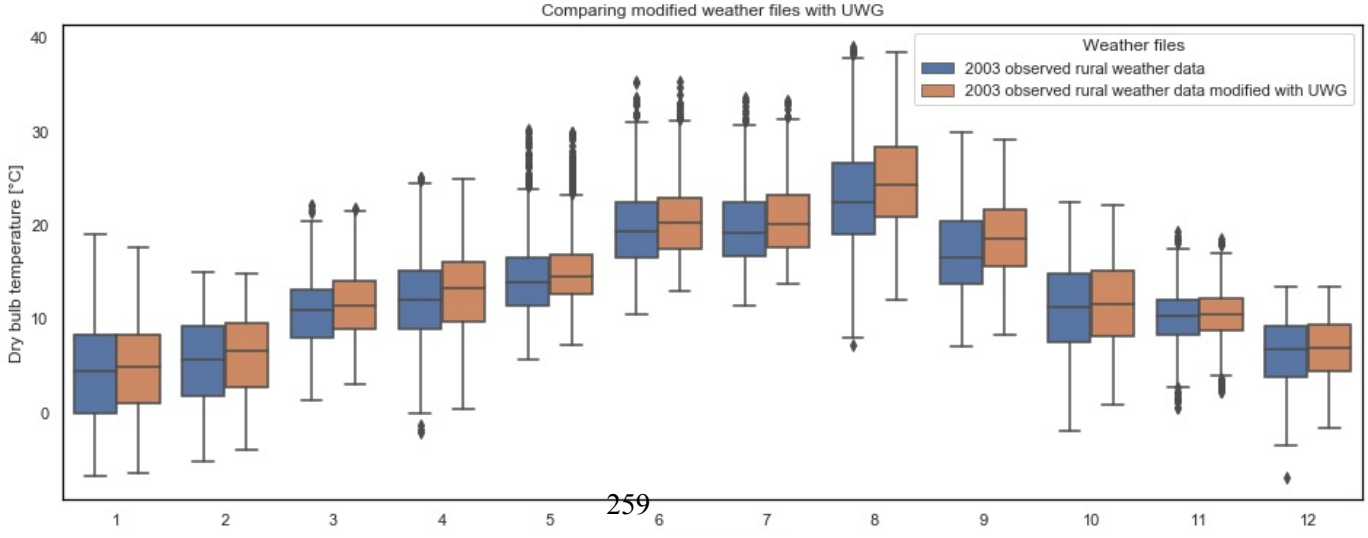
```
Wdata = pd.DataFrame()
Wdata['2003 observed rural weather data'] = epwDataFrame9['Dry Bulb Temperature']
Wdata['2003 observed rural weather data modified with UWG'] = epwDataFrame7['Dry Bulb Temperature']
```

In [5]:

```
# PLotting Boxplots of months
epwFile0=EPW(r'D:\Weather_files\Observed_Nantes_2003.epw')
epwDataList=epwFile0.to_dict()['data_collections']
epwDataDict = OrderedDict()

for dataColumns in epwDataList:
    dataName=dataColumns['header']['data_type']['name']
    epwDataDict[dataName]=dataColumns['values']

epwDataFrame0 = DataFrame(epwDataDict)
a_clms = ['Month']
epwDataFrame0 = epwDataFrame0[a_clms]
epf=pd.concat([epwDataFrame0, Wdata], axis=1)
```

In [6]:

```
df_m = epf.melt(id_vars = 'Month',
                value_vars = ['2003 observed rural weather data',
                              '2003 observed rural weather data modified with UWG'],
                var_name = 'Weather files')
#
sns.set(rc={'figure.figsize':(16,6),'axes.edgecolor':'black','axes.facecolor':'w','axes.grid.which':'both'})

b = sns.boxplot(data =df_m,#.loc[(df_m['Month']==6) or (df_m['Month']==7) or (df_m['Month']==8)],
                hue = 'Weather files',
                x = 'Month',
                y = 'value').set(title='Comparing modified weather files with UWG',
                                 xlabel='Months of the year',
                                 ylabel= 'Dry bulb temperature [\N{DEGREE SIGN}C]')#'Global horizontal Irradiance
plt.grid()
plt.show()
```

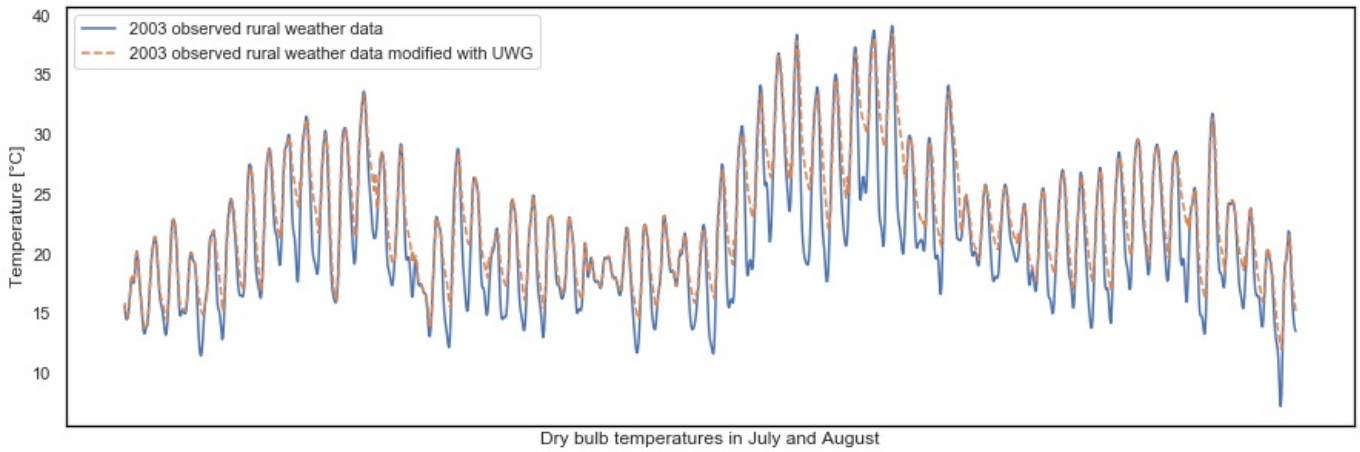


Months of the year

In [19]:
```
df_mean = df_m.groupby(by=["Month","Weather files"]).mean()
```

In [12]:
```
df_mean=df_mean.reindex()
```

In [22]:
```
# Set a range to visualise
zoomed=epf.loc[4345:5833]# 4344 = begining of July 5089 = Beginign of August, 5833 = beginign of september

plt.plot(zoomed.index, '2003 observed rural weather data', data=zoomed, linewidth=1.5,linestyle='-',label='2003 o
plt.plot(zoomed.index, '2003 observed rural weather data modified with UWG', data=zoomed, linewidth=1.5,linestyle

# Set the x axis label of the current axis.
plt.xlabel('Dry bulb temperatures in July and August')
# Set the y axis label of the current axis.
plt.ylabel('Temperature [\N{DEGREE SIGN}C]')
# Set a title
#plt.grid()
plt.xticks([])
plt.rcParams['axes.facecolor'] = 'w'
plt.rcParams['axes.edgecolor']= 'black'
plt.rcParams["figure.figsize"] = (15,5)

plt.legend()
plt.show()
```



In [ ]:
```

```

Appendix 3-6:

- Calculating heatwave presence in weather file : a step-by-step process using a temperature based index

In [1]:

```python
## Writing the weather data on epw weather template downloaded from EnergyPlus weather data platform
from ladybug.epw import EPW
from pandas import DataFrame
from collections import OrderedDict
%config Completer.use_jedi = False
import pandas as pd
import numpy as np
import scipy
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.dates as mdates

from pandas import datetime
from matplotlib import pyplot

# creating a series of datatime
from datetime import date, datetime, timedelta
from pandas import Timestamp
```

In [2]:

```python
# File path to EPW file
epwFile=EPW(r'D:\Weather_files\Nantes-RCP8-5_2050.epw')
epwDataList=epwFile.to_dict()['data_collections']
epwDataDict = OrderedDict()

for dataColumns in epwDataList:
    dataName=dataColumns['header']['data_type']['name']
    epwDataDict[dataName]=dataColumns['values']

epwDataFrame = DataFrame(epwDataDict)
epwDataFrame.head(2)
```

Out[2]:

	Year	Month	Day	Hour	Minute	Uncertainty Flags	Dry Bulb Temperature	Dew Point Temperature	Relative Humidity	Atmospheric Station Pressure	...	Ceiling Height	Present Weather Observation	Present Weather Codes	Preci
0	2050	12	31	24	60	*?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?	5.7	5.3	97	100879	...	99999	9	999999999	
1	2050	1	1	1	60	*?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?	12.1	10.6	91	100879	...	99999	9	999999999	

2 rows × 35 columns

In [3]:

```python
# File path to EPW file
epwFile3=EPW(r'D:\Weather_files\FRA_Nantes_072220_IWEC.epw')
epwDataList=epwFile3.to_dict()['data_collections']
epwDataDict = OrderedDict()

for dataColumns in epwDataList:
    dataName=dataColumns['header']['data_type']['name']
    epwDataDict[dataName]=dataColumns['values']

epwDataFrame3 = DataFrame(epwDataDict)
epwDataFrame3.head(2)
```

Out[3]:

	Year	Month	Day	Hour	Minute	Uncertainty Flags	Dry Bulb Temperature	Dew Point Temperature	Relative Humidity	Atmospheric Station Pressure	...	Ceiling Height	Present Weather Observation	Pr We
0	1987	12	31	24	60	C9C9C9C9*0?9?9?9?9? 9?9? 9*0B8B8B8B8*0*0E8*0*0	1.7	0.9	94	100700	...	22000	9	9999
1	1986	1	1	1	60	C9C9C9C9*0?9?9?9?9? 9?9? 9A7A7A7A7A7*0E8*0*0	-0.3	-1.5	91	100500	...	22000	9	9999

2 rows × 35 columns

In [4]:

```python
# File path to EPW file
```

```
epwFile2=EPW(r'D:\Weather_files\Nantes-historic.epw')
epwDataList=epwFile2.to_dict()['data_collections']
epwDataDict = OrderedDict()

for dataColumns in epwDataList:
    dataName=dataColumns['header']['data_type']['name']
    epwDataDict[dataName]=dataColumns['values']

epwDataFrame2 = DataFrame(epwDataDict)
epwDataFrame2.head(1)
```

Out[4]:

	Year	Month	Day	Hour	Minute	Uncertainty Flags	Dry Bulb Temperature	Dew Point Temperature	Relative Humidity	Atmospheric Station Pressure	...	Ceiling Height	Present Weather Observation	Present Weather Codes	Preci
0	2005	12	31	24	60	*?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?	4.2	2.7	90	100705	...	99999	9	999999999	

1 rows × 35 columns

In [5]:

```
# File path to EPW file
epwFile4=EPW(r'D:\Weather_files\Nantes_2000_2019_Contemporary.epw')
epwDataList=epwFile4.to_dict()['data_collections']
epwDataDict = OrderedDict()

for dataColumns in epwDataList:
    dataName=dataColumns['header']['data_type']['name']
    epwDataDict[dataName]=dataColumns['values']

epwDataFrame4 = DataFrame(epwDataDict)
epwDataFrame4.head(2)
```

Out[5]:

	Year	Month	Day	Hour	Minute	Uncertainty Flags	Dry Bulb Temperature	Dew Point Temperature	Relative Humidity	Atmospheric Station Pressure	...	Ceiling Height	Present Weather Observation	Present Weather Codes	Preci
0	2005	12	31	24	60	*?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?	5.7	5.5	98	100879	...	99999	9	999999999	
1	2005	1	1	1	60	*?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?	11.0	9.2	88	100879	...	99999	9	999999999	

2 rows × 35 columns

In [6]:

```
# File path to EPW file
epwFile5=EPW(r'D:\Weather_files\CNRM2040_2070.epw')
epwDataList=epwFile5.to_dict()['data_collections']
epwDataDict = OrderedDict()

for dataColumns in epwDataList:
    dataName=dataColumns['header']['data_type']['name']
    epwDataDict[dataName]=dataColumns['values']

epwDataFrame5 = DataFrame(epwDataDict)
epwDataFrame5.head(2)
```

Out[6]:

	Year	Month	Day	Hour	Minute	Uncertainty Flags	Dry Bulb Temperature	Dew Point Temperature	Relative Humidity	Atmospheric Station Pressure	...	Ceiling Height	Present Weather Observation	Present Weather Codes	Preci
0	2067	12	31	24	60	*?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?	-1.500769	-2.61	92	101217	...	99999	9	999999999	
1	2070	1	1	1	60	*?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?	2.516901	0.56	87	101297	...	99999	9	999999999	

2 rows × 35 columns

In [7]:

```python
# File path to EPW file
epwFile6=EPW(r'D:\Weather_files\IPSL2040_2070.epw')
epwDataList=epwFile6.to_dict()['data_collections']
epwDataDict = OrderedDict()

for dataColumns in epwDataList:
    dataName=dataColumns['header']['data_type']['name']
    epwDataDict[dataName]=dataColumns['values']

epwDataFrame6 = DataFrame(epwDataDict)
epwDataFrame6.head(4)
```

Out[7]:

	Year	Month	Day	Hour	Minute	Uncertainty Flags	Dry Bulb Temperature	Dew Point Temperature	Relative Humidity	Atmospheric Station Pressure	...	Ceiling Height	Present Weather Observation	Present Weather Codes	Preci
0	2054	12	31	24	60	*?	8.976160	6.31	83	102282	...	99999	9	999999999	
1	2049	1	1	1	60	*?	4.558191	3.91	96	102329	...	99999	9	999999999	
2	2049	1	1	2	60	*?	3.399426	2.91	97	102376	...	99999	9	999999999	
3	2049	1	1	3	60	*?	3.313419	2.97	98	102423	...	99999	9	999999999	

4 rows × 35 columns

In [8]:

```python
# File path to EPW file
epwFile7=EPW(r'D:\Weather_files\MPI_REMO.epw')
epwDataList=epwFile7.to_dict()['data_collections']
epwDataDict = OrderedDict()

for dataColumns in epwDataList:
    dataName=dataColumns['header']['data_type']['name']
    epwDataDict[dataName]=dataColumns['values']

epwDataFrame7 = DataFrame(epwDataDict)
epwDataFrame7.head(4)
```

Out[8]:

	Year	Month	Day	Hour	Minute	Uncertainty Flags	Dry Bulb Temperature	Dew Point Temperature	Relative Humidity	Atmospheric Station Pressure	...	Ceiling Height	Present Weather Observation	Present Weather Codes	Preci
0	2065	12	31	24	60	*?	10.496912	10.17	98	99949	...	99999	9	999999999	
1	2064	1	1	1	60	*?	12.984369	12.08	94	99884	...	99999	9	999999999	
2	2064	1	1	2	60	*?	12.483222	11.62	94	99818	...	99999	9	999999999	
3	2064	1	1	3	60	*?	12.673921	11.84	95	99752	...	99999	9	999999999	

4 rows × 35 columns

In [9]:

```
# PLotting Boxplots of months
epwFile0=EPW(r'D:\Weather_files\IPSL_CM5A_SMHI_2003.epw')
epwDataList=epwFile0.to_dict()['data_collections']
epwDataDict = OrderedDict()

for dataColumns in epwDataList:
    dataName=dataColumns['header']['data_type']['name']
    epwDataDict[dataName]=dataColumns['values']

epwDataFrame0 = DataFrame(epwDataDict)
epwDataFrame0.head(3)
```

Out[9]:

	Year	Month	Day	Hour	Minute	Uncertainty Flags	Dry Bulb Temperature	Dew Point Temperature	Relative Humidity	Atmospheric Station Pressure	...	Ceiling Height	Present Weather Observation	Present Weather Codes	Preci
0	2003	12	31	24	60	*?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?	7.184656	3.86	79	103771	...	99999	9	999999999	
1	2003	1	1	1	60	*?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?	8.361597	8.06	98	103734	...	99999	9	999999999	
2	2003	1	1	2	60	*?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?	8.241332	7.88	98	103697	...	99999	9	999999999	

3 rows × 35 columns

In [10]:

```
# File path to EPW file
epwFile9=EPW(r'D:\Weather_files\Observed_Nantes_2003.epw')
epwDataList=epwFile9.to_dict()['data_collections']
epwDataDict = OrderedDict()

for dataColumns in epwDataList:
    dataName=dataColumns['header']['data_type']['name']
    epwDataDict[dataName]=dataColumns['values']

epwDataFrame9 = DataFrame(epwDataDict)
epwDataFrame9.head()
```

Out[10]:

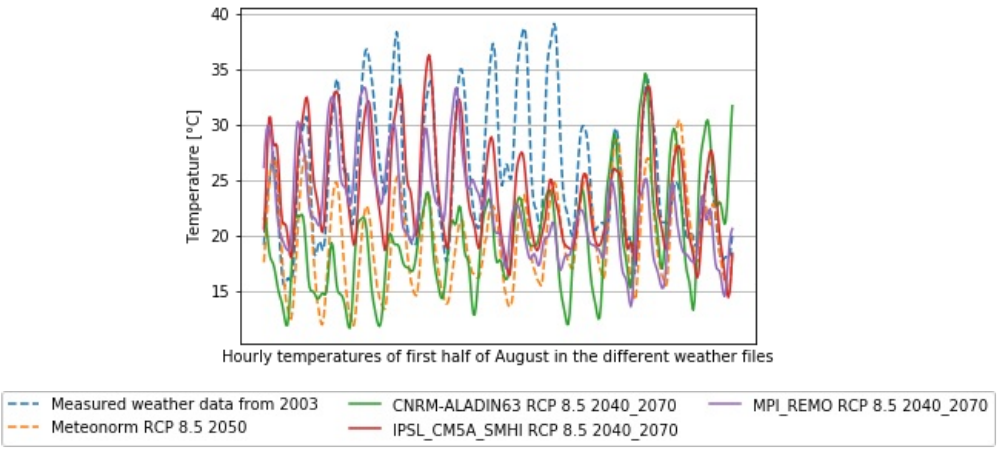
	Year	Month	Day	Hour	Minute	Uncertainty Flags	Dry Bulb Temperature	Dew Point Temperature	Relative Humidity	Atmospheric Station Pressure	...	Ceiling Height	Present Weather Observation	Present Weather Codes	Preci
0	2005	12	31	24	60	*?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?	-6.971197	-3.343560	98	100940	...	99999	9	999999999	
1	2005	1	1	1	60	*?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?	10.600000	9.700000	94	100870	...	99999	9	999999999	
2	2005	1	1	2	60	*?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?	11.853135	11.103876	94	100800	...	99999	9	999999999	
3	2005	1	1	3	60	*?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?	11.626363	10.860776	95	100730	...	99999	9	999999999	
4	2005	1	1	4	60	*?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?*?*?*? *?*?	11.200000	10.400000	95	100620	...	99999	9	999999999	

5 rows × 35 columns

In [11]:
```
# Creating a dataframe where only dry bulb temperature/horizontal irradiance/relative humidity
Wdata = pd.DataFrame()
Wdata['Measured_weather_data_of_2003'] = epwDataFrame9['Dry Bulb Temperature']
Wdata['Meteonorm_RCP_8-5_2050'] = epwDataFrame['Dry Bulb Temperature']
Wdata['CNRM-ALADIN63_RCP_8-5_2040_2070'] = epwDataFrame5['Dry Bulb Temperature']
Wdata['IPSL_CM5A_SMHI_RCP_8-5_2040_2070'] = epwDataFrame6['Dry Bulb Temperature']
Wdata['MPI_REMO_RCP_8-5_2040_2070'] = epwDataFrame7['Dry Bulb Temperature']
```

In [13]:
```
#df_melt.tail()
```

In [14]:
```
# Set a range to visualise
zoomed=Wdata.loc[5097:5457]
plt.plot(zoomed.index, 'Measured_weather_data_of_2003', data=zoomed, linewidth=1.5,linestyle='--', label="Measure
plt.plot(zoomed.index, 'Meteonorm_RCP_8-5_2050', data=zoomed, linewidth=1.5,linestyle='--', label="Meteonorm RCP
plt.plot(zoomed.index, 'CNRM-ALADIN63_RCP_8-5_2040_2070', data=zoomed, linewidth=1.5,linestyle='-', label="CNRM-A
plt.plot(zoomed.index, 'IPSL_CM5A_SMHI_RCP_8-5_2040_2070', data=zoomed, linewidth=1.5,linestyle='-', label="IPSL_
plt.plot(zoomed.index, 'MPI_REMO_RCP_8-5_2040_2070', data=zoomed, linewidth=1.5,linestyle='-', label="MPI_REMO RC
# Set the x axis label of the current axis.
plt.xlabel('Hourly temperatures of first half of August in the different weather files')
# Set the y axis label of the current axis.
plt.ylabel('Temperature [\N{DEGREE SIGN}C]')#'Global horizontal Irradiance [$W/m^2$]')#)
# Set a title
#plt.title('Interpolation of 3hr weather data to hourly')
plt.xticks([])
#plt.rcParams['axes.facecolor'] = 'w'
plt.rcParams['axes.edgecolor']= 'black'
plt.rcParams["figure.figsize"] = (15,5)
plt.grid()
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.12),ncol=3)
plt.show()
```



In [15]:
```
# creating a DataFrame of year, month, day and hours
def datetime_range(start,end,delta):
    current=start
    if not isinstance(delta,timedelta):
        delta = timedelta(**delta)
    while current < end:
        yield current
        current +=delta
```

In [16]:
```
# Dataframe of time and dates: each subsequent row difference is 1 hour
start = datetime(2003,0o1,0o1)
end = datetime(2004,0o1,0o1)

df = pd.DataFrame({'DateTime':datetime_range(start,end,{'days':0,'hours':1})})

# Removing leap day data if year is a leap year
df = df[~((df['DateTime'].dt.month == 2) & (df['DateTime'].dt.day == 29))]
# reindex in place
df= df.reset_index()
## Drop index column
df=df.drop(['index'], axis=1)
# visualize
print(df.head(2))
```

```
print(df.tail(2))
```

```
            DateTime
0 2003-01-01 00:00:00
1 2003-01-01 01:00:00
               DateTime
8758 2003-12-31 22:00:00
8759 2003-12-31 23:00:00
```

In [17]:
```
df=pd.concat([df, Wdata], axis=1)
```

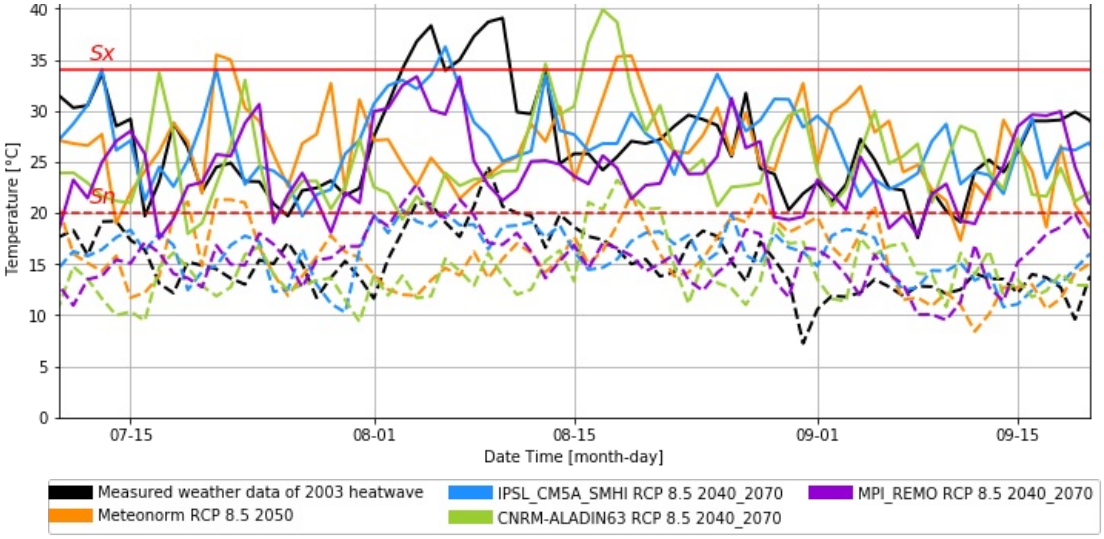
In [18]:
```
df=df.set_index('DateTime')
```

In [19]:
```
### Daily maximum temperatures
df1=df.resample('D').max()
### daily minimum temperatures
df2=df.resample('D').min()
```

In [20]:
```
df3=pd.merge(df1, df2, right_index=True, left_index=True)
```

In [21]:
```
locator = mdates.MonthLocator() # every month
# Specify the format - %b gives us Jan, Feb...
fmt = mdates.DateFormatter('%b')
```

In [93]:
```
### PLotting heatwave days
plt.plot(df3.index, 'Measured_weather_data_of_2003_x', data=df3, color='black',linewidth=2,linestyle='-',label='M
plt.plot(df3.index, 'Measured_weather_data_of_2003_y', data=df3, color='black',linewidth=2,linestyle='--',label='
##
plt.plot(df3.index, 'Meteonorm_RCP_8-5_2050_x', data=df3, color='darkorange',linewidth=2,linestyle='-',label='Met
plt.plot(df3.index, 'Meteonorm_RCP_8-5_2050_y', data=df3, color='darkorange',linewidth=2,linestyle='--',label='_n
###
plt.plot(df3.index, 'IPSL_CM5A_SMHI_RCP_8-5_2040_2070_x', data=df3, color='dodgerblue',linewidth=2,linestyle='-',
plt.plot(df3.index, 'IPSL_CM5A_SMHI_RCP_8-5_2040_2070_y', data=df3, color='dodgerblue',linewidth=2,linestyle='--'
###
##
plt.plot(df3.index, 'CNRM-ALADIN63_RCP_8-5_2040_2070_x', data=df3, color='yellowgreen',linewidth=2,linestyle='-',
plt.plot(df3.index, 'CNRM-ALADIN63_RCP_8-5_2040_2070_y', data=df3, color='yellowgreen',linewidth=2,linestyle='--'
###
##
plt.plot(df3.index, 'MPI_REMO_RCP_8-5_2040_2070_x', data=df3, color='darkviolet',linewidth=2,linestyle='-',label=
plt.plot(df3.index, 'MPI_REMO_RCP_8-5_2040_2070_y', data=df3, color='darkviolet',linewidth=2,linestyle='--',label
###
###threshhold lines for operational heatwave detection across france since 2006
plt.axhline(y=34, color='r',linewidth=1.5, linestyle='-',label='_nolegend_')
plt.axhline(y=20, color='r',linewidth=1.5, linestyle='--',label='_nolegend_')

X = plt.gca().xaxis
#X.set_major_locator(locator)
# Specify formatter
X.set_major_formatter(mdates.DateFormatter('%m-%d'))

plt.xlabel('Date Time [month-day]')
# Set the y axis label of the current axis.
plt.ylabel('Temperature [\N{DEGREE SIGN}C]')
# Set a title
#plt.title('IBM_max & IBM_min for heatwaves')
##
plt.ylim(0, 41)
##setting x-axis range
plt.xlim(pd.Timestamp('2003-07-10'), pd.Timestamp('2003-09-20'))
##
plt.rcParams['axes.edgecolor']= 'black'
plt.rcParams["figure.figsize"] = (12,5)
plt.grid()

plt.text(Timestamp("2003-07-12"), 35,"$Sx$",fontsize=14,color='r')
plt.text(Timestamp("2003-07-12"), 21,"$Sn$",fontsize=14,color='r')

leg = plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.13),ncol=3)
for l in leg.legendHandles:
    l.set_linewidth(9)

plt.show()
```



In [23]:
```
##  Output table
###Average of three consecutive days
```

In [25]:
```
df4=df3.reset_index()
```

Re-run all functions from here after replacing Columns of Maximum and minimum temperatures and parameters

In [26]:
```
df4.columns
```

Out[26]:
```
Index(['DateTime', 'Measured_weather_data_of_2003_x',
       'Meteonorm_RCP_8-5_2050_x', 'CNRM-ALADIN63_RCP_8-5_2040_2070_x',
       'IPSL_CM5A_SMHI_RCP_8-5_2040_2070_x', 'MPI_REMO_RCP_8-5_2040_2070_x',
       'Measured_weather_data_of_2003_y', 'Meteonorm_RCP_8-5_2050_y',
       'CNRM-ALADIN63_RCP_8-5_2040_2070_y',
       'IPSL_CM5A_SMHI_RCP_8-5_2040_2070_y', 'MPI_REMO_RCP_8-5_2040_2070_y'],
      dtype='object')
```

In [27]:
```
### replace the names of columns with your target weather data
# those that end with x are Maximum daily temperature and those that end with y are minimum daily temperatures
df4=df4.rename(columns={'Meteonorm_RCP_8-5_2050_x':'Meteonorm_RCP_85_2050_x',
                        'CNRM-ALADIN63_RCP_8-5_2040_2070_x':'CNRM_ALADIN63_RCP_85_2040_2070_x',
                        'IPSL_CM5A_SMHI_RCP_8-5_2040_2070_x':'IPSL_CM5A_SMHI_RCP_85_2040_2070_x',
                        'MPI_REMO_RCP_8-5_2040_2070_x':'MPI_REMO_RCP_85_2040_2070_x',
                        'Meteonorm_RCP_8-5_2050_y':'Meteonorm_RCP_85_2050_y',
                        'CNRM-ALADIN63_RCP_8-5_2040_2070_y':'CNRM_ALADIN63_RCP_85_2040_2070_y',
                        'IPSL_CM5A_SMHI_RCP_8-5_2040_2070_y':'IPSL_CM5A_SMHI_RCP_85_2040_2070_y',
                        'MPI_REMO_RCP_8-5_2040_2070_y':'MPI_REMO_RCP_85_2040_2070_y'})

#column = ['DateTime','Energyplus_IWEC_EPW_x', 'Meteonorm_historic_1961_1991_x',
#      'Meteonorm_contemporary_2000_2019_x',]#
#df4 = df4[column]
```

In [28]:
```
### Selecting the weather file we decide to analyzie
WeatherMax=df4.Measured_weather_data_of_2003_x
WeatherMin=df4.Measured_weather_data_of_2003_y
IBMax=34
IBMin=20
```

In [29]:
```
### Moving Average of three consective days (maximums)
Aver=[]
for i, row in df4.iterrows():
    if i ==0:
        avera=(WeatherMax[i]+WeatherMax[i+1])/2
    elif i ==364:
        avera=(WeatherMax[i-1]+WeatherMax[i])/2
    else:
        avera=(WeatherMax[i-1]+WeatherMax[i]+WeatherMax[i+1])/3
    Aver.append(avera)
df4['MaxAverage']= Aver
```

```
#t
```

In [30]:
```
### Moving Average of three consective days (minimums)
Aver=[]
for i, row in df4.iterrows():
    if i ==0:
        avera=(WeatherMin[i]+WeatherMin[i+1])/2
    elif i ==364:
        avera=(WeatherMin[i-1]+WeatherMin[i])/2
    else:
        avera=(WeatherMin[i-1]+WeatherMin[i]+WeatherMin[i+1])/3
    Aver.append(avera)
df4['MinAverage']= Aver
#
```

In [31]:
```
#print(df4['DateTime'].loc[((df4['MaxAverage']>=IBMax) & (df4['MinAverage']>=IBMin))])
```

In [32]:
```
#print(df4.loc[((df4['MaxAverage']>=IBMax) & (df4['MinAverage']>=IBMin))])
```

In [33]:
```
###Debut (starting dates of heatwaves)
Aver=[]
for i, row in df4.iterrows():
    if i ==364:
        db=0
    elif ((df4['MaxAverage'][i+1]>=IBMax) & (df4['MinAverage'][i+1]>=IBMin)):
        db=1
    else:
        db=0
    Aver.append(db)
df4['Debut']= Aver
#
```

In [34]:
```
###Middle (middle days of heatwaves )
Aver=[]
for i, row in df4.iterrows():
    if ((df4['MaxAverage'][i]>=IBMax) & (df4['MinAverage'][i]>=IBMin)):
        db=1
    else:
        db=0
    Aver.append(db)
df4['mid']= Aver
#
```

In [35]:
```
###Fin (end dates of heatwaves)
Aver=[]
for i, row in df4.iterrows():
    if i ==0:
        db=0
    elif ((df4['MaxAverage'][i-1]>=IBMax) & (df4['MinAverage'][i-1]>=IBMin)):
        db=1
    else:
        db=0
    Aver.append(db)
df4['Fin']= Aver
#
```

In [36]:
```
###grouped (combined start date, middle dates, and end dates of heatwaves)
Aver=[]
for i, row in df4.iterrows():
    if ((df4['Debut'][i]>0) or (df4['mid'][i]>0) or (df4['Fin'][i]>0)):
        db=1
    else:
        db=0
    Aver.append(db)
df4['HW']= Aver
#
```

In [37]:
```
#df4=df4.drop(['Fin'], axis=1)
```

In [39]:
```
from itertools import groupby, count
```

In [40]:
```
# Group and count consecutive days that meet IBMax and IBMin criteria - HW>0
```

```
# Group and count consecutive days that meet TBmax and TBmin criteria , HW>0
def intervals(data):
    out = []
    counter = count()

    for key, group in groupby(data, key = lambda x: x-next(counter)):
        block = list(group)
        out.append(block)
    return out
```

In [41]:
```
intervals(df4.index[df4['HW']>0].tolist())
```

Out[41]:
```
[[214, 215, 216, 217, 218, 219, 220, 221, 222]]
```

In [43]:
```
##Nuber of heawaves in weather file
NumHW=len(intervals(df4.index[df4['HW']>0].tolist()))
```

In [44]:
```
###Indices of days when peak temperature happens
h=[]
for i in range(NumHW):
    b=df4.index[WeatherMax==max(WeatherMax.loc[intervals(df4.index[df4['HW']>0].tolist())[i]])].tolist()[0]
    h.append(b)
```

In [45]:
```
###Peak temperatures of each heatwave
g=[]
for i in range(NumHW):
    a=WeatherMax.loc[WeatherMax==max(WeatherMax.loc[intervals(df4.index[df4['HW']>0].tolist())[i]])].tolist()[0]
    g.append(round(a,2))
```

In [46]:
```
### Duration of each Heatwave
k=[]
for i in range(NumHW):
    c=len(intervals(df4.index[df4['HW']>0].tolist())[i])
    k.append(c)
```

In [47]:
```
### Mean Severity of maximum temperatures
y=[]
for i in range(NumHW):
    s=sum(WeatherMax.loc[intervals(df4.index[df4['HW']>0].tolist())[i]])/len(intervals(df4.index[df4['HW']>0].tol
    y.append(round(s,2))
```

In [48]:
```
### Mean Severity of minimum temperatures
w=[]
for i in range(NumHW):
    s=sum(WeatherMin.loc[intervals(df4.index[df4['HW']>0].tolist())[i]])/len(intervals(df4.index[df4['HW']>0].tol
    w.append(round(s,2))
```

In [49]:
```
### Cumulative Severity of maximum temperatures
z=[]
for i in range(NumHW):
    f=sum(WeatherMax.loc[intervals(df4.index[df4['HW']>0].tolist())[i]])
    z.append(round(f,2))
```

In [50]:
```
### Cumulative Severity of minimum temperatures
p=[]
for i in range(NumHW):
    f=sum(WeatherMin.loc[intervals(df4.index[df4['HW']>0].tolist())[i]])
    p.append(round(f,2))
```

In [51]:
```
print('Indices = '+ str(intervals(df4.index[df4['HW']>0].tolist())))
print('Num of HW = ' +str(NumHW))
print('Duration of each Heat wave = '+str(k))
print('Julian day number of peak temperature ='+str(h))
print('Peak temperatures ='+str(g))
print('Mean severity of maximum temperatures = '+str(y))
print('Mean severity of minimum temperatures = '+str(w))
print('Cumulative severity of maximum temperatures = '+str(z))
print('Cumulative severity of minimum temperatures = '+str(p))
```

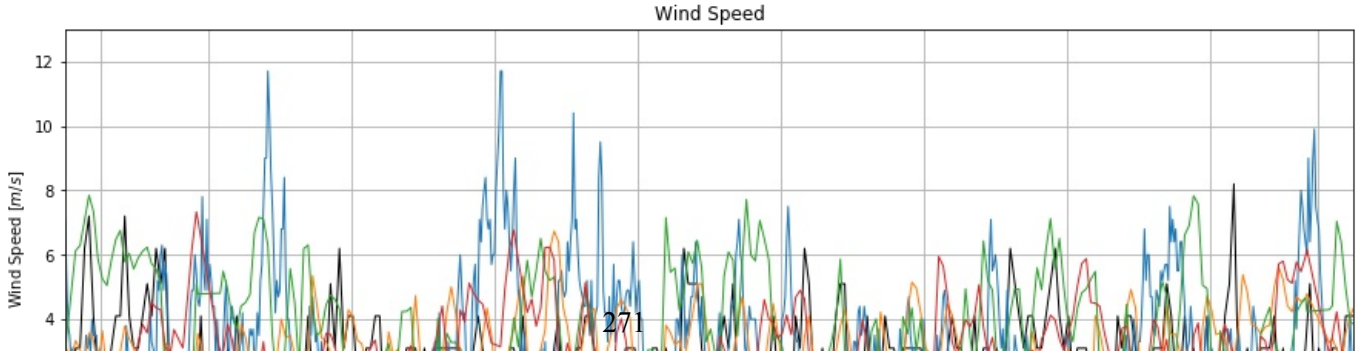
```
Indices = [[214, 215, 216, 217, 218, 219, 220, 221, 222]]
```

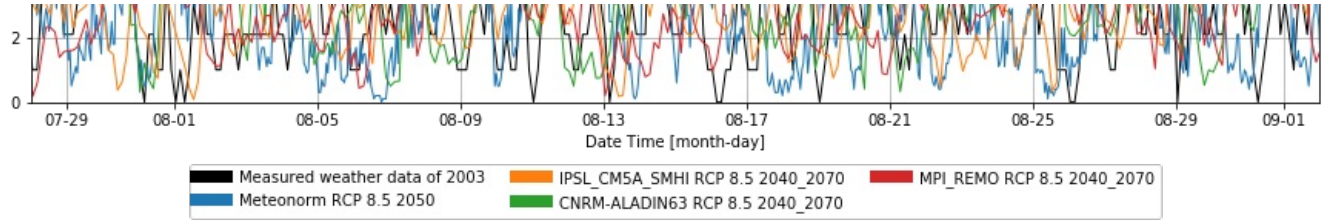
```
Num of HW = 1
Duration of each Heat wave = [9]
Julian day number of peak temperature =[221]
Peak temperatures =[39.1]
Mean severity of maximum temperatures = [35.91]
Mean severity of minimum temperatures = [20.29]
Cumulative severity of maximum temperatures = [323.23]
Cumulative severity of minimum temperatures = [182.62]
```

Plotting solar radiation, relative humidity and wind speed

In [52]:

```
# Global Horizontal Radiation
# Dataframe of time and dates: each subsequent row difference is 3 hours
start = datetime(2003,0o1,0o1)
end = datetime(2004,0o1,0o1)
df4 = pd.DataFrame({'DateTime':datetime_range(start,end,{'days':0,'hours':1})})
# Removing leap day data if year is a leap year
df4 = df4[~((df4['DateTime'].dt.month == 2) & (df4['DateTime'].dt.day == 29))]
# reindex in place
df4= df4.reset_index()
## Drop index column
df4=df4.drop(['index'], axis=1)
# visualize
df4['Measured_weather_data_of_2003'] = epwDataFrame9['Wind Speed']
df4['Meteonorm_RCP_8-5_2050'] = epwDataFrame['Wind Speed']
df4['CNRM-ALADIN63_RCP_8-5_2040_2070'] = epwDataFrame5['Wind Speed']
df4['IPSL_CM5A_SMHI_RCP_8-5_2040_2070'] = epwDataFrame6['Wind Speed']
df4['MPI_REMO_RCP_8-5_2040_2070'] = epwDataFrame7['Wind Speed']
###ser index datetime
df4=df4.set_index('DateTime')
#df3['IPSL_CM5A_SMHI_2003'] = epwDataFrame0['Dry Bulb Temperature']
X = plt.gca().xaxis
#####
plt.plot(df4.index, 'Measured_weather_data_of_2003',color='black', data=df4, linewidth=1,linestyle='-',label='Mea
#
plt.plot(df4.index, 'Meteonorm_RCP_8-5_2050', data=df4, linewidth=1,linestyle='-',label='Meteonorm RCP 8.5 2050')
#plt.plot
plt.plot(df4.index, 'IPSL_CM5A_SMHI_RCP_8-5_2040_2070', data=df4, linewidth=1,linestyle='-',label='IPSL_CM5A_SMHI
##
plt.plot(df4.index, 'CNRM-ALADIN63_RCP_8-5_2040_2070', data=df4, linewidth=1,linestyle='-',label='CNRM-ALADIN63 R
##
plt.plot(df4.index, 'MPI_REMO_RCP_8-5_2040_2070', data=df4, linewidth=1,linestyle='-',label='MPI_REMO RCP 8.5 204

###threshhold lines
#X.set_major_locator(locator)
## Specify formatter
X.set_major_formatter(mdates.DateFormatter('%m-%d'))

plt.xlabel('Date Time [month-day]')
# Set the y axis label of the current axis.
plt.ylabel('Wind Speed [$m/s$]')#'Temperature [\N{DEGREE SIGN}C]')#'Global horizontal Irradiance [$W/m^2$]')#)
# Set a title
plt.title('Wind Speed')
plt.ylim(0, 13)
plt.xlim(pd.Timestamp('2003-07-28'), pd.Timestamp('2003-09-02'))
##
plt.rcParams['axes.edgecolor']= 'black'
plt.rcParams["figure.figsize"] = (15,5)
plt.grid()

leg = plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.13),ncol=3)
for l in leg.legendHandles:
    l.set_linewidth(8)

plt.show()
```





Calculating HDD and CDD in each weather file

In [54]:
```
### Daily average temperatures
df_DD=df.resample('D').mean()
```

In [56]:
```
##Heating degree days
HDD=[]
for column in df_DD[['Measured_weather_data_of_2003', 'Meteonorm_RCP_8-5_2050', 'CNRM-ALADIN63_RCP_8-5_2040_2070'
                    'IPSL_CM5A_SMHI_RCP_8-5_2040_2070','MPI_REMO_RCP_8-5_2040_2070']]:
    ComT=[]
    for i, row in df_DD.iterrows():
        if df_DD[column][i]<15.001:
            TY = 18-(df_DD[column][i])
        else:
            TY = 0
        ComT.append(TY)
    Adap=round(sum(ComT))
    HDD.append(Adap)
HDD_DataFrame=pd.DataFrame(HDD,columns=['HeatingDD'])
HDD_DataFrame.T
```

Out[56]:

	0	1	2	3	4
HeatingDD	2106.0	1741.0	2365.0	1873.0	1595.0

In [57]:
```
##Cooling degree days
CDD=[]
for column in df_DD[['Measured_weather_data_of_2003', 'Meteonorm_RCP_8-5_2050', 'CNRM-ALADIN63_RCP_8-5_2040_2070'
                    'IPSL_CM5A_SMHI_RCP_8-5_2040_2070','MPI_REMO_RCP_8-5_2040_2070']]:
    ComT=[]
    for i, row in df_DD.iterrows():
        if df_DD[column][i]>23.99:
            TY =(df_DD[column][i])-21
        else:
            TY = 0
        ComT.append(TY)
    Adap=round(sum(ComT))
    CDD.append(Adap)
CDD_DataFrame=pd.DataFrame(CDD,columns=['CoolingDD'])
CDD_DataFrame.T
```

Out[57]:

	0	1	2	3	4
CoolingDD	103.0	89.0	62.0	67.0	26.0

In [ ]:
```
```

Appendix 4-1: list of heat stress indices studied by Epstein and Moran

Number	Year	Index	Author(s)
1	1905	Wet-bulb temperature (Tw)	Halden et al
2	1916	Katatharmometer	Hill et al
3	1923	Effective temperature (ET)	Hougton & Yaglou
4	1929	Equivalent temperature (Teq)	Dufton et al
5	1932	Corrected effective temperature (CET)	Vernon & Warner
6	1937	Operative temperature (OpT)	Winslow et al
7	1945	Thermal acceptance ratio (TAR)	Ionides et al
8	1945	Index of physiological effect (E,)	Robinson et al
9	1946	Corrected effective temperature (CET)	Bedford
10	1947	Predicted 4-h sweat rate (P4SR)	McArdel et al
11	1948	Resultant temperature (RT)	Missenard et al
12	1950	Craig index (I)	Craigs
13	1955	Heat stress index (HIS)	Belding & Hatch
14	1957	Wet-bulb globe temperature (WBGT)	Yaglou & Mimard
15	1957	Oxford index (WD)	Lind & Hellen
16	1957	Discomfort index (DT)	Thom
17	1958	Thermal strain index (TST)	Lee & Henschel
18	1959	Discomfort index (DI)	Tennebaum et al
19	1960	Cumulative Discomfort index (CumDI)	Tennebaum et al
20	1960	Index of physiological strain (I)	Hall & Polte’
21	1962	Index of thermal stress (ITS)	Givoni
22	1966	Heat strain index (corrected) (HS])	McKarns & Brief
23	1966	Prediction of heart rate (HR)	Fuller & Brouha
24	1967	Effective radiant field (ERF)	Gagge et al
25	1970	Predicted mean vote (PMV)	Fanger
26		Threshold limit value (TLV)	
27	1970	Prescriptive zone	Lind
28	1971	New effective temperature (ET")	Gagge et al
29	1971	Wet globe temperature (WGT)	Botsford
30	1971	Humid operative temperate	Nishi & Gagge
31	1972	Predicted body core temperature	Givoni & Goldman
32	1972	Skin wettedness	Kerslake
33	1973	Standard effective temperature (SET)	Gagge et al
34	1973	Predicted heart rate	Givoni & Goldman
35	1978	Skin wettedness	Gonzales et al
36	1979	Fighter index of thermal stress (FITS)	Nunneley & Stribley
37	1981	Effective beat strain index (EHSI)	Kamon & Ryan
38	1982	Predicted sweat Loss (m_{sw})	Shapiro et al
39	1984	Apparent temperature (AT)	Steadman et al
40	1985	Required sweating (SWreq)	ISO 7933
41	1986	Predicted mean vote (modified) (PMV)	Gagge et al
42	1996	Cumulative heat strain index (CHSI)	Frank et al
43	1998	Physiological strain index (PSI)/(PET)	Moran et al
44	1999	Modified discomfort index (MDI)	Moran et al
45	2001	Environmental stress index (ESI)	Moran et al
46	2005	Wet-bulb dry temperature (WBDT)	Wallace et al
47	2005	Relative humidity dry temperate (RHDT)	Wallace et al

Appendix 4-2:

- Post-processing of TRNSYS simulations to calculate monthly Energy demand, comfort and overheating indices

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
%config Completer.use_jedi = False
# creating a series of datatime
from datetime import date, datetime, timedelta
# plotting time series data
import seaborn as sns

from matplotlib import pyplot
import matplotlib.dates as mdates

import csv
from math import *
```

In [2]:
```
pd.set_option("display.max_columns", None)
```

Reading outputs of TRNSYS simulation

In [3]:
```
##Output file name from simulation number: of Type0
SimulationNumber='ype6_4_A1_W2_O2'
```

In [4]:
```
file=pd.read_excel(r'D:\SimulationResultsCollected\MonthDayTime.xlsx')
```

In [5]:
```
file1=pd.read_csv(r'D:\3D_models\KM7_6_Area1_Win2_O2\TAIR_TOP.xls',encoding='windows-1252',delimiter='\s+',skipro
```

In [6]:
```
file1['TAIR_3_1'] = file1['TAIR_3_1'].apply(lambda x: x if x<60 else 17)
```

In [7]:
```
## read first csv
file2=pd.read_csv(r'D:\3D_models\KM7_6_Area1_Win2_O2\PMV_RH.xls',delimiter='\s+',skiprows=[1])
```

In [8]:
```
file3=pd.read_csv(r'D:\3D_models\KM7_6_Area1_Win2_O2\ENERGY.xls',delimiter='\s+',skiprows=[1])
```

In [9]:
```
file['TIME'] = file['TIME'].astype(np.int64)
file['Hour'] = file['Hour'].astype(str)
file1['TIME'] = file1['TIME'].astype(np.int64)
```

In [10]:
```
file.Hour.unique()
```

Out[10]:
```
array(['1899-12-30 00:00:00', '01:00:00', '02:00:00', '03:00:00',
       '04:00:00', '05:00:00', '06:00:00', '07:00:00', '08:00:00',
       '09:00:00', '10:00:00', '11:00:00', '12:00:00', '13:00:00',
       '14:00:00', '15:00:00', '16:00:00', '17:00:00', '18:00:00',
       '19:00:00', '20:00:00', '21:00:00', '22:00:00', '23:00:00'],
      dtype=object)
```

In [11]:
```
file['Hour'] = file['Hour'].replace('1899-12-30 00:00:00','00:00:00')
```

In [12]:
```
## merging file2 to file1
file1=pd.merge(file,file1,on='TIME')
```

In [13]:
```
## merging file2 to file1
file1=pd.merge(file1,file2)
```

In [14]:
```
##
file1=pd.merge(file1,file3)
```

In [15]:
```
file1.tail()
```

Out[15]:

	TIME	Month	Weekday	Day	Hour	Dry	TAIR_3_1	TAIR_2_2	TAIR_2_3	TAIR_2_4	TAIR_2_5	TAIR_1_6	TAIR_1_7	TAIR_1_8	TAIR_

8755	8755	12	Tuesday	31	19:00:00	9.2	17.0	17.0	17.0	17.0	17.0	17.0	17.0	17.0	1
8756	8756	12	Tuesday	31	20:00:00	9.1	17.0	17.0	17.0	17.0	17.0	17.0	17.0	17.0	1
8757	8757	12	Tuesday	31	21:00:00	9.5	17.0	17.0	17.0	17.0	17.0	17.0	17.0	17.0	1
8758	8758	12	Tuesday	31	22:00:00	9.6	17.0	17.0	17.0	17.0	17.0	17.0	17.0	17.0	1
8759	8759	12	Tuesday	31	23:00:00	9.3	17.0	17.0	17.0	17.0	17.0	17.0	17.0	17.0	1

In [16]:
```
#cols = ['TIME','Month', 'Weekday', 'Day',
#       'Hour','DryBulbTem', 'TAIR_2_1', 'TAIR_1_2', 'TAIR_1_3', 'TAIR_1_4',
#       'TOP_2_1', 'TOP_1_2', 'TOP_1_3', 'TOP_1_4','RELHUM_2_1',
#       'RELHUM_1_2', 'RELHUM_1_3', 'RELHUM_1_4', 'PMVLW_2_1', 'PMVLW_1_2',
#       'PMVLW_1_3', 'PMVLW_1_4', 'label','SQHEAT_1', 'SQCOOL_1',
#       'QHEAT_2_1', 'QHEAT_1_2', 'QHEAT_1_3', 'QHEAT_1_4', 'QCOOL_2_1',
#       'QCOOL_1_2', 'QCOOL_1_3', 'QCOOL_1_4']
```

In [17]:
```
#file1=file1[cols]
file1=file1.rename(columns={"Dry":"TAmb",
                            "SQCOOL_1":"SQCOOL",
                            "SQHEAT_1":"SQHEAT"})
```

In [18]:
```
#file1.to_excel(r'D:\SimulationResultsCollected\Outputs\T'+str(SimulationNumber)+'.xlsx',sheet_name='Sheet1', inc
```

Analyzing results

In [19]:
```
#df=pd.read_excel(r'D:\SimulationResultsCollected\Outputs\T'+str(SimulationNumber)+'.xlsx',sheet_name='Sheet1') #
```

In [20]:
```
df=file1
```

Daily average and running mean outdoor temperature

In [21]:
```
#
dfMean=df.groupby(['Month','Day'], as_index=False)['TAmb'].mean()
dfMean=dfMean.rename(columns={"TAmb": "DailyAverage"})
```

In [22]:
```
#Calculating exponentially decaying running mean outdoor temperature (EDRMOT) for the whole year (ref: Yaqubi)
## dfMean has 365 days [0 to 364]
alpha=0.8
TRM=[]
#df.TAIR_3_1[i]
for i, row in dfMean.iterrows():
    if i==0: # in the first 7 day of the year 7 days of the end of year will calculated as prior days
        TY = (1-alpha)*(dfMean.DailyAverage[364]+
                        alpha*(dfMean.DailyAverage[364-1])+
                        (alpha**2)*dfMean.DailyAverage[364-2]+
                        (alpha**3)*dfMean.DailyAverage[364-3]+
                        (alpha**4)*dfMean.DailyAverage[364-4]+
                        (alpha**5)*dfMean.DailyAverage[364-5]+
                        (alpha**6)*dfMean.DailyAverage[364-6])
    elif i==1:
        TY = (1-alpha)*(dfMean.DailyAverage[i-1]+
                        alpha*(dfMean.DailyAverage[364])+
                        (alpha**2)*dfMean.DailyAverage[364-1]+
                        (alpha**3)*dfMean.DailyAverage[364-2]+
                        (alpha**4)*dfMean.DailyAverage[364-3]+
                        (alpha**5)*dfMean.DailyAverage[364-4]+
                        (alpha**6)*dfMean.DailyAverage[364-5])
    elif i==2:
        TY = (1-alpha)*(dfMean.DailyAverage[i-1]+
                        alpha*(dfMean.DailyAverage[i-2])+
                        (alpha**2)*dfMean.DailyAverage[364]+
                        (alpha**3)*dfMean.DailyAverage[364-1]+
                        (alpha**4)*dfMean.DailyAverage[364-2]+
                        (alpha**5)*dfMean.DailyAverage[364-3]+
                        (alpha**6)*dfMean.DailyAverage[364-4])
    elif i==3:
        TY = (1-alpha)*(dfMean.DailyAverage[i-1]+
                        alpha*(dfMean.DailyAverage[i-2])+
                        (alpha**2)*dfMean.DailyAverage[i-3]+
                        (alpha**3)*dfMean.DailyAverage[364]+
```

```
                        (alpha**4)*dfMean.DailyAverage[364-1]+
                        (alpha**5)*dfMean.DailyAverage[364-2]+
                        (alpha**6)*dfMean.DailyAverage[364-3])
    elif i==4:
        TY = (1-alpha)*(dfMean.DailyAverage[i-1]+
                        alpha*(dfMean.DailyAverage[i-2])+
                        (alpha**2)*dfMean.DailyAverage[i-3]+
                        (alpha**3)*dfMean.DailyAverage[i-4]+
                        (alpha**4)*dfMean.DailyAverage[364]+
                        (alpha**5)*dfMean.DailyAverage[364-1]+
                        (alpha**6)*dfMean.DailyAverage[364-2])
    elif i==5:
        TY = (1-alpha)*(dfMean.DailyAverage[i-1]+
                        alpha*(dfMean.DailyAverage[i-2])+
                        (alpha**2)*dfMean.DailyAverage[i-3]+
                        (alpha**3)*dfMean.DailyAverage[i-4]+
                        (alpha**4)*dfMean.DailyAverage[i-5]+
                        (alpha**5)*dfMean.DailyAverage[364]+
                        (alpha**6)*dfMean.DailyAverage[364-1])
    elif i==6:
        TY = (1-alpha)*(dfMean.DailyAverage[i-1]+
                        alpha*(dfMean.DailyAverage[i-2])+
                        (alpha**2)*dfMean.DailyAverage[i-3]+
                        (alpha**3)*dfMean.DailyAverage[i-4]+
                        (alpha**4)*dfMean.DailyAverage[i-5]+
                        (alpha**5)*dfMean.DailyAverage[i-6]+
                        (alpha**6)*dfMean.DailyAverage[364])
    else:
        TY = (1-alpha)*(dfMean.DailyAverage[i-1]+
                        alpha*(dfMean.DailyAverage[i-2])+
                        (alpha**2)*dfMean.DailyAverage[i-3]+
                        (alpha**3)*dfMean.DailyAverage[i-4]+
                        (alpha**4)*dfMean.DailyAverage[i-5]+
                        (alpha**5)*dfMean.DailyAverage[i-6]+
                        (alpha**6)*dfMean.DailyAverage[i-7])
    TRM.append(TY)
dfMean['EDRMOT']= TRM
```

In [23]:

```
##Optimal Comfort temperature when ERDMOT is in within (10,30) interval
ComT=[]
for i, row in dfMean.iterrows():
    if dfMean.EDRMOT[i]>30: # in the first 7 day of the year 7 days of the end of year will calculated as prior d
        TY = (0.31*30)+18.8
    elif dfMean.EDRMOT[i]<10:
        TY = (0.31*10)+18.8
    else:
        TY = 0.31*dfMean.EDRMOT[i]+18.8
    ComT.append(TY)
dfMean['ComfortTemp']= ComT
```

In [24]:

```
### maximum acceptable category 1, 2,3 of EN 16798-1:2015/2019
dfMean['TMaxCat_I']=dfMean['ComfortTemp']+2
dfMean['TMinCat_I']=dfMean['ComfortTemp']-3
dfMean['TMaxCat_II']=dfMean['ComfortTemp']+3
dfMean['TMinCat_II']=dfMean['ComfortTemp']-4
dfMean['TMaxCat_III']=dfMean['ComfortTemp']+4
dfMean['TMinCat_III']=dfMean['ComfortTemp']-5
```

In [25]:

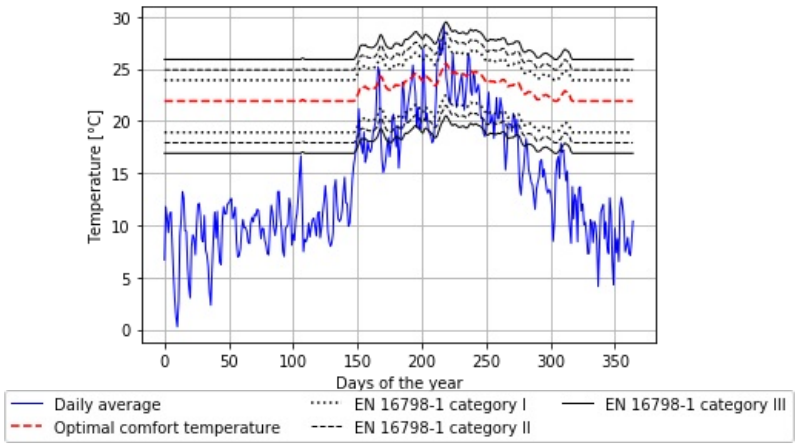
```
##Merging mean and comfort temperature from EN 16978-1:2019 to original database
df=pd.merge(df, dfMean, on=['Month', 'Day'])
```

In [26]:

```
plt.plot(dfMean.index, 'DailyAverage', data=dfMean, color='blue',linewidth=1,linestyle='-',label='Daily average')
plt.plot(dfMean.index, 'ComfortTemp', data=dfMean, color='red',linewidth=1.5,linestyle='--',label='Optimal comfor
plt.plot(dfMean.index, 'TMaxCat_I', data=dfMean, color='black',linewidth=1.5,linestyle=':',label='EN 16798-1 cate
plt.plot(dfMean.index, 'TMinCat_I', data=dfMean, color='black',linewidth=1.5,linestyle=':',label='_nolegend_')
plt.plot(dfMean.index, 'TMaxCat_II', data=dfMean, color='black',linewidth=1,linestyle='--',label='EN 16798-1 cate
plt.plot(dfMean.index, 'TMinCat_II', data=dfMean, color='black',linewidth=1,linestyle='--',label='_nolegend_')
plt.plot(dfMean.index, 'TMaxCat_III', data=dfMean, color='black',linewidth=1,linestyle='-',label='EN 16798-1 cate
plt.plot(dfMean.index, 'TMinCat_III', data=dfMean, color='black',linewidth=1,linestyle='-',label='_nolegend_')
plt.xlabel('Days of the year')
# Set the y axis label of the current axis.
plt.ylabel('Temperature [\N{DEGREE SIGN}C]')#'Global horizontal Irradiance [$W/m^2$]')#)
# Set a title
#plt.title('Interpolation of 3hr weather data to hourly')
#plt.xticks([])
#plt.rcParams['axes.facecolor'] = 'w'
plt.rcParams['axes.edgecolor']= 'black'
plt.rcParams["figure.figsize"] = (10,5)
```

```
plt.grid()
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.12),ncol=3)
plt.show()
```

```
C:\Users\obaidullah.yaqubi\AppData\Roaming\Python\Python38\site-packages\matplotlib\cbook\__init__.py:1402: Futur
eWarning: Support for multi-dimensional indexing (e.g. `obj[:, None]`) is deprecated and will be removed in a fut
ure version.  Convert to a numpy array before indexing instead.
  ndim = x[:, None].ndim
C:\Users\obaidullah.yaqubi\AppData\Roaming\Python\Python38\site-packages\matplotlib\axes\_base.py:276: FutureWarn
ing: Support for multi-dimensional indexing (e.g. `obj[:, None]`) is deprecated and will be removed in a future v
ersion.  Convert to a numpy array before indexing instead.
  x = x[:, np.newaxis]
C:\Users\obaidullah.yaqubi\AppData\Roaming\Python\Python38\site-packages\matplotlib\axes\_base.py:278: FutureWarn
ing: Support for multi-dimensional indexing (e.g. `obj[:, None]`) is deprecated and will be removed in a future v
ersion.  Convert to a numpy array before indexing instead.
  y = y[:, np.newaxis]
```



Energy study

In [27]:
```
df_en=df[['QHEAT_3_1',
          'QHEAT_2_2',
          'QHEAT_2_3',
          'QHEAT_2_4',
          'QHEAT_2_5',
          'QHEAT_1_6',
          'QHEAT_1_7',
          'QHEAT_1_8',
          'QHEAT_1_9',
          'SQHEAT']].groupby(df['Month']).sum().abs()
```

In [28]:
```
### Transpose table to put months as columns
df_en_Transposed = df_en.T
df_en_Transposed.columns=df_en_Transposed.columns.astype(str)
df_en_Transposed=df_en_Transposed.rename(columns={'1':'SQHEAT_Jan',
                                                  '2':'SQHEAT_Feb',
                                                  '3':'SQHEAT_Mar',
                                                  '4':'SQHEAT_Apr',
                                                  '5':'SQHEAT_May',
                                                  '6':'SQHEAT_Jun',
                                                  '7':'SQHEAT_Jul',
                                                  '8':'SQHEAT_Aug',
                                                  '9':'SQHEAT_Sep',
                                                  '10':'SQHEAT_Oct',
                                                  '11':'SQHEAT_Nov',
                                                  '12':'SQHEAT_Dec'})
```

In [29]:
```
df_cn=df[['QCOOL_3_1',
          'QCOOL_2_2',
          'QCOOL_2_3',
          'QCOOL_2_4',
          'QCOOL_2_5',
          'QCOOL_1_6',
          'QCOOL_1_7',
          'QCOOL_1_8',
          'QCOOL_1_9',
          'SQCOOL']].groupby(df['Month']).sum().abs()
```

In [30]:

```
### Transpose table to put months as columns
df_cn_Transposed = df_cn.T
df_cn_Transposed.columns=df_cn_Transposed.columns.astype(str)
df_cn_Transposed=df_cn_Transposed.rename(columns={'1':'SQCOOL_Jan',
                                                  '2':'SQCOOL_Feb',
                                                  '3':'SQCOOL_Mar',
                                                  '4':'SQCOOL_Apr',
                                                  '5':'SQCOOL_May',
                                                  '6':'SQCOOL_Jun',
                                                  '7':'SQCOOL_Jul',
                                                  '8':'SQCOOL_Aug',
                                                  '9':'SQCOOL_Sep',
                                                  '10':'SQCOOL_Oct',
                                                  '11':'SQCOOL_Nov',
                                                  '12':'SQCOOL_Dec'})
```

In [31]:
```
df_energy=pd.concat([df_en_Transposed.reset_index(drop=False),
                     df_cn_Transposed.reset_index(drop=True)], axis=1)
```

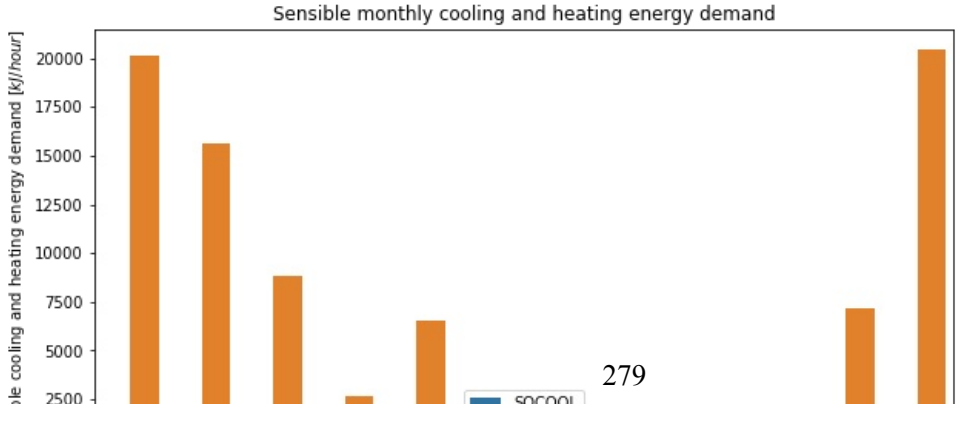
In [32]:
```
df_energy.head(8)
```

Out[32]:

Month	index	SQHEAT_Jan	SQHEAT_Feb	SQHEAT_Mar	SQHEAT_Apr	SQHEAT_May	SQHEAT_Jun	SQHEAT_Jul	SQHEAT_Aug	SQHEA
0	QHEAT_3_1	5.688414e+06	4.183485e+06	2.700528e+06	1.078472e+06	1.942347e+06	0.0	0.0	0.0	
1	QHEAT_2_2	1.227682e+06	7.962169e+05	4.671454e+05	7.431993e+04	4.150390e+05	0.0	0.0	0.0	
2	QHEAT_2_3	1.579100e+06	1.160258e+06	6.656470e+05	1.755179e+05	4.496832e+05	0.0	0.0	0.0	
3	QHEAT_2_4	1.323587e+06	9.569263e+05	5.859630e+05	1.320321e+05	3.384746e+05	0.0	0.0	0.0	
4	QHEAT_2_5	1.439967e+06	8.882666e+05	5.425898e+05	8.363373e+04	5.615692e+05	0.0	0.0	0.0	
5	QHEAT_1_6	7.966642e+05	4.904240e+05	3.089804e+05	4.320200e+04	3.249862e+05	0.0	0.0	0.0	
6	QHEAT_1_7	7.632480e+05	6.043276e+05	3.754274e+05	1.036655e+05	7.722142e+04	0.0	0.0	0.0	
7	QHEAT_1_8	1.035644e+06	7.440669e+05	5.108983e+05	1.658104e+05	3.347633e+05	0.0	0.0	0.0	

In [33]:
```
df_energy2 = df.melt(id_vars = 'Month',
                     value_vars = ['SQCOOL',
#                                   'QSENS_2_2',
#                                   'QSENS_2_3',
#                                   'QSENS_2_4',
#                                   'QSENS_1_5',
#                                   'QSENS_1_6',
#                                   'QSENS_1_7'],
                                  'SQHEAT'],
                     var_name = 'Zones')
```

In [34]:
```
df_energy2['value'] = df_energy2['value'].apply(abs)
sns.barplot(x = 'Month',
            y = 'value',
            hue = 'Zones',
            data = df_energy2,  #
#            estimator = sum,
            ci = 0).set(title='Sensible monthly cooling and heating energy demand',
                                xlabel='Months of the year',
                                ylabel='Sensible cooling and heating energy demand [$kJ/hour$]',label=['Sens'])
#leg = plt.legend(loc='upper center',labels=['Whole Building'])
#for l in leg.legendHandles:
#    l.set_linewidth(8)
plt.legend()
plt.show()
```



Indoor thermal condition calculations

Givoni index

In [35]:
```
#Py_df=pd.DataFrame()
import psychrolib
psychrolib.SetUnitSystem(psychrolib.SI)
from shapely.geometry import Point
from shapely.geometry.polygon import Polygon
```

In [36]:
```
## Altitude in meters to pressure in Pascals
## Reference  https://www.engineeringtoolbox.com/air-altitude-pressure-d_462.html
### Altitude of location in Meters
import math
AltitudeFromSeaLevel = 30
p = (101325)*(1-(2.25577*(10)**(-5))*(AltitudeFromSeaLevel))**5.25588
p
```

Out[36]: 100965.12412724759

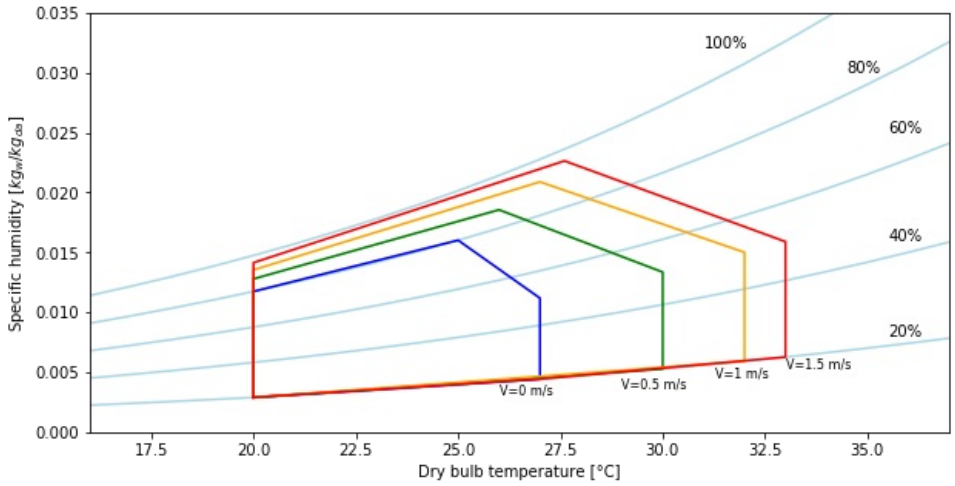
In [37]:
```
##link to documentation of package:  https://psychrometrics.github.io/psychrolib/api_docs.html
psychrolib.GetHumRatioFromRelHum(25, 0.5, p)
```

Out[37]: 0.00991682484151132

In [38]:
```
## Intitial data table to plot Givoni index plot
Table=[]
for RH in [0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]:
    T1 = []
    for i in range(0,60):
        T1a=psychrolib.GetHumRatioFromRelHum(i, RH, p)
        T1.append(T1a)
        df_T=pd.DataFrame(T1,columns=[str(round(100*RH))+' %'])
    Table.append(df_T)
PsyDataFrame=pd.concat(Table,axis=1)
```

In [39]:
```
PsyDataFrame.plot(y=['20 %','40 %','60 %','80 %','100 %'],legend=None,color="lightblue")
plt.text(35.5, 0.008, '20%')
plt.text(35.5, 0.016, '40%')
plt.text(35.5, 0.025, '60%')
plt.text(34.5, 0.030, '80%')
plt.text(31, 0.032, '100%')
##First polygon
poly1=Polygon([(20,psychrolib.GetHumRatioFromRelHum(20, 0.2, p)),
               (20,psychrolib.GetHumRatioFromRelHum(20, 0.8, p)),
               (25,psychrolib.GetHumRatioFromRelHum(25, 0.8, p)),
               (27,psychrolib.GetHumRatioFromRelHum(27, 0.5, p)),
               (27,psychrolib.GetHumRatioFromRelHum(27, 0.2, p))])

x,y = poly1.exterior.xy
plt.plot(x,y,color="blue")
##Second plygon
poly2=Polygon([(20,psychrolib.GetHumRatioFromRelHum(20, 0.2, p)),
               (20,psychrolib.GetHumRatioFromRelHum(20, 0.87, p)),
               (26,psychrolib.GetHumRatioFromRelHum(26, 0.87, p)),
               (30,psychrolib.GetHumRatioFromRelHum(30, 0.5, p)),
               (30,psychrolib.GetHumRatioFromRelHum(30, 0.2, p))])
x1,y1 = poly2.exterior.xy
plt.plot(x1,y1,color="green")
##Third plygon
poly3=Polygon([(20,psychrolib.GetHumRatioFromRelHum(20, 0.2, p)),
               (20,psychrolib.GetHumRatioFromRelHum(20, 0.92, p)),
               (27,psychrolib.GetHumRatioFromRelHum(27, 0.92, p)),
               (32,psychrolib.GetHumRatioFromRelHum(32, 0.5, p)),
               (32,psychrolib.GetHumRatioFromRelHum(32, 0.2, p))])
```

```
x3,y3 = poly3.exterior.xy
plt.plot(x3,y3,color="orange")
##Fourth plygon
poly4=Polygon([(20,psychrolib.GetHumRatioFromRelHum(20, 0.2, p)),
               (20,psychrolib.GetHumRatioFromRelHum(20, 0.96, p)),
               (27.6,psychrolib.GetHumRatioFromRelHum(27.6, 0.96, p)),
               (33,psychrolib.GetHumRatioFromRelHum(33, 0.5, p)),
               (33,psychrolib.GetHumRatioFromRelHum(33, 0.2, p)),
               (27,psychrolib.GetHumRatioFromRelHum(27, 0.2, p))])
x4,y4 = poly4.exterior.xy
plt.plot(x4,y4,color="red")
## Txt on the graph
plt.text(26, psychrolib.GetHumRatioFromRelHum(26, 0.15, p), 'V=0 m/s',fontsize=8)
plt.text(29, psychrolib.GetHumRatioFromRelHum(29, 0.15, p), 'V=0.5 m/s',fontsize=8)
plt.text(31.3, psychrolib.GetHumRatioFromRelHum(31.3, 0.16, p), 'V=1 m/s',fontsize=8)
plt.text(33, psychrolib.GetHumRatioFromRelHum(33, 0.17, p), 'V=1.5 m/s',fontsize=8)

## Data points
##plot scatter chart

plt.rcParams["figure.figsize"] = (10,5)
plt.xlim([16, 37])
plt.ylim([0, 0.035])
plt.ylabel("Specific humidity [$kg_w /kg_{da}$]")
plt.xlabel("Dry bulb temperature [\N{DEGREE SIGN}C]")
plt.show()
```



In [40]:

```
for zone in ['3_1','2_2','2_3','2_4','2_5','1_6','1_7','1_8','1_9']:
    Givoni=[]
    for i, row in df.iterrows():
        TAI = df['TAIR_'+str(zone)].values[i]
        RH = df['RELHUM_'+str(zone)].values[i]

        point=Point(TAI,psychrolib.GetHumRatioFromRelHum(TAI,RH/100,p))
        if TAI<20:
            TY = 'Givoni under 20 degrees'
        elif poly1.contains(point):
            TY = 'Givoni V=0 m/s'
        elif poly2.contains(point):
            TY = 'Givoni V=0.5 m/s'
        elif poly3.contains(point):
            TY = 'Givoni V=1 m/s'
        elif poly4.contains(point):
            TY = 'Givoni V=1.5 m/s'
        else:
            TY = 'Thermally stressful'
        Givoni.append(TY)
    df['Givoni_'+str(zone)]= Givoni
```

In [41]:

```
df.Givoni_3_1.unique()
```

Out[41]:

```
array(['Thermally stressful', 'Givoni under 20 degrees', 'Givoni V=0 m/s',
       'Givoni V=0.5 m/s', 'Givoni V=1 m/s', 'Givoni V=1.5 m/s'],
      dtype=object)
```

PMV calculation

In [42]:
```
for zone in ['3_1','2_2','2_3','2_4','2_5','1_6','1_7','1_8','1_9']:
    PMV=[]
    for i, row in df.iterrows():
        y = df['PMVLW_'+str(zone)].values[i]
        if y <= -2.5:
            TY = 'Cold (-3 to -2.5]'
        elif y <= -1.5:
            TY = 'Cool (-2.5 to -1.5]'
        elif y < -0.5:
            TY = 'Slightly Cool (-1.5 to -0.5)'
        elif y <= 0.5:
            TY = 'Comfortable [-0.5 to 0.5]'
        elif y < 1.5:
            TY = 'Slightly Warm (0.5 to 1.5)'
        elif y <= 2.5:
            TY = 'Warm [1.5 to 2.5)'
        elif y <= 3:
            TY = 'Hot [2.5 to 3)'
        PMV.append(TY)
    df['PMV_'+str(zone)]= PMV
```

In [43]:
```
### PLot adaptive temperature with ambaint, indoor operative temperatures for 5 summer months
```

Heat stress index

In [44]:
```
#Heat stress index when the unit of measurement is SI
def heat_index(tdb, rh):
    hi = -8.784695 + 1.61139411 * tdb + 2.338549 * rh - 0.14611605 * tdb * rh
    hi += -1.2308094 * 10 ** -2 * tdb ** 2 - 1.6424828 * 10 ** -2 * rh ** 2
    hi += 2.211732 * 10 ** -3 * tdb ** 2 * rh + 7.2546 * 10 ** -4 * tdb * rh ** 2
    hi += -3.582 * 10 ** -6 * tdb ** 2 * rh ** 2
    return round(hi, 1)
```

In [45]:
```
# Test
heat_index(39.86,20.3)
```

Out[45]: 39.3

Discomfort index instead of WBGT index

In [46]:
```
# https://urbansis.eu/thom-discomfort-index/
def discomfort_index(tdb,rh):
    DI = 0.5*tdb + 0.5*(tdb * atan(0.151977*(rh + 8.313659)**(1/2))
                        + atan(tdb + rh)-atan(rh-1.676331)
                        + (0.00391838 *(rh)**(3/2))*atan(0.023101*rh)-4.686035)
    return round(DI,1)
```

In [47]:
```
# Test
discomfort_index(40.5, 90)
```

Out[47]: 39.7

In [48]:
```
# Test
print(discomfort_index(40.5, 90))
DIndex= 40.5-0.55*(1-0.01*90)*(40.5-14.5)
print(DIndex)
```

```
39.7
39.07
```

In [49]:
```
##Calculating HI and DI in each zone
for zone in ['3_1','2_2','2_3','2_4','2_5','1_6','1_7','1_8','1_9']:
    HI=[]
    DI=[]
    for i, row in df.iterrows():
        TAI = df['TAIR_'+str(zone)].values[i]
        RH = df['RELHUM_'+str(zone)].values[i]
```

```
        H = heat_index(TAI,RH)
        D = discomfort_index(TAI,RH)
        HI.append(H)
        DI.append(D)
    df['HI_'+str(zone)]= HI
    df['DI_'+str(zone)]= DI
```

Summarizing Summertime comfort measurement indices

In [50]:
```
## Percentage
df1=df[df['Month'].isin([5,6,7,8,9])].reset_index()
```

In [51]:
```
df1.tail(3)
```

Out[51]:

	index	TIME	Month	Weekday	Day	Hour	TAmb	TAIR_3_1	TAIR_2_2	TAIR_2_3	TAIR_2_4	TAIR_2_5	TAIR_1_6	TAIR_1_7	TAIR
3669	6549	6549	9	Monday	30	21:00:00	17.45	24.055789	24.766608	23.637479	24.229031	25.007340	25.111854	23.853510	23.84
3670	6550	6550	9	Monday	30	22:00:00	17.00	23.505057	24.505271	23.382917	24.048649	24.636860	24.861932	23.815129	23.66
3671	6551	6551	9	Monday	30	23:00:00	16.60	23.075977	24.251079	23.141229	23.929865	24.410239	24.622160	23.771040	23.56

In [52]:
```
## largest value
#print(df1['HI_4_1'].loc[df1.index.isin(df1.index[df1['HI_4_1']==max(df1['HI_4_1'])].tolist())])
#print(df1['DI_4_1'].loc[df1.index.isin(df1.index[df1['DI_4_1']==max(df1['DI_4_1'])].tolist())])
#print(df1['TAIR_4_1'].loc[df1.index.isin(df1.index[df1['TAIR_4_1']==max(df1['TAIR_4_1'])].tolist())])
#print(df1['RELHUM_4_1'].loc[df1.index.isin(df1.index[df1['TAIR_4_1']==max(df1['TAIR_4_1'])].tolist())])
#print(df1['RELHUM_4_1'].loc[df1.index.isin(df1.index[df1['RELHUM_4_1']==max(df1['RELHUM_4_1'])].tolist())])
#print(df1['TAIR_4_1'].loc[3535])
# Points above 98th percentile
#df1['HI_4_1'].loc[df1.index.isin(df1.index[df1['HI_4_1']>np.percentile(df1['HI_4_1'],98)].tolist())]
```

Degree hours above adaptive comfort thresholds

In [53]:
```
##TMax category I of EN 16798
adapt=[]
for column in df1[['TOP_3_1',
                   'TOP_2_2','TOP_2_3','TOP_2_4','TOP_2_5','TOP_1_6',
                   'TOP_1_7','TOP_1_8','TOP_1_9']]:
    ComT=[]
    for i, row in df1.iterrows():
        if df1.TMaxCat_I[i]<df1[column][i]:
            TY = df1[column][i]-df1.TMaxCat_I[i]
        else:
            TY = 0
        ComT.append(TY)
    Adap=sum(ComT)
    adapt.append(Adap)
TMaxIDataFrame=pd.DataFrame(adapt,columns=['Degree hours above TMaxCat_I'])
```

In [54]:
```
##TMax category II of EN 16798
adapt=[]
for column in df1[['TOP_3_1',
                   'TOP_2_2','TOP_2_3','TOP_2_4','TOP_2_5','TOP_1_6',
                   'TOP_1_7','TOP_1_8','TOP_1_9']]:
    ComT=[]
    for i, row in df1.iterrows():
        if df1.TMaxCat_II[i]<df1[column][i]:
            TY = df1[column][i]-df1.TMaxCat_II[i]
        else:
            TY = 0
        ComT.append(TY)
    Adap=sum(ComT)
    adapt.append(Adap)
TMaxIIDataFrame=pd.DataFrame(adapt,columns=['Degree hours above TMaxCat_II'])
```

In [55]:
```
##TMax category III of EN 16798
```

```
adapt=[]
for column in df1[['TOP_3_1',
                   'TOP_2_2','TOP_2_3','TOP_2_4','TOP_2_5','TOP_1_6',
                   'TOP_1_7','TOP_1_8','TOP_1_9']]:
    ComT=[]
    for i, row in df1.iterrows():
        if df1.TMaxCat_III[i]<df1[column][i]:
            TY = df1[column][i]-df1.TMaxCat_III[i]
        else:
            TY = 0
        ComT.append(TY)
    Adap=sum(ComT)
    adapt.append(Adap)
TMaxIIIDataFrame=pd.DataFrame(adapt,columns=['Degree hours above TMaxCat_III'])
```

In [56]:
```
## Joining reindexed files of energy, TAir, and PMV
AdaptDataFrame=pd.concat([TMaxIDataFrame.reset_index(drop=True),
                          TMaxIIDataFrame.reset_index(drop=True),
                          TMaxIIIDataFrame.reset_index(drop=True)], axis=1)
```

In [57]:
```
AdaptDataFrame.head(8)
```

Out[57]:

	Degree hours above TMaxCat_I	Degree hours above TMaxCat_II	Degree hours above TMaxCat_III
0	5788.186506	4156.328555	2861.688395
1	4726.193131	2976.962012	1760.476916
2	5371.478303	3577.477095	2228.966241
3	3197.922314	1846.901473	988.661557
4	4437.367510	2805.063562	1671.758399
5	3897.292226	2335.833545	1331.289310
6	3090.179758	1737.102278	866.272351
7	2428.028733	1318.873884	631.162198

Percentage of hours above adaptive comfort thresholds

In [58]:
```
##TMax category I of EN 16798
adapt=[]
for column in df1[['TOP_3_1',
                   'TOP_2_2','TOP_2_3','TOP_2_4','TOP_2_5','TOP_1_6',
                   'TOP_1_7','TOP_1_8','TOP_1_9']]:
    ComT=[]
    for i, row in df1.iterrows():
        if df1.TMaxCat_I[i]<df1[column][i]:
            TY = 1
        else:
            TY = 0
        ComT.append(TY)
    Adap=round(100*sum(ComT)/len(ComT),1)
    adapt.append(Adap)
PMaxIDataFrame=pd.DataFrame(adapt,columns=['Percentage of hours above TMaxCat_I'])
```

In [59]:
```
##TMax category II of EN 16798
adapt=[]
for column in df1[['TOP_3_1',
                   'TOP_2_2','TOP_2_3','TOP_2_4','TOP_2_5','TOP_1_6',
                   'TOP_1_7','TOP_1_8','TOP_1_9']]:
    ComT=[]
    for i, row in df1.iterrows():
        if df1.TMaxCat_II[i]<df1[column][i]:
            TY = 1
        else:
            TY = 0
        ComT.append(TY)
    Adap=round(100*sum(ComT)/len(ComT),1)
    adapt.append(Adap)
PMaxIIDataFrame=pd.DataFrame(adapt,columns=['Percentage of hours above TMaxCat_II'])
```

In [60]:
```
##TMax category III of EN 16798
adapt=[]
for column in df1[['TOP_3_1',
                   'TOP_2_2','TOP_2_3','TOP_2_4','TOP_2_5','TOP_1_6',
```

```
                    'TOP_1_7','TOP_1_8','TOP_1_9']]:
    ComT=[]
    for i, row in df1.iterrows():
        if df1.TMaxCat_III[i]<df1[column][i]:
            TY = 1
        else:
            TY = 0
        ComT.append(TY)
    Adap=round(100*sum(ComT)/len(ComT),1)
    adapt.append(Adap)
PMaxIIIDataFrame=pd.DataFrame(adapt,columns=['Percentage of hours above TMaxCat_III'])
```

In [61]:

```
## Joining reindexed files of energy, TAir, and PMV
PercentageDataFrame=pd.concat([PMaxIDataFrame.reset_index(drop=True),
                               PMaxIIDataFrame.reset_index(drop=True),
                               PMaxIIIDataFrame.reset_index(drop=True)], axis=1)
```

In [62]:

```
PercentageDataFrame
```

Out[62]:

	Percentage of hours above TMaxCat_I	Percentage of hours above TMaxCat_II	Percentage of hours above TMaxCat_III
0	49.0	39.6	31.0
1	54.7	40.7	25.9
2	55.0	42.9	31.0
3	44.6	29.2	17.8
4	51.5	37.4	24.7
5	50.7	34.4	21.1
6	46.3	29.1	19.7
7	37.4	23.7	14.3
8	48.8	34.1	21.7

Givoni index in summer months percentage in each polygon

In [63]:

```
givoni=[]
for column in df1[['Givoni_3_1','Givoni_2_2','Givoni_2_3','Givoni_2_4',
                   'Givoni_2_5','Givoni_1_6','Givoni_1_7','Givoni_1_8','Givoni_1_9']]:
    TY=df1[column].value_counts(normalize=True).mul(100).round(1).to_dict()
    s=pd.DataFrame.from_dict(TY,orient='index',columns=[column])
    givoni.append(s)
GoDataFrame= pd.concat(givoni,sort=True,axis=1)
GoDataFrame=GoDataFrame.T
GoDataFrame.head(3)
```

Out[63]:

	Givoni V=0 m/s	Givoni V=0.5 m/s	Givoni V=1 m/s	Givoni V=1.5 m/s	Givoni under 20 degrees	Thermally stressful
Givoni_3_1	47.5	22.8	8.1	2.0	15.4	4.1
Givoni_2_2	48.3	23.7	7.0	1.5	17.5	1.9
Givoni_2_3	46.4	26.1	7.7	1.8	15.7	2.3

In [64]:

```
column_title = ['Givoni under 20 degrees','Givoni V=0 m/s','Givoni V=0.5 m/s',
                'Givoni V=1 m/s','Givoni V=1.5 m/s','Thermally stressful']
GoDataFrame=GoDataFrame.reindex(columns=column_title)
```

Number of hours Indoor operative Temperature above 26,27,28,30 in summer months

In [65]:

```
# Months dictionary
monthDict = {1:'Jan',2:'Feb',3:'Mar',4:'Apr',5:'May',6:'Jun',7:'Jul',8:'Aug',9:'Sep',10:'Oct',11:'Nov',12:'Dec'}
Thresholds= [26,27,28,32]
```

In [66]:

```
## number of hours Indoor operative temperature is above 26 degree
OT=[]
for column in df1[['TOP_3_1',
                   'TOP_2_2','TOP_2_3','TOP_2_4','TOP_2_5','TOP_1_6',
                   'TOP_1_7','TOP_1_8','TOP_1_9']]:
    ComT=[]
```

```
    for i, row in df1.iterrows():
        if df1[column][i]>26:
            TY = 1
        else:
            TY = 0
        ComT.append(TY)
    Adap=sum(ComT)
    OT.append(Adap)
T26DataFrame=pd.DataFrame(OT,columns=['Number of hours above 26'])
```

In [67]:

```
## number of hours Indoor operative temperature is above 27 degree
OT=[]
for column in df1[['TOP_3_1',
                   'TOP_2_2','TOP_2_3','TOP_2_4','TOP_2_5','TOP_1_6',
                   'TOP_1_7','TOP_1_8','TOP_1_9']]:
    ComT=[]
    for i, row in df1.iterrows():
        if df1[column][i]>27:
            TY = 1
        else:
            TY = 0
        ComT.append(TY)
    Adap=sum(ComT)
    OT.append(Adap)
T27DataFrame=pd.DataFrame(OT,columns=['Number of hours above 27'])
```

In [68]:

```
## number of hours Indoor operative temperature is above 28 degree
OT=[]
for column in df1[['TOP_3_1',
                   'TOP_2_2','TOP_2_3','TOP_2_4','TOP_2_5','TOP_1_6',
                   'TOP_1_7','TOP_1_8','TOP_1_9']]:
    ComT=[]
    for i, row in df1.iterrows():
        if df1[column][i]>28:
            TY = 1
        else:
            TY = 0
        ComT.append(TY)
    Adap=sum(ComT)
    OT.append(Adap)
T28DataFrame=pd.DataFrame(OT,columns=['Number of hours above 28'])
```

In [69]:

```
## number of hours Indoor operative temperature is above 30 degree
OT=[]
for column in df1[['TOP_3_1',
                   'TOP_2_2','TOP_2_3','TOP_2_4','TOP_2_5','TOP_1_6',
                   'TOP_1_7','TOP_1_8','TOP_1_9']]:
    ComT=[]
    for i, row in df1.iterrows():
        if df1[column][i]>30:
            TY = 1
        else:
            TY = 0
        ComT.append(TY)
    Adap=sum(ComT)
    OT.append(Adap)
T30DataFrame=pd.DataFrame(OT,columns=['Number of hours above 30'])
```

In [70]:

```
## Joining reindexed files of energy, TAir, and PMV
TempDataFrame=pd.concat([T26DataFrame.reset_index(drop=True),
                         T27DataFrame.reset_index(drop=True),
                         T28DataFrame.reset_index(drop=True),
                         T30DataFrame.reset_index(drop=True)], axis=1)
```

In [71]:

```
#TempDataFrame
```

Maximum Consecutive number of hours temperature is above 27

In [72]:

```
from itertools import groupby, count
```

In [73]:

```
## number of hours Indoor operative temperature is consecutively above 27 degree
OT=[]
for column in df1[['TOP_3_1',
                   'TOP_2_2','TOP_2_3','TOP_2_4','TOP_2_5','TOP_1_6',
```

```
                    'TOP_1_7','TOP_1_8','TOP_1_9']]:
    ComT=[]
    for i, row in df1.iterrows():
        if df1[column][i]>27:
            TY = 1
        else:
            TY = 0
        ComT.append(TY)
    Adap=max([(k, sum(1 for i in g)) for k,g in groupby(ComT)])[1]
#    Adap=[Adap if (Adap<743) else None]
    OT.append(Adap)
ConsecDataFrame=pd.DataFrame(OT,columns=['Maximum consective hours above 27'])
```

In [74]:
```
#[(k, sum(1 for i in g)) for k,g in groupby(ComT)]
```

In [75]:
```
## Peak temperature of the zone
OT=[]
for column in df1[['TOP_3_1',
                   'TOP_2_2','TOP_2_3','TOP_2_4','TOP_2_5','TOP_1_6',
                   'TOP_1_7','TOP_1_8','TOP_1_9']]:
    ComT=[]
    for i, row in df1.iterrows():
        if df1[column][i]>27:
            TY = df1[column][i]
        else:
            TY = 0
        ComT.append(TY)
    Adap=round(max([(k, sum(1 for i in g)) for k,g in groupby(ComT)])[0])
    OT.append(Adap)
ConsecDataFrame['Peak operative temperature']=OT
```

In [76]:
```
#ConsecDataFrame
```

PMV percentage in summer months

In [77]:
```
PMV=[]
for column in df1[['PMV_3_1','PMV_2_2','PMV_2_3','PMV_2_4','PMV_2_5',
                   'PMV_1_6','PMV_1_7','PMV_1_8','PMV_1_9']]:
    TY=df1[column].value_counts(normalize=True).mul(100).round(1).to_dict()
    s=pd.DataFrame.from_dict(TY,orient='index',columns=[column])
    PMV.append(s)
PMVDataFrame= pd.concat(PMV,sort=True,axis=1)
PMVDataFrame=PMVDataFrame.T
#PMVDataFrame
```

In [78]:
```
column_titless = ['Cold (-3 to -2.5]','Cool (-2.5 to -1.5]','Slightly Cool (-1.5 to -0.5)','Comfortable [-0.5 to
                  'Slightly Warm (0.5 to 1.5)','Warm [1.5 to 2.5)','Hot [2.5 to 3)']
PMVDataFrame=PMVDataFrame.reindex(columns=column_titless)
```

Degree hours above RT2020 thresholds without consideration for occupied hours

In [79]:
```
# Night-time hours
NightHours =['00:00:00', '01:00:00', '02:00:00', '03:00:00', '04:00:00','05:00:00', '06:00:00','23:00:00']
OccupiedDayHours =['07:00:00', '08:00:00', '09:00:00', '18:00:00', '19:00:00','20:00:00', '21:00:00', '22:00:00']
```

In [80]:
```
##TMax category II of EN 16798
adapt=[]
for column in df1[['TOP_3_1',
                   'TOP_2_2','TOP_2_3','TOP_2_4','TOP_2_5','TOP_1_6',
                   'TOP_1_7','TOP_1_8','TOP_1_9']]:
    ComT=[]
    for i, row in df1.iterrows():
        T=min(max(df1.TMaxCat_II[i],26),28)
        hour=df1.Hour[i]
        temp=df1[column][i]

        if hour in NightHours and temp>26:
            TY=temp-26
        elif hour not in NightHours and temp>T:
            TY=temp-T
        else:
            TY=0
```

```
        ComT.append(TY)
    Adap=sum(ComT)
    adapt.append(Adap)
TMaxIIDataFrame=pd.DataFrame(adapt,columns=['RT2020 without unoccupancy'])
```

Degree hours above RT2020 thresholds with consideration for occupied hours

In [81]:
```
##TMax category II of EN 16798
adapt=[]
for column in df1[['TOP_3_1',
                   'TOP_2_2','TOP_2_3','TOP_2_4','TOP_2_5','TOP_1_6',
                   'TOP_1_7','TOP_1_8','TOP_1_9']]:
    ComT=[]
    for i, row in df1.iterrows():
        T=min(max(df1.TMaxCat_II[i],26),28)
        hour=df1.Hour[i]
        temp=df1[column][i]

        if hour in NightHours and temp>26:
            TY=temp-26
        elif hour in OccupiedDayHours and temp>T:
            TY=temp-T
        else:
            TY=0
        ComT.append(TY)
    Adap=sum(ComT)
    adapt.append(Adap)
TMaxIIDataFrameV=pd.DataFrame(adapt,columns=['RT2020 with unoccupancy'])
```

In [82]:
```
## Joining reindexed files of energy, TAir, and PMV
RT2020DataFrame=pd.concat([TMaxIIDataFrame.reset_index(drop=True),
                           TMaxIIDataFrameV.reset_index(drop=True)], axis=1)
```

Percentage of hours in each Heat Index threshold during summertime

In [ ]:
```
HI = pd.DataFrame({'No Hazard': [],
                   'Caution': [],
                   'Extreme Caution': [],
                   'Danger': [],
                   'Extreme Danger': []})
for column in df1[['HI_3_1','HI_2_2','HI_2_3','HI_2_4','HI_2_5',
                   'HI_1_6','HI_1_7','HI_1_8','HI_1_9']]:
    DT=[]
    for i, row in df1.iterrows():
        if df1[column][i]<26.9:
            T = 'No Hazard'
        elif df1[column][i]<31.9:
            T = 'Caution'
        elif df1[column][i]<40.9:
            T = 'Extreme Caution'
        elif df1[column][i]<50.9:
            T = 'Danger'
        else:
            T = 'Extreme Danger'
        DT.append(T)
    TY=pd.Series(DT).value_counts(normalize=True).mul(100).round(2).to_dict()
    S=pd.DataFrame.from_dict(TY,orient='index',columns=[column]).T
    HI=HI.append(S,ignore_index=False,sort=True)
```

In [84]:
```
column_titles = ['No Hazard','Caution','Extreme Caution','Danger','Extreme Danger']
HI=HI.reindex(columns=column_titles)
```

In [85]:
```
#HI
```

Percentage of hours in each Discomfort Index threshold

In [2]:
```
# https://www.researchgate.net/publication/260262908_Thermal_remote_sensing_of_Thom%27s_Discomfort_Index_DI_Compa

DI = pd.DataFrame({'No discomfort': [],
                   'Less than 50% feel discomfort': [],
                   'More than 50% feel discomfort': [],
                   'Most population suffer discomfort': [],
```

```
                   'Everyone suffer sever heat stress': [],
                   'Medical support required': []})

for column in df1[['DI_3_1','DI_2_2','DI_2_3','DI_2_4','DI_2_5',
                   'DI_1_6','DI_1_7','DI_1_8','DI_1_9']]:
    DT=[]
    for i, row in df1.iterrows():
        if df1[column][i]<21:
            T = 'No discomfort'
        elif df1[column][i]<24:
            T = 'Less than 50% feel discomfort'
        elif df1[column][i]<27:
            T = 'More than 50% feel discomfort'
        elif df1[column][i]<29:
            T = 'Most population suffer discomfort'
        elif df1[column][i]<32:
            T = 'Everyone suffer sever heat stress'
        else:
            T = 'Medical support required'
        DT.append(T)
    TY=pd.Series(DT).value_counts(normalize=True).mul(100).round(2).to_dict()
    S=pd.DataFrame.from_dict(TY,orient='index',columns=[column]).T
    DI=DI.append(S,ignore_index=False,sort=True)
```

In [87]:

```
column_tit = ['No discomfort','Less than 50% feel discomfort',
              'More than 50% feel discomfort','Most population suffer discomfort',
              'Everyone suffer sever heat stress','Medical support required']
DI=DI.reindex(columns=column_tit)
```

Joining Energy,temperature,Givoni, PMV

In [88]:

```
## Joining reindexed files of energy, TAir, and PMV
file2=pd.concat([df_energy.reset_index(drop=False),
                 AdaptDataFrame.reset_index(drop=True),
                 PercentageDataFrame.reset_index(drop=True),
                 TempDataFrame.reset_index(drop=True),
                 ConsecDataFrame.reset_index(drop=True),
                 RT2020DataFrame.reset_index(drop=True),
                 HI.reset_index(drop=True),
                 DI.reset_index(drop=True),
                 PMVDataFrame.reset_index(drop=True),
                 GoDataFrame.reset_index(drop=True)], axis=1)
```

In [89]:

```
file2
```

Out[89]:

	level_0	index	SQHEAT_Jan	SQHEAT_Feb	SQHEAT_Mar	SQHEAT_Apr	SQHEAT_May	SQHEAT_Jun	SQHEAT_Jul	SQHEAT_Aug	SQ
0	0	QHEAT_3_1	5.688414e+06	4.183485e+06	2.700528e+06	1.078472e+06	1.942347e+06	0.0	0.0	0.0	
1	1	QHEAT_2_2	1.227682e+06	7.962169e+05	4.671454e+05	7.431993e+04	4.150390e+05	0.0	0.0	0.0	
2	2	QHEAT_2_3	1.579100e+06	1.160258e+06	6.656470e+05	1.755179e+05	4.496832e+05	0.0	0.0	0.0	
3	3	QHEAT_2_4	1.323587e+06	9.569263e+05	5.859630e+05	1.320321e+05	3.384746e+05	0.0	0.0	0.0	
4	4	QHEAT_2_5	1.439967e+06	8.882666e+05	5.425898e+05	8.363373e+04	5.615692e+05	0.0	0.0	0.0	
5	5	QHEAT_1_6	7.966642e+05	4.904240e+05	3.089804e+05	4.320200e+04	3.249862e+05	0.0	0.0	0.0	
6	6	QHEAT_1_7	7.632480e+05	6.043276e+05	3.754274e+05	1.036655e+05	7.722142e+04	0.0	0.0	0.0	
7	7	QHEAT_1_8	1.035644e+06	7.440669e+05	5.108983e+05	1.658104e+05	3.347633e+05	0.0	0.0	0.0	
8	8	QHEAT_1_9	1.128261e+06	6.680329e+05	4.070887e+05	5.700407e+04	4.143142e+05	0.0	0.0	0.0	
9	9	SQHEAT	1.498257e+07	1.049200e+07	6.564268e+06	1.913658e+06	4.858398e+06	0.0	0.0	0.0	

In [90]:

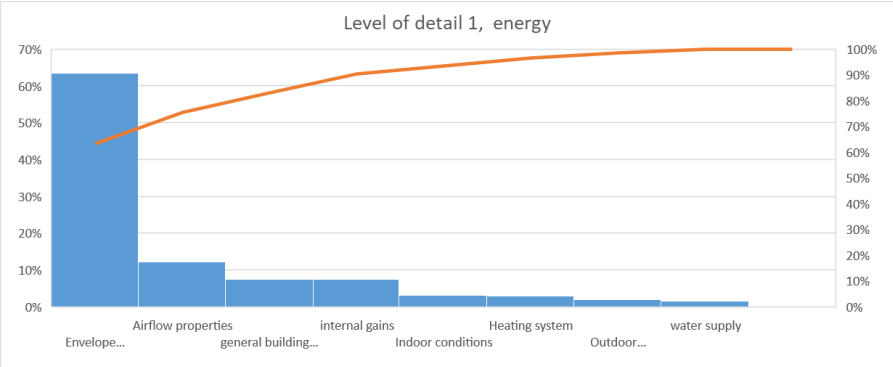
```
df.to_excel(r'D:\SimulationResultsCollected\Outputs\T'+str(SimulationNumber)+'.xlsx',sheet_name='Sheet1', index =
```

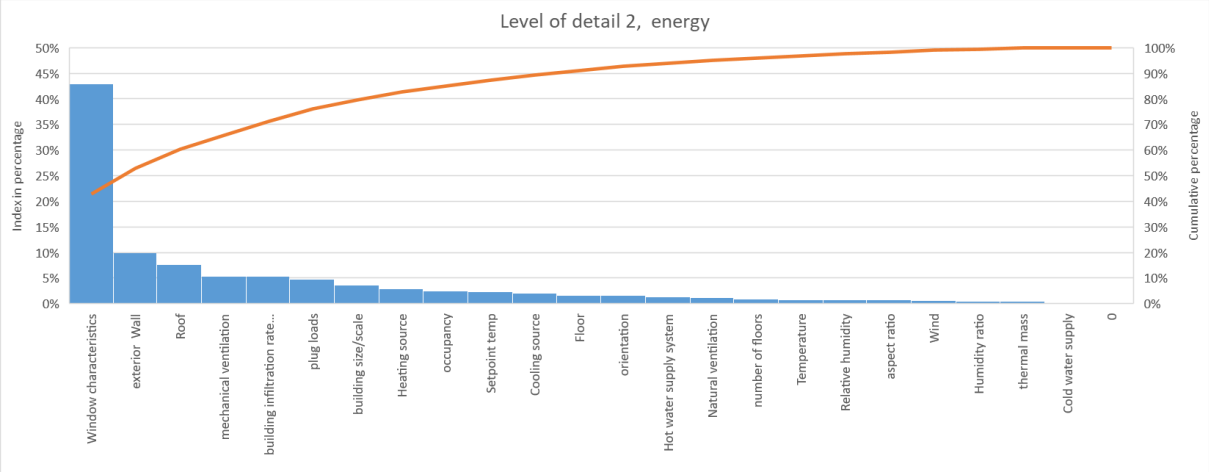
In [91]:

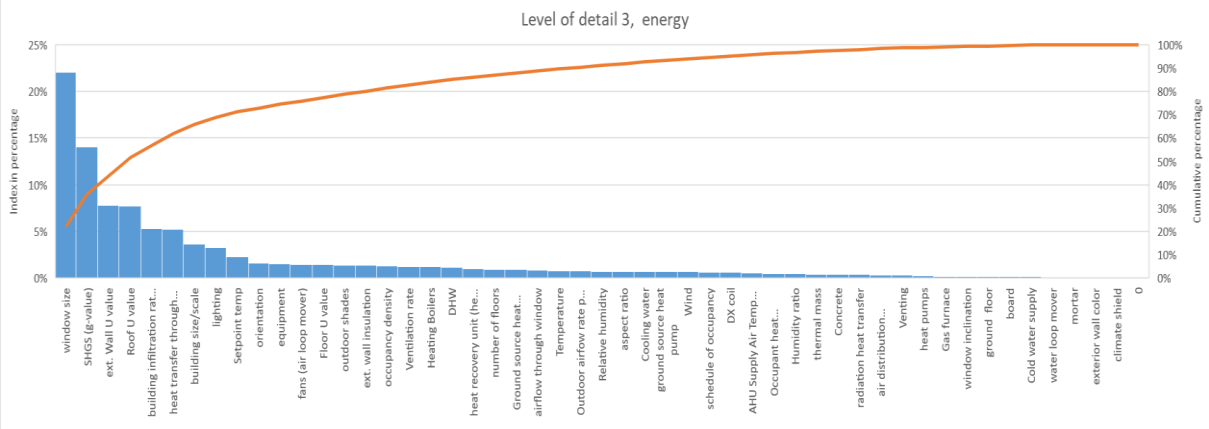
```
with pd.ExcelWriter(r'D:\SimulationResultsCollected\Outputs\T'+str(SimulationNumber)+'.xlsx',engine="openpyxl", m
    file2.to_excel(writer, sheet_name='summerized', index=False)
```

Appendix 4-3:

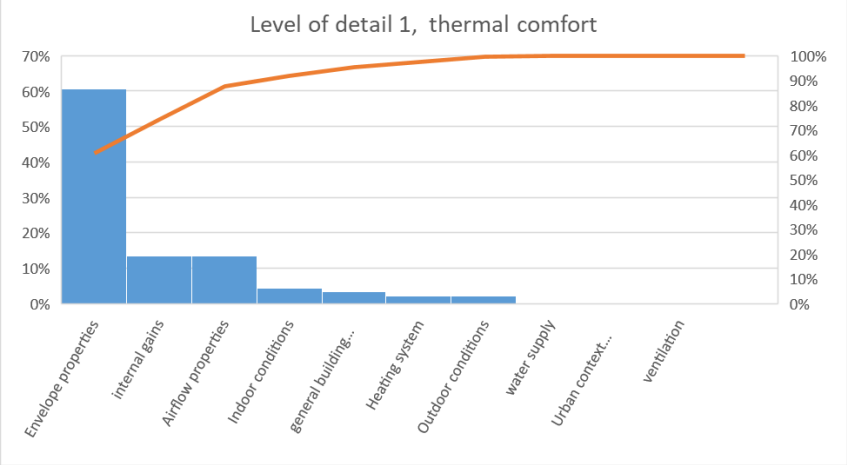
Energy (Cooling, heating, electricity) parameters: at various levels of details

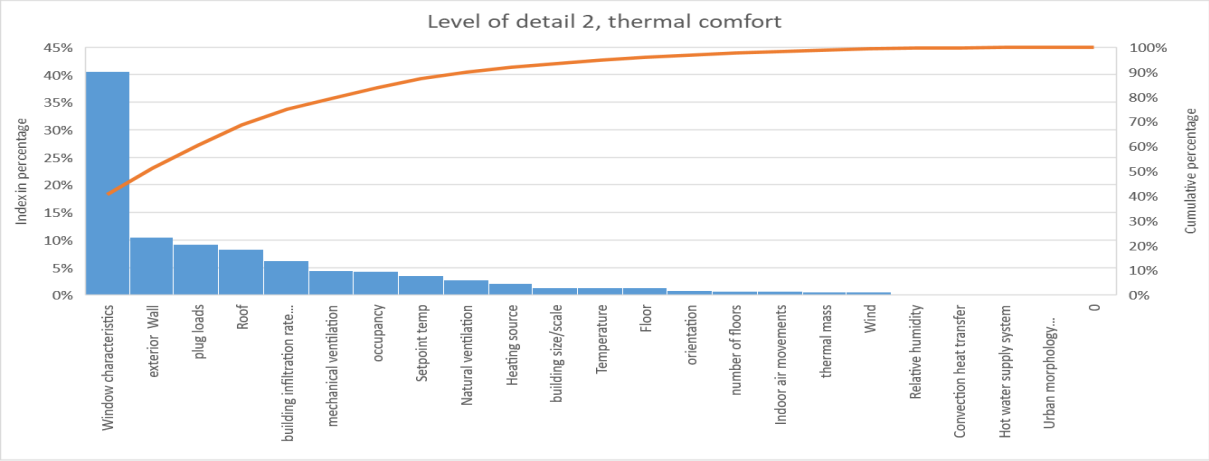


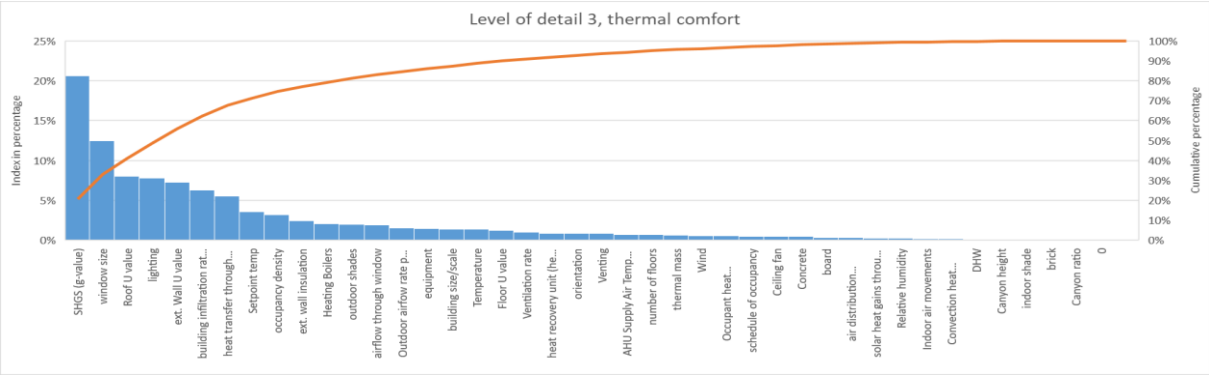




Comfort and summer overheating parameters: at various levels of details







Appendix 5-1:

Table of experimental design variations for KM7_5

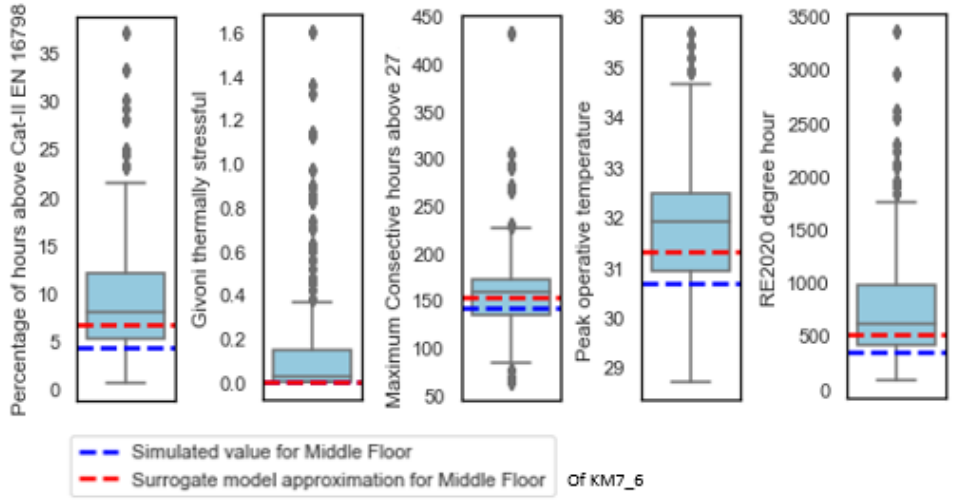
Building type: KM7_5	Surface Area of building [m2]	U-value of exterior wall	U-value of exterior roof	U-value of intermediate wall	Type of window	Adaptive capacity of user = External shading status & Window status (air inflow= infiltration + naural ventilation)	Window to wall ratio (window to area ratio)	Principle orientation [degrees]
Type	**Con.**	**Con.**	**Con.**	**Con.**	**Cat.**	**Cat.**	**Cat.**	**Cat.**
Level	**3**	**3**	**3**	**3**	**2**	**3**	**2**	**2**
Min [-1]	70 [A-1]	0.24	0.2	0.44	[1] Simple	[1] Not adaptive	[W1] Small, R<20%	NS [O1]
mean [0]	95 [A0]	1.25	0.792	1.158	[2] Double	[2] Interm. user	[W2] Large, R>20%	EW [O2]
Max [1]	120 [A1]	3	2.42	2.081		[3] Highly adaptive		
A0_W1_O2	0	-1	0	1	2	3	1	2
A0_W1_O2	0	1	0	-1	1	3	1	2
A0_W2_O1	0	1	0	1	1	3	2	1
A0_W2_O1	0	-1	0	-1	2	3	2	1
A-1_W2_O2	-1	-1	0	0	1	3	2	2
A1_W1_O1	1	-1	0	0	1	3	1	1
A1_W2_O2	1	1	0	0	2	3	2	2
A0_W1_O2	0	-1	-1	0	1	1	1	2
A0_W2_O2	0	-1	1	0	2	2	2	2
A0_W1_O1	0	1	1	0	1	2	1	1
A0_W2_O1	0	1	-1	0	2	1	2	1
A0_W1_O1	0	-1	1	0	2	1	1	1
A0_W1_O2	0	1	-1	0	2	2	1	2
A0_W2_O1	0	-1	-1	0	1	2	2	1
A0_W2_O2	0	1	1	0	1	1	2	2
A1_W1_O1	1	0	1	0	2	3	1	1
A-1_W2_O2	-1	0	1	0	2	3	2	2
A0_W1_O2	0	0	-1	-1	2	3	1	2
A0_W2_O1	0	0	-1	-1	1	3	2	1
A1_W2_O2	1	0	1	0	1	3	2	2
A-1_W1_O1	-1	0	1	0	1	3	1	1
A0_W1_O2	0	0	-1	1	1	3	1	2
A1_W1_O2	1	0	0	-1	2	1	1	2
A-1_W1_O1	-1	0	0	1	1	1	1	1
A1_W2_O1	1	0	0	-1	1	1	2	1
A-1_W2_O2	-1	0	0	1	2	1	2	2
A-1_W1_O2	-1	0	1	0	1	2	1	2
A0_W2_O2	0	0	-1	-1	1	2	2	2
A1_W1_O2	1	0	0	1	1	2	1	2
A-1_W2_O2	-1	0	0	-1	2	1	2	2
A-1_W1_O1	-1	0	0	-1	1	1	1	1
A0_W1_O2	0	0	1	-1	2	2	1	2
A0_W1_O1	0	-1	-1	0	2	1	1	1
A0_W2_O2	0	1	-1	0	1	1	2	2
A1_W2_O1	1	1	0	0	1	2	2	1
A1_W1_O2	1	-1	0	0	2	2	1	2
A1_W1_O2	1	0	0	1	2	1	1	2
A0_W2_O2	0	-1	0	1	1	1	2	2
A0_W1_O1	0	1	0	-1	2	1	1	1
A0_W2_O2	0	1	0	1	2	2	2	2
A0_W1_O1	0	-1	0	1	1	2	1	1
A0_W1_O2	0	-1	1	0	1	1	1	2
A-1_W2_O1	-1	1	0	0	1	2	2	1

A0_W2_O1	0	1	1	0	2	1	2	1
A-1_W2_O1	-1	-1	0	0	1	1	2	1
A1_W1_O1	1	1	0	0	1	1	1	1
A1_W2_O2	1	0	-1	0	1	3	2	2
A-1_W1_O2	-1	1	0	0	2	1	1	2
A-1_W1_O2	-1	-1	0	0	2	2	1	2
A0_W1_O2	0	-1	0	-1	1	3	1	2
A1_W2_O1	1	0	0	1	1	1	2	1
A0_W1_O2	0	0	1	1	1	3	1	2
A0_W2_O1	0	-1	0	1	2	3	2	1
A0_W2_O2	0	1	0	-1	2	2	2	2
A1_W2_O2	1	-1	0	0	2	1	2	2
A1_W1_O1	1	0	-1	0	2	3	1	1
A-1_W2_O2	-1	1	0	0	1	3	2	2
A-1_W1_O1	-1	0	-1	0	2	3	1	1
A0_W1_O1	0	0	1	1	2	2	1	1
A0_W2_O2	0	0	0	0	1	2	2	2
A0_W1_O1	0	-1	0	-1	1	2	1	1
A1_W2_O1	1	0	0	-1	2	2	2	1
A0_W2_O1	0	0	1	-1	1	3	2	1
A0_W2_O1	0	-1	1	0	1	2	2	1
A0_W1_O1	0	1	0	1	2	3	1	1
A0_W1_O1	0	1	-1	0	1	2	1	1
A-1_W1_O2	-1	0	-1	0	1	2	1	2
A0_W2_O1	0	0	-1	1	2	2	2	1
A-1_W2_O2	-1	0	-1	0	2	3	2	2
A0_W1_O1	0	0	0	0	2	2	1	1
A1_W1_O2	1	0	0	-1	1	2	1	2
A-1_W2_O1	-1	0	0	-1	2	2	2	1

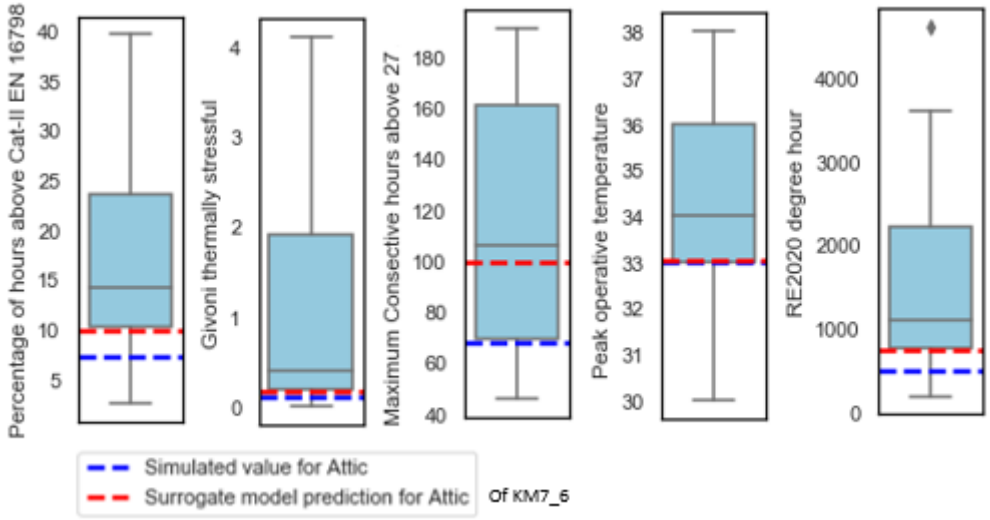
In this table: con. Is short for continuous and cat. Is for categorical data type.

Appendix 5-2:

Validation data for middle floor of KM7_6:



Validation data for attic of KM7_6:



Appendix 5-3:

- Pre-processing data to train for Gradient Boosting Regression and Multinomial Logistic Regression.
- Application of trained Gradient Boosting Regressor and Multinomial Logistic Regression to approximate indoor performance of buildings at city-scale.

In [1]:
```
import numpy as np
np.version.version
```

Out[1]:
```
'1.22.3'
```

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set()
import plotly.express as px
import math
%config Completer.use_jedi = False
import xlrd
from scipy import stats
from datetime import date, datetime, timedelta
```

In [3]:
```
#file1=pd.read_csv(r'D:\3D_models\KM7_3\T_PMV.xls',delimiter='\s+',skiprows=[1])
```

Importing data and filtering

In [4]:
```
pd.set_option("display.max_columns", None)
```

In [5]:
```
file = pd.read_excel(r'D:\SimulationResultsCollected\DatabaseOfResults.xlsx')
```

In [6]:
```
## Reformat some of data in the file
file['Building'] = file['Building'].str.strip()
file['Weatherfile'] = file['Weatherfile'].str.strip()
file['Main OrientationZone'] = file['Main OrientationZone'].str.strip()
file['House']=file['House'].str.strip()
file['UHI']=file['UHI'].str.strip()
file['Floor']=file['Floor'].str.strip()
file['Ratio of window area to zone area ']=round(file['Ratio of window area to zone area '],3)
file['Window_Area_floor']=round(file['Area of windows in the floor in which the zone is located'],3)
```

In [7]:
```
#df1=df[df['Building'].str.contains('KM7_3')]
df=file[file['Building'].str.contains('KM7_5')]
df=df[df['Floor'].notnull()]
#df1=df[df['Floor']!='Attic']
```

In [8]:
```
df=df.rename(columns={'Main OrientationZone': "Orientation",
                      'Area of floor in which the zone is located': "Area",
                      'Area of windows in the floor in which the zone is located':"Area of windows",
                      'Number of windows in the floor in which the zone is located':"Number of windows",
                      'WWR (openning Area/all v.area) of the floor in which the zone is located':"Vertical WWR",
                      'Adaptive capacity of occupant = External shading control & Window status (air inflow + inf
                      'U_ExterWall [W/m²K]\n  \n -1 = 0.242 \n  0 = 1.25 \n  1 = 3 ':"U-value of exterior wall",
                      'U_ExterRoof [W/m²K]\n \n-1 = 0.2 \n 0 = 0.792\n 1 = 2.42':"U-value of exterior roof",
                      'U_AdjacWall\n  \n -1 = 0.44  \n  0 = 1.158 \n  1 = 2.081':"U-value of adjacent wall",
                      'U_Window (window type: simple or double glazing window) ':"Type of window",
                      'Percentage of hours above Cat ii of EN 16798 per storey average':"Percentage of hours abov
                      'Percentage of hours in thermally stressful region of Givoni index per storey average':"Giv
                      'Maximum Consective hours above 27 per storey average':"Maximum Consective hours above 27 A
                      'Peak operative temperature per storey average':"Peak operative temperature Avg",
                      'RE2020 without unoccupancy per storey average':"RE2020 degree hour",
                      'Performance as according to  RE2020 per storey average':"RE2020 situation"})
```

Regression

The original D-optimal was built for a quadratic model with interactions. to run it in python, it is necessary to calculate interactions of values of variables as well

In [9]:
```
from statsmodels.formula.api import ols
from scipy import stats
import statsmodels.api as sm
```

```
C:\Users\obaidullah.yaqubi\Anaconda3\envs\geo_env\lib\site-packages\statsmodels\tsa\base\tsa_model.py:7: FutureWa
```

```
rning: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with
the appropriate dtype instead.
  from pandas import (to_datetime, Int64Index, DatetimeIndex, Period,
C:\Users\obaidullah.yaqubi\Anaconda3\envs\geo_env\lib\site-packages\statsmodels\tsa\base\tsa_model.py:7: FutureWa
rning: pandas.Float64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index wi
th the appropriate dtype instead.
  from pandas import (to_datetime, Int64Index, DatetimeIndex, Period,
```

In [10]:
```
# Coding/transforming categorical variables to dummy variables
# In this transformation, special attention should be given to the type of categorical variable (ordinal or nomi
# Our categorical input variables are nominal categorical variables and their order does not have any significanc
```

In [11]:
```
df1_dummy=pd.get_dummies(df["Type of occupant"])
names=df1_dummy.columns.tolist()
df1_dummy=df1_dummy.rename(columns={names[0]:"Occupant type 1",names[1]:"Occupant type 2",names[2]:"Occupant type
```

In [12]:
```
df1_dummy.head(2)
```

Out[12]:

	Occupant type 1	Occupant type 2	Occupant type 3
38	0	0	1
39	0	0	1

In [13]:
```
df2_dummy=pd.get_dummies(df["Type of window"])
names2=df2_dummy.columns.tolist()
df2_dummy=df2_dummy.rename(columns={names[0]:"Single glazing window",names[1]:"Double glazing window"})
```

In [14]:
```
df2_dummy.head(2)
```

Out[14]:

	Single glazing window	Double glazing window
38	0	1
39	0	1

In [15]:
```
df3_dummy=pd.get_dummies(df["Orientation"])
names3=df3_dummy.columns.tolist()
```

In [16]:
```
df3_dummy.head(2)
```

Out[16]:

	East	East-west	North	North south	South	West
38	0	1	0	0	0	0
39	0	0	0	0	0	1

In [17]:
```
df4_dummy=pd.get_dummies(df["Floor"])
names4=df4_dummy.columns.tolist()
```

In [18]:
```
df4_dummy.head(2)
```

Out[18]:

	Attic	Ground floor	Middle floor
38	1	0	0
39	0	0	1

In [19]:
```
df1=pd.concat([df,df1_dummy,df2_dummy,df3_dummy,df4_dummy],axis=1)
```

In [20]:
```
#df1.columns.tolist()
## Regression analysis on categorical variable in Python
```

In [21]:
```
### saving validation data and input data for meta-model in separate dataframes
df_validation=df1[df1['Building']=='KM7_5_V']
###
df1=df1[df1['Building']=='KM7_5']
```

In [22]:
```
#df1.columns.tolist()
```

In [23]:
```
COLS=['Area','Number of windows','Area of windows','Vertical WWR',
      'U-value of exterior wall','U-value of exterior roof','U-value of adjacent wall',
      'Occupant type 1','Occupant type 2','Occupant type 3',
      'Single glazing window',
      'Double glazing window',
      'East','East-west','North','North south','South','West',
      'Attic','Ground floor','Middle floor',
      'Percentage of hours above Cat-II EN 16798',
      'Givoni thermally stressful','Maximum Consective hours above 27 Avg',
      'Peak operative temperature Avg','RE2020 degree hour',
      'RE2020 situation']
```

In [24]:
```
df1=df1[COLS]
df_validation=df_validation[COLS]
```

Parameters such as WinType_2 could be removed from the list because if WinType_1 is "0" it means WinType_2 is "1" # same principal applies to Occupant types. we keep "Occup_2", and "Occup_3" but remove occup_1. we could remove "1st floor" and "East" orientation as well # Now that we have generated parameters we are going to generate polynomial and interaction features

In [25]:
```
## if a cell is empty (the results, fill it with zero)
df1=df1.fillna(0)
df_validation=df_validation.fillna(0)
```

In [26]:
```
##
InputColumns=['Area','Number of windows','Area of windows','Vertical WWR',
              'U-value of exterior wall','U-value of exterior roof','U-value of adjacent wall',
              'Occupant type 2','Occupant type 3',
              'Double glazing window',
              'East-west','North','North south','South','West',
              'Attic','Ground floor','Middle floor']
### Output columns without the RE2020 situation - because it is categorical and logistic regression will be teste
OutputColumns=['Percentage of hours above Cat-II EN 16798',
               'Givoni thermally stressful',
               'Maximum Consective hours above 27 Avg',
               'Peak operative temperature Avg',
               'RE2020 degree hour']
##
CategoricalOutput=['RE2020 situation']
```

In [27]:
```
### if attic only un-comment the following expressions
#unwanted_num={'Attic','Ground floor','Middle floor','East-west','East','North','South','West'}
#InputColumns = [ele for ele in InputColumns if ele not in unwanted_num]
#df1=df1[(df1.Attic==1)]
#df_validation=df_validation[(df_validation.Attic==1)]
```

In [28]:
```
### if only middle floor or only ground floor, un-comment the following expressions
#unwanted_num={'Attic','Ground floor','Middle floor','East','East-west','North south'}
#InputColumns = [ele for ele in InputColumns if ele not in unwanted_num]
#df1=df1[(df1['Middle floor']==1)] # or Ground floor
#df_validation=df_validation[(df_validation['Middle floor']==1)]
```

In [29]:
```
### if ground floor AND middle floor, un-comment the following expressions
unwanted_num={'Attic','Middle floor','East','East-west','North south',
              'U-value of exterior roof','Number of windows'}
InputColumns = [ele for ele in InputColumns if ele not in unwanted_num]
df1=df1[(df1['Attic']!=1)]
df_validation=df_validation[(df_validation['Attic']!=1)]
```

In [30]:
```
from sklearn.preprocessing import PolynomialFeatures
import scipy.special
```

In [31]:
```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
```

Checking correlations and Splitting data into training and testing data

In [32]:
```
## Input parameters for regression model
Xr =df1[InputColumns]
Y_true = df1[OutputColumns]
Y_cat=df1[CategoricalOutput]
```

Checking correlation in the input data

In [33]:
```
from sklearn import metrics
```

Sklearn MultiOutput Regressor - Gradient Boosting Regressor training for KM7_5

In [34]:
```
from sklearn.multioutput import MultiOutputRegressor
```

In [35]:
```
from sklearn.ensemble import GradientBoostingRegressor
```

In [36]:
```
##Tuned model
gbr = GradientBoostingRegressor(learning_rate=0.1,n_estimators=100)
model = MultiOutputRegressor(estimator=gbr)
model.fit(Xr,Y_true)
#Predicted = model.predict(X_test)
```

Out[36]:
```
MultiOutputRegressor(estimator=GradientBoostingRegressor())
```

In [37]:
```
model.predict(Xr.iloc[0:1])
```

Out[37]:
```
array([[1.34773608e+01, 4.51118714e-02, 2.70234760e+02, 3.12722593e+01,
        1.07060758e+03]])
```

Multinomial Logistic regression to predict RE2020 situation

In [38]:
```
from sklearn.linear_model import LogisticRegression
```

In [39]:
```
# Training on all dataset using the tuned model
MLR_model = LogisticRegression(multi_class='multinomial',solver='lbfgs', penalty='l2', C=10)
MLR_model.fit(Xr,Y_cat)
```

```
C:\Users\obaidullah.yaqubi\AppData\Roaming\Python\Python38\site-packages\sklearn\utils\validation.py:72: DataConv
ersionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samp
les, ), for example using ravel().
  return f(**kwargs)
C:\Users\obaidullah.yaqubi\AppData\Roaming\Python\Python38\site-packages\sklearn\linear_model\_logistic.py:762: C
onvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[39]:
```
LogisticRegression(C=10, multi_class='multinomial')
```

Prediction the rest of build-stock using the GB regressor and multinomial logistic regression

In [40]:
```
import geopandas as gpd
import math
```

In [41]:
```
data=gpd.read_file(r'D:\Donnees Bat. Cerema\BDTOPO_CEREMA_2\buildings_clustered_CRS_corrected.shp')
```

```
data.head(2)
```

Out[41]:

	OrigIndex	UID	ID	USAGE1	Unique_ID	T_perimi	NEIGHBORS	indices	Total_area	Total_free	NumNeigh	A
0	0	51664	BATIMENT0000000302944746	RÃ?Æ?Ã?â???Ã?Æ?Ã¢â?¬Å¡Ã?â??Ã?Â©sidentiel	51665	35	51664, 51666	51663, 51665	344.774418	219.099459	2	
1	11	51621	BATIMENT0000000302926068	RÃ?Æ?Ã?â???Ã?Æ?Ã¢â?¬Å¡Ã?â??Ã?Â©sidentiel	51622	39	51557, 51621	51556, 51620	319.118976	232.048138	2	

In [42]:

```
data.rename(columns={'Cerema_jan':'Year of construction'},inplace=True)
```

In [43]:

```
data['Area']=round(data['A_building'],1)
data['Year of construction']=data['Year of construction'].astype(int)

#Window to wall ratio as function of year of construction for cluster KM7_5

WWR_List=[]

for i, row in data.iterrows():
    Y=data['Year of construction'][i]
    C=data['KM7'][i]

    if Y >1900 and C==5:
        W=0.25
    elif Y <=1900 and C==5:
        W=0.15
    else:
        W=None
    WWR_List.append(W)

data['Vertical WWR']=WWR_List
```

In [44]:

```
## Area of windows
data['Area of windows']=data['Vertical WWR']*data['T_perimi']*3
```

In [45]:

```
## U-value of retrofitted and non-retrofitted external wall and adj wall based on Tabula info
Ext_wall_r=[]
Ext_wall=[]

Ext_roof_r=[]
Ext_roof=[]

Adj_wall=[]

for i, row in data.iterrows():
    y=data['Year of construction'][i]
    C=data['KM7'][i]

    if y <1915 and C==5:
        wall_r = 0.24
        wall=1.7
        Roof_r=0.2
        Roof=1.3
        Adj=1.2

    elif y<1949 and C==5:
        wall_r =0.24
        wall=2.3
        Roof_r=0.1
        Roof=2.42
        Adj=1.2

    elif y<1968 and C==5:
        wall_r =0.19
        wall=3
        Roof_r=0.1
        Roof=2.42
        Adj=1.2

    elif y<1975 and C==5:
```

```
        wall_r =0.19
        wall=0.78
        Roof_r=0.23
        Roof=0.76
        Adj=1.2

    elif y<1982 and C==5:
        wall_r =0.19
        wall=0.61
        Roof_r=0.1
        Roof=0.49
        Adj=1.2

    elif y<1990 and C==5:
        wall_r =0.19
        wall=0.36
        Roof_r=0.23
        Roof=0.62
        Adj=1.2

    elif y<2000 and C==5:
        wall_r =0.19
        wall=0.36
        Roof_r=0.23
        Roof=0.43
        Adj=1.2

    elif y<2006 and C==5:
        wall_r =0.19
        wall=0.33
        Roof_r=0.23
        Roof=0.28
        Adj=1.2

    elif y<2013 and C==5:
        wall_r =0.19
        wall=0.3
        Roof_r=0.19
        Roof=0.19
        Adj=1.2

    elif y<2020 and C==5:
        wall_r =0.12
        wall=0.19
        Roof_r=0.12
        Roof=0.28
        Adj=1.2

    else:
        wall_r =None
        wall=None
        Roof_r=None
        Roof=None
        Adj=None

    Ext_wall_r.append(wall_r)
    Ext_wall.append(wall)
    Ext_roof_r.append(Roof_r)
    Ext_roof.append(Roof)
    Adj_wall.append(Adj)

data['U-value of exterior wall retrofitted']=Ext_wall_r
data['U-value of exterior wall']=Ext_wall

data['U-value of exterior roof retroffited']=Ext_roof_r
data['U-value of exterior roof']=Ext_roof

data['U-value of adjacent wall']=Adj_wall
```

In [46]:

```
#    Occupant_type_2=0 # Default value is occupant type 1=1-only takes value 0 or 1: 0 means non and 1 yes
#    Occupant_type_3=0 # Default value is occupant type 1. only takes value 0 or 1: 0 means non and 1 yes
#    Double_glazing_window=1 # Default value is 1 - only takes value 0 or 1: 0 means No and 1 Yes
#    North=1 # default value
#    South=0 # Default value
#    West=0 # default value
#    Ground_floor=0# deault value 0. when it is zero, then the studied issue is middle floor
```

In [47]:

```
# Set default values from here..
## Prediction function of continuous dependent variables
def predict_GBR(Area, Area_of_windows, Vertical_WWR, U_value_exterior_wall,
                U_value_adjacent_wall,
```

```
                Occupant_type_2=0,Occupant_type_3=0,
                Double_glazing_window=1,
                North=0,South=0,West=0,
                Ground_floor=0):

    array=np.array([Area, Area_of_windows, Vertical_WWR, U_value_exterior_wall,
                    U_value_adjacent_wall, Occupant_type_2, Occupant_type_3,
                    Double_glazing_window, North, South, West, Ground_floor]).reshape(1,12)

    Predict_input=pd.DataFrame(array)

    Predict_output=model.predict(Predict_input)

    return Predict_output
```

In [48]:
```
predict_GBR(109,25,0.12,U_value_exterior_wall=0.3,U_value_adjacent_wall=1.2)[0][0]
```

Out[48]: 32.3537809436301

In [49]:
```
## Prediction function of categorical variables using multinomial logistic regression
def predict_MLR(Area, Area_of_windows, Vertical_WWR, U_value_exterior_wall,
                U_value_adjacent_wall,
                Occupant_type_2=0,Occupant_type_3=0,
                Double_glazing_window=1,
                North=0,South=0,West=0,
                Ground_floor=0):

    array=np.array([Area, Area_of_windows, Vertical_WWR, U_value_exterior_wall,
                    U_value_adjacent_wall, Occupant_type_2, Occupant_type_3,
                    Double_glazing_window, North, South, West, Ground_floor]).reshape(1,12)

    Predict_input=pd.DataFrame(array)

    Predict_output=MLR_model.predict(Predict_input)

    return Predict_output
```

In [50]:
```
predict_MLR(109,25,0.12,U_value_exterior_wall=0.3,U_value_adjacent_wall=1.2)[0]
```

Out[50]: 'Not_regulatory'

In [51]:
```
EN_adaptive=[]
GivoniIndex=[]
Consecutive=[]
Peak_Temper=[]
RE2020_DegH=[]
RE_Situation=[]

for i, row in data.iterrows():

    C=data['KM7'][i] #Cluster number
    Surface=data['Area'][i]
    Win_area=data['Area of windows'][i]
    Win_ratio=data['Vertical WWR'][i]
    Ext_wall=data['U-value of exterior wall retrofitted'][i]# Retrofitted or non-retrofitted buildings
    Ext_roof=data['U-value of exterior roof retroffited'][i]# Retrofitted or non-retrofitted buildings
    Adj_wall=data['U-value of adjacent wall'][i]
    # Assumptions
    Occupant_type_2=0
    Occupant_type_3=1
    Double_glazing_window=1
    North=1
    South=0
    West=0
    Ground_floor=0

    if Surface >20 and Surface <400 and C==5:
        #Continuous outputs
        Answer=predict_GBR(Area=Surface,Area_of_windows=Win_area, Vertical_WWR=Win_ratio,
                           U_value_exterior_wall=Ext_wall,
                           U_value_adjacent_wall=Adj_wall,
                           Occupant_type_2=Occupant_type_2,Occupant_type_3=Occupant_type_3,
                           Double_glazing_window=Double_glazing_window,
                           North=North,South=South,West=West,
```

```
                        Ground_floor=Ground_floor)
        ##Categorical output
        Re_Answer=predict_MLR(Area=Surface,Area_of_windows=Win_area, Vertical_WWR=Win_ratio,
                        U_value_exterior_wall=Ext_wall,
                        U_value_adjacent_wall=Adj_wall,
                        Occupant_type_2=Occupant_type_2,Occupant_type_3=Occupant_type_3,
                        Double_glazing_window=Double_glazing_window,
                        North=North,South=South,West=West,
                        Ground_floor=Ground_floor)

        EN_adp=round(Answer[0][0],2)
        Givoni=round(Answer[0][1],4)
        Consec=round(Answer[0][2],1)
        Peak_T=round(Answer[0][3],1)
        RE2020=round(Answer[0][4],1)
        RE_Sit=Re_Answer[0]

    else:
        EN_adp=None
        Givoni=None
        Consec=None
        Peak_T=None
        RE2020=None
        RE_Sit=None

    EN_adaptive.append(EN_adp)
    GivoniIndex.append(Givoni)
    Consecutive.append(Consec)
    Peak_Temper.append(Peak_T)
    RE2020_DegH.append(RE2020)
    RE_Situation.append(RE_Sit)

data['Percentage of hours above Cat-II EN 16798']=EN_adaptive
data['Givoni thermally stressful']=GivoniIndex
data['Maximum Consective hours above 27']=Consecutive
data['Peak operative temperature']=Peak_Temper
data['RE2020 degree hour']=RE2020_DegH
data['RE2020 Situation']=RE_Situation
```

In [52]:
```
#data.loc[data.KM7==5].head(5)
```

In [53]:
```
water=gpd.read_file(r'D:\Donnees Bat. Cerema\BDTOPO_CEREMA_2\water.shp')
water.head(2)
```

Out[53]:

	ID	NATURE	PERSISTANC	ORIGINE	geometry
0	SURF_EAU0000000028807085	Retenue	Permanent	Artificielle	POLYGON Z ((-1.57582 47.24305 27.60000, -1.576...
1	SURF_EAU0000000028806895	Retenue	Permanent	Artificielle	POLYGON Z ((-1.55785 47.25849 5.10000, -1.5580...

In [54]:
```
RSU=gpd.read_file(r'D:\Donnees Bat. Cerema\BDTOPO_CEREMA_2\RSU.shp')
RSU.head(1)
```

Out[54]:

	pk_usr	build_dens	RSU_ID	RSU_Area	RSU_h	A_Building	A_Buildi_1	Vegetation	Vegetati_1	Water_area	Water_pc	A_free_ver	A_free_v
0	180760	0.0	1	175.989	0.1	0.0	0.0	0.0	0.0	0.0	0.0	NaN	Na

In [55]:
```
vegetation=gpd.read_file(r'D:\Donnees Bat. Cerema\BDTOPO_CEREMA_2\vegetation.shp')
vegetation.head(1)
```

Out[55]:

	ID	NATURE	geometry
0	ZONEVEGE0000000301886284	Bois	POLYGON ((-1.58783 47.21959, -1.58782 47.21953...

In [56]:
```
#data.crs
```

In [57]:
```
#water.crs
```

In [58]:
```
#data.loc[data.KM7==5].head()
```

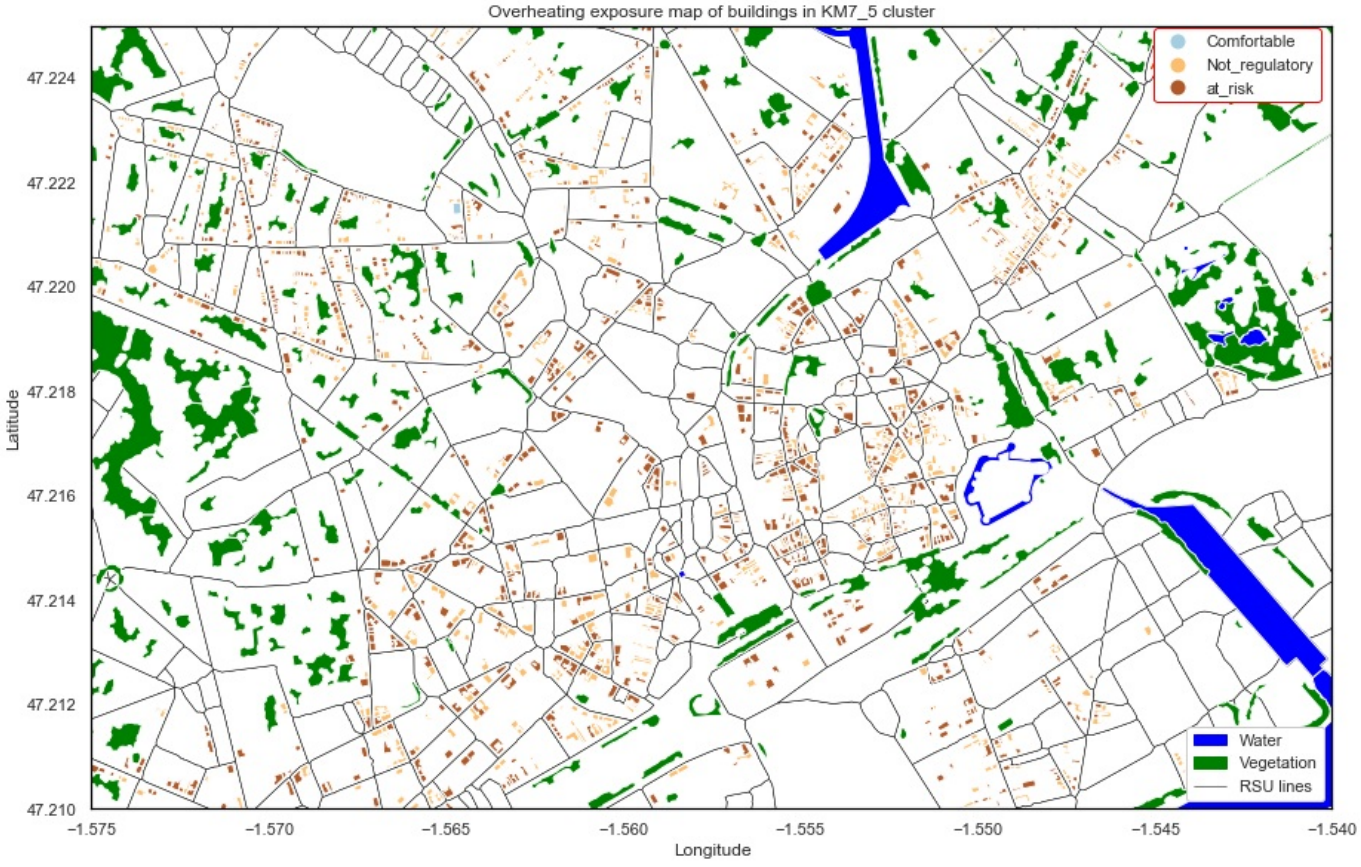
In [59]:
```
#plt.scatter(data['Area of windows'],data['Percentage of hours above Cat-II EN 16798'])
```

Heat exposure map

In [60]:
```
import mapclassify
```

In [61]:
```
import matplotlib.patches as mpatches
import matplotlib.lines as mlines
import matplotlib
```

In [62]:
```
matplotlib.rc('axes',edgecolor='black')

fig,ax=plt.subplots(figsize=(15,15),edgecolor='black')

RSU.plot(ax=ax,edgecolors='black',linewidths=0.5,color='white',label='RSU')
water.plot(ax=ax,color='blue')
vegetation.plot(ax=ax,color='green')

# Map legends
RSU_line = mlines.Line2D([], [], color='black', linestyle='-',linewidth=0.5,label='RSU lines')
blue_patch = mpatches.Patch(color='blue', label='Water')
green_patch=mpatches.Patch(color='green',label='Vegetation')
first_legend=plt.legend(handles=[blue_patch,green_patch,RSU_line],facecolor='white',loc='lower right',framealpha=
plt.gca().add_artist(first_legend)

# Automatic legend of variables (plotting variable)
data.loc[data.KM7==5].plot(ax=ax,cmap='Paired',column='RE2020 Situation',
                           categorical=True,legend=True,
                           legend_kwds={'loc': 'center left', 'bbox_to_anchor':(0.85,0.95)
                                        ,"edgecolor": "red",'facecolor':'white','framealpha':1})
ax.set_xlim(-1.575, -1.54)
ax.set_ylim(47.21, 47.225)
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.set_title('Overheating exposure map of buildings in KM7_5 cluster')
#ax.axis('off')
plt.show()
```

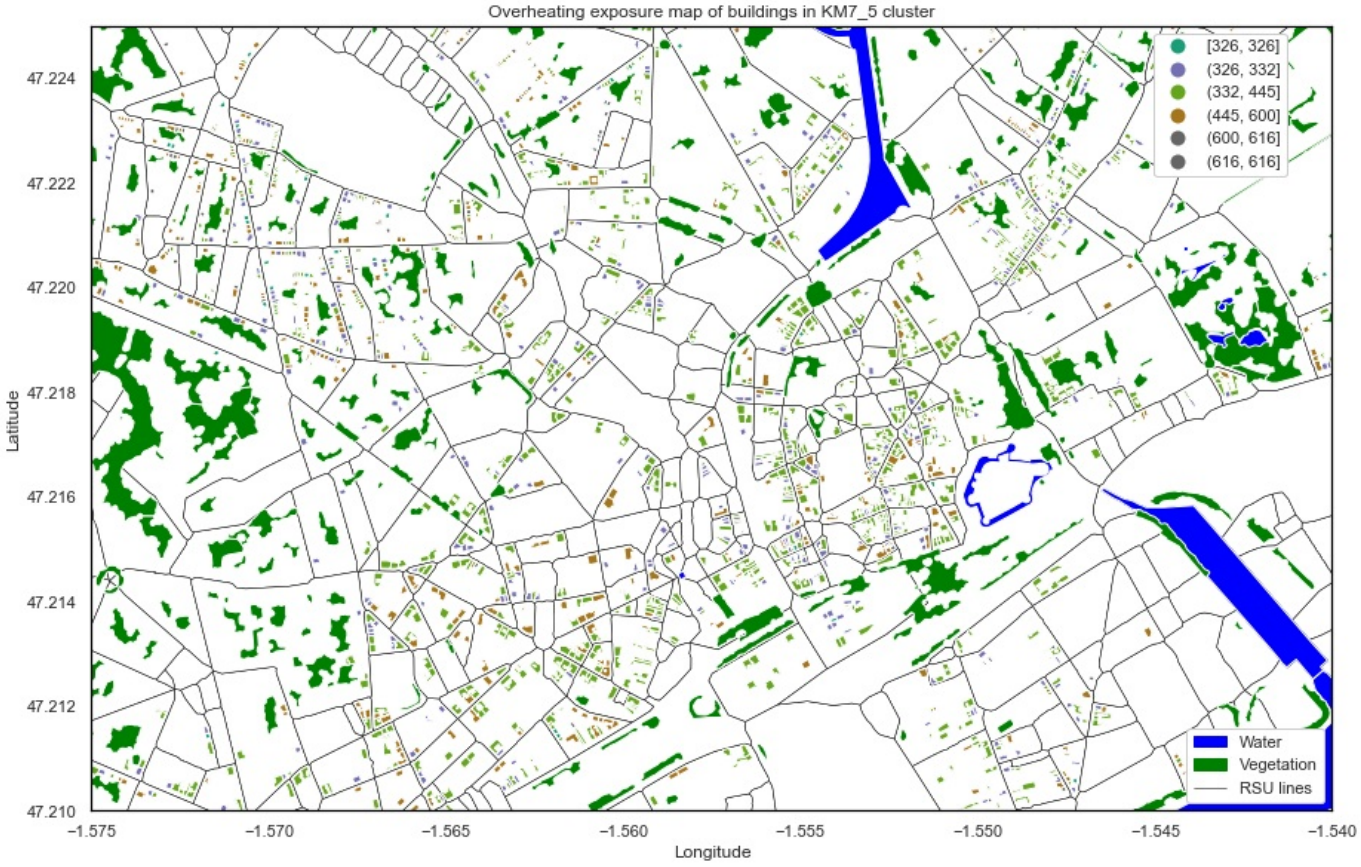


Continuous variable

Continuous variable

In [63]:

```
#matplotlib.rc('axes',edgecolor='black')
fig,ax=plt.subplots(figsize=(15,15),edgecolor='black')

RSU.plot(ax=ax,edgecolors='black',linewidths=0.5,color='white',label='RSU')
water.plot(ax=ax,color='blue')
vegetation.plot(ax=ax,color='green')

# Map legends
RSU_line = mlines.Line2D([], [], color='black', linestyle='-',linewidth=0.5,label='RSU lines')
blue_patch = mpatches.Patch(color='blue', label='Water')
green_patch=mpatches.Patch(color='green',label='Vegetation')
first_legend=plt.legend(handles=[blue_patch,green_patch,RSU_line],facecolor='white',loc='lower right',framealpha=
plt.gca().add_artist(first_legend)

# Automatic legend of variables (plotting variable)
data.loc[data.KM7==5].plot(ax=ax,column='Maximum Consective hours above 27',scheme='Percentiles',cmap='Dark2', le
                           legend_kwds={'loc': 'center left','facecolor':'white','framealpha':1,
                                        'bbox_to_anchor':(0.85,0.9),'fmt': "{:.0f}"})

## schema user_defined
#shape.plot(column='rain', cmap='jet', scheme="User_Defined",
#         legend=True, legend_kwds={'loc': 'upper right','fontsize':'x-small'},
#         classification_kwds=dict(bins=[10,20,30,50,70,100,200]),
#         ax=ax)

ax.set_xlim(-1.575, -1.54)
ax.set_ylim(47.21, 47.225)
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.set_title('Overheating exposure map of buildings in KM7_5 cluster')
#ax.axis('off')
plt.show()
```



In [64]:

```
data.loc[data.KM7==5].head(3)
```

Out[64]:

	OrigIndex	UID		ID	USAGE1	Unique_ID	T_perimi	NEIGHBORS	indices	Total_area	Total_free	NumNeigh
33	81	51563	BATIMENT0000000302925022		RÃ?Æ?Ã?â???Ã?Æ?Ã¢â?¬Å¡Ã?â??	305	51564	32	51550, 51556, 51565,	51549, 51555, 51564,	353.882164	208.940849	4

				Ã? Â©sidentiel			51570	51569			
39	95	51569	BATIMENT0000000302925030	RÃ?Æ?Ã? â???Ã?Æ? Ã¢â? ¬Å¡Ã?â?? Ã? Â©sidentiel	51570	36	51549, 51550, 51564	51548, 51549, 51563	291.448002	184.314811	3
87	211	51693	BATIMENT0000000302924141	RÃ?Æ?Ã? â???Ã?Æ? Ã¢â? ¬Å¡Ã?â?? Ã? Â©sidentiel	51694	60	51681, 51693, 51695	51680, 51692, 51694	352.337755	281.473625	3

In [ ]:

Titre : Contribution au développement d'une méthodologie pour construire des cartes de vulnérabilité à la surchauffe intérieure à l'échelle de la ville intégrant des données sur le changement climatique et l'îlot de chaleur urbain

Mots clés : Changement climatique, surchauffe intérieure, échelle de la ville, carte de vulnérabilité, îlot de chaleur urbain.

Résumé : Le problème de la surchauffe intérieure des bâtiments devient de plus en plus un sujet d'intérêt pour la communauté scientifique ainsi que les décideurs politiques en matière d'urbanisme en raison de l'augmentation de la température moyenne de la Terre, de l'augmentation de la fréquence des événements météorologiques extrêmes, de l'effet d'îlot de chaleur urbain, et le fait que de nos jours la plupart des gens passent la majorité de leur temps à l'intérieur des bâtiments. Le présent travail de recherche porte sur le développement d'une méthodologie pour l'évaluation de la vulnérabilité à la surchauffe intérieure à l'échelle urbaine, visant à soutenir la prise de décision stratégique dans la planification urbaine. Compte tenu de la nature interconnectée des questions à traiter, ce manuscrit commence par un chapitre d'introduction détaillé présentant les concepts clés, les énoncés du problème, l'objectif de la thèse et la méthodologie globale employée. Chaque chapitre suivant est consacré à une partie spécifique du travail de thèse.

Le deuxième chapitre de ce manuscrit porte sur la définition des typologies des bâtiments et l'identification des bâtiments représentatifs utilisés. Le troisième chapitre porte sur la prise en compte du changement climatique et les données sur les îlots de chaleur urbains dans les fichiers météo utilisés dans les simulations. Le quatrième chapitre présente les paramètres du bâtiment qui influencent les performances énergétiques et thermiques ainsi que les indices de mesure de la surchauffe intérieure. Le cinquième chapitre présente les modèles réduits ou métamodèles développés et la manière d'extrapolation des résultats de simulations des bâtiments représentatifs au reste du parc bâti à l'échelle de la ville.

Enfin, la conclusion retrace les principales idées développées au cours de la thèse. Elle pose ensuite les limites de ce travail de thèse dans chacun des domaines abordés et propose quelques pistes complémentaires de réflexion dans les perspectives.

Title : Contribution to the development of a methodology to build indoor overheating vulnerability maps at the city scale integrating climate change data and urban heat island

Keywords: Climate change, indoor overheating, city-scale, vulnerability map, urban heat island.

Abstract: The problem of indoor overheating is increasingly becoming a subject of interest to the scientific community as well as the policy makers in urban planning due to the rise in global average temperature, increase in the frequency of extreme weather events, urban heat island effect, and the fact that nowadays most of people spend the majority of their time indoors. The present research demonstrates a methodology for an urban-scale indoor overheating vulnerability assessment, aimed to support strategic decision making in urban planning for climate-change adaptation policy interventions. Given the inter-connected nature of questions needed to be handled, this manuscript starts with a detailed introduction chapter outlining the key concepts, presenting problem statements, research objective, and overall method employed.

Each subsequent chapter is dedicated to a specific part of the research effort. The second chapter of this manuscript is about building typologies definition and identification of representative buildings. Third chapter is concerned with climate change and urban heat island data. Fourth chapter presents indoor overheating measurement indices and attempts to identify the most influential building parameters through a literature review. Fifth chapter covers surrogate models and how to extrapolate the simulations results of representative buildings to the rest of build stock. Finally, the conclusion traces the main ideas developed during the thesis. It then sets out the limits of the study in each of the areas covered and develops some additional avenues for reflection in the perspectives.