Genetic mutations combinatorics

Raphael Champeimont

THÈSE DE DOCTORAT DE
L'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Raphaël CHAMPEIMONT

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

Combinatoire des mutations génétiques

soutenue le 15 décembre 2014

devant le jury composé de :

Dr. Claudine Devauchelle	Rapporteur
Pr. Alain Denise	Rapporteur
Pr. Bernard Dujon	Examinateur
Dr. François Pénin	Examinateur
Pr. Alessandra Carbone	Directrice de thèse

What is truly revolutionary about molecular biology
in the post-Watson-Crick era is that it has become digital.

Richard Dawkins, in *River out of Eden* (1995)

Contents

Chapitre 1

Résumé en français

Sommaire

1.1 Conservation et coévolution

1.1.1 Contexte biologique

L'ère de la génomique

Il est maintenant bien connu que les acides nucléiques, ADN et ARN, sont les molécules utilisées par les organismes vivants pour stocker leur information génétique. L'ADN a été isolé pour la première fois en 1869 par Friedrich Miescher, puis sa structure en double-hélice a été découverte en 1953 par James Watson et Francis Crick. Depuis, l'intérêt pour la génomique n'a cessé de croître. Lors des dernières décennies, des progrès technologiques spectaculaires ont été réalisés, permettant le séquençage des génomes de nombreuses espèces.

Avec toutes ces données génomiques, on aimerait être capable de prédire l'effet de mutations dans la séquence d'ADN sur les protéines synthétisées, en particulier sur la

perte ou non de leur fonction. Pour répondre à cette question, l'étude des séquences homologues d'un gène chez d'autres espèces apporte des informations précieuses. Dans cette partie, je me suis intéressé à la prédiction de la pathogénicité des mutations sur la protéine P53 dans un premier temps, et sur un grand nombre de protéines impliquées dans des maladies génétiques dans un second temps. Enfin, j'ai utilisé les méthodes de coévolution pour prédire les interactions protéine-protéine du virus de l'hépatite C.

Qu'est-ce que la coévolution ?

Avant d'aller plus loin, définissions les termes conservation et coévolution.

La notion de conservation réfère à une propriété que l'on retrouve conservée entre différentes espèces. Par exemple, dans une séquence protéique, un acide aminé sera dit "conservé" si il est identique dans la plupart des espèces étudiées pour une position donnée, et sera dit "variable" dans le cas inverse. La conservation est souvent la marque d'une propriété importante, que la sélection naturelle a permis de conserver au cours de l'évolution car son absence diminuait le nombre de descendants de l'individu.

La coévolution est l'évolution parallèle de deux entités, chacune étant influencée par l'autre. Par exemple, les parasites et leurs hôtes coévoluent parfois, car le parasite cherche à infecter l'hôte le plus efficacement possible, alors que l'hôte évolue pour contrer les adaptations du parasite. Le parasite à son tour évolue pour s'adapter aux évolutions de l'hôte.

Dans le cadre de cette thèse, la coévolution qui nous intéresse s'effectue au niveau moléculaire. On peut en effet observer la coévolution de deux résidus dans une protéine, ou dans deux protéines différentes, de façon à préserver une interaction entre ces deux résidus. Un changement dans un résidu, disons A vers A', rendra la protéine moins stable et par conséquent les individus avec A' moins adaptés (sélection naturelle). Mais un changement d'un autre résidu (disons B vers B') pourra compenser le changement de A en A' et restaurer une interaction optimale. Les individus avec A/B ou A'/B' seront donc les plus adaptés et seront sélectionnés. A posteriori, on observera donc des espèces avec A et B et d'autres avec A' et B', mais pas les autres variantes. C'est cette "signature" de la coévolution que l'on cherche à détecter lorsqu'on parle de détection de coévolution.

1.1.2 Prédiction des positions critiques de la P53

Rôle dans la cellule

La protéine P53 joue un rôle clé dans la prévention du cancer. À ce titre, elle a parfois été nommée "la gardienne du génome". Son rôle est d'arrêter le cycle cellulaire, voire de provoquer volontairement la mort de la cellule, en cas de détection de risque de tumeur. Cette protéine permet donc d'empêcher la prolifération des cellules cancéreuses. Malheureusement, si cette protéine elle-même est mutée, elle ne peut plus remplir ce rôle et la cellule peut alors dégénérer en tumeur, à condition que d'autres mutations aient eu lieu par ailleurs dans d'autres gènes. Cela explique qu'on retrouve le gène P53 muté dans la plupart des tumeurs.

Mais le séquençage du gène dans les tumeurs apporte une information supplémentaire. En mesurant la fréquence des mutations à chaque position dans le gène P53, on peut en déduire quelles positions sont critiques pour sa fonction, car on sait que dans les tumeurs

la protéine ne remplit plus sa fonction. Inversement, si la mutation n'annulait pas la fonction de la protéine, la cellule n'aurait pas pu former de tumeur et on ne l'observerait pas.

Le gène P53 possède deux paralogues, c'est-à-dire deux autres gènes présents chez l'homme issus d'une duplication d'un même gène chez une espèce ancestrale (en l'occurrence l'ancêtre des vertébrés). Ces paralogues, P63 et P73, ont une fonction différente. En effet, alors que la suppression du gène P53 entraîne une mort prématurée par le cancer chez les souris (Donehower et al., 1992; Jacks et al., 1994), la suppression du gène P63 entraîne de sévères troubles du développement (Yang et al., 1999; Mills et al., 1999), alors que la suppression du gène P73 entraîne d'autres troubles moins importants (Yang et al., 2000).

Prédiction et évaluation

Dans cette partie, mon but était de prédire les positions critiques pour la fonction de la protéine à partir des séquences homologues chez les autres espèces. L'hypothèse est qu'une position non critique verra son acide aminé varier dans différentes espèces, alors qu'à une position critique on devrait s'attendre à la conservation de son acide aminé dans toutes les espèces. Implicitement, cela suppose que la fonction de la protéine est suffisamment similaire dans les espèces étudiées pour qu'une mutation délétère dans une espèce le soit aussi dans une autre.

J'ai donc collecté un certain nombre de séquences homologues, chez diverses espèces animales, pour le gène P53 et ses deux paralogues P63 et P73. Ces séquences forment les données d'entrée qui seront utilisées par la méthode de prédiction. Pour évaluer la performance de la méthode à prédire les positions critiques, j'ai défini un ensemble de 35 positions critiques en prenant les positions les plus mutées dans la base de données de Edlund et al. (2012).

Chaque méthode de prédiction étudiée fournit un score pour chaque position de la protéine, qu'on espère plus grand pour les positions critiques que pour les autres. Pour évaluer la corrélation entre le score et la présence de la position dans l'ensemble des 35 positions critiques, j'ai tracé une courbe ROC et une courbe PR (Precision-Recall) et j'ai mesuré l'aire sous ces courbes (qu'on appelle respectivement AUROC et AUPR). En raison du faible nombre de positions critiques (35) comparé au nombre de positions total (393), l'AUPR est un meilleur indicateur de performance.

Résultats de l'évaluation

J'ai évalué de nombreuses méthodes basées sur la mesure de la conservation, c'est-à-dire de la tendance pour une position dans la protéine à avoir le même acide aminé dans beaucoup d'espèces. La méthode la plus naïve consiste à compter simplement le nombre de paires d'espèces ayant le même acide aminé à une position donnée, et utiliser ce nombre pour classer les positions de la plus à la moins probablement critique. Cette méthode permet déjà d'obtenir une AUPR de 0.671 et une AUROC de 0.929.

La prise en compte de la similarité des propriétés physico-chimiques des différents acides aminés permet d'augmenter la qualité de la prédiction, avec une AUPR de 0.697 et une AUROC de 0.959. J'ai également conçu moi-même une nouvelle méthode, conservationTree, qui permet de pondérer la similarité entre acides aminés par la proximité entre séquences sur l'arbre phylogénétique. Cela permet de considérer des positions conservées

différemment dans chaque sous-arbre comme fortement conservées, alors qu'elles seraient considérées comme faiblement conservées sinon. En particulier, comme on considère les paralogues de la P53, on a souvent des résidus conservés dans les sous-arbres correspondant à chacun des paralogues. Cette méthode est la meilleure selon la courbe PR, avec une AUPR de 0.720. Par contre, elle est inférieure à la conservation physico-chimique sur la courbe ROC, avec une AUROC de 0.932.

Une des hypothèses originales du sujet de la thèse était que l'étude des positions coévoluant permettrait d'apporter davantage d'informations pour cette prédiction. Les positions ayant une signature de coévolution correspondraient donc à des interactions au sein de la protéine, et serait donc des résidus critiques. Certaines méthodes de détection de coévolution, telles SCA (Lockless and Ranganathan, 1999) et ELSC (Dekker et al., 2004) ne permettent de détecter que la coévolution mais pas la conservation. Après analyse, il s'avère que ces méthodes ne marchent pas du tout pour prédire les positions critiques de la P53. Cela s'explique par le fait qu'une grande partie de ces positions sont strictement conservées, et ne peuvent donc pas varier ensemble puisqu'elles ne varient pas du tout. En revanche, les méthodes MST (Baussand and Carbone, 2009) et BIS (Dib and Carbone, 2012b) auraient pu produire de meilleures prédictions, car elles prennent en compte les phénomènes de conservation et de coévolution. Néanmoins, leurs prédictions sont bien inférieures à celles produites avec les méthodes de conservation présentées précédemment.

En plus des méthodes citées ci-dessus, j'ai également testé des méthodes basées sur la structure 3D, mais celles-ci n'ont pas permis d'obtenir de meilleurs résultats. Enfin, la méthode PolyPhen (Adzhubei et al., 2010) a produit des résultats inférieurs, mais on peut les expliquer par la nécessité pour PolyPhen de préciser le résidu par lequel on remplace le résidu normal à une position, ce qu'on ne fait pas ici (on évite le problème en moyennant ou en prenant le maximum sur les 19 résidus).

Autres paramètres importants

D'autres paramètres qui déterminent l'ensemble de séquences interviennent dans la prédiction. Il ressort de l'analyse que le meilleur choix de base de données est RefSeq (Pruitt et al., 2007), qu'il est utile de considérer les séquences avec un faible nombre de résidus identiques à la séquence de référence humaine (20%), et qu'il est utile de considérer les paralogues P63 et P73. En revanche, d'autres paramètres ont une influence faible sur les prédictions, comme le logiciel utilisé pour aligner les séquences ou générer l'arbre phylogénétique, ou le choix de la couverture minimale de la séquence humaine.

Conclusion

Il est possible de prédire avec une précision importante les positions critiques de la protéine P53. Les meilleurs méthodes pour cela sont basées sur la conservation des résidus dans différentes espèces et sont améliorées par certaines pondérations, comme la prise en compte des propriétés physico-chimiques ou la distance dans l'arbre phylogénétique.

1.1.3 Prédictions des mutations pathogènes

Méthodologie

Afin de tester les différentes méthodes de prédiction à large échelle sur un grand nombre de protéines, j'ai utilisé la base de données Varibench (Nair and Vihinen, 2013) qui répertorie un grand nombre de mutations (> 40 000) sur différentes protéines avec l'indication de leur pathogénicité. Je me suis restreint à une sous-partie de la base de données, celle des protéines pour lesquelles à la fois des mutations pathogènes et des mutations neutres sont connues, ce qui représente 482 protéines.

La différence fondamentale entre cette comparaison de méthodes et celle de la P53 réside dans la spécification du résidu de remplacement. En effet, dans le cas de cette nouvelle comparaison, on demande à la méthode de prédire l'effet du remplacement du résidu à une position p par le résidu x, où x est un des 19 autres acides aminés possibles, alors que pour l'évaluation des méthodes avec la P53, seule p était donnée à la méthode de prédiction. La spécification du résidu x est utilisée par certaines méthodes comme PolyPhen, qui peuvent donc en tirer profit pour faire des prédictions plus fiables. Par contre, les méthodes qui mesurent la conservation n'en tiennent pas compte.

Il y a deux manières d'évaluer la performance des différentes méthodes. On peut évaluer la capacité de la méthode à distinguer les mutations pathogènes des mutations neutres sur chaque protéine, soit prendre toutes les mutations en compte en même temps et évaluer la capacité d'une méthode à distinguer les deux cas. Le deuxième choix est préférable, car il permet de tester la méthode avec un seuil global (pour distinguer les scores haut des scores bas), alors que dans le premier cas, on suppose implicitement que le seuil peut être adapté idéalement pour chaque protéine, ce qui n'est pas réaliste.

FreqDiff et PSIC

La possibilité de prendre en compte le résidu de remplacement m'a permis de tester une nouvelle méthode que j'ai conçue, "FreqDiff", qui consiste simplement à comparer la proportion des espèces présentant l'acide aminé retrouvé chez l'homme non malade à la proportion de l'acide aminé de remplacement. Le score FreqDiff est alors simplement la différence entre ces deux proportions.

Le logiciel PSIC (Sunyaev et al., 1999) est basé sur une idée similaire, mais avec un modèle probabiliste plus complexe. Le score PSIC est utilisé entre autres par PolyPhen (Adzhubei et al., 2010) qui le prend en compte, au même titre que des propriétés structures (accessibilité au solvant), et des annotations (pont disulfure, structure secondaire).

Résultats

Au final, PolyPhen-2 (variante HumVar) surpasse les autres méthodes, avec une AUROC de 0.824. Si l'on utilise seulement le score PSIC utilisé par PolyPhen, la prédiction est de qualité inférieure (AUROC=0.788), ce qui montre que les autres informations utilisées par PolyPhen apportent un gain notable. Il est intéressant de constater que si on applique la méthode PSIC à notre ensemble de séquences, et non pas à celui proposé par PolyPhen (qui utilise son propre algorithme pour choisir les séquences homologues), on obtient un résultat presque identique (AUROC=0.802), sans toutefois dépasser celui de PolyPhen complet.

Notre méthode FreqDiff, à laquelle on ajoute une normalisation Z-score, obtient un score quasiment identique à celui de PSIC (AUROC=0.802), montrant que malgré sa simplicité, cette méthode fournit des résultats intéressants. Ces prédictions sont clairement supérieures à celle de la simple conservation sans prise en compte de l'acide aminé de remplacement (AUROC=0.790).

Contrairement au cas de la P53, la prise en compte des propriétés physico-chimiques n'améliore pas la qualité de la prédiction. De même, la méthode conservationTree n'améliore pas la prédiction non plus. Il faut préciser qu'ici, on n'a pas pris en compte les paralogues, ce qui était la raison de la supériorité de conservationTree dans le cas de la P53.

Enfin, il est aussi intéressant d'évaluer les différentes méthodes dans le cas d'une prédiction à 3 états : pathogène, neutre et non décidé. En fixant les taux de faux positifs et faux négatifs à 10%, on peut mesurer la performance des différentes méthodes pour prédire la plus grande proportion des mutations et en classer le moins possible comme "non décidées". Là encore, on retrouve que la méthode PolyPhen donne les meilleurs résultats, avec un taux de vrais positifs de 52% et un taux de vrais négatifs de 53%.

Conclusion

PolyPhen-2 est la méthode la plus efficace parmi celles testées pour prédire la pathogénicité des mutations. Il s'agit d'une méthode basée sur la conservation majoritairement, avec néanmoins d'autres critères qui permettent d'améliorer la performance. Les taux de vrais positifs et négatifs autour de 50% montrent néanmoins qu'il reste une grande partie des mutations pour lesquelles ont est incapable de faire une prédiction. Enfin, comme avec la P53, l'intégration de la coévolution n'a pas permis d'augmenter la qualité des prédictions.

1.1.4 Prédictions des interactions protéine-protéine chez HCV

L'hépatite C est un problème de santé majeur à l'échelle mondiale, avec entre 130 et 180 millions de personnes infectées (0,84% en France). Il n'existe pas de vaccin contre le virus de l'hépatite C, contrairement aux hépatites A et B. Néanmoins, un progrès important dans la lutte contre cette maladie a été fait avec la mise sur le marché d'un nouveau médicament en décembre 2013 (Solvadi), avec des taux de réussite allant de 50% à 90% des patients. Malgré cela, la lutte contre le virus de l'hépatite C (HCV) est loin d'être terminée.

Ici nous avons tenté d'aider la découverte d'interactions entres les différentes protéines du virus. Celui-ci est composé de 10 protéines, qui ont la particularité d'être synthétisées en une seule fois, car étant codées par un même gène. La polyprotéine ainsi obtenue est ensuite découpée par des enzymes.

Analyse avec BIS

Nous avons utilisé la méthode BIS (Dib and Carbone, 2012b) afin de détecter les résidus ayant une signature de coévolution. Ces résidus peuvent être au sein d'une même protéine ou dans deux protéines différentes. Nous avons utilisé différents ensembles de séquences, provenant de différents génotypes du virus. Le programme BIS calcule des groupes de positions ayant été détectées comme coévoluant entre elles. Certains de ces groupes peuvent être éliminés car ils contiennent un trop grand nombre de positions, et ne permettraient donc pas de déduire des interactions protéine-protéine spécifiques.

L'analyse avec BIS permet de prédire plusieurs centaines d'interactions. Afin de résumer cette information, nous avons calculé une matrice carrée où les lignes et les colonnes correspondent aux différents domaines de toutes les protéines. La ligne i et la colonne j contient alors le nombre de positions détectées indiquant une possible interaction entre un résidu du domaine i et un résidu du domaine j.

Comme il n'est pas possible de tester expérimentalement toutes ces prédictions, nous avons utilisé une autre technique de validation. Notre méthode prédit aussi bien les interactions inter qu'intra-protéines, or la structure 3D de certaines protéines étant connue, il est donc possible de vérifier si les interactions intra-protéine sont correctes.

Pour vérifier la qualité des prédictions, j'ai effectué un test statistique pour vérifier si les paires de positions prédites par BIS étaient plus souvent proches dans la protéine qu'attendu par hasard. Le résultat est significatif avec une p-value inférieur à 1%.

1.1.5 Filtrage des séquences avec PruneTree

Algorithme

PruneTree est un programme que j'ai développé en collaboration avec Anne Lopes qui permet de supprimer d'un ensemble de séquences homologues celles qui sont trop isolées dans l'arbre phylogénétique. La méthode travaille sur un arbre phylogénétique inféré à partir des séquences. Elle calcule la distribution des distances entre paires de feuilles (correspondant aux séquences), en particulier la moyenne μ et l'écart-type σ, et supprime les feuilles trop isolées, c'est-à-dire présentant un nombre de voisins inférieur à un seuil paramétrable. Deux feuilles sont considérées comme voisines si elles sont séparées par une distance paramétrable elle aussi. L'opération est ensuite répétée jusqu'à ce qu'on ne supprime plus de feuilles.

Les deux paramètres sont r et α. Le paramètre r est le nombre de voisins proches minimum en proportion du nombre total de feuilles. Le paramètre α est utilisé pour définir le seuil $M = \mu + \alpha\sigma$ qui est la distance maximum pour que deux feuilles soient considérées comme proches.

Validation avec Guidance

Afin de valider la méthode dans sa capacité à améliorer la qualité des alignements de séquences, et de définir de bons paramètres par défaut, nous avons testé PruneTree sur la base de données PFAM (Finn et al., 2014). Pour chaque famille de PFAM, nous avons généré une famille réduite en appliquant PruneTree. Nous avons ensuite comparé avec Guidance (Penn et al., 2010) le score de qualité d'alignement chaque famille originale avec la famille réduite par PruneTree. Nous avons pu constater une nette amélioration. Nous avons également étudié les raisons de la différence d'efficacité de PruneTree en fonction des familles. Il en ressort que les familles contenant des eukaroytes présentent une nette amélioration de la qualité de leur alignement après traitement par PruneTree.

Exemple d'application

Pour montrer l'intérêt pratique de PruneTree, nous l'avons appliqué à la détection de coévolution dans la protéine A – domaine B, comme cela avait était fait par Dib and

Carbone (2012b). Les auteurs avaient alors enlevé manuellement un sous-ensemble des séquences. Avec PruneTree, les séquences supprimées automatiquement correspondent pour la plupart à celles éliminées à la main. On retrouve alors la nette amélioration des résultats que l'on observait déjà avec le nettoyage manuel.

1.2 Transformées de Fourier et analyse des génomes

1.2.1 Analyse de Fourier sur les génomes

Les transformées de Fourier discrètes sont un outil mathématique remarquable pour l'analyse de nombreux phénomènes périodiques. J'ai ici conçu deux méthodes les utilisant afin d'analyser les phénomènes périodiques dans les génomes.

Analyse d'un ensemble de positions

En s'inspirant d'une idée originalement développée par Wright et al. (2007), Mathelier and Carbone (2010) ont développé une méthode de détection de périodicité pour un ensemble de positions sur les génomes circulaires. J'ai alors adapté cette méthode aux génomes composés de plusieurs chromosomes linéaires.

Supposons que l'on ait, par exemple, déterminé la position des hotspots de recombinaison (zones où des recombinaisons ont eu souvent lieu), et qu'on cherche à savoir si ces positions ont tendance à être espacées d'un même nombre de nucléotides.

La méthode que j'ai développée commence par calculer les distances entre toutes les paires de positions situées sur le même chromosome. On calcule ensuite un histogramme de toutes ces distances auquel on applique l'algorithme FFT qui effectue une transformée de Fourier discrète.

En appliquant cette méthode aux hotspots de recombinaisons de *S. cerevisiae*, on trouve une période importante à 15 400 nucléotides. Afin de vérifier qu'il s'agit d'un résultat statistiquement significatif, j'ai construit plusieurs modèles aléatoires qui m'ont permis de générer des fausses données auxquelles j'ai appliqué la même méthode d'analyse. Pour les hotspots précédemment cités, les différents modèles adéquats donnent une p-value de 10^{-4} ou inférieure. Le résultat est donc statistiquement significatif.

Analyse des résultats

Afin d'analyser plus en profondeur les résultats de la transformée de Fourier, j'ai développé différentes méthodes pour visualiser la périodicité des positions, en particulier en projetant sur un axe la position modulo la période. Cela m'a permis de tester sur chaque chromosome la distribution périodique selon la période de 15.4 kb trouvée précédemment pour les hotspots de *S. cerevisiae*. On retrouve alors cette période de manière significative sur les chromosomes 7, 8, 9, 10 et 16 (p-value inférieure ou égale à 1%).

Analyse locale de la périodicité

La seconde méthode d'analyse que j'ai développée permet de détecter la périodicité d'une distribution le long du génome même si celle-ci n'existe que par endroits. J'avais originalement développé cette méthodologie dans le but d'analyser les signaux de recombinaison

chez *S. cerevisiae*, comme la quantité de protéines de recombinaison Red1 fixées à l'ADN tout le long du génome (expérience ChIP-on-chip).

Le principe de l'analyse consiste à déplacer une fenêtre le long du génome et à effectuer la transformée de Fourier de la distribution sur cette fenêtre. Des détails techniques importants doivent être pris en compte pour que cette analyse donne des résultats précis, comme expliqué dans le rapport complet.

Après avoir effectué cette analyse sur tout le génome, on obtient des milliers de transformées de Fourier. Pour extraire l'information intéressante, j'ai décidé de ne prendre en compte que la période ayant le coefficient le plus fort sur chaque fenêtre. Ensuite, grâce à une normalisation par la quantité de signal totale, j'ai pu extraire le sous-ensemble des fenêtres sur lesquelles une période importante est trouvée. Lorsqu'on regarde les courbes, on peut observer que cette notion correspond bien à l'intuition que l'on a d'une périodicité forte.

L'analyse de distribution de la protéine Red1, qui forme l'axe chromosomique lors de la méiose chez *S. cerevisiae*, montre des zones périodiques, ce qui correspond au modèle de boucles connu pour cet axe. Il n'est pas évident de donner une explication biologique de cette périodicité. Néanmoins, nous avons créé un modèle biologique qui permet de prédire ces distributions (voir le modèle SPoRE plus loin).

En revanche, j'ai appliqué cette méthodologie avec succès sur des données produites par une autre équipe du laboratoire qui travaille sur *Phaeodactylum tricornotum*. Leur expérience a permis de mesurer la quantité de petits ARN synthétisés et de les aligner le long du génome. Le signal étudié est donc la couverture (nombre d'ARN détectés) le long du génome. On retrouve un grand nombre de zones périodiques, qui couvrent une petite partie du génome (1%). Les périodes trouvées sont majoritairement dans l'intervalle 180-200 nucléotides. Ces résultats ont été publiés dans *BMC Genomics* (Rogato et al., 2014).

1.2.2 SPoRE : Un modèle pour les protéines de recombinaison

Après avoir analysé la périodicité de la distribution des protéines de recombinaison, nous avons conclu qu'il n'y avait pas de périodicité générale, mais qu'il y avait néanmoins une distribution non aléatoire de ces protéines. Nous avons donc essayé de construire un modèle pour cette distribution, nommé SPoRE (pour "SPots of REcombination"), que nous présentons ici.

La méiose

Lors de la méiose, les chromosomes forment une structure bien particulière. En certains points du génome, des protéines se fixent (cohésine, Red1, Hop1), puis ces protéines se rapprochent les unes des autres, formant ainsi l'axe du chromosome. Les axes des deux chromosomes homologues sont ensuite joints par d'autres protéines pour former le complexe synaptonémal. Le reste de l'ADN forme alors des boucles, où chaque boucle est la partie de l'ADN qui se trouve entre deux sites de fixation. La distribution de ces protéines a été mesurée par (Panizza et al., 2011) par la méthode ChIP-on-chip. Comme nous l'avons vu précédemment, la distribution de ces protéines est parfois localement périodique, ce qui signifie qu'on trouve plusieurs boucles de même longueur à la suite. Mais nous pouvons dire beaucoup plus, en particulier nous pouvons prédire ces sites de fixation et la quantité de protéines fixées, grâce à la méthode SPoRE que j'ai développée.

Par ailleurs, un second phénomène nous intéresse. Les hotspots de recombinaison dont nous avons parlé précédemment sont des zones du génome où le nombre d'évènements de recombinaison (cross-over et non-cross-over) est anormalement élevé. Mais nous pouvons nous intéresser au phénomène biologique à l'origine de ces recombinaisons. Celui-ci est la cassure de l'ADN double-brin par les protéines Spo11 (nom de la protéine chez *S. cerevisiae*). La fréquence de ces cassures tout le long du génome a été mesurée avec grande précision par Pan et al. (2011). On appellera ces cassures "DSB" par la suite (pour Double-Strand Breaks). Comme nous allons le voir, notre programme SPoRE permet également de prédire leur distribution.

Observations

En observant les données produites par Panizza et al. (2011) et Pan et al. (2011) sur *S. cerevisiae*, et en se basant sur les connaissances disponibles dans la littérature, on peut noter un certain nombre de faits nous permettant de construire un modèle pour la distribution des protéines de l'axe chromosomal et des DSB. Notons tout d'abord que les protéines de l'axe Red1 et Hop1 ont une distribution le long du génome quasiment identique. On modélise donc indifféremment l'une ou l'autre.

On peut observer que la distribution des protéines de l'axe présente des pics (maxima locaux) dans les régions intergéniques, mais que cette densité dépend du sens de transcription des gènes. Trois cas sont possibles pour une région intergénique. On peut se trouver du côté du début des deux gènes, on dira alors qu'il s'agit d'une région intergénique divergente. Si on est au début des gènes on dira qu'il s'agit d'une zone divergente. Enfin, si les deux gènes sont orientés dans le même sens, on dira qu'il s'agit d'une zone orientée. Les protéines de l'axe ont une densité plus élevée dans le cas des zones convergentes, une densité intermédiaire dans le cas des zones orientées et une densité faible dans les zones divergentes. Une explication de ce phénomène a été proposée par Glynn et al. (2004) qui ont montré que la cohésine était poussée par les ARN-polymérases à la fin des gènes, expliquant ainsi les distributions dans les régions intergéniques observées.

Pour les DSB, on observe exactement l'inverse, avec une absence dans les régions intergéniques convergentes, la présence de DSB dans les régions orientées, et une quantité deux fois plus grande encore dans les zones divergentes. Une des hypothèses pour expliquer ce phénomène est l'absence de nucléosomes dans les promoteurs des gènes, ce qui favoriserait les DSB dans ces zones.

Il est également connu que le GC-content moyen dans une fenêtre est fortement corrélé avec le nombre de DSB. Une des hypothèses expliquant cela est la correction biaisée en faveur de G/C lors de la réparation des DSB.

Modèle des protéines de l'axe

Le modèle que nous proposons pour les protéines de l'axe est basé sur les faits présentés ci-dessus. Ce modèle consiste à placer les protéines à la fin des gènes dans une quantité proportionnelle à la longueur du gène. Cela est logique d'après l'hypothèse avancée par Glynn et al. (2004). En effet, plus le gène est long, plus le nombre de protéines qui s'y fixent par hasard sera grande, et donc plus le nombre de protéines poussées à la fin du gène le sera aussi. Avec ce modèle, on obtient une corrélation locale avec les données

de Panizza et al. (2011) de 0.58. Sans l'aspect proportionnel, c'est-à-dire avec une quantité fixe, on obtient une corrélation de 0.14, ce qui montre que la prise en compte de ce phénomène est fondamentale.

On peut améliorer légèrement le modèle en considérant une croissance linéaire le long du gène au lieu d'une simple accumulation à la fin des gènes. On obtient une corrélation de 0.63 avec cette méthode. Cela représente une amélioration mais le modèle est plus complexe.

Modèle des DSB

Pour le modèle des DSB, on peut commencer par tester l'équivalent des modèles des protéines de l'axe, mais avec le début des gènes. Si on place un pic de taille fixe au début de chaque gène, on obtient une corrélation locale de 0.34 avec les données de Pan et al. (2011). Si on considère une quantité proportionnelle à la longueur du gène, cette corrélation baisse pour atteindre 0.26, montrant ainsi qu'il n'y a pas de phénomène lié à la longueur du gène.

Mais ici, ce ne sont pas les débuts des gènes à proprement parler (codon "start") qui importent, mais plutôt les promoteurs, placés en amont des gènes. On propose ici d'approximer leur position par le centre des régions intergéniques dans le cas de deux gènes dans le même sens, et par 1/3 de la longueur de la région intergénique en amont du gène dans le cas de gènes divergents. Même en utilisant des pics de taille fixe, ce modèle permet déjà d'obtenir une corrélation de 0.48. Si on suppose que la fréquence de DSB est proportionnelle à la longueur de la région intergénique, la corrélation locale atteint 0.50.

Si enfin on ajoute le GC-content dans une fenêtre, on peut encore augmenter la qualité du modèle, avec une corrélation de 0.58 si on ne prend pas en compte la longueur de la région intergénique, et avec 0.62 si on la prend en compte.

Pics

Une autre manière d'évaluer la qualité des prédictions est de considérer les pics prédits par le modèle et de les comparer aux pics dans les données expérimentales. Pour le modèle des protéines de l'axe, on trouve que 62% des pics prédits sont corrects. Inversement, 62% des pics réels sont correctement prédits. Pour les DSB, 64% des pics prédits sont corrects, et 68% des pics réels sont prédits. Les deux modèles sont donc assez performants dans la prédiction de pics de concentration des protéines de l'axe et des DSB.

Comparaison avec d'autres méthodes

D'autres méthodes comme IDQD (Liu et al., 2012) et iRSpot-PseDNC (Chen et al., 2013) ont été proposées pour prédire les hotspots de recombinaison. Il est important de noter que les hotspots de recombinaison et de DSB ne sont pas deux choses entièrement identiques. Si les DSB sont un pré-requis pour la recombinaison, une zone avec un grand nombre de DSB n'est pas nécessairement un hotspot pour la recombinaison.

Le programme iRSpot-PseDNC a été conçu pour prédire les hotspots de recombinaison. Il utilise pour cela la séquence d'ADN et analyse la fréquence en dinucléotides (paires de nucléotides qui se suivent) car leur hypothèse est que les hotspots ont une distribution différente pour certains dinucléotides par rapport au reste du génome. Les

auteurs ont également testé leur programme sur les hotspots de DSB, en se comparant comme nous aux données de Pan et al. (2011). Mais leur analyse est incomplète car ils n'ont considéré que des exemples positifs (hotspots). Si on considère autant d'exemples négatifs (zones sans DSB) que positifs, leur méthode a alors un taux de réussite de seulement 54%, à comparer à 50% attendus si la prédiction était faite au hasard. Notre modèle a quant à lui un taux de réussite de 84% sur ces mêmes données.

Si on s'intéresse maintenant à la prédiction des hotspots de recombinaison, iRSpot-PseDNC a un taux de réussite de 85% et IDQD un taux de réussite de 80%. Notre modèle n'est pas adapté à cette prédiction car nous modélisons les DSB et non la recombinaison. Néanmoins, on peut utiliser le GC-content, qui est lui-même utilisé par notre modèle, pour effectuer une prédiction. On a alors un taux de réussite de 83%, dépassant la performance de IDQD et se rapprochant de celle de iRSpot-PseDNC. Il semble donc que ces modèles sophistiqués n'aient au final pas réussi à capturer plus d'information que le simple GC-content (qui apparaît dans les fréquences en dinucléotides). De plus, ces taux de réussites sont obtenus en effectuant un apprentissage sur 4/5 des données, et en prédisant le dernier 1/5. Cette méthodologie de test suppose qu'une grande partie de l'information est déjà connue, ce qui n'est pas nécessairement réaliste. À l'inverse, une prédiction basée sur le GC-content ne nécessite aucune connaissance préalable de hotspots.

Prédiction sur d'autres levures

Nous avons utilisé notre modèle pour prédire la distribution des protéines de l'axe et des DSB chez d'autres levures : *Kluyveromyces lactis*, *Lachancea kluyveri* et *Schizosaccharomyces pombe*. Cela permet de montrer que notre modèle ne contient pas de paramètres spécifiques de *S. cerevisiae* et peut s'appliquer à d'autres levures. Néanmoins, il n'est pas possible de comparer les prédictions à de vrais données biologiques dans le cas de *Kluyveromyces lactis* et *Lachancea kluyveri*, car les expériences correspondantes n'ont pas été faites. Par contre c'est possible avec *Schizosaccharomyces pombe*, car une expérience similaire à celle de Pan et al. (2011) a été faite par Fowler et al. (2014). Lorsque nous comparons notre modèle à ces données, la corrélation locale est de seulement 0.36. Cette performance inférieure peut s'expliquer par les grandes différences entre les deux espèces. En effet, chez *Schizosaccharomyces pombe*, les DSB ne se produisent pas dans tous les promoteurs, contrairement à *S. cerevisiae*. Inversement, les régions convergentes peuvent contenir des DSB, contrairement à celles de *S. cerevisiae* qui en possèdent très rarement. Par ailleurs, des motifs d'ADN spécifiques influencent les DSB chez *Schizosaccharomyces pombe*. Ces observations rendent le modèle SPoRE insuffisant pour rendre compte de la totalité des DSB chez cette espèce.

Conclusion

Notre modèle SPoRE permet de prédire la densité en protéines de l'axe et en DSB le long du génome de *S. cerevisiae*. Il peut s'appliquer également à des levures proches mais ces prédictions ne peuvent pas être testées pour le moment du fait de l'absence de données expérimentales. En revanche, le modèle s'applique mal à *Schizosaccharomyces pombe* qui est plus éloignée.

1.3 Autres travaux

1.3.1 Simulation de l'évolution des génomes

J'ai été amené à travailler sur un projet annexe, la simulation de l'évolution des génomes, dans le but d'évaluer la qualité des prédictions de PhyChro, programme développé au laboratoire, permettant de reconstruire un arbre phylogénétique à partir des génomes annotés des espèces.

La particularité de PhyChro est d'utiliser les blocs de synténie, c'est-à-dire des suites de gènes dont les homologues sont successifs également dans une autre espèce. PhyChro a été conçu pour fonctionner avec le programme SynChro (Drillon et al., 2014), qui calcule les blocs de synténie entre différentes espèces à partir de leurs génomes annotés. PhyChro utilise alors cette information pour produire un arbre phylogénétique.

PhyChro a été évalué sur des données réelles issues des vertébrés et des levures, et a permis de reconstruire avec succès les arbres phylogénétiques de ces deux groupes d'espèces. Néanmoins, il peut être utile d'effectuer des simulations, de manière à tester la reproductibilité des résultats et de mesurer la robustesse de la méthode.

Modèle

Afin de modéliser l'évolution des génomes, j'ai considéré un modèle simple dans lequel on part d'un ancêtre commun avec un nombre fixé de gènes et de chromosomes. On créé ensuite un arbre en utilisant un modèle aléatoire, puis on place sur chacune des branches de l'arbre un nombre d'évènements suivant une loi de Poisson dont le paramètre est proportionnel à la longueur de la branche.

Pour chaque évènement, on choisit ensuite un type avec une probabilité définie parmi : inversion (60%), translocation réciproque (29,79%), duplication (5%), délétion (5%), fusion (0,1%), fission (0,1%), duplication complète du génome (0,01%). Ces probabilités ont été choisies pour correspondre approximativement aux estimations faites sur les levures et les vertébrés.

Les inversions, duplication et délétion ont elles-mêmes un paramètre qui est le nombre de gènes moyen à inverser (5), dupliquer (5) ou supprimer (1). Le nombre est alors choisi selon une loi de Poisson. Dans le cas d'une duplication complète du génome, chaque paire de gène créée a 80% de chances voir une des deux copies disparaître et 20% de chances de conserver les deux copies.

Lors de ces simulations, on ne prend pas en compte la composition en nucléotides de chaque gène. Chaque gène est simplement représenté par un numéro, qui indique sa classe d'homologie (deux gènes avec le même numéro sont homologues). Un chromosome est alors simplement une suite de nombre entiers. La présence de plusieurs gènes avec le même numéro dans un même génome est possible du fait des duplications (ce sont des paralogues).

Vertébrés et levures

Afin de simuler une évolution réaliste, deux types des simulations ont été effectuées, imitant l'évolution des vertébrés et des levures. Pour les vertébrés, l'ancêtre est défini avec 18 000 gènes et 23 chromosomes. Pour les levures, l'ancêtre est défini avec 5 000 gènes

et 8 chromosomes. Le nombre de gènes n'a pas besoin d'être rigoureusement exact, on souhaite simplement que l'ordre de grandeur soit réaliste.

Pour définir la fréquence générale des évènements, on se base sur le nombre d'évènements inférés entre les deux espèces les plus éloignées du groupe. On estime alors que le nombre d'évènements entre moyen l'ancêtre et une espèce finale (que l'on observe aujourd'hui) est de l'ordre de 1 000 pour les vertébrés et de 500 pour les levures. Cela permet de fixer le paramètre de la loi de Poisson qui définit le nombre d'évènements.

On fixe également le nombre d'espèces à simuler, 13 pour les vertébrés et 21 pour les levures, de façon à correspondre à l'arbre réel sur lequel PhyChro a été testé précédemment.

Les génomes simulés pour les levures ressemblent à ceux des véritables levures. En effet, en cas de duplication complète du génome, le modèle produit des espèces avec environ 6 000 gènes, ce qui est réaliste pour les levures ayant leur génome dupliqué (*S. cerevisiae* a 6275 gènes par exemple).

Résultats

Afin de tester la qualité des prédictions de PhyChro, on génère 100 simulations pour les vertébrés et 100 pour les levures. Chaque simulation consiste en un arbre et un génome par espèce avec les relations d'homologie entre les gènes de ces génomes. On demande ensuite à PhyChro de trouver l'arbre à partir des génomes, que l'on peut ensuite comparer à l'arbre véritable (qu'on connaît puisque c'est une simulation).

Sur les 100 simulations de levures, PhyChro a reconstruit l'arbre parfaitement dans 61% des cas. Pour les vertébrés, PhyChro a reconstruit parfaitement 79% des arbres. Mais il faut noter que même les arbres incorrects consistent généralement en une seule espèce mal placée.

Pour être plus précis, on peut compter dans chaque arbre le nombre de bipartitions correctes et incorrectes. En effet, en coupant une branche de l'arbre, on définit une bipartition entre les espèces qui sont d'un côté et celles de l'autre. On peut donc définir autant de bipartitions qu'il y a de branches internes dans l'arbre (les branches avec une seule espèce d'un côté ne sont pas intéressantes). Dans chaque arbre donné par PhyChro, on peut donc vérifier si chaque bipartition existe aussi dans l'arbre réel. Ainsi, en prenant en compte les bipartitions des 100 simulations, on trouve que PhyChro a raison dans 97% des cas. PhyChro est donc très performant avec un taux d'erreur de seulement 3%.

Par ailleurs, il est intéressant d'observer que les bipartitions incorrectes ne correspondent pas à n'importe quelles branches. En effet, on observe une forte corrélation entre les bipartitions incorrectes et le fait que la branche soit courte, c'est-à-dire contenant peu d'évènements. Cela signifie que PhyChro se trompe quand il y a peu d'information disponible, ce qui est compréhensible.

Enfin, les simulations ont un grand intérêt pour évaluer un autre aspect de PhyChro qui est le calcul de scores de confiance. En effet, PhyChro attribue un score de confiance à chaque branche, qui indique la confiance que PhyChro a dans la branche reconstruite. On peut donc vérifier pour les branches incorrectes si PhyChro avait indiqué une confiance basse. Réciproquement, on espère que les branches auxquelles PhyChro donne un score de confiance élevé seront bien correctes. Grâce à notre simulation, nous pouvons vérifier ces propriétés, qui sont vraies toutes les deux. Cela permet donc de démontrer la pertinence des scores de confiance donnés par PhyChro.

1.3.2 R-CLAG : un paquet R de clustering

CLAG est un programme de classification automatique (clustering) présenté dans Dib and Carbone (2012a). Afin de rendre plus facile son utilisation, j'ai créé un paquet pour R (le langage de statistiques), qui a été inclus dans la bibliothèque officielle des paquets R (CRAN). Le paquet R est nommé simplement "CLAG", mais nous y ferons référence sous le nom de R-CLAG pour le différencier du logiciel original.

Le programme CLAG dispose de certains avantages le rendant adapté à l'analyse de données biologiques. En particulier, il a été originalement développé pour analyser les matrices de scores de coévolution, par exemple ceux produits par la méthode MST (Baussand and Carbone, 2009). Dans la méthode BIS (Dib and Carbone, 2012b), il forme une étape à part entière et est automatiquement utilisé.

Malheureusement, l'utilisation de CLAG n'est pas aussi facile que celle des autres méthodes classiques comme la classification hiérarchique, k-means ou les méthodes EM, pour lesquelles il suffit d'appeler une fonction fournie par le langage. C'est pourquoi, j'ai décidé de créer le paquet CLAG pour R. Il s'agit en fait simplement d'une interface, le calcul étant toujours effectué par l'implémentation originale de l'algorithme.

En plus de l'implémentation d'une interface, j'ai ajouté au paquet R la possibilité d'utiliser différentes méthodes de normalisation pour les variables. Ces méthodes sont utiles dans le cas où les différentes variables mesurent des quantités ayant une échelle très différente, voir ayant une unité différente.

Au final, R-CLAG a été inclus dans la bibliothèque officielle des paquets R (CRAN) après avoir passé les tests demandés, en particulier la documentation complète, la compatibilité avec les différents systèmes d'exploitation et des exemples rapides à exécuter.

Introduction

Evolution plays a key role in biology at very different scales, ranging from single nucleotides to the order of genes in genomes. During my PhD thesis, I have worked on several of these scales, and I present the corresponding work in different parts of the thesis.

In the first part, I show the work I have done on molecular evolution. First, I present the general biological background and the measures that allow us to detect both conservation and coevolution at the amino-acid level. Then, I present an application of these measures to the detection of critical residues in the cancer protein P53. To this end, I have made a benchmark of different prediction methods. I then use the same methodology on a large scale database of pathogenic mutations linked to genetic diseases. After that, I show how residue-level coevolution can help us discover protein-protein interactions in the hepatitis C virus. Finally, I present the PruneTree algorithm, which allows filtering sequence sets used as input for molecular coevolution detection methods.

In the second part, I have studied evolution at the genome level, in particular recombination mechanisms that occur during meiosis. I have looked at the recombination rates along the genomes and its primary cause, the double-strand breaks, but also at the density of other proteins involved in recombination. This part is divided in several chapters in which I present the biological background, a method based on Fourier transforms to analyze these genomic signals, and a model for the distribution along the genome of double-strand breaks and recombination proteins. I have also applied Fourier transforms to the analysis of the distribution of small RNAs binding to the *Phaeodactylum tricornotum* genome where a periodic pattern of 180nt is found in the small RNA distribution on methylated regions, possibly associated to nucleosome positioning.

Finally, in the last part, I present the other tools I have developed, which are linked particularly neither to molecular coevolution nor to meiotic recombination. I describe a novel algorithm that can simulate the evolution of genomes in order to benchmark the phylogenetic reconstruction algorithm PhyChro. Finally, I present the R-CLAG package that allows for easy use of the clustering algorithm CLAG.

Part I

Conservation and coevolution

Chapter 2

Coevolution methodology

Contents

In this chapter, we present the general questions we are trying to answer about mutations, and how evolution can help us predict their effect. We discuss both the information brought by conservation and coevolution.

2.1 Biological background

2.1.1 The genomics era

It is now well known that nucleic acids, DNA and RNA, are the molecules that all living organisms use to store their genetic information. The DNA molecule was first isolated by Friedrich Miescher in 1869 (Miescher-Rüsch, 1871), while the double-helix structure was

discovered in 1953 by James Watson and Francis Crick (Watson et al., 1953). This has led to a great interest in the sequencing and understanding of the information stored in DNA. In the last decades, enormous technological progress has been made in this field. DNA sequencing techniques have been massively improved and the cost of DNA sequencing has fallen exponentially. As a result, public databases now contain a huge set of DNA sequences as well as complete genomes.

The study of genomic information has enabled many applications. Human genomics have allowed biologists to better understand genetic diseases, cancer and human history. Animal, plant and microorganism genomics have enabled the creation of genetically engineered species and the improvement of selective breeding. And finally, sequencing pathogenic organisms (bacteria and viruses) has allowed us to understand them in order to fight them more efficiently.

2.1.2 Basics of molecular biology

The biologist can skip this section, in which we explain the basics of molecular biology as they are required to understand the next chapters. These fundamental facts can be found in molecular biology books such as Lodish et al. (2000).

Proteins Proteins are the molecules that constitute the building blocks of living organisms. They have a role both in forming the structure of organisms, as well as in performing chemical reactions (these latter proteins are called *enzymes*). A protein is formed by one or several chains of amino-acids. Amino-acids are small molecules that are present in all living organisms. There are 20 different "main" amino-acids (there are a few rare others in some organisms). These 20 amino-acids are conventionally denoted by letters (A for Alanine, V for valine, etc.) to make the display of an amino-acid sequence simple, as it can be displayed as a succession of letters. Amino-acids that form a chain are also called *residues* (short for amino-acid residues) because they lose their acid group when they bind to each other.

DNA The DNA molecule is a double-helix formed of two different strands facing each other. Each strand is a succession of (deoxyribo-)nucleotides. There are 4 types of nucleotides in DNA, which differ by the nucleobase they contain: guanine (G), adenine (A), thymine (T), or cytosine (C). The information in the genome is (mostly) stored as the succession of these four types of nucleotides. Sequencing a genome refers to reading the succession of A, G, T and C along the DNA. The other strand of DNA is a copy of the same information, with each nucleotide replaced by the complement base (G with C, A with T). For example, if one strand contains the sequence ACCT, the other contains TGGA. Since the information of one strand can be easily deduced from the other, only one strand (chosen arbitrarily) is stored in databases and is considered as the reference genome.

RNA The RNA molecule is similar to DNA, except for a few details. First, it often occurs in the form of a single-strand molecule. Then, it chemical composition is a little different compared to DNA, it is composed of ribonucleotides instead of deoxyribonucleotides. The difference lies in an extra oxygen atom in the case of ribonucleotides. And

finally, the thymine base (T) is replaced by the uracil base (U). The "four letters" of RNA are therefore A, C, G, and U.

Protein synthesis Protein synthesis occurs all the time in all cells of living organisms. Many proteins are always being synthesized at the same time, as a constant renewal of them is necessary to the life of cells. The first step in protein synthesis is to read a small part of the DNA sequence, called a *gene*, and create an RNA molecule with nucleotides having the same sequence as the DNA, except that T (thymine) is replaced with U (uracil). This RNA molecule, called a *messenger RNA*, is then itself read and the information it contains is used to create an amino-acid sequence. The translation between the information stored in the RNA into a sequence of amino-acids is done according to a rule, called the *genetic code*, in which three nucleotides (called a *codon*) correspond to an amino-acid. The sequence of amino-acids does not stay in the form of a long thread. Instead, physico-chemical interactions make it fold in a specific way, giving a specific shape to the protein. This shape is very important, as it is what allows proteins to play their role in the cell.

2.1.3 Natural selection

The process of evolution by natural selection was discovered independently by Charles Darwin and Alfred Russell Wallace in 1858 (Darwin and Wallace, 1858; Darwin, 1859). Natural selection is the consequence of two biological mechanisms.

First, there is a natural variation among individuals of the same species (animals or other living organisms). Individuals may differ in size, resistance, strength, color, etc. This variation is a random process. Thus, it does not necessarily result in a survival improvement for the individual. Some changes can be beneficial while others are neutral or deleterious. This variation has the consequence that some individuals will give birth to more offspring than others. We say that these individuals have a greater *fitness* than others.

The other mechanism that is essential to natural selection is heredity. Some of these characters that result in a higher fitness are hereditary, which means that their offspring is more likely to have them than other individuals. As a consequence, the next generation sees an increase in the proportion of individuals that have the hereditary characters that result in a high fitness. Generation after generation, this proportion increases until the valuable character is present in the whole population. The species has therefore *evolved* by means of natural selection. The combination of natural selection with heredity is called *evolution* and explains the birth of every character, protein or organ in every living organism.

Finally, we should say a few words about the process of speciation. For sexually reproducing organisms, a species is defined as a set of individuals that can successfully interbreed to produce fertile offspring. In other organisms, high DNA sequence similarity is used to define species. Speciation is the process by which a single species evolves to give birth to two species. For animals, this can happen for example when a group of individuals is physically separated from the rest, although other circumstances can lead to speciation. After a long time of independent evolution, the two groups become too different to be able to interbreed, and therefore, we have two species.

2.1.4 Homologous genes

The combination of the discovery of natural selection with genomics allows us to understand many things about proteins. When studying a protein-coding gene, we can often find a similar gene in the same species and in other species. We say the two genes are *homologous*. The are different types of homology, which we explain now (Koonin, 2005).

Since all living organisms have a common ancestor, it is not surprise that they have similar genes that were inherited from their common ancestor. These genes have been preserved for many generations but because of evolution, they are not exactly identical to the gene of the ancestor, but they remain very similar. This similarity often corresponds to a similarity of function, but it is not necessarily the case. This leads us to define the concept of *orthology*. We say that these genes are *orthologous* when their similarity is explained by the presence of the ancestral gene in the common ancestor.

We can also observe two similar genes in the same species. In this case, the explanation is gene duplication. Chromosomal rearrangements during meiosis can lead to some parts of the genome to be duplicated. There can also be a whole genome duplication, in which case the complete genome is duplicated. For each duplicated gene, evolution may either retain the two copies or delete on of them (or make it non-functional). If the two copies are retained, the they may evolve to get different functions in the organism. The result is that, many generation later, we can observe two similar genes in the same species. We say that these two genes are *paralogous*. In the special case where the duplication event was a whole genome duplication, we say the two genes are *ohnologous*.

2.2 The biological question

2.2.1 Predicting the effect of mutations

In protein synthesis, some steps can be easily predicted. The transcription from DNA to RNA is obvious, and can be done by a computer automatically. Then, some alternative splicing may occur, making the process a little more difficult to predict. Finally, the translation from an RNA to an amino-acid sequence is simple and can be done by a computer in which the genetic code has been recorded. A very difficult question, on the other hand, is to predict what the structure of the protein will be, knowing only the sequence of amino-acids. Even more difficult is to predict the function of the protein, or its interactions with other molecules.

But here, this is not the question we are interested in. We try to solve a simpler problem. Suppose that the function of the protein is known. Then, we observe a change in the sequence of nucleotides of the gene. Thanks to the genetic code, we can predict the resulting change in the amino-acid sequence. This may result in an identical amino-acid sequence, because of the genetic code redundancy. However, suppose that this results in a single change in the amino-acid sequence. Will the structure or the function of the protein be disrupted? It is well known that some changes in the DNA sequence have no effect at all, while others can lead to lethal diseases (for example muscular dystrophy, cystic fibrosis and phenylketonuria). But the effect depends on many factors, which are not trivial. Figure 2.1 shows a diagram summarizing the question.

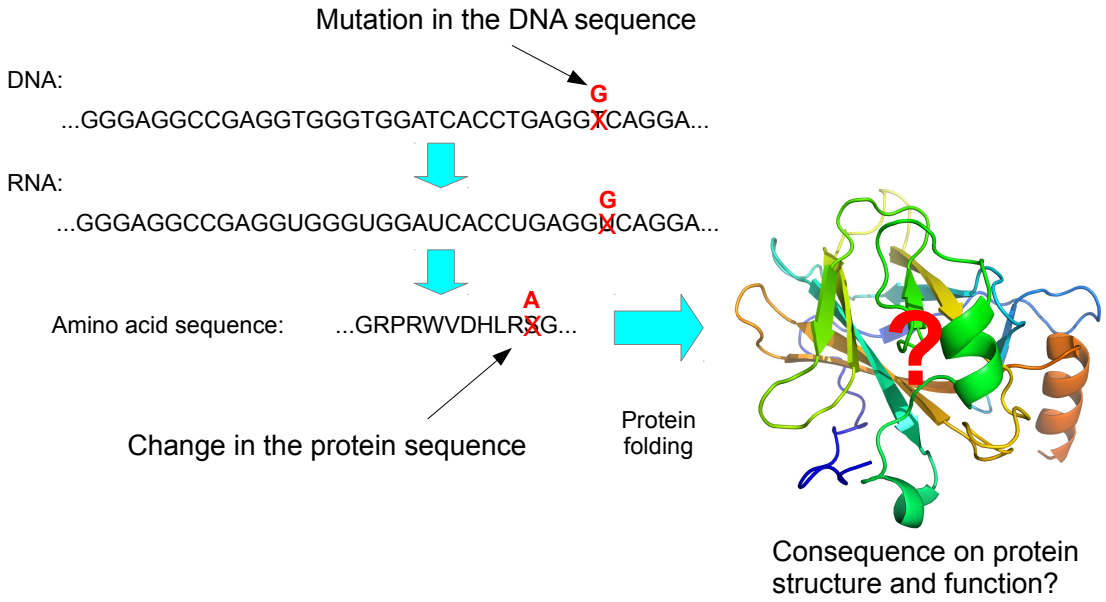

Figure 2.1: **The question of mutation effect prediction.**

2.2.2 What can evolution tell us?

Looking at orthologs (orthologous genes) can provide information about what is important in a protein. Some regions in a protein are more critical than others, and a mutation in them will have a more deleterious effect. When comparing orthologous genes, we can see that regions in the protein have a varying degree of variability. Some regions are highly variable and show a nearly random distribution of residues, while some others show a much smaller set of possible residues. In the extreme case, some residues in a protein are identical in all species. This is the result of variation and natural selection, which have allowed variations in some regions while preventing it in others. We call *conservation* the fact that natural selection has to some extent prevented variation in a region or at a position.

We can measure the conservation of a position by the fraction of studied species that have the same residue as the human sequence at this position. With this measure, a very variable position can get a value of 10%, while a very conserved position can get a value around 90%. Note however, that conservation does not necessarily involve a single possible residue. We may for instance have a position at which 40% of the sequences have a valine and the remaining 60% have an alanine. There are many different methods to measure conservation, which we shall discuss in the next chapter.

Conservation does not imply that mutations do not occur in these regions, but rather that individuals with a variation in conserved regions have a lower fitness than others, and therefore, this variation has a lower frequency in the population thanks to natural selection. The comparison of orthologous sequences tells us which mutations have been retained by natural selection. If we assume that the protein has a similar function in the different species, we can assume that the differences we observe correspond to mutations that do not affect negatively the protein function. We can therefore use a conservation measure as a prediction for mutation pathogenicity. This is a very good predictor, although it is not perfect. For example, will see that with protein P53 we have a 93% probability to distinguish an important position from an unimportant one thanks to a conservation measure.

Note that here we are not trying to simulate evolution by natural selection or to compute or measure fitness of different sequences. There are mathematical models of fitness and algorithms to simulate evolution within a defined fitness landscape, but in the next chapters we only use existing DNA sequences as a trace of past evolution, without trying to simulate the process generating them.

2.2.3 What is coevolution?

The general meaning of the word *coevolution* is the evolution of two different objects, each influenced by the other. These things can be amino-acids residues, proteins or even complete organisms.

For example, it is well-known that parasites evolve to better adapt to their host, by natural selection. Suppose that the parasite becomes so adapted that a huge part of the host population is infected. In this population, it happens that a few individuals have a genetic variation making them more resistant to the parasite. By the principle of natural selection, these individuals will have more offspring, as they will not die from the parasite. Generation after generation, the anti-parasite gene (or genes) will spread in the host population.

In turn, if a few parasites have a variation making them able to infect the "resistant hosts", they will get a selective advantage, and will spread. This process is repeated again and again, each species evolving to adapt to the evolution of the other. This is coevolution.

But coevolution can occur also at the molecular level. In living organisms, many proteins interact with each other. Such interactions require the two proteins to be adapted to each other, like a key is adapted to a lock. In this case, it is the fitness of the organisms hosting the two proteins that drives the evolution of both proteins.

Finally, we can look at an even smaller scale, which is coevolution between residues of the same protein. In a protein structure, some residues interact together by non-covalent bonds such as salt bridges or hydrogen bonds for example, playing a role in the final 3D structure. These interactions require compatible physico-chemical properties of the interacting residues. If one of them is replaced, the interaction is likely to be broken, or less stable. However, if the other interacting residue is also changed correctly, the new pair might form a correct binding. Therefore, a simultaneous change of both residues may be accepted by natural selection, or might even have a higher fitness. In fact, the two changes may not be really simultaneous, but rather sequential and separated by a short time (at the evolution scale). It might happen that the first change only decreases the fitness without being lethal, and later the second change restores a higher fitness.

This type of event leaves a specific signature on today's species genes. In Figure 2.2, we show an artificial example of this signature, with a perfect coevolution pattern. It is a perfect pattern because we see each possible residue in the first marked column to correspond to a specific residue in the second marked column (S with K, A with E, K with G and W with H). In general, coevolution can simply be a good correlation between residue types, so there may be sequences that do not follow the distribution.

In Figure 2.3, we have aligned amino-acid sequences from different strains of HCV (Hepatitis C Virus). Let us look at positions 3 and 14. We can see that in every strain in which we have an S at position 3, there is an A at position 14. On the other hand, when position 3 is a T, position 14 is an S. We never find S at 3 with S at 14 nor do we find T at 3 with A at 14. There is therefore a strong correlation between the types of amino acid at these two positions.

It is this kind of signal that we call coevolution. Here we may make the hypothesis that the residues at the two positions interact, and that this interaction is functional only with combinations S-A and T-S, but not with S-S and A-T. In fact, it may be that the residue at position 34, which also exhibits the same distribution, is in fact interacting with 3 or 14, so coevolution can give us a hint but we need other information to predict the correct physical interaction. Nevertheless, residue coevolution detection methods are aimed at detecting this type of signal, which is the first step to a better understanding.

2.3 Coevolution detection methods

In order to detect coevolution patterns, several methods have been developed:

- Mutual information (Shannon and Weaver, 1949)
- SCA (Statistical Coupling Analysis) by Lockless and Ranganathan (1999)
- ELSC (Explicit Likelihood of Subset Co-variation) by Dekker et al. (2004)

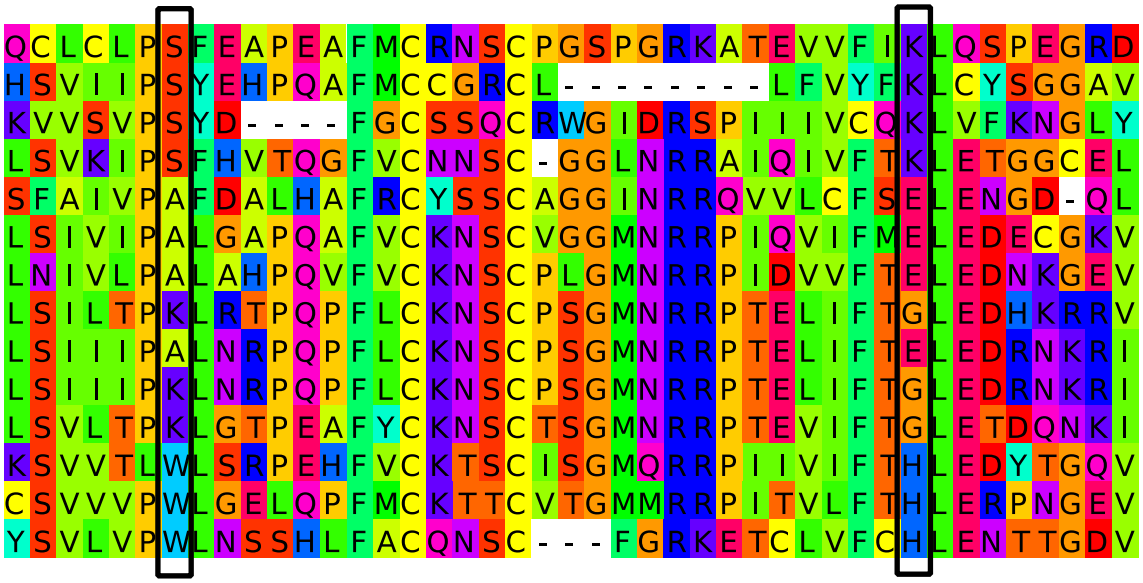

Figure 2.2: **An example of 2 positions exhibiting a coevolution pattern.**
These artificial sequences are an example showing that, in theory, coevolution can involve many different residues, that are not necessarily conserved.

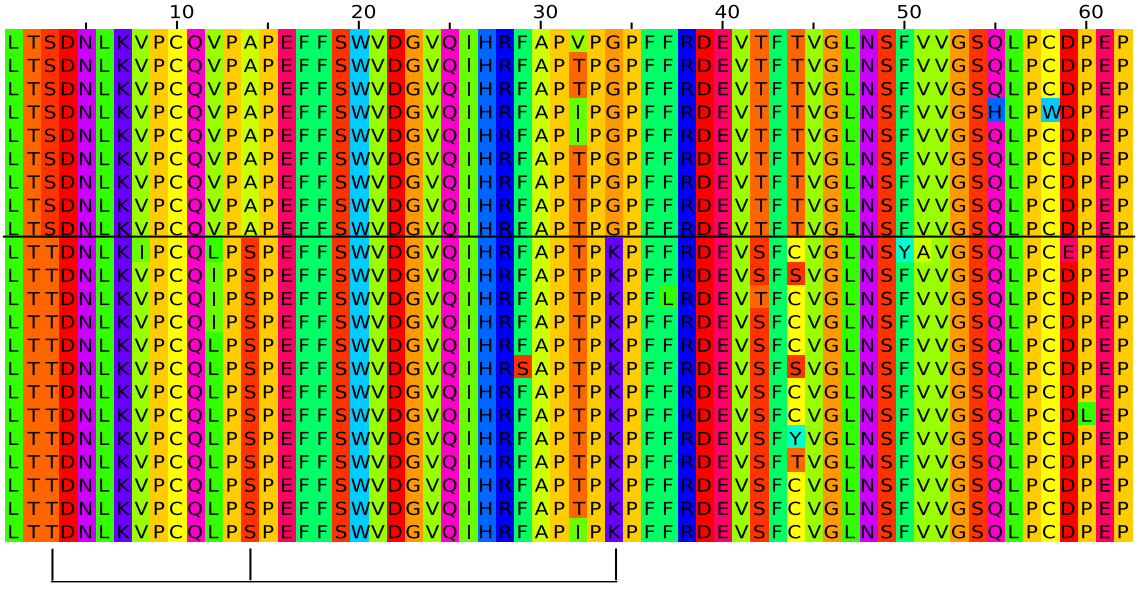

Figure 2.3: **An example of 3 positions exhibiting a coevolution pattern.**
Here we show a subset of a sequence alignment from HCV (Hepatitis C Virus) protein NS5A. Each row is the amino-acid sequence from a different strain of the virus. The three marked columns have a similar distribution of residues in sequences, and are a possible signature of residue coevolution.

- MST (Maximal SubTree) by Baussand and Carbone (2009)
- DCA (Direct Coupling Analysis) by Morcos et al. (2011)
- BIS (Blocks In Sequences) by Dib and Carbone (2012b)

All coevolution detection methods are not designed for the same types of sequence data. We may define two classes of methods. The first class is *statistical methods*, and includes mutual information, SCA, DCA and ELSC. They try to measure the statistical dependence between the residue at two positions, and require a high number of sequences with sufficient variation. The other class may be called *combinatorial* methods. They are based on counting individual sequences, and include MST and BIS. They are more adapted to fewer sequences with higher conservation.

2.3.1 Mutual information

Mutual information is a measure of dependence between two random variables (Shannon and Weaver, 1949). It is possible to use it to measure the covariation at two positions in a protein along homologous sequences. Here the two random variables are the two positions, while the observations are the residues at these positions in homologous sequences. Mutual information can therefore be used as an indicator of coevolution between two residues in a protein.

The mutual information of two discrete random variables X and Y is defined as:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p(x)\, p(y)} \right),$$

where $p(x,y)$ is the joint probability distribution function of X and Y, and $p(x)$ and $p(y)$ are the marginal probability distribution functions of X and Y respectively.

2.3.2 SCA and ELSC

The SCA method, standing for Statistical Coupling Analysis (Lockless and Ranganathan, 1999), is the first method specifically designed to detect residue coevolution. When detecting coevolution between two positions, SCA estimates the conditional probabilities to find each residue at a position, given each residue at the other position. To make this estimation reliable, many sequences are required. Typically, it is good to have at least 100 sequences. Moreover, these sequence have to be quite different from each other, as measured by the number of identical residues. If we have many nearly identical sequences, they will not bring new residues in the distribution, so they will not provide extra information, and will only bias the distribution, and hence they should be removed. This is why SCA requires at least around 100 sequences that are divergent enough.

The output of SCA is a square matrix, with as many rows and columns as residues in the protein. Each cell $M_{i,j}$ in the matrix is a score which measures the coevolution between the two positions.

ELSC, standing for Explicit Likelihood of Subset Co-variation (Dekker et al., 2004), is an alternative to SCA. It is very similar because it is aimed at discovering the same signal and also produces a matrix of coevolution coefficients. According to Dekker et al. (2004), it is more sensitive than SCA.

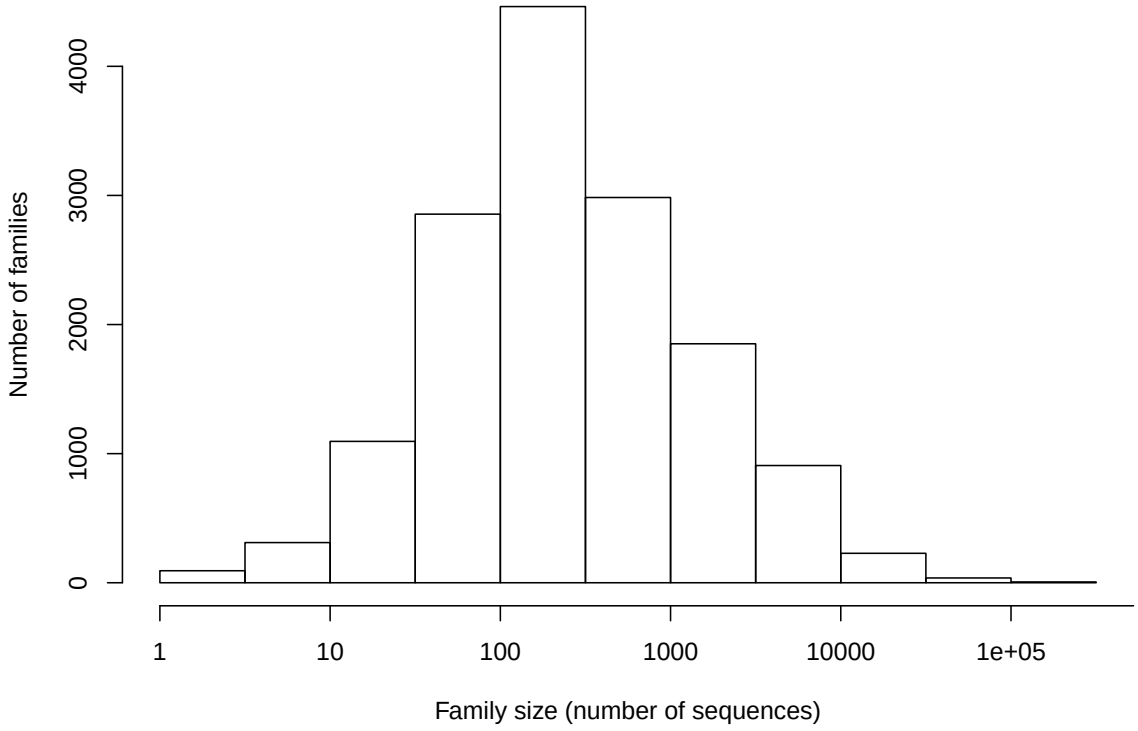

Figure 2.4: **PFAM 27.0 family size distribution, in logarithmic scale.**
Exactly identical sequences in families are counted only once.

2.3.3 DCA

DCA is a method aimed at predicting contacts in proteins. Its unique feature is its ability to recognize correlations that correspond to actual interactions from correlations that correspond to indirect correlations (hence the name of *Direct* Coupling Analysis). For example, if residues A interacts with B, and B interacts with C, we may also observe a correlation between A and C, although there is no interaction. This method have proven successful in predicting interactions (Morcos et al., 2011) but requires even more sequences than SCA, typically more than 1000. This can be realistic for bacteria sequences, but not for animal sequences.

2.3.4 MST

These limitations have led computer scientists to design methods that use individual sequence counting instead of statistics. The MST method, standing for Maximal SubTree, in particular is adapted to the characteristics of animal sequences, in terms of both sequence number of average similarity.

The MST method differs from the previous methods because it uses the phylogenetic tree associated to the sequences in the alignment. By carefully tracing the way residues evolved within the phylogenetic tree of sequences of a protein family, MST captures the transition along the time scale evolution of a conserved position to a coevolved position, and provides a numerical evaluation of the degree of coevolution of pairs of coevolved residues in a protein. It also detects conservation patterns using the same methodology, which is adapted to the detection of both signals.

The final result given by MST is a matrix, like with SCA, but it includes only rows and columns corresponding to positions, called *seeds*, for which MST considers that coevolution is worth detecting. This does not make a critical difference though, since we can translate the matrix given by MST into a full matrix with all positions, simply putting zeroes for scores of positions that MST chose to ignore.

The authors of MST also suggested that clustering the matrix would be useful for detecting clusters of coevolving positions, and used it to compute clusters that are biologically relevant (Baussand and Carbone, 2009). The authors provide a custom-made method, but it is also possible to use a general-purpose clustering method such as hierarchical clustering (Ward, 1963) or a specifically adapted clustering method such as CLAG (Dib and Carbone, 2012a).

2.3.5 BIS

The BIS method, for Blocks In Sequences (Dib and Carbone, 2012b), is similar to MST in that it uses a phylogenetic tree and a sequence alignment to detect conservation and coevolution. But its unique feature is the use of a block (or fragment) as its unit of coevolution. A block is a sequence of consecutive residues in the protein. It makes sense to use blocks because it interactions are often between fragments instead of single residues. However, blocks with a single position are allowed, so the method is able to detect single position coevolution even if searching for blocks. The BIS program also includes an option to disable the use of blocks and use only individual positions (but in this thesis we always use the block mode). For all pairs of blocks, BIS computes

a correlation score, which represents how close the correlated are the two residues in homologous sequences. The result can be presented as a square matrix, where rows and columns are the blocks, and the values are the correlation scores.

The BIS method was directly designed to be used with a clustering of blocks (or individual positions). In fact, the CLAG clustering method (Dib and Carbone, 2012a) is integrated in BIS, which allows to generate automatically a list of clusters. For example, the three residues marked in Figure 2.3 would be in the same cluster, since they have a common coevolution pattern. As a consequence of CLAG design, these clusters may not cover the complete sequence (since not all positions are conserved and coevolving), and may also overlap.

The clusters are provided with two scores, which allow us to get an idea of the cluster significance: symmetric score and environment score, both between -1 and 1 (but clusters with scores lower than 0 are discarded). The symmetric score measures the average rank of correlation coefficients between all blocks in the cluster. For example, the cluster in Figure 2.3 would get a score of 1 because the correlation is perfect. Environment score measures the similarity between rows in the matrix that correspond to blocks present in the cluster. It can be thought of as the equivalent of the distance used in other clustering methods, such as the distance matrix used as input in hierarchical clustering. Again, in the example of Figure 2.3, this score would be 1.

We also need to say a word about the parameters. BIS requires to specify a *dimension*, or number of exception d (integer). When looking at a column, sometimes there is only a single residue of a kind. For example, if in an alignment column (a fixed position in all homologous sequences) we have `IIIIILLLLLLEKKD`, then E and D are "exceptions" because they occur only once. Here there are two exceptions (E and D). The parameter d allows ignoring d exceptions. So if this column forms a coevolution pattern with another one on all sequences except the ones with the E and the D, it would still be detected, if the d parameter is 2.

There are also two modes in BIS, "d mode" and "d+ mode". The d+ mode allows making clusters that mix positions that comes from columns with a different number of exceptions. According to the authors of BIS, d+ mode is better so I always use it in the analyses presented in the next chapters.

Finally, the clustering part also involves a parameter Δ, which calibrates how different scores are allowed to cluster together. Its default value is 0.05 but other values are possible, like for example 0.1. In the next chapters I use only $\Delta = 0.05$.

2.4 Conclusion

The two main classes of existing coevolution detection methods are not adapted to the same sets of sequence data. Statistical methods generally require more sequences and more variation than combinatorial methods. Typically, statistical methods require at least 100 sequences with substantial variation between them. In this thesis, we work either on animal sequences, in the case of P53 and genetic diseases, and on virus sequences in the case of HCV. In both cases, the number of sufficiently divergent sequences is lower than 100. Moreover, many protein families are small, as can be seen in Figure 2.4, which shows a distribution of PFAM family sizes. As much as 29% of families contain less than 100 different sequences. So working with less than 100 is often necessary.

This number of sequences excludes DCA, but could be compatible with SCA. However, the sequence similarity we have is too high for SCA, which requires much more diverse sequences. This is true for our animal sequences, in which on average sequences have 60% of identical residues. In the case of HCV (hepatitis C virus), this number even reaches 80% (as can be observed when looking at Figure 2.3). If we delete sequences that are too close to each other, which lowers the average similarity, we end up with too few sequences for SCA, so the problem cannot be solved by simple filtering.

On the other hand, the MST method is adapted to the characteristics of our animal sequences, in terms of both sequence number and average similarity. The BIS method takes this step further as it is designed for even smaller sets that are even more conserved, exactly as our HCV sequences. These combinatorial methods are therefore more adapted to the characteristics of our data, which is why we decided to use them.

Chapter 3

Predicting critical positions of protein P53

Contents

The first application of coevolution we wanted to try was protein P53. The first reason is that P53 is of great medical interest because of its involvement in cancer. In tumor biopsies, mutations are often found in the P53 gene, in all types of cancer. Then, there is

a lot of data available on this protein, since cancer is a major research subject. Also, it has been observed that the mutations involved in different cancers are not the same. We therefore wondered whether we could detect the positions involved in each type of cancer as coevolving with together.

As we will see, our analysis highlights that the most important factors for making a successful prediction are the prediction method, the database quality and the divergence of homologous sequences.

3.1 Presentation of P53

The P53 protein is composed of 5 domains, totaling 393 amino-acids: a transactivation domain which interacts with the mdm2 protein, a proline-rich domain, a DNA-binding domain (Figure 3.1), an oligomerization and a carboxy-terminus domain. When running our analyses, we take in account the complete sequence, but the most interesting domain is the third one, ranging from positions 101 to 300. It is the longest, the most conserved and the most mutated in tumors (90% of tumor mutations are in this domain).

When a cell is healthy, the P53 gene is downregulated by several other genes, so its level is very low in normal cells. However, in case of stress, the gene is upregulated. Then, the P53 protein itself regulates other genes (Figure 3.1 shows its binding to DNA), with the final result being either cell cycle arrest or cell death. It is therefore clear that its role is very important to prevent cancer, since this gene will "decide" to kill the abnormal cell, or at least stop it from spreading and forming a tumor. Experiments have been made on mice which show that the absence of the P53 gene leads to a premature death from cancer (Donehower et al., 1992; Jacks et al., 1994).

However, errors occur in the process of DNA replication. Moreover, exposure to carcinogenic factors (radiations, some chemicals and pathogens, etc.) can increase the mutation rate. This can lead to the P53 gene being mutated and losing its function. If the same cell has other mutations that could lead it to form a tumor, P53 will not be able to initiate cell death or cell cycle arrest, and the cell will create a tumor. This is why tumor cell very often have mutations in the P53 gene, since this gene has to be disabled before the cell can form a tumor. So by looking at mutations in tumors, we can, a posteriori, discover which genes and more precisely which positions in these genes are critical to prevent cancer.

It has been discovered that P53 has two paralogous genes: P63 and P73. The duplications that gave birth to them dates back from the common ancestor of vertebrates. The three proteins have a different function, as proven by knockout experiments in mice. Without P63, mice have strong developmental defects (Yang et al., 1999; Mills et al., 1999), while mice without P73 have other defects, but do not show tumors (Yang et al., 2000).

3.2 Experimental data

We used the database available at `http://www.p53.fr` by Edlund et al. (2012). It is a collection of observed mutations in the P53 gene in tumors. It has been manually curated using statistical methods in order to remove the incorrect information. Figure 3.2 shows

the distribution of mutations in the database along the protein, with a total of around 23 115 observed mutations.

Figure 3.3 shows the mutation frequency along the protein for 3 different cancers. We can see for instance that colon and breast cancers have a similar distribution, with peaks at the same positions. On the other hand, liver cancer (green curve) has a strong peak at a very specific position (249) that is not a peak for the two other cancers. This figure shows only 3 types of cancers, but we have in fact 27 different cancers with at least 100 observations each. Other cancers have less than 100 observations, making the position-specific distribution unreliable. These other cancers represent a total of around 3000 observations, while the 27 "big" ones represent around 20 000 observations. The database is therefore quite big, considering the size of the protein (393 residues), and this allows us to have a reliable estimation of the relative mutation frequency of each position.

It should be reminded that this mutation frequency does not represent the inherent probability for a position to be mutated, but rather which mutations make the protein non-functional. This is because these observations are done in tumor cells, so the hypothesis here is that this mutation was probably the cause of the tumor. Cells with mutations at other positions did not create tumors and therefore are not observed here. However, there are probably a few fortuitous mutations that are not linked to the cancer, but they will have a very low frequency that can be clearly distinguished from critical positions.

3.3 Benchmarking

3.3.1 General methodology

Our goal is to be able to predict the set of positions which are critical for the function of the protein, by using information from homologous sequences. To do this we first need to define the set of positions which we consider as critical. I discuss the exact way to do this later (see section 3.4), but the general idea is to get those positions which correspond to the high peaks in Figure 3.2 above a threshold. The exact number of critical positions depends on the definition chosen, but it is around 40, which is about 10% of the residues (there are 393 residues in P53).

Then, I used a prediction method that I applied to every position in the protein, which gives scores to positions, measuring how much the method predicts the position to be important. For example, pairwise conservation can be used as a prediction method. This means that for each position in the protein, we count how many pairs of sequences have the same residue at the position. We divide it by the total number of pairs, which gives a number between 0 and 1. This ratio, which is computed for each position, is a possible prediction method.

We can now test whether critical positions defined from experimental data are given higher scores by the prediction method than others. Ideally, we would like all those positions to have a higher score than any non-critical position. For the conservation example, this means that if we define a conservation threshold, we would like most critical positions to be above this threshold, but we would not want non-critical positions to score above it. Figure 3.4 gives the actual results for this method.

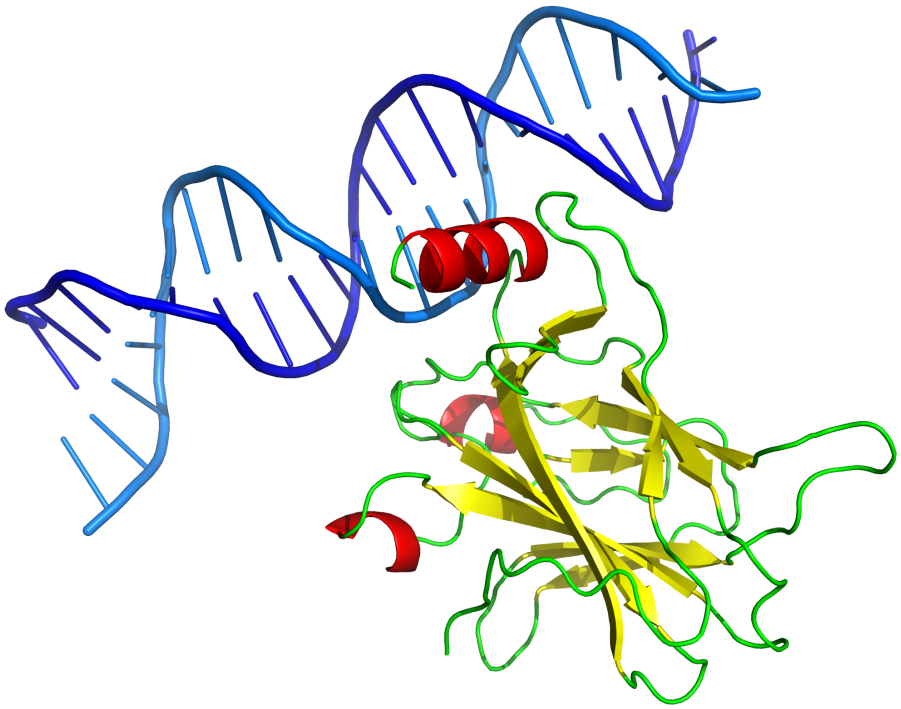

Figure 3.1: **P53 interacting with DNA** (Cho et al., 1994)

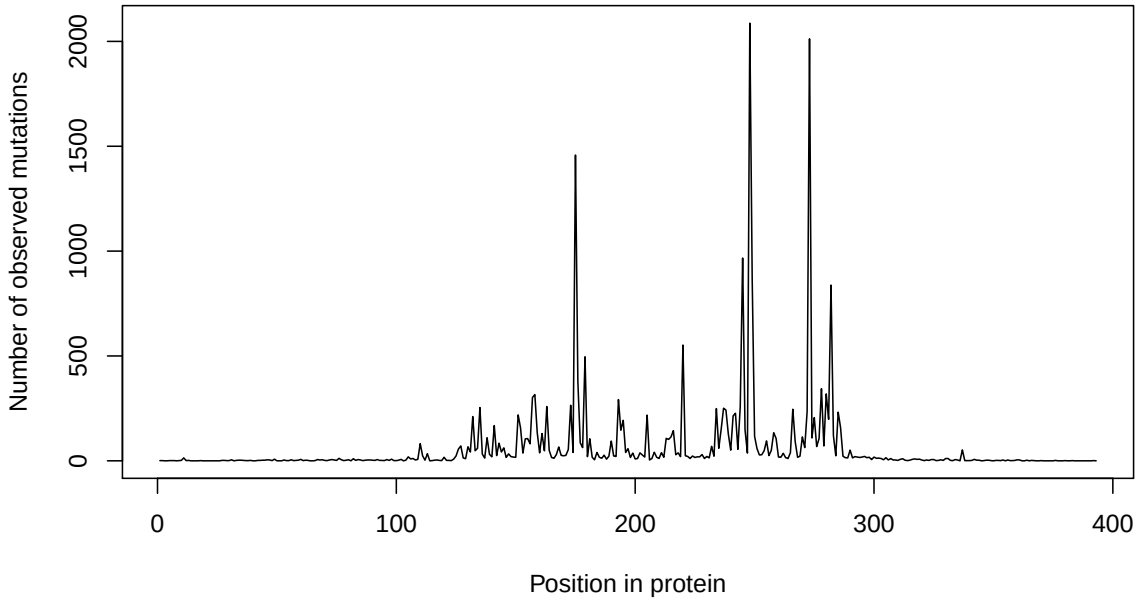

Figure 3.2: **Number of observed mutations in tumors along P53** in the Edlund et al. (2012) database. Residues from 101 to 300 form the DNA-binding domain.

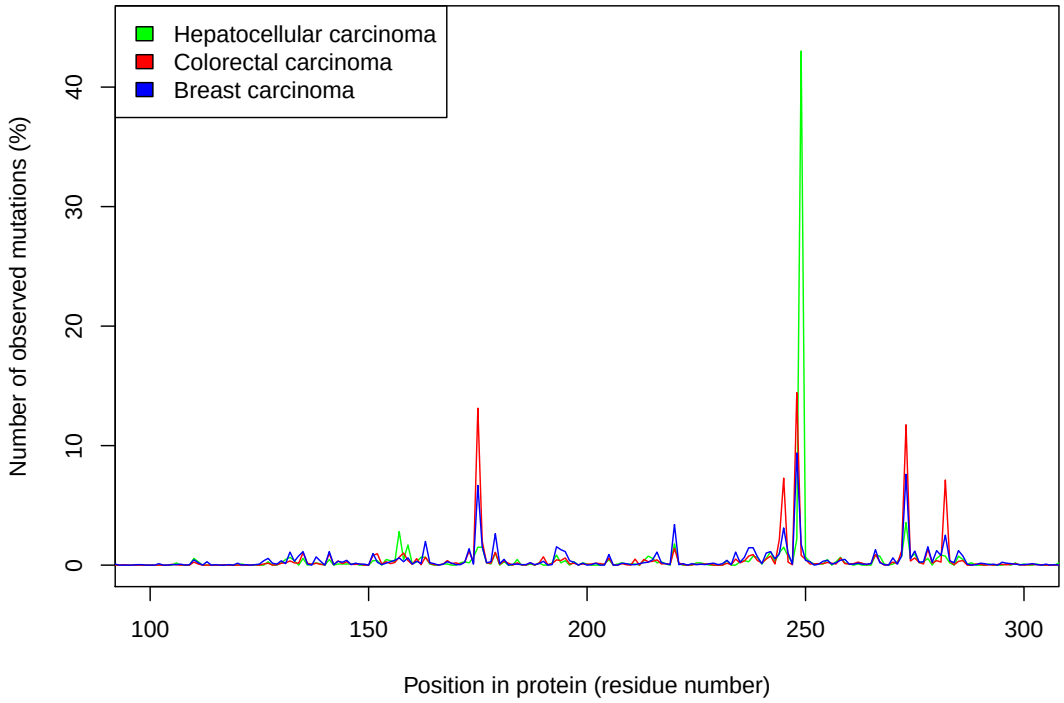

Figure 3.3: **Fraction of mutations observed at each position for 3 different cancers.** The protein has 393 residues, but only the range 100-300 is shown because the mutation frequency is low elsewhere. Data from Edlund et al. (2012) database.

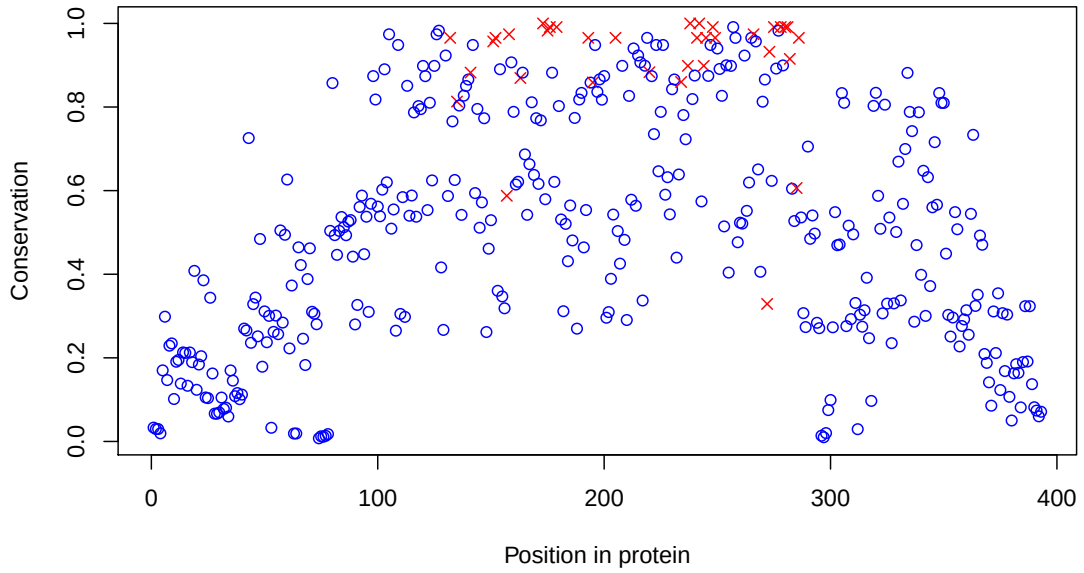

Figure 3.4: **Residue conservation (in different species) along the protein.** Red crosses mark the critical residues while blue circles mark the non-critical residues. As can be seen, critical residues tend to have a high conservation score. However, there are also many conserved residues which are not critical (blue circles at the top).

3.3.2 ROC curve

When a threshold is chosen, the set of positions is divided between those that are above the threshold and those that are below. We use standard methods for assessing the performance of predictions, by counting the number of true and false positives and negatives. Here a positive prediction is a position that has a score above (or equal to) the threshold, and a negative prediction is a position below the threshold. We are trying to predict the fact that the position belongs to the critical set, so a real positive is a position in the set, while a real negative is a position outside the set. For each position, we can compare the prediction with the truth, which gives four cases. If it is correctly predicted as positive, it is called a true positive. If it is wrongly predicted as positive, it is a false positive. If it is correctly predicted as negative, it is a true negative. If it is wrongly predicted as negative, it is a false negative. We can count how many positions fall in each of these four cases.

However, simply doing this would leave an arbitrary choice, which is the threshold to use. If we set it very high, predicted positions are likely to be correct, but we will not get many of them. Conversely, if it is set too low, we might get all critical positions, but also many false positives, making the prediction very unreliable. To solve this problem, a widely used method is to plot a ROC curve (Receiver Operating Characteristic). The principle is to try every possible threshold for the prediction method score, and measure the true and false positives rates, which are the fraction of correct predictions in respectively the sets of critical and non-critical positions. We can then plot the true positive rate against the false positive rate. Figure 3.5A shows an example of a ROC curve.

A widely used performance indicator is the area under this curve, often called "AUROC" (Area Under Receiver Operating Characteristic). This area has a particular mathematical property. If we consider a randomly chosen real positive position (inside the critical set) and a randomly chosen real negative position (outside the critical set), the AUROC is equal to the probability that the method gives a higher score to the real positive one. Ideally, we would like the AUROC to be 1. If the prediction method score was not correlated at all with the critical positions, we would get a value around 0.5 (as can be seen in Figure 3.5A for the "random" method).

3.3.3 PR curve

There is another way to assess the prediction quality, which is to use a Precision-Recall (PR) curve. This curve is based on the measure of Positive Predictive Value (PPV), which is the fraction of actually critical positions in the set of predicted critical positions. This gives us a measure of reliability of predictions. A PR curve is then a plot of PPV against the number of true positives. We may also divide the number of true positives by the total number of real positives, in which case it is called sensitivity, but it does not change the curve shapes (only the axis numbers). Here it is better to use a PR curve instead of a ROC curve because the ratio of critical positions to the number of total positions is small, as I will explain.

When going from the bottom left of the ROC curve to the top right, we decrease the threshold so that more and more positions are predicted as critical. However, there are actually 35 critical positions in Figure 3.5 while there are 358 non-critical positions. This means that when the rank threshold exceeds 35, there are necessarily false positives. For example, if we predict 75 positions, we get in the best case only $75 - 35 = 40$

false positives. This means that we have a false positive rate around 0.1. But this is not interesting, because we obviously do not want to predict that many positions. What we do want is to predict a few positions, and want to have most of them correct. The consequence is that in fact, in Figure 3.5A, all the part of the plot after $FPR > 0.1$ is useless. The only really interesting part is the beginning of the curve with $FPR < 0.1$, which is a small part of the plot on the left where we cannot really distinguish the curves. This also means that when computing the area under the curve, 90% of the information is the "bad one" with $FPR > 0.1$, making this indicator less interesting.

To give an intuitive idea of the problem with ROC curves, imagine a disease diagnosis test that would predict that half of the population has the disease, while in fact less than 1% of the people have it. However, if the half of what is predicted positive contains the 1%, this method would appear quite good on a ROC curve, although it is an extremely bad method. Instead, the interesting questions are: What proportion of people with the disease would the test detect? When the test is positive, what is the probability that it is correct? These two numbers are exactly what are shown on a PR curve. What is common between disease-testing and our question is that the number of actual positives is very small compared to the total number of things we test.

The same results are presented with a PR curve in Figure 3.5B. We can see more clearly the difference between the methods. The area under the PR curve (AUPR) is also shown. As can be seen, the best method according to the area under the ROC curve (phys. conservation) is not the same under the PR curve (conservationTree). In fact, there are two possible cases when comparing two curves (from different prediction methods) in either ROC or PR space:

- The curves cross both in the ROC and the PR space. The method with the highest AUROC is also the method with the highest AUPR.
- The curves cross neither in the PR nor in the ROC space. According to the area under the curve, the best method in one space may or may not be the best in the other space. The difference comes from the different weighting given to different parts of the curve in the two spaces.

The area under the PR curve is computed using the AUCCalculator program by Davis and Goadrich (2006).

3.4 Defining critical positions

3.4.1 Global mutation frequency

Now that we have this benchmarking methodology, we need to make sure to define critical positions in an appropriate way. The simplest way would be to count the number of observed mutations in tumors present in the database at every position, and define critical positions as those above a "high" value. For example, we can use $\mu + 0.5\sigma$ and get 35 positions as in Figure 3.6. However, this may seem like a naive idea, because it mixes different types of cancers, while we have seen that they have sometimes a very different distribution (Figure 3.3). Taking all cancers together is implicitly like summing the curves for different cancers, with a weight proportional to the number of observations in each

cancer, which is proportional to the amount of information that is present for each cancer in the database.

3.4.2 Cancer-specific mutation frequency

Another idea could then be to apply a threshold for each cancer (for example $\mu + 2\sigma$), and then take the union of all these positions. This idea seems more rational because it gives the same weight to every cancer, while the naive method above gives a higher weight to cancers for which there is more information available.

However, there are some issues with this method too. First, some cancers have very few recorded mutations, so it is not clear whether it is meaningful to consider the mutation frequency at positions for this cancer. The hypothesis behind the calculation of this distribution is that non-observed mutations are neutral while observed mutations are pathogenic. But if there is little data for a cancer, missing positions might just be missing information, and conversely a few false observations might pass the threshold since $\mu+2\sigma$ is then a smaller number.

To mitigate this issue, we can restrict the analysis to cancers which have at least 100 observations (they cover 85% of the database). In the worst case, i.e. for the cancer that passes this criterion with the smallest number of observations (acute myeloid leukemia, with 106 observations), the threshold $\mu + 2\sigma$ correspond to having at least 3 observations of a mutation.

To avoid loosing the remaining information of the database (the remaining 15% of mutations), we take all the remaining cancers together and consider the union of all the associated mutations. We consider them as a single cancer, labeled "others", and apply the threshold again. With this method, the number of mutations is high enough for a correct estimation of the distribution.

At the end of this process we get a union of 46 positions. If we do all this with $\mu + 3\sigma$ instead, we get 33 positions. They are not a subset of the 35 positions given by the naive method, as 7 of them are not in the set of 35.

To decide which of these data sets to consider as the reference for critical positions, we decided to look at how well we can predict them with conservation methods. If we use simple conservation for instance, we get an AUPR[1] of 0.671. When using the cancer-specific threshold, we get AUPR=0.621 with $\mu + 2\sigma$, and AUPR=0.490 with $\mu + 3\sigma$. A similar change is observed on all prediction methods.

Note that the reasoning here is the reverse of the benchmark, because I choose the reference experimental data, so as to maximize the quality of predictions when compared to it. So it seems that in fact the naive method for choosing the critical residues is the best one. We cannot completely rule out the hypothesis that all methods are bad, and that by doing this choice we are just making the task simpler for prediction methods. However, it seems more likely, since this effect is true for all prediction methods, that the choice of a cancer-specific threshold used as a reference introduces noise. This is why we finally decided to use the 35 residues of the naive method, shown in Figure 3.6.

[1] Area Under Precision-Recall curve, see section 3.3.3

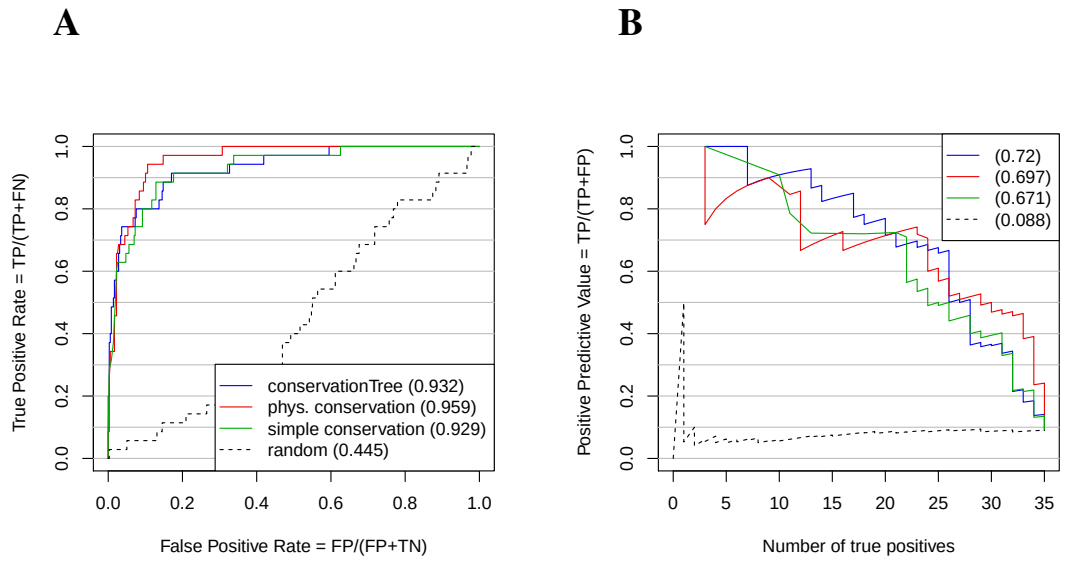

Figure 3.5: **ROC (A) and PR curves (B) for several prediction methods.**
The numbers in parentheses are the area under the curves. The method titled "random" is simply a random raking of all positions.

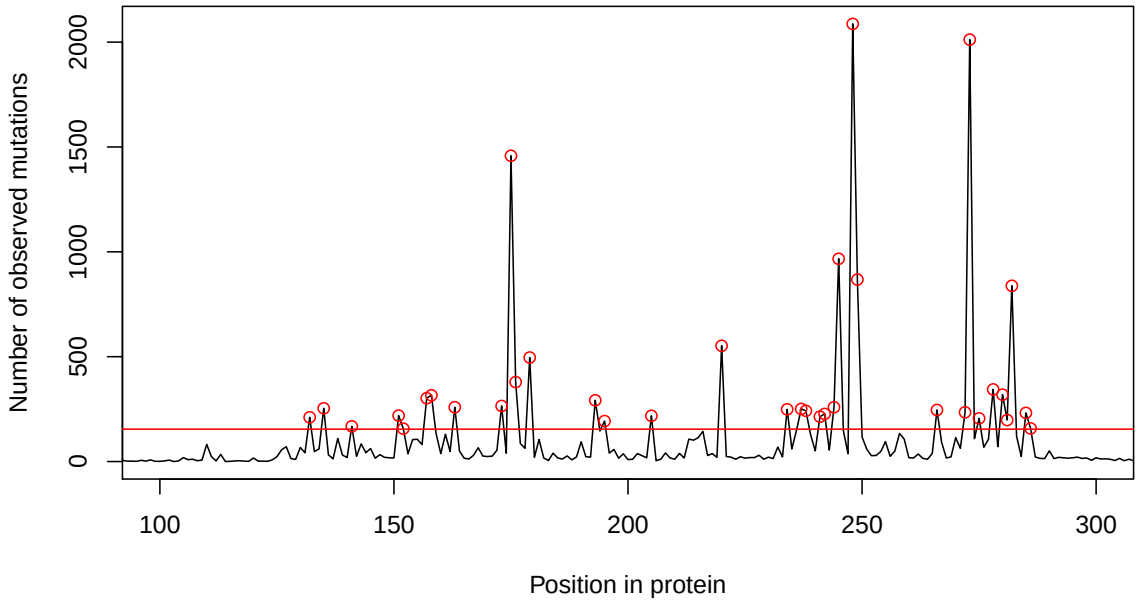

Figure 3.6: **Number of observed mutations along P53.**
The red line marks the threshold we have chosen: $\mu + 0.5\sigma$. The 35 positions that pass the threshold are circled in red.

3.5 Comparison of methods

I tested different prediction methods and different sequence selection criteria. Some of these criteria, especially database and identity threshold, give very different predictions. Therefore I have tested various choices, which resulted in several sequence sets. I have then run the benchmark on all combinations of prediction method and sequence set.

However, here I present only results based on the best sequence set, and I leave the discussion about sequence choice for the next section. In the section, I assume to know a sequence alignment and to use it for all sequence-based prediction methods. For each method, we give both the AUPR and AUROC, which are two performance measures presented in section 3.3.1.

3.5.1 Methods based on conservation

Simple conservation with the reference (AUPR = 0.670 ; AUROC = 0.930)

The simplest way to predict critical positions is to measure residue conservation in all homologs. The simplest choice is to count, for a given position, how many homologous sequences have the same residue as the reference one. Here the reference is the human P53 wild-type sequence.

Simple pairwise conservation (AUPR = 0.671 ; AUROC = 0.929)

There is a variant, which we can call "simple pairwise conservation", which counts the number of pairs of sequences that have the same residue at the tested position. It means that instead of using a reference sequence, each sequence is given the same importance. It might make a difference in cases where the human is the "exception" compared to other species, because it that case, conservation with the reference would be low, while this pairwise conservation would give a high score. However, this case is rare, especially because species close to human have been more studied than others, and therefore the sequence set contains many species close to human. For example, chimpanzee P53 is exactly identical to the human one. The consequence is that human is hardly ever an exception. The result is that both the AUPR and AUROC of this method are very similar to those of the previous method. Simple pairwise conservation method is the green line in Figure 3.5B.

Shannon entropy (AUPR = 0.685 ; AUROC = 0.942)

An alternative for measuring conservation is to use the Shannon entropy (Shannon, 1948). In information theory, entropy is a measure of the uncertainty in a random variable. In this context, the term usually refers to the Shannon entropy, which quantifies the expected value of the information contained in a message. It is defined as (for a given position):

$$-\sum_{i=1}^{20} \frac{n_i}{N} \cdot log_{20} \frac{n_i}{N}$$

where n_i is the number of sequences with residue i (we suppose we number the different type of residues from 1 to 20), and N is the total number of sequences.

The idea is that if entropy is high, we expect the position to allow a lot of variation, while if the entropy is low, we expect it to be critical. When running the benchmark, we get better results than simple conservation.

Physico-chemical conservation (AUPR = 0.697 ; AUROC = 0.959)

A possible improvement on the simple conservation methods is to take in account physico-chemical properties. In the set of critical positions, we see for example that position 272 is given a rank of 259 (ranks go from 1 to 393). In fact, at this position, there are three main residues (V, A, and G), each being present in around one third of sequences. So it might seem that this position is not very conserved. However, these 3 residues have similar physico-chemical properties, so in fact this position is very important for the protein, but there are three valid choices for the residue. But it is still a critical position, because in tumors it is present, but mutated into other residues than V, A and G.

One way to improve prediction performance for this kind of positions is to take into account physico-chemical properties when computing conservation. Instead of counting how many pairs of residues are identical in homologous sequences, we weight them by a number which quantifies how similar the two residues are to each other. Instead of summing zeros and ones, we sum these numbers.

To define similarity between residues, we adapt the idea by Livingstone and Barton (1993). In their work, the authors proposed to define a distance between residues as the number of physico-chemical properties by which they differ. They are defined as 10 binary variables (yes or no). The classification of residues by properties is shown in Figure 3.7.

I adapted this distance into a similarity measure between 0 and 1, by simply counting the number of identical (instead of different) physico-chemical properties, and dividing by the total number of properties. I kept the original idea by Livingstone and Barton (1993) to have the gap considered as an amino-acid with all properties set to yes.

The two conservation methods explained in section 3.5.1 can be adapted with this similarity matrix by using these numbers instead of 0 for different or 1 for identical. When we adapt the reference-based conservation, we get an AUPR 0.680 and AUROC of 0.947. If instead we adapt pairwise conservation (which was a little better), we get an AUPR of 0.697 and an AUROC of 0.959 (red line in Figure 3.5). So this last method is better than simple conservation and entropy.

Conservation weighted by BLOSUM62 (AUPR = 0.669 ; AUROC = 0.945)

We have presented in the previous section a conservation measure that takes in account physico-chemical properties, which brings a significant improvement. In this method, we have defined a residue similarity matrix. However, many programs use residue similarity matrices, like for example BLAST (Altschul et al., 1990) which uses them to measure sequence similarity. The most widely used matrix is BLOSUM62 (Henikoff and Henikoff, 1992), and is generally a good choice for various sequence comparisons.

So we create a prediction method like in the previous section, but using the BLOSUM62 matrix instead of our physico-chemical similarity matrix. For a given position in the protein, let $b_{i,j}$ be the value in the BLOSUM62 matrix for residues found in sequences

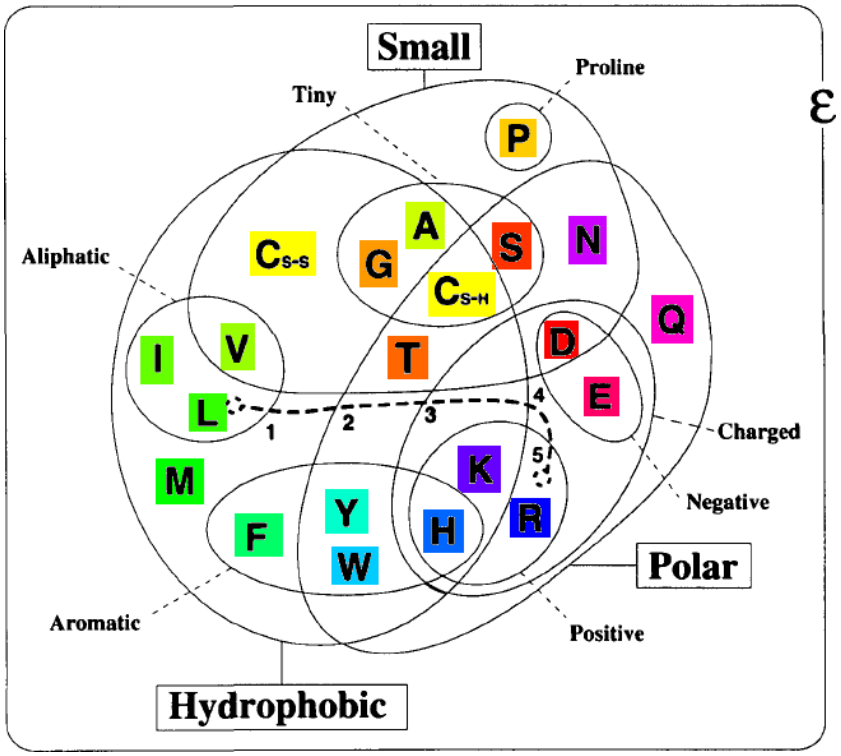

Figure 3.7: **Residue physico-chemical properties.**
This classification can be used to define a distance between residues as the number of lines to cross. The line from L to R shows an example of distance: it crosses 5 lines so the distance is 5. The original black and white figure was made by Livingstone and Barton (1993). Residues are colored as proposed by Taylor (1997).

i and j. We define the score of the position as:

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \frac{b_{ij}}{b_{ii}}$$

When running the benchmark, this method scores lower than simple conservation in AUPR, but higher in AUROC. However, in both cases, it is beaten by physico-chemical conservation as defined in the previous section.

SCA 5.0 conservation measure (AUPR = 0.027 ; AUROC = 0.932)

The SCA software (Lockless and Ranganathan, 1999) is designed to detect coevolution (see section 2.3.2). However, it also provides a measure of conservation. The particular feature of this measure its its inclusion of residue rarity as an weighting factor. This rarity is measured on a complete database of proteins in order to measure the average residue frequency in living organisms.

This rarity factor can make a difference between residues that are conserved in all sequences, because any other conservation method would score them with the highest score, while SCA conservation measure will make a distinction and mark as more interesting the conservation of a rare residue.

According to our benchmark, this measure is a little better than simple conservation in AUROC, but worse in AUPR. In both cases, physico-chemical conservation and conservationTree perform better.

3.5.2 Methods based on conservation and phylogenetic trees

Reconstructing a phylogenetic tree from a set of sequences is a well-known operation in bioinformatics. Famous methods for doing it are UPGMA (Sokal, 1958) and Neighbor-Joining (Saitou and Nei, 1987). These methods are both based on distances. They take a distance matrix as their input, which contains all the distances between the pairs of sequences. This distance aims at reflecting the passed time since a sequence pair and their latest common ancestor. It may be measured by the identity between sequences (percentage of identical residues), or by a more sophisticated measure, like the similarity weighted by BLOSUM62 for instance.

The UPGMA method performs a hierarchical clustering of sequences (Ward, 1963), which produces a tree. The implicit hypothesis between UPGMA is a constant rate of evolution in all branches of the tree. This hypothesis may be true for small sets of species (for example the HCV sequences in chapter 5), but it is incorrect for the sequences we use here. Neighbor-Joining (NJ) is better because it does not make this hypothesis.

An alternative to these two methods is PhyML, which is based on maximum likelihood estimation. This method takes advantage of the sequence data and uses a model of sequence evolution to estimate the most likely tree according to the observed sequences. This is much more accurate because it can generate a tree that is consistent with different changes in the sequence (mutations, insertions or deletions) and attributes them to inferred common ancestors. A distance-based method, on the other hand, can only use a single number to compare two sequences, which is much more limited. The main disadvantage of PhyML over Neighbor-Joining is its low speed, but it is not a concern here, as we work

with small sets of sequences (less than 500). I therefore used PhyML 3.0 (Guindon et al., 2010) to infer the phylogenetic tree, which is shown in Figure 3.8.

This tree shows a nice decomposition in 4 groups. The P53 gene was duplicated twice in the ancestor of vertebrates, therefore invertebrates have a single P53 homolog (group 4 in Figure 3.8), while vertebrates have an ortholog for each of the 3 genes: P53, P63 and P73. This explains why, in Figure 3.8, we can see 3 vertebrates groups (1, 2 and 3). Each of them contains the tree of vertebrates, because each of the three genes has a complete independent evolution in vertebrates. Note, however, that some vertebrate species have lost one or two of the three genes, so the three subtrees (1, 2, 3) do not necessarily contain exactly the same species.

Several questions arise from this particular tree shape: Are the three paralogs interesting, or should we just use P53 orthologs? Do the invertebrates bring useful information to our analysis, or are they too far to be relevant for the human P53? These questions are discussed in section 3.6, where we will show that it is actually useful to take all the sequences we have here. So here I keep all the sequences, and extract as much useful information as possible from them.

When looking at a given position, we can look at the distribution of different residues in the tree. For some of them, we see a very interesting pattern, which is that the dominant residue is different in the 4 groups defined in Figure 3.8. For position 201 (in the human sequence) for example, each of the three vertebrate groups have a different dominant residue. The P53 sequences have a leucine (L), the P63 sequences have a serine (S), and P73 sequences have an asparagine (N). In invertebrates, there are many variants. This residue is therefore well conserved among each paralog in vertebrates but not across the 3 paralogs, which leads to the hypothesis that this residue plays an important and specific role in each paralog of the p53 family.

If on the other hand, we had these 3 residues but distributed randomly among sequences in the tree, this would mean that the specific nature of the residue is not critical, and that leucine, asparagine and serine are freely substitutable at this position. The distribution of the nature of a residue among the tree is therefore an important information that is not contained in a simple frequency of each type of residue.

ConservationTree (AUPR = 0.720 ; AUROC = 0.931)

This reasoning led me to think that it could be important to distinguish these two cases when measuring conservation, giving a greater weight to residue conservation when comparing sequences that are close in the tree. This is why I created a method, "conservationTree", which consists in measuring conservation between the $N(N-1)/2$ pairs of sequences at each position (like simple conservation of section 3.5.1), but weights each pair by the global similarity of the two sequences.

Let us call r_{ip} the residue in sequence i and position p. It may be a gap if the residue at position p in the human sequence (reference) has no corresponding position in sequence i. We define the conservationTree score of position p as:

$$s_p = \frac{2}{N(N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \left(\mathbf{1}(r_{ip} = r_{jp}) \cdot w_{ij} \right)$$

where $\mathbf{1}(condition)$ is 1 if *condition* is true, and 0 otherwise. The w_{ij} coefficient is a

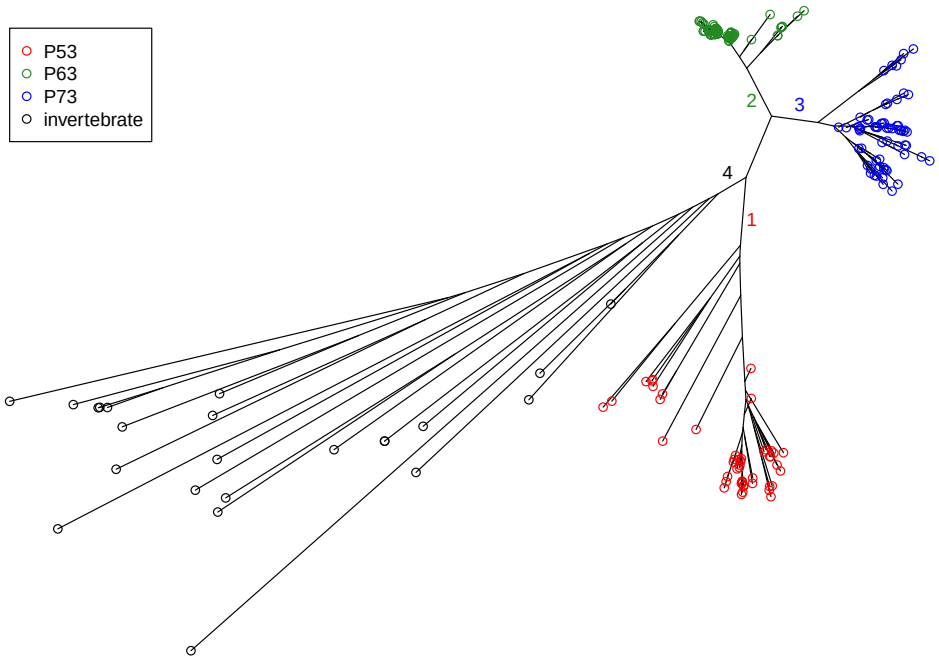

Figure 3.8: **Reconstructed tree from P53 homologous sequences.**
Leaves have been colored according to whether the corresponding sequence is annotated as P53, P63, or P73 when the species is a vertebrate. In invertebrates, the P53 gene is not duplicated, therefore annotating sequences as P53, P63 or P73 does not make sense, so they are left in black.
Vertebrate species present in the tree are: *Homo sapiens, Pan troglodytes, Pan paniscus, Gorilla gorilla gorilla, Pongo abelii, Papio anubis, Macaca mulatta, Callithrix jacchus, Saimiri boliviensis boliviensis, Nomascus leucogenys, Oryctolagus cuniculus, Otolemur garnettii, Ailuropoda melanoleuca, Cavia porcellus, Sus scrofa, Equus caballus, Cricetulus griseus, Canis lupus familiaris, Mus musculus, Bos taurus, Ovis aries, Rattus norvegicus, Felis catus, Loxodonta africana, Sarcophilus harrisii, Oncorhynchus mykiss, Gallus gallus, Danio rerio, Ictalurus punctatus, Takifugu rubripes, Oreochromis niloticus, Oryzias latipes, Meleagris gallopavo, Taeniopygia guttata, Ailuropoda melanoleuca, Ornithorhynchus anatinus, Xenopus laevis, Xenopus (Silurana) tropicalis, Anolis carolinensis.*
Invertebrate species present in the tree are: *Branchiostoma floridae, Saccoglossus kowalevskii, Ciona intestinalis, Trichoplax adhaerens, Metaseiulus occidentalis, Tribolium castaneum, Nematostella vectensis, Acyrthosiphon pisum, Pediculus humanus corporis, Ixodes scapularis, Megachile rotundata, Apis florea, Apis mellifera, Nasonia vitripennis, Hydra magnipapillata, Drosophila grimshawi.*

number between 0 and 1 which represent the proximity of sequences i and j in the tree. Note that if w_{ij} were always 1, this definition would be exactly identical to simple (pairwise) conservation defined in section 3.5.1, so this method is in fact a weighted version of conservation.

To define w_{ij}, we consider the distance d_{ij} between sequences i and j in the tree, and calculate w_{ij} like this:

$$w_{ij} = 1 - \frac{\text{rank}(d_{ij}) - 1}{N^2 - 1}$$

The rank is defined on all pairs of d_{ij}. If several values have the same rank, their ranks are replaced by their average original rank.

When measuring AUPR, conservationTree performed better than any other tested method (see Figure 3.5). It AUROC on the other hand is better than simple conservation, but worse than physico-chemical conservation.

Physico-chemical conservationTree (AUPR = 0.704 ; AUROC = 0.955)

Since both physico-chemical conservation and conservationTree showed an improvement over conservation, I wondered if using the two would improve even more the prediction accuracy. The idea is simply to weight the physico-chemical conservation coefficient with the same weights as in conservationTree. However, this method gives disappointing results. It is not better than both conservationTree and physico-chemical conservation (without the tree). This is true both for AUPR and AUROC.

Weighted conservation (AUPR = 0.688 ; AUROC = 0.931)

After the good results obtained with conservationTree, it is tempting to test a simplified weighting, which, instead of using the tree distance, would simply use direct sequence similarity, while retaining the same idea. The advantage is that it does not require inferring a tree, so it is simpler and faster to compute. It can be thought of as an approximated variant of conservationTree.

We define the weighted conservation of position p as:

$$s_p = \frac{2}{N(N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \left(\mathbf{1}(r_{ip} = r_{jp}) \times w_{ij} \right)$$

where the weight w_{ij} is the similarity between sequences i and j defined as:

$$w_{ij} = \frac{1}{L} \sum_{q=1}^{L} \mathbf{1}(r_{ip} = r_{jp} \text{ and } r_{ip} \neq gap)$$

This is basically the fraction of identical residues; with the only exception that when both sequences contain a gap, they are considered different (this is necessary to avoid giving a higher weight to shorter sequences).

These bennchmark results are both lower than conservationTree, so the added complexity of conservationTree is worth it, and we cannot simplify the weighting to a simple fraction of identical residues. This can be explained by the better distance in the tree computed by PhyML, which is much more sophisticated than a simple count of identical residues.

3.5.3 Methods based on coevolution

Coevolution methods use the type of signal we have described in section 2.2.3. Instead of detecting columns which have conservation patterns (one or a few residues in a column), we detect pairs of columns which have co-variation patterns. These co-variation patterns are understood as a signature of coevolution. The hypothesis behind the use of these methods is that a coevolution pattern between positions i and j means that there is an interaction between i and j. This interaction may be either direct (physical contact) or indirect, in which case there is a chain of direct interactions that link i to j. In both cases, this means that positions i and j are probably important and can therefore be predicted as critical positions.

Here we have tested SCA5 (Lockless and Ranganathan, 1999), ELSC (Dekker et al., 2004), MST (Baussand and Carbone, 2009), and BIS (Dib and Carbone, 2012b). Some methods, like SCA and ELSC, detect only coevolution patterns, while methods like MST and BIS detect both conservation and coevolution patterns. See section 2.3 for a description of these methods.

SCA 5.0 (AUPR = 0.064 ; AUROC = 0.355)

The SCA program gives a symmetric square matrix with as many rows (and columns) as positions in the sequence (see section 2.3.2 for details). Each value M_{ij} in the matrix is a score that measures the presence of a coevolution pattern between columns i and j. Here we need a score for every position to run our benchmark, so we need to translate this $N \times N$ matrix in a single vector of N numbers. We do it by taking the maximum of each row.

Let us explain why. Suppose we look at row i. Taking the maximum of row i means that we take the maximum of M_{ij} on all possible columns j. This means that we take the position j for which the strongest coevolution pattern with i is found. The logic is that, if there is a strong coevolution pattern between two positions i and j, both i and j should get a high score. If there is a weaker coevolution pattern between i and k, then k should get the weak score M_{ik}, but i should retain the high score that comes from M_{ij}.

As part of the general benchmark methodology explained earlier, a threshold is then used to predict critical positions (above the threshold), and the threshold is varied to produce the PR and ROC curves. When considering a threshold λ, this is equivalent to saying that we consider all cells M_{ij} in the matrix where $M_{ij} \geq \lambda$ and consider all the rows and columns of these cells as the predicted critical positions. This means that our "maximum rule" is equivalent to taking all positions involved in a coevolution pattern with a score of at least λ.

When testing SCA in our benchmark using this methodology, the results are very poor. They are even worse than random (which correspond to AUROC=0.5). This means that the presence of a coevolution pattern is negatively correlated with the importance of the position in cancer. We can explain this by the fact that most critical positions are very conserved, so they cannot form coevolution patterns.

ELSC (AUPR = 0.050 ; AUROC = 0.113)

ELSC can be tested with the same methodology as SCA since it provides its results in the exact same form (a matrix). The results are similarly bad for the same reason as SCA.

MST (AUPR = 0.556 ; AUROC = 0.927)

The MST method has the advantage of being able to detect both conservation and co-evolution patterns (see section 2.3.4 for a description of this method). It could avoid the problem we get with SCA and ELSC in the previous section, which is the inability of pure coevolution methods to detect very conserved residues. However, this method can only work better than conservation measures if there are coevolving position in the critical positions set.

MST requires a phylogenetic tree as an extra input. I used the same tree as for conservationTree, that is a tree inferred by PhyML 3.0. MST gives a matrix of coevolution scores, to which we apply the maximum as explained in the previous sections for SCA and ELSC.

The AUPR is worse than even simple conservation, but the AUROC is very similar. This means that MST does detect conservation patterns correctly, but does not bring extra information compared to a simple conservation analysis, and is worse than physico-chemical conservation and conservationTree.

BIS (AUPR = 0.649 ; AUROC = 0.854)

The BIS method detects pairs of coevolving fragments, as explained in section 2.3.5. The list of detected fragments is then automatically given to the CLAG clustering method (Dib and Carbone, 2012a), which clusters them together. The final result is a list of clusters, each of these clustering corresponding to a list of fragments showing a common coevolution pattern.

We need to translate this list of clusters into something we can benchmark. In the benchmark, what matters is the rank of scores of positions (although there can be ties). We therefore need to translate the list of clusters into a ranking of positions. To do this, I use the scores given by BIS for the clusters and rank the clusters by symmetric score (highest score comes first). If two clusters have the same symmetric score, I use the environment score to decide which to put first. If both scores are equal, the clusters are ranked the same. I then translate this ranking of clusters in a ranking of positions in a simple way. Each position gets the rank we gave to the cluster. If a position is in several clusters (which is possible), it gets the best rank in the rank of clusters to which it belongs. Finally, all positions that do not belong to any cluster are left at the end of the ranking (with an identical rank).

I have chosen to rank by symmetric score first, and to only use environment score to break ties, because it turns out that this gives better results in the benchmark. I also tried to do the reverse (environment score as the major criterion), and to use the sum of the two scores, but these give worse predictions.

For the d parameter (see section 2.3.5), I tested different values from 1 to 8 and ran the benchmark on all. There is also the CLAG clustering parameter Δ to choose. I took $\Delta = 5\%$ as recommended by the authors.

The maximum AUPR and AUROC are both reached for $d = 5$. This is much better than ELSC and SCA, but still lower than even simple conservation. Again, it seems that coevolution does not add extra information compared to conservation for this problem.

3.5.4 Methods based on the prediction of mutation effect

Here I present a set of methods that are designed to answer the following question: If residue x at position p is mutated into residue y, should we expect the protein function to be disrupted? This question is a little different from the one we are trying to answer, but as I will explain, we can still adapt them to our problem.

PolyPhen 2 (AUPR = 0.342 ; AUROC = 0.883)

PolyPhen (Adzhubei et al., 2010) is a program aimed at predicting whether a mutation is pathogenic. It takes as its input the UniProt id of a human protein, a position p and the residue in which you change the residue at position p. It outputs both a probability that measure how sure it thinks that the change is damaging (0 is neutral for sure, 1 is pathogenic for sure), and a class that is one of the three between "neutral", "possibly damaging" and "probably damaging".

Its answer may be different depending on the replacement residue. For instance, replacing a leucine by an isoleucine might be acceptable which replacing it by a cysteine might break the protein structure. Our benchmark is based on predicting positions, not position-residue pairs, so we need to "translate" our problem into the question PolyPhen is designed to handle.

When testing a position p, there are 19 different residues that one can put instead of the wild type one. So I tested all 19. Then, we need to derive a score which we can benchmark. There are several possibilities that we can imagine:

- average the 19 probabilities
- take the minimum of the 19 probabilities
- take the maximum of the 19 probabilities
- count the number of "probably damaging"
- count the number of "probably damaging" or "possibly damaging"

On both AUPR and AUROC, the best of these choices is to average the 19 probabilities. However, these values are lower than what we get with simple conservation.

FATHMM (AUPR = 0.406 ; AUROC = 0.911)

I also tested the FATHMM method created by Shihab et al. (2013). This method has a special model for cancer mutations, so is specially designed to solve the question we are interested in. Like Polyphen, it requires the specification of the replacement amino acid. Its output is a score, which is higher when the method thinks the mutation is damaging.

As with PolyPhen, I tested all 19 residues for every position. Then, we can again choose to take the minimum, average or maximum of these 19 scores. The best is to

take the maximum (both for AUPR and AUROC), although the average produces almost exactly the same results. According to my benchmark, This is better than PolyPhen, but still worse than simple conservation.

FoldX 3 (AUPR = 0.360 ; AUROC = 0.866)

Here we have talked about predicting the effect of mutations from sequence information. However, it is also possible to predict the effect of mutations from protein 3D structure. To test a few of these methods, I used the SPROUTS web server (Lonquety et al., 2009), which allows to automatically run many methods on a protein. I used the PDB structure 2FEJ.

As with PolyPhen and FATHMM, we need to specify the replacement residue. All methods give a score, called $\Delta\Delta G$, which correspond to a variation in energy caused by the mutation. Since we have 19 scores for each position, we can again take the minimum, average or maximum. The tested methods are FoldX 3 (Schymkowitz et al., 2005), I-Mutant2 (Capriotti et al., 2005) and I-Mutant3 (Capriotti et al., 2008) with sequence only or both sequence and structure, and MUpro (Cheng et al., 2006).

After testing all these methods with minimum, maximum and average, we find that FoldX3 is the best prediction method with the maximum, both in PR and ROC spaces. It has AUPR=0.360 and AUROC=0.866. This is comparable to the results given by PolyPhen-2, but worse than even simple conservation.

3.5.5 Conclusion on prediction methods

When measuring AUPR, conservationTree performed better than any other tested method, with an AUPR of 0.720 (see Figure 3.5). When measuring AUROC, physico-chemical conservation is the best method with 0.959. However, as we have argued earlier, AUPR is a better performance indicator for this specific problem, so the conclusion here is that conservationTree is the best method for predicting P53 critical positions. It is followed by physico-chemical conservation with an AUPR of 0.697.

In particular, we see that the structure-based methods available in SPROUTS are all worse than even simple conservation, which is the most naive method based on sequence homology. We may explain this observation in two different ways. First, it might be that methods like FoldX, I-Mutant, MUpro, and PolyPhen are optimized to distinguish between different replacement residues, as they all require its specification. Here we are focusing on only *position* prediction, which is not directly what those methods are designed to predict. So it could be argued that we do not benchmark the best ability of those methods.

Second, this observation might mean that, in fact, with a sufficient number of sequences, evolution tells everything about viability of the mutated P53 protein. More precisely, homologous sequences show us every viable change in the protein (changes that do not lead to tumor formation), so it allows us to know which mutations are pathogenic based only on homologous sequences. It means that natural selection tells us what is possible or not. On the other hand, with structure-based methods, we do not know automatically whether a mutation will be pathogenic or not. We can predict if it will destabilize the structure, but even if it is not the case, it might break the interface and make the protein useless. Natural selection on the other hand will remove this variant, which explains

that we do not see it in our list of homologous sequences. Sequence-based prediction can therefore capture both structure and function disruption, and allows us to make good predictions.

Finally, it does not seem that coevolution pattern detection brings any improvement in the detection of critical positions. Overall, most of these positions are conserved, and those which are not do not show more coevolution than non-critical positions.

3.6 Sequence alignment preparation for the benchmark

In the previous section, I have assessed only one piece of the prediction pipeline: the prediction method. Although this is the most interesting part to benchmark, there are other choices that need to be made which we have not discussed so far. I assumed that we had selected "correctly" a set of homologous sequences. The precise list of operations I do are:

1. Use BLAST (Altschul et al., 1990) on the RefSeq database (Pruitt et al., 2007) to find homologous sequences. The query is the human P53 sequence (`P53_HUMAN` in Swiss-Prot).

2. Retain only sequences with at least 20% of identity (number of identical residues) with the reference, and 60% coverage (fraction of the query that is mapped to the sequence found). Sequences with an E-value above 0.01 are also discarded, however, these sequences never achieve both 20% identity and 60% coverage anyway.

3. Align the sequences with MAFFT (Katoh et al., 2002) to produce a first alignment.

4. Infer the phylogenetic tree with PhyML (Guindon et al., 2010). Store the tree file.

5. Remove all columns with a gap in the human P53 sequence (we are not interested in them, as we work on the human sequence). This gives us the final sequence alignment.

All these steps produce two files: a sequence alignment and a tree. The alignment is the input used by the sequence-based prediction methods (all methods except PolyPhen and FoldX). Some methods like conservationTree, MST and BIS also use the tree.

3.6.1 Query coverage, alignment and tree inference

The choice of 60% allows the homologous sequences to cover a reasonable part of the query. Finally, most sequences have a coverage ratio higher than 80%. Increasing this threshold to 70% would remove less than 2% of our sequences. In fact, the main reason for this threshold is to avoid small fragments that are sometimes present in databases.

I chose to use the MAFFT aligner (Katoh et al., 2002), which is very fast. Alternative include Muscle (Edgar, 2004), which is slower but more accurate and PRANK, which is even slower but more accurate (Loytynoja and Goldman, 2010).

In particular, PRANK is known for distinguishing better insertions and deletions from substitutions, while traditional methods tend to align together whatever happens to be in front of the residues in homologous sequences. However, here we are not very interested

in insertions and deletions, because our methods rely only on substitution information. Typically, the difference between PRANK and other methods appears only in the least conserved parts of the protein, which in the case of P53 are residues before 100 and after 300. Some homologous sequences may lack this parts, and instead have a completely difference fragment. Mafft and Muscle would force them to map together, resulting in columns with a lot of different residues, while PRANK would infer (correctly) a deletion and substitution, resulting in gaps in the columns. Whether we have many gaps of much variation, conservation method would give a very low score. So in the end, this choice is not very important. To be sure, I tested predictions based on alignment created from PRANK, but it did not improve accuracy.

A reason for using Muscle is its ability to align very different sequences. However, P53 is easy to align. This is general to alignments between proteins from animal species because animals evolve slower than, for instance, bacteria or bacteriophages, and are therefore more conserved. Again, I also tested Muscle, but the results were comparable to those obtained with MAFFT.

We can also choose different methods for tree inference. As discussed in section 3.5.2, PhyML is better than other methods. However, I was interested in quantifying this choice impact. Therefore I tested all tree-based prediction methods (of section 3.5.2) with a tree made by Neighbor-Joining. The results are not significantly different. If a randomly regenerated (incorrect) tree is used instead, the results are less accurate. This shows that inferring a correct tree is important, but that both PhyML and Neighbor-Joining give a good tree for the purpose of our predictions, because the sequences are quite conserved and therefore easy to align.

3.6.2 Sequence database

A more complicated choice is the database, because in that case, it really influences the results. I tested 3 possible choices: NCBI nr (Sayers et al., 2011), NBCI RefSeq (Pruitt et al., 2007) and UniProt (Consortium, 2014).

The NCBI “nr” (non-redundant) database is the default search database for BLAST. It contains as much protein sequences as possible, in particular, it contains all CDS translations from GenBank, all PDB sequences and all Swiss-Prot sequences. RefSeq on the other hand is a database of manually curated reference sequences. They come only from non-mutated reference CDS. Finally, UniProt has a similar aim as nr. It is composed of two sections, Swiss-Prot, which is manually annotated, and TrEMBL, which is automatically annotated.

When using BLAST and our identity and coverage thresholds, we find 230 sequences in RefSeq, 436 sequences in nr and 333 sequence in UniProt. With each of them, we benchmark all methods, and take the best method on each. I decided to proceed like this because it avoid the possible problem that some methods might be better for different data. These 3 tests are reported in Table 3.1 and colored in green and yellow. As can be seen, the best results are obtained with RefSeq, both in AUPR and AUROC.

This can easily be explained by the fact that our methodology makes the hypothesis that all sequences are wild-type sequences for functional proteins. This is verified in the case of RefSeq sequences, but not necessarily in the case of nr or UniProt. For example, nr includes PDB sequences, which often have mutations that are required to make a crys-

Analysis	N	API	Method with best AUPR	Best AUPR	Method with best AUROC	Best AUROC
refseq – 20% identity	230	60 %	conservationTree	0.720	phys. conservation	0.959
refseq – 30% identity	224	62 %	conservationTree	0.687	phys. conservation	0.958
refseq – 40% identity	209	68 %	SCA5 conservation	0.539	phys. conservationTree	0.939
refseq – 50% identity	49	70 %	SCA5 conservation	0.455	phys. conservation	0.903
refseq – 60% identity	38	83 %	SCA5 conservation	0.307	SCA5 conservation	0.867
refseq – 20% – orthologs only	103	51 %	phys. conservationTree	0.666	phys. conservation	0.957
nr – 20% identity	436	54 %	Shannon entropy	0.681	phys. conservation	0.954
nr – 30% identity	428	55 %	phys. conservation	0.667	phys. conservation	0.953
nr – 40% identity	384	60 %	SCA5 conservation	0.585	phys. conservation	0.930
nr – 50% identity	158	72 %	SCA5 conservation	0.455	phys. conservation	0.907
nr – 60% identity	130	83 %	BIS d=4	0.332	SCA5 conservation	0.869
UniProt – 20% identity	333	49 %	phys. conservation	0.641	phys. conservation	0.952
UniProt – 30% identity	328	50 %	SCA5 conservation	0.616	phys. conservation	0.945
UniProt – 40% identity	281	55 %	SCA5 conservation	0.589	SCA5 conservation	0.923
UniProt – 50% identity	131	67 %	SCA5 conservation	0.500	SCA5 conservation	0.908
UniProt – 60% identity	100	78 %	BLOSUM62	0.323	BLOSUM62	0.868
SPROUTS on 2FEJ			max ΔΔG FoldX3	0.360	max ΔΔG FoldX3	0.866
PolyPhen 2			Polyphen average prob.	0.342	Polyphen average prob.	0.883
FATHMM			max FATHMM	0.406	max FATHMM	0.911

Table 3.1: **Summary of all benchmarks made on P53 sorted by database.**
The general maxima are highlighted in green. The maxima for nr and UniProt are highlighted in yellow. N is the number of sequences. API is the average percentage of identity between all pairs of sequences.

tallographic structure. This highlights the need to use high-quality verified sequences for our analyses.

Interestingly, it can be noted by looking at Table 3.1 that the best method choice is not always the same for the three databases. Shannon entropy outperforms conservationTree in nr, while physico-chemical conservation is the best for UniProt. But overall, RefSeq with conservationTree is the best combination.

3.6.3 Sequence identity

Another interesting question is the percentage of identity to require. When looking at the results from BLAST, we remove sequences that have too few identical residues with our reference sequence. On one hand, getting lower identity sequences gives more information, but on the other hand, the protein might become too different for this information to be relevant.

Here I tested different identity threshold, from 20% to 60% (see Table 3.1). As can be seen, on all three databases, and in both AUPR and AUROC, the 20% threshold choice outperforms everything else. Moreover, there is a clear inverse correlation between identity threshold and prediction performance. The conclusion is that these lower identity sequences bring useful information.

When looking at the sequences manually, this effect can be easily explained. When retaining only high identity sequences (like with 50% identity), many positions are 100% identical in all sequences. Using conservation to predict critical residues would give them the highest score possible. But perfectly conserved positions are more numerous than the critical positions we are searching for. We therefore get many false positives in our prediction. When adding lower identity sequences (going from 50% to 40% for example), the new sequences are not necessarily identical at all previously perfectly conserved residues. But other residues remain identical. The interesting observation is that critical positions keep being conserved. As can be seen in Table 3.1, each 10% decrease brings an improvement. Finally, positions which remain conserved, even when we have included low-identity sequences (20%), are really the critical ones.

When looking at the actual species in these sets, we can see that the 60% identity set contains only mammals. The 50% identity set extends to other vertebrates (mostly birds and fishes). The 40% identity contains also mostly vertebrates, but adds the orthologs of P63 and P73, instead of being limited to P53 orthologs. Finally, the 20% and 30% set add mollusks and insects.

Again, we can observe that the best method is not always the same when identity is increased (see Table 3.1). In high-identity sets, SCA conservation is often better. This is because in these sets, many positions are 100% conserved, which all get the same score (the maximum possible score) in conservation, physico-chemical conservation, or conservationTree. But SCA conservation adds extra information which is the residue rarity. It can therefore rank these positions differently, which other methods consider them equal. However, it is still better to use more sequences than to rely on this information, since SCA conservation is behind other methods in low-identity sets (which are overall better).

3.6.4 Homology or orthology?

In all these analyses we have considered both orthologs and paralogs, this means that in fact we have orthologs for P53, P63 and P73. However, we could restrict our analysis to only orthologs of P53. An approximate way to do this is to take only the closest sequence to the reference for each species. The hypothesis is that a P53 protein from a given species is more similar to the human P53 than the P63 or P73 of this species. This is mostly true, with a few exceptions (18), that arise because in some species there is no P53 but only a P63. Finally, this reduced set of sequences (103 instead of 230) produces lower quality predictions, with only AUPR=0.666 and AUROC=0.957.

3.7 Conclusion

We can summarize the discoveries we made with all our benchmarks:

- Residue conservation in homologs is a good predictor of position importance in tumors.
- Taking in account physico-chemical properties of residues or their distribution in the gene tree improves the prediction quality.
- Coevolution patterns detection does not improve prediction accuracy.
- Purely sequence-based methods perform better than structure-based ones.
- Lower identity sequences add relevant information.
- Taking in account paralogs instead of only orthologs is better.
- The manually curated RefSeq database is the best choice.
- The choices of aligner and tree inference program have a minor impact.

To conclude, the P53 benchmark shows which parameters are important for the success of predictions and which are not. It allows us to know how to proceed for a large-scale analysis on many proteins, which we detail in the next chapter.

Chapter 4

Predicting pathogenic mutations

Contents

With all the knowledge we have gathered on P53, we decided to run the same type of analysis at a large scale on many proteins. We wanted to test whether the methodology developed for P53 can be applied to many proteins involved in (human) genetic diseases. Here we used the Varibench database, which is a database of both pathogenic and neutral protein mutations.

4.1 Methodology

4.1.1 Varibench

The Varibench database (Nair and Vihinen, 2013) contains 23,683 neutral mutations and 19,335 pathogenic mutations. However, for many proteins, we do not have both known neutral mutations and pathogenic mutations. If we want to test the ability of a prediction method to distinguish between pathogenic and neutral mutations in a protein, we need at least one known pathogenic and one neutral mutation for this protein. This is why I decided to restrict the analysis to proteins for which at least one neutral mutation and one pathogenic mutation are known. There are 482 proteins that have this property.

We remove two of them, the first because it has too few homologous sequences (less than 5) for conservation analysis to make sense. The other one we remove is P53, because we want this analysis to be independent of the previous one, and also because P53 has so much information (around one third of the dataset), that we would lead to a major bias of the results. Other proteins, typically, have a number of recorded mutations lower than 100. In the end, we have 480 proteins with a total of 7968 recorded mutations.

4.1.2 Gathering homologous sequences

To gather homologous sequences for our 480 proteins, we apply the same process as with P53. Thanks to our benchmark on P53, we now know that using BLAST on RefSeq with a 20% identity threshold is the best option. However, we cannot proceed exactly as with P53. Some human proteins are well conserved in all living organisms, and bacteria sometimes have the protein with more than 20% of identical residues with the human protein. But there are many bacterial sequences (several thousands) compared to animal sequences (typically around a hundred), which makes the analysis too long to be realistic.

To make computation faster, I decided, for the beginning, to limit the number of sequences by applying two rules. First, we take only animal sequences. Second, we allow only for one sequence per species for each protein. Note that this second rule goes against what we concluded for P53 i.e. that restricting the analysis to orthologs decreases prediction accuracy. In fact, this observation does not seem to apply to this analysis and seems only valid for P53, as we will see later.

For each protein, we therefore have a set of sequences, each corresponding to a different animal. The good side of this is that when we will measure conservation, we will really be counting species, not just sequences. This is different from what we did with P53 because for some species, several isoforms are present in the RefSeq database. This gives a higher weight to these species, as they appear several times in the alignment. Here on the other hand, each species counts for exactly one. However, we need to be careful

that sometimes, an animal species will have a sequence that is exactly identical to another. For example, with P53, the chimpanzee sequence is exactly identical to the human one, so it was not present in the alignment (because RefSeq contains only unique entries). Since we are now really counting species, we should take them into account. So unlike with the P53 analysis, we may have duplicated sequences (corresponding to different species).

4.1.3 Alignment and tree reconstruction

As with P53, we use MAFFT to align the sequences, because it is the fastest method, and we have seen that it does not significantly change the results. For the tree reconstruction (required for methods that need also a tree as input), I chose to use Neighbor-Joining (Saitou and Nei, 1987), because it is much faster than PhyML. Here, we have to run the analysis on 480 proteins, not just one like with P53, so we need to optimize speed.

4.1.4 Testing by position and residue

When testing PolyPhen on P53, we made the hypothesis that it was not successful because it was designed to take position-residue pairs that give both the position to mutate and the residue to put instead. In our P53 benchmark, we did not look at the replacement residue. This time, we want to use this information. Moreover, Varibench sometimes provides both a pathogenic and a neutral replacement residue *for the same position*. So here the elements to predict are position-residue pairs. The question one is asking to the method is for example "What happens if I replace the residue at position 87 by an alanine?". The two input are position 87 and residue alanine.

The advantage is that we do not need to do the artificial task of summarizing the information on 19 residues into a single number. So this benchmarking approach is fair to methods like PolyPhen. On the contrary, methods like simple conservation do not care which residue we put, since they measure something that applies to the position. These methods would give an identical prediction, for a given position, to a pathogenic replacement residue and to a neutral one.

4.2 Individual benchmark

There are two choices for running the benchmark. The first is simply to do the same computations as for P53 but for each of the 480 proteins, each one producing a PR and a ROC curve, and then to find a way to summarize this information. The other is to mix all positions and scores and create a single ROC curve. In this section, we present the first solution, which we call individual benchmark. The second solution is explained in the next section.

The methodology for this benchmark is exactly the same as for P53. We therefore get 480 PR and ROC curves. Unlike P53, we do not have the problem that the number of pathogenic observations has a smaller order of magnitude compared to the number of neutral ones. In fact, there are more pathogenic observations (6738) than neutral ones (1230) in our data set. Here the role played by the two types is symmetric. Here we benchmark the ability of the method to distinguish pathogenic mutations from neutral ones inside each protein.

To summarize the results for each method, we take the AUROC on each protein and plot their distribution (Figure 4.1), and also compute their average.

4.2.1 Conservation

We tested all the methods we presented for P53. Many of these methods do no take the extra information we have here which is the replacement amino acid. These methods will therefore never be able to distinguish a neutral and a pathogenic mutation at the same positions if the replacement amino acid differs.

Purely position-based methods give good AUROC. Surprisingly, the ranking differs from the P53 benchmark. Here, simple conservation (AUROC=0.794) performs better than physico-chemical conservation (AUROC=0.766) and similarly to conservationTree (AUROC=0.790) or weighted conservation (AUROC=0.793). In the case of conservationTree, the presence of different subtrees for different paralogs is not relevant any more, since we do not consider paralogs here. Therefore the rationale for this method is not valid any more. Weighted conservation AUROC can be explained by the same reason. Considering simple conservation is a simpler method, it appears as the best choice here.

4.2.2 PolyPhen 2

PolyPhen is based on many different types of information to make its prediction. It uses conservation information in homologous sequences (like our conservation methods), solvent accessibility of the residue in the 3D structure (if available), annotations on the sequence in UniProt (binding, active site, lipid and metal), and secondary structure annotation.

PolyPhen gathers its homologous sequences with a process similar to ours, with different parameters and database choices. PolyPhen performs a BLAST search against UniRef100 (while we use RefSeq). Sequences with an identity higher than 30% and lower than 94% are retained (while we use 20% as a minimum and no maximum). The required coverage is 75 residues (while we use 60% of the reference sequence length).

With these sequences, PolyPhen does not simply measure conservation. Instead, it uses a method called PSIC (Sunyaev et al., 1999), for Position-Specific Independent Counts, which takes into account the frequencies of the wild-type residue and the replacement residue. The intuition behind this method is quite simple. Suppose that we have a position with a cysteine (C) conserved in 99% of homologous sequences. We clearly want to predict any mutation as pathogenic, like with simple conservation, and this is what PSIC does.

If, on the other hand, we have 80% of alanine (A), including the reference sequence (human), and 20% of phenylalanine (F), the pairwise conservation will be quite high (68%), so we would predict any mutation as damaging with a method like simple conservation. However, suppose that this mutation is from A to F, then we know these two residues are possible variants, so we should predict it as neutral. This is exactly what PSIC does. It would predict the A to F mutation as likely to be neutral, while A to Y (tyrosine) would be predicted as pathogenic (because there is not homologous with a Y at this position).

To be more precise, PSIC works by computing the logarithmic ratios of the likelihood of a given amino acid occurring at a particular position to the likelihood of this amino acid occurring at any position (background frequency). It does this for every position and every residue. The score it gives to a mutation (like A to F) at a given position is then the difference between the two log ratios.

PolyPhen performs well. When running PolyPhen, two choices are offered: HumDiv (default) and HumVar. The difference is the data set on which PolyPhen was trained. The authors of PolyPhen (Adzhubei et al., 2010) obtained better results with the HumDiv variant, however, here we get better results with HumVar (AUROC=0.804) compared to HumDiv (AUROC=0.801), although in the latter case it is still the second-best method.

Although these results are better than simple conservation, the difference is very small, especially considering the very high complexity of PolyPhen compared to simple conservation.

4.2.3 PSIC

Now, we can ask two interesting questions. What if PolyPhen used PSIC alone and no other information? What if we use PSIC on our own sequences instead of the ones chosen by PolyPhen? The first choice results in an AUROC of 0.774, which is lower than PolyPhen (0.804), showing that the other factors PolyPhen takes into account are really useful to improve prediction accuracy. When using PSIC on our set of sequences, we get an AUROC of 0.767, suggesting that PolyPhen chooses better the sequences to give to PSIC than we do.

4.2.4 FreqDiff

Although PSIC does not give very good results here, the idea to correct for "minority" residues in homologous sequences at a given position seems interesting. In the example above, it would clearly be wrong to predict the A to F mutation with the same score as A to Y, which is what conservation does. So I created a very simple variant, that I call FreqDiff (for frequency difference), which is simply the difference in frequency of the original residue and the replacement residue. In our A to F example, the FreqDiff score is simply:

$$f_A - f_F = 0.80 - 0.20 = 0.60$$

On the other hand, A to Y would give:

$$f_A - f_Y = 0.80 - 0 = 0.80$$

which is higher, and therefore predicted as more pathogenic.

What is interesting about this method is that in the case of a single dominant residue, it gives a similar result to a conservation method. For example, suppose we have 100 sequences, with 90 arginines (R) and 10 other different residues including only one tryptophan (W). FreqDiff would give almost the same score as simple conservation for the R to W mutation:

$$f_R - f_W = 0.90 - 0.01 = 0.89$$

which is very similar to the conservation with the reference (0.90), and also quite close to pairwise conservation (0.811).

The result of the benchmark is that FreqDiff performs quite well with an AUROC of 0.803, which is almost identical to the best method (PolyPhen HumVar), although it is incredibly simpler.

4.2.5 Functional Impact Score

Functional Impact Score (FIS) is a prediction method proposed by Reva et al. (2011) with a similar aim as PolyPhen. In this article, the authors made a benchmark on mutation effect prediction on cancer proteins, including the P53, and have obtained good results.

The functional impact score is the sum of two scores. The first score, called "specificity score", is a measure of conservation based on a similar idea as conservationTree, that is, the conservation of a residue in subtrees. Instead of using weighting as in conservationTree, they cluster the sequences and then measure the conservation in each cluster. Their second score is called "conservation score", and is based on a similar idea as FreqDiff (or PSIC) because it measures the difference of conservation between the wild-type residue and the replacement residue.

The authors provide a web server that, like PolyPhen, automatically collects the homologous sequences, and computes the score based on them. When compared against Varibench, we get an AUROC of 0.787, which is close to the result obtained with conservationTree (0.790).

4.3 Global benchmark

4.3.1 Why is it better?

The previous benchmark on individual proteins is interesting, but has a few drawbacks. As can be seen in Figure 4.1, the distribution of AUROC has peaks at 0, 0.5 and 1. This is because for many proteins, there are a few recorded mutations. For instance, 34% of proteins have less than 5 recorded mutations. In the worst case, there are only one pathogenic and one neutral mutation (10% of proteins). In this case, there are only 3 possible values of AUROC. Either the pathogenic mutations has a higher score, in which case the AUROC is 1, either it has a lower score, which gives an AUROC of 0, or finally, it has an identical score, giving an AUROC of 0.5. This is what explains the three peaks. Proteins with more information account for the non-zero area in 0.5-1 range in Figure 4.1.

Another consequence of this way to average the results for the different proteins is that the weight given to each protein is exactly the same. This might seem good but in fact, it means that a test on a protein with 2 recorded mutations is given the same weight as a test made on 50 recorded mutations. In fact, it would be better to give the same weight to every experiment in the data set.

Finally, an important problem with this benchmark methodology is the fact that the threshold is taken as an independent parameter for every protein. In a ROC curve, the rationale for varying the threshold is to make the benchmark independent on the threshold choice. But ultimately, the user will need to decide which value counts as "high" or "low". So in the end one has to decide one (or several) thresholds to classify the mutations as pathogenic or neutral.

But with the individual benchmark, the threshold is varied independently on every protein, as if we could choose the best threshold differently for every protein. In the real world, this is not possible, unless we can run a protein-specific benchmark like on P53, but the amount of information on this protein is exceptional. What we want here is a score for which a general protein-independent threshold makes sense. For example, we want to be able to say things like "With a threshold at 0.99 for PolyPhen, you can expect a false discovery rate of at most 10%".

4.3.2 Methodology

All these reasons explain why I think the best option is to take all known mutations together and run a single benchmark on all of them. We therefore produce a single ROC curve that we can directly look at, and a single AUROC that we can use to rank the methods from best to worst.

However, there is a small difficulty with this methodology. Since the benchmark is done on all mutations at the same time, the score should be comparable from protein to protein. For example, if we measure conservation by the absolute number of sequences with the same residue, instead of the *fraction*, it will not work, because the number of sequences for each protein is different, so comparing these numbers does not make sense. On the other hand, the fractions between 0 and 1 are comparable so they can be used here. In the individual benchmark, both choices were equivalent, since multiplying every score by a single constant (in this case $1/N$, with N the number of sequences) would not change the ranking.

What we need to make sure is that every method gives a score that is in a unit independent on the protein. This is required, but this may not be sufficient. For example, if we use the example above, and choose to use the fraction of identical residues (simple pairwise conservation), the unit is comparable, but it may not be wise to compare it directly. For some proteins, the average identity will be around 40% while for others it can be as high as 80%. A critical position in the first case could be a residue with only 70% of conservation, which is high compared to 40%. But in the latter case, a threshold of 70% would include every residue in the protein, and a threshold of 90% would be better. But since the threshold must be the same for all proteins, we need a way to transform these percentages in something that will make the 70% of the first case equal to the 90% of the second.

4.3.3 Z-score

A possible solution is to measure how much above average a score is. A widely used solution is the Z-score which is defined as:

$$Z = \frac{x - \mu}{\sigma}$$

where x is the original measure (for example residue conservation at the position), and μ and σ are respectively the mean and standard deviation of the score along the protein.

By definition, the Z-score represents how many standard deviations a value is above the mean. This correction improves a lot the predictions. For example, with simple conservation, the AUROC with the Z-score correction is 0.790, while it is only 0.764

without it. An alternative to the Z-score would be to divide by the mean, that is, to measure the percentage of variation to the mean. This gives an intermediate result, with an AUROC of 0.788. This pattern is also true for physico-chemical conservation and weighted conservation. This is why in Figure 4.2, we present only the results with the Z-score variants.

4.3.4 Z-cons FreqDiff

With the FreqDiff method, the correction cannot be applied directly, as the score applies, like for PolyPhen, to a position-residue pair. Therefore, the "average over the protein" is not defined, because there are different possible values for each position. We could average the 20 value for the 20 residues, and then average these averages (this gives an AUROC of 0.801), but it is simpler to compute the average and standard deviation of only the conservation factor in the FreqDiff formula (i.e. the first term) over all reference residues. More precisely, for each position, we measure the fraction of sequences with the same residue as the reference one. This gives as many values as positions in the protein. We then compute the average μ and standard deviation σ of these. Then, the "Z-cons FreqDiff" score for a mutation from residue X to residue Y at a position is defined as:

$$\frac{f_X - f_Y - \mu}{\sigma}$$

where f_X and f_Y are the frequencies (fraction of the number of sequences) with residue X and Y at the position.

I call it "Z-cons FreqDiff" because the Z-score correction is based on conservation only. Its AUROC is 0.802.

4.3.5 Results for conservation-based methods

Overall, we can see on Figure 4.2 that PolyPhen is the best prediction method, with an AUROC of 0.824. If we remove all other information except the PSIC score in PolyPhen, i.e. we have the PSIC score on PolyPhen sequences, we get an AUROC of 0.788 (see Figure 4.2). The difference between these two methods is the use of secondary structure annotations, structure and other annotations in the sequence.

When comparing PSIC on PolyPhen sequence and PSIC on our own sequences, we can see that PSIC performs much better (0.802 instead of 0.788). Unlike what we found with the individual benchmark, it seems here that we choose our sequences better for PSIC than PolyPhen does. Indeed, we use a much bigger database, and a lot more sequences (we have less stringent thresholds), which could explain the difference. Interestingly, when comparing PSIC, FIS and Z-cons FreqDiff, we get an identical AUROC of 0.802. The three methods are based on the same idea, but FreqDiff is simpler. However, one might argue that the Z-score correction makes it more complex, compensating the lack of probabilistic model compared to PSIC, or the complex combination of two scores in the case of FIS. In the end, their equivalent score suggests that they use the information (the three use residue frequencies) with an equal performance.

As with the individual benchmark, the simple and weighted conservation have a similar AUROC (0.794 and 0.790), while physico-chemical conservation does not produce a good prediction (0.766).

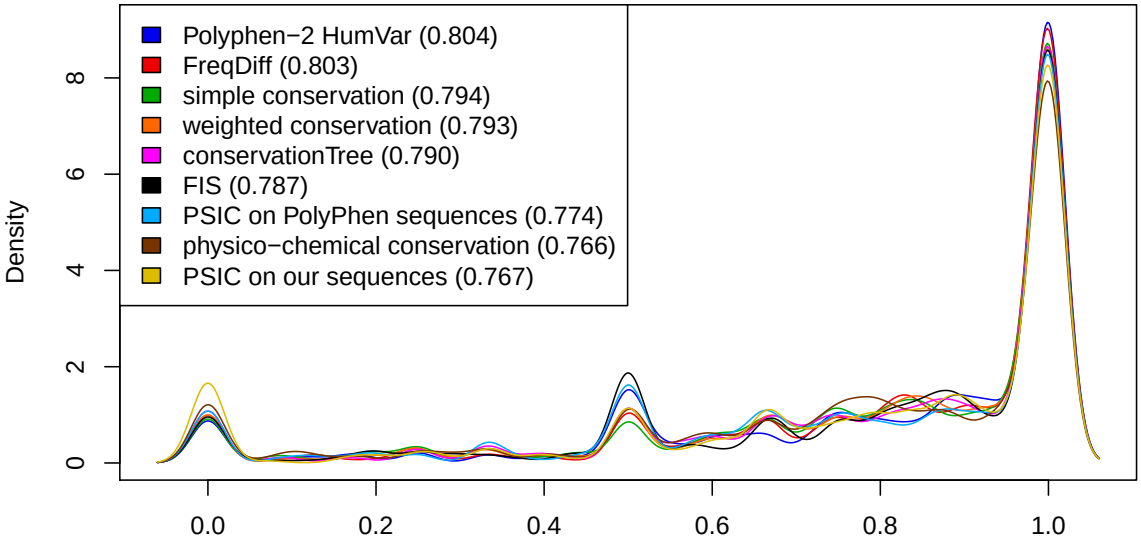

Figure 4.1: **Individual benchmark: Distribution of AUROC of 480 proteins with various methods.** The average AUROC on all 480 proteins is written in parentheses.

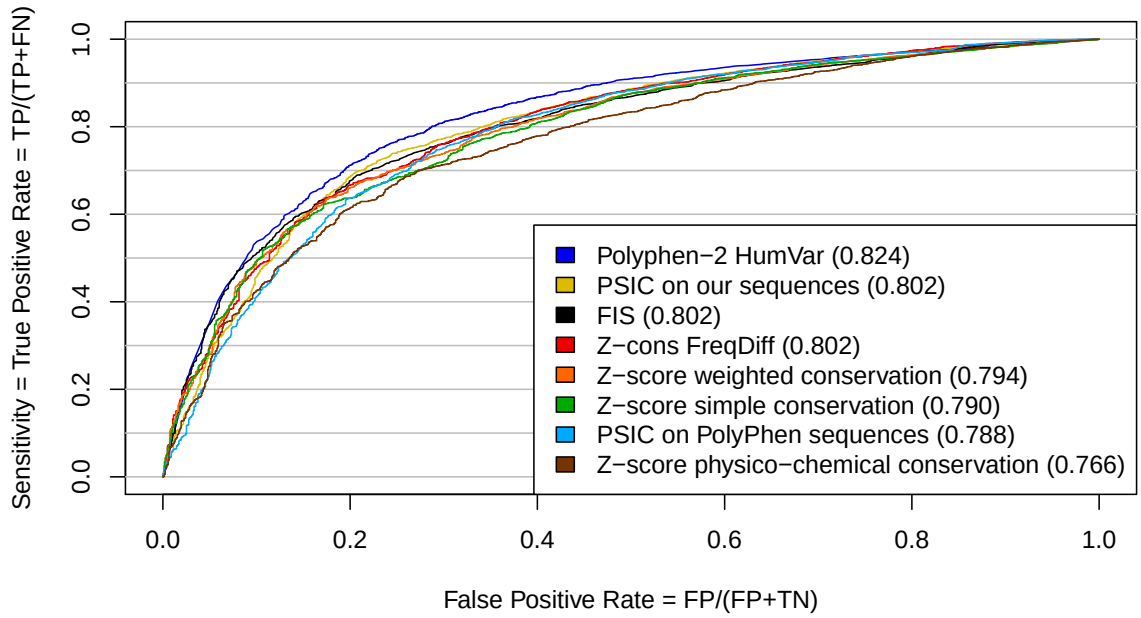

Figure 4.2: **Global benchmark: ROC curves for benchmark made on 7968 mutations on 480 proteins.** The area under the curves is written in parentheses.

Overall, it should be noted that there is a strong agreement between all these conservation methods, as can be seen in Figure 4.3. Simple and weighted conservation produce extremely similar rankings, with a Spearman coefficient of 0.99.

4.3.6 Results for coevolution-based methods

Here we have presented only conservation-based methods (FreqDiff, PSIC, simple conservation, etc.), or methods that use both conservation and non-sequence information (PolyPhen). But we also tested coevolution methods. As with P53, we tried to use BIS method directly (Dib and Carbone, 2012b). However, this method is adapted only to very highly conserved proteins, which explains its poor predictions with an AUROC of 0.711 (with $d = 1$, d+ mode, and Z-score correction). An alternative is to use the MST method (Baussand and Carbone, 2009), which is more adapted to less conserved proteins. However, using MST gives an AUROC of 0.693, which is even worse.

A possible explanation could be that coevolution methods are good only at detecting extra residues as important, but for non-coevolving residues, we should rely on another method. To test this, we used PolyPhen, because it is the best method, and added coevolution in the following way: If the residue is detected as coevolving, set the probability to be pathogenic to 1, otherwise, use the native PolyPhen probability. We use the MST method, which gives scores, and set a threshold of 1.5 (scores are on a scale of 0 to 2) to define a residue as coevolving. The result is an AUROC of 0.760, which is much lower than regular PolyPhen (0.824).

4.4 Double threshold

4.4.1 Purpose

We have supposed until now that we were interested in a single threshold that would divide best the pathogenic and the neutral positions. However, we might be interested in another question, which is to classify mutations in three sets:

1. Mutations that can be marked as pathogenic with reasonable confidence.
2. Mutations for which further experiments are required.
3. Mutations that can be marked as neutral with reasonable confidence.

This approach consists in defining two thresholds, a high threshold, which separates cases 1 and 2, and a low threshold, which separates cases 2 and 3.

To set the two thresholds, we need to define a maximum "error rate" that we accept. Here I chose to define it as a maximum false positive rate (FPR), which is the fraction of negatives (neutral) that we incorrectly predict as positive (pathogenic), and a maximum false negative rate (FNR), which is the fraction of positives that we incorrectly predict as negative.

If we set them both to 10% for example, we can then compute the two thresholds to use. For example, with Z-cons FreqDiff, these thresholds are -0.6685 (low) and 0.8303 (high). Figure 4.4A shows the distribution of Z-cons FreqDiff on all tested mutations,

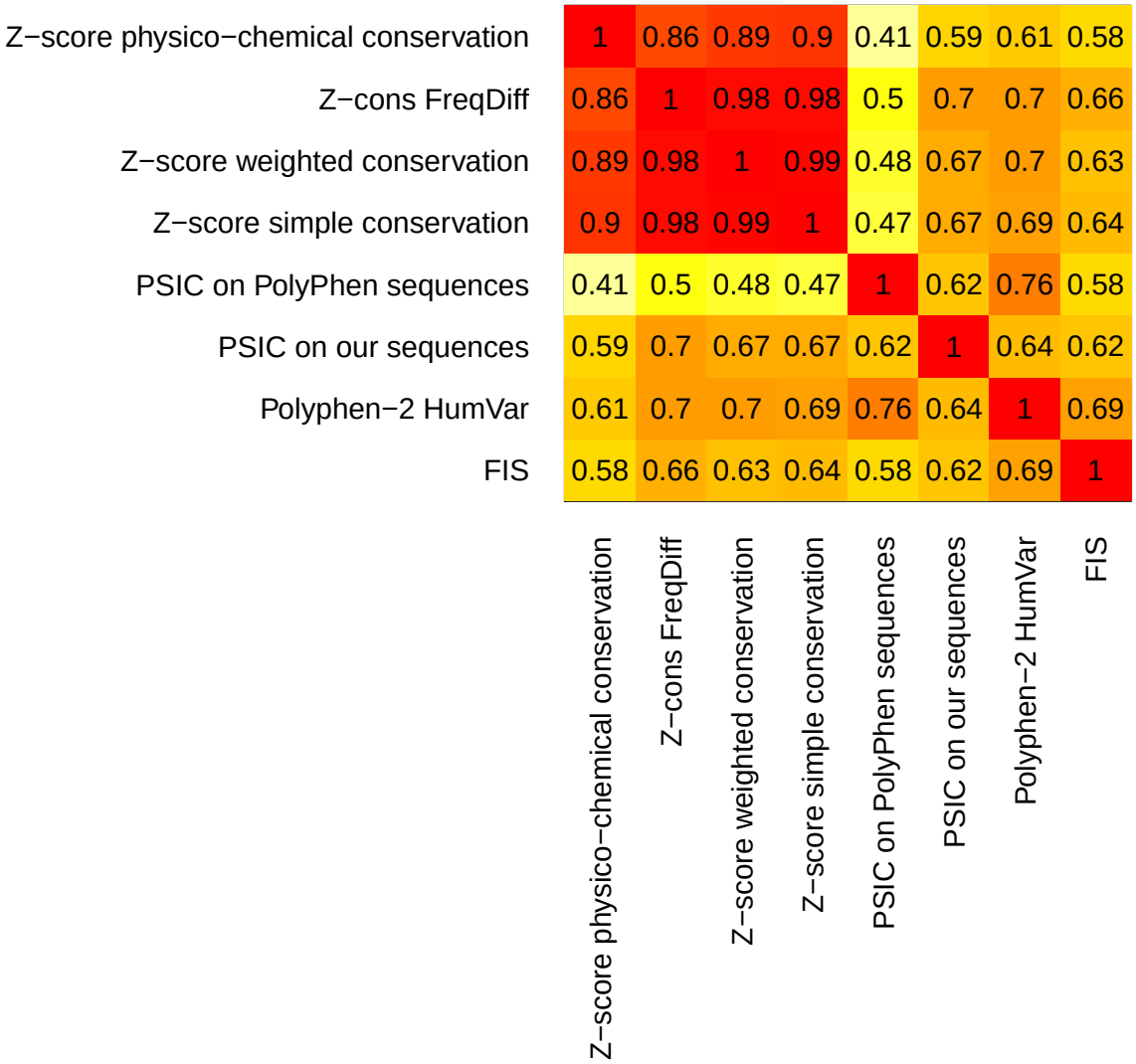

Figure 4.3: **Spearman correlations between scores given by predictions methods in the global benchmark.**
Spearman coefficients, instead of Pearson coefficients, are used because only the ranking can be compared and matters between different methods (which use different units in their scores).

with two vertical lines to mark the two thresholds. Now we can look at the fraction of mutations classified in the 3 categories (pathogenic, neutral and undecided) for the set of actually neutral and pathogenic mutations. If we then look at the neutral mutations, 10% are incorrectly predicted as pathogenic (by definition of the threshold), 45% are correctly predicted as neutral (TNR) and 45% are left undecided (the rest). The three values are shown by the histogram on the left of Figure 4.4B. We can look at the same classification for actually pathogenic mutations (Figure 4.4B, right).

This provides a new way to look at the results. With a fixed FPR and FNR of 10%, we can compare all methods by their TNR (True Negative Rate) and TPR (True Positive Rate). The undecided classes are simply the remaining fraction, so it is not extra information. Instead of 10%, we could also choose 5%. However, it is difficult to reach this error rate, because even with very extreme thresholds, all methods still make mistakes. Moreover, the database itself (Varibench) may contain errors. So in fact 10% can be seen as the sum of errors from the method and database. For example it could be that the method makes 5% of errors and the database contains 5% errors too.

4.4.2 Results

Here again, PolyPhen (HumVar variant) turns out to be the best method with TNR=53% and TPR=52% (Figure 4.5). Then we have, with comparable performance, PSIC (on our own sequences) with TNR=47% and TPR=46%, and Z-cons FreqDiff (Figure 4.4) with TNR=45% and TPR=48%. This ranking of the best three is identical to the one obtained with AUROC previously.

These results highlight the fact that in practice, if a reasonable error rate is specified (10%), there are still many mutations that cannot be classified as pathogenic or neutral. Even with PolyPhen, they still represent 38% of both neutral and pathogenic data sets. The fraction of correctly classified mutations is just above half (53% and 52%), which means prediction methods are still unable to correctly make a prediction in half of cases.

4.5 Comparison with another benchmark

A similar benchmark has been made by Flanagan et al. (2010). The authors assessed the performance of SIFT (Ng and Henikoff, 2001), PolyPhen and simple conservation in predicting the pathogenicity of 141 mutations that they tested experimentally. They found that the sensitivity of SIFT and PolyPhen was high, but their specificity was low. With simple conservation, they were able to obtain a high specificity. However, their benchmark methodology does not allow to test for all possible thresholds, like a ROC curve would allow. For conservation, they used a threshold of 100% in a set of 7 species. For PolyPhen and SIFT, they used the built-in thresholds proposed by the programs, which is also a choice that may make sense since these tools automatically classify mutations based on it. Finally, although their results seem to contradict ours, this can be explained by the different methodology. This suggests that high sensitivity can be achieved with PolyPhen when setting a higher threshold on the probability than what PolyPhen suggests by default.

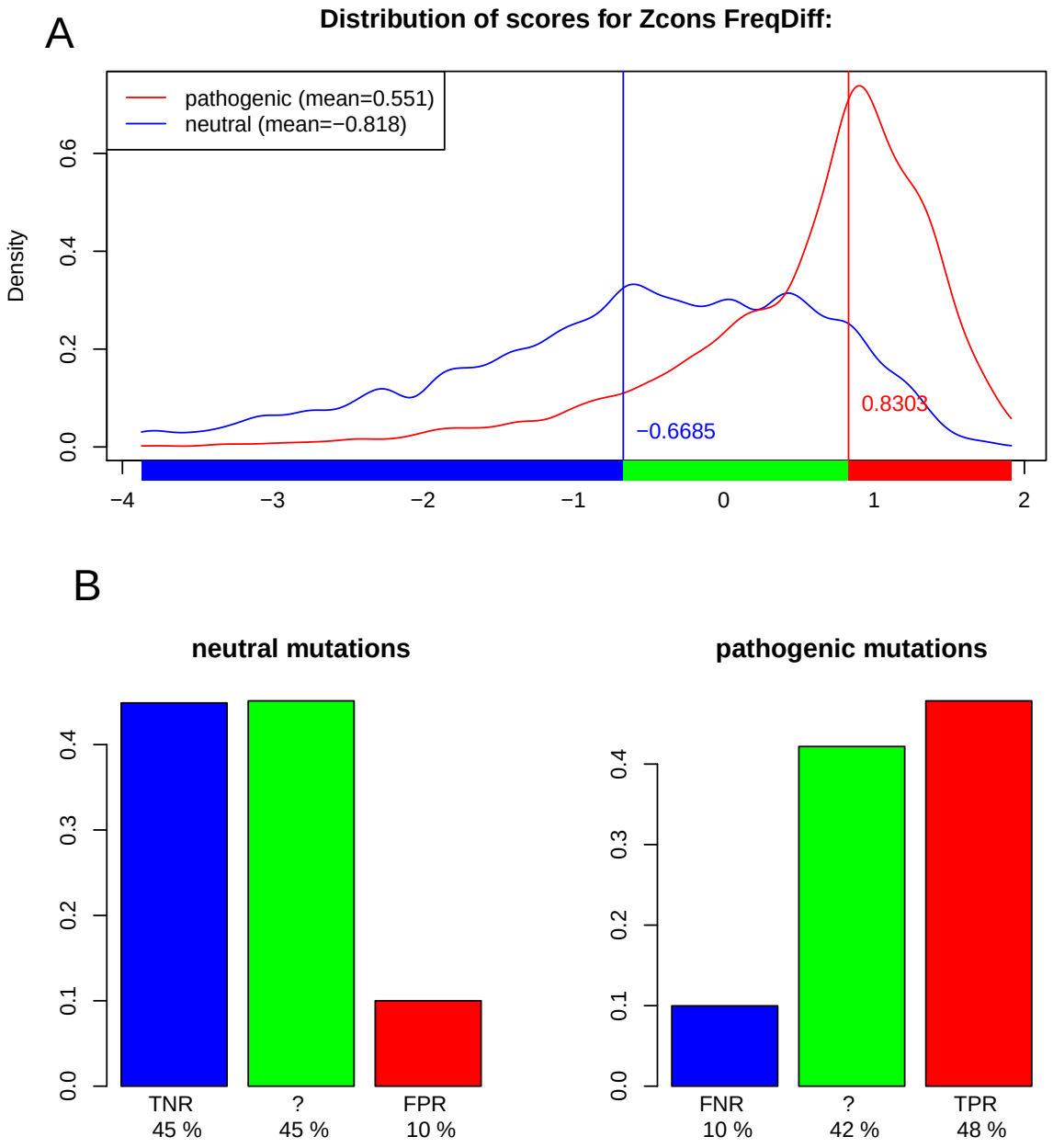

Figure 4.4: **Z-cons FreqDiff for an FPR and FNR of 10%**

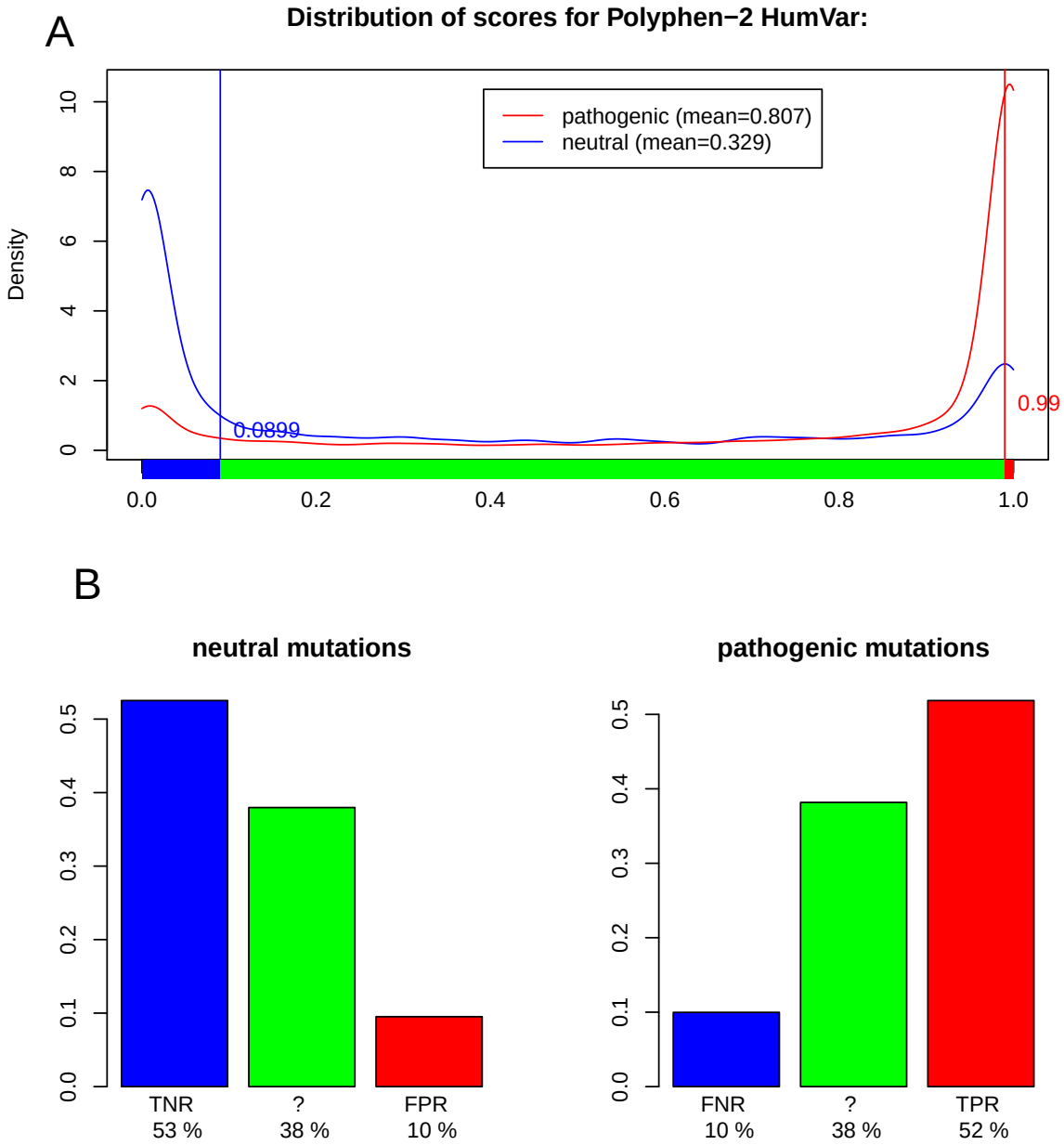

Figure 4.5: **PolyPhen for an FPR and FNR of 10%**

4.6 Conclusion

As we have seen, the individual benchmark has several drawbacks, and the global benchmark seems more appropriate. The best prediction method in our benchmark is PolyPhen 2 (Adzhubei et al., 2010). While most of the signal can be obtained with simply the difference of residue frequency between the wild-type residue and the mutated one, the extra information used by PolyPhen (secondary structure, solvent accessibility, etc.) improves the prediction accuracy. However, nearly half of the mutations cannot be reliably as pathogenic or neutral when an error rate of 10% maximum is allowed.

The difference with P53, where PolyPhen performed poorly, can be explained the by the fact that PolyPhen works much better when the alternative residue is specified, while the P53 benchmark was only designed to predict positions. It should also be noted that the Varibench database is expected to be of lower quality than the P53 database, which was cured with a sophisticated procedure (Edlund et al., 2012).

Similarly to P53, we did not improve prediction accuracy by considering coevolving positions. Here it seems that conservation, which can include several possible residues at a single position, is the only kind of sequence-based signal that is useful to predict mutation pathogenicity.

Chapter 5

Predicting HCV protein-protein interactions

Contents

We have seen in the previous chapters that coevolution signals are not very helpful to detect critical positions in proteins. However, it may be more useful to detect protein-protein interactions. The participation of the lab in the "MAPPING" project, and in particular the collaboration with François Pénin, has led me to work on the hepatitis C virus (HCV), in order to try to discover interactions between residues from different proteins with coevolution.

5.1 Biological background

Hepatitis C is a major health problem in the world, with 130 to 180 million people infected (0.84% in France). It is responsible of 350,000 deaths every year, and unlike hepatitis A and B, there is no vaccine against the virus. In December 2013, the new drug Solvadi (sofosbuvir) was released on the market. It is very effective, showing a success in curing 50% to 90% of patients, which is a major progress compared to the previously available treatments. However, hepatitis C is not a solved problem, since the treatment is not effective in 100% of cases.

The virus is composed of 10 proteins, which are synthesized in a special way. Instead of containing 10 genes, the virus contains a single gene, which is translated at once in a single big protein, called a polyprotein, which contains around 3000 residues. It is then cut into 10 separate proteins by enzymes (see Figure 5.1).

HCV strains can be divided in several genotypes, numbered from 1 to 7, which are dominant in different parts of the world. Genotypes 1 and 2 are the two most studied, and are themselves divided in sub-genotypes 1a, 1b, 2a and 2b. Between two different genotypes, there are typically 90% of identical residues, while there are around 95% identical residues between 1a and 1b or between 2a and 2b. Figure 5.2 shows a tree of the HCV strains we use in our analysis, extracted from euHCVdb (Combet et al., 2007).

5.2 Perspective

The question we are interested in is the discovery of protein-protein interactions, which would lead to a better understanding of the virus. The method usually used by biologists to discover interactions is to mutate a residue in one protein, let the virus evolve, and look at what other mutations the virus has used to compensate for the induced one. They can then infer that there is an interaction between the two residues.

Our methodology also uses the idea of compensatory mutations to discover protein-protein interactions. But where biologists create a mutant and let the virus evolve, we look at past evolution and try to detect the coevolution signature in available genomes, which shows how the virus had to evolve to survive. The advantage of our method is that it can be applied to the whole genome at once, while experiments are specific to a few tested residues.

Using the BIS method (see section 2.3.5), we can detect residues that probably coevolved together. If these residues are on different proteins, this suggests that these two proteins are interacting, with an interaction involving the two residues. The inferred interactions could then be tested in experiments where both supposedly interacting residues are replaced by an alternative we observed. The BIS method is especially adapted here because sequence conservation is very high in specific genotypes (90%), which is the type of problem the method was designed to solve.

The specific architecture of the genome is an advantage for analysis, because we can work directly with the complete genome as a single protein sequence. Unlike detecting protein-protein interactions between separately available genes, there is no problem in matching the strains together.

5.3 Analysis with BIS

To run the analysis, we take a subset of sequences, for example all amino-acid sequences from genotype 1a, align them and compute their phylogenetic tree. Then, we use the BIS coevolution detection method (Dib and Carbone, 2012b) to detect clusters of coevolving positions.

I have run this analysis on 7 sequences subsets:

- sequences from different patients with a similar genotype:

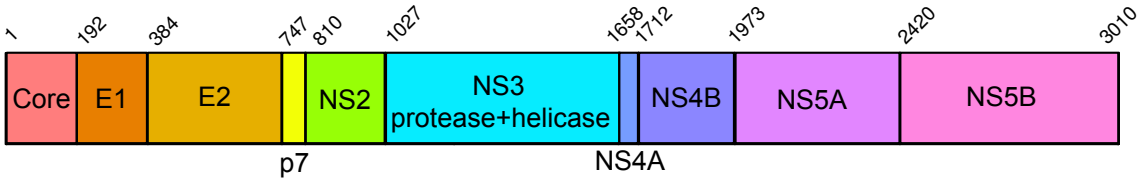

Figure 5.1: **The HCV polyprotein.** All HCV proteins are synthesized at once in a single polyprotein (shown here as a stripe), which is then cut by enzymes into 10 separate proteins (shown with different colors). Although NS3 is a single protein, it is composed of two parts, protease and helicase, that perform a different function, and are, for this reason, analyzed separately. The positions on top of the stripe are the residue numbers in the Con1 strain polyprotein.

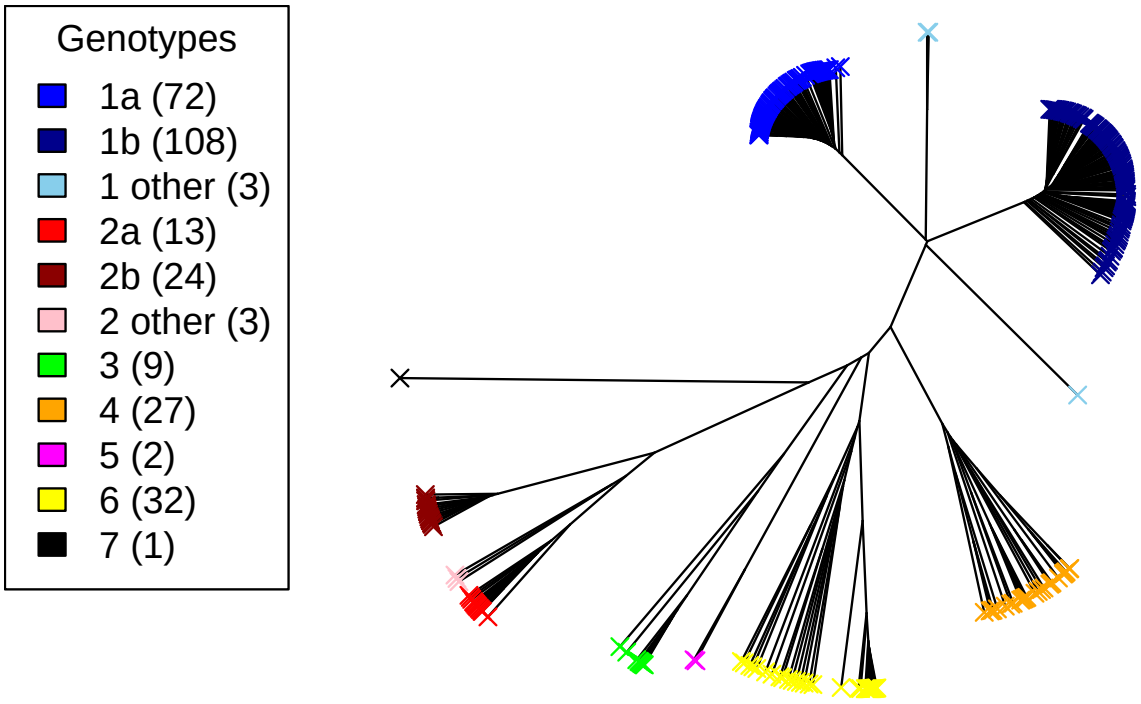

Figure 5.2: **Phylogenetic tree of the 325 HCV strains we use in our analyses.** The number of sequences (leaves) considered for each genotype (color) is written in parentheses.

  - genotype 1a (72 sequences)
  - genotype 1b (108 sequences)
  - genotype 2a (13 sequences)
  - genotype 2b (24 sequences)
  - genotype 4 (27 sequences)

- sequences from a limited number of patients:

  - 1a-Timm from genotype 1a (56 sequences)
  - 1b-MD from genotype 1b (40 sequences)
  - 1b-US-BID also from genotype 1b (112 sequences)

I did not analyze genotypes 3, 5 and 7 because they contain too few sequences (less than 15). I did not analyze genotype 6 either because it is in fact composed of many sub-genotypes (6a, 6b, 6c...), each of them with too few sequences.

To analyze the results with BIS, we set the number of allowed exceptions to, $d = 1$ (see section 2.3.5 for an explanation of this parameter), and the clustering parameter $\Delta = 5\%$ (default value). To be highly specific, we keep only clusters with symmetry and environment scores equal to 1 (the maximum), and, for the further analysis, consider only hits (positions with a pattern) and not blocks (hit + extension). This parameters mean that we will detect only clusters of coevolving positions with perfect patterns (like the ones shown on Figure 2.3), except on 1 sequence if this sequence is the only one to have a residue of one kind at one or some of the coevolving positions.

A question one might ask is why do I need to run the analysis on different genotypes separately, since I could align all sequences together and run BIS only once. In fact, I have tried this alternative by both mixing subgenotypes together (2a with 2b for example) or even all genotypes together (1, 2 and 4), but in that case we get many positions in the same coevolution patterns, which prevents us from knowing which precise positions interact with which other positions. The problem is that the evolution time scale is much longer, which results in many mutations, and we cannot distinguish those that coevolved together. On the other hand, by concentrating on a smaller time scale (with subgenotypes separately), we can see the residue-residue coevolution precisely.

BIS detects both conserved and coevolving positions in clusters. We get both clusters of perfect or nearly perfect (except for one sequence because $d = 1$) conservation, and clusters with patterns of coevolution. As we are not interested in conservation (as opposed with the P53 and Varibench benchmarks), we remove all these conserved clusters from the results.

With this filtering, we have only clusters showing true coevolution patterns. Each position found in a cluster will have at least two different residues, each one present in at least 2 sequences. So for example, an alignment column with VVVVVVVA is not valid (it is conserved except for one), neither is EEEEEEEE (it is fully conserved), but on the other hand, KKKKKKWW is valid. So, for example, the column displaying the motif KKKKKWW could be clustered with the column CCCCCCCEE.

5.4 Filtering clusters

As a first filtering step, I considered only clusters with a true coevolution pattern. Since BIS also detected conservation patterns, I had to remove clusters that are in that case. They consist in alignment columns with the same residue in all sequences, or columns with all but one sequence with the same residue. The second case can happen because we set the BIS parameter d to 1, which allows one sequence to violate the detected pattern.

With the remaining clusters, I performed a statistical test. Suppose we have a cluster where a hit column j has k sequences that have a residue and the l others have another residue. Because we used only clusters with symmetric and environment scores equal to 1, this property is true for all other hit columns of the same cluster. This means that any other hit column j' in the same cluster has also k residues of one type and l residues of another type, and in the corresponding sequences. Therefore this k/l distribution is a property of the whole cluster and not only of a single position. In some clusters, we also have 3 different residues, in which case we have a $k/l/m$ distribution.

For each k/l or $k/l/m$ distribution, I performed a statistical test which measures the probability to observe this pattern by chance. I do this by performing a Fisher test on a 2×2 or 3×3 matrix, respectively for 2 or 3 different residues, with zeroes in all cells except the diagonal, which contains the integers k and l (2 residues) or k, l and m (3 residues). This Fisher test gives us a p-value, which measures the probability to observe such a good (or better) pattern by chance.

Note that this statistical test measures only the probability to see such a distribution to appear by chance, given two positions, but not the probability to see it appear at any position in the sequence. It is therefore not based on a random sequence model, but is rather used as a method for sorting the most interesting clusters.

With this statistical test, we get a p-value for each cluster. I then use the Benjamini-Hochberg algorithm (Benjamini and Hochberg, 1995) to adjust the p-values for multiple testing, which allows us to control the False Discover Rate (FDR). We keep all clusters with an FDR lower or equal to 1%. We observe in particular that this automatically excludes all clusters with a k/l distribution where l or k equal to 1.

5.5 Results

When looking at the results from the different subsets, some show very few coevolving residues. This is the case for 1a, 1a-Timm, 1b, 1b-US-BID, and 2a. The number of detected positions in them ranges from 2 to 15. On the other hand, the most interesting analyses are 1b-MD, 2b, and 4, which typically have more than 40 detected positions. This is why we will concentrate on these analyses, and consider only them in the following post-analysis. This represents a set of 62 clusters.

The lack of results in the first group of analyses, except for 2a, can be explained by the too high number of sequences (ranging from 56 to 112) compared to the "successful" analyses which have a lower number of sequences (ranging from 24 to 40). It might seem surprising that a prediction method has problems with the presence of *more* information, be it nevertheless seems to be the case. We can explain this phenomenon by the number of exceptions. If we have more sequences, we are more likely to have more exceptions to coevolution patterns, but the way BIS is designed, it does not allow for several

sequences to have the same exceptional residue. For example in an alignment column KKKKVVVVVWA, the W and A are ignored if the exception parameter d is set to 2 because these two residues appear only once, but if we have KKKKVVVVVAA, then the two A cannot be ignored. With a high number of sequences (> 50), the probability to have twice the same exceptional residue becomes higher.

In the case of 2a, the problem is the reverse. We have so few sequences (only 13) that the Fisher tests are not significant, so the results are eliminated by the FDR criterion.

My solution here is simply to concentrate on data sets which produce enough results, which is why in the following part, I take the analyses from 1b-MD, 2b and 4.

5.6 Visualization

5.6.1 Interaction matrix

In order to visualize the results, I created a square matrix with as many rows and columns as proteins, where the values represent the amount of evidence for an interaction we got from coevolution. The matrices for the analyses with different genotypes are shown in Figure 5.3. For example, if we have a cluster with 3 positions in protein E1 and 5 positions in protein NS5A, we count 5 + 3 = 8 positions for the E1-NS5A interaction, so we add 8 to the cells E1-NS5A and NS5A-E1 (the matrix is symmetric). We also add 3 to cell E1-E1 and 5 to cell NS5A-NS5A, since these hits might also correspond to an intra-protein interaction. On the other hand, if we have only 1 hit in E1, we would not count any hit for the E1-E1 interaction, since a single position cannot be evidence for an interaction.

The result is shown in Figure 5.3. The main limitation of this approach is that longer proteins are naturally more likely to have more detected interactions. This is why we see that NS5A and NS5B have more interactions.

A possible solution to this bias would be to divide the number of interactions by the sum of the length of the two interacting proteins. I tried it, and the result is good, but it is even more interesting to split proteins by domain, as it solves the length issue (longer proteins are divided in more domains), but also adds extra information that are useful to biologists. The resulting matrix can be seen in Figure 5.4.

5.6.2 Interaction circle

I also created another visualization that allows to see the interactions on a circle that represents the complete polyprotein (Figure 5.5). For each cluster, a line is drawn between all pairs of positions in the cluster. When there are more than two positions in a cluster, this does not necessarily mean that all the positions in the cluster are effectively interacting with one another. This figure should then be understood as a set of possible interactions compatible with our coevolution patterns, rather than a set of validated interactions.

5.6.3 An example on structures

An interesting example to look at is the interaction we predict between p7 and NS2. When mapping it to the available structures, we can see the two interactions are consistent with experiments (Cook et al., 2013), as can be seen on Figure 5.6.

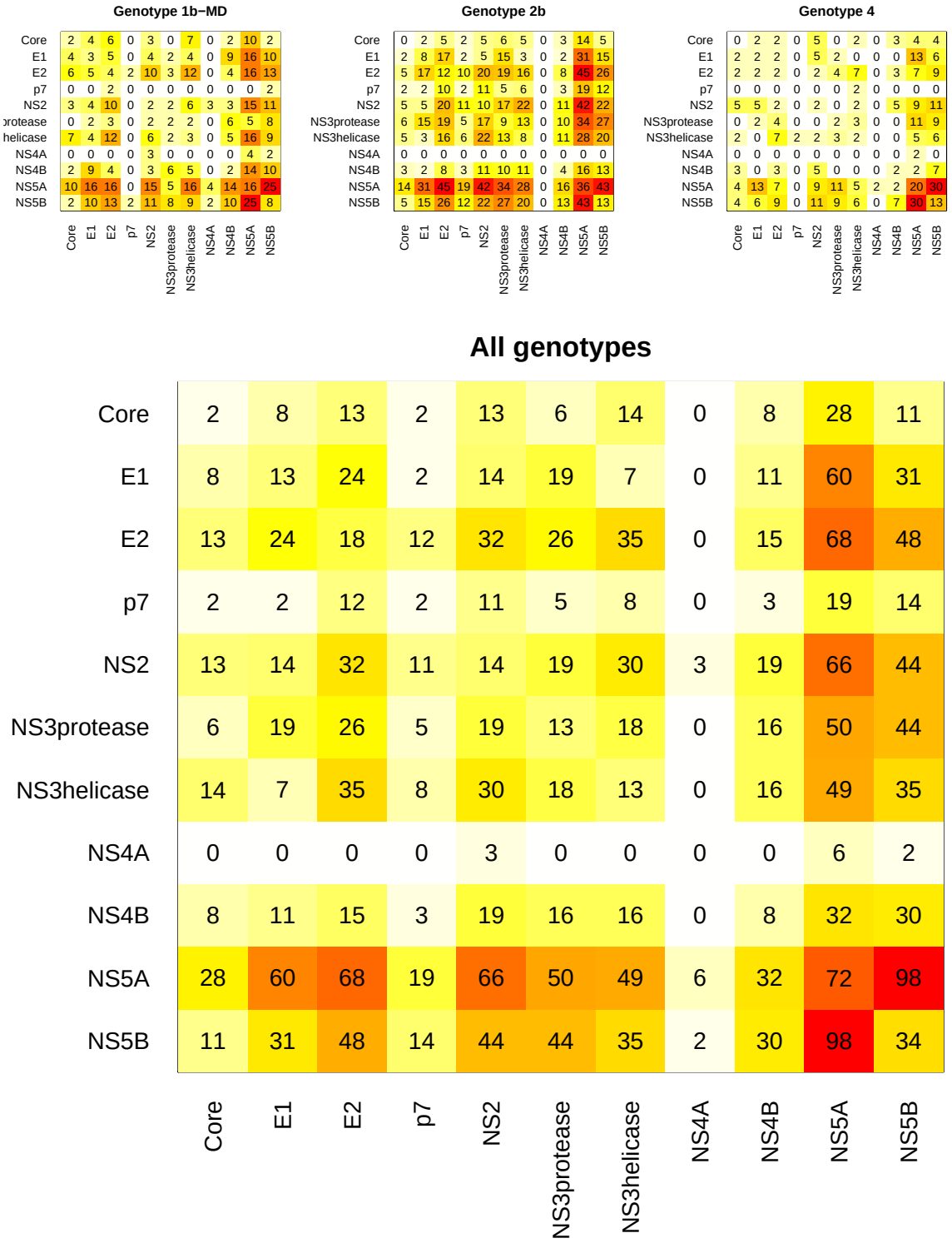

Figure 5.3: **Number of detected coevolving positions for every protein-protein interaction.** Small matrices (top) show the result for each genotype, and the big matrix (bottom) shows the sum over all genotypes. The diagonal (from the top left to the bottom right) corresponds to internal interactions between residues of the same protein.

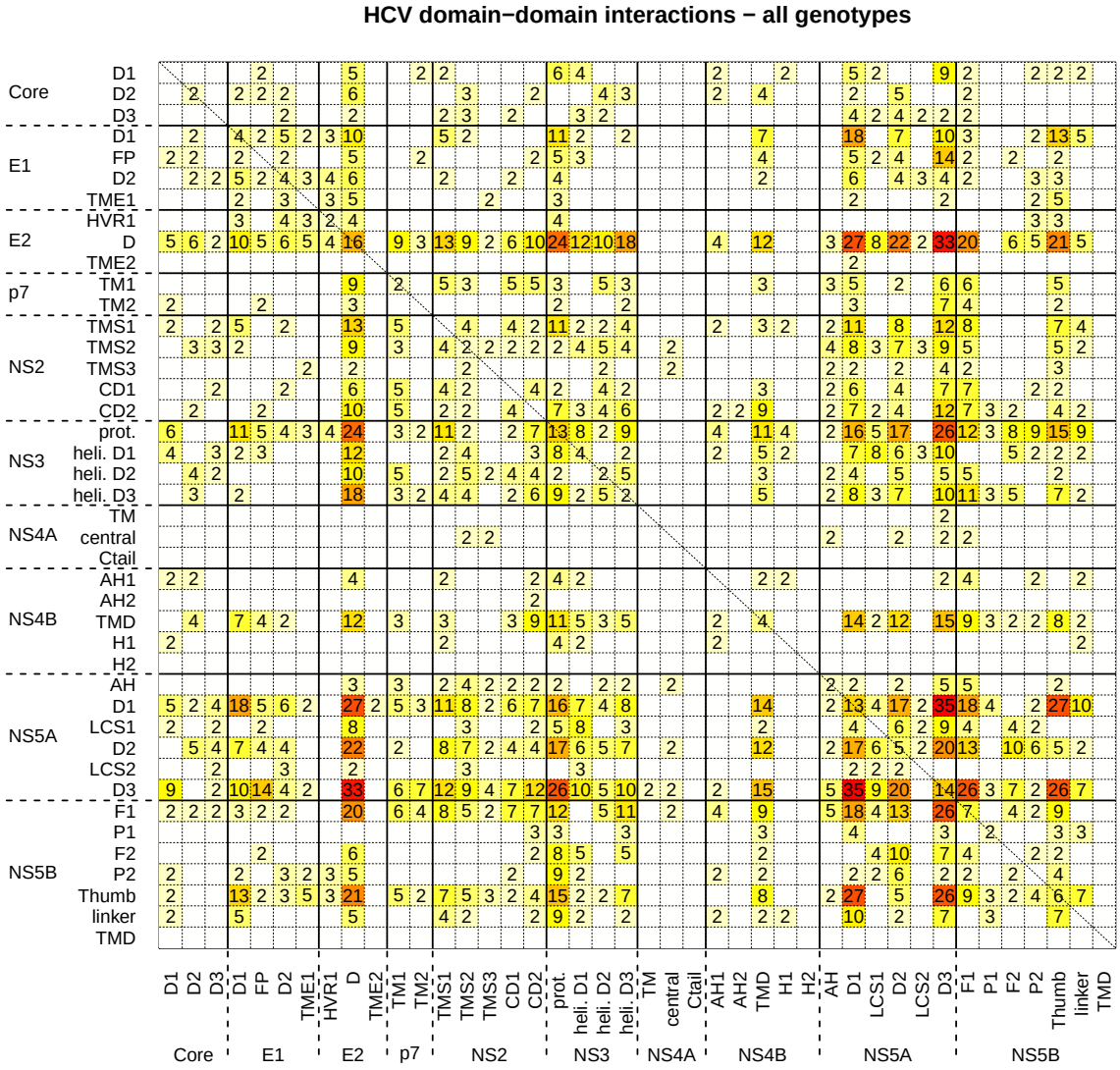

Figure 5.4: **Number of detected coevolving positions for every domain-domain interaction.** These values were summed over several analyses: 1b-MD, 2a, 2b, 4. Black lines are drawn to delimit different proteins.

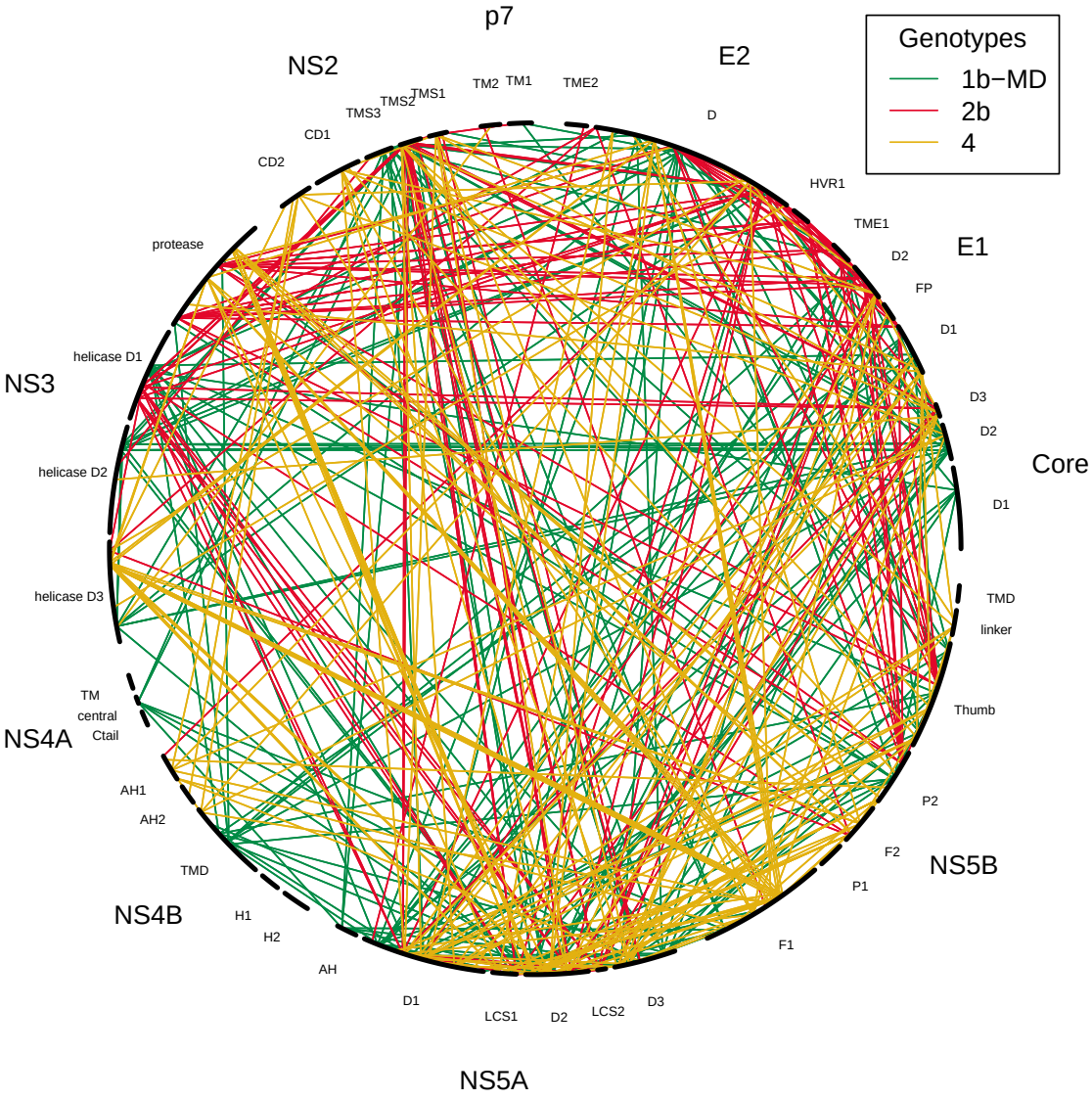

Figure 5.5: **Predicted interactions between residues.** The polyprotein is divided in proteins and domains, which an arc length proportional to the number of residues in each domain.

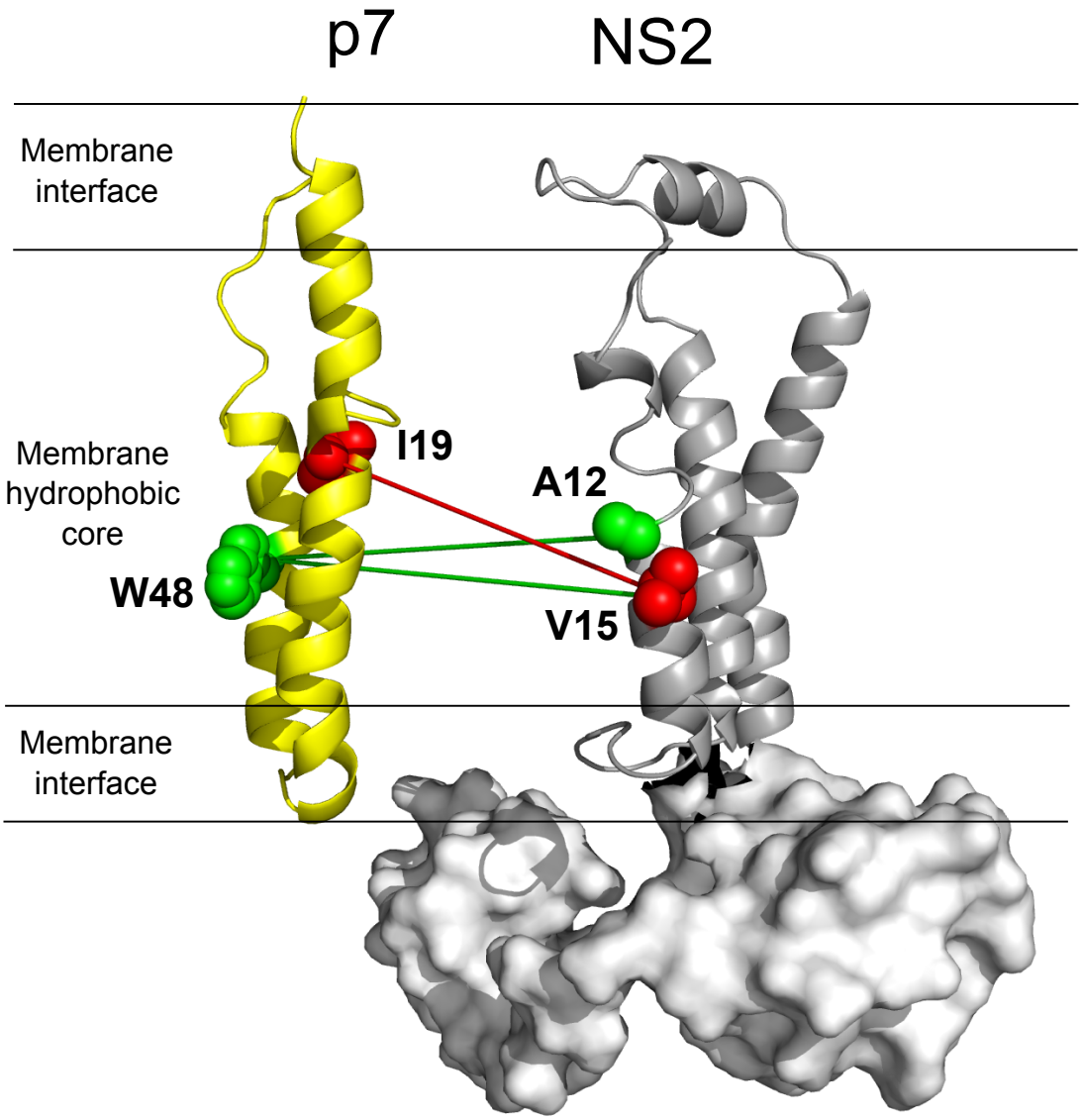

Figure 5.6: **Interactions between the p7 and NS2 proteins.**
The red line is an interaction we predicted with coevolution, while the green lines correspond to experimentally proven interactions from Cook et al. (2013)

5.7 Intra-protein interactions

Our goal with HCV is to detect protein-protein interactions. However, BIS does not discriminate between inter-protein and intra-protein interactions, as it simply detects interactions between residues, wherever they are. But intra-protein interactions can be useful, because they allow us to verify if the prediction method works on these proteins for which the structure is known. For many HCV proteins, we have a known 3D structure, so we can map the clusters discovered by BIS and see if some of them make sense. Typically, we expect to see small clusters corresponding to close positions in the 3D structure.

An example for the NS3 protein is shown on Figure 5.7. I have done the same for the other proteins, resulting in 8 similar structures. As can be seen, there are many clusters with only two positions, and we can see that they are close to each other, which we define here as closer than 10Å.

However, we may wonder whether this apparent proximity of coevolving residues is statistically significant. To check that, I have counted the number of interactions that are predicted and correspond to close residues, i.e. the lines I have drawn in Figure 5.7, and compared it to its distribution under a null model. I have built the null model by randomly shuffling positions (renumbering residues) in the predicted interactions, and repeated this 10,000 times to get a good p-value estimate. For the protein shown in Figure 5.7, the estimated p-value is 0.0102.

But it is better to make a general p-value estimation for all proteins because it avoids the problem of multiple testing, and allows us to gain statistical power. We simply do the same on all proteins, and get a global number of correctly predicted interactions (13), which we compare to the null distribution of this global number under the null model, which consists of a random permutation of residues in each protein while keeping the same organization of clusters. The general p-value we get is 0.0025, which is clearly significant, since it is lower than 1%. We can conclude that intra-protein coevolving positions detected by BIS tend to be close to each other more often than expected by chance.

5.8 Conclusion

We have shown that coevolving residues detected by BIS are reliable and correspond to true interactions between residues in proteins. We can therefore trust that at least some of the predictions of protein-protein interactions will be true. This result will help biologists who work on HCV to choose which interactions to test in wet-lab experiments, which will be necessary to prove the existence of the predicted interactions.

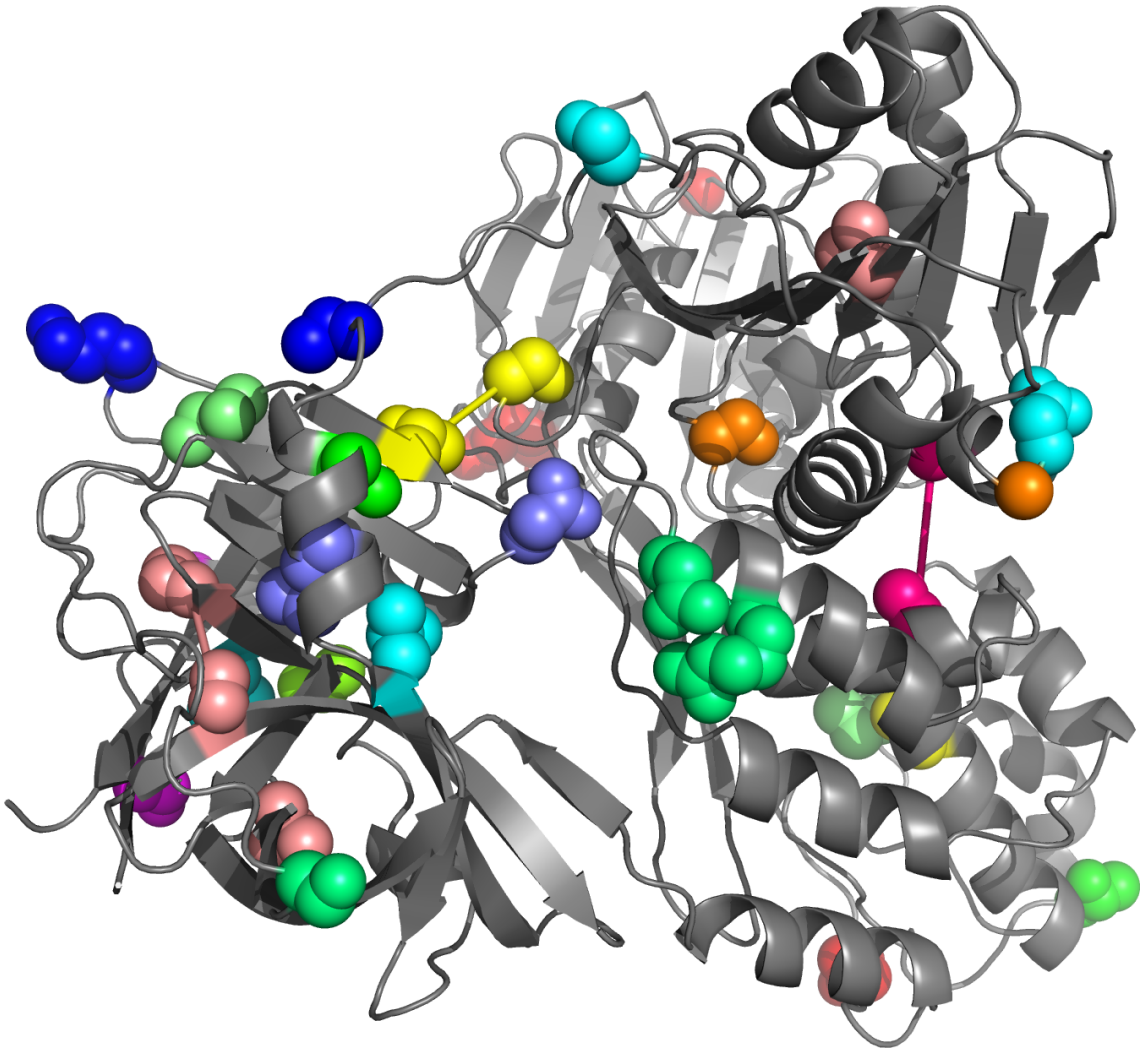

Figure 5.7: **Clusters of coevolving residues on the NS3 structure.**
Coevolving residues are represented by spheres and colored the same. Lines have been drawn between residues that are detected as coevolving and are close to each other (less than 10Å). These clusters come from several analyses: 1b-MD, 2b, 4.

Chapter 6

PruneTree: a sequence filtering algorithm

Contents

PruneTree is a program that allows to remove some (DNA or amino-acid) sequences in a set when they are too different from the rest, compared to the average level of similarity between sequences. I came to work on this method because this filtering process has already been used as a pre-processing step before detecting coevolution (Dib and Carbone, 2012b), but was previously done manually. It was observed that this pre-processing would greatly improve the sensitivity of combinatorial methods like BIS by removing the noise from coevolution signals. Here the goal was to design a method to perform this filtering automatically. I worked in collaboration with Anne Lopes, who had the original idea for this program when she was a postdoc at the lab. We improved the method together, and we created a benchmarking process, which I implemented. In this chapter, we describe the PruneTree method, the benchmarking process, and an example of application. This results will be written in an article (in preparation).

6.1 Algorithm

The goal of PruneTree is to remove some sequences in a set of homologous DNA or protein sequences. PruneTree does not work with the sequences directly, but instead uses the phylogenetic tree of these sequences, which can typically be computed from the sequences with a method like UPGMA (Sokal, 1958), Neighbor-Joining (Saitou and Nei, 1987) or PhyML (Guindon et al., 2010).

The PruneTree algorithm is detailed below. There are two parameters: $\alpha \in \mathbb{R}$ and $r \in [0, 1]$. The α parameter is a number of standard deviations, therefore values that make sense are typically in the $[-3, 3]$ range, although any real number is theoretically possible. Their default values (defined after an important parameterization step) are $\alpha = 0$ and $r = 0.1$.

1. The input is a tree T with L leaves.
2. Compute the distance between all pairs of (distinct) leaves in the tree. The distance between two leaves is defined as the sum of the lengths of the branches that one needs to follow to reach one leaf from the other.
3. Compute the mean μ and the standard deviation σ of these distances.
4. Define the distance threshold $M = \mu + \alpha\sigma$.
5. For each leaf x, count the number of other leaves that have a distance at most M with x. If this number is less than $r \times L$ mark x as "to be removed". Mark the others as "to be kept".
6. If the number of sequences to be kept is lower than 5, stop the algorithm and return T as the final tree. (This is a safety check, but this rarely happens.)
7. If all L leaves are marked as "to be kept", stop the algorithm and return T as the final tree.
8. Otherwise, remove the sequences marked as "to be removed" from T. Go back to step 1 with this new modified tree T.

If we use the default parameters we propose $\alpha = 0$ and $r = 0.1$, then this algorithm can be summarized by "keep only leaves that have at least 10% of leaves closer to them than the average distance between leaves", and "repeat it until no leaves are removed". The original idea of this algorithm was from Anne Lopes, while I added the idea to iterate it until a fixed point is reached (step 8 above).

The intuition behind this algorithm is to remove groups of leaves that are isolated. An isolated leaf is a leaf with too few close leaves. The words "too few" and "close" need to be defined. This is respectively the purpose of the M threshold and the r ratio. With $r = 0.1$, 10% of the leaves needs to be close to a leaf x so that x is not considered as isolated. As explained in the algorithm, M is defined to be $\mu + \alpha\sigma$. So the notion of closeness is relative to the distribution of distances in the tree.

There are two ways to choose the parameters. If we want to keep as many sequences as possible, and remove only very isolated sequences, we can choose to use for example $\alpha = 1$ and $r = 0.05$. This would remove groups of less than 5% of sequences which are at least at a distance $M = \mu + \sigma$ away from the rest the leaves. A group that is far away but has more than 5% is kept, because we consider that is sequences are not isolated since they are sufficiently numerous in the group. If a group of 5% is considered too small, then we can choose $r = 0.1$ to get groups of 10% of sequences at least. The general idea here is to have a big α value (for example 1, 1.5, or 2), which means that only sequences far away from the rest will be removed.

There is another way to set the parameters. If the goal is to get homogeneously distributed distances, we can use the default parameters $\alpha = 0$ and $r = 0.1$. The main point here is the value of α. Here M becomes simply the mean μ, so unlike the previous case, M does not define a notion of "far away" any more. Here the intuition is that each leaf should have a reasonable group of neighbors that are "at average distance". Precisely, the criterion is that 10% of leaves should be at less than the average, which is what can be expected for leaves that are "in the main group". It is this second approach that is used in the benchmark. We tested different pairs of (α, r) as explained in the section 6.3. We propose it as the default parameters. An example with these parameters can be seen in Figures 6.1 and 6.2.

I added the idea to iterate the algorithm until a fixed point is reached because I think it makes the method more consistent. Since it is a method that is expected to "clean" the data, it should consider its output as "clean." This is not the case if we do not iterate until a fixed point is reached. In Figure 6.1, it can be seen that some extra leaves (4) are removed at a later iteration than the first, which removes 21 of them. This case is representative of what happens in most cases: the first iteration removes many leaves, then the next iterations remove fewer and fewer.

6.2 Benchmark with Guidance

In order to benchmark the algorithm, we decided to use the Guidance program (Penn et al., 2010), and use its score as a measure of alignment quality.

Guidance takes as its input a set of sequences and aligns them, producing the original alignment. It then randomly removes sequences from the original alignment, then realigns the remaining sequences together. It does these two steps 100 times, and therefore produces 100 alignments with varying subsets of sequences from the original alignment. For each pair of aligned residues in the original alignment, it counts how many of the produced 100 alignments contain this pair also aligned together. Therefore, for each pair of residues, it records the fraction of alignments that have this pair (between 0 and 1). This is a measure of reliability of the alignment of these two residues together. These scores can then be averaged by column reflecting their robustness according to perturbations. Then, Guidance suggests to remove columns with low score/consistency.

Here, we do not use Guidance to remove columns. Instead, we are interested in its global score as a measure of alignment reliability. The idea behind, is that robust alignments will lead to good Guidance scores. So we take the "mean residue-pair score" given by Guidance, which is simply the average of all the pair scores it has computed, and use it as a measure of average alignment reliability between the residues of our sequences.

We decided to run a large scale benchmark using the PFAM 27.0 database (Finn et al., 2014), because it is a widely used database for homologous sequences, which contains a large set of families. Each family is set of homologous protein sequences.

First, to limit the computational time, we selected a subset of PFAM families to retain only families with less than 1,000 sequences. This subset contains 10,925 families, compared to a total of 14,831 families in PFAM. We think that this dataset reduction did not fundamentally change the benchmark, as we can expect such a large subset to be representative.

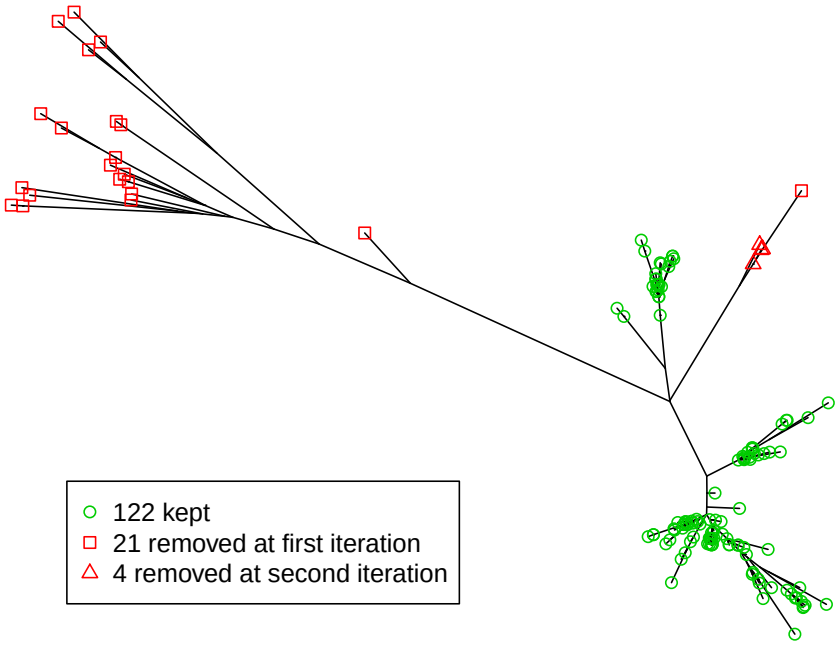

Figure 6.1: **Phylogenetic tree with leaves removed or kept according to PruneTree.** Sequences are from the PFAM family PF00159.13 (Pancreatic hormone peptide) with redundant sequences removed.

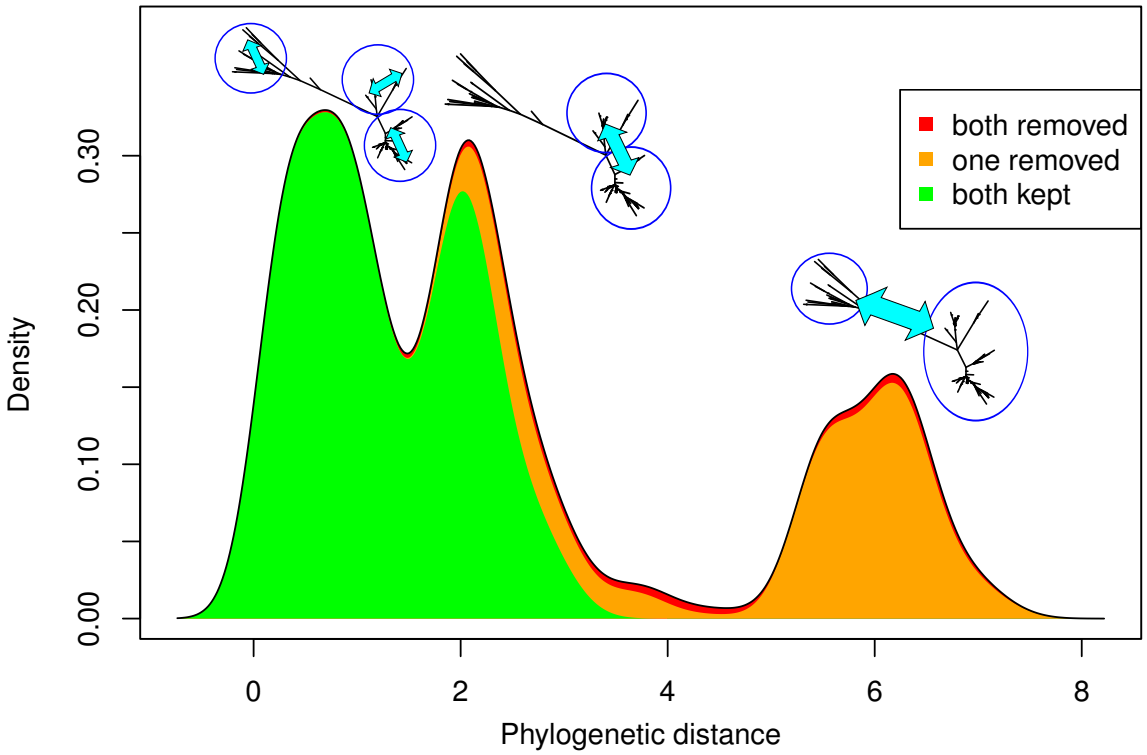

Figure 6.2: **Distribution of distances between all pairs of leaves in the tree of Figure 6.1.** The green corresponds to distances between pairs of green leaves in Figure 6.1, the red part to pairs of red leaves, and the orange part to distances between green and red leaves. If all red leaves are removed, the new distribution is therefore only the green part.

Then, we removed all redundant sequences in each family, that is, sequences that are exactly identical to each other. For each family, we used PhyML (Guindon et al., 2010) to infer the corresponding phylogenetic tree. This process was quite long (1 month for all families on approximately 40 CPUs), and fails for a few families, or was sometimes longer than one week so we canceled it. At, the end, this was the case for only 6 families and we finally obtained 10,919 phylogenetic trees.

On each of these trees, we applied PruneTree, which tells us the leaves to remove. For each family, we therefore have a new, smaller, set of sequences.

We then applied Guidance to both the original sets of sequences (before PruneTree) and the new sets of sequences (after PruneTree) to compare the Guidance scores. Finally, we can compare the Guidance scores of the families before and after PruneTree to see the influence of PruneTree on the reliability of the produced alignments. To do so, different correlation analyses were performed and are detailed in the following section.

6.3 Influence of the parameters

To define the default parameters, we have run the full benchmark process with different parameters. However, as it was too long to run the analysis for the 10,919 families, so we selected a subset of 1,000 randomly chosen families in the whole dataset. The results are shown in Table 6.1.

We want a good compromise between the Guidance score improvement (which we want high) and the number of removed sequences, which we do not want to be too high as this would remove too much information. As can be seen, the default parameters allow the average Guidance score to increase by 0.061, from an average of 0.798 to 0.860. On average, 16% of the sequences of each alignment are removed. Choosing for instance $\alpha = 0$ and $r = 0.15$ would remove 27% of sequences, which might be too much. On the other hand, reducing r to 0.05 would give a too small improvement of Guidance scores (0.037). However, different users may have different views on this choice, which is why we have designed PruneTree to allow the user to change these parameters.

6.4 Correlation analysis

For this part, we consider the full set of families (10,919) instead of the random selection. To understand better what the results mean, especially to see if there is a correlation between the gain in alignment reliability obtained with PruneTree and some properties of the PFAM family, I performed a correlation analysis on the results.

We decided to divide the families in three categories, depending on the original and new Guidance scores, which will help us analyze the results. Remember that Guidance scores are real numbers between 0 and 1.

- “improved” set: families where the Guidance score is improved of at least 0.05
- “already good” set: families where the original Guidance score is already at least 0.9, and that are not in the first set
- “not improved” set: all other families, i.e. where the score is improved by less than 0.05 (or made worse), and that do not already have a good Guidance score (> 0.9)

α	*r*	Average guidance score			Averge number of sequences		
		before PruneTree	after PruneTree	difference	before PruneTree	after PruneTree	% removed
0	0.05	0.798	0.836	0.037	240	222	7
0	0.1		0.860	0.061		200	16
0	0.15		0.884	0.085		171	27
0.5	0.05		0.820	0.022		232	3
0.5	0.1		0.835	0.037		223	8
0.5	0.15		0.848	0.050		212	13
1	0.05		0.809	0.010		236	2
1	0.1		0.818	0.019		233	3
1	0.15		0.824	0.025		230	5

Table 6.1: **Benchmark run over 1,000 randomly chosen families with different PruneTree parameters.** The row corresponding to the default parameters of PruneTree is highlighted in green.

Figure 6.3 shows the definition of these three categories applied on our families, and how Guidance scores of families are distributed.

An interesting correlation can also be seen between average sequence identity (the average number of identical residues between two homologous sequences) and Guidance score in Figure 6.4, with a Spearman coefficient of $\rho = 0.70$. We can also see that the "not improved" subset corresponds mostly to families in which PruneTree did not remove sequences, and not to cases where it wrongly removed sequences.

6.5 Species

Finally, it is interesting to look at the species in families. I tried to see if there is a different distribution of the three categories when we consider only bacteria and eukaryotes. The conclusion is that families consisting mostly (> 75%) of eukaryotes are easier to improve than others, with 48% of families in the improved subset, compared to 37% for other families. This is not caused by a simple difference in average identity, which is 39% on average in both cases.

A similar observation can be done with animals. In families with mostly animals, 48% are in the improved set, compared to 40% for others. A possible hypothesis is that animal sequences tend to be biased towards species that are close to human, giving a specific shape to the tree, with leaves being more and more spaced when we go further from human. In such a tree, remote species would be removed (like invertebrates or the few non-animals), giving a good delta Guidance score. On the other hand, bacteria trees do not have this shape, resulting in a tree in which it is harder to choose sequences to cut. Indeed, trees with mostly animals are, on average, cut more (18% leaves removed) than others (15% leaves removed), confirming this hypothesis.

6.6 Application to coevolution detection

Originally, PruneTree was developed to filter sequences prior to a coevolution analysis. In Dib and Carbone (2012b), the authors made a benchmark of the BIS method against several proteins with experimentally known critical positions. They have measured the performance of BIS in predicting them. In order to do that, they needed to perform a PruneTree-like operation, by manually removing sequences that are too far away from the rest. Indeed, they compared the results with and without filtering, showing that the filtering step improves the predictions a lot.

I have tested PruneTree on the Protein A – B domain, with the same sequence sets and benchmark methodology as Dib and Carbone (2012b). The authors analyzed two original sequences sets, which we call 28 and 490, corresponding to the number of sequences. The manually filtered versions have respectively 20 and 452 sequences, and give much better results when analyzed with coevolution (see Table 6.2).

When applying PruneTree to the two original sets, with default parameters ($\alpha = 0$, $r = 0.1$), we got very similar sets, with 24 and 453 sequences. The manually filtered subsets, with 20 and 452 sequences, are each included in these two. As can be seen in Table 6.2, the PruneTree-filtered sets give comparable results to the manually filtered sets,

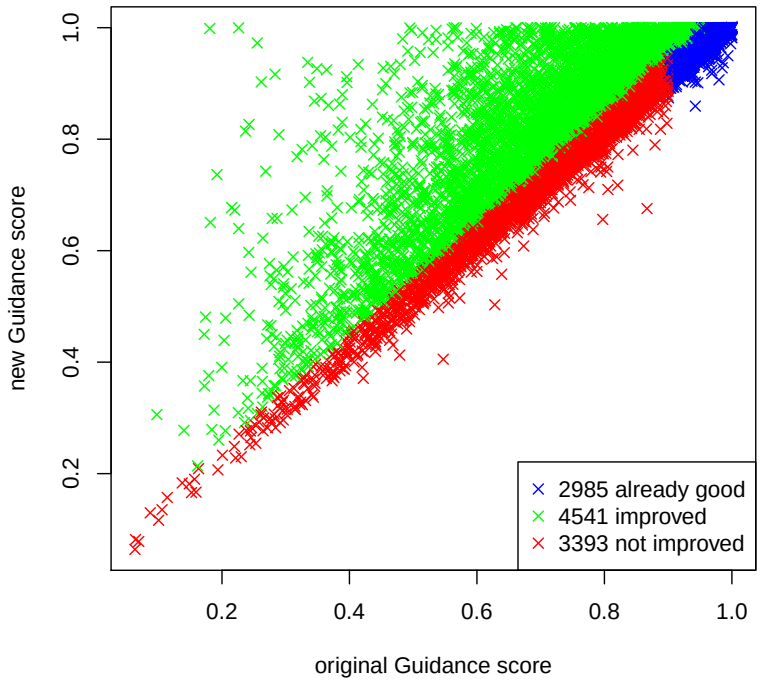

Figure 6.3: **Scatterplot of Guidance scores for all 10,919 families, before and after PruneTree, colored in the three categories we have defined.**

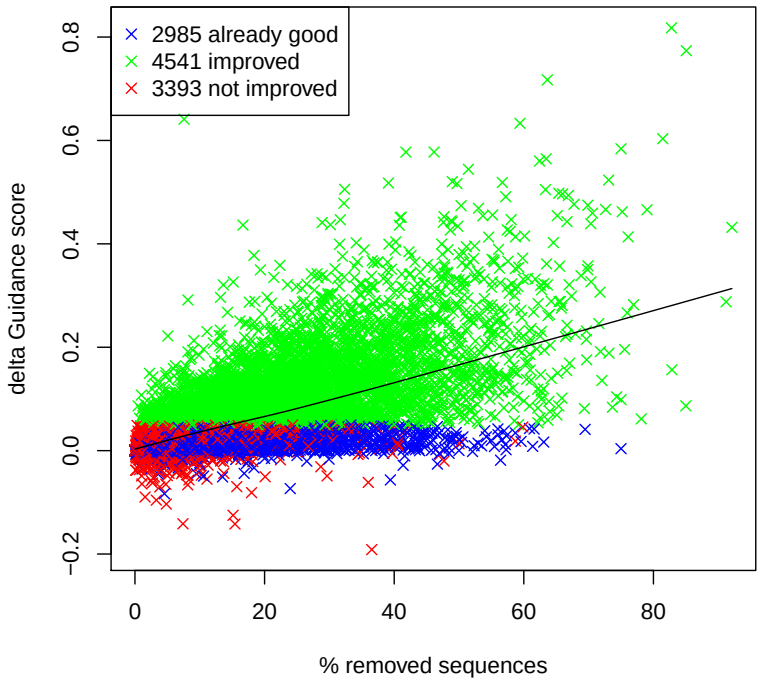

Figure 6.4: **Scatterplot of the difference of Guidance scores before and after PruneTree versus the percentage of sequences removed by PruneTree.**
The black line is a LOWESS regression. The Spearman correlation coefficient is $\rho = 0.70$.

showing that the automatic method reproduces what was done intuitively by looking at the trees.

Sequence set	N	True positives	False negatives	False positives	True negatives	p-value	Sensitivity	PPV
28	28	3	34	2	23	6.83e-01	0.08	0.6
28 manually filtered	20	24	13	4	16	1.30e-03	0.65	0.86
28 after PruneTree	24	19	18	2	20	8.57e-04	0.51	0.9
490	490	3	34	2	23	6.83e-01	0.08	0.6
490 manually filtered	452	22	15	6	14	3.17e-02	0.59	0.79
490 after PruneTree	453	20	17	6	14	7.11e-02	0.54	0.77

Table 6.2: **Benchmark of BIS on Protein A – B domain.**
N is the number of sequences and "p-value" is the p-value of the Fisher test between predicted and experimental positions. Sensitivity is the fraction of experimentally determined positions that were detected by BIS. PPV is the fraction of BIS predictions that are correct. Best values in each column are highlighted in green.

6.7 Conclusion and perspectives

The PruneTree method has shown to improve alignment quality, as measured by the Guidance score, while removing an acceptable amount of sequences. On an example of coevolution detection, it has proven successful by replacing a manual operation while keeping the same prediction accuracy.

However, more testing with larger biologically relevant datasets should be done. Measuring Guidance scores can give us an alignment quality measure but cannot tell use whether the removed sequences are useful or not. On the other hand, the coevolution example is a correct test, but is still a single example and lacks the power of a large scale analysis that could allow general conclusions.

To overcome these limitations, it would be interested to test PruneTree on positive selection detection, with a similar benchmark methodology as was used by Privman et al. (2012) for benchmarking Guidance. This method consists in creating simulated sequences with positive selection, then filtering the columns with Guidance, and finally predicting positive selection. The authors showed an improvement of positive selection identification when filtering the non reliable columns with Guidance. However, some positively selected sites were not identified possibly because of the column filtering operated by Guidance. The same benchmark study as Privman et al. (2012) could then be done with PruneTree. We hope that with PruneTree, which eliminates noisy sequences and thus preserves all the column information, we will be able to improve the positive selection prediction sensitivity.

Part II

Fourier transforms and genome patterns analysis

Chapter 7

Biological background

Contents

In this first chapter, we present quickly the biological background of the pattern analysis explained in the next chapter. Since most examples are related to homologous recombination, we explain this process here, as it typically occurs in eukaryotic cells like the model species *S. cerevisiae*. I also present the experimental data on *Phaeodactylum tricornotum*, which I also analyzed with similar methods. This analysis has been published[1].

7.1 Meiosis

Meiosis is an important step in the reproduction of many species, including most animals and plants. If we consider yeasts, we find that it occurs in some species like *S. cerevisiae* but not in others like *Candida glabrata*. In species that can perform meiosis, cell appear in two forms: haploid and diploid. In haploid cells (for example sperm and egg cells in animals), a single copy of the complete genome is found, so there can only be one allele (variant) of each gene. On the other hand, a diploid cell contains two different copies of the same genome, each one originally coming from a haploid cell. Each gene is present twice, and may therefore be present in two different alleles.

In animals, during the process of egg fertilization, the sperm and eggs cell fuse, making each chromosome present in two variants, called homologous chromosomes. The resulting cell is therefore diploid. In yeast, the process is similar, except that yeast can live both in the form of diploid and haploid cells. Meiosis is the reverse process, by which

[1] Rogato, A., Richard, H., Sarazin, A., Voss, B., Cheminant Navarro, S., Champeimont, R., Navarro, L., Carbone, A., Hess, W., and Falciatore, A. (2014). The diversity of small non-coding RNAs in the diatom *Phaeodactylum tricornutum*. *BMC Genomics*, 15(1):698.

a diploid cell is divided to form haploid cells. However, these haploid cells are not genetically identical to the haploid cells that originally formed them, because the process of homologous recombination occurs. It allows the two original genomes to be mixed, which enables the creation of a new genotype for the resulting haploid cells.

7.1.1 Homologous recombination

A homologous recombination can be either a crossover or a non-crossover. The diagram in Figure 7.1 shows the case of a crossover. During meiosis, the (double-strand) DNA of each chromosome is duplicated in two strictly identical copies called chromatids. Then, when a crossover happens, a part of a chromatid is exchanged with a part of another chromatid from the homologous chromosome (as shown in Figure 7.1). In the case of a non-crossover, instead of a reciprocal exchange, one part of a chromatid is replaced by a copy of the other chromatid.

In *S. cerevisiae*, these recombination events have been extensively studied. It has been observed that they occur more often at some specific places than others. These positions are called recombination hotspots, and their positions make one of the datasets we analyzed (in the next chapter).

If we look closer at the mechanisms that produce the recombination events (both crossovers and non-crossovers), we find that the Spo11 protein induces DNA double-strand breaks, which can either be repaired, in which case no recombination happens, or can lead to a recombination event (Zickler and Kleckner, 1999). In this case, a few proteins are involved in the process, some which interact with the chromosome axis, as observed in *S. cerevisiae* (Panizza et al., 2011). A diagram showing the interactions of these proteins can be seen in Figure 7.2. A group of these proteins have a very similar distribution (Red1, Hop1, Rec114, Mer2), which we try to model in chapter 9. We were also interested in modeling the double-strand break distribution, which has been observed with high resolution in Pan et al. (2011).

7.1.2 Experimental data

The data we used come from various experiments that we describe here.

Recombination hotspots In Mancera et al. (2008), the authors have fused haploid cells from two different strains of *S. cerevisiae* (S288c and YJM789). They therefore have diploid cells in which the homologous chromosomes differ at many positions. Then, meiosis is induced, resulting in haploid cells. The DNA of these haploid cells is then put on a microarray, containing probes for both yeast strains. At every position in the genome that is different in the two strains, it is therefore possible to know from which strain this part of the chromosome comes, by comparing the signal from the probe in one strain to the probe at the same position in the other strain. Every time we see a "switch" between the two strains, we know that a recombination event has occurred between the two successive probes. For example, if at position p the probe with the highest signal is the one from S288c, while at p' the probe with the highest signal is YJM789, then a crossover or non-crossover has occurred between p and p'. The range is typically in the order of 100 nt, so this produces a mapping of recombination that is at a quite high

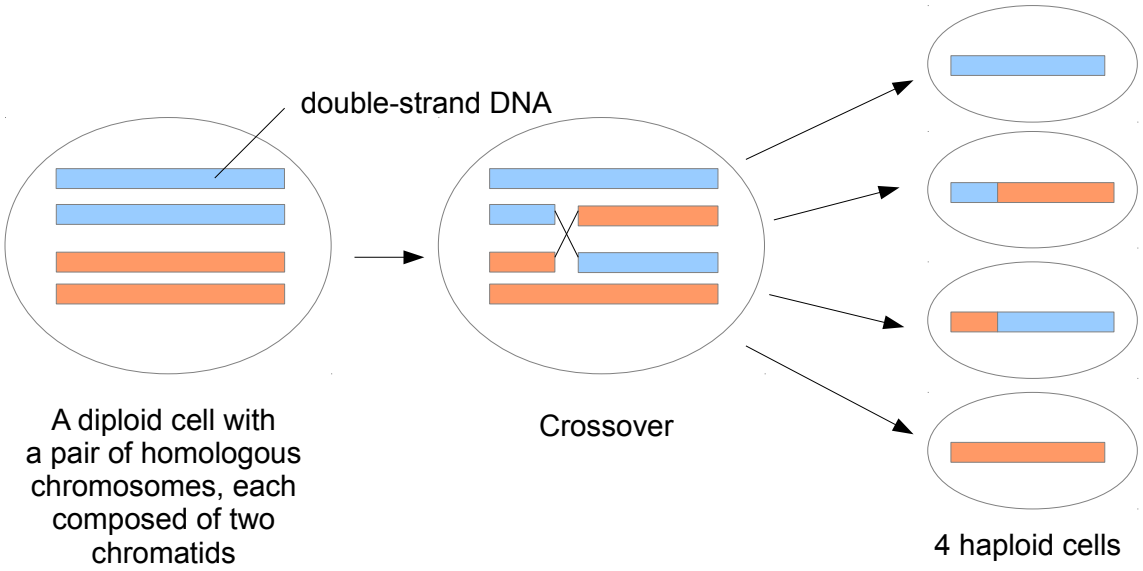

Figure 7.1: **Principle of homologous recombination during meiosis.**
Only a single pair of homologous chromosomes is shown, but this process occurs for each pair. Identical colors represent identical genetic content.

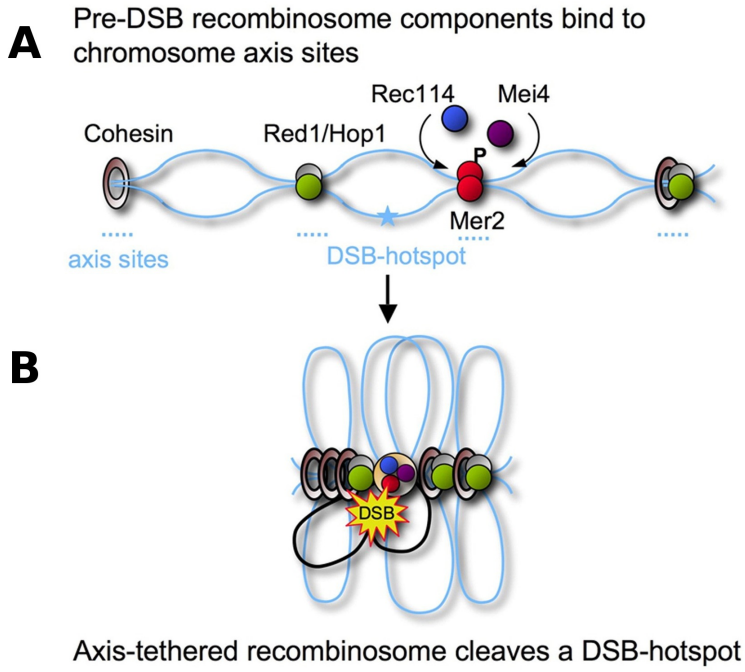

Figure 7.2: **The *S. cerevisiae* recombinosome.**
(A) After premeiotic DNA replication, cohesin, axial element components Hop1 and Red1, and the pre-DSB recombinosome subunits Mer2, Rec114, and Mei4 bind to axis sites. Hotspots are located between axis sites.
(B) Ordered, linear arrays of loops emerge after condensation and sister chromatids are conjoined in the developing axis. The recombinosome with Spo11 is anchored at the axis and interacts for cleavage with one of the surrounding hotspots.
This figure was reproduced from Panizza et al. (2011) with permission from Elsevier Limited.

resolution. The authors then extracted spots in the genome at which recombination occurs more than usual, allowing them to produce a list of recombination hotspots, that we used in our analyses.

Double-strand breaks One data set we used extensively in our analyses is the high-resolution distribution of DNA double-strand breaks during meiosis produced by Pan et al. (2011). The double-strand breaks that happen during meiosis are caused by the Spo11 protein. This protein binds to DNA, creates a double-strand break and then unbinds from DNA, taking a small fragment of DNA. By sequencing these fragments and mapping them on the reference genome, the authors were able to get a high-resolution distribution of Spo11-caused double-strand breaks on the yeast genome. This experimental data has a nucleotide-level precision.

Spo11-accessory proteins The chromosome axis is formed by specific proteins called cohesins (Rec8 in yeast). They bind to the DNA and then to each other, forming the chromosome structure during meiosis. A diagram of this process is shown in Figure 7.2. Several other proteins bind at the same positions as cohesins: Red1 and Hop1. It is not clear whether they bind at these spots because cohesin is there or independently, as their distribution depends on the presence of cohesin in some regions but not in others (Panizza et al., 2011). However, their distribution is very similar to cohesin. They are highly correlated locally with cohesin, while the general amount of cohesin is globally higher near centromeres, unlike Red1 and Hop1. On the other hand, Red1 and Hop1 have an almost identical distribution over the whole genome. Several other proteins also bind to these three: Mer2, Rec114 and Mei4. These proteins are involved in the recombination process. In the case when a double-strand break is caused by Spo11 and results in a recombination, these other proteins play a role in the process. The authors of Panizza et al. (2011) measured the distribution of these proteins along the whole genome, using ChIP-on-chip technology. We designed a model for the distribution these proteins. Since Red1, Hop1, Mer2, Rec114, Mei4 have an almost identical distribution, it does not matter a lot which one we choose as a reference, so we chose Red1 because it is one of the two (with Hop1) that are there before the others. On the other hand, Mer2, Rec114 and Mei4 bind in turn to Hop1 and Red1, so their distribution is a consequence of Red1/Hop1.

7.2 Small RNAs in *Phaeodactylum tricornotum*

I have also worked on a completely unrelated dataset, on which I applied the same methodology as on yeast recombination data. This dataset comes from an experiment made in the lab on *Phaeodactylum tricornotum* (Rogato et al., 2014).

Marine diatoms constitute a major component of eukaryotic phytoplankton and stand at the crossroads of several evolutionary lineages. These microalgae possess peculiar genomic features and novel combinations of genes acquired from bacterial, animal and plant ancestors. Furthermore, they display both DNA methylation and gene silencing activities. Yet, the biogenesis and regulatory function of small RNAs (sRNAs) remain ill defined in diatoms.

This is why the diatom genomics team in our lab made the first comprehensive characterization of the sRNA landscape and its correlation with genomic and epigenomic information in *Phaeodactylum tricornotum*.

This experiment performed in the lab consists in a high-throughput sequencing of short RNA fragments extracted from cells grown under different conditions. It has allowed to get a better knowledge of the diversity and function of non-coding RNAs in this organism. Most of sequenced small RNAs are in the 25 to 30 nucleotides range. In order to find the origin of these RNAs in the genome, they were aligned to the available genome. It is therefore possible to measure the coverage of small RNAs along the whole genome, defined at each position by the number of fragments including this position. When looking at this coverage, they observed a periodical distribution of non-coding RNAs in several highly methylated regions. I could therefore apply the Fourier method I have developed to detect periodicity automatically over the whole genome, and run some statistical analysis on the periods (see next chapter).

Chapter 8

Fourier transforms for genome analysis

Contents

Discrete Fourier transforms are a remarkable mathematical tool for the analysis of periodical signals. They are used in various domains ranging from acoustic signals to electrical power distribution. Their use for signals on genomes is more recent, but has proven successful for a few biological questions. I have extended the previously proposed methods and created new ones, to answer new biological questions about periodicity on genomes.

The first use of discrete Fourier transforms on genome distances dates back from Wright et al. (2007), who studied the periodic distribution of evolutionarily conserved gene pairs using Fourier transforms. This idea was later adapted in our lab by Mathelier and Carbone (2010), who used it to detect a periodic distribution of core genes in *E. coli*. The methodology used there was later re-used by an internship student, Sijia Niu, who applied it to several genomic markers in *S. cerevisiae*. However, this work was unfinished as no statistical assessment of results had been done. My first task was to build a correct statistical model to assess the results, then I extended the approach to further analyze the results. Finally, I developed a new methodology to analyze other types of signals, which was successfully applied to an experiment performed in the lab on *Phaeodactylum tricornotum*. This specific application has been included in an article published in BMC Genomics (Rogato et al., 2014).

8.1 Detecting periodicity on a set of genomic positions

The methodology developed by Mathelier and Carbone (2010) and adapted by Sijia Niu is used to detect a periodic positioning over a genome of several genomic markers. The input data is then simply a list of positions of these genomic markers as coordinates on the genome (chromosome and nucleotide number from the beginning of the chromosome). In Mathelier and Carbone (2010), the authors searched for codon biased genes of *E. coli* using the SCCI method described in Carbone et al. (2003), and then considered the positions of these genes as the set of positions to analyze. The methodology is generic and has then been applied by Sijia Niu to the biased genes of *S. cerevisiae*, but also to replication origins and tRNA positions. Since *S. cerevisiae* has several chromosomes, unlike *E. coli*, the method needed to be generalized, and it is this generalized method that we present here. The actual implementation of the discrete Fourier transform is the widely used Fast Fourier Transform algorithm (FFT), which is so common that the discrete Fourier transform itself is now often referred to as "FFT".

8.1.1 Algorithm

Let S_c be the set of positions (in $\mathbb{N}$) given as the number of nucleotides from a given origin on chromosome c. For instance, if we work with the classic model species *S. cerevisiae*, we have 16 sets $S_1, S_2, \ldots, S_{16}$ corresponding to the 16 chromosomes, and the origin is considered to be the first base of the sequenced chromosome.

The general procedure we apply is the following:

1. First, for each chromosome, we compute distances for all possible pairs. That is, for a given set of positions $S_c = \{p_1, p_2, \ldots, p_{n_c}\}$ along a chromosome c, where n_c

is the cardinality of S_c, we define distances $d_{i,j} = |p_i - p_j|$ for all $j > i$. This gives us a list D_c with $n_c(n_c - 1)/2$ distances:

$$D_c = (|p_i - p_j| : i = 1 \ldots n, j = i + 1 \ldots n)$$

Note that there may be several identical distances in D_c.

2. Then, we take the list D as the concatenation of $D_1, D_2, \ldots, D_k$ where k is the number of chromosomes.

3. We compute a histogram of the values in D. We define $h_i \in \mathbb{N}$ as the heights of the bins.

4. We compute the Fast Fourier Transform (FFT) of the histogram (h_i).

For circular chromosomes, as in bacteria, this model can be easily adapted by modifying the definition of D (step 1 above) to be the smallest distance between two positions in the chromosome.

8.1.2 Periodogram

The result is a series of values that measures the importance of a set of sampled periods (which are the moduli of the complex Fourier coefficients), which is called a "periodogram". If we call w the upper bound of the last bin in the histogram (therefore the length of the longest chromosome), and the histogram has bin size 1 (nucleotide-level resolution), then the sampled periods are:

$$w, \frac{w}{2}, \frac{w}{3}, \frac{w}{4}, \frac{w}{5}, \ldots, 2$$

If we want to make the computation faster by reducing resolution, we can make a histogram with larger bins that 1 nucleotide (at step 3 above). For example we can use 10 or 100 nucleotides, which is not a problem if the periods we are looking for are several thousands nucleotides long. If we call r this bin size, then the sampled periods are a smaller subset:

$$w, \frac{w}{2}, \frac{w}{3}, \frac{w}{4}, \frac{w}{5}, \ldots, 2r$$

So for a genome like *S. cerevisiae*, where the longest chromosome is (approximately) 1.5 Mb[1], and a resolution of $r = 10$, there are 75000 sampled periods:

$$1.5 \text{ Mb}, 1 \text{ Mb}, 500 \text{ kb}, 375 \text{ kb}, 300 \text{ kb}, \ldots, 20.00053, 20.00027, 20$$

An important property of this sampling to keep in mind is that period detection is much more precise for small periods than for big ones.

[1] We write "Mb" for 10^6 nucleotides and "kb" for 10^3 nucleotides.

8.1.3 Positions to distances translation

The general idea of the distance calculation step is that positions may be periodically spaced, but the phase may not be perfectly conserved from one end of the chromosome to the other. Consider a set of positions like

$$S_1 = \{10, 20, 30, 40, 105, 115, 125, 135, 213, 223, 233, 243\}$$

It is composed of 3 independent periodical parts with the same period $T = 10$. Directly analyzing S_1 with an FFT will not detect the periodical spacing because the periodical parts are not in phase with each other. In this case, the sorted list of pairwise distances will be like:

$$\begin{aligned} D_1 = {} & (10, 10, 10, 10, 10, 10, 10, 10, 10, 20, 20, 20, 20, 20, 20, 30, 30, 30, \\ & 65, 75, 75, 78, 85, 85, 85, 88, 88 \ldots) \end{aligned}$$

The periodical spacing inside each part is now translated to the numerous occurrences of 10, 20, and 30 at the beginning of the list. Performing an FFT on a histogram of D will detect the period $T = 10$.

A second advantage for using distances instead of absolute positions is that all the D_i lists can be merged together to analyze the complete genome at once. It would not be possible to merge the S_i sets directly. In fact, a common origin would not make sense since positions come from physically separate chromosomes. Analyzing each S_i separately would be possible but would result in a lower statistical power.

8.1.4 Application to recombination hotspots in *S. cerevisiae*

As an example, we show the analysis of the positions of recombination hotspots in *Saccharomyces cerevisiae*, from the experiment in Mancera et al. (2008). This data consists in a set of positions where recombination events have been observed. We applied the complete procedure explained above. The result of the final FFT is shown in Figure 8.1 (left), where a strong peak appears at 15400 nucleotides.

As you can see the range shown is cut on the right at 500 kb. We cut it because there is always an artifactual peak that corresponds to complete size of the histogram, which in this case would be around 1.5 Mb. Note that because of what has been said in subsection 8.1.2, this corresponds in fact to discarding only 2 values in a set of 75000. On the left we have cut the plot at 1 kb, which on the other hand is a choice related to this analysis, because we are interested in periods at the kb scale.

8.2 Statistical analysis

Performing an FFT analysis as explained above is always possible, and always it yields a spectrum with peaks. Intuitively, strong periods will result in high peaks, and this raises the question of what should be considered to be a high peak.

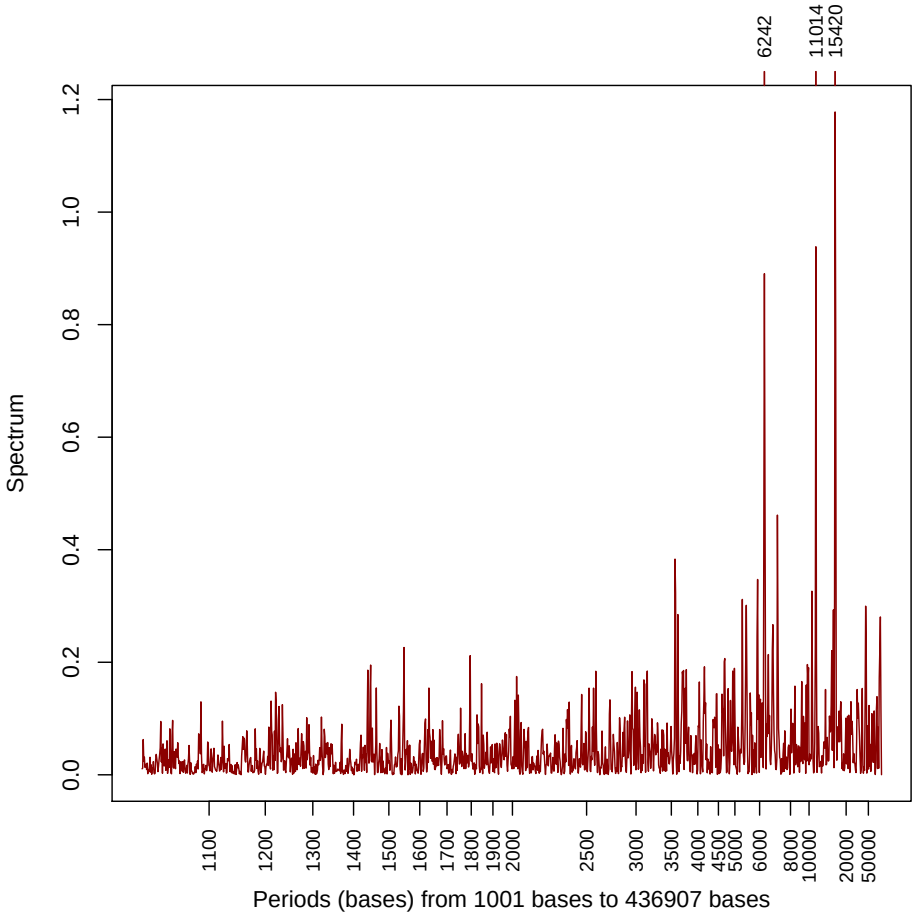

Figure 8.1: **FFT of recombination hotspots in the *Saccharomyces cerevisiae* genome, and distribution of peak heights in random simulations.**
The periodogram produced by the FFT applied to recombination hotspots is shown for periods ranging from 1 kb to 500 kb.

8.2.1 Simulation algorithm

To address the statistical significance of peaks, we perform a random analysis for an interval of periods $[T_{\min}, T_{\max}]$ in which we want to detect the period. We create a random model (which we discuss further) for the data we are analyzing. Then, we compute 10000 instances of this random model and perform the pipeline on each dataset. For each produced periodogram, we record the height of the highest peak, so we get 10000 peak heights. We can then compute the p-value of a peak in the real data analysis by comparing it to this distribution (Figure 8.2). For example, with the 15.4 kb peak in Figure 8.1, we get only 1 peak out of the 10000 simulations that is higher, so we estimate the p-value to 10^{-4}. For the second peak, we apply the same method and find a p-value of 0.0042. These two peaks are significant.

An important practical question when using this statistical analysis is the range $[T_{\min}, T_{\max}]$ to choose. Longer ranges allow to detect more periods; however, the p-value might be less significant.

8.2.2 Random models

Different random models can be used in the simulations:

Uniform model The simplest possibility is to select random positions uniformly on the genome, while preserving the number of positions on each chromosome (i.e. each set S_c has the same size as in the real data). If we use such a model to estimate the p-value of the peaks in Figure 8.1, we get an estimated p-value of $< 10^{-4}$ for both, that is none of the 10000 simulations produces a higher peak.

Gene shuffling model To produce Figure 8.2, we used a more complicated model. Since recombination events are unlikely to have the same rate inside and outside genes, we might suspect that the periodicity we observe is a straightforward consequence of gene and intergenic region lengths. To test this, we created a second random model, called "gene shuffling", that consists in "attaching" each recombination event to the gene or intergenic region where it is located in real data (i.e. we record its relative position to the gene/region), then we randomly reorder the genes and intergenic regions in the chromosomes, while preserving the role of both (i.e. genes are separated by intergenic regions). As mentioned above (see Figure 8.2), this model yields slightly higher p-values than the uniform model, although they remain clearly significant.

Gene selection model The example we have shown here is based on recombination hotspots, which are positions on the genome. However, if instead our method is used to analyze the periodicity of a set of genes, another model also makes sense. As an example, I performed a similar analysis as in Mathelier and Carbone (2010) on the yeast *Candida glabrata*, i.e. I computed the list of codon biased genes for this species using SCCI (Carbone et al., 2003). This results in a high peak for the period 37 kb. The question tested by the gene selection model is whether this set of genes shows a periodicity that would not be expected with a random selection of genes, that is we want to avoid a periodic effect that is general to all gene positions. The random model adapted to this

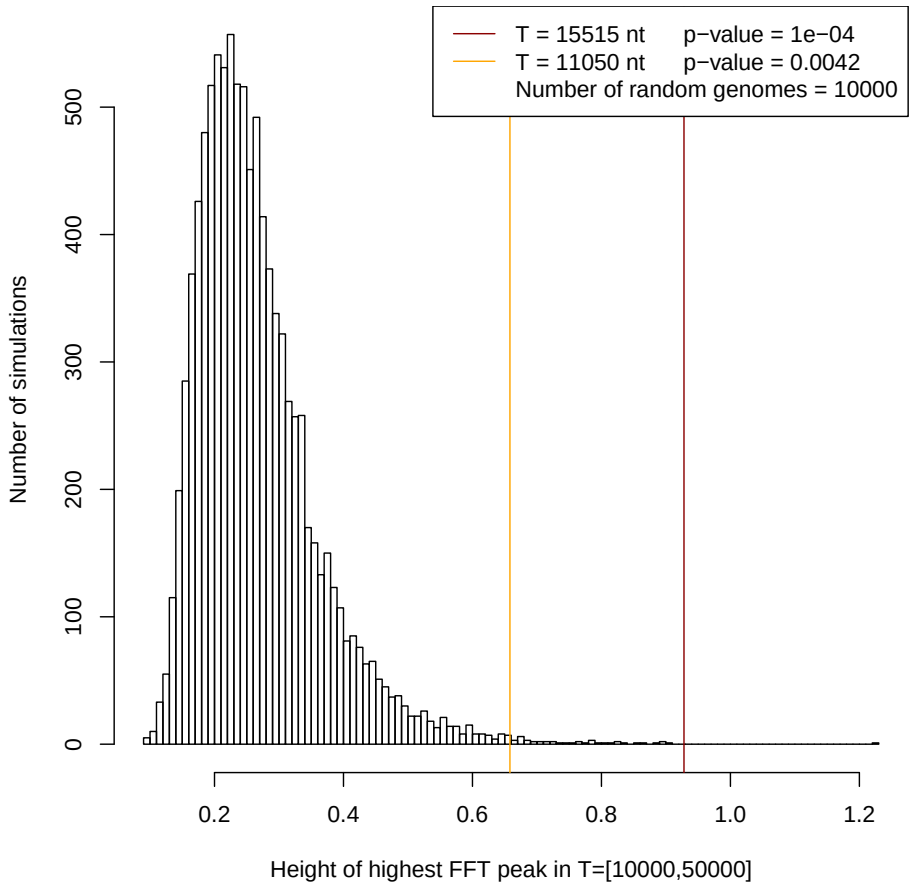

Figure 8.2: **Distribution of peak heights in FFTs in random simulations of recombination hotspots in the *Saccharomyces cerevisiae* genome, with the gene shuffling null model.**
The histogram shows the distribution of the maximum peak height in the 10000 simulations for the 10 to 50 kb interval. The two vertical bars show the position of the first and the second highest peaks in the real data (see Figure 8.1) compared to this distribution. The two strongest periods differ slightly compared to Figure 8.1 because a larger bin width was chosen here ($r = 10$) for faster computation, which is necessary since it is run 10000 times.

situation is therefore a random selection of a subset of n genes, where n is the number of codon biased genes. Using this random model enables us to estimate the p-value for the 37 kb period as $p = 0.08$. The same dataset can also be tested with the gene shuffling model (as done by Mathelier and Carbone (2010) on *E. coli*), in which case we test if biased genes are periodically distributed compared to a random gene order, instead of relabeling genes. In this case, the resulting p-value is $p = 0.11$.

8.3 Further analysis of periods

In order to further understand the signal that leads to the detection of the periodicity, it is helpful to have suitable visualization and statistical tools.

8.3.1 Histogram visualization

When there is such a strong periodicity as shown, in Figure 8.1, the effect might be directly visible on the histogram. To allow for a better visualization, we can instead use density estimation with a Gaussian kernel. The result is shown on Figure 8.3. In this case, the periodicity is directly visible, since we can clearly see peaks that match most of the time multiples of 15.4 kb (mostly on the leftmost part). The individual contribution from each chromosome can be seen using different colors.

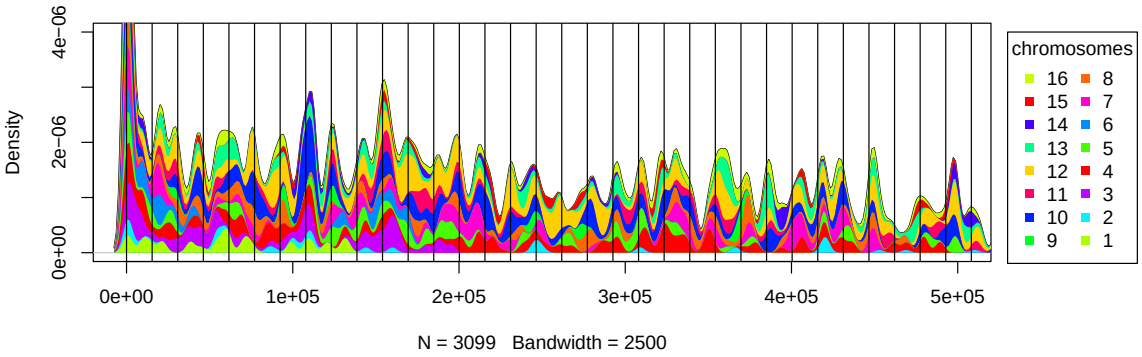

Figure 8.3: **Distribution of distances between recombination hotspots.**
The contribution of each list $D_1 \dots D_{16}$ of distances between hotspots lying in yeast chromosomes, is colored differently. The density is estimated with a Gaussian kernel of $\sigma = 2500$. Vertical lines have been drawn at multiples of 15400 nucleotides to highlight the strong periodicity of the hotspot distribution within this period. Once drawn, it can be clearly identified by eyes. The number of distances considered in the plot, over all chromosomes, is $N = 3099$.

8.3.2 Modulo projections

A useful visualization tool, is the one that projects the positions along the chromosome into an interval of length T, where T is one of the detected periods. This projection is defined by the function:

$$m_T(x) = x \bmod T \qquad \text{where} \quad x \bmod T = x - T \left\lfloor \frac{x}{T} \right\rfloor \in [0, T[$$

and can be visualized through a "coil-like" representation of the chromosome, once divided into slices of length T. Figure 8.4 shows the full chromosomal length organized in successive intervals of length T, drawn from left to right, where the y-axis indicates $m_T(x)$ values. With this representation it is easy to see whether hotspots are separated by lengths that are multiples of T, as, in this case, they would be horizontally aligned.

The sliced chromosome representation (Figure 8.4) highlights that all $m_T(x)$ values for chromosome 10 are located in an interval of length $T/2$.

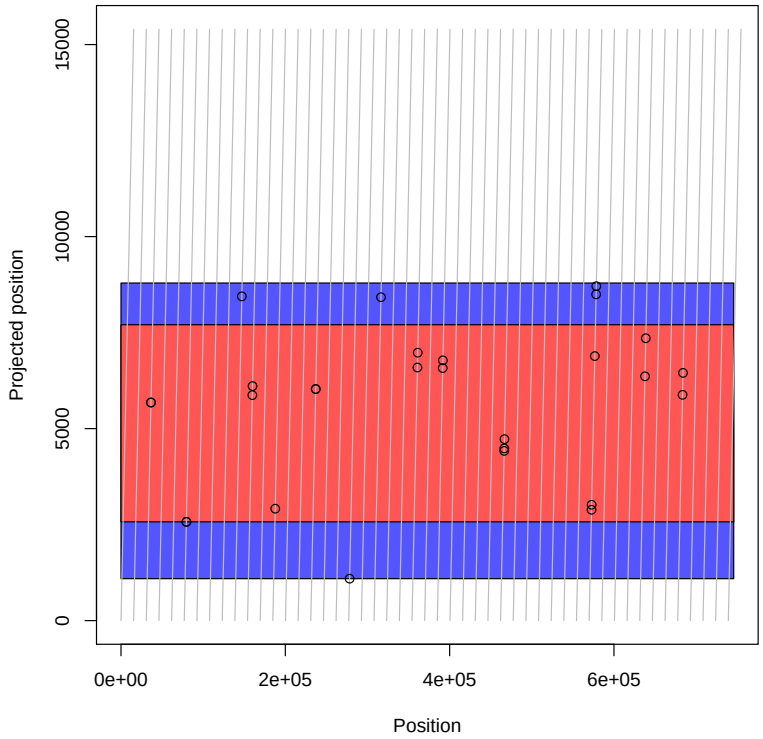

Figure 8.4: **Hotspots projected modulo 15400 nucleotides, on chromosome 10.** Chromosome 10 has been sliced in successive intervals of 15400 nucleotides each, arranged from left to right (gray vertical lines), in order to show the alignment of hotspot positions (circles). The vertical axis corresponds to the projected position, which is the remainder of the division of the chromosomal position by T. The colored area (red+blue) is the window with a size $T/2$ where all hotspots are located. The red part is the windows of size $T/3$ where the highest number of hotspots is located.

8.3.3 Statistical analysis of modulo projections

Although plots like Figure 8.4 give an intuitive idea about whether points are positioned periodically, it is necessary to test for the statistical significance. In other words, we want to check whether the localization of $m_T(x)$ values in a window $[a, b]$ of length $w = T/2$ or $w = T/3$ is statistically significant.

For that purpose, we assume that the null model is a uniform distribution of $m_T(x)$ in $[0, T[$. If the window $[a, b]$ was fixed before knowing the period, the distribution of the number of hotspots in the window would simply be a binomial distribution with $p = w/T$.

Table 8.1: **Statistical analysis of positions along the chromosomes modulo 15400.** For each chromosome, the window of size $w = T/3$, corresponding to 5133 nucleotides, containing most hotspot positions has been selected. The ends of the windows are given in the "from" and "to" columns. Column "inside" gives the number of points in the window while "outside" gives the number of points outside the window. The "percent" column gives the "inside" value as a percentage. Its p-value compared to our null model is given. The stars highlight the significance (1 star = 5%, 2 stars = 1%, 3 stars = 0.1%).

chr	from	to	inside	outside	total	percent	p-value	signif.
1	6197	11330	7	5	12	58	0.6438	
2	8223	13356	7	5	12	58	0.6438	
3	10987	720	10	7	17	59	0.3692	
4	44	5177	17	14	31	55	0.2182	
5	9757	14890	13	5	18	72	0.0200	*
6	8337	13470	7	5	12	58	0.6438	
7	7845	12978	19	5	24	79	0.0001	***
8	2647	7780	14	5	19	74	0.0100	**
9	6289	11422	10	0	10	100	0.0006	***
10	2572	7705	23	5	28	82	0.0000	***
11	6734	11867	11	7	18	61	0.2341	
12	4816	9949	21	13	34	62	0.0198	*
13	7143	12276	13	10	23	57	0.2965	
14	12911	2644	8	4	12	67	0.2648	
15	8542	13675	7	7	14	50	0.9473	
16	8768	13901	14	4	18	78	0.0040	**

However, this assumption would underestimate p-values, because in fact, we select the window $[a, b]$ of length w by maximizing the number of points in it.

Therefore, the best way to estimate the H_0 distribution is to use a simulation. For this, we randomly place the positions in $[0, T[$ with a uniform law, then we select the best interval $[a, b]$, and count the number of points in it (our statistic). We repeat this operation 10000 times and estimate the p-value of our observed value with this distribution. The p-value computed for *S. cerevisiae* on chromosome 10 is $< 10^{-4}$. Table 8.1 shows the result of the analysis for all *S. cerevisiae* chromosomes with a window of size $w = T/3$.

8.3.4 Comparison with Solenoid Coordinate Method

We also compared our results on *S. cerevisiae* recombination hotspots with those obtained with the SCM algorithm (Junier et al., 2010). This algorithm is based on a similar idea as our modulo projections, except that it does this projection as its first step and not as a post-analysis like us. SCM stands for "Solenoid Coordinate Method", where "solenoid coordinate" refers to the same thing as our modulo coordinates. This algorithm does the modulo projections for all possible periods, and has then a way to assess how non-uniform the distribution of projected positions is.

We did not use this method in the first place because it works with the positions directly instead of distance between pairs. This implies that the analysis has to be done separately on each chromosome, and that the phase of the periodicity is synchronized for each chromosome (see section 8.1.3). However, since after this analysis we found that the signal comes from a few chromosomes, and the phase is actually synchronized (the points are horizontally aligned in Figure 8.4), these limitations do no apply here.

Therefore, I applied the SCM algorithm to the recombination hotspots on each chromosome, with the results shown in Figure 8.5. As can be seen, the strongest signal (lowest p-value) of the whole analysis is for period $T = 15418$ on chromosome 10, which corresponds to the period we found. However, we also found it on other chromosomes, while SCM did not highlight it.

Compared to SCM, a major advantage of our methodology is its ability to use the complete genome at once, thanks to the use of pairwise distances. On the other hand, SCM can only work with each chromosome separately. Taking all chromosomes in a single analysis allows to gain more statistical power, which is necessary in many applications since the data may contains a limited number of points.

8.3.5 Going back to the raw data

We might want to apply the same modulo projection to the raw data that was used to produce the initial sets of hotspots, that is, the individual recombination events observed in Mancera et al. (2008). The result can be seen in Figure 8.6A. I also wanted to correlate this information with the double-strand break density, since double-strand breaks are the initial cause of recombination events. So I applied the same modulo projection to the data from Pan et al. (2011), which is a high-resolution mapping of double-strand breaks over the whole *S. cerevisiae* genome. The result can be seen on Figure 8.6B. Both histograms show a clearly non-uniform distribution.

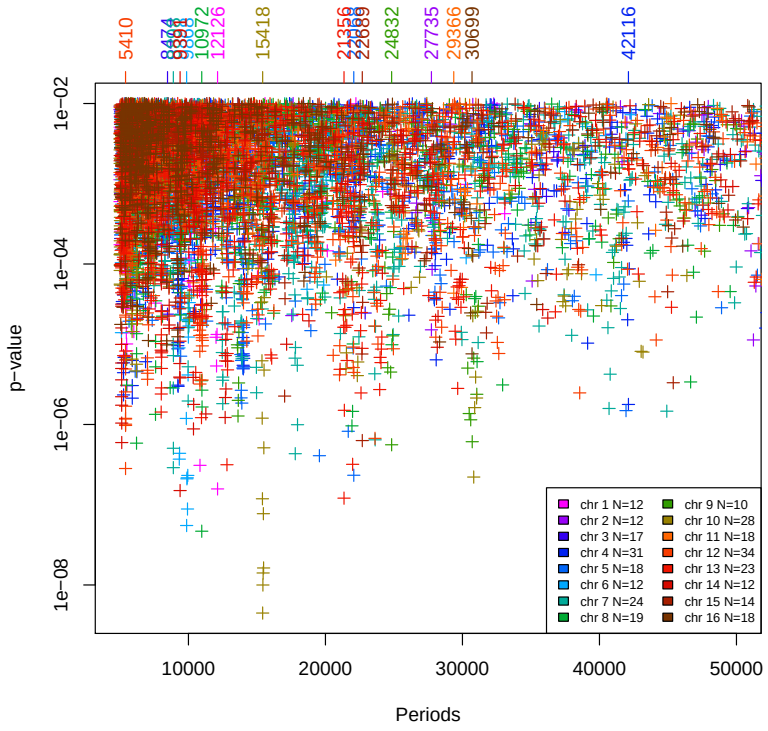

Figure 8.5: **Analysis of recombination hotspots performed with SCM**
For each tested period between 5000 and 50000 nucleotides, the p-value computed by SCM is shown (in logarithmic scale). Only points with p-values below 0.01 are shown. Each color is a different chromosome. The strongest peak is at 15418 nucleotides on chromosome 10. In the legend, *N* refers to the number of hotspots in each chromosome.

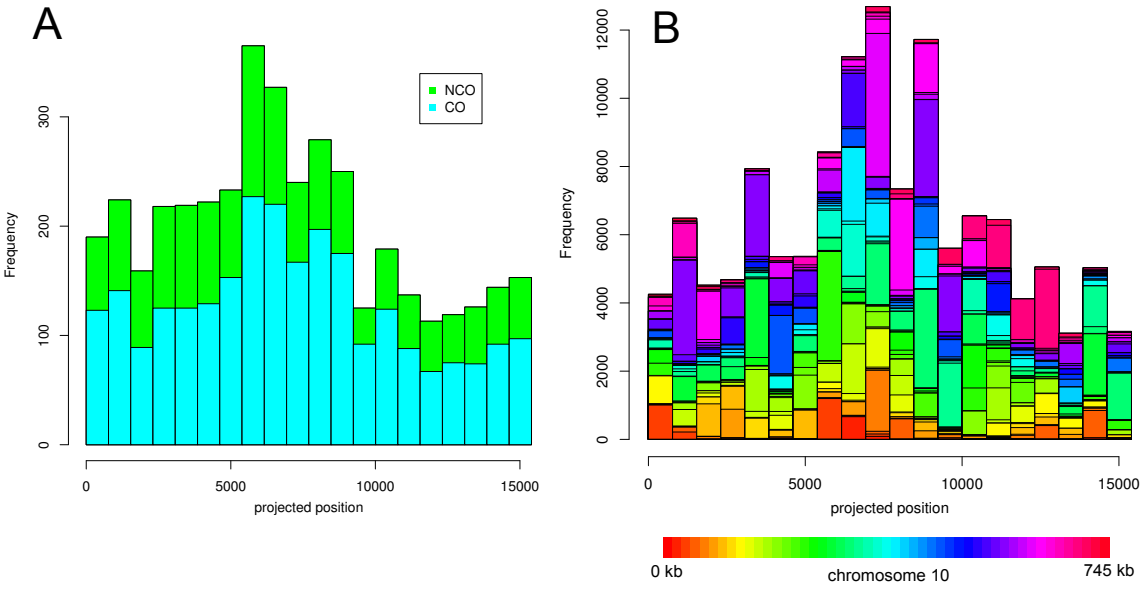

Figure 8.6: **Histogram of projected positions of recombination events (A) and double-strand breaks (B) along chromosome 10.** On figure A the histogram is split in two colors according to the type of recombination events. On figure B the colors represent the 15.4 kb slice of the genome that is considered (that is the integer part of the position divided by 15400).

Although it seems clear that we have made a statistically significant observation, the biological explanation for the 15.4 kb period is at this point still to be found. To better understand the data, I have developed a local periodicity analysis methodology, which we will explain now.

8.4 Local periodicity analysis in high-resolution data

After having discovered the periods with the previous method, we wanted to investigate its cause. We therefore tried to analyze experimental data from double-strand breaks (DSB), which are the initial cause of the recombination events we have presented. Among the possible reasons for the periodic signal, we considered the DSB distribution itself, but also the distribution of other DNA-binding proteins that are involved in recombination after a double-strand break. The DSB experimental data comes from the ChIP-seq experiment by Pan et al. (2011), while the accessory protein data comes from ChIP-on-chip experiment Panizza et al. (2011). These new data sets provide a high-precision measurement of the density of respectively DSB and Spo11-accessory proteins.

Analyzing them globally did not yield any interesting result, so I created a new program to analyze periodicity in sliding windows and to extract only the relevant ones. The idea to compute an FFT on a sliding window is not new (Harris, 1978), however, it has not been used before to detect local periodic signals on genomes.

8.4.1 General methodology

Let us assume to have a set of functions $f_c(x)$ representing the density we want to analyze at position x of chromosome c. As an example, we consider data describing the Red1 protein localization along the *S. cerevisiae* chromosomes, which have been produced by a ChIP-on-chip experiment Panizza et al. (2011). Here $f_c(x)$ represents an estimation of the density of Red1 proteins binding at position x of chromosome c. This density is estimated along the whole genome at a fixed interval (here every 10 nucleotides).

The procedure identifying a periodic distribution is the following:

1. Start at the beginning of the chromosome and select a window $[a, b]$ of range w.

2. Consider the series of w values of the function $f_c(x)$ for $x \in [a, b]$.

3. Add $100 \cdot w$ zeros at the end of the series. Some more zeros are then added until the new length N is a highly composite number.

4. Perform the FFT of this series of N values. All periods of the form N/k where k goes from 1 to $N/2$ are sampled. For each of these periods p we have a value V_p which is the modulus of the Fourier coefficient.

5. Find the period T with the highest spectral value in the periodogram computed with the FFT. Periods higher than $w/2$ are ignored, because it is not reasonable to detect a periodicity that does not have at least two full periods in the input signal.

6. Record the score of that period, defined like this:

$$S_T = \frac{V_T}{\sqrt{\sum_{k=1}^{N/2} V_{N/k}^2}}$$

7. Move the window by $w/10$ nucleotides and go back to step 2 unless the end of the chromosome is reached.

8.4.2 Zero-padding

The purpose of the zero-padding in step 3 is to detect periods with high precision. This procedure is well known in Discrete Fourier Transform, where one can zero-pad to make something a power-of-2, to make circular transform behave like non-circular transform, to re-sample a signal. Here we use it to change frequency resolution. Intuitively, zero padding allows one to use a longer FFT, with more frequency bins that are more closely spaced in frequency, which will produce a longer FFT result vector.

Since sampled periods are always of the form N/k, where k is an integer and N is the series length, it means that we cannot reliably detect periods that are not exact divisors of the series length N. If we take the values in the window alone, there we have $N = w$, so only periods that are divisors of the window length are sampled. This results in a very high imprecision for periods that are not small compared to the window length. To avoid this problem, we pad the series with zeros, resulting in a much bigger length N. This way of doing results in considering extra periods being sampled. Some of them are larger than w and they are not interesting, but the remaining ones are in the range $[0, w]$, as we looked for.

For example, with the Red1 data, the window length we use is $w = 100000$. Without padding, sampled periods would be:

$$100000, 50000, 33333, 25000, 20000, \ldots$$

If the true period to detect is 23000, we will either detect 25000 or 20000, but nothing more precise than this. With padding, the new length is $w + 100 \cdot w$, so we sample:

$$\ldots, 100000, 99020, 98058, 97115, \ldots, 23059, 23007, 22955, \ldots$$

and we get a suitable estimation of 23007 ± 26 nt.

8.4.3 Application to recombination proteins in *S. cerevisiae*

To analyze the *S. cerevisiae* recombination proteins data, I chose to use a window of 100 kb, because it is the right order of magnitude for a period around 15 kb, as it would mean the signal would contain around 6 periods, which allows for a reliable detection. Setting the window size higher, to 500 kb for example, would lead to more than 30 successive periods, which is a too strong requirement. On the other hand, a too small window size ($< 2 \times 15$ kb) would miss the periodicity.

8.4.4 Scoring periodicity signals

In order to distinguish the parts of the genome where the curve is showing a strong periodicity from those where there is none, I have defined a score, which we compute after the FFT on each window for the best period T found in that window:

$$S_T = \frac{V_T}{\sqrt{\sum_{k=1}^{N/2} V_{N/k}^2}}$$

We divide it by the total signal, so that the resulting score is independent from the input signal size. This means that this score is purely a measure of how much a sinus with period T fits the original input data.

8.4.5 Visualization

In order to present the results, we compute the maximum score M over all windows, and divide all scores by M. We then classify as a "strong" periodicity signal any score that is higher than $S_T > 0.75M$, a "weak" signal if the score $S_T > 0.5M$, and a "no" signal otherwise. We use this information to produce a plot like Figure 8.7, where the strong periodicity signals are marked by red triangles.

It can also be helpful to look back at the original data and see how the period "fits" the data. To that end, I have added a post-processing step that creates a plot for parts of the genome where periodicity has been found. On each window where a strong periodicity signal was detected, we draw a sine with a period equal to the period found. Its amplitude is given by the modulus of the Fourier coefficient, its mean is given by the mean of the signal and its phase is given by the argument of the Fourier coefficient. An example of result can be seen on figure 8.8.

8.4.6 Results for *S. cerevisiae* recombination data

The results of the periodicity analysis are moderately convincing. There are a few spots where a strong periodicity has been found, but the period is variable, typically in the 10 to 30 kb range. It is not obvious to give a biological explanation for this phenomenon. Indeed, we later created a model that explains the patterns (see chapter 9), and the periodicity found for DSB hotspots is in fact due to gene positions.

8.5 Application to *Phaeodactylum tricornotum* RNAs

Although it was not the original question that led me to design the local periodicity analysis tool, the biologists from the diatom genomics team in our lab became interested in this technology. They are studying the small RNAs of *Phaeodactylum tricornotum* (see section 7.2). To that end, they have sequenced these RNAs using Illumina technology, so we have the precise genome-wide coverage of these small RNAs. The biologists had already manually observed than these small RNAs have a periodical distribution in highly

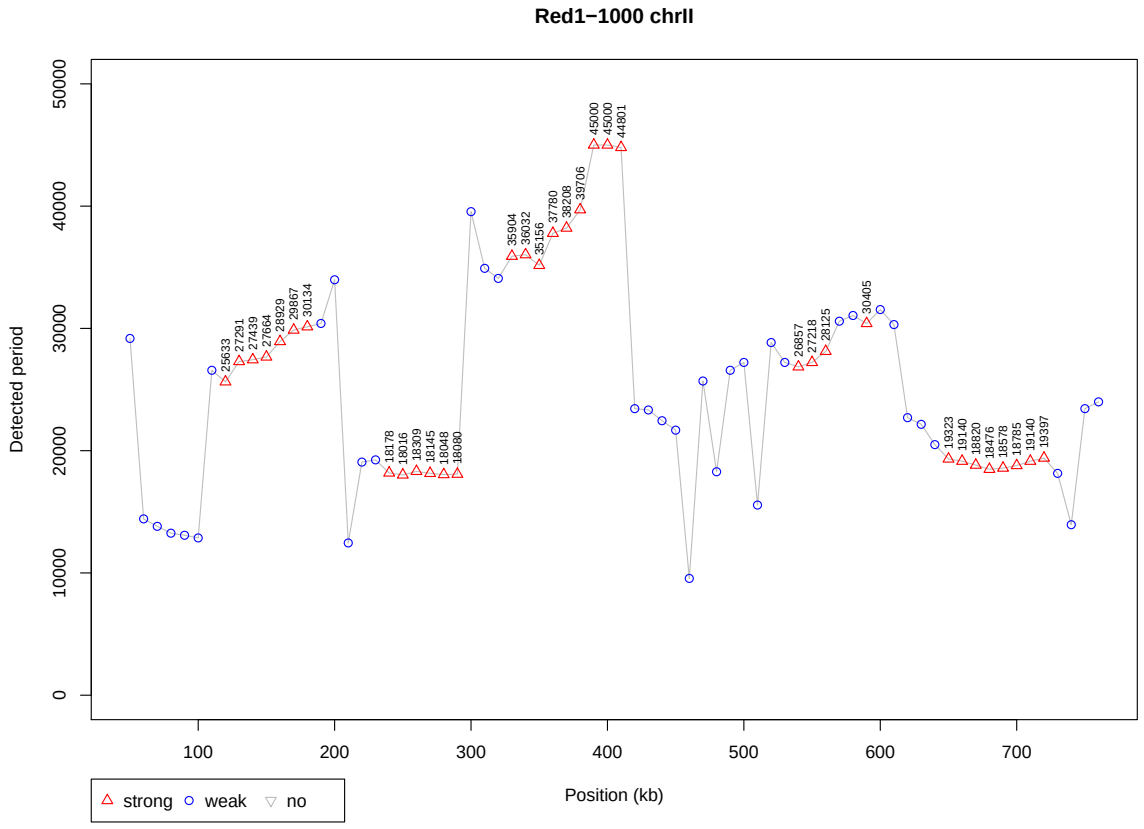

Figure 8.7: **Best local periods detected in the Red1 protein binding density with a sliding window on *S. cerevisiae* chromosome 2.**
Strong and weak periodicity signals are marked by red triangles and blue circles, respectively.

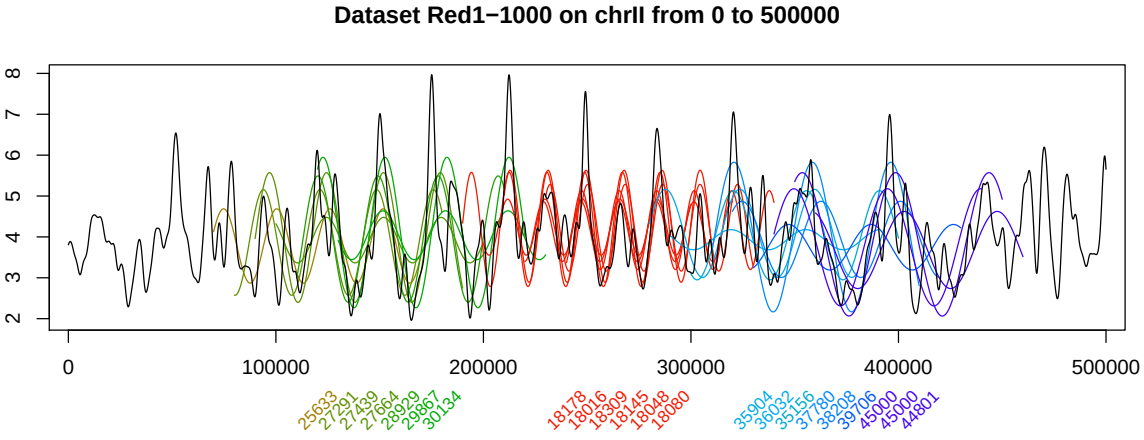

Figure 8.8: **Sines fitted (colored lines) on the original Red1 binding data (black) for each period detected.**
Each number in the bottom margin is a significant period detected in the window centered at its position. For each one, a sine has been drawn on the detection window, to better visualize the period detected. Different colors correspond to different periods.

methylated regions, because in some cases the periodicity is so clear that it can be directly seen by eye (as in Figure 8.9A).

However, they were interested in a genome-wide automatic detection of periodicity, because it would allow to quantify the importance of the phenomenon and to see exactly how long these periods are. Moreover, sometimes the periodical part is not sharply isolated as in Figure 8.9A, which is an extreme case, but the signal is rather like in Figure 8.9B. I could therefore apply my methodology without any change to this new question. The manually observed periods were around 180 nucleotides, so I chose a window of 1 kb to run the analysis.

Most of the genome does not display any periodicity (89%), but a small part (1%) shows a strong periodicity. Unlike the Red1 data, the periods occur in a very specific range, as 70% of them are in the $[170, 230]$ range. The distribution of these periods is shown on Figure 8.10. A probable explanation is the size of nucleosomes, since it is known that they are spaced by 180 nt, which is close to our range of periods. This work has been published in an article in BMC Genomics (Rogato et al., 2014).

8.6 Technical details

FFT implementation I have written all the scripts using the R programming language (R Development Core Team, 2011). The Fourier transform implementation comes from the `spec.pgram` function included in R, which includes by default zero-padding to a high composite number. The extra zero-padding discussed in section 8.4.2 consists simply in setting the argument `pad` to 100. For the rest, the default settings of the function are used.

Experimental data for *S. cerevisiae* The list of recombination hotspots comes from Mancera et al. (2008), while the double-strand break density comes from Pan et al. (2011). The Red1 ChIP-on-chip data is from Panizza et al. (2011) and was smoothed with a Gaussian kernel with standard deviation $\sigma = 1$ kb before analysis.

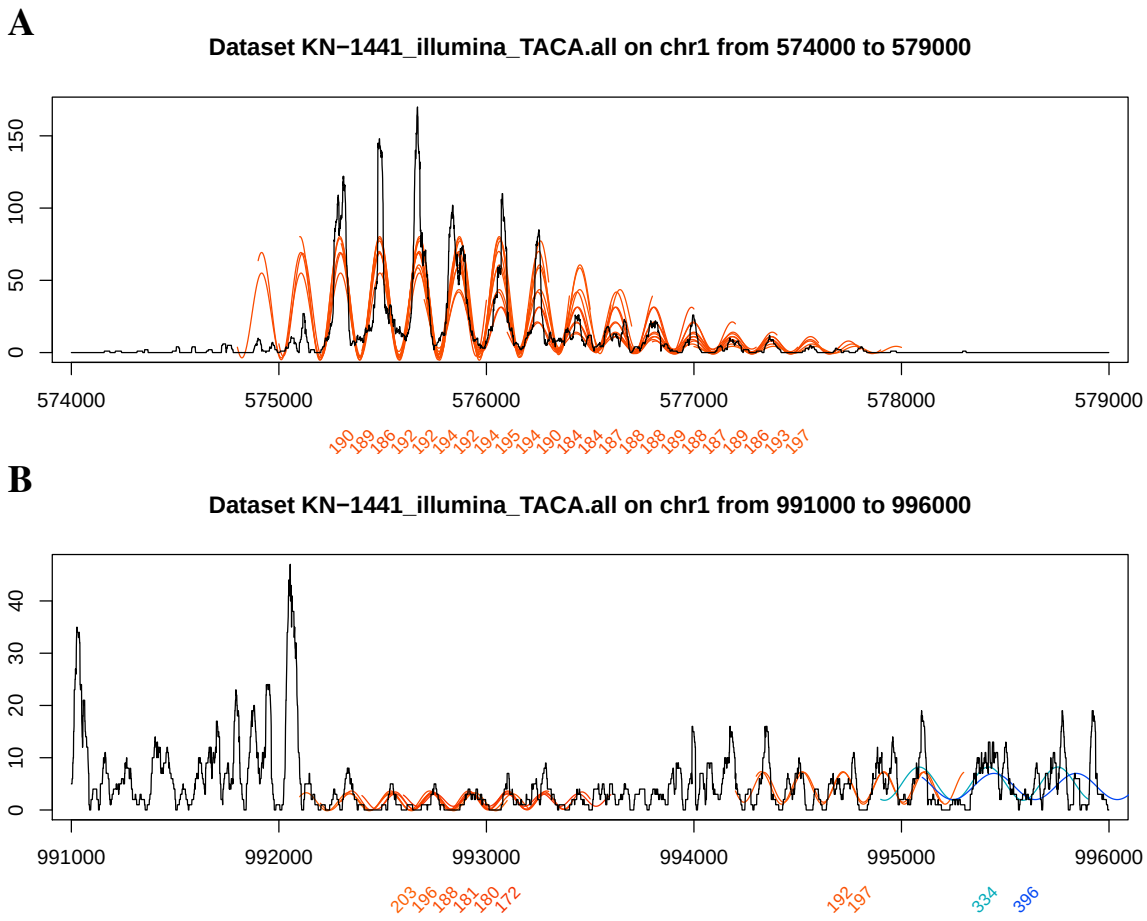

Figure 8.9: **Coverage of small RNAs on *Phaeodactylum tricornotum* (black) with detected periods using Fourier transforms (colors).** The isolated 334 and 396 periods are probably artefactual, while the periods in the 180-203 range show a reproducible signal.

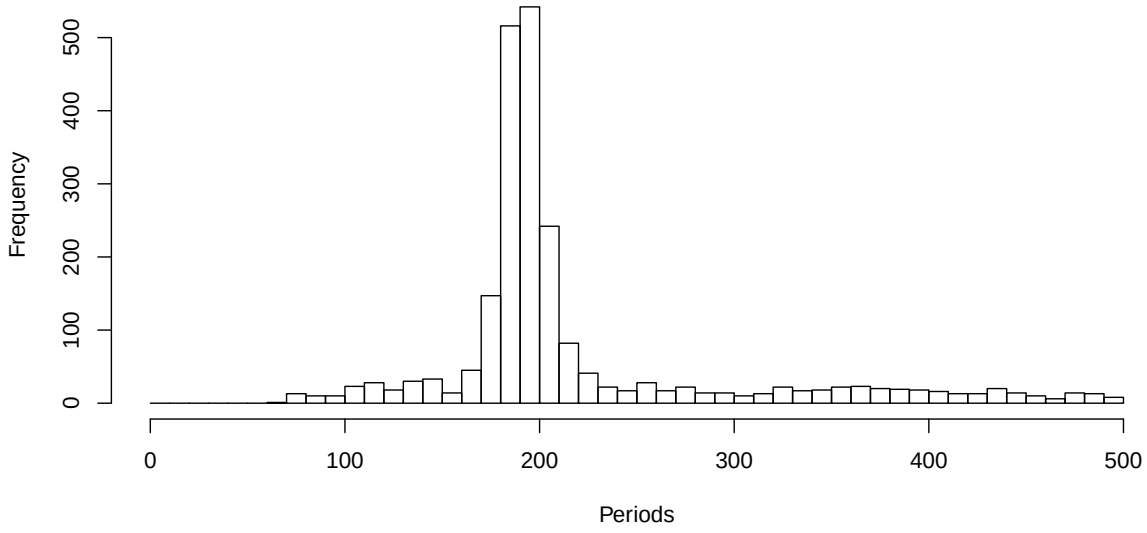

Figure 8.10: **Distribution of periods found on *Phaeodactylum tricornotum* small RNAs.** Only the strong periodicity signals have been considered.

Chapter 9

A model for yeast recombination proteins

Contents

Here I present a model we have created for the distribution of double-strand breaks caused by Spo11 during meiosis and axis proteins along the genome. See section 7.1.2 for description of these biological observations. This work has been submitted to a journal[1] and is under review.

9.1 The SPoRE model and the algorithm

SPoRE modeling of axis proteins and DSBs relies on a general principle that can be summarized in two main steps (Figure 9.1). First, it defines a set of positions on the

[1] Champeimont, R., and Carbone, A. SPoRE: a mathematical model to predict double strand breaks and axis protein sites in meiosis

genome where proteins might accumulate, and sets a weight for each of these positions according to gene annotation. This weight is used as an indicator of the density of the proteins. In the second step, it makes a smooth curve using a Gaussian kernel of the distribution of weights along the genome.

This main computational core in SPoRE, takes as input a genome and its gene annotation, and provides as output the modeling curves describing DSBs and axis proteins distribution along the whole genome (Figure 9.3). A list of Transcription Factor Binding Sites (TFBS) can be provided as input for a more accurate promoter region detection. This intermediate output is used by SPoRE to provide four kinds of data:

1. It produces the curves modeling the density of DSBs and axis proteins along the whole genome, in a format that is ready for browsing (Figure 9.2).

2. Given a list of intervals on the genome, it predicts whether they are hot or cold spots for DSBs.

3. Given a list of intervals on the genome, it predicts whether they are axis sites.

4. Given an experimental curve defined over the genome, it compares the DSB and axis proteins model curves with experimental data and provides Pearson and Spearman local correlation coefficients between them.

Also, it compares the peaks of the model curves with the peaks of the experimental curve, computing PPV and sensitivity.

SPoRE can easily be used by giving as input a genome and its associated gene annotation. All parameters in SPoRE are automatically computed on the input genome. Also, SPoRE works on scaffolds, not only on fully assembled chromosomes, since its minimal requirement is ORF annotation.

9.1.1 Analysis of convergent and divergent regions

Our intuition on the positioning of high density hotspots for axis proteins and DSBs was developed with the analysis of the *S. cerevisiae* experimental data in (Panizza et al., 2011; Pan et al., 2011). In understanding these data, we focused on convergent and divergent regions, instead of considering the start and the end of genes as previously done (compare Figure 9.4 to Figure 9.5). The plots, reported in Figure 9.4A-F, highlight characteristics of the data when displayed for convergent and divergent intergenic regions. Notice that in (Glynn et al., 2004), it was already observed that meiotic cohesin preferably accumulates in convergent regions (Figure 9.4A), with an extreme bias against regions in which transcription is diverging (Figure 9.4B).

By focusing on convergent and divergent regions, we observe (and provide with that a precise numerical evaluation) that:

1. the local negative correlation between Red1 and DSBs localizations observed in (Glynn et al., 2004; Panizza et al., 2011) physically corresponds to convergent and divergent regions, where convergent regions present high average Red1 density and almost no presence of DSBs (Figure 9.4A and D), while divergent regions present a high average Spo11 density and an important decrease in Red1 (Figure 9.4B and E);

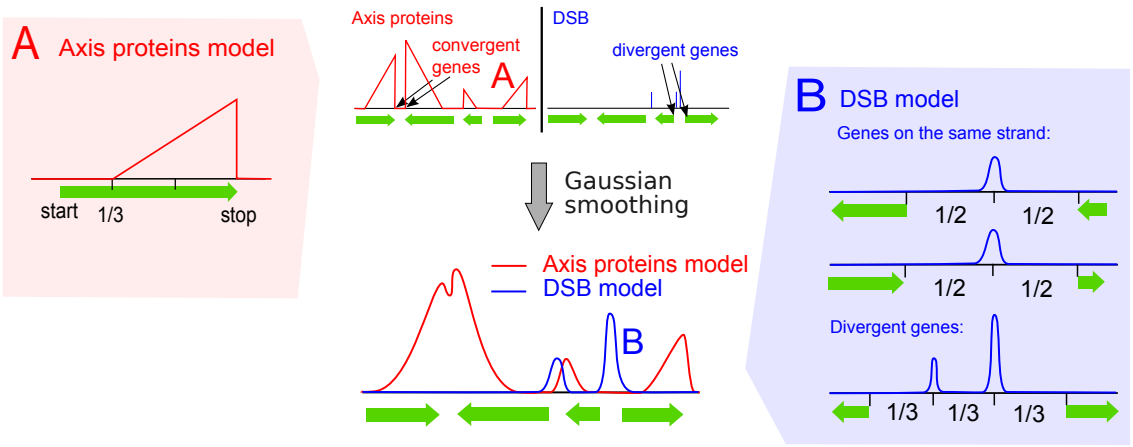

Figure 9.1: **Summary of SPoRE modeling approach.** The approach is constituted by two main steps, described from top to bottom (center). First, SPoRE considers a set of positions to which it assigns weights. Axis proteins (red) and DSBs (blue) involve convergent genes and divergent genes, respectively. In the drawing, locations with non-zero weight are indicated by colored vertical bars (height represents importance) and triangles: convergent genes for axis proteins and divergent regions for DSBs display the highest weights (top). Then, SPoRE smooths the distribution of weights with a Gaussian kernel (bottom) modeling, in this way, the diffusion of the proteins around their main sites. The red box on the left (axis proteins) and the blue box on the right (DSB) describe some details of SPoRE models.

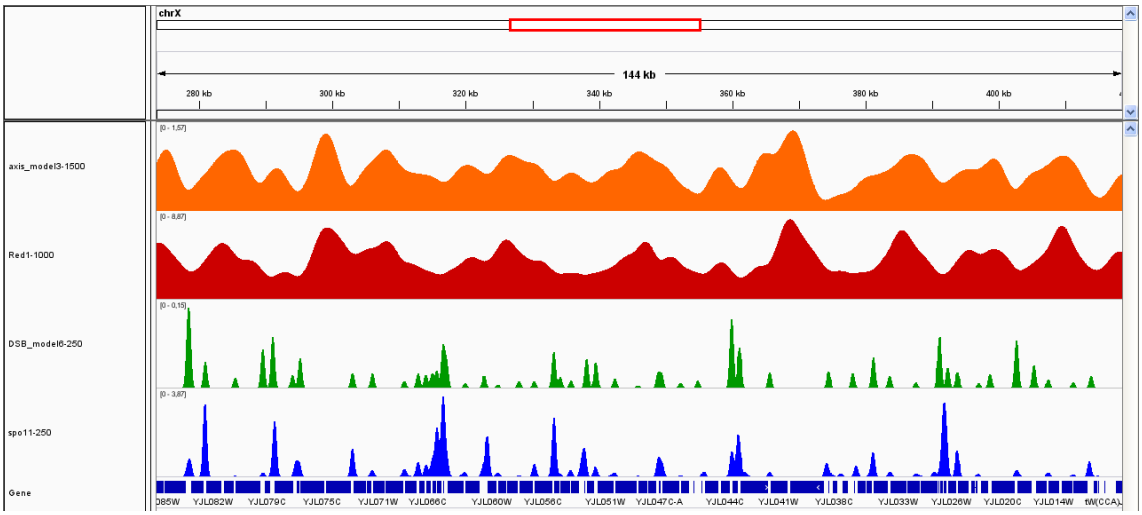

Figure 9.2: **Screenshot of IGV (Thorvaldsdóttir et al., 2013).** Screenshot obtained after loading SPoRE modeling curve for axis proteins (orange) and DSBs (green) and the corresponding experimental data for Red1 (red) and Spo11 (blue). Gene location is reported on the bottom of the page (dark blue rectangles).

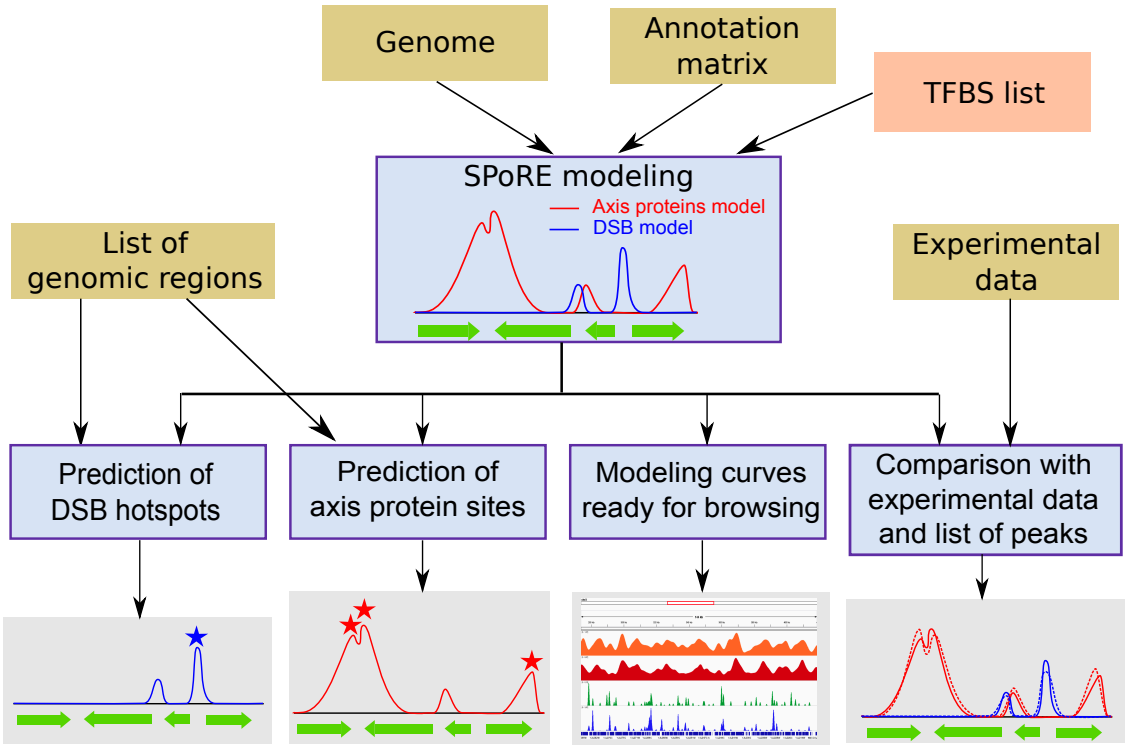

Figure 9.3: **SPoRE flowchart.** SPoRE takes several input files (brown boxes); the input in the orange box is optional. SPoRE implements the construction of the modeling curves for axis proteins and DSBs, as described in Figure 9.1 (blue box, top), and uses these curves as input for 4 algorithmic tasks (bottom blue boxes; outputs in grey boxes): 1. to predict DSB hotspots. Starting from a list of genomic regions, it decides whether these regions are susceptible to DSB or not; 2. to predict axis proteins sites. As in 1, it makes predictions starting from a list of genomic regions provided by the user; 3. to produce ready for browsing output files describing the axis proteins and the DSB modeling curves (see Figure 9.2); 4. to compare SPoRE models (solid line) and experimental data (dashed lines).

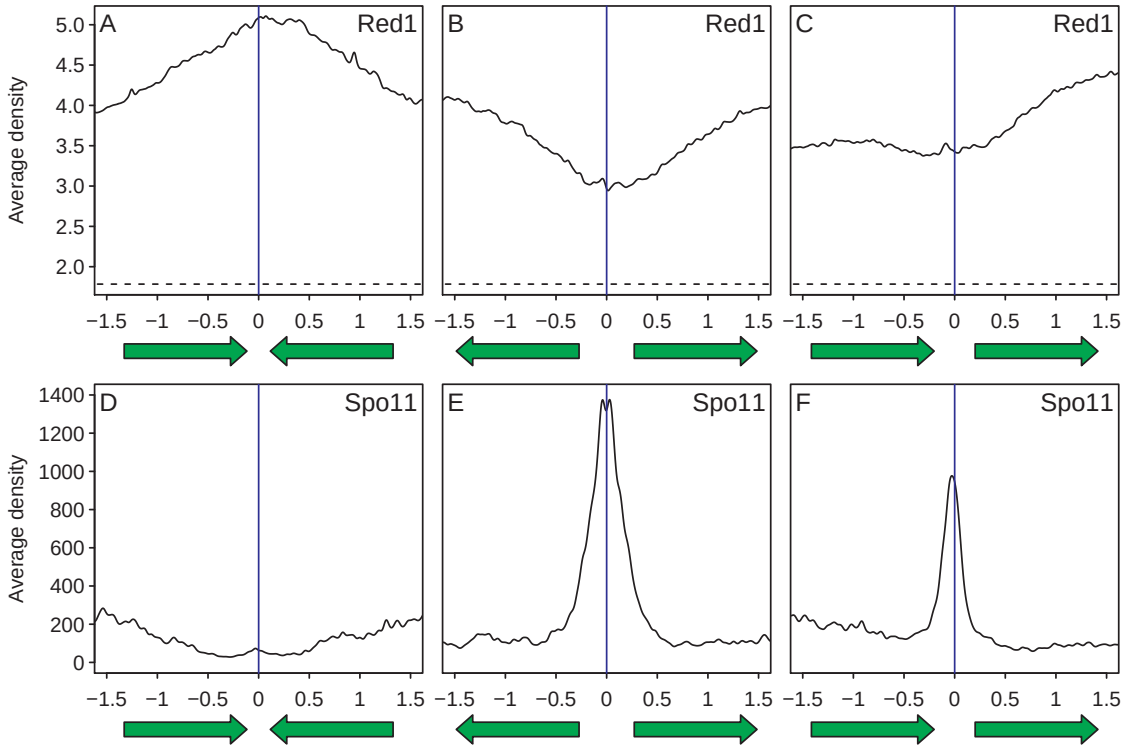

Figure 9.4: Proteins density in convergent and divergent intergenic regions of the *S. cerevisiae* genome. Average Red1 density (Panizza et al., 2011) in a 3 kb window around intergenic region centers, for regions with convergent genes (A), divergent genes (B), or oriented genes (C). The scale is defined as microarray intensity. The dotted line shows the "base noise level" corresponding to the first decile (0.85) in the intensity distribution, considered as "no or low signal" in (Panizza et al., 2011). Plots D, E and F show the same distribution for the Spo11 protein, based on (Pan et al., 2011) experimental data, measured in reads per kb. Genes drawn below plots show the median gene length (1212 nt) and the median intergenic region length for the type of region shown (231, 545, and 410 nt for convergent, divergent, and oriented regions, respectively).

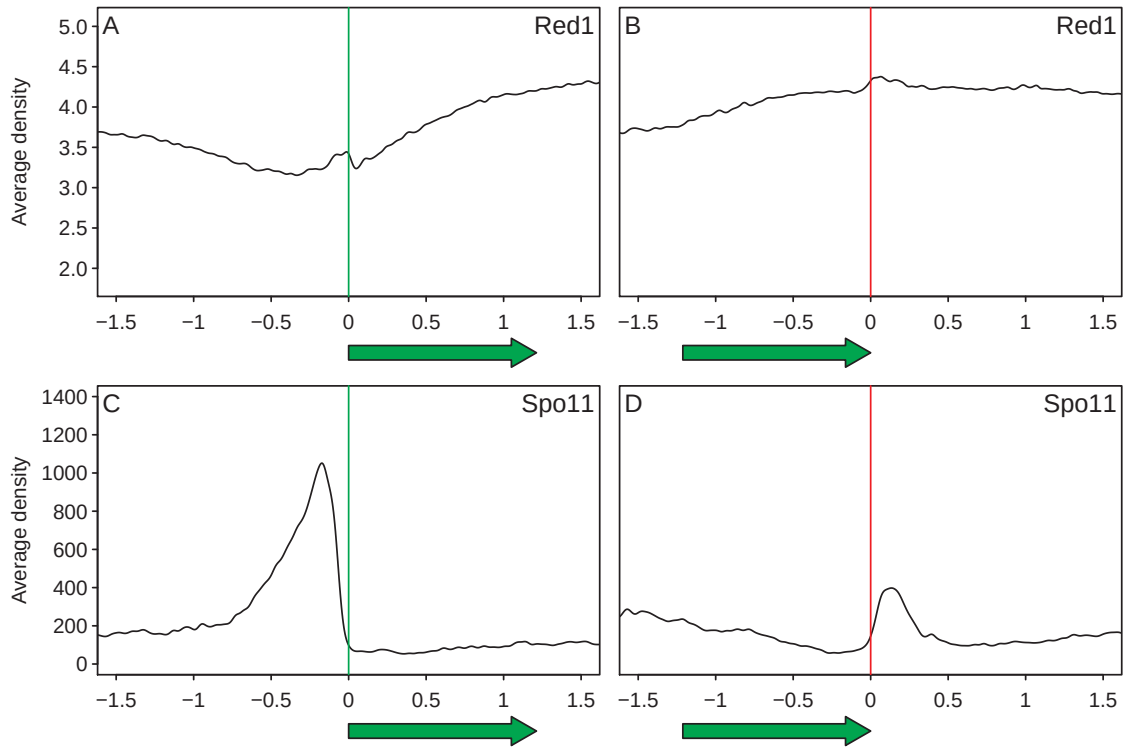

Figure 9.5: Average Red1 density (Panizza et al., 2011) in a 3 kb window centered at gene 5' ends (A) and gene 3' ends (B). The zero is respectively the gene start and stop. Figure C shows the average Spo11 density at gene 5' ends observed in (Pan et al., 2011), while figure D shows it at gene 3' ends.

2. Red1 density is much higher at gene 3' ends than it is at gene 5' ends, and yet even higher when we consider only convergent intergenic regions, having two gene 3' ends (Figure 9.4A and C);

3. DSB density is twice as high in divergent regions, having two gene starts, than in *oriented regions*, that is intergenic regions between co-directional genes (Figure 9.4E-F);

4. DSB peaks are localized in promoter regions. This is shown by DSB distribution in large divergent regions. For the vast majority of intergenic regions (of $< 800nt$ in length), the DSB peaks appear roughly centered in the middle of divergent regions (Figure 9.4E-F), this position well approximating promoter locations.

5. From Panizza et al. (2011) data, we also observed that the shape of the distribution of Red1 proteins along genes (Figure 9.6A), highlights a linear increase of the amount of Red1 proteins towards the gene end. On single genes this increasing distribution is not sharply distinguishable but when considering all genes together, it becomes gradually more pronounced in longer genes. In particular, the area under the distribution curves augments proportionally to gene length.

9.1.2 Axis proteins model

In a first attempt, axis proteins could be modeled by using gene 3' ends as reference positions and by associating to each position a weight corresponding to the length of the relative gene. This simple model implies that convergent regions are governed by weights defined as the "sum" of two gene lengths, that oriented regions are modeled by the length of only one gene, and that divergent regions are ignored. It captures well some characteristics observed in *S. cerevisiae* experimental data: convergent regions host about the double amount of Red1 compared to oriented regions (relatively to "base noise level", see Figure 9.4A and Figure 9.4C) and the amount of Red1 at gene 3' ends augments with gene length.

SPoRE is based on this simple model but it also describes, in an explicit way, the spread of Red1 proteins along the gene. This Red1 spreading is likely due to two processes, one of diffusion and one of convection of proteins. Since experimental measures of diffusion constants produced highly varying values depending on the organism and on the protein (Tkačik and Bialek, 2009), and that measures of convection constants are also organism and gene dependent (Pérez-Ortín et al., 2007), we cannot directly use them to model the curves in Figure 9.6A. Then, we discretely approximated the curves through a linearly increasing curve which starts at the start of the gene and increases to its maximum value at the gene end, as in Figure 9.6C. Since we wish the amount of axis proteins per gene to be proportional to gene length, we set the "triangle" height to be the same for all genes. As a consequence, the area of the triangle is proportional to gene length, as described by experimental data (Figure 9.6A).

The precise mathematical formulation of SPoRE model is the following. First we define the raw curve before smoothing:

$$h(x) = \sum_{g \in G} \mathbf{1}_{[a_g, b_g]}(x) . \frac{x - a_g}{b_g - a_g}$$

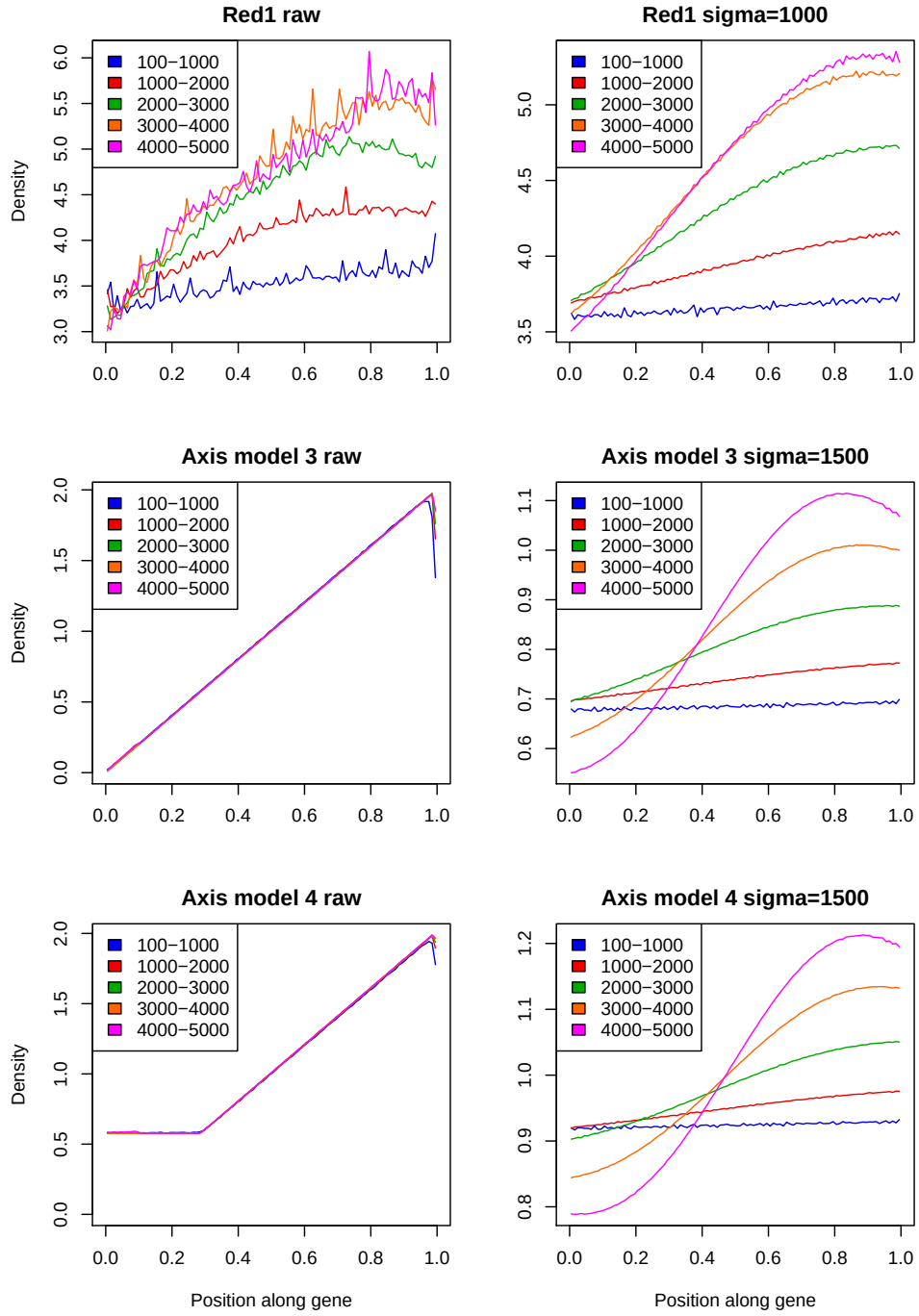

Figure 9.6: (A) Experimental curves describing the accumulation of Red1 proteins in *S. cerevisiae* (Panizza et al., 2011) along sets of genes of comparable length. Five different curves are plot, corresponding to gene lengths in the intervals $[100, 1000]$, $(1000, 2000]$, $(2000, 3000]$, $(3000, 4000]$ and $(4000, 5000]$. Gene lengths are normalized over the interval $[0, 1]$. (B) The experimental distribution curve in A has been smoothed (with a smoothing coefficient $\sigma = 1000nt$) along the entire chromosomes and the resulting smoothed curve corresponding to gene regions is plot again for sharper visualization. (C) Modeling curve used to approximate experimental data in SPoRE. (D) The modeling distribution curve in C has been smoothed (with a smoothing coefficient of $\sigma = 1500nt$) along the entire chromosomes and the resulting smoothed curve corresponding to gene regions is plot again for sharper visualization. (E) As in C, but including background noise (model not included in Table 9.1; see text). Noise level is computed from experimental data in (Panizza et al., 2011). (F) As in D, but based on the modeling distribution curve in E.

where G is the set of all genes and x the position (in nucleotides) on the genome, a_g is the position of the start codon of g, and b_g is the position of its stop codon. The function $\mathbf{1}_{[a,b]}(x)$ has value 1 if $x \in [a, b]$ and 0 otherwise. Then we apply a kernel-based smoothing with a Gaussian kernel to $h(x)$. Namely, we compute the convolution with a gaussian kernel K to obtain the final function f_{Red1} which is our Red1 model curve:

$$f_{\text{Red1}}(x) = (h * K)(x) = \int_{-\infty}^{+\infty} h(x) \cdot e^{-\frac{(t-x)^2}{2\sigma^2_{\text{smooth}}}} .dt$$

where σ_{smooth} is 1500 nucleotides.

9.1.3 DSB model

SPoRE localizes DSBs in promoter regions. Since these regions are not easily identifiable, SPoRE follows a few rules to approximate their position in an intergenic region: 1. if the region is convergent, then no DSB is supposed to occur in it, 2. if the region is oriented, then DSBs are located at the center of the intergenic region, accounting for the promoter of the starting gene, 3. if the region is divergent, then DSBs are located at two positions, corresponding to the two promoters, at $1/3$ and at $2/3$ of the intergenic region. In cases 2 and 3, the amount of DSBs is also modeled to be dependent on the average GC-content within a window (see Methods). If TFBS are available, SPoRE can use them to identify the promoter region of a gene and replace the location identified by steps 2 and 3 above with a more accurate evaluation of the promoter location.

SPoRE adds one more contributing factor to the above model: the intergenic region length. For this, it makes sure that the contribution of very long intergenic regions would not be penalized by high weights, and fixes a maximum weight threshold to a value IRL_{max}.

Formally, SPoRE modeling curve $f_{\text{DSB}}(x)$ is defined as:

$$\sum_{g \in G} \min(\text{irl}_g, \text{IRL}_{\text{max}}) \cdot (\max(0, \text{gc}(p_g) - \text{GC}_{\text{min}}))^2 \cdot e^{-\frac{(x-p_g)^2}{2\sigma^2_{\text{smooth}}}}$$

where G is the set of all genes, x the position (in nucleotides) on the genome, irl_g is the intergenic region length before the gene (on the strand where g is lying). The position p_g depends on both the orientation of g and the position of gene g' preceding g; $\text{gc}(p_g)$ is the smoothed GC content at position p_g. Let $[a, b]$ be the intergenic region and a be the start codon position of g, then:

$$p_g = \begin{cases} a + (b-a)/2 & \textit{if } g \textit{ and } g' \textit{ are on the same strand} \\ a + (b-a)/3 & \textit{if } g \textit{ and } g' \textit{ are on opposite strands} \end{cases}$$

The two thresholds IRL_{max} and GC_{min} are defined as $\text{IRL}_{\text{max}} = \mu_{\text{IRL}} + \sigma_{\text{IRL}}$ and $\text{GC}_{\text{min}} = \mu_{\text{GC}} - 3\sigma_{\text{GC}}$, where μ_{IRL} (μ_{GC}) and σ_{IRL} (σ_{GC}) are mean and standard deviation of the distribution of intergenic region lengths (*GC* content) over the whole genome. The quadratic term describes a preferred DSB concentration in regions with a higher GC content.

This model takes into account the observation that divergent regions host about the double amount of DSBs compared to oriented regions (indeed, 2 gene starts instead of

1 in an intergenic region influence twice as much the average DSB density) and that, at large scale, on the thousands of basepairs scale, GC-content correlates with DSBs (Pan et al., 2011).

9.2 Comparison with experimental data

SPoRE has been constructed to predict DSB and axis proteins distribution along chromosomes, and to measure the importance of different factors in this prediction. To evaluate how accurate SPoRE modeling is, we performed four types of analysis:

- experimental data on Red1 (Pan et al., 2011) and Spo11 (Panizza et al., 2011) proteins obtained for the *S. cerevisiae* genome were considered and the local/global Pearson and Spearman correlations between SPoRE modeling curves and experimental curves were computed. The distribution of peaks, characterizing sites of highest protein concentration, along the two curves was studied. Several models, characterized by different combinations of genomic signals, were tested to numerically evaluate the impact of each signal.

- coherence of SPoRE predictions was tested on two experimental datasets (Panizza et al., 2011; Pokholok et al., 2005) related to axis proteins and DSBs.

- SPoRE was run on four yeast species.

- SPoRE was compared to existing DSB predictors, all based on machine learning (Liu et al., 2012; Jiang et al., 2007; Chen et al., 2013).

9.2.1 SPoRE model and axis proteins in *S. cerevisiae*

SPoRE model (that is model 3 in Table 9.1) is based on the hypothesis that axis proteins accumulate at the end of genes, that intergenic region length is the main factor for protein density, and that taking into account protein diffusion and convection along the gene improves precision. SPoRE reaches average Pearson local (global) correlation $r = 0.63$ ($r = 0.54$; Figure 9.7A) and Spearman's local (global) correlation $\rho = 0.63$ ($\rho = 0.60$). We note that lower correlations are obtained when an increasing distribution of proteins along the gene is omitted (model 2 in Table 9.1): Pearson's local (global) correlation is $r = 0.58$ ($r = 0.52$), and Spearman's local (global) correlation is $\rho = 0.54$ ($\rho = 0.51$).

Red1 localization is well predicted by the position of the peaks of SPoRE modeling curve (Figure 9.8). For instance, along all chromosomes, 62% of real peaks are found by our model at a distance of at most $\Delta = 1$ kb from a predicted peak (74% at 1.5 kb), and 62% of the predicted peaks are at most 1 kb away from a real peak (73% at 1.5 kb). Sensitivity and PPV at increasing Δ values are illustrated by the curve plot in Figure 9.9A. We notice that random models, based on random selections of spots along the genome (see Methods), give much lower PPV and sensitivity values.

It is worth noticing that the usage of constant weights makes the model performance very poor, as the correlation with real data falls down to $r = 0.14$ (model 1 Table 9.1).

Axis proteins - Red1						
Model description			Pearson correlation		Spearman correlation	
Id	**Positions**	**Weights**	**loc**	**glo**	**loc**	**glo**
1	Gene ends	1	0.14	0.11	0.13	0.11
2	Gene ends	gene length	0.58	0.52	0.54	0.51
3	Diffusion along gene	gene length	**0.63**	**0.54**	**0.63**	**0.60**
DSB - Spo11						
Model description			Pearson correlation		Spearman correlation	
Id	**Positions**	**Weights**	**loc**	**glo**	**loc**	**glo**
1	Gene starts	1	0.34	0.28	0.68	0.65
2	Gene starts	gene length	0.26	0.21	0.65	0.63
3	Promoters	1	0.48	0.40	0.74	0.71
4	Promoters	*IRL*	0.50	0.41	0.74	0.70
5	Promoters	*GC*	0.58	0.52	0.75	**0.72**
6	Promoters	$GC \times IRL$	**0.62**	**0.56**	**0.76**	**0.72**

Table 9.1: **Performance of SPoRE and other models for axis proteins and for DSBs.** Local and global Pearson and Spearman correlation coefficients have been calculated between different model curves and *S. cerevisiae* experimental data for axis proteins (Pan et al., 2011) and DSBs (Panizza et al., 2011). Best performance is highlighted by bold characters. Different models are characterized by different weighting factors (column "weights"). For DSB analysis, *GC* is GC-content smoothed with a Gaussian kernel of 1000 nucleotides; *IRL* is the intergenic region length, or IRL_{max} if the region is too large (see Methods). SPoRE model for axis proteins is number 3, and for DSBs is number 6. Values are output of the SPoRE program (Figure 9.3, bottom right). See also the correlation curves for models 3 and 6 in Figure 9.7.

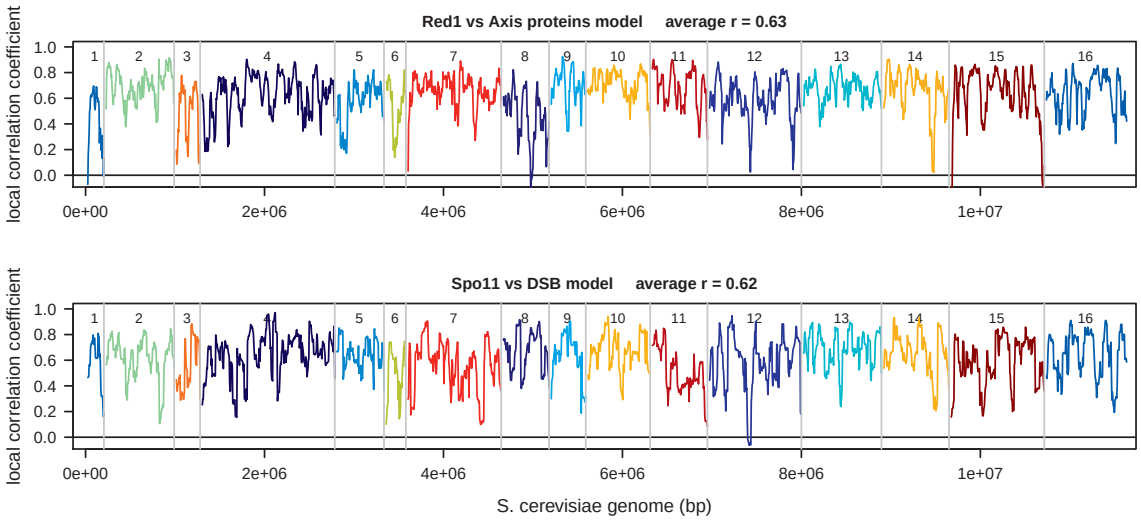

Figure 9.7: **Local correlation between experimental data and SPoRE models along the whole genome, colored by chromosome**

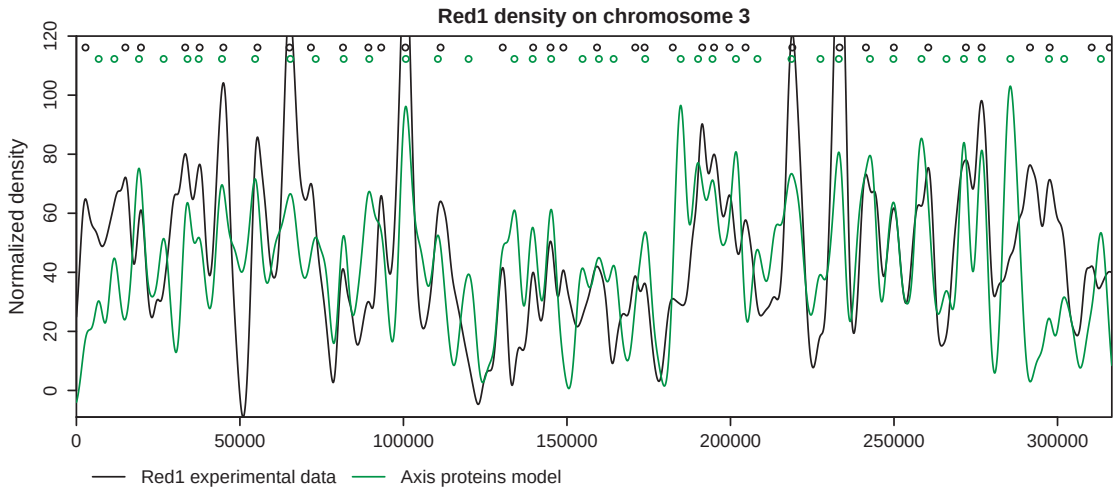

Figure 9.8: **SPoRE model for axis proteins compared to experimental data in *S. cerevisiae* chromosome 3.** Red1 density curve (Panizza et al., 2011) (black) and SPoRE axis proteins modeling curve (green) on chromosome 3. Peaks of the curves are marked by colored circles on the top of the plot.

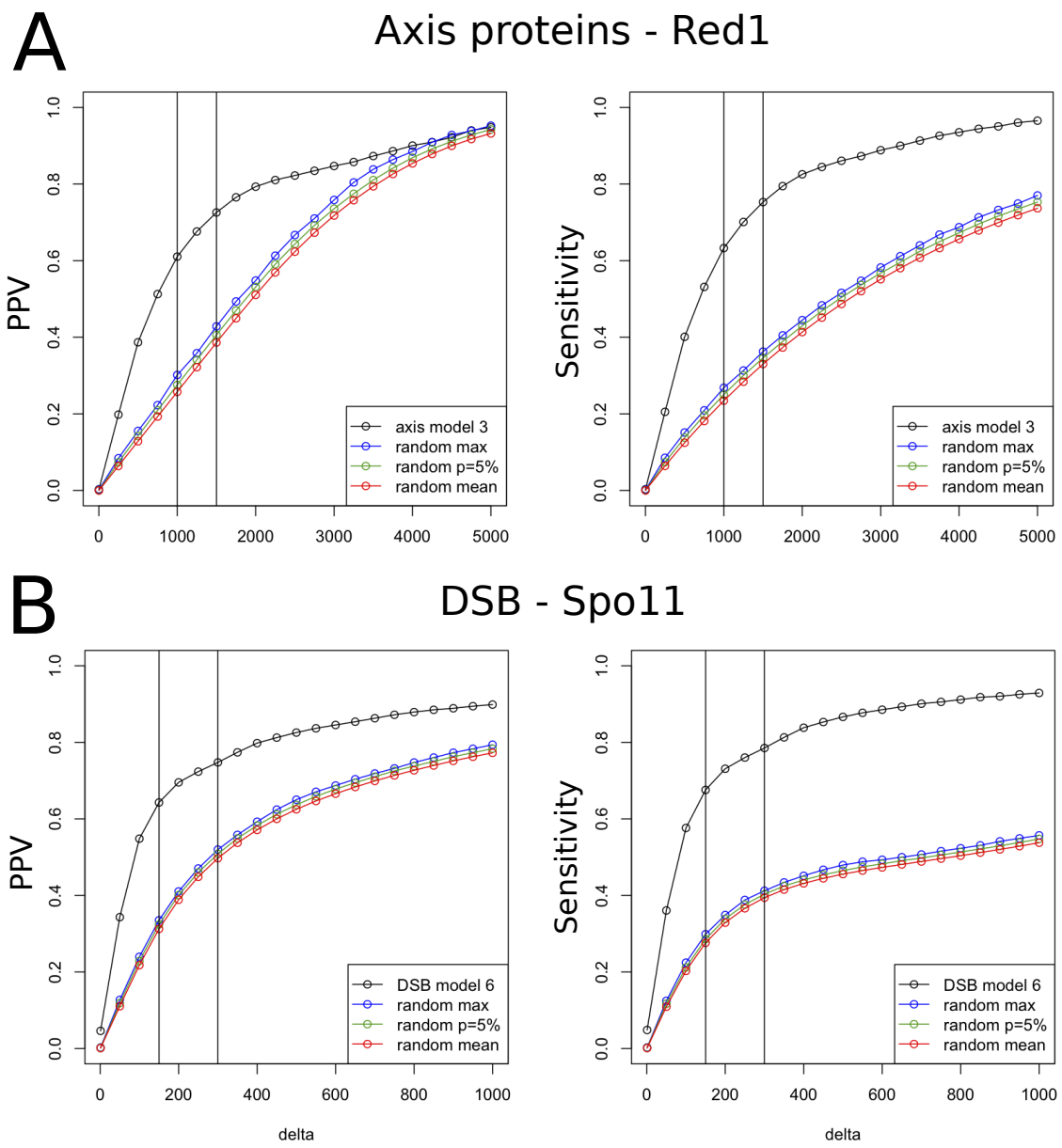

Figure 9.9: **SPoRE performance in detecting axis proteins and DSB hotspots for *S. cerevisiae*.** Peaks localization (not density) in SPoRE curves is compared to peaks localization in experimental curves for axis proteins Pan et al. (2011) (A) and DSBs (Panizza et al., 2011) (B). Positive Predictive Value (PPV) and Sensitivity (see Methods) obtained with SPoRE models (number 3 for axis proteins and number 6 for DSBs) are reported for increasing values of the parameter Δ, representing the maximum distance allowed between two peaks to say that they match. The vertical bars in the plots correspond to Δ = 1kb and 1.5kb in A and to Δ = 150nt and 300nt in B. Different random models are used to analyse SPoRE behavior (see Methods): best PPV/sensitivity over 1000 simulations (blue), PPV/sensitivity for a p-value of 5% (green), average PPV/sensitivity over 1000 simulations.

Strictly speaking, even the positional analysis of the peaks, as discussed above, is dependent on appropriate weight values, because a smoothing is performed before extracting the peaks (Gaussian window with σ = 1.5kb). Therefore, peaks result from the accumulation of high weights and they are not simply modeling gene ends. This is why model 1 (Table 9.1) has much lower PPV and sensitivity than model 2.

Finally, since experimental data highlight the existence of a background noise inducing a basic level of Red1 distribution along chromosomes, we verified whether, by including a fixed noise level in SPoRE model (see Methods), predictions in *S. cerevisiae* would be improving the fit or not. A minor improvement in Pearson correlation coefficients (local at $r = 0.64$ and global at $r = 0.56$) is observed.

9.2.2 SPoRE model and DSBs in *S. cerevisiae*

The SPoRE model (that is model 6 in Table 9.1) assumes that DSBs concentrate in gene promoter positions, and that intergenic region length and GC-content are key factors for explaining DSB density. SPoRE displays a local Pearson correlation $r = 0.62$ and a Spearman correlation $\rho = 0.76$ with experimental data (Pan et al., 2011). The heatmap of the experimental Spo11 distribution curve (Pan et al., 2011) and the Spo11 SPoRE modeling curve, reported in Figure 9.10, shows a sharp diagonal confirming the accurate prediction of the model and in particular the precise prediction of regions with high DSB density or DSB absence.

Localization of DSB high density spots is well predicted by the position of the peaks of our modeling curve. For instance, 64% of the predicted peaks are found at most Δ = 150nt away from a real peak (PPV) and 68% of the real peaks are found at less than 150nt away from a predicted peak (sensitivity). Sensitivity and PPV at increasing Δ values are reported in Figure 9.9B. In comparison, a random model based on a random selection of spots in intergenic regions (see Methods), displays much lower PPV and sensitivity.

Although SPoRE identifies a subset of the peaks found by the model at constant weights (see sensitivity in model 3, Table 9.1), it clearly predicts better their heights when GC-richness and, to a lesser extent, intergenic region length are considered. The performance of these different models is reported in Table 9.1.

Finally, we tested whether the knowledge of TFBSs in *S. cerevisiae* (Chang et al., 2011), leading to a more accurate promoter region localization, improves SPoRE predictions or not. There is no improvement on peak heights prediction (Pearson and Spearman local and global correlation coefficients do not increase). For peak localization, PPV slightly increases to 67% and sensitivity to 69% for Δ = 150nt, and we conclude that a precise estimation of promoter regions helps modeling DSB localization. The effect of TFBS availability in modeling remains limited though.

9.2.3 Coherence of SPoRE predictions with two large-scale experimental datasets

SPoRE modeling curves can be used for comparison with experimental data of different origin. In this respect, we considered two different datasets.

First, as mentioned in the introduction, it has been shown previously that Red1 and Hop1 patterns are influenced by Rec8 (cohesin) patterns (Panizza et al., 2011). Hop1, for

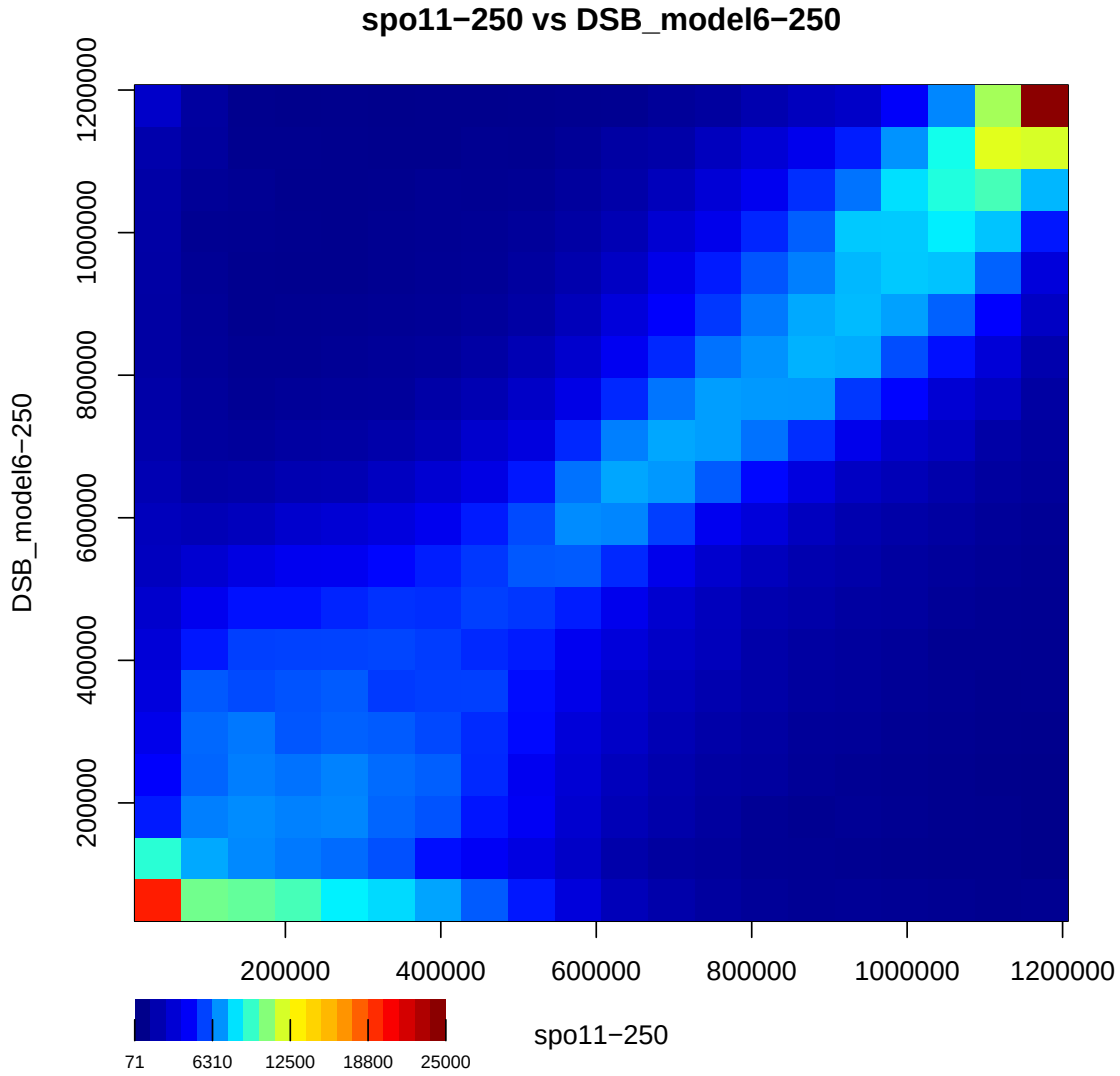

Figure 9.10: **Heatmap of the experimental Spo11 distribution curve (Pan et al., 2011) and the Spo11 SPoRE curve on the *S. cerevisiae* genome.** Pairs of y-values belonging to the two curves have been recorded every 10nt along the chromosomes, and a total amount of about 1.2 millions points (y_1, y_2) were identified, where y_1 and y_2 are the y-coordinates of the experimental and modeling curves, respectively. In the plot, the y-coordinates have been replaced by their ranks to allow for better visualization. The x-axis reports ranks from the experimental curve and the y-axis reports ranks from the SPoRE modeling curve. Each square in the plot describes the number of points falling into the corresponding interval of rank values. The dark red square on the top right collects picks with the highest y-ranks and the red square on the bottom left collects points in the experimental curve displaying no Spo11 accumulation, and therefore no DSBs.

instance, is distributed almost like Red1 (local correlation is $r = 0.92$, global is $r = 0.88$) with which it interacts (Hollingsworth and Ponte, 1997; Woltering et al., 2000). On the other hand, Rec8 is more abundant around centromeres than Red1/Hop1, although local variations are the same. Therefore, Rec8 global correlation with Red1 is only $r = 0.57$, while its local correlation is still $r = 0.83$. Because of these correlations, we expect SPoRE to be locally well correlated with Hop1 and Rec8 (data from (Panizza et al., 2011)). Indeed, we find that SPoRE model has a local correlation of $r = 0.62$ with Hop1 and $r = 0.60$ with Rec8, compared to $r = 0.64$ with Red1. This confirms that SPoRE local distribution patterns are shared by the three axial proteins. Consistently, if we look at global correlation coefficients, SPoRE is well correlated with Hop1 ($r = 0.55$) and Red1 ($r = 0.56$) but weakly correlated with Rec8 ($r = 0.33$).

Second, we compared SPoRE curves to histone trimethylation data. It has been observed before that H3K4 trimethylation (H3K4me3) is linked to DSBs (Borde et al., 2009). Then, we computed correlations between H3K4me3 (data from (Pokholok et al., 2005)) and SPoRE modeling curve for Spo11. We find $r = 0.25$, which is comparable to $r = 0.21$ obtained when we correlate H3K4me3 and DSB experimental data. Similarly, with Spearman coefficients, we find $\rho = 0.61$ between H3K4me3 and our model, and $\rho = 0.52$ between H3K4me3 and DSB experimental data. We conclude that SPoRE model is consistent with this known interaction.

Both these examples confirm that the modeling curves are faithful approximations of experimental curves and that biological conclusions can be safely derived from them.

9.2.4 SPoRE predictions on several yeast species

The large number of sequencing projects on yeast clades and the upcoming new projects (still a few today) exploring the molecular biology of yeast species encourages the usage of predictive tools for learning about the distribution of DSB and axial proteins sites, to start comparative studies on yeasts across clades. We run SPoRE on *Lachancea kluyveri* and *Kluyveromyces lactis*. The genome of *L. kluyveri* shows a particularly high GC-content on the left-arm of the C chromosome and SPoRE predicts a higher concentration of DSBs in this chromosomal arm. We note that the number of peaks within the C-left arm is comparable to other chromosomal arms, and that SPoRE detects the same number of peaks (353) than model 4, which excludes the GC factor. Hence, the GC factor in SPoRE exclusively influences DSB density and not DSB positioning, and the position of DSB hotspots along the C-left arm is therefore a consequence of high GC. Experiments are expected to confirm this prediction.

We have also run SPoRE on *Schizosaccharomyces pombe* where recombination is known to be partially dependent on DNA motifs. As expected in this species (de Castro et al., 2012), SPoRE predicts a large number of DSBs in large intergenic regions. It should be noticed that in *S. pombe*, divergent and oriented regions are unusually large compared to other yeast species. In *S. cerevisiae*, *L. kluyveri* and *K. lactis* for instance, the mean length of divergent and oriented regions, is approximately 700nt while it is 1200nt for *S. pombe*. Since SPoRE favors DSBs in oriented and divergent regions, and since the size of these regions plays an explicit role in the model, SPoRE prediction confirms the previous observations.

When comparing SPoRE predictions with the DSB distribution in *S. pombe* (Fowler

et al., 2014), results are much less accurate than with *S. cerevisiae*. We get a local Pearson correlation of $r = 0.36$ (global correlation is $r = 0.26$). Spearman correlation is better with $\rho = 0.43$ (global correlation is $\rho = 0.42$). This can be explained by the major differences between *S. cerevisiae* and *S. pombe*. As explained by Fowler et al. (2014), in *S. pombe*, DSB do not occur in most promoters and can occur in convergent regions. More precisely, in *S. cerevisiae*, 91% of divergent intergenic regions contain a DSB peak, while this number is only 70% in *S. pombe*. In *S. cerevisiae* the ratio between the number of DSB per kb in divergent versus convergent regions is around 14, while it is only 3 in *S. pombe*. Both these observations are in contradiction with our model, and that explains its poor performance for this species.

9.2.5 Comparison between SPoRE and other predictive tools

Several tools, based on nucleotide sequence analysis (considering k-mers, for $k \geq 2$) have been proposed (Liu et al., 2012; Jiang et al., 2007; Chen et al., 2013) as predictors of recombination or DSB hotspots.

We compared to the most recent one, iRSpot-PseDNC (Chen et al., 2013), which improved above the others. In (Chen et al., 2013), the authors compared their predictions of DSB sites against 452 hotspots on chromosome IV extracted from the same Spo11 experimental data (Pan et al., 2011) that we compared to. They found that their program predicts as as hot 347 of these hotspots, corresponding to a true positive rate of 77% (Chen et al., 2013). When applying the same test to our model, we predicted as hot 265 of these 452 hotspots, corresponding to a true positive rate of 59%. However, to perform a proper benchmark, negative instances (coldspots) should be included in the test set, so that the false positive rate can also be measured. We therefore enlarged the dataset by adding 452 randomly chosen coldspots in the same experimental data and on chromosome IV (see Methods). On this symmetric test set, the overall success rate of iRSpot-PseDNC falls to 54% against 84% for our model (see Methods), compared to an expected 50% for a random prediction. This is due to the fact that iRSpot-PseDNC detects 309 false positives (false positive rate is 68%) while we only detect 6 of them (false positive rate is 1%). This shows that iRSpot-PseDNC is little better than random in detecting DSB hotspots. It should be noted that comparison is realized on hotspot sites localization but that no prediction on protein density is made by iRSpot-PseDNC, contrary to SPoRE, where estimations of density can be directly inferred from the modeling curve.

A second test was realized on the same dataset used in (Chen et al., 2013) to compare iRSpot-PseDNC to IDQD (Liu et al., 2012). This dataset, defined in (Liu et al., 2012), is composed of 490 hot ORFs and 591 cold ORFs, where the hot ORFs describe a set of recombination hotspots. Notice that a recombination hotspot is expected to be located close to a DSB site but not the viceversa, and that SPoRE cannot be directly used for predicting recombination hotspots since it was designed to predict DSB hotspots. Hence, we decided to test how much the smoothed GC-content, that we used as a factor in SPoRE, contributes to the identification of recombination hotspots. By using only GC-content, we obtained an accuracy of 83% (see Methods), against the 80% reached by IDQD and the 85% reached by iRSpot-PseDNC (based on a 5-fold cross-validation of the SVM approach they implement). The conclusion is that even though iRSpot-PseDNC is based on the actual DNA content (taking dinucleotide frequency as its predictor), it appears that

almost all the signal can in fact be recovered simply with the GC-content in a window.

9.3 Conclusions

We explored the hypothesis that genomic signals coordinate the formation of the loops (their position and length) in the 3D chromosomal structure during meiosis. Our aim here is not to study the dynamics of a protein localization process but rather to identify the genomic information that influences the 3D structure formation and quantify the importance of the genomic factors. SPoRE allows us to test whether genomic signals play a role or not, and at what extent, in the accumulation of axis proteins and DSBs along chromosomes. All genomic factors considered in the model are linear functions with the exception of a quadratic factor modeling the impact of GC content. New parameters can be easily added to the model for the evaluation of new genomic markers effects. The interest in this modeling approach comes from a straightforward biological interpretation of the parameters that helps to reason on plausible biological mechanisms forming protein accumulation.

9.3.1 Orientation of genes and chromosomal axis formation

We have shown through a formal model that the distribution of the chromosomal axis proteins is encoded in gene organization along DNA. The orientation of the genes influences the formation of the loops within the 3D axial structure during meiosis and to reach an understanding of this 3D structure formation, this fact should be combined with the existence of a random process governing the binding of the axis proteins to DNA and with a pervasive transcriptomic activity inducing a repositioning of the proteins in specific sites along the genome. In this respect, SPoRE model could help to design appropriate genomic signatures for synthetic chromosomes that should form a functional synaptonemal complex structure.

9.3.2 Modeling organisms other than yeast

SPoRE could be used to infer localization and density of axis proteins and DSBs sites at large scale for those yeast species for which whole genome experiments have not been made yet. Today, more than 40 yeast genomes have been completely sequenced and for many of these yeast species, meiosis either exist or can be induced. It might be interesting to apply SPoRE model to these species to check, through comparative genomics, whether syntenic region boundaries correspond to DSB hotspots or not across species, whether the genetic content of DSB hotspots and of their neighborhoods are conserved in different species and so on.

Axial chromosome structures formation has been experimentally observed across many sexually reproducing eukaryotic species, from fungi to vertebrates. In yeast, our model highlights that axial chromosome structures and DSB distribution are governed by a rather simple combination of genomic signals. For other organisms, the model might be expected to become more complex. For the mouse, for instance, other factors such as DNA binding sites targeted by axial proteins have been demonstrated to play an active role in DSB localization (Smagulova et al., 2011). In this respect, SPoRE might be taken as a

nutshell to add extra signals and reach appropriate descriptions of experimental data in other organisms, possibly multicellular ones. SPoRE software is provided to allow users for further development and testing of new genomic factors.

9.4 Technical details

Visualization in a genome browser

To allow biologists to visualize easily SPoRE modeling curves, SPoRE provides its results in the WIG file format. They can be loaded in the UCSC genome browser (http://genome.ucsc.edu/), in the genome browser available at http://yeastgenome.org/ and in the IGV software (see Figure 9.2) (Thorvaldsdóttir et al., 2013). For the four yeast genomes that we analyzed, the corresponding wig files are available at http://www.lcqb.upmc.fr/SPoRE/. For convenience, we also provide the corresponding *S. cerevisiae* experimental data in the same format, to allow for easy comparison.

Software availability

SPoRE program is provided to the users that would like to apply it to yeast species, others than those we already considered here, or modify it for other organisms. The "readme" file explains what are the parameters that should be set for other organisms. The software is available at http://www.lcqb.upmc.fr/SPoRE/

Annotation

The reference strain we used to validate SPoRE is *Saccharomyces cerevisiae* S288C. The gene annotations were retrieved from the *Saccharomyces* Genome Database (http://www.yeastgenome.org/), release 64. We included 4879 "verified" ORFs and 895 "uncharacterized" ORFs in our set of coding genes, but not "dubious" ORFs. We also considered transposons by taking the 89 features labeled "transposable element gene", rRNAs (RDN37-1, RDN37-2, RDN5-1, RDN5-2, RDN5-3, RDN5-4, RDN5-5, and RDN5-6), and pseudogenes (21). For *Lachancea kluyveri* and *Kluyveromyces lactis*, genomes and annotations were downloaded from Genolevures (http://www.genolevures.org/). Only features named "CDS" were taken into account in our models. For *Schizosaccharomyces pombe*, genome and annotation were downloaded from PomBase (http://www.pombase.org/). We used features labelled "CDS", representing exons, and merged them together to get intervals defining genes in our models.

Protein density data used for SPoRE validation

We use protein density data along the genome from Spo11 immunoprecipitation/454 sequencing for DSB (Pan et al., 2011) and from ChIP-on-chip for Red1, Hop1 and Rec8 (Panizza et al., 2011). They were mapped on the *S. cerevisiae* S288C genome, even though strain SK1 was used in the experiments. Raw data were used for computing all correlations reported in Tables 9.1. They were retrieved from supplementary data in (Pan et al., 2011) for Spo11, and from the GEO dataset GSE29860 for Red1/Hop1/Rec8.

Smoothing

To smooth the curves, we use a kernel-based smoothing with a Gaussian kernel. We use the "density" function provided in R (R Development Core Team, 2011) for all our models, the Spo11 experimental data and the GC-content. We use the "ksmooth" R function for Red1 experimental data to take into account correctly the irregular spacing of the tiling array probes. When referring to σ nt smoothing, we mean that the Gaussian kernel we use has a standard deviation of σ.

For DSBs, we used $\sigma = 250$ nt for both data and models. Notice that Spo11 experimental data have a nucleotide-level precision and that the smoothing we use takes into account the range in which Spo11 might cut DNA around hotspots. For axis proteins, we used $\sigma = 1000$ nt for the Red1 experimental data, and $\sigma = 1500$ nt for our models. The rationale behind the different values is that ChIP-on-chip experiments produce large fragments of DNA where proteins bind and, as a consequence, they are detected by a large range of probes in the microarray. The accumulation of probes does the equivalent of a smoothing, and because of this, we need to smooth the data less than in the model. The two parameters were adjusted so that the number of peaks detected on both smoothed curves are approximately the same (1558 for *S. cerevisiae* data, 1615 in SPoRE model). More precise experimental data might correspond to a different smoothing constant σ and the software allows for easy changes.

Normalized density and experimental noise

Normalized density (y axis in Figure 9.8) is defined by translating and scaling the values such that the first percentile maps to 1 and the 99th percentile maps to 99. This is a way to scale the data approximately between 0 and 100 without taking into account extreme values. In fact, these latter might be a consequence of the experimental noise. In Red1 model 4 (Table 9.1), noise was estimated from data by considering the 1st percentile m and the 99th percentile M, where $m = 2.169456$ and $M = 7.622778$ for *S. cerevisiae*. The ratio $M/m = 3.5$ has been used to estimate the noise level in Figure S3E.

Correlations between model and experimental curves

To estimate the local correlation between two curves, we considered a window of 50 kb in which we compute the correlation coefficient (Pearson or Spearman) between points of the two curves every 10 nt. Then we move the window by 10% of its size (ie. 5 kb) and repeat the computation until we reach the end of the chromosome. We repeat these operations for each chromosome, and finally, we take the average of all these correlation coefficients (from all windows from all chromosomes).

Global correlation is computed by considering the complete genome at once (all points every 10 nt), instead of a sliding window. It provides a single correlation coefficient.

Peak predictions and their evaluation

High density spots for both axis proteins and DSBs are computed as the peaks of the corresponding smoothed curves. They are defined as local maxima that are at least $\varepsilon = 1$ normalized density unit (see "Normalized density") above the surrounding local minima.

To evaluate high density spot predictions versus experimental hotspots, we used two standard measures, sensitivity and Positive Predicted Value (PPV). Namely, for each peak in the experimental curve at position x, we look for a peak in the model lying in the interval $[x - \Delta, x + \Delta]$. If there is such a peak then we count it as a true positive. Sensitivity is defined as the fraction of true positives over the number of real peaks. Positive Predictive Value is defined symmetrically to sensitivity, by reversing real and predicted peaks. It is the fraction of real peaks over the number of predicted peaks.

Random models for axis proteins and DSBs sites

In order to test whether sensitivity and PPV values scored by SPoRE for axis proteins and DSB spots predictions are not the result of chance, we generated 1000 random models for the two kinds of loci. For axis proteins, the models were generated by randomly selecting 1615 positions along the whole *S. cerevisiae* genome, that is, the same number of peaks as in SPoRE model 3 in Table 9.1. For DSB spots, the models were generated by randomly selecting 4242 positions in *S. cerevisiae* intergenic regions, that is, the same number of peaks as in SPoRE model 6 in Table 9.1. We explicitly considered intergenic regions because it is already known that DSB spots occur there. We wished to test whether our predictions are closer to real axis proteins or DSB spots than a random choice. After generating the random positions, we evaluated the position against experimental peaks by using the same method employed for SPoRE (see above).

Intergenic region lengths

SPoRE model for DSBs uses intergenic regions lengths as a contributing weight. Precisely, given an input genome, we compute the distribution of its intergenic region lengths and set the threshold $\mathrm{IRL}_{\max} = \mu + \sigma$, where μ and σ are average and standard deviation of the distribution. For *S. cerevisiae*, this value is 1202 nt (the first analysis of these regions in *S. cerevisiae* dates back to (Dujon, 1996)). For intergenic regions that are "too large", that is $> \mu + \sigma$, we set the weight to $\mathrm{IRL}_{\max}$, that is, the weight stops growing after the threshold.

GC content

When taking into account GC content in our model, we use a kernel-based smoothing of the GC distribution, obtained from a Gaussian kernel with standard deviation 1 kb. Then we define all GC-based values with the smoothed GC curve: μ_{GC}, σ_{GC} and $\mathrm{gc}(p_g)$ (see above). All along the genome, we assume the presence of a minimal amount of GC content expressed by the threshold $\mathrm{GC}_{\min} = \mu_{\mathrm{GC}} - 3\sigma_{\mathrm{GC}}$.

Gene projections

Plots in Figure 9.4 and Figure 9.5 were created by first smoothing the experimental data, then summing Red1/Spo11 smoothed curves after centering them on reference positions (gene 5' end, gene 3' end, intergenic region centers). The smoothing Gaussian kernel standard deviation used is $\sigma = 20$ nt.

Promoters and Transcription Factor Binding Sites (TFBS)

SPoRE can model DSBs either by approximating the position of promoter regions proportionally to the length of the associated intergenic region (see DSB model description above), or by exploiting knowledge of TFBS when available. For the latter, given a gene, it considers the set of its TFBS and computes the average of their positions as the reference position to set the weight of the SPoRE model. In case a gene has no known TFBS, then SPoRE models its promoter location based on the length of its intergenic region. For *S. cerevisiae*, we used TFSB positions indicated in the Yeast Promoter Atlas (Chang et al., 2011) repository, available at http://ypa.ee.ncku.edu.tw/.

Comparison with iRSpot-PseDNC on DSB data

Comparison between SPoRE DSB model and iRSpot-PseDNC (Chen et al., 2013) was realized on the dataset of 452 experimentally annotated (Pan et al., 2011) recombination hotspots for the *S. cerevisiae* chromosome IV. This set, originally used to evaluate iRSpot-PseDNC in (Chen et al., 2013), has been extended with 452 coldspots that we extracted from the same experiment (Pan et al., 2011). This extension was done in order to test both systems for false positives. More precisely, for each hotspot in the dataset, we randomly selected a fragment of DNA on chromosome IV with the same length as the hotspot, but without any experimentally detected DSB, and verified that these fragments do not overlap each other. (Notice that 17% of the *S. cerevisiae* genome is made of regions that are larger than 242 nt, that is the average size of a hotspot, and that contain no peak. We have randomly selected coldspots within these regions.) Hence, we obtained a set of coldspots with the same number of sequences and the same length distribution as the set of hotspots. We then tested iRSpot-PseDNC online by providing the server with the DNA sequences in the dataset (the file is available at `http://www.lcqb.upmc.fr/SPoRE/`). To test our model, we simply predicted as a hotspot any fragment on which the average of our curve is higher than the average over the whole genome.

A second dataset was used for comparison with iRSpot-PseDNC and IDQD (Liu et al., 2012). It is defined in (Liu et al., 2012) and it is downloadable as SI of (Chen et al., 2013). This set is defined by ORFs, but since SPoRE uses information about intergenic regions instead, we benchmarked SPoRE on this dataset by predicting hotspots on the intergenic regions lying before the gene start. Namely, we compared the average of our modeling curve in this region to its mean μ and standard deviation σ by predicting hotspots when the average of the curve is $\geq \mu + \sigma$. When the GC-content curve has been tested as a predictor of recombination hotspots in this dataset, formally, we compared the maximum of the smoothed GC-content curve in the gene and intergenic region preceeding it to $\mu_{GC} + \sigma_{GC}$, where μ_{GC}, σ_{GC} are the mean and the standard deviation of the GC-content curve on the full genome. Notice that this GC-curve could be replaced by a much simpler model. In fact, we could just consider a 4kb window centered at the start of a gene, compute its GC-content, and obtain identical accuracy.

Part III

Side works

Chapter 10

Simulating the evolution of genomes

Contents

I have been involved in a work that is related to an algorithm, PhyChro, which has been developed in the lab by Guénola Drillon. The algorithm is part of a bigger package that also includes the SynChro algorithm that has already been published (Drillon et al., 2014). The purpose of PhyChro is to reconstruct phylogenetic trees from synteny blocks. My work here was to create a program that simulates the evolution of genomes, in order to benchmark the PhyChro algorithm. This benchmark will be included in the revised version of the PhyChro manuscript[1], as running simulations was requested by the reviewers.

10.1 Phylogenetic tree inference with PhyChro

The purpose of these two programs is to reconstruct phylogenetic trees using synteny blocks. SynChro computes the synteny blocks from the species genomes and their anno-

[1] Drillon, G., Champeimont, R., Oteri, F., Fischer, G., and Carbone, A. Phylogenetic reconstruction based on chromosomal rearrangements

tations, while PhyChro uses the synteny blocks computed by SynChro to reconstruct the phylogenetic tree.

Let G and G' be two genomes. We call a synteny block a succession of genes, for example 3 genes A, B, C, such that, if we consider their respective homologous genes A', B', C', they appear successively and in the same relative order in G'. This means that somewhere in G', there is the sequence $A'B'C'$ or $C'B'A'$. In fact, in SynChro, the $A'B'C'$ sequence is allowed to contain some other inserted genes that are homologous to genes not in the synteny block (for example $A'B'D'C'$), provided they are less than Δ, which is a parameter of the algorithm.

The first task SynChro has to perform is homologous gene matching, i.e. it has to find the corresponding A', B' and C' genes in the G' genome of our example. In fact, a gene may have several homologous genes in the other genome. This case is handled by SynChro by considering reciprocal best hits (RBH), which are pairs of genes (A, A'), with A in G and A' in G', such that A' is the closest gene to A in G' and A is the closest gene to A' in G. Both conditions may not hold, in which case SynChro marks it as a non-RBH homologous pair of genes.

Using both the information from RBH and non-RBH homologous genes, SynChro generates a list of synteny blocks. For a more detailed explanation of the algorithm, see Drillon et al. (2014).

In order to reconstruct the phylogenetic tree of a group of species, one has to run SynChro on all pairs of species, in order to find synteny blocks between all pairs of genomes. Then, PhyChro takes this information and is able to reconstruct the phylogenetic tree of these species.

10.2 Motivation for simulations

The PhyChro program has been applied to a set of 13 vertebrate species and 21 yeast species, and was able to reconstruct the phylogenetic tree correctly. A biological validation is a very good way to assess the algorithm performance, because it is biological applications that matter in the end, and nothing can be more realistic that biological data itself. However, there are still motivations for using simulations to further benchmark the algorithm:

- A simulation allows a large-scale testing, that can show the robustness of the algorithm, even if biological data is scarce. We can then be sure it did not find the correct answer "by chance" but works reliably.
- Since PhyChro provides confidence scores, a simulation can tell us whether these scores are useful information, in which case a high score should be correlated to a correct reconstruction.
- A simulation allows us to test the limits of the algorithm, and to know how much data and which data quality is required for the algorithm to work.

However, we should avoid the pitfalls related to simulations:

- Making the model so close to an experiment that it does not bring new information compared to the benchmark on biological data.

- Making the model so complex that we cannot know reliably the parameters to set.
- Making the model too simplistic so that it is not representative at all of what biological data would be like.
- Setting parameters that are good for the reconstruction program instead of being realistic.

We first tried to use the program MagSimus 2 developed (but not published yet) by Hugues Roest Crollius at the *Institut de Biologie de l'Ecole Normale Supérieure*. However, this program requires many parameters that are difficult to estimate, like the probability of each kind of event on each branch (problem 2 above). It is possible to fit it with biological data, but we thought it would make the simulation too close to the biological data (problem 1 above). We think the right level of imitation of the biological data is to take a yeast-like and a vertebrate-like ancestor, but not to re-create the same species that were studied, i.e. we do not want species-specific parameters. As a result, we decided to create our own simulation tool.

10.3 The model

I worked in collaboration with Gilles Fischer (head of the Biology of Genomes team in the lab) to create the model. I designed the mathematical formulation of the model and wrote the program, while he provided the biological knowledge of what events we should simulate, how many, with which frequency, etc. It is especially important that the model is realistic so we need to set the parameters to values which are the closest possible to biological truth.

10.3.1 Making the tree

We decided to create two different simulations, one that would mimic the evolution of vertebrates, and one that would mimic the evolution of yeasts. We want to get 13 vertebrates species and 21 yeast species, which have the same number of species as in the analysis reported in the PhyChro article.

We define time as going from 0, at the root, to 1, when we get the final species (leaves of the tree). We put the first branching at the root, as we are not interested in simulating events that are common to all species. Then, to get the right number of species, we choose randomly speciation times (branching) uniformly over $[0, 1]$. Then, for each speciation time, we have to split a leaf in two. We select that leaf by starting from the root of the tree, then, at each branching, we choose one of the two directions with a $1/2$ probability. When we reach a leaf, we split it in two at the speciation time previously defined. We repeat this operation until we have the right number of species.

Note that this procedure implies that there is not an identical probability to split every leaf, since a leaf that can be reached in less internal nodes is likelier to be chosen at every step. This has the consequence to make the tree more equilibrated than if leaves were chosen uniformly.

We do not claim that this tree shape is necessarily a perfect simulation of actual speciation in nature. Creating realistic speciation models is hard, and would not in fact be

really relevant, because PhyChro is designed to be used on actually sequenced and annotated genomes, which are not randomly selected in the set of all existing species. There is a bias towards some species that are easier to find or study, more interesting than others, either for biological properties, industrial or medical applications. Some species are more ubiquitous than others and therefore discovered earlier, etc. The result is that we should not expect the set of studied species to be a uniform choice over all clades. This is especially obvious when looking at the vertebrates tree, which contains as many apes as fishes, while the number of species in these two groups have in fact a different order of magnitude.

An example of a tree generated with this method can be seen on Figure 10.1.

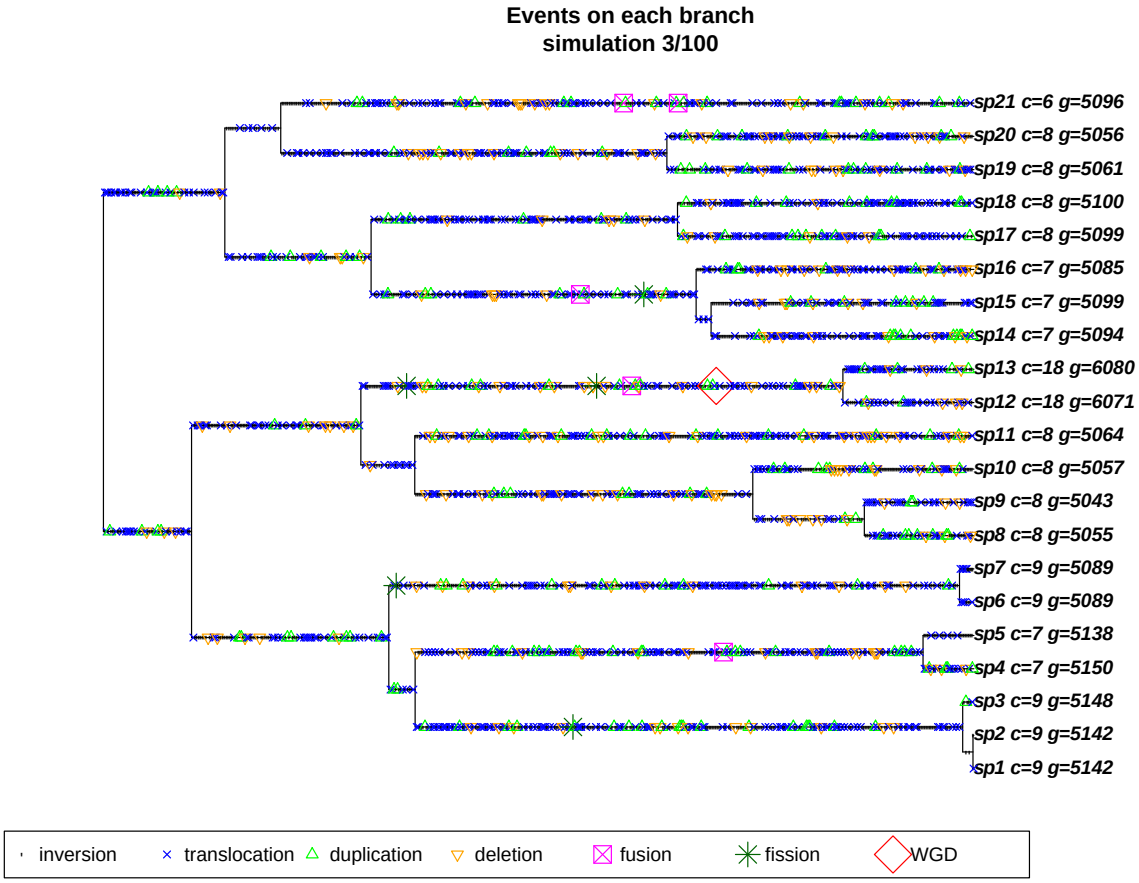

Figure 10.1: **A simulated tree with rearrangements events.**
The ancestor was given 5000 genes and 8 chromosomes, which are thought to be similar to the actual yeast ancestor (Gordon et al., 2009). The number of final simulated genes and chromosomes in each artificial species is given (resp. g and c). We have chosen to show this specific simulated tree (simulation # 3 in a set of 100) because it is the first to contain a WGD.

10.3.2 Genome model

We start with an ancestor with a number of genes and chromosomes which correspond to what can be inferred about the vertebrate and yeast ancestors. For the vertebrate ancestor,

we have chosen 23 chromosomes and 18000 genes, while for the yeast ancestor, we have chosen 8 chromosomes and 5000 genes (Gordon et al., 2009). These genomes are implemented as lists of signed integers. Each integer represents a gene. A special case is the 0 which is used to mark the centromere. If two integers have an equal absolute value, it means that the genes are homologous: orthologs when between different species, paralogs within the same species. A different sign means the gene is on a different strand. In the ancestor, all genes are different (i.e. all integers appear exactly once).

10.3.3 Simulating the number of rearrangements

One of the parameters we define is the average number of events E from the root to any species. We know approximately what this number is if we take the two most remote species, estimate the number of rearrangements between them, and divide by two. This has led us to choose $E = 500$ events for yeasts and $E = 1000$ events for vertebrates (Drillon and Fischer, 2011).

To choose the number of events on each specific branch (internal or leading to a leaf), we use a Poisson distribution and set its parameter to $\lambda = E \times L$ where L is the branch length (in the time unit defined in the previous section). Choosing a Poisson distribution means that we make the hypothesis that the probability for a rearrangement to occur is the same at any point in the tree. Note that this Poisson distribution has the consequence that the expected number of events from the root to any species is E, which is consistent with the definition of this parameter.

There is only one exception to this rule, if a specific branch has less than m events (when chosen by the Poisson distribution), we still put m events on it. We choose $m = 1$, which means we forbid branches with no events at all. This makes sense because in the absence of any event, no algorithm can detect the split associated to the branch. We have also tested $m = 10$ which we discuss later.

10.3.4 Simulating each type of event

Once the number of events on each branch has been computed, we need to decide the type of each event. To do this, we select randomly an event type in this list with the following probabilities:

- Inversion (60%): A random sequence of genes is selected uniformly over the genome. This sequence is reversed. For example if we have $(1, 2, 3, 4, 5)$ and reverse the 3 genes in the middle, we get $(1, -4, -3, -2, 5)$. This phenomenon is the most frequent type of rearrangement in a genome. The number of genes involved in the inversion is chosen according to a Poisson distribution with an average of 5 genes. The number of 5 genes is chosen to be a realistic estimate of an average inversion size. The Poisson distribution was chosen for the sake of simplicity, since it requires no other parameter than the average, which is the only one we can reliably estimate. If the randomly chosen integer is 0, we try again until we get a non-zero value.

- Reciprocal translocation (29.79%): Two positions are chosen uniformly over the genome, but on different chromosomes. We then consider the part of the two chro-

mosomes that start at the position chosen and reach the telomere, without crossing the centromere (therefore there is only one choice in doing so). These two parts are then exchanged between the chromosomes, such that the end that is the telomere remains the telomere in the other chromosome. The probability was chosen to be close to 30% but was decreased a little so that the probabilities sum up to 100% at the end.

- Duplication (5%): A randomly selected sequence of genes is duplicated. The number of genes involved is chosen with a Poisson distribution with an average of 5 genes for the same reasons as for inversions (see above). If the centromere is duplicated, randomly select one of the two centromeres and delete it.

- Deletion (5%): A randomly selected sequence of genes is deleted. The number of genes involved is chosen with a Poisson distribution with an average of 1 gene. Note that the value 0 is forbidden (since it would produce no event). If the sequence includes the centromere, we take care not to delete it.

- Fusion (0.1%): Two positions are randomly selected on the genome, on different chromosomes (like for reciprocal translocations). The two chromosomes are merged at random ends. The new centromere is randomly chosen between the two previous centromeres.

- Fission (0.1%): A random position is chosen uniformly over the genome. The chromosome to which it belongs is split at this position to form two chromosomes. One of the two chromosomes then lacks a centromere. We add one at a random position in the chromosome.

- Whole Genome Duplication (WGD) (0.01%): All chromosomes are duplicated. Each gene is then found in two copies. For each pair, we have an 80% probability to delete one of them (chosen randomly) and a 20% probability to keep both. The probability of WGD (0.01%) was chosen so that the average number of WGD in the yeast tree is 0.5 (most trees have 0 or 1 WGD, sometimes a few more).

In the special case in which we have only one chromosome in the species, translocation and fusion cannot be chosen, so the probabilities are redistributed between other events (according to their respective relative probabilities).

The whole process is run 100 times to create 100 trees with the associated species genomes. We do this simulation with both the vertebrate parameters and yeast parameters, resulting in two sets of 100 trees. The only differences between the two sets is the number of species per tree, the general frequency of events and the number of chromosomes and genes in the ancestor.

An example of simulation can be seen on Figure 10.1. In this tree, simulated WGD species sp12 has 18 chromosomes and 6071 genes, which is comparable to real WGD species *S. cerevisiae*, which has 16 chromosomes and 6275 genes. At the opposite, non-WGD species sp1 has 9 chromosomes and 5142 genes, similar to real non-WGD species *Lachancea kluyveri*, which has 8 chromosomes and 5321 genes. This means that our simulation simulates realistically the evolution of both WGD and non-WGD yeast species.

10.4 Benchmarking PhyChro

10.4.1 Preparing synteny blocks

After having created these two sets of 100 trees, we need to test PhyChro. However, a last step is needed because PhyChro takes a list of synteny blocks, not the genomes directly. On real genomes, we could use SynChro which would create the list of synteny blocks, but in our case, we do not have real genomes with DNA sequences. Instead, we have directly the homology relations (since integers with the same absolute value represent homologous genes). The step we need to add is the computation of synteny blocks. So I wrote an extra part in my simulation program which creates a list of synteny blocks. Here, I have chosen a strict definition of synteny block, that is a sequence of genes $x_1, x_2, \ldots, x_n$ (integers) such that the other genome contains either the sequence $x_1, x_2, \ldots, x_n$ or the sequence $-x_n, -x_{n-1}, \ldots, -x_1$. Contrary to SynChro, no insertion is allowed in the synteny block. This may make it harder for PhyChro to work, but this allows to test it as an independent program and to assess its performance even when it is not coupled with SynChro.

10.4.2 Results

After PhyChro has run on the 100 genomes and reconstructed a tree for each of them, we can compare these trees with the corresponding correct trees that were produced by the simulation program. The result for yeasts is shown in Figure 10.2. In the 100 trees, 61% have been perfectly inferred, i.e. have the same topology as the correct tree. With the vertebrate simulation, 79% of trees are perfectly reconstructed.

For the trees that are not perfectly reconstructed, we need to define a notion of error in order to count them. To do this, we use the notion of split. In a tree, each branch B defines a corresponding "split", which is the bipartition of species formed by the two set of species corresponding to the two remaining subtrees if branch B is deleted. A tree of n species has $n-3$ splits. The correct tree and the PhyChro tree both define a set of splits. In order to count the number of errors in the PhyChro tree, we can count how many splits of the correct tree are missing in the PhyChro tree. This gives us a number of errors, which is reported in Figure 10.2 for the yeast simulation. As can be seen, only 9% of trees have more than 1 error.

Another way to measure performance is to consider the total ratio between the correct splits in all trees and the total number of splits. In this case, we find that PhyChro gets 97% of the splits correctly for both the yeasts and vertebrates simulations.

10.4.3 Further analysis

To further understand what makes it difficult for PhyChro to reconstruct the tree, we decided to measure several parameters of the branches which PhyChro fails to reconstruct properly. Figure 10.3 shows the distribution of branch lengths (in the correct tree) in different colors depending on whether PhyChro find the corresponding split. As can be seen, small branches are more difficult to reconstruct. This is expected, since it means PhyChro has less information to get the split right.

This observation has led us to test what the performance of PhyChro would be if we increase the minimal number of events on a branch (see section 10.3.3). When we increase

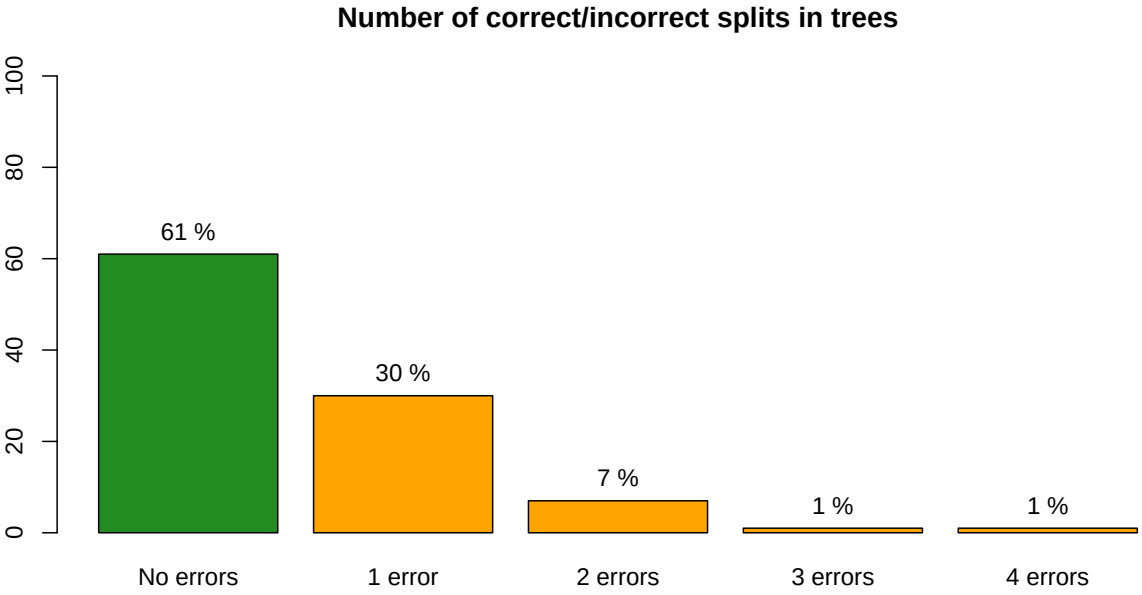

Figure 10.2: **PhyChro benchmark results for the yeast-like simulation.**
This histogram shows the number of incorrect splits in the 100 trees reconstructed by PhyChro.

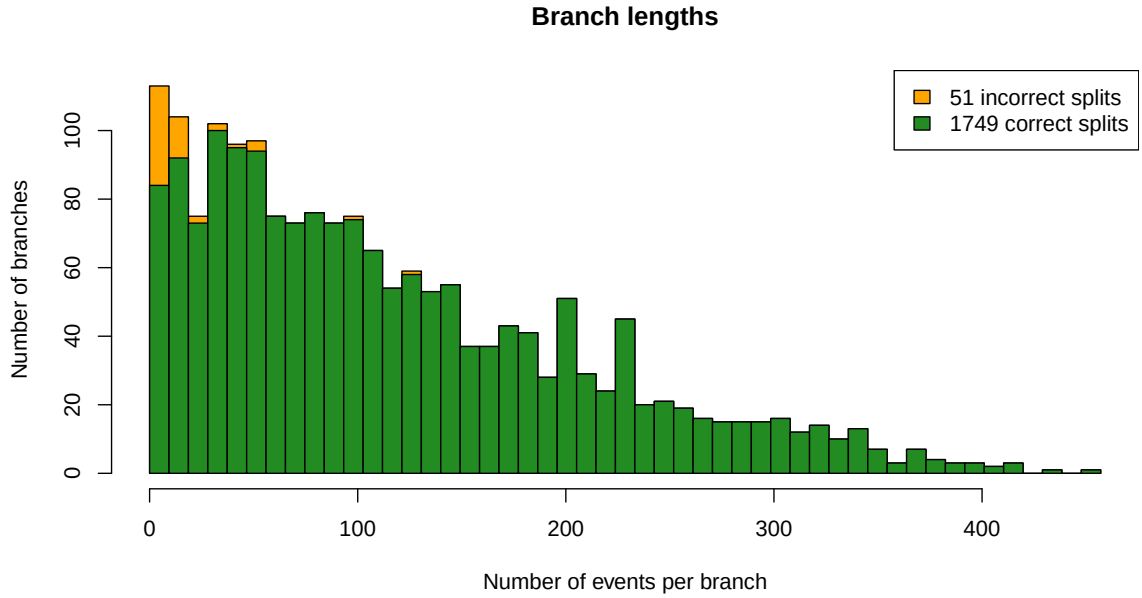

Figure 10.3: **Branch length distribution in simulated yeast trees.**
The histogram is colored differently depending on whether PhyChro has correctly inferred the split, i.e. whether the split is present in the tree it reconstructed.

it to 10 instead of 1, PhyChro is able to get 69% of the yeast trees correctly (instead of 61%), and 86% of the vertebrate trees correctly (instead of 79%). This confirms that this effect of small internal branches is significant.

Another test that is very interesting is to compare the confidence score given by PhyChro for correct and incorrect branches. The results are reported on Figure 10.4 for the yeast simulation. We can see that incorrect branches are, most of the times, given low confidence scores by PhyChro, which means PhyChro knows the branch is likely to be incorrect.

We made a last test which consisted in checking whether a certain type of event (inversion, duplication, etc.) was more associated with incorrectly inferred branches. However, none of the statistical tests were significant.

10.5 Conclusion

This simulation has enabled us to benchmark the PhyChro tree reconstruction algorithm. Overall, the algorithm performs remarkably well, deals properly with all types of rearrangements, and gives relevant confidence scores.

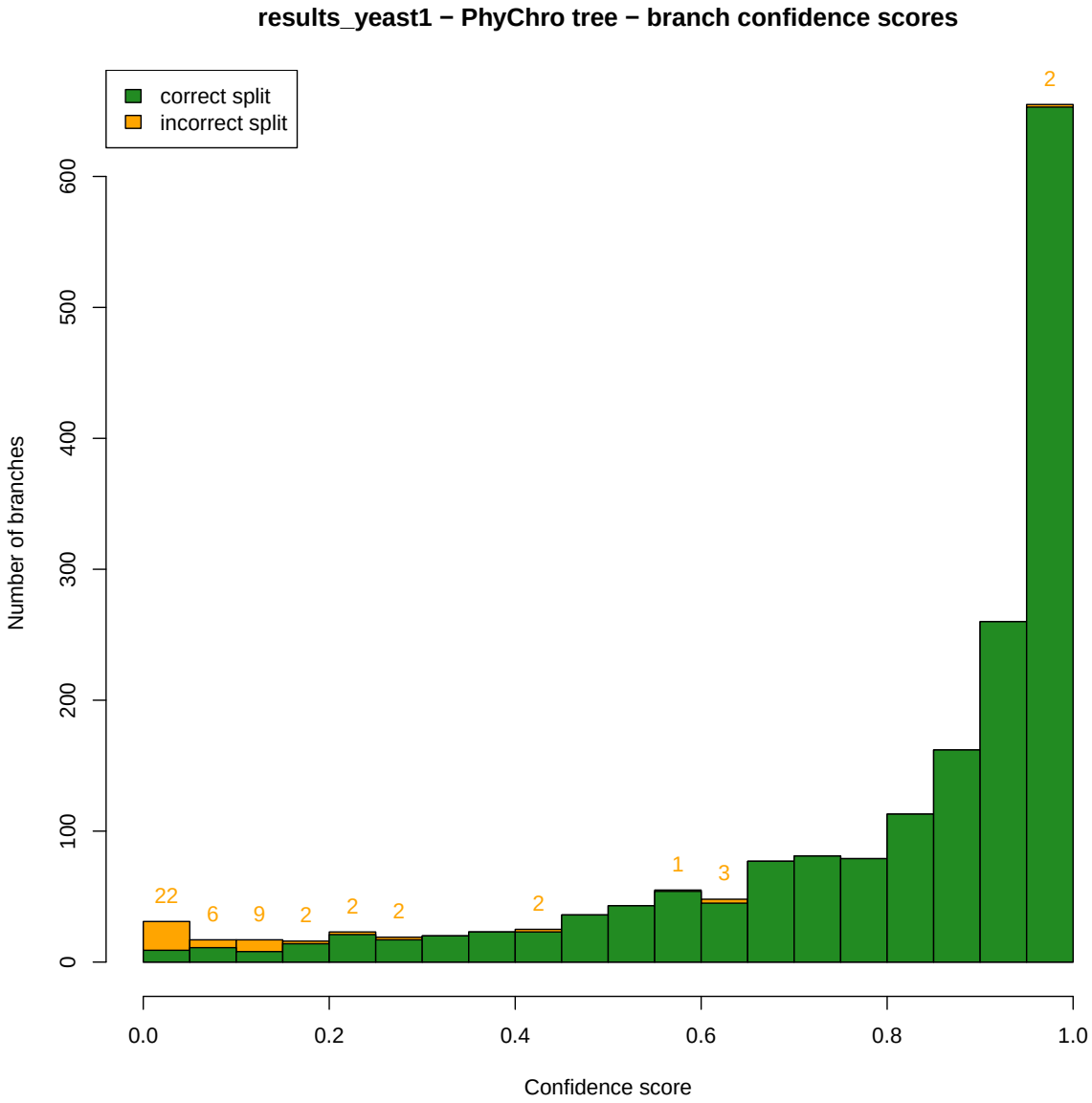

Figure 10.4: **Histogram of branch confidence scores given by PhyChro.**
The colors show whether the branch created by PhyChro corresponds to a correct split. The number of branches corresponding to the orange parts of the bars is written on top of them to make visualization easier.

Chapter 11

R-CLAG

Contents

In order to use coevolution analysis methods (discussed in the previous part), I needed to use the CLAG clustering algorithm, developed recently in the lab (Dib and Carbone, 2012a). However, CLAG is designed to be used as a standalone program and not as a library, and is written in the Perl programming language, so it is not very practical to use from the R statistical environment which I use for most of my work. Therefore, I decided to create a set of R functions that would automatically prepare the input and parse the output, so that is can be easily used like other classical methods like hierarchical clustering or k-means, which are available in R by default.

We then thought that this work could benefit the scientific community so I created an R package with the functions I had written. I then submitted this package to the official R package repository (CRAN), where it was reviewed and accepted. The R package is named "CLAG" in CRAN, but we refer to it as "R-CLAG" to distinguish it from the underlying program CLAG. The requirements for uploading a package mainly consist in providing an exhaustive documentation of the functions, code examples that can be automatically tested, and compatibility with the main operating systems. It is therefore now possible to install the R CLAG package easily from R.

Another purpose of R-CLAG was to add a few normalization methods that are useful to prepare the data before running CLAG.

11.1 What is CLAG?

CLAG, for clusters aggregation (Dib and Carbone, 2012a), is an unsupervised non hierarchical clustering algorithm especially adapted for datasets where the number of variables is of the same order of magnitude or (possibly much) higher than the number of observations. It has been applied successfully to biological data like gene expression matrices (to cluster experiments, which are usually few compared to the number of genes), correlation matrices for residues in protein families, and, more generally, to sets of data points described by multidimensional vectors. Compared to other clustering methods, CLAG has the following advantages:

- It automatically detects the number of clusters.
- It does not force all data points to cluster, so it clusters only elements which really show a common pattern.
- It works with datasets with a high number of variables, even if the number of data points is small.
- It is deterministic, making its results reproducible.
- In case we want to cluster a similarity matrix, it can use the actual values in the matrix to make clusters of similar elements, while ignoring groups of values that show no similarity.

CLAG was compared to other clustering methods in Dib and Carbone (2012a): hierarchical clustering (Ward, 1963), k-means (MacQueen et al., 1967) and the EM-based algorithm mclust (Fraley et al., 2012), all available in R. The main drawback of these methods is the need of predefining a number of clusters (k-means), the impossibility to distinguish clusters of observations represented by high values from those based on low values (only the equivalent of the CLAG "environment score" is captured but not of the CLAG "symmetry score", see later) (hierarchical clustering), the unsuccessful applicability to most biological datasets, where the number of variables is too high compared to the number of observations (mclust).

11.2 Description of the CLAG algorithm

CLAG input is a matrix A of N rows which are elements, or observations, and M columns which are characters, or variables. CLAG supports 3 kinds of matrices adapted to different types of data: i. real-valued; ii. discrete (each character takes its values in a finite set); iii. real-valued where the observations are a (possibly strict) subset of characters for which we want to compute the symmetric score (typically this applies to similarity matrices where $M_{i,j}$ is a similarity measure between observations i and j). On real-valued matrices (i and iii), CLAG works as follows (Dib and Carbone, 2012a):

1. It normalizes all values to [0, 1].

2. It computes the distribution of all values in the matrix and computes a histogram with bin boundaries corresponding to quantiles. The quantile range is to be set as a parameter $\Delta \in [0, 1]$. If some bins are abnormally large they are divided in smaller bins. A second histogram, with all bins moved by "half" the size of a bin, is computed.

3. It measures "proximity" of pairs of observations with respect to their characters, by counting the number of characters whose values are "close" to each other, that is, that fall in the same bin in at least one of the two histograms. This defines an "environmental" score between -1 (no values are closed to each other) and 1 (all values are close to each other).

4. (For data type iii only.) It calculates the symmetric score for pairs of observations (if the input is a similarity matrix, high symmetric scores account for high similarity).

5. It creates a first set of clusters (possibly overlapping, and possibly not fully covering the original set) based on the following rule: for a pair of observations V, Z, it clusters the observation W with them if W is close to (far from) V on the same characters for which V and Z are close to (far from) each other. This means clusters are formed of observations which are close to (far from) each other on the same subset of characters. This also implies all pairs of observations in a cluster have the same environmental score, which we can then define as the environmental score *of the cluster*. In the case of data type iii, an additional condition must be met by observations in the same cluster: they must have close similarity values in the original matrix.

6. CLAG retains clusters with an environmental score that passes a fixed threshold (to be set as a parameter).

7. It groups these clusters together by an aggregation step to form bigger and disjoint clusters called "key aggregates", constituting CLAG's output. The principle of this aggregation is to "solve" overlapping clusters by either merging them together, or by creating a third cluster containing the intersection.

In the case of discrete variables (ii), the first two steps are skipped and the notion of "proximity" is replaced by strict equality.

11.3 Normalization methods in R-CLAG

11.3.1 Theory

To analyze real-valued data, CLAG original version (Dib and Carbone, 2012a) proposes a data normalization constituted by a single affine transform (*global affine normalization*) applied to all values of the matrix, such that the minimum is mapped to 0 and the maximum to 1. The "proximity" of pairs of observations is then based on the distribution of all values in the matrix. This asks for all variables to have comparable magnitude and distribution. While this can be the case for many biologically relevant problems (like

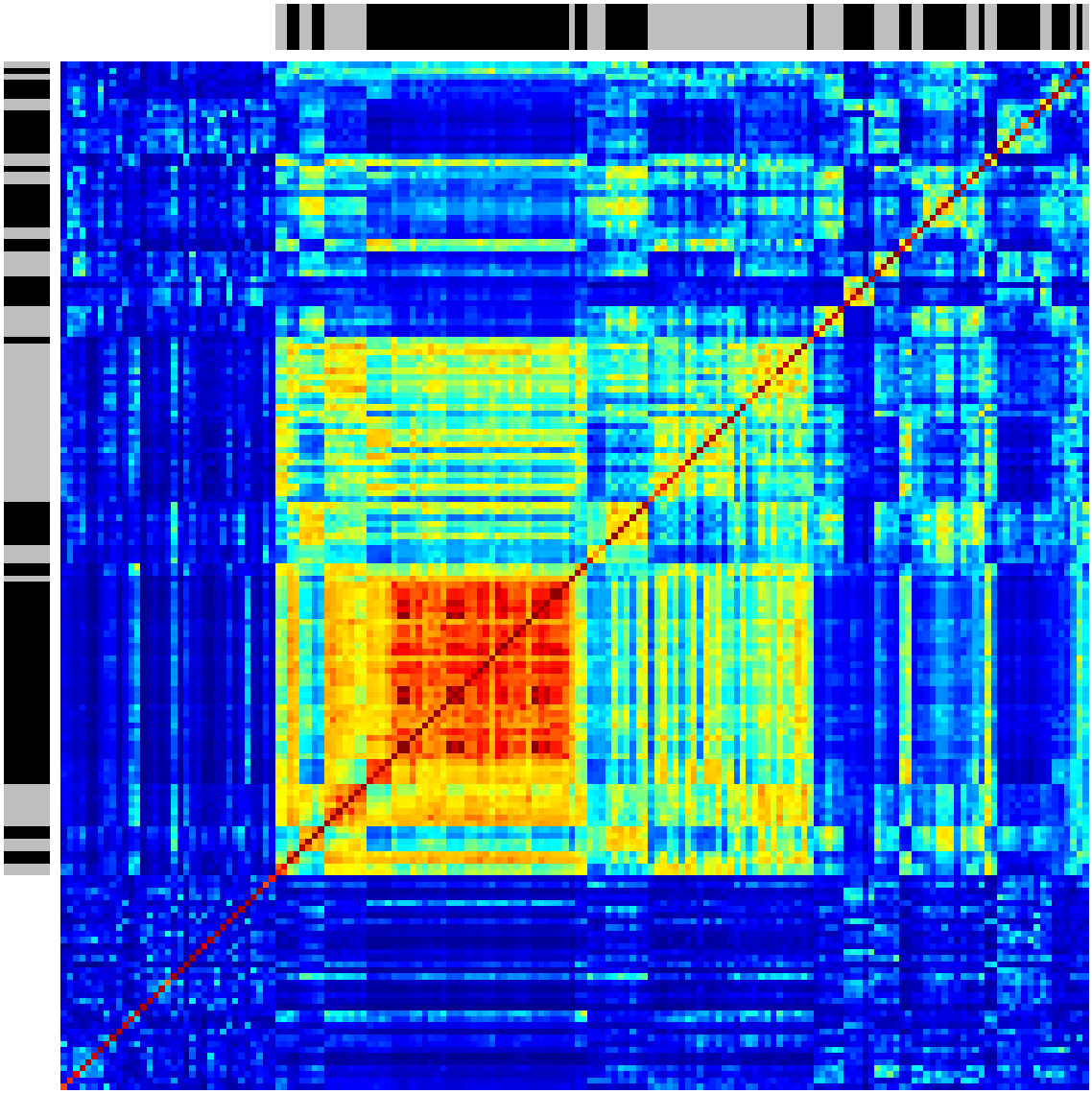

Figure 11.1: A symmetric matrix of coevolution scores between positions in the P53 protein sequence, produced with the MST method (Baussand and Carbone, 2009). The clustered matrix shows rows and columns grouping P53 positions in CLAG clusters. R-CLAG was run with a symmetric analysis (type iii) with $\Delta = 0.2$ and threshold = 0.5. Stripes with alternating colors at the left and top show cluster ranges. The non-colored part corresponds to unclustered positions. Colors from blue (low) to red (high) are assigned to different levels of coevolution.

expression levels or correlation matrices), it may not be the case when characters are represented by different units of measurement. This problem is also present in methods like k-means or hierarchical clustering based on a Euclidean distance or the L_1 norm, where differences or squared differences between different characters are added up. To help the user normalize the data properly, we propose two additional normalization methods. The first is a *character-specific affine normalization*, where we apply an affine transform to every character such that the minimum for this character is mapped to 0 and the maximum to 1. The second is a *character-specific rank normalization* where for every character, we replace the N values by their rank from 1 to N and scale it to $[0, 1]$. Different data require different normalization methods:

- If the matrix is symmetric (e.g. a correlation matrix like in Figure 11.1), global normalization must be used. Any character-based normalization would break the symmetry.

- If characters represent non-comparable values (e.g. if they are in different units of measurement), a character-based normalization should be used. If the distribution of every character is bounded between two extreme values and is roughly uniform (e.g. if they are all percentages), both character-based affine and rank normalization can be used. If any of these two conditions is not satisfied, character-based rank normalization should be used.

- If characters represent comparable values (e.g. if they are different genes expression levels), both global and character normalization may make sense. If the range of the distribution of some characters is smaller compared to other characters, global normalization will give them a lower weight, since they are compared to the distribution of all values from all characters. On the opposite, character-based normalization will rescale them so that range size is the same for all characters, so variation between values in characters with a small range will influence the analysis as much as variation on larger range characters.

11.3.2 Application to a toy example

The set `DIM128_subset` is provided with the R-CLAG package and contains 103 elements and 128 characters, belonging to 16 clusters (Figure 11.2B). It is a subset of the rows of the DIM128 dataset downloaded from http://cs.joensuu.fi/sipu/datasets/. The dataset was constructed by defining 16 points in the 128-dimensional space and by adding points around them by using a Gaussian distribution. All values belong to the range $[0, 250]$ (Figure 11.2A). The correct clustering is found by R-CLAG (Figure 11.2B) with $\Delta = 0.05$ and threshold 0 (default value), for all three normalization methods.

To show the purpose of the different normalization methods, we created two modified datasets from `DIM128_subset`. For the first, M_1, we applied a translation: for every character X_k, we added the constant $C_k = 250k$ to all values so that each character has a disjoint range (Figure 11.2C) since original values were in $[0, 250]$. With the default global normalization, the matrix is normalized to $[0, 1]$, and each character distribution has a range < 0.01. This means that all values for a character fall in at most two bins in the R-CLAG grid (histogram-like structure). Hence, all elements appear close to each other

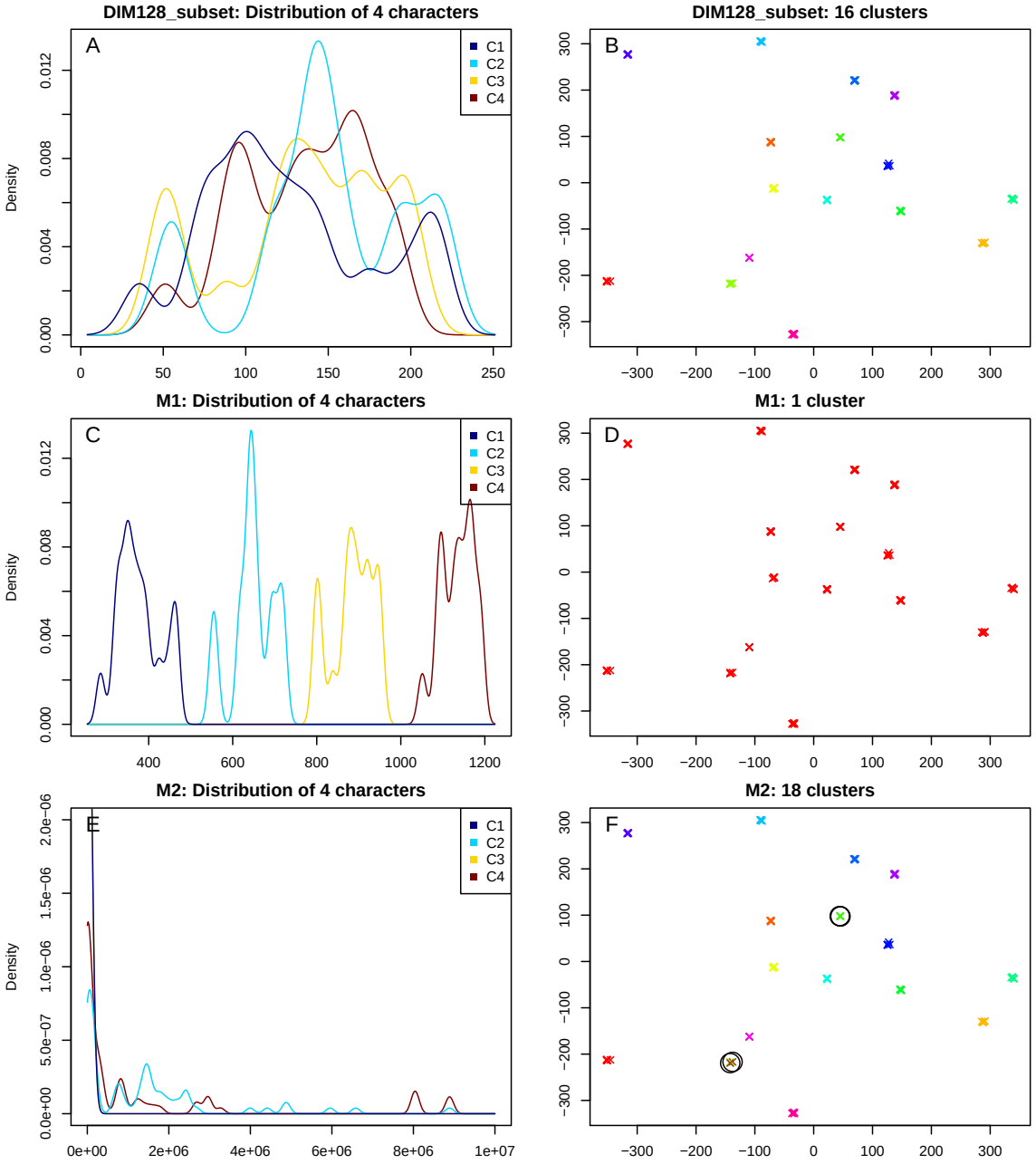

Figure 11.2: Panels A, C and E show the distribution of the first 4 characters in `DIM128_subset` and in the modified sets M_1 and M_2. Panels B, D, and F show the data points of the unmodified `DIM128_subset` projected along the first two axes of a principal component analysis for best visualization. Points are colored according to the clusters found using the global normalization, on the original dataset (B), M_1 (D) and M_2 (F). In B, all points are clustered correctly in the 16 clusters. In D, only a single cluster is found, while in F, 18 clusters are found (incorrectly clustered elements are circled).

and R-CLAG finds a single big cluster containing all of them (Figure 11.2D, Table 11.1). On the other hand, with a character-based normalization, every character is mapped to [0, 1], so every character is properly separated in different bins, making R-CLAG able to distinguish the points and infer the 16 clusters correctly. Rank normalization also works and R-CLAG finds all clusters.

	Normalization		
Data set	**Affine global**	**Affine character**	**Rank character**
original DIM128_subset	16 (correct)	16 (correct)	16 (correct)
M1 (translation)	1 (96 errors)	16 (correct)	16 (correct)
M2 (change in distribution)	18 (5 errors)	18 (3 errors)	16 (correct)

Table 11.1: Number of clusters found (16 is the correct number) for each of the three datasets described in the text and the three normalization methods. The number of incorrectly clustered elements (errors) is added in parentheses.

To test a set of characters with different distribution shapes, we applied the following transform to create a new set M_2: let X_k be the vector of character k, and define its new value as $X'_k = X_k + 250k$ if k is odd, and $X'_k = \exp(X_k/10) + 250k$ if k is even. Even-numbered characters are therefore distributed very differently from odd-numbered ones (Figure 11.2E). With this new matrix, the global affine normalization produces 5 incorrectly clustered elements (Figure 11.2F), while character-specific affine normalization produces 3 incorrectly clustered elements. The rank normalization, on the other hand, finds the 16 clusters correctly. This is expected, since the transform is monotonic, the ranks are unchanged and, therefore, the rank-normalized data is the same for all three datasets.

11.4 Clustering comparison in R-CLAG

R-CLAG provides the `mapClusterings` function to count the number of differently clustered elements in two different clustering C_1 and C_2. The function can be used to count the number of errors when comparing R-CLAG clustering to a known dataset partition as in Table 11.1. An option in `mapClusterings` allows the user to choose a built-in branch and bound algorithm or the R-function `solve_LSAP`. The `solve_LSAP` function implements a polynomial time algorithm and requires the installation of the R-package "clue" (Hornik, 2005).

An element is defined to be *differently clustered* if it lies in a cluster $c \in C_1$ but if it does not lie in the "corresponding" cluster $c' \in C_2$, where the correspondence is determined by a bijection f between a subset of C_1 and a subset of C_2 (we refer to subsets because some clusters in clustering C_1 may have no counterpart in C_2). Choosing the "wrong" f would result in a high number of elements belonging to different clusters in C_1 and C_2. Hence, f should be chosen to maximize the number of elements e such that e belongs to $c \in C_1$ and to $f(c) \in C_2$. The algorithm is implemented under the name `mapClusterings` in the package.

11.5 Conclusion

R-CLAG is user-friendly and it is developed in a widely-used statistics environment. It is available on all major platforms. Its normalization methods allow using it on heterogeneous real-valued matrices, its new analysis mode makes possible to analyze discrete matrices and it also provides an in-built function to compare different clusterings. These new features together make easy the application of CLAG to a wide range of biological data.

Bibliography

Adzhubei, I. A., Schmidt, S., Peshkin, L., Ramensky, V. E., Gerasimova, A., Bork, P., Kondrashov, A. S., and Sunyaev, S. R. (2010). A method and server for predicting damaging missense mutations. *Nature Methods*, 7(4):248–249.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403 – 410.

Baussand, J. and Carbone, A. (2009). A combinatorial approach to detect coevolved amino acid networks in protein families of variable divergence. *PLoS Comput Biol*, 5(9):e1000488.

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 289–300.

Borde, V., Robine, N., Lin, W., Bonfils, S., Geli, V., and Nicolas, A. (2009). Histone H3 lysine 4 trimethylation marks meiotic recombination initiation sites. *EMBO Journal*, 28(2):99–111.

Capriotti, E., Fariselli, P., and Casadio, R. (2005). I-mutant2.0: predicting stability changes upon mutation from the protein sequence or structure. *Nucleic Acids Research*, 33(suppl 2):W306–W310.

Capriotti, E., Fariselli, P., Rossi, I., and Casadio, R. (2008). A three-state prediction of single point mutations on protein stability changes. *BMC Bioinformatics*, 9(Suppl 2):S6.

Carbone, A., Zinovyev, A., and Képès, F. (2003). Codon adaptation index as a measure of dominating codon bias. *Bioinformatics*, 19(16):2005–2015.

Chang, D. T.-H., Huang, C.-Y., Wu, C.-Y., and Wu, W.-S. (2011). YPA: an integrated repository of promoter features in saccharomyces cerevisiae. *Nucleic Acids Research*, 39(suppl 1):D647–D652.

Chen, W., Feng, P., Lin, H., and Chou, K. (2013). irspot-psednc: identify recombination spots with pseudo dinucleotide composition. *Nucleic Acids Res.*, 41(6):e68.

Cheng, J., Randall, A., and Baldi, P. (2006). Prediction of protein stability changes for single-site mutations using support vector machines. *Proteins: Structure, Function, and Bioinformatics*, 62(4):1125–1132.

Cho, Y., Gorina, S., Jeffrey, P. D., and Pavletich, N. P. (1994). Crystal structure of a p53 tumor suppressor-dna complex: Understanding tumorigenic mutations. *Science*, 265(5170):pp. 346–355.

Combet, C., Garnier, N., Charavay, C., Grando, D., Crisan, D., Lopez, J., Dehne-Garcia, A., Geourjon, C., Bettler, E., Hulo, C., Mercier, P. L., Bartenschlager, R., Diepolder, H., Moradpour, D., Pawlotsky, J.-M., Rice, C. M., Trépo, C., Penin, F., and Deléage, G. (2007). euhcvdb: the european hepatitis c virus database. *Nucleic Acids Research*, 35(suppl 1):D363–D366.

Consortium, T. U. (2014). Activities at the universal protein resource (uniprot). *Nucleic Acids Research*, 42(D1):D191–D198.

Cook, G. A., Dawson, L. A., Tian, Y., and Opella, S. J. (2013). Three-dimensional structure and interaction studies of hepatitis C virus p7 in 1,2-dihexanoyl-sn-glycero-3-phosphocholine by solution nuclear magnetic resonance. *Biochemistry*, 52(31):5295–5303.

Darwin, C. (1859). On the origin of species. *London: Murray*.

Darwin, C. and Wallace, A. (1858). On the tendency of species to form varieties; and on the perpetuation of varieties and species by natural means of selection. *Journal of the proceedings of the Linnean Society of London. Zoology*, 3(9):45–62.

Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 233–240, New York, NY, USA. ACM.

de Castro, E., Soriano, I., Marín, L., Serrano, R., Quintales, L., and Antequera, F. (2012). Nucleosomal organization of replication origins and meiotic recombination hotspots in fission yeast. *The EMBO Journal*, 31(1):124–137.

Dekker, J. P., Fodor, A., Aldrich, R. W., and Yellen, G. (2004). A perturbation-based method for calculating explicit likelihood of evolutionary co-variance in multiple sequence alignments. *Bioinformatics*, 20(10):1565–1572.

Dib, L. and Carbone, A. (2012a). CLAG: an unsupervised non hierarchical clustering algorithm handling biological data. *BMC Bioinformatics*, 13(1):194.

Dib, L. and Carbone, A. (2012b). Protein fragments: Functional and structural roles of their coevolution networks. *PLoS ONE*, 7(11):e48124.

Donehower, L. A., Harvey, M., Slagle, B. L., McArthur, M. J., Montgomery, C. A., Butel, J. S., et al. (1992). Mice deficient for p53 are developmentally normal but susceptible to spontaneous tumours. *Nature*.

Drillon, G., Carbone, A., and Fischer, G. (2014). SynChro: A fast and easy tool to reconstruct and visualize synteny blocks along eukaryotic chromosomes. *PLoS ONE*, 9(3):e92621.

Drillon, G. and Fischer, G. (2011). Comparative study on synteny between yeasts and vertebrates. *Comptes Rendus Biologies*, 334(8–9):629 – 638. Ten years of genomic exploration in eukaryotes : strategy and progress of Genolevures Dix ans d'exploration genomique chez les eucaryotes : la strategie et les avancees de Genolevures.

Dujon, B. (1996). The yeast genome project: what did we learn? *Trends Genet.*, 12(7):263 – 270.

Edgar, R. C. (2004). Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797.

Edlund, K., Larsson, O., Ameur, A., Bunikis, I., Gyllensten, U., Leroy, B., Sundström, M., Micke, P., Botling, J., and Soussi, T. (2012). Data-driven unbiased curation of the tp53 tumor suppressor gene mutation database and validation by ultradeep sequencing of human tumors. *Proceedings of the National Academy of Sciences*, 109(24):9551–9556.

Finn, R. D., Bateman, A., Clements, J., Coggill, P., Eberhardt, R. Y., Eddy, S. R., Heger, A., Hetherington, K., Holm, L., Mistry, J., Sonnhammer, E. L. L., Tate, J., and Punta, M. (2014). Pfam: the protein families database. *Nucleic Acids Research*, 42(D1):D222–D230.

Flanagan, S. E., Patch, A.-M., and Ellard, S. (2010). Using sift and polyphen to predict loss-of-function and gain-of-function mutations. *Genetic testing and molecular biomarkers*, 14(4):533–537.

Fowler, K. R., Sasaki, M., Milman, N., Keeney, S., and Smith, G. R. (2014). Evolutionarily diverse determinants of meiotic dna break and recombination landscapes across the genome. *Genome Research*.

Fraley, C., Raftery, A. E., Murphy, T. B., and Scrucca, L. (2012). Mclust version 4 for r: Normal mixture modeling for model-based clustering, classification, and density estimation. Technical Report 597, Department of Statistics, University of Washington.

Glynn, E. F., Megee, P. C., Yu, H.-G., Mistrot, C., Unal, E., Koshland, D. E., DeRisi, J. L., and Gerton, J. L. (2004). Genome-wide mapping of the cohesin complex in the yeast *saccharomyces cerevisiae*. *PLoS Biol*, 2(9):e259.

Gordon, J. L., Byrne, K. P., and Wolfe, K. H. (2009). Additions, losses, and rearrangements on the evolutionary route from a reconstructed ancestor to the modern *Saccharomyces cerevisiae* genome. *PLoS Genet*, 5(5):e1000485.

Guindon, S., Dufayard, J.-F., Lefort, V., Anisimova, M., Hordijk, W., and Gascuel, O. (2010). New algorithms and methods to estimate maximum-likelihood phylogenies: Assessing the performance of phyml 3.0. *Systematic Biology*, 59(3):307–321.

Harris, F. J. (1978). On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1):51–83.

Henikoff, S. and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919.

Hollingsworth, N. M. and Ponte, L. (1997). Genetic interactions between hop1, red1 and mek1 suggest that mek1 regulates assembly of axial element components during meiosis in the yeast saccharomyces cerevisiae. *Genetics*, 147(1):33–42.

Hornik, K. (2005). A clue for cluster ensembles. *Journal of Statistical Software*, 14(12):1–25.

Jacks, T., Remington, L., Williams, B. O., Schmitt, E. M., Halachmi, S., Bronson, R. T., and Weinberg, R. A. (1994). Tumor spectrum analysis in p53-mutant mice. *Current Biology*, 4(1):1 – 7.

Jiang, P., Wu, H., Wei, J., Sang, F., Sun, X., , and Lu, Z. (2007). RF-DYMHC: detecting the yeast meiotic recombination hotspots and coldspots by random forest model using gapped dinucleotide composition features. *Nucleic Acids Res.*, 35(Web Server issue):W47–W51.

Junier, I., Herisson, J., and Kepes, F. (2010). Periodic pattern detection in sparse boolean sequences. *Algorithms for Molecular Biology*, 5(1):31.

Katoh, K., Misawa, K., Kuma, K., and Miyata, T. (2002). Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform. *Nucleic Acids Research*, 30(14):3059–3066.

Koonin, E. V. (2005). Orthologs, paralogs, and evolutionary genomics. *Annual Review of Genetics*, 39(1):309–338. PMID: 16285863.

Liu, G., Liu, J., Cui, X., and Cai, L. (2012). Sequence-dependent prediction of recombination hotspots in *saccharomyces cerevisiae*. *Journal of Theoretical Biology*, 293:49 – 54.

Livingstone, C. D. and Barton, G. J. (1993). Protein sequence alignments: a strategy for the hierarchical analysis of residue conservation. *Computer applications in the biosciences : CABIOS*, 9(6):745–756.

Lockless, S. W. and Ranganathan, R. (1999). Evolutionarily conserved pathways of energetic connectivity in protein families. *Science*, 286(5438):295–299.

Lodish, H., Berk, A., and Zipursky, S. (2000). *Molecular Cell Biology 4th edition*. National Center for Biotechnology InformationÕs Bookshelf.

Lonquety, M., Lacroix, Z., Papandreou, N., and Chomilier, J. (2009). Sprouts: a database for the evaluation of protein stability upon point mutation. *Nucleic Acids Research*, 37(suppl 1):D374–D379.

Loytynoja, A. and Goldman, N. (2010). webprank: a phylogeny-aware multiple sequence aligner with interactive alignment browser. *BMC Bioinformatics*, 11(1):579.

MacQueen, James, et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, page 14. California, USA.

Mancera, E., Bourgon, R., Brozzi, A., Huber, W., and Steinmetz, L. (2008). High-resolution mapping of meiotic crossovers and non-crossovers in yeast. *Nature*, 454(7203):479–485.

Mathelier, A. and Carbone, A. (2010). Chromosomal periodicity and positional networks of genes in Escherichia coli. *Molecular Systems Biology*, 6.

Miescher-Rüsch, F. (1871). *Ueber die chemische Zusammensetzung der Eiterzellen*. Verlag von August Hirschwald.

Mills, A. A., Zheng, B., Wang, X.-J., Vogel, H., Roop, D. R., and Bradley, A. (1999). p63 is a p53 homologue required for limb and epidermal morphogenesis. *Nature*, 398(6729):708–713.

Morcos, F., Pagnani, A., Lunt, B., Bertolino, A., Marks, D. S., Sander, C., Zecchina, R., Onuchic, J. N., Hwa, T., and Weigt, M. (2011). Direct-coupling analysis of residue coevolution captures native contacts across many protein families. *Proceedings of the National Academy of Sciences*, 108(49):E1293–E1301.

Nair, P. S. and Vihinen, M. (2013). Varibench: A benchmark database for variations. *Human Mutation*, 34(1):42–49.

Ng, P. C. and Henikoff, S. (2001). Predicting deleterious amino acid substitutions. *Genome Research*, 11(5):863–874.

Pan, J., Sasaki, M., Kniewel, R., Murakami, H., Blitzblau, H. G., Tischfield, S. E., Zhu, X., Neale, M. J., Jasin, M., Socci, N. D., Hochwagen, A., and Keeney, S. (2011). A hierarchical combination of factors shapes the genome-wide topography of yeast meiotic recombination initiation. *Cell*, 144(5):719 – 731.

Panizza, S., Mendoza, M., Berlinger, M., Huang, L., Nicolas, A., Shirahige, K., and Klein, F. (2011). Spo11-accessory proteins link double-strand break sites to the chromosome axis in early meiotic recombination. *Cell*, 146(3):372 – 383.

Penn, O., Privman, E., Ashkenazy, H., Landan, G., Graur, D., and Pupko, T. (2010). Guidance: a web server for assessing alignment confidence scores. *Nucleic Acids Research*, 38(suppl 2):W23–W28.

Pérez-Ortín, J., Alepuz, P., and Moreno, J. (2007). Genomics and gene transcription kinetics in yeast. *TRENDS in Genetics*, 23(5):250–257.

Pokholok, D. K., Harbison, C. T., Levine, S., Cole, M., Hannett, N. M., Lee, T. I., Bell, G. W., Walker, K., Rolfe, P. A., Herbolsheimer, E., Zeitlinger, J., Lewitter, F., Gifford, D. K., and Young, R. A. (2005). Genome-wide map of nucleosome acetylation and methylation in yeast. *Cell*, 122(4):517 – 527.

Privman, E., Penn, O., and Pupko, T. (2012). Improving the performance of positive selection inference by filtering unreliable alignment regions. *Molecular Biology and Evolution*, 29(1):1–5.

Pruitt, K. D., Tatusova, T., and Maglott, D. R. (2007). Ncbi reference sequences (refseq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Research*, 35(suppl 1):D61–D65.

R Development Core Team (2011). *R: A Language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Reva, B., Antipin, Y., and Sander, C. (2011). Predicting the functional impact of protein mutations: application to cancer genomics. *Nucleic Acids Research*, 39(17):e118.

Rogato, A., Richard, H., Sarazin, A., Voss, B., Cheminant Navarro, S., Champeimont, R., Navarro, L., Carbone, A., Hess, W., and Falciatore, A. (2014). The diversity of small non-coding RNAs in the diatom Phaeodactylum tricornutum. *BMC Genomics*, 15(1):698.

Saitou, N. and Nei, M. (1987). The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425.

Sayers, E. W., Barrett, T., Benson, D. A., Bolton, E., Bryant, S. H., Canese, K., Chetvernin, V., Church, D. M., DiCuccio, M., Federhen, S., Feolo, M., Fingerman, I. M., Geer, L. Y., Helmberg, W., Kapustin, Y., Landsman, D., Lipman, D. J., Lu, Z., Madden, T. L., Madej, T., Maglott, D. R., Marchler-Bauer, A., Miller, V., Mizrachi, I., Ostell, J., Panchenko, A., Phan, L., Pruitt, K. D., Schuler, G. D., Sequeira, E., Sherry, S. T., Shumway, M., Sirotkin, K., Slotta, D., Souvorov, A., Starchenko, G., Tatusova, T. A., Wagner, L., Wang, Y., Wilbur, W. J., Yaschenko, E., and Ye, J. (2011). Database resources of the national center for biotechnology information. *Nucleic Acids Research*, 39(suppl 1):D38–D51.

Schymkowitz, J., Borg, J., Stricher, F., Nys, R., Rousseau, F., and Serrano, L. (2005). The foldx web server: an online force field. *Nucleic Acids Research*, 33(suppl 2):W382–W388.

Shannon, C. (1948). A mathematical theory of communication. *Bell System Technical Journal, The*, 27(3):379–423.

Shannon, C. E. and Weaver, W. (1949). *Themathematical theory of communication*.

Shihab, H. A., Gough, J., Cooper, D. N., Day, I. N. M., and Gaunt, T. R. (2013). Predicting the functional consequences of cancer-associated amino acid substitutions. *Bioinformatics*, 29(12):1504–1510.

Smagulova, F., Gregoretti, I. V., Brick, K., Khil, P., Camerini-Otero, R. D., and Petukhova, G. V. (2011). Genome-wide analysis reveals novel molecular features of mouse recombination hotspots. *Nature*, 472(7343):375–378.

Sokal, R. R. (1958). A statistical method for evaluating systematic relationships. *Univ Kans Sci Bull*, 38:1409–1438.

Sunyaev, S. R., Eisenhaber, F., Rodchenkov, I. V., Eisenhaber, B., Tumanyan, V. G., and Kuznetsov, E. N. (1999). Psic: profile extraction from sequence alignments with

position-specific counts of independent observations. *Protein Engineering*, 12(5):387–394.

Taylor, W. (1997). Residual colours: a proposal for aminochromography. *Protein Engineering*, 10(7):743–746.

Thorvaldsdóttir, H., Robinson, J. T., and Mesirov, J. P. (2013). Integrative genomics viewer (IGV): high-performance genomics data visualization and exploration. *Briefings in Bioinformatics*, 14(2):178–192.

Tkačik, G. and Bialek, W. (2009). Diffusion, dimensionality, and noise in transcriptional regulation. *Physical Review E*, 79:051901.

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244.

Watson, J. D., Crick, F. H., et al. (1953). Molecular structure of nucleic acids. *Nature*, 171(4356):737–738.

Woltering, D., Baumgartner, B., Bagchi, S., Larkin, B., Loidl, J., de los Santos, T., and Hollingsworth, N. M. (2000). Meiotic segregation, synapsis, and recombination checkpoint functions require physical interaction between the chromosomal proteins red1p and hop1p. *Molecular and Cellular Biology*, 20(18):6646–6658.

Wright, M., Kharchenko, P., Church, G., and Segrè, D. (2007). Chromosomal periodicity of evolutionarily conserved gene pairs. *Proceedings of the National Academy of Sciences of the United States of America*, 104(25):10559–10564.

Yang, A., Schweitzer, R., Sun, D., Kaghad, M., Walker, N., Bronson, R. T., Tabin, C., Sharpe, A., Caput, D., Crum, C., et al. (1999). p63 is essential for regenerative proliferation in limb, craniofacial and epithelial development. *Nature*, 398(6729):714–718.

Yang, A., Walker, N., Bronson, R., Kaghad, M., Oosterwegel, M., Bonnin, J., Vagner, C., Bonnet, H., Dikkes, P., Sharpe, A., et al. (2000). p73-deficient mice have neurological, pheromonal and inflammatory defects but lack spontaneous tumours. *Nature*, 404(6773):99–103.

Zickler, D. and Kleckner, N. (1999). Meiotic chromosomes: Integrating structure and function. *Annual Review of Genetics*, 33(1):603–754. PMID: 10690419.