IUT de Perpignan Via Domitia
Dpt STID (Carcassonne)

Laboratoire de Génétique Cellulaire
INRA, Castanet-Tolosan

Analyse de données transcriptomiques par réseaux de co-expression génique

Nicolas Edwards

Stage de fin d'études - DUT STID Carcassonne
2011/2012

Tutrices de stage : Magali SanCristobal, Laurence Liaubet (LGC, INRA) & Nathalie Viguerie (INSERM) Tutrice pédagogique : Nathalie Villa-Vialaneix

Table des matières

Résumé :
Ce stage de fin de DUT STID de 10 semaines, du 2 avril au 8 juin 2012, a été effectué au sein de l'INRA de Toulouse (31) ; c'est un organisme public qui a pour but la recherche agronomique. Plus précisément, j'étais intégré au Laboratoire de Génétique Cellulaire (UMR0444) dont la recherche est axée sur la compréhension du génome des animaux domestiqués. Le stage a été mené dans le cadre de deux projets de recherche : « DeLiSus », un projet financé par l'ANR (Agence Nationale de la Recherche) sur la génétique du cochon, et « Diogènes », un projet financé par l'Union Européenne pour l'étude de l'obésité. Il s'est organisé autour de groupes de travail avec des statisticiens et des biologistes.

Le but du stage était d'analyser les corrélations multiples dans l'expression de gènes par une approche « réseau ». Le travail de ce stage a été effectué en utilisant le logiciel statistique libre R : des scripts génériques et des fonctions ont été développés dans le but de pouvoir être réutilisés pour d'autres données de même nature. Différentes approches mathématiques d'inférence de réseaux ont été comparées, sur des données réelles et simulées, et les graphes retenus ont été validés par les biologistes pour vérifier que l'information apportée était pertinente. In fine, ce travail a permis d'extraire des gènes importants dans le fonctionnement du tissu adipeux, renforçant la connaissance biologique de ce domaine.

Une partie du travail a été intégrée dans un article en révision pour publication dans le journal scientifique *PLoS Genetics* [Viguerie et al., 2012], et l'ensemble des résultats de la comparaison seront exposés lors des premières Rencontres R de BoRdeaux[1], le 3 juillet 2012 [Villa-Vialaneix et al., 2012].

Ce stage m'a permis de m'approprier un nouveau modèle mathématique, de développer ma rigueur personnelle en programmation, et d'acquérir de la confiance en moi, notamment lors des groupes de travail où je devais régulièrement exposer l'avancée de mes travaux.

[1] http://r2012.bordeaux.inria.fr

Abstract:

This 10 week work placement, took place at the Toulouse section of the INRA, between April 2nd and June 8th, 2012; the INRA is a public institute that does research on themes related to agronomy. More specifically, I was at the "Laboratoire de Génétique Cellulaire" (UMR0444), whose aim is to understand the genome of domesticated animals. This study was done as part of two research projects: "DeLiSus" (whose goal is to understand pig transcriptome), financed by the National Research Agency (ANR), and "Diogènes" (fighting obesity), financed by the European Union; thus, regular meetings and workgroups were scheduled during the internship.

The objective was to analyse multiple correlations between gene expression data through a "network" approach. The work was done using R, an open-source statistics program: generic scripts and functions were written, in such a way that they can be re-used with similar data. Several mathematical network inference approaches were compared on real and simulated data, and the best graphs were validated by biologists to check that the information contained inside them was relevant. In the end, genes that play an important part in how adipose tissue works were extracted from the data, reinforcing biological knowledge of this field.

A part of the results obtained during this placement are in an article submitted for publication in the scientific journal *PLoS Genetics*, and the work done on method comparison will be exposed at the *premières rencontres R*, at BoRdeaux[2] on July 3rd, 2012.

[2] http://r2012.bordeaux.inria.fr

Première partie .

Introduction

Au cours de ma seconde année de DUT STID, j'ai eu la possibilité d'effectuer un stage de 10 semaines, nécessaire pour valider mon diplôme. Intéressé par les sciences depuis longtemps, je souhaitais effectuer mon stage dans un domaine d'application scientifique. Or, mon projet de statistique de deuxième année portait sur des données proposées par des chercheurs de l'INRA ; l'étude de ces données à l'aide de techniques statistiques diverses m'ayant beaucoup plu, j'ai décidé de postuler pour mon stage de fin de DUT au sein de l'INRA de Toulouse, où j'ai été retenu. Ainsi, j'ai été intégré au Laboratoire de Génétique Cellulaire (UMR0444), dont la recherche est axée sur la compréhension du génome des animaux domestiqués. Le stage a été mené dans le cadre de deux projets de recherche : « DeLiSus », un projet financé par l'ANR (Agence Nationale de la Recherche) sur la génétique du cochon, et « Diogènes », un projet financé par l'Union Européenne pour l'étude de l'obésité. Il s'est organisé autour de groupes de travail avec des statisticiens et des biologistes.

Mon stage m'a permis de travailler dans des domaines de recherche relativements neufs, car seules les nouvelles technologies permettent de les étudier pleinement. C'est le cas de la recherche sur le transcriptome, c'est à dire l'expression du génome. Il conditionne la production de protéines donc le phénotype de l'individu. Dans ce rapport, on s'intéresse à la valeur de l'expression d'une multitude de gènes (autrement dit à la quantité d'ARNm que chacun produit à un instant donné dans une cellule donnée), en modélisant l'ensemble des corrélations entre ces gènes par des *réseaux de co-expression génique*. Dans ce contexte, j'ai été amené à écrire des scripts pour construire et analyser de tels objets, avec diverses méthodologies correspondant à des modèles statistiques différents.

Dans la suite de ce rapport, je présenterai dans un premier temps ma structure d'accueil dans la section II. Ensuite je décrirai les données que l'on m'a confiées et la problématique de mon stage dans la section 3, avant d'aborder les outils et méthodes utilisés pour ce travail dans la section 4. Enfin j'exposerai les résultats d'un point de vue statistique, ainsi que les conclusions biologiques qui en ont été tirées dans la section 5.

Deuxième partie .

Description de l’entreprise d’accueil

1. L'INRA

L'INRA[1] est un Etablissement Public à caractère Scientifique et Technologique (EPST, au même titre que le CNRS ou que l'INSERM), essentiellement financé par des fonds publics (il rend compte de son activité et de sa gestion à ses ministères de tutelle, le ministère de l'Enseignement supérieur et de la Recherche et le ministère de l'Alimentation, de l'Agriculture et de la Pêche). L'INRA compte environ 400 unités de recherche réparties dans 19 centres (localisations) et 14 départements (grandes thématiques) de recherche (une unité correspond à un centre et a un département de rattachement). Les unités expérimentales de l'INRA couvrent environ 12 000 hectares dont 3 000 hectares de forêts. Parmi le cheptel de l'INRA, on peut compter environ 6000 bovins, 16 000 ovins, 8 000 porcins, 300 équins, 34 000 volailles, une centaine de cervidés et une dizaine de lamas. L'INRA renforce ses activités autour de trois champs :

1. le développement d'une agriculture durable,
2. l'alimentation et son rôle sur la santé humaine
3. l'environnement et les territoires

et a pour missions de produire et diffuser des connaissances scientifiques et des innovations, contribuer à la formation et, par la recherche, à la diffusion de la culture scientifique et au débat science/société, participer par son expertise à éclairer les décisions des acteurs publics et privés.

Les quatre priorités de recherche de l'INRA sont :

1. protéger les ressources naturelles,
2. manger sain et sûr,
3. passer des génomes aux populations végétales et animales,
4. travailler avec l'informatique et la biologie à haut débit.

L'organigramme de l'INRA est fourni dans la figure 1.1.

[1] http://www.inra.fr

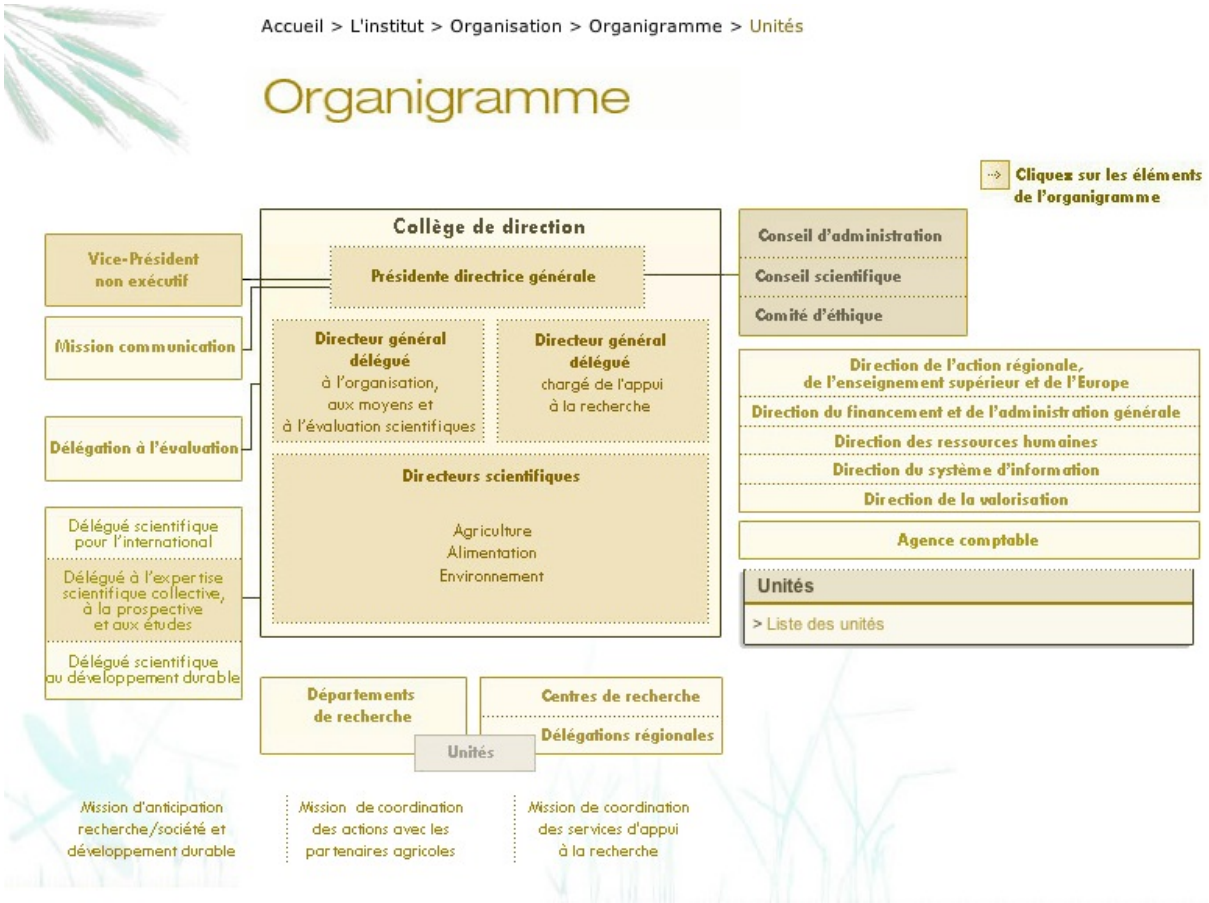


FIGURE 1.1.: Organigramme de l'INRA

2. Le Laboratoire de Génétique Cellulaire

Mon département de rattachement était le département Génétique Animale, et mon centre de rattachement était Toulouse (campus d'Auzeville, 31). Les recherches efffectuées au département GA se font principalement autour de trois thématiques, qui correspondent à trois échelles du vivant : la structure du génome, l'analyse de la variabilité génotypique, et les méthodes de gestion des populations. Il se compose de :

3 Unités de Recherches,

7 Unités Mixtes de Recherche (mixte puisque en association avec des écoles, des universités ou d'autres organismes de recherche) ; c'est au sein de l'UMR 0444 (Laboratoire de Génétique Cellulaire, LGC) que mon stage a été effectué,

12 Unités Expérimentales,

3 Unités de Service.

Le département de génétique animale est présent principalement en région parisienne (50% des chercheurs, à Toulouse (35 % des chercheurs) et en Guadeloupe, comme l'illustre la carte 2.1 de l'implémentation GA en France.

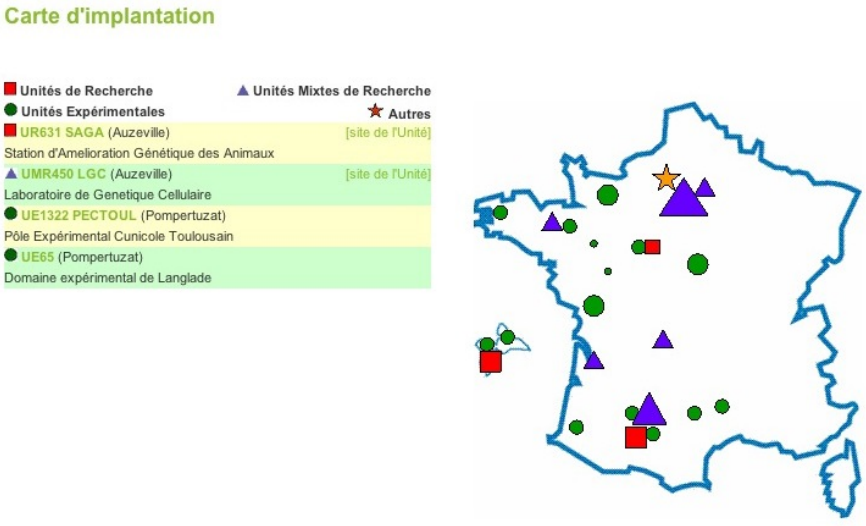


FIGURE 2.1.: Carte de l'implantation du département Génétique Animale

La vocation du Laboratoire de Génétique Cellulaire (LGC, l'endroit où ce stage a été effectué) est l'étude du génome des espèces animales domestiques, dans ses aspects fonctionnels et structuraux.

Le stage a été encadré par Magali SanCristobal (Directrice de Recherches, Statistique, LGC, INRA), Nathalie Villa-Vialaneix (Maîtresse de Conférences, Statistique, UPVD, IUT, département STID & équipe SAMM, Université Paris 1), Laurence Liaubet (Chargée de Recherche, Biologie, LGC, INRA) et Nathalie Viguerie (Chargée de Recherche, Biologie, Obesity Research Laboratory, INSERM). Le travail effectué s'inscrit dans le cadre de deux gros projets de recherche :

- Le projet DéLiSus financé par l'ANR[1] ayant pour but l'étude de la variabilité haplotypique du génome porcin à haute densité.
- Le projet Diogènes[2], projet international de lutte contre l'obésité, qui est subventionné par l'Union Européenne.

Pour permettre une meilleure interaction entre les divers acteurs, une mise en commun périodique des résultats a été effectuée, au travers de groupes de travail. Ces groupes de travail étaient organisés avec les statisticiens et les biologistes de l'INRA et de l'INSERM : la présentation des résultats et la mise en commun des réflexions et connaissances de chacun a permis d'apporter régulièrement des modifications, comme des fonctionnalités supplémentaires aux scripts écrits, des modifications des fichiers de données ou le test de nouvelles approches méthodologiques.

[1] http://www.agence-nationale-recherche.fr, projet Delisus ; voir : http://www.inra.fr/les_partenariats/programmes_anr/genomique/genanimal/appel_a_projets_2007/delisus

[2] http://www.diog\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{e\global\mathchardef\accent@spacefactor\spacefactor}\accent18e\egroup\spacefactor\accent@spacefactornes-eu.org

Troisième partie .

Description du travail effectué

3. Description des données et de la problématique

3.1. Description des données

Durant le stage, l'étude de plusieurs jeux de données a été effectuée, de manière indépendante les uns des autres. Pour chacun d'entre eux, on s'intéressait à la différence dans la corrélation entre l'expression de gènes entre deux conditions environnementales.

3.1.1. Quelques notions biologiques sur l'expression de gènes

La majorité des données utilisées dans ce travail se composent d'expressions de gènes. Pour expliquer ce qu'est l'expression d'un gène et comment elle se mesure, cette section présente quelques notions de biologie et de génétique.

L'ADN

L'acide désoxyribonucléique, plus connu sous le nom d'ADN, est une molécule retrouvée dans toutes les cellules vivantes, qui renferme l'ensemble des informations nécessaires au développement et au fonctionnement d'un organisme. L'ADN est composé de séquences de nucléotides. Chaque nucléotide est constituée de trois éléments liés entre eux :

- un groupe phosphate lié à :
- un sucre, le désoxyribose, lui-même lié à :
- une base azotée.

Il existe quatre bases azotées différentes : l'adénine (notée A), la thymine (notée T), la cytosine (notée C) et la guanine (notée G).

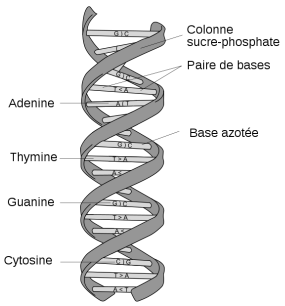


FIGURE 3.1.: Schéma de la double hélice d'ADN

L'ADN est composé de deux brins se faisant face, et formant une double hélice. Ceci est possible car les nucléotides présents dans un brin possèdent des nucléotides

complémentaires avec lesquels ils peuvent interagir. La figure 3.1[1] représente l'ADN avec les nucléotides et les deux brins se faisant face.

L'ARN messager (ARNm)

L'acide ribonucléique messager, que l'on notera dorénavant ARNm, est une copie de l'ADN utilisée comme intermédiaire par les cellules dans la synthèse des protéines. L'ARNm emploie la base U (Uracile), contre la base T dans l'ADN. La figure 3.2[2] schématise les étapes amenant à la création d'une protéine à partir d'un morceau de brin d'ADN (gène).

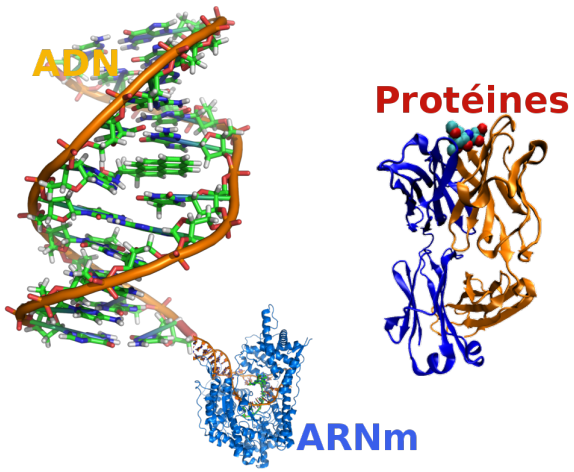


FIGURE 3.2.: Schématisation de la fabrication d'une protéine à partir de l'ADN

Le processus de fabrication d'une protéine est composé de deux étapes : la transcription de la séquence d'ADN du gène en un ARNm, puis, la traduction de l'ARNm en protéine. Le transcriptome représente tous les transcrits, c'est à dire les ARNm, exprimés à un moment donné par le génome dans une cellule, un tissu et un individu donnés. Le protéome est l'ensemble des protéines générées dans ces mêmes conditions à partir des ARNm. Dans notre étude, nous resterons au stade du transcriptome et par conséquent à l'étude de la quantité d'ARNm produite. L'expression d'un gène est mesurée par la mesure quantitative de ses transcrits.

Biopuces

L'expression des gènes est la mesure de la quantité d'ARNm produit par un gène donné présent dans un tissu sélectionné à un moment donné. L'expression des gènes se mesure par biopuces (aussi appellées « microarray » en anglais, ce sont des membranes en nylon de 2.5 x 7.5 cm). Une photo d'un fragment de biopuce peut être visualisé dans la figure 3.3[3].

Les diverses étapes de la fabrication d'une biopuce sont schématisées dans la figure 3.4[4].

[1] L'image provient de Wikimedia Commons et est attribuable à MesserWoland avec des modifications de Dosto.
[2] L'image est un montage réalisé à partir d'images provenant de Wikimedia Commons et attribuables à Zephyris et Thomas Splettstoesser.
[3] L'image provient de Wikimedia Commons et est attribuable à Mangapoco.
[4] L'image est attribuable à SarahKusala.

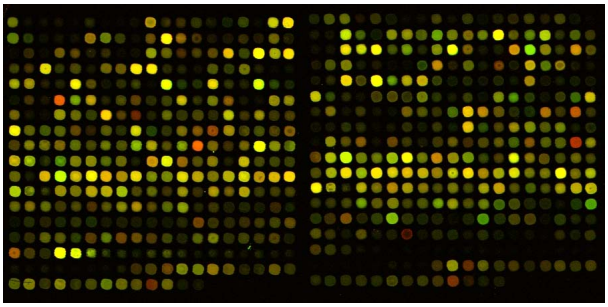


FIGURE 3.3.: Photo d'un fragment de biopuce

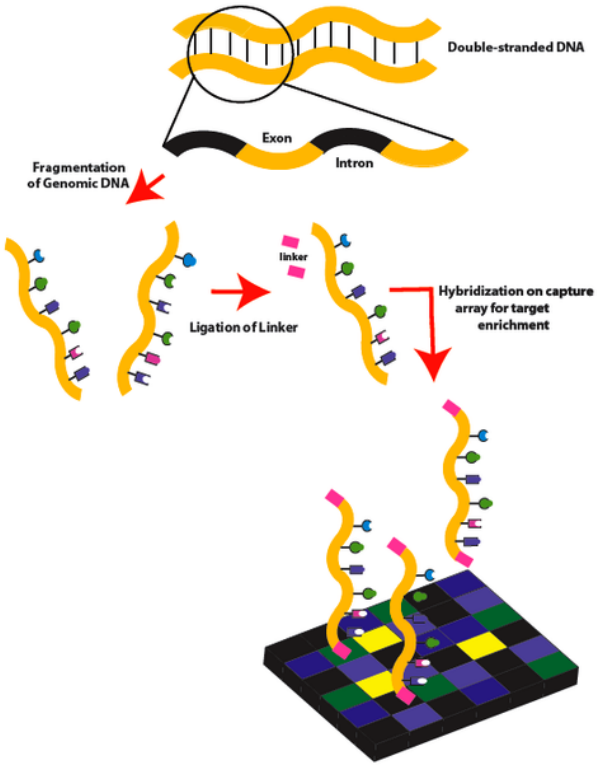


FIGURE 3.4.: Schéma des étapes de la fabrication d'une biopuce

1. On crée une plaque avec de l'ADN connu, avec un gène sur chaque « spot » (on a plusieurs milliers de spots par biopuce) ;
2. On crée une sonde complexe (les ADN complémentaires créés à partir de l'ARNm de la cellule à étudier, tagués par fluorescence ou radioactivement) ;
3. On met en contact la sonde et la plaque. Si les brins d'ADN sont identiques, ils s'associeront, et le spot sera plus ou moins fluorescent/radioactif selon la quantité de brins présents dans la sonde complexe).

On récupère donc comme information, pour chaque gène, la mesure de son expression (à travers la quantité de fluorescence/radioactivité émise par chaque « spot »). Des tailles typiques pour ce genre de données est une dizaine voire une centaine d'individus, et plusieurs milliers voire dizaine de milliers de gènes dont l'expression est mesurée pour chaque individu.

3.1.2. Données en relation avec la qualité de la viande de porc

Les deux premiers jeux de données sur lesquels j'ai travaillé provenait d'un projet financé par l'ANR. La problématique de ce projet est l'analyse fine de la diversité génétique des différentes races porcines françaises. Dans ce cadre, deux jeux de données ont été étudiés contenant l'expression de gènes sélectionnés pour des animaux de races « Large White (LW) » (condition 1) et « Landrace » (condition 2) :

1. le premier jeu de données contenait, pour 84 cochons (les individus statistiques), de deux races différentes (LW, Landrace), la mesure de 464 gènes qui avaient au prélalable été sélectionnés (parmi 12 358) pour leur expression différentielle pour le facteur race ;

2. le second jeu de données contenait, pour 84 cochons (les individus statistiques), de deux races différentes (LW, Landrace), la mesure d'un groupe de 1 058 gènes pour lesquels la corrélation au sein de chaque race était la plus forte.

3.1.3. Données en relation avec l'obésité

Les autres jeux de données provenait du projet européen Diogènes de lutte contre l'obésité [5]. Au sein de ce projet, on a relevé l'expression d'un grand nombre de gènes chez des personnes obèses dans 3 conditions expérimentales séquentielles : avant un régime sévère, après ce régime, et 6 mois après sans contrainte stricte sur l'alimentation.

1. Dans le premier, les données des mêmes 271 gènes sur les mêmes individus (214 femmes obèses) sont observées avant et après le régime (2 conditions avec des individus appariés).

2. Dans le deuxième, 23 variables génétiques et 16 variables cliniques ont été choisies par Nathalie Viguerie, chargée de recherches à l'INSERM. Ces mesures ont été établies sur 202 femmes (obèses) avant et après le régime. Les 204 individus considérés ne sont pas un sous-ensemble des 214 obèses du premier jeu de données mais sont toutes les femmes prises dans un ensemble plus large d'individus).

3. Le troisième jeu de données ne diffère du précédent que par le choix des variables : on a choisi 38 expressions de gènes et 12 conditions physiologiques à étudier.

4. Dans le quatrième, les données ont suivi un pré-traitement : en prenant le logarithme du quotient des données mesurées avant régime et après régime, puis le logarithme quotient des données mesurées avant régime et 6 mois après, on a construit un jeu de données ayant deux conditions avec des individus appariés.

5. Le cinquième jeu de données a les mêmes variables que le troisième (38 expressions de gènes et 12 conditions physiologiques). Les conditions sont définies par le sexe des individus statistiques (356 femmes dans la condition 1, 189 hommes dans la condition 2), les individus ne sont donc pas appariés.

Dans la suite, des abbréviations seront utilisées pour désigner les divers jeux de données. Un tableau récapitulatif est présenté dans le tableau 3.1 dans un objectif de clarté, ainsi que les appelations des jeux de données qui seront utilisées par la suite.

[5] http://www.diogenes-eu.org

appellation	extrait de	condition expérimentale	appariés ?	nombre d' individus	nombre de gènes
ll1	DeLiSus	race	non	51 LW et 33 LR	464
ll1b	DeLiSus	race	non	51 LW et 33 LR	1058
ll2	DeLiSus	race	non	51 LW et 33 LR	123
nv1	Diogènes	avant/après régime	oui	214	271
nv1b	Diogènes	avant/après régime	oui	214	266
nv2	Diogènes	avant/après régime	oui	204	39
nv3	Diogènes	avant/après régime	oui	204	50
nv4	Diogènes	quotients	oui	204	39
nv5	Diogènes	sexe	non	356 F et 189 H	50

TABLE 3.1.: Légende des abbréviations de jeu de données

3.2. Problématique du stage

Dans un premier temps, la problématique est de modéliser les relations entre les gènes, puis d'observer l'impact de conditions (physiologiques, comme la race, ou environnementales, comme le fait de subir un régime) sur ces relations. Dans la suite, la modélisation sera faite en inférant des *réseaux de co-expression génique* à partir des données.

3.2.1. Graphes

Qu'est ce qu'un graphe ?

Définition 1 (Graphe)**.** *On appelle* graphe *(ou réseau)* $G = (V, E)$ *(non pondéré) ou parfois* $G = (V, E, W)$ *(pondéré) un ensemble d'entités,* V*, appelées* nœuds *(ou* vertex *en anglais) qui peuvent (ou pas) être reliées, deux à deux, par une relation donnée (appelée* arêtes *ou* edge *en anglais). L'ensemble des paires de sommets liés par une relation est notée* $E \subset V \times V$*.*
Dans le cas pondéré, les relations sont chacune munie de poids (notés W*) qui sont des réels positifs.*
Les arêtes peuvent être orientées ou non et la matrice des poids, W *est alors symétrique ou non.*

Remarque 1. *Un graphe* $G = (V, E)$ *non orienté, avec* p *nœuds est équivalent à une matrice triangulaire* A *dite « d'adjacence » de dimension* $p \times p$*, définissant pour tout* j*,* j' $(j \neq j')$ *la présence éventuelle d'une arête entre le nœud* j *et le nœud* j' *de la manière suivante :*

- *si* $\alpha_{j,j'} = 1$*, alors si il existe une arête entre le nœud* j *et le nœud* j' *;*
- *si* $\alpha_{j,j'} = 0$*, alors il n'existe pas d'arête entre le nœud* j *et le nœud* j'*.*

Remarque 2. *Comme l'on n'autorise pas les boucles,* $\forall j, \alpha_{j,j} = 0$

La figure 3.5 illustre l'équivalence entre un graphe à 4 sommets et 4 arêtes et sa matrice d'adjacence. On remarque que l'épaisseur ou la longueur des arêtes n'a aucune importance ici : seule l'existence ou la non-existence de relations entre sommets sont modélisées par le graphe.

Gènes	1	2	3	4
1	x	x	x	x
2	0	x	x	x
3	1	0	x	x
4	1	1	1	x

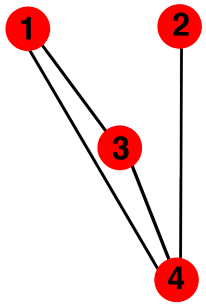


FIGURE 3.5.: Exemple d'équivalence entre une matrice d'adjacence et un graphe

Un certain nombre de notions classiques en théorie des graphes, qui seront utilisées dans la suite du travail, sont exposées ici :

Définition 2. *La densité d'un graphe est le quotient du nombre d'arêtes qu'il contient par le nombre d'arêtes maximal possible d'un graphe ayant le même nombre de nœuds :*

$$D = \frac{\textit{nombre d'arêtes}}{\textit{nombre de paires de nœuds}}$$

Définition 3. *La* transitivité *d'un graphe est la probabilité que deux nœuds adjacents à un même nœud soient eux aussi connectés. (Intuitivement, dans un contexte social, la probabilité parmis tous nos amis que deux d'entre eux aient des amis en commun).*

Définition 4. *Le* degré *d'un nœud est tout simplement le nombre d'arêtes émanant de ce nœud.*

Définition 5. *La* betweenness *(appelée aussi parfois "intermédiarité") d'un nœud v est une mesure de sa centralité. Elle se calcule de la manière suivante :*

1. *Pour chaque paire de nœuds, $\{t, u\}$, calculer le nombre de plus courts chemins σ_{tu} qui les relient.*
2. *Ensuite, observer quelle fraction de ces plus courts chemins passe par le nœud en question, $\frac{\sigma_{tu}(v)}{\sigma_{tu}}$ où $\sigma_{tu}(v)$ est le nombre de plus courts chemins entre t et u qui passent par v.*
3. *Enfin, faire la somme de cette quantité pour toutes les paires de nœuds $\{t, u\}$ du graphe.*

La formule calculatoire est donnée ci dessous :

$$C(v) = \sum_{t \neq u \neq v \subset V} \frac{\sigma_{tu}(v)}{\sigma_{tu}}$$

Il a été dit précédemment que la longueur des arêtes, donc l'agencement des nœuds dans la visualisation d'un graphe, n'a pas de signification mathématique dans ce contexte. Pour représenter les réseaux visuellement, il existe plusieurs algorithmes qui permettent de calculer un agencement facilitant la lecture du réseau, par exemple en rapprochant les nœuds les plus connectés entre eux et en éloignant les moins connectés. C'est le cas de l'algorithme décrit dans [Fruchterman and Reingold, 1991], qui a été majoritairement utilisé pour générer des agencements le long de ce stage.

Application au contexte biologique

Dans le contexte biologique, on recherche des liaisons (de co-expression) entre un ensemble de gènes : dans ce problème, les gènes sont donc les nœuds du graphe. Cette liaison sera mesurée à l'aide d'une corrélation, dont on précisera la nature dans la partie 3.2.2.

3.2.2. Inférence de réseaux de co-expression génique

À partir de mesure sur plusieurs individus de l'expression de chaque gène, on peut calculer les corrélations deux à deux entre l'expression des gènes. Ces corrélations sont la base pour définir des *réseaux de co-expression génique*. Dans de tels réseaux, on représente chaque gène par un nœud (*vertex*), et chaque corrélation directe et significative par une arête (*edge*) entre les deux nœuds concernés. On entend par **significative** le fait que cette corrélation dépasse un certain seuil, éventuellement déterminé par un test statistique, et par **directe** le fait qu'elle s'exprime indépendamment des autres gènes. On effectue donc un calcul de la corrélation partielle entre les deux gènes pour obtenir la valeur numérique permettant de déterminer si l'arrête doit être ajoutée au réseau ou non.

De manière formelle, notons $X^1, X^2, \ldots, X^p$ l'expression de p gènes, mesurée sur n unités statistiques (typiquement, sur des données de biopuces, p est de l'ordre de 10^4, et n est de l'ordre de 10^1). Toute la suite de ce travail se place dans le cadre du *Modèle Graphique Gaussien* (noté GGM) : on considère donc, par la suite, que le vecteur aléatoire des expressions de gènes est un vecteur gaussien : $X = [X^1, \ldots, X^p] \sim \mathcal{N}(0, \Sigma)$.

On s'intéresse alors à la matrice des corrélations *partielles* (notée Π) entre les expressions de gène, qui est de taille $p \times p$. Ses coefficients, $\forall j \neq j' (j < p,\ j' < p)$, sont définis de la manière suivante :

$$\pi_{jj'} = \mathrm{Cor}(X^j, X^{j'} | X^k, k \neq j, j').$$

Deux gènes seront liés par une arête dans le graphe si et seulement si leur corrélation partielle est significativement non nulle. On peut démontrer (voir [Edwards, 1995]) que ces corrélations partielles sont liées à la « matrice de concentration » $S = \Sigma^{-1}$ par la relation :

$$\pi_{jj'} = -\frac{S_{jj'}}{\sqrt{S_{jj} S_{j'j'}}}.$$

Il faut donc inverser la matrice des variances-covariances, de taille $p \times p$.

L'inversion d'une matrice dans le cas des hautes dimensions ($p \gg n$) s'avère être un problème mathématiquement mal posé : l'estimateur de Σ calculé à partir des données, $\hat{\Sigma}$, est instable par inversion donc son inverse est un mauvais estimateur de S. Il existe plusieurs méthodes pour estimer la matrice S, qui seront présentées ci-dessous.

En particulier, à cause de l'hypothèse Gaussienne, on peut associer à ce contexte un modèle linéaire :

$$X^j = \sum_{k \neq j} \beta_{jk} X^k + \epsilon \tag{3.1}$$

dans lequel l'expression du gène j s'écrit en fonction de l'expression de tous les autres gènes $X^{j'}$, $(\forall j \neq j')$. On estime alors les coefficients β_{jk} de ce modèle par maximum de vraisemblance et on peut démontrer que

$$\beta_{j,j'} = -\frac{S_{jj'}}{S_{jj}}.$$

On peut donc en inférant B retrouver la matrice $S = \Sigma^{-1}$ donc les corrélations partielles Π.

Remarque 3. *Pour élaborer la matrice d'adjacence (et donc le graphe) à partir de la matrice* $\Pi = (\pi_{jj'})_{j=1,\dots,p;j\neq j'}$*, une étape préalabe s'impose : le choix des arêtes à conserver. Comme l'on ne veut garder que les corrélations* significatives*, il faudra effectuer un seuillage sur la matrice* Π*, et ne conserver que les valeurs supérieures ou égales au seuil (qui n'est pas déterminé numériquement dans toutes les méthodes) dans la construction de la matrice d'adjacence* A*. Le processus d'élaboration du graphe à partir des corrélations partielles est illustré dans la figure 3.6* [6]*.*

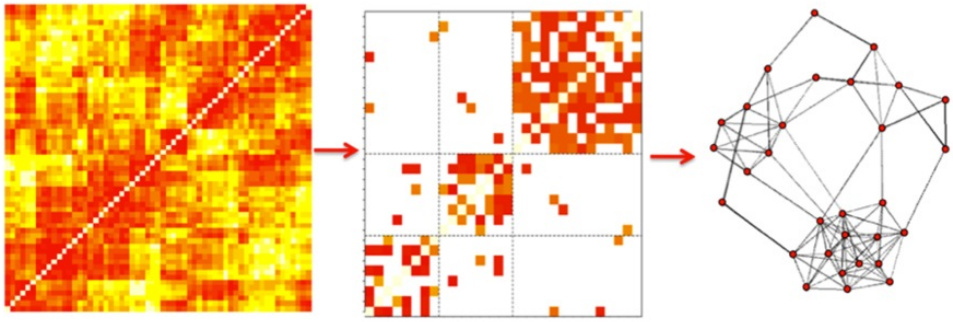


FIGURE 3.6.: Visualisation du passage de la matrice des corrélations partielles (à gauche), à la matrice d'adjacence (après seuillage, au milieu), au graphe (à droite).

Le problème d'inférence de structure peut alors se réduire à une estimation des coefficients $\beta_{j,j'}$ à partir des données. Par la suite, différentes méthodes réelles d'inférence de graphes sont proposées ci dessous, et mises en application dans les packages correspondants, décrits dans la partie 4.

Méthode de [Schäfer and Strimmer, 2005]

Dans un premier temps, cette méthode vise à rendre la « matrice de concentration » plus stable par inversion, en l'estimant de la manière suivante : $\hat{S} = (\hat{\Sigma} + \lambda I)^{-1}$. Ensuite, un test bayésien avec une correction de tests multiples est proposé pour déterminer les arêtes significatives.

Méthodes par Maximum de Vraisemblance

Il existe deux méthodes d'estimation de Σ^{-1} à l'aide du maximum de vraisemblance : celle de [Friedman et al., 2008] et celle de [Meinshausen and Bühlmann, 2006].

[6] Cette image a été extraite d'une présentation effectuée aux Journées de la Société Française de Statistique (JdS 2012, Bruxelles, Belgique, 21-25 mai) ; elle a été gracieusement prêtée par http://www.agrocampus-ouest.fr/math/causeur, Professeur à Agrocampus Ouest.

Dans l'approche de [Meinshausen and Bühlmann, 2006], le critère à maximiser pour chaque gène j est le suivant :

$$\max_{(\beta_{j,j'})_{j'\neq j}} \left(\log \mathrm{ML}_j - \lambda \sum_{j'\neq j} |\beta_{j,j'}| \right)$$

avec $\log \mathrm{ML}_j \sim -\sum_{i=1}^{n} \left(X_i^j - \sum_{j'\neq j} \beta_{j,j'} X_i^{j'} \right)^2$.

Le premier membre du critère est le logarithme du maximum de vraisemblance pour le modèle linéaire (3.1), et le second est un membre de régularisation qui dépend de la norme L^1 de tous les $\beta_{j,j'}$: $|\beta_j|_{L^1} = \sum_{j'\neq j} |\beta_{j,j'}|$. Par le biais de ce type de pénalisation, une grande partie des coefficients $\beta_{j,j'}$ sont amenés à s'annuler : c'est le principe de la pénalisation « sparse », L^1 ou LASSO. Il s'agit alors de faire varier le paramètre de régularisation (noté ici λ), pour obtenir plus ou moins de coefficients non nuls dans la matrice d'adjacence, donc plus ou moins d'arêtes dans le graphe.

La matrice $B = (\beta_{j,j'})_{j,j'=1,\dots,p}$ en résultant est asymétrique : la valeur à la $j^{\text{ème}}$ ligne et $j'^{\text{ème}}$ colonne ($j \neq j'$) de la matrice peut être différente de celle à la $j'^{\text{ème}}$ ligne et la $j^{\text{ème}}$ colonne. On se retrouve dans une situation où potentiellement, dans la case (j, j') la corrélation est non nulle mais dans la cellule (j', j), elle ne l'est pas, et inversement. Il faut alors prendre une décision, et appliquer un des deux opérateurs logiques « ET » ou « OU » aux résultats obtenus pour déterminer si il faut mettre une arête entre le $j^{\text{ème}}$ et le $j'^{\text{ème}}$ sommet, ($j \neq j'$). On parlera dans la suite respectivement des approches MB-AND et MB-OR.

Dans l'approche de [Friedman et al., 2008], on considère le critère (3.2), que l'on cherchera par la suite à minimiser en fonction des coefficients $(\beta_{j,j'})$, pour le modèle entier :

$$\min_{(\beta_{j,j'})_{j,j',\ j\neq j'}} \left(\sum_j \log \mathrm{ML}_j + \lambda \sum_{j\neq j'} |\beta_{j,j'}| \right) \tag{3.2}$$

Si la matrice d'adjacence est obtenue par la méthode de [Friedman et al., 2008], elle est symétrique : la corrélation du $j^{\text{ème}}$ gène avec le $j'^{\text{ème}}$ ($j \neq j'$) est la même que celle entre le $j'^{\text{ème}}$ et le $j^{\text{ème}}$. Il n'y a pas d'ambiguité sur les corrélations, on peut directement construire la matrice d'adjacence.

3.2.3. Méthodes d'inférence jointes

Les méthodes jointes partent du principe qu'il existe dans les réseaux issus des deux conditions expérimentales, une part commune et une part différentiée. On cherchera alors par la suite à combiner une partie des informations issues des deux conditions dans l'inférence. Cette théorie peut être mise en application à l'aide du package **simone**.

Des approches jointes alternatives ont également été développées, basées sur les packages **GeneNet** et **simone**. Les détails des diverses approches sont présentés ci dessous :

simone

Ce package propose 3 méthodes d'inférence jointe décrites dans l'article [Chiquet et al., 2011] :

Intertwined L'idée derrière cette sous-méthode est de « rapprocher » les conditions à la condition moyenne dans la matrice des variances-covariances. Puisque la formule du maximum de vraisemblance peut s'écrire uniquement en fonction de la variance empirique $\hat{\Sigma}$, on la remplacera alors par (3.3)

$$\hat{\Sigma}^{cond\ j} = (\alpha)\Sigma^{cond\ j} + (1-\alpha)\bar{\Sigma} \quad (\bar{\Sigma} = \sum_{k=1}^{C} \Sigma^{condition\ k}) \tag{3.3}$$

avec :

- C le nombre de conditions
- α le paramètre de la combinaison convexe (entre 0 et 1)

Grouplasso Dans ce cas, on fait l'hypothèse la part differentiée de la condition expérimentale est nulle (ou que l'on est dans le cas de répétitions) ; dans les calculs, la pénalité *sparse* tient compte (pour chaque gène) de la valeur de $\beta_{j,j'}$ au sein de toutes les conditions. Par voie de conséquence, la probabilité que $\beta_{j,j'}$ soit nul dans une condition et non nul dans l'autre vaut 0, donc cette méthode revient à contraindre les arêtes à être identiques à travers toutes les conditions.

Cooplasso On part dans cette méthode de l'hypothèse biologique que une corrélation partielle ne peut pas être à la fois significative dans les deux conditions, positive dans l'une et négative dans l'autre. Pour la mettre en place, le package utilise les mêmes bases que Grouplasso, en ajoutant la condition suivante : elle pénalise fortement le cas où une corrélation partielle est significativement positive dans une condition et négative dans l'autre et aura tendance, dans ce cas, à ne conserver que celle qui est la plus éloignée de 0 (la plus grande en valeur absolue).

Méthode jointe appariée avec GeneNet

Cette méthode est faite pour combiner des informations provenant d'échantillons *appariés*, c'est à dire plusieurs mesures de l'expression des mêmes gênes sur les mêmes individus (c'est le cas pour les données d'obésité décrites en 3.1.3). Ici, on pose l'hypothèse que les données ont une part commune et une part différentiée. Par conséquence on remplacera la valeur de l'expression du gène dans chacune des conditions par (3.4)

$$\hat{X}^{cond\ j} = 1/2 X^{cond\ j} + 1/2 \bar{X} \quad \left(\bar{X} = \sum_{k=1}^{2} X^{cond\ k}\right) \tag{3.4}$$

On retrouve donc dans ce cas une influence de 3/4 de la condition étudiée et 1/4 de l'autre condition sur les données étudiées.

therese

L'idée derrière cette modification de simone est la suivante : combiner de manière convexe les informations de simone au niveau des corrélations partielles. Pour ce faire, on va combiner les corrélations partielles issues de la matrice Π de la condition avec celle d'une condition « moyenne », comme illustré dans l'équation (3.5)

$$\hat{\Pi}^{cond\ j} = (\alpha)\Pi^{cond\ j} + (1-\alpha)\bar{\Pi} \qquad \left(\bar{\Pi} = \sum_{k=1}^{C} \Pi^{cond\ k}\right) \tag{3.5}$$

avec :

- C le nombre de conditions
- α le paramètre de la combinaison convexe (entre 0 et 1)

(Cette idée a été proposée par Magali SanCristobal, ma tutrice de stage.)

4. Outils et méthodes

4.1. Git

Dans l'objectif de fournir un suivi de meilleure qualité des évolutions de mes travaux, j'ai été amené à utiliser un logiciel de *contrôle de versions* nommé Git[1]. Il a été développé par Linus Torvalds en 2005 pour permettre le développement collaboratif du noyau Linux.

Utilisé dans un répertoire, il permet une gestion fine des versions de fichiers, en sauvegardant toutes leurs modifications, et offre la possibilité de synchroniser le répertoire entre plusieurs ordinateurs via un serveur. Cela a eu pour effet de simplifier largement la correction et le suivi des scripts R produits lors de mon stage.

Voici un processus de travail git typique :

1. Un utilisateur modifie un fichier existant, qui se situe dans le répertoire git.
2. Il écrit ses modifications au git, en tapant `git commit [nom du fichier]` dans son invite de commande.
3. Il envoie ses modifications au serveur git, avec `git push`.
4. Tous les autres utilisateurs (qui ont droit d'accès au serveur git) peuvent récupérer ces modifications à l'aide d'un `git pull`.

Le logiciel Git propose également une interface graphique, nommée gitk, pour visualiser simplement les versions successives des différents fichiers, présentée en figure 4.1.

Chaque ligne dans les deux sous-fenêtres en haut de la capture d'écran est une date de `commit`, c'est à dire le moment ou l'auteur a voulu enregistrer ses modifications sur le git (il est de bon usage d'ajouter une description succinte des ajouts et/ou modifications lors d'un `commit`, ici en haut à gauche). Lorsque l'on sélectionne un `commit`, le ou les fichiers modifiés dans celui-ci apparait en bas à droite de la fenêtre ; une fois qu'un fichier est sélectionné, la sous-fenêtre en bas à gauche affiche les ajouts au fichier en vert et les suppressions en rouge.

De part ses origines collaboratives, Git gère nativement les conflits d'édition ; dans le cas où un même fichier est modifié en même temps par deux utilisateurs, si il est modifié à des endroits différents, le fichier est automatiquement fusionné : les modifications des deux utilisateurs sont incorporées au fichier. Si le fichier est modifié au même endroit, le deuxième utilisateur qui envoie son `commit` au serveur avec un `push` se voit son `push` refusé par le serveur. Il doit faire un `pull` pour récupérer les modifications du premier, choisir quelle version garder, puis faire un `push` pour propager la modification retenue au serveur.

[1] Site web officiel : http://git-scm.com

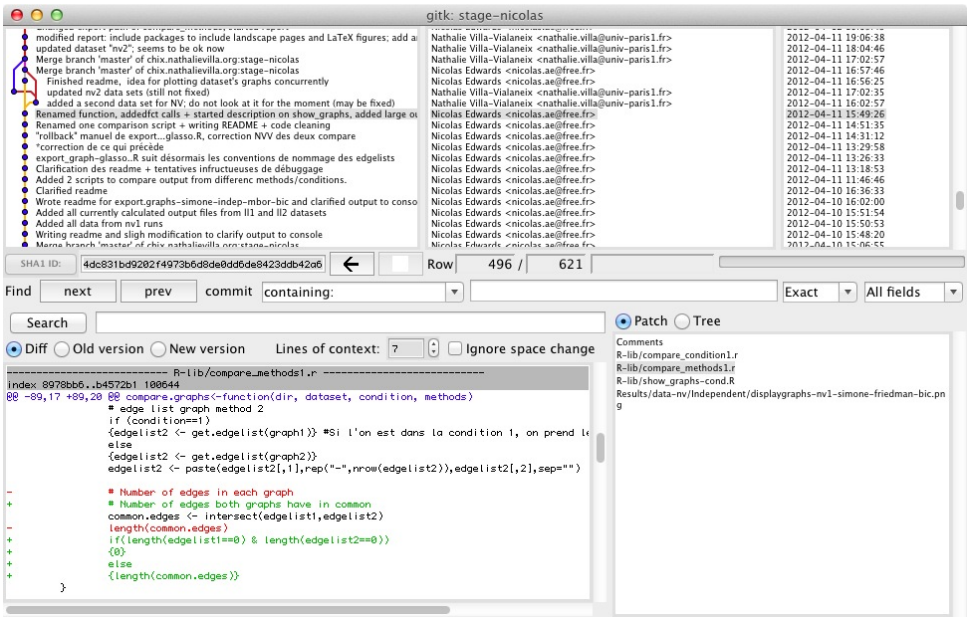


FIGURE 4.1.: Capture d'écran de gitk

4.2. Packages R et outils logiciels utilisés pour les graphes

L'outil de travail principalement utilisé était R [R Development Core Team, 2012], qui est un logiciel libre sous licence GNU GPL. Le mardi 3 Avril, j'ai suivi au sein de l'INRA une formation de 7 heures sur les graphes, qui se décomposait en 3 heures de cours et 4 heures de TP sur logiciel. Les cours m'ont permis d'assimiler les notions de base de la théorie des réseaux et leur étude dans le cadre génétique. Puis durant le TP, les packages suivants ont été étudiés :

- **GeneNet** : ce package implémente l'approche décrite dans l'article de [Schäfer and Strimmer, 2005] ;
- **glasso** : ce package propose deux méthodes d'inférence de graphes : celle de [Friedman et al., 2008] et celle de [Meinshausen and Bühlmann, 2006] ;
- **simone** : ce package propose de nombreuses méthodes d'inférence, dont celles de [Friedman et al., 2008] et celle de [Meinshausen and Bühlmann, 2006] (pour des approches GGM « standard », utilisées dans le cadre dit indépendant). Il intègre aussi des approches jointes « Interwined », « GroupLasso » et « CoopLasso » décrites plus haut : celles-ci ont été introduites dans l'article de [Chiquet et al., 2011]. Ce dernier package n'a pas été abordé lors de la formation, mais il a été très utilisé dans les scripts qui ont été développés par la suite.
- **igraph** : ce package sert à créer et manipuler des objets « graphe » dans R.

Durant le TP et le stage, j'ai été amené à utiliser le logiciel Gephi [Bastian et al., 2011]. Un programme Java qui implémente l'algorithme de Früchterman-Reingold pour l'agencement des nœuds sur une surface plane a été également utilisé.

Durant le TP et le stage, j'ai aussi été amené à utiliser le logiciel libre de visualisation de graphes, Gephi[2] [Bastian et al., 2011] et notamment, j'ai aidé les biologistes à utiliser ce logiciel pour explorer les graphes qui avaient été générés par les scripts décrits ci-dessous.

Enfin, j'ai utilisé, en ligne de commandes, un programme Java[3] qui implémente l'algorithme de Früchterman et Reingold [Fruchterman and Reingold, 1991] pour calculer un agencement lisible des nœuds d'un graphe donné sur une surface plane.

4.3. Scripts développés

Les scripts écrits lors de mon stage appartiennent à une des 3 catégories suivantes : *importation*, *inférence de graphe*, *analyse*. Pour faciliter l'utilisation des scripts complexes, j'ai créé une catégorie *exécution*, et pour accélérer certains calculs massifs, une catégorie *parallèle*, que j'évoquerai dans les sections correspondantes. Une vue synthétique du processus de travail avec ces outils est présenté en figure A.1 (annexes), et une description un peu plus détaillée des fichiers est disponible dans les annexes, section A.2

Ces scripts ont été construits de manière à être réutilisables : concrètement, tant que l'on obéit aux conditions de nommages décrites ci-dessous, on utilise le même script d'inférence quel que soit le jeu de données importé, et le même script d'analyse quel que soit le script d'inférence. Ce travail de généralisation m'a été possible grâce à deux éléments : d'une part, les fonctions dans le language R, qui permettent de passer des paramètres variables à un programme, appelé fonction. Au lieu de réecrire du code spécifique pour chaque jeu de données, j'ai écrit un code générique avec des parties variables (c'est la fonction) puis des appels de fonction, qui l'exécutent en déclinant la partie variable en autant de méthodes, de jeux de données, ... qu'il y a à traiter. L'autre élément essentiel à la généralisation était une réflexion préalable sur les conventions de nommage strictes que je décrirai en même temps que les catégories de fichiers. Dans la suite, je mettrais les noms de fichiers en `police monospace`, les parties variables de ces noms entre guillemets et en *'italique'*, et des parties optionnelles entre [crochets].

Dans un souci d'utilité future de l'ensemble des scripts, un mode d'emploi avec le descriptif de chaque fichier a été également rédigé, entièrement en anglais. Il est disponible en annexe A.3

Importation

Pour chaque jeu de données traité, il existe un fichier d'importation correspondant. Il s'occupe de lire le fichier de données et de « standardiser » son format, et peut éventuellement s'occuper de pré-traitements mineurs, tels l'imputation à une valeur minimale des valeurs manquantes. Ce sont aussi les seuls fichiers (à quelques exceptions non-testées près) qu'il faut écrire si l'on veut se servir des scripts pour de nouveaux jeux de données. Pour faciliter et, dans certains cas, automatiser l'importation, *tous* les noms des fichiers d'importation obéissent à la convention de nommage suivante : `import_'jeu de données'.r` (des noms de jeu de donnés utilisés sont : « ll1 », « ll2 », « nv1 », « nv2 », « nv3 », ...) Les fichiers d'importation créent deux objets (data.frame) R enregistrés en mémoire, appelés « cond1 » et « cond2 ».

[2] http://www.gephi.org

[3] Le script Java a été implémenté et gracieusement prêté par http://www.apiacoa.org, Professeur à l'Université Paris 1.

Inférence de graphe

Une fois que le jeu de données est importé, il faut inférer des graphes à partir de ceux-ci, ce qui est fait à l'aide de fonctions. Dans les méthodes indépendantes, il faut importer les données en exécutant le fichier d'importation correspondant ; dans les méthodes jointes, on fournit le nom du jeu de données directement et la fonction s'occupe d'un pré-traitement spécifique aux méthodes jointes. Pour faciliter le référencement des fichiers d'inférence, leurs noms obéissent à la convention de nommage suivante : `export_graph-'méthode'-'type d'inférence'-'critère de sélection'.r`

Certaines méthodes proposent un critère de sélection, d'autres pas. Des noms typiques de méthodes sont « simone », « genenet » ; les noms de type d'inférences sont « indep » et « joined » ; le nom du critère de sélection est « bic ».

Toutes les méthodes d'inférence écrivent des fichiers dans un répertoire fixe : « ../Results/Independent » ou « ../Results/Joined », selon le type d'inférence. Le détail des fichiers écrits est présenté ci-dessous :

- 1 fichier au format `.RData`, qui contient deux objets *igraph* (un par condition) nommés « graph1 » et « graph2 », respectivement. Leur nom se présente de la manière suivante :

  `graph-'dataset'-cond12-'méthode'-'options'.RData`
  où méthode et option désignent la méthode de création et les options affectées à ces méthodes. Ces objets contiennent la structure du graphe (c'est à dire l'ensemble des nœuds présents, arêtes (ou paires de nœuds connectés), et des informations complémentaires ajoutées par noeud ou par arête, comme par exemple la valeur de la corrélation partielle, par arête inférée)

- 2 fichiers au format `.graphml`, qui contiennent les graphes des deux conditions dans un format standard inter-logiciel (comme le .txt pour le texte ou le .csv pour les données numériques). Leur nom est structuré comme suit : `graph-'dataset'-'condition'-'méthode'- 'options'.graphml`, où *condition* désigne la condition expérimentale, et a pour valeur possible « cond1 » et « cond2 ».

- 2 fichiers au format `.txt`, contenant, pour chaque condition expérimentale, la liste de paires de nœuds qui forment les arêtes dans les graphes.

Analyse des graphes obtenus

La section analyse se décompose en deux volets : la représentation graphique, et l'analyse comparative numérique.

La représentation graphique se fait après avoir fixé un agencement des nœuds, à l'aide du script `create_layout.r`. A l'aide d'un type d'agencement, il écrit un fichier `.txt` décrivant la position sur l'abscisse et l'ordonnée de chaque nœud. Ensuite, à l'aide d'un répertoire, d'un nom de jeu de données et d'une méthode, le script `show_graphs.r` écrit deux images de taille différente au format `.png`, de nom « displaygraph-**dataset**-method.png »contenant la visualisation des deux réseaux définis par les deux conditions. Il existe de multiples options graphiques (nom, forme, couleur des noeuds ; taille, couleur des arêtes, présence de légende, ...)

L'analyse comparative numérique est de deux sortes : comparaison des méthodes entre elles à condition fixée, et comparaison des deux conditions pour chaque méthode.

- La première comparaison s'effectue par le script `compare_methods.r`, et écrit au format `.csv` une matrice triangulaire avec chaque méthode (passée en paramètre à la fonction) en ligne et en colonne, à jeu de donnée fixé ; sur la diagonale figure le nombre d'arêtes de la méthode, et dans la partie triangulaire se situe le nombre d'arêtes communes aux conditions.
- La deuxième s'effectue par le biais de `compare_conditions.r` ; pour un jeu de données et chaque méthode passée en paramètre à sa fonction, il va lister le nombre d'arêtes dans la première et seconde condition, puis le nombre d'arêtes à l'intersection des deux, avant d'écrire le résultat au format `.csv`.

Exécution

Comme il a été énoncé précédemment, l'aspect fonctionnel des scripts se décompose en deux volets : la fonction générique et les appels de fonction qui la rendent spécifique (à un jeu de donnée, une méthode, une option de méthode, ...) Dans un souci de clarté, les appels ont été placés dans des fichiers différents. En préambule de ces fichiers, on lit le script qui contient la fonction à proprement dit, de manière à ce qu'en exécutant tout le fichier « appel » d'un coup, on puisse faire itérer la fonction à travers tous ses appels.

Parallélisation

Sur certains packages, notamment **simone**, les temps de calcul de l'inférence explosent au vu du nombre de gènes ; dans chaque étape, on a environ p régressions à faire, avec $p-1$ paramètres à estimer ; et il y a environ p étapes successives pour obtenir un réseau dans un bon intervalle de densité. Le temps de calcul est donc sensiblement proportionnel à p^3. Pour une quarantaine de gènes, sur mon ordinateur personnel, l'inférence est rapide (de l'ordre de 5 secondes) mais pour les autres jeux de données, dont le plus grand comporte 1058 gènes, le temps de calcul est démultiplié, et celui ci aurait pu durer plusieurs jours pour le traitement de données simulées (où il fallait inférer 100 graphes pour chaque méthode). J'ai donc, à l'aide du package **doMC**, parallélisé les inférences de très haute dimension.

Simulations

Un travail en amont de comparaison des méthodes d'inférence de graphes décrites en sections 3.2.2 et 3.2.3 a été effectué à l'aide de données simulées. Pour ce faire, nous avons simulé 3 jeux de données, dans deux conditions différentes. Une topologie de réseau avec 5 groupes d'attachement préférentiel a été choisie, c'est à dire 5 groupes de nœuds densément connectés avec d'autres nœuds du même groupe, et rarement connectés avec des nœuds d'un autre groupe. Ce choix a été fait arbitraitement, pour imiter l'existence de plusieurs processus biologiques. Pour chaque jeu de données, deux réseaux, correspondant à deux conditions, ont été créés de la manière suivante : à partir d'un réseau « mère », des arêtes ont été permutées de manière aléatoire pour former deux réseaux « enfants ». Pour chaque jeu de données, 100 paires de graphes ont ainsi été générés de manière aléatoire, avec une même densité cible fixée (entre 3% et 5% selon les jeux de données) et un nombre

de sommets (représentant les gènes) qui correspondait approximativement à certains jeux de données réel décrit dans les sections 3.1.2 et 3.1.3.

On a ensuite inféré des réseaux (appelés « simulés »), et mesuré des caractéristiques de la différence entre les réseaux « vrais » et les réseaux « simulés » (nombre de faux positifs, de vrais positifs, de faux négatifs et de vrais négatifs). À partir de ces chiffres là, des outils d'analyse comparative (du vrai graphe contre le graphe inféré) ont été construits, en utilisant tout d'abord un classement des arêtes en quatre catégories détaillées dans la figure 4.2, puis en déduisant du nombre d'arêtes dans chaque catégorie des statistiques spécialisées détaillées ci-dessous, telles la précision et le recall, dans le but de mieux connaître la fiabilité des différentes méthodes d'inférence.

Définition 6 (Précision)**.** *On appelle* précision *le nombre d'arêtes prédites présentes dans le graphe « vrai » (« vrais positifs », tp) divisé par le nombre total d'arêtes prédites (« positifs », soit « vrais positifs » et « faux positifs », $tp + fp$) :* $P = \frac{tp}{tp+fp}$.

Définition 7 (Recall (ou Rappel))**.** *On appelle* recall *(ou parfois* rappel *pour ne pas utiliser le terme anglais) le nombre d'arêtes prédites présentes dans le graphe « vrai » (« vrais positifs », tp) divisé par le nombre d'arêtes dans le graphe « vrai » (« vrais positifs » et « faux négatifs », $tp + fn$) :* $R = \frac{tp}{tp+fn}$.

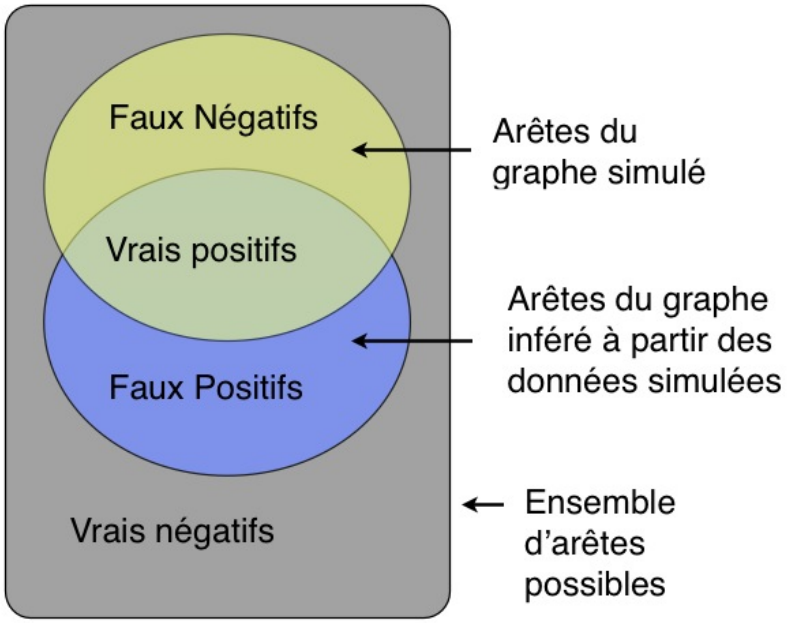


FIGURE 4.2.: Segmentation des arêtes du graphe « vrai » et du graphe inféré en 4 catégories permettant de calculer les statistiques de performance.

5. Résultats

Une partie de ces résultats seront publiés dans des articles : le travail sur les comparaisons de méthodes sera exposé lors des *Rencontres R*, qui se déroulent les 2 et 3 juillet 2012 à Bordeaux[1], [Villa-Vialaneix et al., 2012]. Par ailleurs, une partie du travail effectué (sur les jeux de données « nv4 » et « nv5 ») est intégrée à un article actuellement en révision pour publication dans la revue scientifique *PLoS Genetics*[2], [Viguerie et al., 2012].

5.1. Données simulées

La figure 5.1 est l'illustration d'un résultat de simulation : pour chacun des 100 jeux de données simulées ressemblant aux données « nv2 », les points correspondant aux valeurs de précision et de recall de chacun des couples de 100 graphes ont été représentés pour cinq méthodes de simulations différentes permettant ainsi la comparison des performances de ces cinq méthodes pour ces données.

Sur cet exemple, les résultats des simulations indiquent des tendances claires : tout d'abord, la précision ne dépasse pas la valeur de 70%, et pour une majorité écrasante des simulations se situe en dessous de 60%, ce qui indique que à peine plus de la moitié des arêtes inférées sont aussi dans le graphe initial. Ensuite, le recall atteint 100% dans une part non négligeable des simulations, mais ce qui est gagné en recall est en partie perdu en précision : dans les cas où toutes les arêtes initiales sont prédites, on en prédira en moyenne deux fois plus, autrement dit, la moitié de nos résultats seront des faux positifs. Au niveau de la comparaison entre méthodes, plusieurs points sont remarquables :

- la méthode de [Schäfer and Strimmer, 2005], même si pouvant obtenir des bons résultats de recall (80%), n'est pas très performante au vu de sa variabilité (pour la précision et le recall) ;
- pour l'approche therese, si l'on augmente le nombre d'arêtes à inférer (en comparant therese-friedman-bic-d5-a0dot5 et therese-friedman-bic-d10-a0dot5, établis à une densité fixée de 5 et 10% respectivement), le recall croît de 60% jusqu'à 100%, tandis que la précision diminue de manière moins marquée de 45% à 35% ;
- pour la méthode de [Chiquet et al., 2011] (intertwined), on obtient une précision supérieure à celle de l'approche therese, mais la variabilité du recall est aussi très importante, tout comme pour la méthode de [Friedman et al., 2008].

Plusieurs critiques de ces résultats peuvent être formulées : tout d'abord, le choix arbitraire de 5 groupes d'attachement préférentiel, et les autres choix de paramètres initiaux pourrait mal modéliser la réalité biologique. Ensuite, le processus de simulation de

[1] http://r2012.bordeaux.inria.fr

[2] PLoS (Public Library of Science) Genetics est une revue scientifique à communauté de lecture qui reflète la nature inter-disciplinaire de la recherche génétique et génomique en publiant des contributions dans tous les domaines de la biologie : http://www.plosgenetics.org.

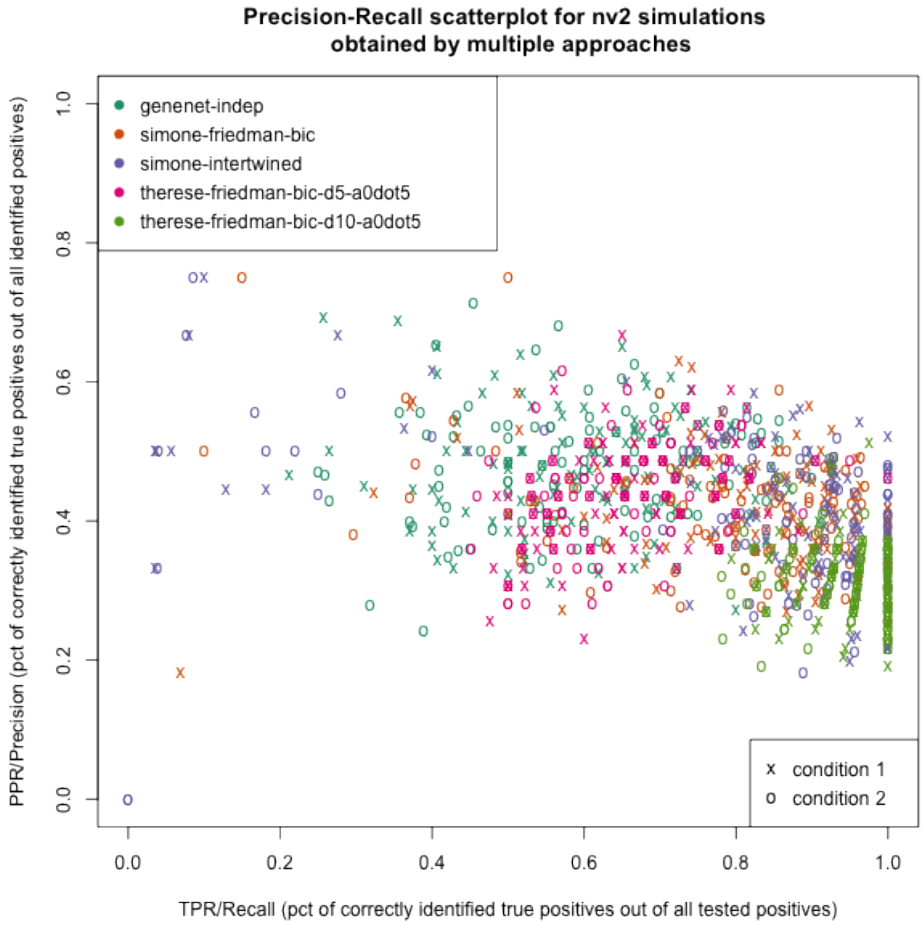


FIGURE 5.1.: Graphique de la précision en fonction du recall pour les jeux de données simulés à partir d'un « vrai » réseau, pour cinq méthodes d'inférence

données transcriptomiques est effectué à l'aide de **simone**, donc les méthodes d'inférence implémentées dans ce package pourraient être privilégiées lors de l'inférence.

Les résultats des simulations pour les deux autres jeux de données ont surtout mis en valeur une inadéquation des méthodes à la très haute dimension : en effet, dans ces données simulées, le nombre de variables (« gènes ») était très inférieur au nombre d'observations (nombre de données d'expression pour chaque gène). Dans ces cas, pour toutes les méthodes d'inférence, le recall était quasiment systématiquement nul ou très faible, quelle que soit la précision.

5.2. Données réelles

5.2.1. Résumé des inférences

Un total de 73 graphes a été inféré sur les données réelles. Leur répartition est indiquée dans le tableau de la page 34. Les différences entres les différents réseaux inférés pour un même jeu de données peuvent être la méthode d'inférence ou des paramètres de calibration de l'inférence propre à une méthode donnée : les paramétrages spécifiques sont dans les 4 premières colonnes du tableau et ils déterminent de manière unique un type d'inférence.

Ensuite, les 9 jeux de données sur lesquels le travail a été effectué sont listés en colonne.

Le second jeu de données extrait de Diogènes (« nv2 ») a été traité par une grande partie des méthodes disponibles. Le travail s'est concentré initialement sur les quatre jeux de données suivants : « ll1 », « ll2 », « nv1 », « nv2 ». Suite à plusieurs groupes de travail avec biologistes et statisticiens, des options ont été rajoutées aux programmes pour affiner les résultats (choix d'une densité cible pour les graphes résultants, par exemple) ainsi que des options supplémentaires permettant de garder (ou non) des méta-informations (comme la valeur même de la corrélation partielle). Parallèlement, des nouveaux jeux de données, qui après le retour des premiers résultats ont semblé plus pertinents, ont été ajoutés et certaines des méthodes d'inférence ont été appliquées à ceux-ci.

	méthode	sous.méthode	pénalisation	Méta.informations.incluses	ll1	ll1b	ll2	nv1	nv1b	nv2	nv3	nv4	nv5
1	glasso	aucune	aucune	aucune	x		x	x		x			
2	genenet-indep	aucune	aucune	aucune	x	x	x	x		x			
3	genenet-joined	aucune	aucune	aucune				x		x			
4	simone-indep	mbor	pénalité minimale : 0.15	aucune	x		x	x					
5	simone-indep	mbor	aucune	aucune						x			
6	simone-indep	mbor	densité maximale : 0.05	aucune		x				x			
7	simone-indep	mbor	aucune	corrélations partielles						x			
8	simone-indep	mbor	pénalité minimale : 0.15	corrélations partielles						x			
9	simone-indep	mband	pénalité minimale : 0.15	aucune	x		x	x		x			
10	simone-indep	mband	aucune	aucune						x			
11	simone-indep	mband	aucune	corrélations partielles						x			
12	simone-indep	mband	pénalité minimale : 0.15	corrélations partielles						x			
13	simone-indep	mband	densité maximale : 0.05	corrélations partielles	x	x			x		x		
14	simone-indep	mband	densité maximale : 0.10	corrélations partielles							x	x	
15	simone-indep	mband	densité maximale : 0.15	corrélations partielles								x	
16	simone-indep	friedman	pénalité minimale : 0.15	aucune	x		x	x					
17	simone-indep	friedman	aucune	aucune						x			
18	simone-indep	friedman	aucune	corrélations partielles						x			
19	simone-indep	friedman	pénalité minimale : 0.15	corrélations partielles						x			
20	simone-indep	friedman	densité maximale : 0.05	corrélations partielles	x	x			x		x		
21	simone-indep	friedman	densité maximale : 0.10	corrélations partielles							x	x	
22	simone-indep	friedman	densité maximale : 0.15	corrélations partielles								x	
23	simone-joined	coopLasso	aucune	aucune			x						
24	simone-joined	coopLasso	pénalité minimale : 0.06949	aucune	x			x		x			
25	simone-joined	groupLasso	aucune	aucune	x		x						
26	simone-joined	groupLasso	pénalité minimale : 0.0001	aucune				x		x			
27	simone-joined	intertwined	aucune	aucune			x	x					
28	simone-joined	intertwined	pénalité minimale : 0.06949	aucune	x								
29	simone-joined	intertwined	pénalité minimale : 0.0001	aucune						x			
30	simone-joined	intertwined	densité maximale : 0.05	corrélations partielles	x	x			x	x	x		x
31	simone-joined	intertwined	densité maximale : 0.10	corrélations partielles						x	x		x
32	simone-joined	intertwined	densité maximale : 0.15	corrélations partielles									x
33	therese-joined	friedman alpha=0.5	densité maximale : 0.05	corrélations partielles		x				x			
34	therese-joined	friedman alpha=0.5	densité maximale : 0.10	corrélations partielles		x				x			
35	therese-joined	friedman alpha=0.5	pénalité minimale : 0.15	corrélations partielles						x			

5.2.2. Résumé des résultats

Toutes les inférences n'ont pas abouti à des résultats convenables (réseaux trop peu denses ou beaucoup trop denses, réseaux qui semblaient peu intéressants pour l'analyse avec une faible transitivité).

Les principales conclusions qui ont pu être tirées de ces résultats sont :

1. Les jeux de données à haute dimension (lorsque le nombre d'individus est petit comparé au nombre de variables) ne donnent pas de réseaux exploitables. Ce résultat avait été montré dans le cadre du GGM dans [Verzelen, 2012].

2. Globalement, les réseaux inférés dans les deux conditions (deux races, deux sexes, avant/après régime...) sont similaires : $60 - 80\%$ d'arêtes communes aux conditions dans la plupart des jeux de données et des méthodes ;

3. Les méthodes jointes sont caractérisées par un fort taux d'arêtes communes ($70 - 90\%$) entre les deux conditions expérimentales : alors que on ne dénombre pas plus de $50 - 80\%$ d'arêtes communes lorsque l'on fait de l'inférence indépendante.

4. Dans la plupart des inférences, on observe un grand consensus entre les méthodes utilisées (à l'exception de GeneNet) : on a entre 55% et 80% d'arêtes communes entre méthodes dans les jeux de données les plus réduits.

Les deux sections suivantes présentent des résultats plus détaillés concernant deux jeux de données, un issu du projet DeLiSus et l'autre issu du projet Diogènes.

Zoom sur le jeu de données « ll1b »

Cette partie se concentre sur les résultats des inférences du jeu de données appelé « ll1b » et extrait des données DeLiSus ; il est composé des 123 gènes les plus différentiellement exprimés par rapport à la race.

Un tableau comparatif des différentes approches (conduites de telle manière à avoir 350-400 arêtes, soit une densité de graphe d'environ 5%) est proposé dans le tableau 5.1.

méthode sous-méthode	sous-méthode	simone mband	simone mbor	simone friedman	simone intertwined	therese friedman	genenet
simone	mband	382	272	270	268	215	0
simone	mbor		377	319	227	264	0
simone	friedman			390	235	293	0
simone	intertwined				353	206	0
therese	friedman					375	0
genenet							36

TABLE 5.1.: Tableau comparatif du nombre d'arêtes communes aux graphes inférés par différentes méthodes, sur le jeu de données « ll1b » pour la race Large White

Sur cet exemple, il existe un grand consensus entre les diverses méthodes issues du **simone** ; en contrôlant le nombre par le biais d'un calibrage, les réseaux présentés en contiennent tous environ 370, et le pourcentage d'arêtes communes à une paire de réseaux obtenus par deux méthodes différentes est ici entre 65 et 85%. La figure 5.2 représente la

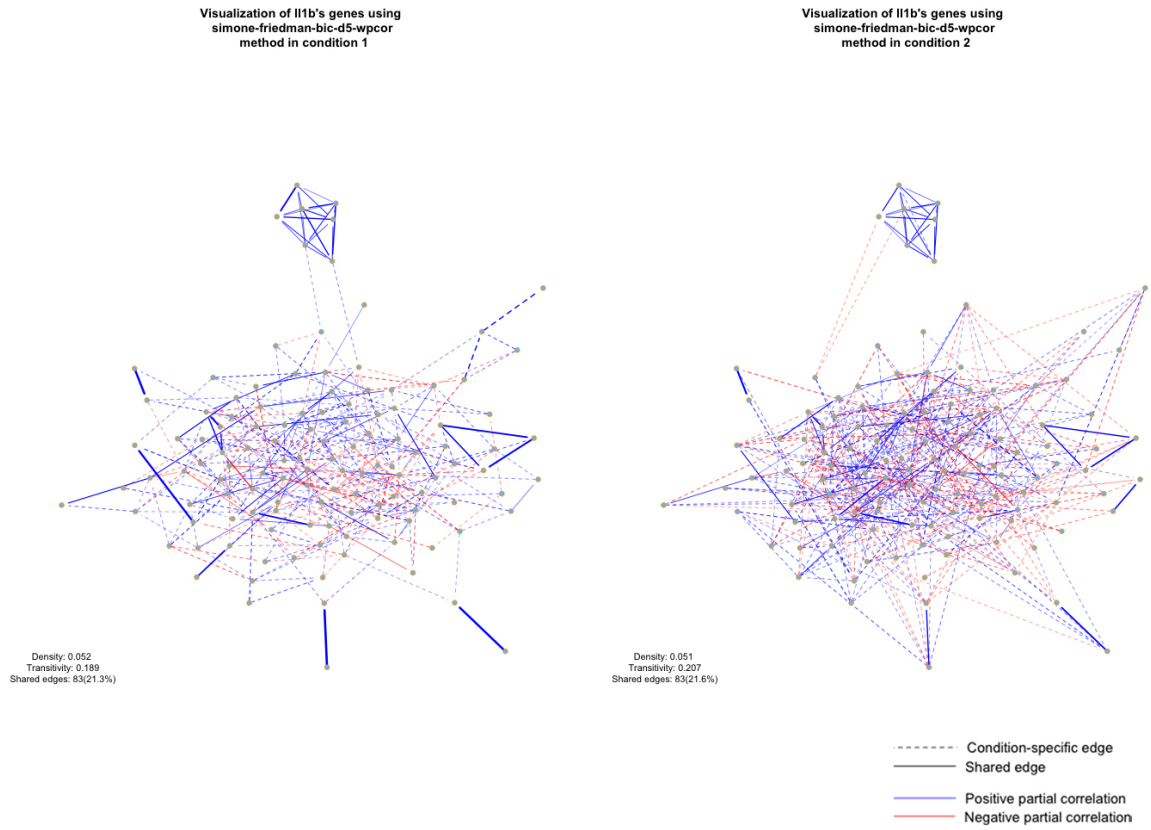


FIGURE 5.2.: Représentation des réseaux inférés à partir des 123 gènes les plus différentiellement exprimés du jeu de données *DeLiSus*, pour les cochons de race Large White et Landrace, respectivement, en utilisant l'approche de [Friedman et al., 2008] telle qu'implémentée dans le package **simone** et en fixant la densité cible à 5%. L'épaisseur des arêtes illustre la valeur de la corrélation partielle.

paire de graphes (pour les deux races Landrace et Large White) obtenue avec la méthode de [Friedman et al., 2008] telle qu'implémentée dans le package **simone**, en fixant la densité cible à 5%. Cette approche est une approche indépendante, c'est-à-dire, que les deux graphes sont inférés de manière totalement séparée. Dans ce cas, le pourcentage d'arêtes communes entre les deux graphes est faible, de l'ordre de 20% mais on retrouve quelques motifs simliaires. En particulier, on peut clairement observer des paires ou des amas de nœuds qui se distinguent des autres par leur corrélation partielle fortement positive (traits épais et bleus). Après transmission de ces résultats aux biologistes, ils se sont rendus compte que la plupart des expressions de gènes hautement corrélées étaient en fait les expressions d'un même gène placé à plusieurs endroits de la biopuce. Ce travail doit conduire à la fusion des gènes identiques, après leur identification fine, puis à l'inférence d'autres réseaux issus de ces données « nettoyées ».

Zoom sur le jeu de données « nv4 »

Dans cette section, l'intérêt se porte sur des ratio, plutôt que sur les variables elles-mêmes, avec pour particularité de mélanger des expressions de gènes et des variables cliniques. La mesure des expressions de gènes cibles et des variables cliniques a été effectuée à trois

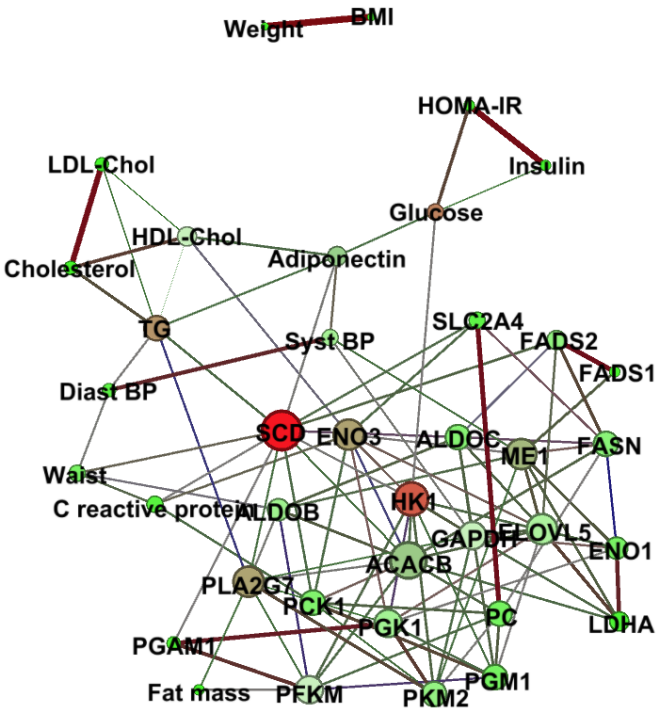


FIGURE 5.3.: Représentation du réseau des ratio d'expressions et de données cliniques entre avant et après le régime.

moments différents : au premier jour avant un régime, au dernier jour du régime, puis six mois après le régime sans que des contraintes alimentaires aient été imposés aux sujets. Les deux conditions correspondent aux ratios suivants :

$$\text{Condition 1} = \frac{\text{après régime}}{\text{avant régime}} \qquad \text{Condition 2} = \frac{\text{6 mois après le régime}}{\text{avant régime}}$$

L'idée est de capturer l'information sur quelles variables évoluent de la même manière pendant le régime, et quelles variables évoluent de la même manière du début à la fin de la période étudiée. Ceci dit, le choix des nœuds (données transcriptomiques et conditions cliniques), et le fait qu'il existe des étapes biologiques intermédiaires entre ces deux types d'informations (par exemple le métabolome) engendre que les associations inférées entre ces deux types de variables peuvent être « faibles » en comparaison des associations entre variables d'un même type.

Les figures 5.3 et 5.4 sont les deux graphes inférés avec l'approche de [Meinshausen and Bühlmann, 2006], polique « AND » avec une densité cible de 5%. Ces deux graphes sont inclus dans un article actuellement en révision pour publication dans la revue PloS Genetics [Viguerie et al., 2012]. Ils ont été représentés avec Gephi, par Nathalie Viguerie, à partir de fichiers au format GraphML qui ont été générés par les scripts décrits dans la section **??**.

Les réseaux inférés montrent que, *indépendamment de la condition expérimentale*, il existe une arête reliant le poids (« Weight ») et l'Indice de Masse Corporelle (« BMI »), ce qui est un résultat attendu et rassurant d'un point de vue biologique. Également, il existe une arête entre la pression sanguine diastolique (« Diast BP ») et systolique (« Syst BP ») ; le fait que ces variables soient corrélées, ou plus précisément dans ce contexte, qu'elles évoluent « de la même manière » est également un résultat attendu.

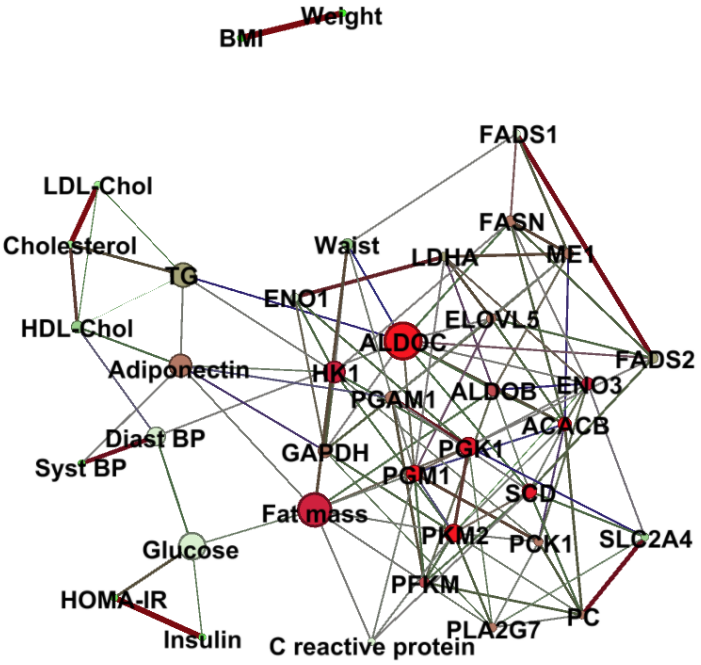


FIGURE 5.4.: Représentation du réseau des ratio d'expressions et de données cliniques entre le début et la fin de la période de suivi.

Par ailleurs, l'analyse de ces réseaux a permis des conclusions biologiques intéressantes : la couleur des arêtes renvoie à leur « betweenness » : dans la condition 1, les nœuds *SCD*, *HK1* et dans une moindre mesure le nœud « glucose » sont au milieu des plus courts chemins entre beaucoup de gènes : ils ont donc une importance particulière pour « connecté le réseau » ce qui souligne leur importance dans le processus biologique étudié.

Par ailleurs, le gène *HK1* (même si il n'était pas différentiellement régulé) de par sa grande connectivité avec des nœuds tels que *SCD* ou « glucose », a été identifiée comme un gène de grand interêt grâce à cette approche. Le nœud *SCD*, gène connu comme étant impliqué dans la liposynthèse, est connecté à l'indice du syndrome métabolique, au tour de taille et au taux de triglycérides. C'est donc la confirmation de son importance dans le mécanisme de régulation lipidique. Enfin, le réseau montre une centralité du nœud décrivant la masse lipidique (« Fat Mass »), connectée au taux de glucose (Glucose) et aux protéines C réactive (C reactive protein) ainsi qu'une centralité du nœud *ALDOC*, connecté à des gènes lipogéniques (*FASN*, *SCD*, *FADS2* et *ELOVL5*), aux triglycérides (« TG ») et au tour de taille (« waist »).

Quatrième partie .

Conclusion

Après avoir pris connaissance des cadres mathématique et biologique ainsi que des modalités de mon stage, des comparaisons d'inférence de réseaux sur des données simulées ont été effectuées. Ensuite, à l'aide de scripts rédigés de manière générique et entièrement documentés en anglais, des graphes ont été inférés sur des données transcriptomiques d'origines diverses. Suite à une validation biologique des réseaux retenus, certains d'entre eux ont été intégrés dans un article en révision pour une publication dans le journal scientifique *PLoS Genetics*. Le travail effectué sera aussi présenté lors des *Rencontres R de BoRdeaux*, les 2 et 3 juillet 2012[3].

D'un point de vue professionnel, j'ai eu l'occasion de construire et de documenter des outils génériques et pertinents pour l'analyse de données transcriptomiques, puis de les utiliser avec l'objectif d'enrichir les connaissances scientifiques dans le domaine génétique.

D'un point de vue personnel, ce stage m'a enrichi de plusieurs manières : tout d'abord, la difficulté de m'approprier un nouveau modèle mathématique, couplé avec la rigueur exigée par le besoin de programmation générique m'ont stimulé au point de dépasser mes propres attentes quand au travail fourni. Le principe du groupe de travail, que je ne connaissais pas réellement jusqu'a ce stage, m'a beaucoup plu et motivé car il s'agissait d'exposer un travail, d'échanger des idées avec des professionnels, et de pouvoir valoriser mes compétences et mes acquis, et de ce fait m'a permis de développer ma confiance en moi.

[3] http://r2012.bordeaux.inria.fr

Cinquième partie .

Annexes

A. Programmes réalisés

A.1. Organisation des programmes

L'organisation des programmes est décrite dans le schéma de la figure A.1.

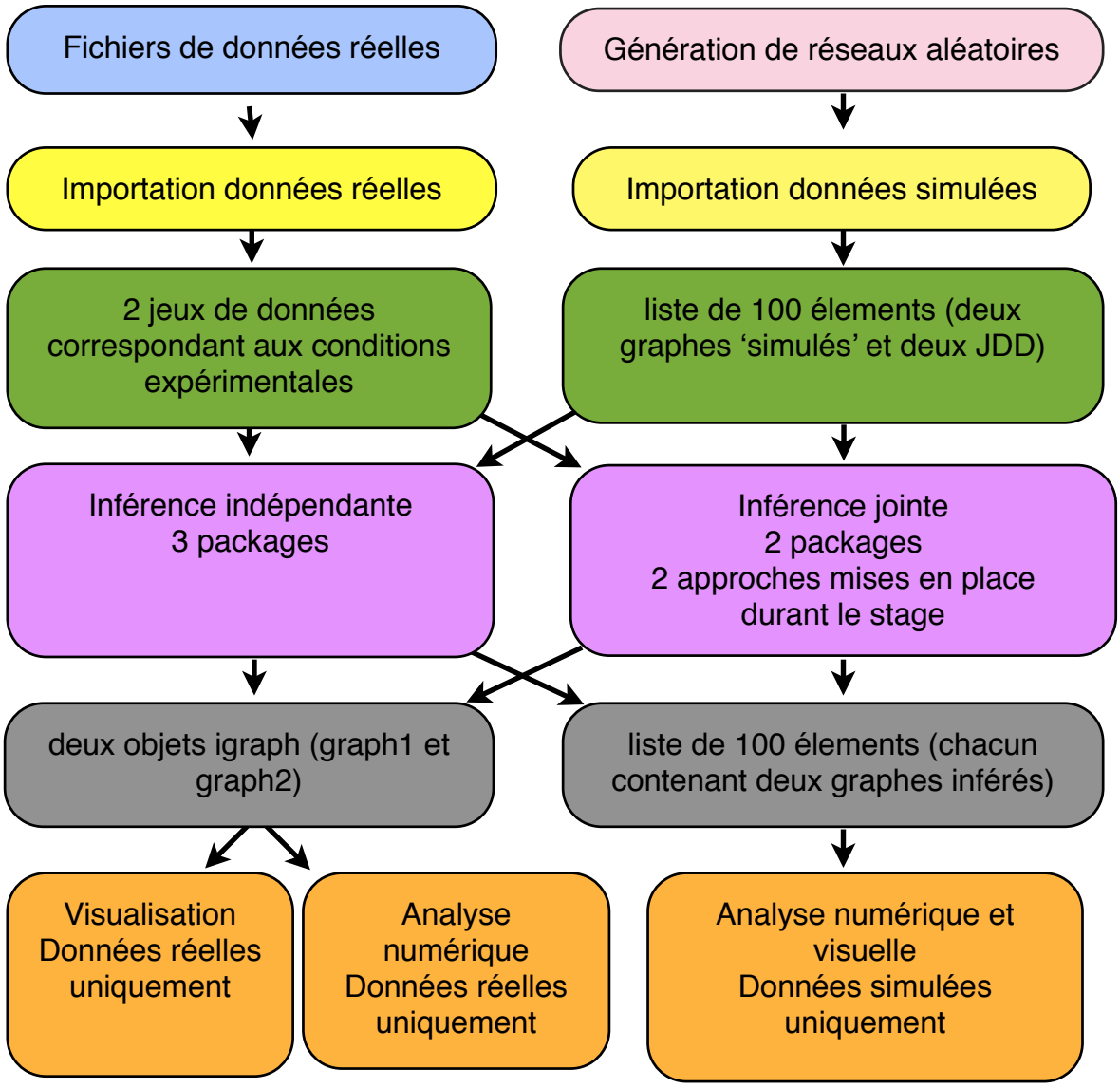


FIGURE A.1.: Organisation des programmes

Pour chaque fonction réalisée, la documentation en anglais des diverses options de la fonction a été insérée en début de fichiers. En parallèle, un fichier README (présent

dans l'annexe A.3), également en anglais, a été rédigé afin de donner une vue d'ensemble des divers scripts disponibles et de leurs usages. Les noms des fichiers de script ont été pensés de manière à ce que deux scripts ayant des usages similaires aient des noms parallèles (par exemple `export_graph-genenet-indep.R`, `export_graph-glasso-indep.R` pour les scripts contenant permettant d'inférer un réseau avec les packages, respectivement **GeneNet** et **glasso** et `run-genenet-indep.R` pour le script contenant les appels de la fonction contenue dans `export_graph-genenet-indep.R`).

A.2. Scripts R

A.2.1. Importation et mise en forme des divers jeux de données

Les fichiers de données ont été fournis sous des formes diverses (format `csv`, `Rdata` ; un seul fichier de données pour les deux conditions étudiées ou un fichier de données par condition). De plus, certains jeux de données nécessitaient des pré-traitements supplémentaires (suppression de certaines variables à la demande des biologistes, par exemple). Un script d'importation a donc été créé pour chaque fichier de données afin d'harmoniser ces formats et d'obtenir, pour chaque fichier de données, deux objets de type `data.frame`, nommés respectivement `cond1` et `cond2` et contenant les observations des expressions de gènes dans chacune des deux conditions. Au total, il y a 6 fichiers d'importation : `import_ll1.r`, `import_ll2.r`, `import_nv1.r`, `import_nv1b.r`, `import_nv2.r` et `import_nv3.r`.

Dans l'exemple ci dessous, `import_nv1.r`, on importe des données à partir de deux fichiers texte et on élimine certaines variables non souhaitées pour le traitement final (BAALC, RPL6, etc) de ces données.

```
# 214 obese people, 271 genes.
cond1<-NULL
cond1<-read.table("../Data/data-nv/norm_gene_exp-S1.txt")
cond1 <- cond1[,-match(c("BAALC","RPL6","IRX1","FUZ"),names(cond1))]
dim(cond1)

cond2<-NULL
cond2<-read.table("../Data/data-nv/norm_gene_exp-S2.txt")
dim(cond2)
cond2 <- cond2[,-match(c("BAALC","RPL6","IRX1","FUZ"),names(cond2))]
```

A.2.2. Inférences de réseaux

Comme précisé dans le Chapitre 4, l'inférence de réseaux a été réalisée à partir de divers packages R, en prenant en compte diverses options. Dans cette section, sont listées les diverses fonctions permettant d'inférer un réseau selon une méthode donnée ainsi que quelques exemples d'appels de ces fonctions.

Inférence de réseaux indépendants par glasso

La fonction `creategraph.glasso` du fichier `export_graph-glasso-indep.R` permet d'inférer un réseau à partir du package **glasso**, avec l'approche de [Meinshausen and Bühlmann, 2006], stratégie OR. Elle prend pour argument `condition`, qui est un objet de type `data.frame` contenant les observations des données d'expression. En particulier, les sorties des scripts décrits dans la section A.2.1 peuvent être passées

directement comme entrée de cette fonction. Une deuxième option est `lambda`, qui est la valeur du paramètre de pénalisation utilisée lors de l'inférence.

```
#-creategraph.glasso------------------- #
#
# Input :   dataset (condition), penalty scalar (lambda)
# Output : -function : graphe (graph1, graph2)
#                  -console : "vertices", "edges", "density", "is.
   connected"
#                  -file : 2 files : (to ../Results/'dataset'/
   Independent/)
#                                graph-'dataset'-'condition'-glasso-'
   method'.graphml
#                                graph-'dataset'-'condition'-glasso-'
   method'.Rdata
#                                example : graph-ll2-cond1-glasso-mbor.
   graphml
# Description : creates a graph using the Meinshausen and Buhlmann
   method, with the "or" policy (package glasso)
# outputs the graph and logs basic info to the console, as well as
   three files : two graph files and a Rdata file.
#------------------------------ #

creategraph.glasso<-function(condition, lambda)
        {
        est.nets2 <- glasso(cov(condition, use="pairwise.complete.obs"
           ),rho=lambda, approx=T)
        est.edges1.o<-(abs(est.nets2$wi)!=0)|(t(abs(est.nets2$wi)!=0))
        diag(est.edges1.o)<-FALSE
        graph<-graph.adjacency(est.edges1.o, mode="undirected")
        V(graph)$name<-names(condition)
        print(paste(length(V(graph)), length(E(graph)), graph.density(
           graph), is.connected(graph)))
        graph
        }
```

Inférence de réseaux indépendants par GeneNet

La fonction `creategraph.genenet` contenu dans le fichier `export_graph-genenet-indep.R` permet d'inférer un réseau à partir de données individuelles d'expression de gènes par le package **GeneNet**. La fonction construit et exporte le graphe obtenu par sélection des arêtes après un test de significativité d'un niveau de 5%.

```
#-------------------------------------------------------------
#
##creategraph.genenet(condition) -
#       This file, contains a function to create a graph
#       from the condition (entered as an argument) using cond1 and
   cond2 objects in R
#       with the genenet approach.
#       Then, it exports graphs and information to various files.
#
#       Input : the condition for which one wants to compute a graph.
#       When names are available in the dataset, it adds them to the
   vertices.
#
```

```
#       A function used to build the graph is called twice per dataset
    (for both condition)
#       Additionaly, before the function call on nv2 data, a function
   is used on all columns (in practice, on clinical data) to replace
   missing values with the mean of the column (to preserve average and
    variance)
#
#       Output :  ( to "../Results/Independent/data-xx/")
#               a .RData file ("graph-xxx-condx-genenet.RData")
   containing 2 R objects : graph1 and graph2 for both conditions resp
   .
#               2 .graphml standard graph files ("graph-xxx-condx-
   genenet.graphml")
#               2 edgelist ("edgelist-xxx-condx-genenet.txt") files
   with one pair of gene names representing an edge
#
#-------------------------------------------------------------


createGraph.genenet<-function(condition) {
        # for dealing with "NA"
        condition[is.na(condition)]<-min(condition, na.rm=T)
        # build network
        estimated.pcor<-ggm.estimate.pcor(condition, "static")
        dim(estimated.pcor)
        est.pval <- ggm.test.edges(estimated.pcor,plot=F)
        sel.edges <- which(est.pval$qval<0.05) # qval is the corrected
            p-value
        # selected edges
        genenet.edges <- est.pval[sel.edges,2:3]
        genenet.edges<-as.matrix(genenet.edges)

        # T/F matrix instead of edges list
        mat<-matrix(FALSE, nrow=nrow(estimated.pcor), ncol=ncol(
           estimated.pcor))
        mat[genenet.edges]<-TRUE
        mat<-mat+t(mat)

        # create graph
        graph<-graph.adjacency(mat, mode="undirected")
        #V(graph)$name <- names(condition)

        #print basic info
        print(cat(
                "Vertices : ",length(V(graph)),
                "\n Edges : ",length(E(graph)),
                "\n Density : ",graph.density(graph),
                "\n Transitivity : ",transitivity(graph),
                "\n Connectivity : ",is.connected(graph), "\n"))

        # output graph
        graph
        }
```

Le fichier `run-genenet-indep.R` contient les appels à cette fonction pour générer les différents réseaux. Un exemple d'appel de cette fonction est donné ci-dessous :

```
library(igraph)
```

```
library(GeneNet)
source("export_graph-genenet-indep.R")
# ll1 dataset
source("import_ll1.r")
graph1<-creategraph.genenet(cond1)
V(graph1)$nom1<-genes.names$V1
V(graph1)$nom2<-genes.names$V2
V(graph1)$nom3<-genes.names$V3
write.graph(graph1, file="../Results/data-ll/Independent/graph-ll1-
   cond1-genenet.graphml", format="graphml")
write.table(get.edgelist(graph1), file="../Results/data-ll/Independent
   /edgelist-ll1-cond1-genenet.txt", row.names=F, col.names=F)
```

Inférence de réseaux indépendants par simone

La fonction `creategraph.simone` contenue dans le fichier `export_graph-simone-indep-bic` permet d'inférer un réseau à l'aide d'une méthode choisie (et passée comme argument) parmi plusieurs proposées (l'approche de [Meinshausen and Bühlmann, 2006], stratégie AND, l'approche de [Meinshausen and Bühlmann, 2006], stratégie OR ou l'approche de [Friedman et al., 2008]). Le package simone infère une succession de réseaux, en diminuant pas à pas le paramètre de pénalisation λ, ce qui a pour effet d'augmenter progressivement le nombre d'arêtes. On obtient par ce biais une succession de réseaux à nombre d'arêtes croissant. Il existe différents critères de choix dans `simone` ; dans ce script, le BIC peut être utilisé pour choisir le réseau optimum (c'est d'ailleurs l'option par défaut). La sélection peut aussi se faire en fixant le nombre maximum d'arêtes, ou la valeur minimale de λ avant l'exécution, auquel cas l'inférence s'arrête quand la valeur spécifiée est dépassée. On peut aussi choisir de garder la valeur de la corrélation partielle pour chaque arête inférée ; cette dernière option permettra, lors de la visualisation du graphe, un affichage différentié des arêtes selon la valeur et le signe de la corrélation partielle.

```
# This file sources 3 datasets, and contains a function that is used
   to create a graph/network with the package "simone"
# It works with two "conditions", and thus creates two networks, that
   are independent (i.e. they were not created with prior knowledge of
    data from the other condition)
# the main function is "creategraph.simone"
# It has four arguments :
#       - condition, the condition dataset's name in the sourced
   datasets (in "cond1, cond2")
#       - method, either "mbor", "mband" or "friedman"
#       - min.pen : tha value of the penalty scalar for which simone
   should stop processing ( in big datasets, if this option is not set
   , simone will
# go "out of convergence" )
#       - cor : whether the script should keep partial correlation
   values in the edge attributes of the igraph object
# (it will add -wpcor (With Partial CORrelations) to the end of the
   file name, may need to be changed to something more consistent than
    this)
#       - max.edges : (numeric) simone will stop inferring edges at
   the first network whose number of edges is above max.edges.
```

```
#       - use.bic (boolean) whether to use the BIC to choose networks,
    or to force last generated network selection. Made to be used with
    max.edges or min.pen.
#
#
#
#
#It  outputs, for each condition and method it is called for:
# -1 .Rdata file containing graph1 and graph2, the graphs for both
   conditions
# -2 .txt edgelists for both conditions
# -2 .graphml graph files for both conditions
#


creategraph.simone<-function(condition, method, min.pen=NULL, cor=F,
   max.edges=NULL, use.bic=T) {

        # condition=cond1
        # method="mband"
        # cor=T
        # min.pen=NULL
        # max.edges=78
        # use.bic=T

        # for dealing with "NA"
        condition[is.na(condition)]<-min(condition, na.rm=T)
        # build network
        if (length(min.pen)==0) {
                if(length(max.edges)==0) {
                        #min.pen is not set, max.edges either
                        simonesays<-switch(method,
                                "mbor"=simone(condition, control=
                                   setOptions(edges.steady="
                                   neighborhood.selection", edges.sym.
                                   rule ="OR")),
                                "mband"=simone(condition,control=
                                   setOptions(edges.steady="
                                   neighborhood.selection", edges.sym.
                                   rule ="AND")),
                                "friedman"=simone(condition, control=
                                   setOptions(edges.steady= "graphical
                                   .lasso"))
                        )
                } else {
                        #min.pen is not set, max.edges is
                        simonesays=switch(method,
                                "mbor"= simone(condition, control=
                                   setOptions(edges.steady="
                                   neighborhood.selection", edges.sym.
                                   rule ="OR", edges.max=max.edges)),
                                "mband"= simone(condition,control=
                                   setOptions(edges.steady="
                                   neighborhood.selection", edges.sym.
                                   rule ="AND", edges.max=max.edges)),
                                "friedman"= simone(condition, control=
                                   setOptions(edges.steady= "graphical
                                   .lasso", edges.max=max.edges))
```

```
                        )
                }
        } else {
                if(length(max.edges)==0) {
                        #min.pen is set, max.edges is not
                        simonesays<-switch(method,
                                "mbor"= simone(condition, control=
                                   setOptions(edges.steady="
                                   neighborhood.selection", edges.sym.
                                   rule ="OR",penalty.min=min.pen)),
                                "mband"= simone(condition,control=
                                   setOptions(edges.steady="
                                   neighborhood.selection", edges.sym.
                                   rule ="AND",penalty.min=min.pen)),
                                "friedman"= simone(condition, control=
                                   setOptions(edges.steady= "graphical
                                   .lasso",penalty.min=min.pen))
                        )
                } else {
                        #both min.pen and max.edges are set
                        simonesays<-switch(method,
                                "mbor"= simone(condition, control=
                                   setOptions(edges.steady="
                                   neighborhood.selection", edges.sym.
                                   rule ="OR",penalty.min=min.pen,
                                   edges.max=max.edges)),
                                "mband"= simone(condition,control=
                                   setOptions(edges.steady="
                                   neighborhood.selection", edges.sym.
                                   rule ="AND",penalty.min=min.pen,
                                   edges.max=max.edges)),
                                "friedman"= simone(condition, control=
                                   setOptions(edges.steady= "graphical
                                   .lasso",penalty.min=min.pen, edges.
                                   max=max.edges))
                        )
                }
        }

        # select best network (maximum BIC)
        if (use.bic){
        simoneadj<-simonesays$network[[which.max(simonesays$BIC)]]
        }else{
        # choose the last network inferred
        simoneadj<-simonesays$networks[length(simonesays$networks)
           ][[1]]
        }
        heatmap(simoneadj)
        # create graph
        if(cor) {
                graph<-graph.adjacency(simoneadj-diag(diag(simoneadj))
                   , mode="plus", weighted=T)
                # not sure about which methods give which partial
                   correlation values, will see.
                E(graph)$weight<-(-E(graph)$weight)

        } else {
                        # create T/F matrix
```

```
                simone.edges<-(simoneadj!=0)
                diag(simone.edges)<-FALSE
                graph<-graph.adjacency(simone.edges, mode="undirected"
                    )
        }

        #print basic info
        print(cat(
        # removed the graph name because of a crash when sources with
           parallel-export_simone-indep.R
                "Graph generated using ", method," approach; \n",
                "Vertices : ",length(V(graph)),
                "\n Edges : ",length(E(graph)),
                "\n Density : ",graph.density(graph),
                "\n Transitivity : ",transitivity(graph),
                "\n Connectivity : ",is.connected(graph), "\n", sep=""
                    )
        )


        # output graph
        graph
}
```

Le fichier `run-simone-indep.R` applique la fonction `creategraph.simone` aux divers jeux de données pour inférer des graphes. Un extrait de ce fichier est présenté ci-dessous.

```
library(igraph)
library(simone)
source("export_graph-simone-indep-bic.R")

# ll1 dataset
## ll1 MB-OR, min.pen=0.15
source("import_ll1.r")
graph1<-creategraph.simone(cond1, "mbor",0.15)
V(graph1)$nom1<-genes.names$V1
V(graph1)$nom2<-genes.names$V2
V(graph1)$nom3<-genes.names$V3
graph.density(graph1)
write.graph(graph1, file="../Results/data-ll/Independent/graph-ll1-
   cond1-simone-mbor-bic.graphml", format="graphml")
write.table(get.edgelist(graph1), file="../Results/data-ll/Independent
   /edgelist-ll1-cond1-simone-mbor-bic.txt", row.names=F, col.names=F)
```

Inférence de réseaux joints par GeneNet

Dans la fonction `creategraph.genenet.joined` du fichier `export_graph-genenet-joined.R`, on utilise le package **GeneNet** pour inférer des réseaux pour des mesures appariées pour chaque condition (c'est à dire lorsque les individus sont les mêmes entre les deux condition) ; l'idée sous-jacente étant que comme les individus sont les mêmes, on peut s'attendre à ce qu'il y ait une part commune et une part différentiée dans les réseaux qui découlent des données : pour ce forcer la ressemblance entre les deux réseaux, on mélange donc les expressions spécifiques à une condition avec une expression moyenne inter-conditions : la nouvelle expression du gène j pour l'individu i dans la condition c (qui est utilisée pour l'inférence du réseau

spécifique à la condition c) est donc :

$$\tilde{x}_i^{j,c} = x_i^{j,c} + \bar{x}_i^j$$

où $\bar{x}_i^j = \frac{1}{2}\sum_{c=1,2} x_i^{j,c}$.

```
#   For NV datasets only
#
#       This file should be placed in the same directory as an "import
    " file.
#       This script takes
#       - a dataset name ("nv1", "nv2", etc) as input, and
# sources the corresponding import file, and "cleans" (puts n/a values
    to the corresponding variable mean).
# It then runs a custom GeneNet graph inference.
# The theory is that since the two conditions are somewhat similar,
# the resulting graphs should be forced to share common points.
# The idea is here implemented at the partial correlation estimation
   step :
# Instead of estimating separately both partial correlations, it
   estimates the pcor
# of an individual that is half "current" condition, half "average"
   condition:
# for a given condition, the values of that condition are weighted 1/
   2, then the
# "average" value across both conditions is taken and weighted 1/2,
   and that gives the
# value that is used for the estimation.
#
# The value of the function is the only output :
# let dataset="nv1"; at the end of execution
# creategraph.genenet.joined("nv1")$graph1 : contains the graph object
    for cond1
# creategraph.genenet.joined("nv1")$graph2 : contains the graph object
    for cond2
#
#
# Function calls are located in the corresponding "run" file (e.g. run
   -genenet-joined.R)
#

creategraph.genenet.joined<-function(dataset) {
        source(paste("import_", dataset, ".r", sep=""))
        if(dataset=="nv2") {
                input.mean <- function(column) {
                        column[is.na(column)]<-mean(column, na.rm=T)
                        column
                }
                # saving column names
                n1<-colnames(cond1)
                # applying function to remove NA's (removes column
                   names)
                cond1<-apply(cond1, 2, input.mean)
                cond2<-apply(cond2, 2, input.mean)
                # inserting column names
                colnames(cond1)<-n1
                colnames(cond2)<-n1
                #converting back (from matrix) to data.frame
                cond1<-data.frame(cond1)
```

```
                cond2<-data.frame(cond2)
        }
        cond1[is.na(cond1)]<-min(cond1, na.rm=T)
        cond2[is.na(cond2)]<-min(cond2, na.rm=T)

        # for first condition
        estimated.pcor1<-ggm.estimate.pcor( 0.5* cond1 + 0.5* (cond1+
            cond2) )
        est.pval1 <- ggm.test.edges(estimated.pcor1,plot=F)
        sel.edges1 <- which(est.pval1$qval<0.05) # qval is the
            corrected p-value
        # selected edges
        genenet.edges1 <- est.pval1[sel.edges1,2:3]
        genenet.edges1<-as.matrix(genenet.edges1)
        # T/F matrix instead of edges list
        mat1<-matrix(FALSE, nrow=nrow(estimated.pcor1), ncol=ncol(
            estimated.pcor1))
        mat1[genenet.edges1]<-TRUE
        mat1<-mat1+t(mat1)
        # create graph
        graph1<-graph.adjacency(mat1, mode="undirected")
        V(graph1)$name <- names(cond1)
        # print basic info
        print(cat(
                "Vertices : ",length(V(graph1)),
                "\n Edges : ",length(E(graph1)),
                "\n Density : ",graph.density(graph1),
                "\n Transitivity : ",transitivity(graph1),
                "\n Connectivity : ",is.connected(graph1), "\n"))

        # for second condition
        estimated.pcor2<-ggm.estimate.pcor( 0.5* cond2 + 0.5* (cond1+
            cond2) )
        est.pval2 <- ggm.test.edges(estimated.pcor2,plot=F)
        sel.edges2 <- which(est.pval2$qval<0.05) # qval is the
            corrected p-value
        # selected edges
        genenet.edges2 <- est.pval2[sel.edges2,2:3]
        genenet.edges2<-as.matrix(genenet.edges2)
        # T/F matrix instead of edges list
        mat2<-matrix(FALSE, nrow=nrow(estimated.pcor2), ncol=ncol(
            estimated.pcor2))
        mat2[genenet.edges2]<-TRUE
        mat2<-mat2+t(mat2)
        # create graph
        graph2<-graph.adjacency(mat2, mode="undirected")
        V(graph2)$name <- names(cond2)
        #print basic info
        print(cat(
                "Vertices : ",length(V(graph2)),
                "\n Edges : ",length(E(graph2)),
                "\n Density : ",graph.density(graph2),
                "\n Transitivity : ",transitivity(graph2),
                "\n Connectivity : ",is.connected(graph2), "\n"))

        # output graphs
        (list("graph1"=graph1, "graph2"=graph2))
}
```

Inférence de réseaux joints par simone

La fonction `creategraph.simone.joined` contenue dans le fichier `export_graph-simone-joined-bic.R` permet l'inférence jointe de réseaux (inférer un réseau pour une condition sachant l'expression des gènes dans les deux conditions). On doit préciser à la fonction quelle approche choisir pour cette inférence (parmi « GroupLasso », « CoopLasso » ou « Intertwined », comme décrit dans [Chiquet et al., 2011]), ainsi que le jeu de données (identifié par un nom, `ll1`, `nv2`, ...) pour lequel on souhaite inférer les réseaux. Diverses options sont introduites, permettant, par exemple, de garder la valeur de la corrélation partielle pour chaque arête inférée. Cette dernière option permettra, lors de la visualisation du graphe, un affichage différentié des arêtes selon la valeur et le signe de la corrélation partielle.

```
# creategraph.simone.joined
#-----Input---------------------------------------
#arguments :    method : graph construction approach ("coopLasso", "
   groupLasso" or "intertwined", see simone:setOptions and "Inferring
   Multiple Graphical Structures", Chiquet, et al., 2011)
#                              penmin : the minimal penalty level for
    which to try creating a graph (has to be between 0 and 1. By
   default, simone does this automatically  )
#       creategraph.simone.joined<-function(dataset, method, min.pen=
   NULL)
#
#       Input   : dataset: the working dataset (from ("ll1", "ll2", "
   nv1", "nv2"), for importation preprocessor)
#                       : method : graph construction approach ("
   coopLasso", "groupLasso" or "intertwined", see simone:setOptions
   and "Inferring Multiple Graphical Structures", Chiquet, et al.,
   2011)
#                       : penmin : the minimal penalty level for which
    to try creating a graph (has to be between 0 and 1. By default,
   Simone deals with penalty levels, but one can specify his in case
   simone crashes or hangs, )
#                       : max.edges (integer) in a sense, equivalent
   to penmin, will stop inferring networks once they have more than '
   max.edges' edges (the last network generated will have a number of
   edges that is equal or slightly above 'max.edges')
#                       : cor : (boolean), adjusts whether the partial
    correlations should be kept in edges. Has an effect on current
   implementation of show_graphs.

#----Function-------------------------------
#
#       Function : the function sources the "preprocessing" file (see
   "preprocess_export_graph-simone-joined.r" and corresponding readme
   entry), uses its main function and ends up with cond12 and tasks.
#It gives both objects to simone, which computes a series of graph
   pairs, selects the best pair by maximum BIC (done by simone and
   since we haven't been able to check the way simone does this, we
   are henceforth assuming that it works!(tm))
#----Output---------------------------------
# It then exports both graph's basic info to the console, and writes 5
    files per graph (10 total) :
# Per condition/dataset/method : 1 edgelist file and 1 .graphml file
# Per dataset/method : 1 .RData file containing graph1 and graph2 R
   objects.
```

```
#-------------------------------------------
# Example calls : see function calls after the function

creategraph.simone.joined<-function(dataset, method, min.pen=NULL, cor
   =F, max.edges=NULL) {
        # merging the two conditions
        res <- fusion(dataset)

        # for dealing with "NA"
        res$cond12[is.na(res$cond12)]<-min(res$cond12, na.rm=T)
        # build network
        if (length(min.pen)==0) {
                #min pen is not set
                if (length(max.edges)==0) {     # both min.pen and max
                   .edges are not set
                        simonesays <- simone(res$cond12, tasks=res$
                           tasks, control=setOptions(edges.coupling=
                           method))
                } else {
                        # min.pen is not set, max.edges is set
                        simonesays <- simone(res$cond12, tasks=res$
                           tasks, control=setOptions(edges.coupling=
                           method, edges.max=max.edges))
                }
        } else {
                #min pen is set
                if (length(max.edges)==0) {
                        # min pen is set, max.edges is not set
                        simonesays <- simone(res$cond12, tasks=res$
                           tasks, control=setOptions(edges.coupling=
                           method, penalty.min=min.pen))
                } else {
                        # both max.edges and min.pen are set
                        simonesays <- simone(res$cond12, tasks=res$
                           tasks, control=setOptions(edges.coupling=
                           method, penalty.min=min.pen, edges.max=max.
                           edges))
                }
        }
        # select best network (maximum BIC)
        simoneadj<-simonesays$network[[which.max(simonesays$BIC)]]

        # create graph
        if(cor) {
                graph1<-graph.adjacency(simoneadj[[1]]-diag(diag(
                   simoneadj[[1]])), mode="plus", weighted=T)
                E(graph1)$weight<-(-E(graph1)$weight)
        } else {
                # create T/F matrix, graph1
                simone.edges1<-(as.data.frame(simoneadj[[1]])!=0)
                diag(simone.edges1)<-FALSE
                graph1<-graph.adjacency(simone.edges1, mode="
                   undirected")
        }

        #print basic info
        print(cat(
                "Graph for ", dataset," with ", method," approach; \n"
```

```
                    ,
                "Vertices : ",length(V(graph1)),
                "\n Edges : ",length(E(graph1)),
                "\n Density : ",graph.density(graph1),
                "\n Transitivity : ",transitivity(graph1),
                "\n Connectivity : ",is.connected(graph1), "\n", sep="
                    ")
        )

        # create graph
        if (cor) {
                graph2<-graph.adjacency(simoneadj[[2]]-diag(diag(
                    simoneadj[[2]])), mode="plus", weighted=T)
                E(graph2)$weight<-(-E(graph2)$weight)

        } else {
                # create T/F matrix, graph2
                simone.edges2<-(as.data.frame(simoneadj[[2]])!=0)
                diag(simone.edges2)<-FALSE
                graph2<-graph.adjacency(simone.edges2, mode="
                    undirected")
        }

        #print basic info
        print(cat(
                "Graph for ", dataset," with ", method," approach; \n"
                    ,
                "Vertices : ",length(V(graph2)),
                "\n Edges : ",length(E(graph2)),
                "\n Density : ",graph.density(graph2),
                "\n Transitivity : ",transitivity(graph2),
                "\n Connectivity : ",is.connected(graph2), "\n", sep="
                    ")
        )

        # output graphs
        list("graph1"=graph1, "graph2"=graph2)
}
```

Cette fonction fait appel à la fonction `fusion` du script `preprocess_export_graph-simone-joined.r` qui permet de mettre en forme les données pour les passer de manière correcte en argument à la fonction `simone` du package **simone**.

```
#---------------------------------------#
# # fusion
#       Input : the dataset ("ll1", "ll2", "nv1", "nv2")
#       This function sources the specified dataset's importation
    script and thus gets cond1 and cond2 R objects. It then row-binds
    both conditions, and creates a tasks vector (whose length is equal
    to the sum of both condition's number of rows), and whose value is
    1 or 2 depending on which condition the row came from
#       Output : R objects (kept : cond1, cond1), cond12 (in theory, a
     matrix, in R, a list), tasks (in theory a vector, in practice a
    double)
#---------------------------------------#

fusion<-function(dataset) {
```

```
        # Sources the right file to import


        filepath<-paste("import_", dataset, ".r", sep="")
        source(filepath)

        # Specific pre-processing for nv2 dataset (should probably be
           put in import_nv2.r)
        if(dataset=="nv2")
        {
                input.mean<-function(column)
                {
                        column[is.na(column)]<-mean(column, na.rm=T)
                        column
                }
                # saving column names
                n1<-colnames(cond1)
                # applying function to remove NA's (removes column
                   names)
                cond1<-apply(cond1, 2, input.mean)
                cond2<-apply(cond2, 2, input.mean)
                # inserting column names
                colnames(cond1)<-n1
                colnames(cond2)<-n1
                #converting back (from matrix) to data.frame
                cond1<-data.frame(cond1)
                cond2<-data.frame(cond2)
        }
        # Creates a third element, by consolidating vertically cond1
           and cond2 into cond12
        cond12<-rbind(cond1, cond2)
        cond12<-as.matrix(cond12)

        # Creates a tasks vector for simone
        tasks<-factor(c(rep(1, nrow(cond1)), rep(2, nrow(cond2))))
        list("cond12"=cond12, "tasks"=tasks)
        }
#outputs: cond12, tasks
```

Le fichier `run-simone-joined.R` se sert de la fonction définie ci-dessus et l'applique à chaque jeu de donnée. Un extrait de ce fichier contenant un appel de fonction est présenté ci dessous :

```
library(igraph)
library(simone)
source("export_graph-simone-indep-bic.R")
# ll1 dataset
source("import_ll1.r")
graph1<-creategraph.simone(cond1, "mbor",0.15)
V(graph1)$nom1<-genes.names$V1
V(graph1)$nom2<-genes.names$V2
V(graph1)$nom3<-genes.names$V3
graph.density(graph1)
write.graph(graph1, file="../Results/data-ll/Independent/graph-ll1-
   cond1-simone-mbor-bic.graphml", format="graphml")
write.table(get.edgelist(graph1), file="../Results/data-ll/Independent
   /edgelist-ll1-cond1-simone-mbor-bic.txt", row.names=F, col.names=F)
```

A.2.3. Fonctions d'analyse et de comparaison des réseaux générés

Comparaison des graphes obtenus entre méthodes

Le fichier `compare_methods.R` contient la fonction `compare.graphs`, qui sert à comparer les graphes résultant de plusieurs méthodes, à condition et à jeu de donnée fixé (les trois renseignements sont passés en paramètre, ainsi que le répertoire du fichier où se trouvent les graphes). Quand la fonction est appelée, elle génère une matrice triangulaire, avec en ligne et en colonne les méthodes, et à l'intersection de deux méthodes, le nombre d'arêtes communes au graphe intersection (la diagonale contient le nombre d'arêtes pour la méthode), et elle écrit cette matrice dans un fichier au format csv.

```
#---    this file contains 2 functions :
#       -compare.graphs
#       -intersection.graphs
#
#
#----------     -compare.graphs
#                                (The data is assumed to be organized
   as "<data-group>/<method type>/<data>",
#                               where <data-group> should be put in '
   dir.in', and several <data>@<method type> should be put
#                               in method (The method type should be
   the name of the data's subdirectory))
# Input         : dir.in : directory containing the (input) data files
    (ex : "../Results/data-nv/")
#                       : dir.out : the directory to which the output
   files whould be put
#                       : dataset : name of the dataset in naming
   convention (ll1, ll2 or nv1)
#                       : condition (1 or 2)
#                       : methods : vector of conditions for
   comparison ,
############ (in the following naming convention : <methodname>@<
   method type>)
#
#       This function creates a symetric matrix (whose triangular
   bottom-left section will be NA) whose rows and columns are methods.
#Let i, j elements contained in 0:length(methods); j>=i;
#the [i, j]th element of the matrix will be the number of common edges
    in the graphs built by both methods, with the specified condition
   and dataset
#the [i, i] (or [j, j])th element of the matrix is the number of edges
    in the graph built by the ith (or jth) method, with the specified
   condition and dataset.
# It is filled in practice by the function "intersection.graphs"
#
#
# Output        : CSV, triangular matrix with list of conditions in
   row and column, and at each point, the
#                       number of edges present in graphs obtained by
   both methods, issued from a specified condition and dataset.
#
#----------



#----------     -intersection.graphs (this function calculated the
```

```
   number of common edges from graphs obtained by 2 specified methods,
    from a specified condition and dataset)
#
# Input         : dir.in : directory containing the data files
#                        : dataset : name of the dataset in naming
   convention (ll1, ll2 or nv1)
#                        : condition (1 or 2)
#                        : meth1 : first method (in methods vector)
#                        : meth2 : second method (in methods vector)
#
# Output        : 1 scalar : the number of intersecting edges in both
   methods for specified dataset and condition (or the number of ).
#
#----------

compare.graphs<-function(dir.in, dir.out="../Results/comparison/",
   dataset, condition, methods) {
        # dir.in="../Results/data-nv/"
        # dataset="nv2"
        # methods=c("glasso-mbor@Independent", "simone-mband-
           bic@Independent", "simone-mbor-bic@Independent", "simone-
           friedman-bic@Independent", "genenet@Independent", "simone-
           cooplasso-bic@Joined", "simone-grouplasso-bic@Joined", "
           simone-intertwined-bic@Joined")
        # condition=1

        # initialization of the matrix
        mat.compare<-matrix(nrow=length(methods), ncol=length(methods)
           )
        for (i in 1:length(methods)) {
                # for a fixed value of i, gregexpr("[@]",methods[i],
                   perl=F)[[1]] is the position of the "@" symbol in
                   the string "methods[i]"
                # This code will put the methodname of the ith method
                   in meth1, and the associated subdirectory ("Joined"
                    or " Independent") in subdir.in1
                method1<-substr(methods[i],1, gregexpr("[@]",methods[i
                   ],perl=F)[[1]]-1)
                subdir.in1<-substr(methods[i], gregexpr("[@]",methods[
                   i],perl=F)[[1]]+1, nchar(methods[i]))
                #meth1 : the first method's name
                #sub1 : the subdirectory for the first method
                for (j in i:length(methods)) {
                        method2<-substr(methods[j],1, gregexpr("[@]",
                           methods[j],perl=F)[[1]]-1)
                        subdir.in2<-substr(methods[j], gregexpr("[@]",
                           methods[j],perl=F)[[1]]+1, nchar(methods[j
                           ]))
                        mat.compare[i,j]<-intersection.graphs(dir.in,
                           subdir.in1, subdir.in2, dataset, condition,
                            method1, method2)
                }
        }
        rownames(mat.compare)<-methods
        colnames(mat.compare)<-methods
        write.csv(mat.compare, paste(dir.out,"compare-edges-",dataset,
           "-cond",condition,".csv", sep=""))
        tex.mat.compare <- mat.compare
```

```
        rownames(tex.mat.compare) <- paste("$[",1:length(methods),"]$"
            ,sep="")
        colnames(tex.mat.compare) <- paste("$[",1:length(methods),"]$"
            ,sep="")
        # scaling by the minimum number of edges between two methods
        D1 <- diag(mat.compare)%*%t(rep(1,length(methods)))
        D2 <- rep(1,length(methods))%*%t(diag(mat.compare))
        scalingby <- D1*((D1-D2)<0)+D2*((D2-D1)<=0)
        tex.mat.compare <- tex.mat.compare/scalingby
        xtable(tex.mat.compare)
}

# function called on each iteration
intersection.graphs<-function(dir.in, subdir.in1, subdir.in2, dataset,
    condition, method1, method2) {
        # load graph method 1
        datafile<-paste("graph-",dataset, "-cond12-",method1,".RData",
            sep="")
        # edge list graph method 1
        load(paste(dir.in, subdir.in1, "/", datafile, sep=""))
        if (condition==1) {edgelist1 <- get.edgelist(graph1)}
        else {edgelist1 <- get.edgelist(graph2)}
        edgelist1 <- paste(edgelist1[,1],rep("-",nrow(edgelist1)),
            edgelist1[,2],sep="")
        rm(graph1)
        rm(graph2)
        # load graph method 2
        datafile<-paste("graph-",dataset, "-cond12-", method2,".RData"
            , sep="")
        load(paste(dir.in, subdir.in2, "/", datafile, sep=""))
        # edge list graph method 2
        if (condition==1) {edgelist2 <- get.edgelist(graph1)}
        else {edgelist2 <- get.edgelist(graph2)}
        edgelist2 <- paste(edgelist2[,1],rep("-",nrow(edgelist2)),
            edgelist2[,2],sep="")
        # Number of edges both graphs have in common
        common.edges <- intersect(edgelist1,edgelist2)
        if (length(edgelist1)==0 & length(edgelist2)==0) {res <- 0}
        else {res <- length(common.edges)}
        res
        }
```

Comparaison des graphes obtenus entre conditions

Le fichier `compare_conditions.R` contenant la fonction `compare.conditions` est utilisé pour la comparaison des résultats de deux conditions dans un jeu de données. On passe à la fonction le jeu de données, son emplacement dans l'arborescence des fichiers, et une liste de méthodes, et il crée une matrice : pour chaque méthode, en ligne, on retrouve le nombre d'arêtes inférées en condition 1, en condition 2 et à l'intersection des deux conditions. La fonction écrit ensuite cette matrice au format `.csv`.

```
        #-------------------- compare.conditions<-function(dir.in,
            dataset, methods)
        #This function allows one to compare the edge count of graphs
            created from two conditions (and their intersection) across
             several methods.
```

```
        #
        ## -Input       : dir.in : the directory where the .RData file
            containing objects "graph1" and "graph2" are stored
        #                : dir.out : the directory where the
           output files should be saved.
        #                : dataset : the name of the current
           dataset
        #                : methods : a vector of method names,
           formatted in the following fashion : <methodname>@<
           subdirectory>
        #
        #
        # It creates a matrix, containing as many lines as "methods"
           has elements, and three columns.
        # For each method (a line in a matrix)
        # the function will load the corresponding .RData file and
           output a line, containing :
        # 3 elements (respectively "number of edges in condition 1", "
           number of edges in condition 2", and "number of edges in
           intersection") to said matrix.
        #
        #       -Output : matrix as a .csv file with the following
           naming convention : "compare-edges-",dataset,"-cond12inter.
           csv"
        #
        # An example call to this function would be :
        #
        ## compare.methods("../Results/", "ll1", c("glasso-mbor", "
           simone-mband-bic", "simone-mbor-bic", "simone-friedman-bic
           "))
        #
        ## where :
        # dir.in : "../Results/data-ll/"
        # dir.out : by default, "../Results/comparison"
        # dataset : "ll1"
        # methods : c("glasso-mbor@Independent", "simone-mband-
           bic@Independent", "simone-mbor-bic@Independent", "simone-
           friedman-bic@Independent", "genenet@Independent", "simone-
           cooplasso-bic@Joined", "simone-grouplasso-bic@Joined", "
           simone-intertwined-bic@Joined")
        #---------------------

compare.conditions<-function(dir.in, dir.out="../Results/comparison",
   dataset, methods) {
        compare<-matrix(nrow=length(methods), ncol=3)
        stats.compare <- matrix(nrow=length(methods),ncol=5)

        for ( i in 1:length(methods)) {
                # methods loop
                # current method
                method<-substr(methods[i],1, gregexpr("[@]",methods[i
                   ],perl=F)[[1]]-1)
                subdir.in<-substr(methods[i], gregexpr("[@]",methods[i
                   ],perl=F)[[1]]+1, nchar(methods[i]))

                # load dataset
                datafile<-paste("graph-",dataset, "-cond12-",method,".
                   RData", sep="")
```

```
                load(paste(dir.in, subdir.in, "/", datafile, sep=""))
                # edge lists
                edgelist1 <- get.edgelist(graph1)
                edgelist2 <- get.edgelist(graph2)
                # update matrix
                edgelist1<-paste(edgelist1[,1],rep("-",nrow(edgelist1)
                    ),edgelist1[,2],sep="")
                edgelist2<-paste(edgelist2[,1],rep("-",nrow(edgelist2)
                    ),edgelist2[,2],sep="")
                compare[i,1]<-length(edgelist1)
                compare[i,2]<-length(edgelist2)
                compare[i,3]<-length(intersect(edgelist1, edgelist2))
                stats.compare[i,1] <- graph.density(graph1)
                stats.compare[i,2] <- graph.density(graph2)
                stats.compare[i,3] <- transitivity(graph1)
                stats.compare[i,4] <- transitivity(graph2)
                stats.compare[i,5] <- compare[i,3]/min(compare[i,1:2])
                }
        rownames(compare)<-methods
        colnames(compare)<-c("Condition 1","Condition 2","Intersection
            ")
        colnames(stats.compare) <- c("density-cond1","density-cond2","
            transitivity-cond1","transitivity-cond2","shared")
        rownames(stats.compare) <- methods
        write.csv(compare, file=paste(dir.out, "/compare-edges-",
            dataset,"-cond12inter.csv", sep=""))
        write.csv(stats.compare,file=paste(dir.out, "/stats-compare-",
            dataset,"-cond12inter.csv", sep=""))
        print(xtable(stats.compare[,c(1,3,5)]))
}
```

La comparaison des jeux de données se fait dans un fichier externe, `run-comparisons.R` qui « source » les deux fonctions (exécute les deux fichiers ci dessous et rend leurs fonctions respectives directement utilisables). Dans ce fichier, sont faits tous les appels nécessaires aux comparaison des méthodes. Un extrait est présenté ci dessous :

```
library(igraph)

## Compare conditions
source("compare_conditions.R")
compare.conditions("../Results/data-ll/", "ll1", c("glasso-
    mbor@Independent", "simone-mband-bic@Independent", "simone-mband-
    bic-d5-wpcor@Independent", "simone-mbor-bic@Independent", "simone-
    friedman-bic@Independent","simone-friedman-bic-d5-wpcor@Independent
    ", "genenet@Independent", "simone-cooplasso-bic@Joined", "simone-
    grouplasso-bic@Joined", "simone-intertwined-bic@Joined","simone-
    intertwined-bic-d5-wpcor@Joined"))

[..]

## Compare methods
source("compare_methods.R")

# create the two .csv files
compare.graphs("../Results/data-ll/", "ll1", 1, c("glasso-
    mbor@Independent", "simone-mband-bic@Independent", "simone-mbor-
    bic@Independent", "simone-friedman-bic@Independent", "
    genenet@Independent", "simone-cooplasso-bic@Joined", "simone-
```

```
    grouplasso-bic@Joined", "simone-intertwined-bic@Joined"))
```

Visualisation des graphes obtenus entre conditions

La visualisaion des graphes obtenus (avec deux conditions côte à côte) se fait au moyen du fichier `show_graphs-cond.R`. Il contient la fonction `show.graphs`, à laquelle on passe le répertoire contenant les données, le nom du jeu de données, le nom de la méthode d'inférence, puis quelques options graphiques (si l'on veut des étiquettes sur chaque nœud, leur couleur, leur forme, si l'on veut la légende, et le fichier de coordonnées des sommets dans la représentation (généré par un programme java `fr.jar`, qui m'a été fourni et implémente la méthode de [Fruchterman and Reingold, 1991] ou par la fonction contenue dans le fichier `create_layout.R`; voir ci-dessous)

```
##
#------------------ - show.graphs
# Input :       dir : the directory where the graph files are located
    ()
#                       dataset : the dataset whose graphs will be
    used
#                       method : the method used to generate the
    graphs
#                       mode="condition" // unused
#                       layout="cond1-simone-mbor-bic" : the layout
    file from which the edges and the vertices are arranged on the
    graphs.
#                       show.labels : determines whether the script
    should show the vertex names or not. defaults to false.
#                       vshape : a vector of shapes the same length as
     the number of vertices : vertices will be drawn with specified
    shape. Is not used by default
#                       col : a color vector for painting the vertices
    . Is not used by default.

################### edge.width


#
# Process :
#               Imports a layoutfile generated by "fr.jar" (for
    further documentation : ../Java-lib/README ) named with the
    following
#       scheme : "layout-",dataset, "-cond1-",method,".txt"
#               Loads both graph files
#               Gets edgelists
#
#               TODO : talk about create_layout.R
#               TODO : talk about what happens when the partial
    correlations are kept
#
#
#
#               if the "weight" attribute is set, it will use line
    widths to represent the s
#
#
#               Creates a color vector (which edges are Shared/unique)
```

```
#               plots the 2 adjacent graph representations using the
   layout file, and whose edges are colored with said color vector,
#               with basic graph information written underneath.
#
# Output: (to console) : number of edges (cond1, cond2), number of
   Shared edges
#                 (to plot device) : plot described above
#                 (to file) : 2 PNG files with the plot described
   above("displaygraphs-big-",dataset,"-",method,".png" and "
   displaygraphs-",dataset,"-",method,".png"). Sizes :
#                       -standard       : 700x500 pixels
#                       -big            : 1400x1000 pixels
#
#------------------
##

show.graph<-function(dir, dataset, method, mode="condition",layout="
   cond1-glasso-mbor", show.labels=F,vshape="circle", col="orange",
   appart.leg=F) {

# dir<-"../Results/data-nv/Independent/"
# dataset<-"nv2"
# method<-"simone-friedman-bic-wpcor"
# mode="condition"
# layout="cond1-simone-mbor-bic"
# vshape=c(rep("square", 23),rep("circle", 16))
# col=c(rep("pink", 23), rep("lightgreen", 16))
# show.labels=TRUE
# appart.leg=TRUE

        # load the layout
        layoutfile1<-paste("layout-",dataset, "-",layout,".txt", sep="
           ")
        layouttext1<-read.table(paste(dir,layoutfile1, sep=""),
           stringsAsFactors=F)
        # load the graphs
        datafile<-paste("graph-",dataset, "-cond12-",method,".RData",
           sep="")
        load(paste(dir, datafile, sep=""))


        if (is.numeric(layouttext1[,1])) {# The node numbers are used
           in the layout file
             layoutmatrix1<-as.matrix(layouttext1[order(layouttext1
                [,1]), 2:3])
        } else {# The node names are used in the layout file
             ordered.ind <- match(V(graph1)$name,layouttext1[,1])
             layoutmatrix1<-as.matrix(layouttext1[ordered.ind,
                2:3])
        }
        # edge lists
        edgelist1 <- get.edgelist(graph1)
        edgelist2 <- get.edgelist(graph2)
        edgelist1 <- paste(edgelist1[,1],rep("-",nrow(edgelist1)),
           edgelist1[,2],sep="")
        edgelist2 <- paste(edgelist2[,1],rep("-",nrow(edgelist2)),
           edgelist2[,2],sep="")
        # number of Shared edges between the two graphs
```

```
        Shared.edges <- intersect(edgelist1,edgelist2)
        # which edges?
        e1.in.Shared <- match(Shared.edges,edgelist1)
        e2.in.Shared <- match(Shared.edges,edgelist2)

        # display graphs
        if (appart.leg) {
                layout(matrix(c(1,2,1,2,3,4), ncol=2, byrow=T))
        } else {
                par(mfrow=c(1,2))
        }
        if (length(E(graph1)$weight)>0) {
                # Generate line width constant (the same for both
                   graphs for consistency
                A <- 3/max(c(abs(E(graph1)$weight),abs(E(graph2)$
                   weight)))
                # Edge color determined by sign of partial correlation
                    (if weight)
                edge.color.sign1<-rep("red", length(E(graph1)$weight))
                edge.color.sign1[(E(graph1)$weight>0)]<-"blue"
                edge.color.sign2<-rep("red", length(E(graph2)$weight))
                edge.color.sign2[(E(graph2)$weight>0)]<-"blue"

                # Edge line type (if weight)
                edges.lty.g1 <- rep(2,length(E(graph1)))
                edges.lty.g2 <- rep(2,length(E(graph2)))
                edges.lty.g1[e1.in.Shared] <- 1
                edges.lty.g2[e2.in.Shared] <- 1

                if(show.labels==T) {
                        plot(graph1,
                        layout=layoutmatrix1,
                        vertex.size=10,
                        vertex.frame.color=col,
                        vertex.color=col,
                        vertex.label=V(graph1)$name,
                        vertex.shape=vshape,
                        edge.width=0.5+A*abs(E(graph1)$weight),
                        main=paste("Visualization of ", dataset, "'s
                           genes using \n ", method, " \n method in
                           condition 1", sep=""),
                        edge.lty = edges.lty.g1,
                        edge.color=edge.color.sign1)
                } else {
                        plot(graph1,
                        layout=layoutmatrix1,
                        vertex.size=2,
                        vertex.frame.color=col,
                        vertex.label=NA,
                        edge.width=0.5+A*abs(E(graph1)$weight),
                        main=paste("Visualization of ", dataset, "'s
                           genes using \n ", method, " \n method in
                           condition 1", sep=""),
                        edge.lty = edges.lty.g1,
                        edge.color=edge.color.sign1)
                }

                text(-1,-1,paste("Density: ",round(graph.density(
```

```
        graph1),3),"\n Transitivity: ",round(transitivity(
        graph1),3),"\n Shared edges: ",length(E(graph.
        intersection(graph1,graph2))),"(",round(length(E(
        graph.intersection(graph1,graph2)))/length(E(graph1
        ))*100,1),"%)",sep=""), cex=0.9)

    if(show.labels==T) {
            plot(graph2,
            layout=layoutmatrix1,
            vertex.size=10,
            vertex.frame.color=col,
            vertex.color=col,
            vertex.label=V(graph2)$name,
            vertex.shape=vshape,
            edge.width=0.5+A*abs(E(graph2)$weight),
            main=paste("Visualization of ", dataset, "'s
                genes using \n ", method, " \n method in
                condition 2", sep=""),
            edge.lty = edges.lty.g2,
            edge.color=edge.color.sign2)
    } else {
            plot(graph2,
            layout=layoutmatrix1,
            vertex.size=2,
            vertex.frame.color=col,
            vertex.label=NA,
            edge.width=0.5+A*abs(E(graph2)$weight),
            main=paste("Visualization of ", dataset, "'s
                genes using \n ", method, " \n method in
                condition 2", sep=""),
            edge.lty = edges.lty.g2,
            edge.color=edge.color.sign2)
    }

    text(-1,-1,paste("Density: ",round(graph.density(
        graph2),3),"\n Transitivity: ",round(transitivity(
        graph2),3),"\n Shared edges: ",length(E(graph.
        intersection(graph1,graph2))),"(",round(length(E(
        graph.intersection(graph1,graph2)))/length(E(graph2
        ))*100,1),"%)",sep=""), cex=0.9)

    # displaying parts 3 and 4 of layout
    ## TODO (eventually but you'll never have the time to)
        : make it more general
    if (appart.leg) {
            plot(0,0, type="n", axes=F, yaxt="n", xaxt="n"
                , ann=F)
            legend(-0.5, 1.5, c("Clinical data", "Gene
                expressions"), pch=c(19,15), col=c("
                lightgreen", "pink"), cex=1.7, bty="n")
            plot(0,0, type="n", axes=F, yaxt="n", xaxt="n"
                , ann=F)
            legend(-1,1.5, c("Condition-specific edge", "
                Shared edge"), lty=c(2,1), cex=1.7, bty="n"
                )
            legend(-1, 0.5, c("Positive partial
                correlation", "Negative partial correlation
                "), col=c("blue", "red"), lty=c(1,1), cex
```

```
                        =1.7, bty="n")
                }

                #Printing out
                dev.print(device=png,file=paste(dir,"displaygraphs-big
                    -",dataset,"-",method,".png",sep=""), width=1400,
                    height=1000)
                dev.print(device=png,file=paste(dir,"displaygraphs-",
                    dataset,"-",method,".png",sep=""), width=700,
                    height=500)
        } else {
                #case weight does not exist - as usual
                # edges color
                col.edges.g1 <- rep("red",length(E(graph1)))
                col.edges.g2 <- rep("red",length(E(graph2)))
                col.edges.g1[e1.in.Shared] <- "blue"
                col.edges.g2[e2.in.Shared] <- "blue"
                if(show.labels==T) {
                        plot(graph1,
                        layout=layoutmatrix1,
                        vertex.size=10,
                        vertex.frame.color=col,
                        vertex.color=col,
                        vertex.label=V(graph1)$name,
                        vertex.shape=vshape,
                        edge.width=1,
                        main=paste("Visualization of ", dataset, "'s
                            genes using \n ", method, " \n method in
                            condition 1", sep=""),
                        edge.color = col.edges.g1)
                } else {
                        plot(graph1,
                        layout=layoutmatrix1,
                        vertex.size=2,
                        vertex.frame.color=col,
                        vertex.label=NA,
                        edge.width=1,
                        main=paste("Visualization of ", dataset, "'s
                            genes using \n ", method, " \n method in
                            condition 1", sep=""),
                        edge.color = col.edges.g1)
                }
                text(-1,-1,paste("Density: ",round(graph.density(
                    graph1),3),"\n Transitivity: ",round(transitivity(
                    graph1),3),"\n Shared edges: ",length(E(graph.
                    intersection(graph1,graph2))),"(",round(length(E(
                    graph.intersection(graph1,graph2)))/length(E(graph1
                    ))*100,1),"%)",sep=""), cex=0.9)

                if(show.labels==T) {
                        plot(graph2,
                        layout=layoutmatrix1,
                        vertex.size=10,
                        vertex.frame.color=col,
                        vertex.color=col,
                        vertex.label=V(graph2)$name,
                        vertex.shape=vshape,
                        edge.width=1,
```

```
                        main=paste("Visualization of ", dataset, "'s
                            genes using \n ", method, " \n method in
                            condition 2", sep=""),
                        edge.color = col.edges.g2)
                        } else {
                        plot(graph2,
                        layout=layoutmatrix1,
                        vertex.size=2,
                        vertex.frame.color=col,
                        vertex.label=NA,
                        edge.width=1,
                        main=paste("Visualization of ", dataset, "'s
                            genes using \n ", method, " \n method in
                            condition 2", sep=""),
                        edge.color = col.edges.g2)}

                text(-1,-1,paste("Density: ",round(graph.density(
                    graph2),3),"\n Transitivity: ",round(transitivity(
                    graph2),3),"\n Shared edges: ",length(E(graph.
                    intersection(graph1,graph2))),"(",round(length(E(
                    graph.intersection(graph1,graph2)))/length(E(graph2
                    ))*100,1),"%)",sep=""), cex=0.9)
                dev.print(device=png,file=paste(dir,"displaygraphs-big
                    -",dataset,"-",method,".png",sep=""),width=1400,
                    height=1000)
                dev.print(device=png,file=paste(dir,"displaygraphs-",
                    dataset,"-",method,".png",sep=""),width=700,height
                    =500)
        }
}
```

Les appels à la fonction `show.graphs` se font dans un fichier externe, `run-show_graphs-cond.R`. Ceux-ci sont faits pour divers jeux de données et méthodes. Un extrait de ce fichier est présenté ci dessous.

```
library(igraph)
source("show_graphs-cond.R")
# source("create_layout.R")

## ll1
show.graph("../Results/data-ll/Independent/", "ll1", "glasso-mbor")
show.graph("../Results/data-ll/Independent/", "ll1", "simone-mbor-bic"
    )
show.graph("../Results/data-ll/Independent/", "ll1", "simone-mband-bic
    ")
show.graph("../Results/data-ll/Independent/", "ll1", "simone-friedman-
    bic")

[..]

## nv3
show.graph("../Results/data-nv/Independent/", "nv3", "simone-mband-bic
    -d5-wpcor",layout="cond1-simone-mband-bic-d5-wpcor-igraph-fr",
    vshape=c(rep("square",38), rep("circle",12)),col=c(rep("pink", 38),
     rep("lightgreen", 12)), show.labels=T)
# create.layout("nv3","simone-mband-bic-d5-wpcor","../Results/data-nv/
    Independent/","../Results/data-nv/Independent/", "kk")
```

```
show.graph("../Results/data-nv/Independent/", "nv3", "simone-mband-bic
    -d5-wpcor",layout="cond1-simone-mband-bic-d5-wpcor-igraph-kk",
    vshape=c(rep("square",38), rep("circle",12)),col=c(rep("pink", 38),
     rep("lightgreen", 12)), show.labels=T)
# create.layout("nv3","simone-mband-bic-d10-wpcor","../Results/data-nv
    /Independent/","../Results/data-nv/Independent/", "kk")
show.graph("../Results/data-nv/Independent/", "nv3", "simone-mband-bic
    -d10-wpcor",layout="cond1-simone-mband-bic-d10-wpcor-igraph-kk",
    vshape=c(rep("square",38), rep("circle",12)),col=c(rep("pink", 38),
     rep("lightgreen", 12)), show.labels=T)
```

Pour pouvoir visualiser un réseau, il faut fournir à la fonction `plot.igraph` du package **igraph** une matrice de coordonnées pour chaque sommet. Cette matrice peut être obtenue directement à partir du package `igraph` (même si l'approche n'est pas optimale et que nous avons souvent préféré utiliser un programme java spécifique `fr.jar`. La génération des coordonnées des sommets du graphe par **igraph** peut se faire à l'aide du fichier `create_layout.R`. La fonction qu'il contient, `create.layout`, prend 5 paramètres en entrée (répertoire des données, répertoire de sortie, jeu de données, méthode et type de visualisation comme implémentées dans `create_layout.R`). Elle produit un fichier `.txt` dans le répertoire de sortie, contenant pour chaque arête du jeu de données une position x et une position y, obtenues avec le type d'agencement spécifié parmi 3.

```
###
#
#       create.new.layout
#
# This file contains a single function whose purpose is
# to generate fruchterman-reingold layouts for graph objects,
# and exports them to text files.
#
# IN :
#       -dir.in : string, the directory, relative to R-lib/, where the
    file containing the graph objects is
#       Should be named as such "graph-",dataset,"-cond12-",method,".
   RData" and should contain
#       2 graph objects named graph1 and graph2
#       -dataset (in "ll1", "ll2, "nv1", "nv2")                 }
#       -method : string, the method used to create the graph   } =>
   to reconstitute the filename
#       -dir.out : string, the directory where the file should be
   saved
#       -type : in ("fr", "kk", "lgl"),  choice of layout method :
   Fruchterman-Reingold, Kamada-Kawai, or Large Graph Layout (see
   igraph layout help)
#
# OUT
#       -2 .csv files, named as following :
#               "edgelist-igraph-", dataset, "-cond1-", method, ".txt"
#               "edgelist-igraph-", dataset, "-cond2-", method, ".txt"
#               formatted as folows : 'vertex name' 'x coordinate' 'y
   coordinate'
#                       Ex : ACACB 7.60253204478105 -13.3603667589903
#
##


create.layout<-function(dataset, method, dir.in, dir.out, type="fr") {
```

```
        # loading igraph object
        load(paste(dir.in,"graph-",dataset,"-cond12-",method,".RData",
            sep=""))
        # Creating the layout
        lay1<-switch(type,
                "fr"=layout.kamada.kawai(graph1),
                "kk"=layout.fruchterman.reingold(graph1),
                "drl"=layout.drl(graph1)
        )
        lay2<-switch(type,
                "fr"=layout.kamada.kawai(graph2),
                "kk"=layout.fruchterman.reingold(graph2),
                "drl"=layout.drl(graph2)
        )

        par(mfrow=c(1,2))
        # plotting
        plot(graph1, layout=lay1)
        plot(graph2, layout=lay2)
        # writing out graph layout
        write.table(cbind(V(graph1)$name, lay1), file=paste(dir.out, "
            layout-", dataset, "-cond1-", method, "-igraph-",type,".txt
            ", sep=""), sep=" ", quote=FALSE, row.names=FALSE, col.
            names=F)
        write.table(cbind(V(graph2)$name, lay2), file=paste(dir.out, "
            layout-", dataset, "-cond2-", method, "-igraph-",type,".txt
            ", sep=""), sep=" ", quote=FALSE, row.names=FALSE, col.
            names=F)
        list("lay1"=cbind(V(graph1)$name, lay1),"lay2"=cbind(V(graph2)
            $name, lay2) )
}

create.layout("nv2","simone-mbor-bic","../Results/data-nv/Independent/
    ","../Results/data-nv/Independent/", "drl")
```

A.2.4. Créations et traitement de jeux de données simulés

Génération de jeux de données simulés

La génération des jeux de données simulés se fait avec le fichier `generate_random_network.R`. La fonction `parallel.generation` qu'il contient prend pour argument un nombre de simulations, un répertoire, un nom de fichier, puis quelques caractéristiques des graphes que l'on veut créer et à partir desquels on veut simuler des données, et les sauvegarde dans le répertoire spécifié et avec le nom de fichier choisi. Cette fonction génère un graphe « vrai », puis, grâce à des fonctions du package **simone**, un assez grand nombre de données (expression de gènes simulées) dont l'inférence devrait amener à retrouver le graphe « vrai », puis sauvegarde le tout dans un fichier.

```
########## --- Description --- ##########
# inputs:
# number: number of random graphs to be simulated
# dir: where to save the generated graphs?
# n: vector (length 2) with the number of observations in a given task
# d: number of genes
```

```
# p: proportion of edges in the network
# add.e: proportion of permuted edges (from mother network)

# outputs: list (having length number) with tg1, tg2 being the true
   networks and cond1, cond2 being the simulated data; saved in
   corresponding directory

########## --- Function --- ##########
generate.a.random.network<-function(n,d,p,add.e) {
        pi <- matrix(p/2,ncol=5,nrow=5)
        diag(pi) <- p*1.5
        alpha <- rep(1/5,5)
        mother.graph <- rNetwork(d,pi,alpha)
        nb.edges <- sum(mother.graph$A!=0)
        true.graph1 <- coNetwork(mother.graph,round(add.e*nb.edges))
        true.graph2 <- coNetwork(mother.graph,round(add.e*nb.edges))
        cond1 <- rTranscriptData(n[1],true.graph1)
        names(cond1) <- 1:p
        cond2 <- rTranscriptData(n[2],true.graph2)
        names(cond2) <- 1:p
        true.graph1 <- graph.adjacency(true.graph1$A,mode="undirected"
           )
        V(true.graph1)$name <- 1:d
        true.graph2 <- graph.adjacency(true.graph2$A,mode="undirected"
           )
        V(true.graph2)$name <- 1:d
        list("tg1"=true.graph1,"tg2"=true.graph2,"cond1"=cond1$"1","
           cond2"=cond2$"1")
}
# Test (ll1)
# res=generate.a.random.network(c(51,33),464,0.15,0.1)
# summary(res$tg1)
# graph.density(res$tg1)
# summary(res$tg2)
# summary(graph.intersection(res$tg1,res$tg2))

# Test (nv1)
# res=generate.a.random.network(c(214,214),276,0.15,0.06)
# summary(res$tg1)
# graph.density(res$tg1)
# summary(res$tg2)
# summary(graph.intersection(res$tg1,res$tg2))

# Test (nv2)
# res=generate.a.random.network(c(204,204),39,0.1,0.06)
# summary(res$tg1)
# graph.density(res$tg1)
# summary(res$tg2)
# summary(graph.intersection(res$tg1,res$tg2))

parallel.generation <- function(number,dir,filename,n,d,p,add.e) {
        number.ind <- cut(1:number,20,labels=F)
        simus <- foreach(simu=1:20,.combine=c) %dopar% {
                nb.simu <- sum(number.ind==simu)
                the.simus <- list()
                for (ind.simu in 1:nb.simu) {
                        the.simus[[ind.simu]] <- generate.a.random.
                           network(n,d,p,add.e)
```

```
                }
        the.simus
        }
        simus
        save(simus,file=paste(dir,filename,sep=""))
}


########## --- Function calls --- ##########

library(simone)
library(igraph)
library(doMC)
registerDoMC()

# generating ll1-like data
set.seed(4534)
parallel.generation(100,dir="../Data/simulations/",filename="simu-ll1.
    Rdata",c(51,33),464,0.15,0.1)

# generating nv1-like data
set.seed(112358)
parallel.generation(100,dir="../Data/simulations/",filename="simu-nv1.
    Rdata",c(214,214),276,0.15,0.06)

# generating nv2-like data
set.seed(31415)
parallel.generation(100,dir="../Data/simulations/",filename="simu-nv2.
    Rdata",c(204,204),39,0.1,0.06)
```

Importation de jeux de données simulés

L'importation des jeux de données simulés à partir des fichiers crées par `generate_random_network.R` se fait à l'aide des fichiers `import_simu-ll1.r`, `import_simu-nv1.r` et `import_simu-nv2.r`. Lors de cette importation, les variables (qui simulent des gènes) sont identifiés par des numéros pour permettre des comparaisons avec le vrai réseau qui a permis de générer les données. Le fichier `import_simu-nv2.r` est présenté comme exemple ; les autres fichiers suivent cette trame.

```
load("../Data/simulations/simu-nv2.Rdata")
# give names to variables (genes) to avoid a bug
simus<-lapply(simus, function(x){
colnames(x$cond1)<-seq(1,ncol(x$cond1))
colnames(x$cond2)<-seq(1,ncol(x$cond2))
x$cond1<-as.data.frame(x$cond1)
x$cond2<-as.data.frame(x$cond2)
x})
```

Inférence de réseaux sur les données simulées

Au vu des grandes quantités de données à traiter, une parallélisation de l'inférence à partir des données simulées à été mise en place, ce qui permet de diviser le temps de calcul par le nombre de coeurs (processeurs) disponibles. Concrètement la parallélisation du traitement s'effectue avec les fichiers `parallel-export_genenet-indep.R`, `parallel-export_glasso-indep.R`, ...

Les fichiers `parallel` fonctionnent tous à peu près de la même manière : tout d'abord, la liste des 100 jeux de données transcriptomiques simulées est passé en argument à la fonction `parallel.inference` qui, à l'aide du package **doMC** et de la fonction `foreach`, parallélise des appels aux fonctions d'inférence de graphes décrites précédemment, et permet de récupérer la liste des 100 graphes inférés. Ces programmes ont été exécutés en des temps raisonnables sur un serveur de calcul 32 cœurs. Un exemple de fichier de parallélisation (ici, pour la méthode de [Schäfer and Strimmer, 2005], en faisant appel à la fonction `creategraph.genenet`) est donné ci-dessous :

```
library(doMC)
registerDoMC()

library(igraph)
library(GeneNet)
source("export_graph-genenet-indep.R")

parallel.inference <- function(graphlist) {
        listl <- length(graphlist)
        all.graphs <- foreach(simu=1:listl,.combine=c) %dopar% {
        g <- list()
        g[[1]] <- list()
        g[[1]]$graph1 <- creategraph.genenet(graphlist[[simu]]$"cond1"
            )
        g[[1]]$graph2 <- creategraph.genenet(graphlist[[simu]]$"cond2"
            )
        g[[1]]$number<-simu
        g
        }
        all.graphs
}

# # Function call for ll1
# source("import_simu-ll1.r")
# all.graphs <- parallel.inference(simus)
# save(all.graphs,file="../Results/simulations/simu-ll1-genenet-indep.
    RData")

# # Function call for nv1
# source("import_simu-nv1.r")
# all.graphs <- parallel.inference(simus)
# save(all.graphs,file="../Results/simulations/simu-nv1-genenet-indep.
    RData")

# Function call for nv2
# source("import_simu-nv2.r")
# all.graphs <- parallel.inference(simus)
# save(all.graphs,file="../Results/simulations/simu-nv2-genenet-indep.
    RData")
```

Comparaison de réseaux obtenus par simulation de données

Il s'agit de comparer les réseaux obtenus par inférence des données simulées aux « vrai » graphe utilisé pour générer ces données. C'est le but de la fonction `compare.methods.simul2true` du fichier `compare_methods_simul2true.R`. Cette fonction calcule, pour chacun des 100 graphes simulés par jeu de données, et chaque méthode

de la liste passée en paramètre, la différence précise en nombre d'arêtes entre les graphes inférés par la méthode considérée et le « vrai » graphe duquel les données sont issues : sont calculés, le nombre de vrais positifs, de vrai négatifs, de faux négatifs et de faux positifs, deux indicateurs de performance (la précision et le recall, obtenus à partir des 4 premières valeurs), et finalement le nombre d'arêtes communes aux deux conditions dans les « vrais » graphes et dans les graphes inférés. L'ensemble de ces informations est écrite dans un fichier au format .csv. Cette fonction édite également le nuage de points précision/recall, qui permet de comparer de les performances des diverses méthodes sur un type de graphe donné, avec une couleur pour chaque méthode passée en argument.

```
#
   ------------------------------------------------------------------------#

#
# This file is made to compare different graph inference
# methods. It is method-agnostic and only requires igraph R graph
   objects.
#
#
#
#
# It serves two related purposes : to compute for each graph given in
   input,
# several quantitative characteristics and export them as numbers,
# and  to plot the Precision- Recall curve of a series of simulated
   graphs,
# in comparison to a "true" graph,
# (used to generate simulated data that must be preemptively exported
   .)
#
# INPUT : 5 arguments, 3 have defaults (d:)
# - dir.graphs : (d:"../Results/simulations/") the directory in which
   the simulations are. They have to be named : "simu-", dataset,"-",
   method, ".RData", and contain "all.graphs", list of simulated
   graphs ($graph1, $graph2 for each component)
# - dir.truth (d:"../Data/simulations/") Directory containing the
   original graph files. They have to be named : "simu-", dataset, ".
   Rdata"
# - dir.out (d:"../Results/comparison") Directory to which the 4 files
    will be exported.
# - dataset<- the name of the dataset (for finding the correct file
   and display)
# - methods= A list of methodsone wants to compare, for example :c("
   genenet-indep", "simone-friedman-bic")
#
# It contains two functions : one is called at runtime to prepare the
   graph, and then for
# each method in the input argument, it calls the second function
   which computes the characteristics
# and lays out the points.
#
#
# OUTPUT : 3 files
# - 1 .csv data file ; colums for :
# [no. True Positives], [no. False Positives], [no. True Negatives], [
   no. False Negatives], [Precision], [Recall] for each (condition 1,
   condition 2, and sum/average))
```

```
# and at the end the inter-condition difference for : the "true"
    graphs, and each simulation
# are binded and written out.
# - 1 .png file containing the Precision-Recall curve (size : width
    =600)
#
#
    ------------------------------------------------------------------#



gen.compare<-function(dataset, dir.graphs, dir.truth, dir.out,  method
    , col.method) {
        ## inferred graphs
        load(paste(dir.graphs, "simu-", dataset,"-", method, ".RData",
            sep=""))
        ## original data
        load(paste(dir.truth, "simu-", dataset, ".Rdata", sep=""))

        # Vector Initialisation
        tp1<-NULL
        fp1<-NULL
        tn1<-NULL
        fn1<-NULL
        tp2<-NULL
        fp2<-NULL
        tn2<-NULL
        fn2<-NULL
        true.intercond<-NULL
        simul.intercond<-NULL

        for (ind in 1:length(simus)) {
                # Condition 1
                n1<-length(V(simus[[ind]]$tg1))
                #no : number of
                no.edges.graph1<-length(E(all.graphs[[ind]]$graph1))
                no.edges.tg1<-length(E(simus[[ind]]$tg1))
                no.edges.inter1<-length(E(graph.intersection(all.
                    graphs[[ind]]$graph1, simus[[ind]]$tg1)))
                tp1<-cbind(tp1, no.edges.inter1) # hit
                fp1<-cbind(fp1, no.edges.graph1 - no.edges.inter1) #
                    Alpha
                tn1<-cbind(tn1,  (n1*(n1-1))/2 - no.edges.graph1 - no.
                    edges.tg1 + no.edges.inter1 ) #correct rejection
                fn1<-cbind(fn1, no.edges.tg1 - no.edges.inter1) # Beta

                rownames(tp1)<-"True Positives c1"
                rownames(fp1)<-"False Positives c1"
                rownames(tn1)<-"True Negatives c1"
                rownames(fn1)<-"False Negatives c1"
                colnames(tp1)<-seq(1, ncol(tp1))
                colnames(fp1)<-seq(1, ncol(fp1))
                colnames(tn1)<-seq(1, ncol(tn1))
                colnames(fn1)<-seq(1, ncol(fn1))

                # Condition 2
                n2<-length(V(simus[[ind]]$tg2))
                # no : number of
```

```
            no.edges.graph2<-length(E(all.graphs[[ind]]$graph2))
            no.edges.tg2<-length(E(simus[[ind]]$tg2))
            no.edges.inter2<-length(E(graph.intersection(all.
               graphs[[ind]]$graph2, simus[[ind]]$tg2)))

            tp2<-cbind(tp2,no.edges.inter2)
            fp2<-cbind(fp2, no.edges.graph2 - no.edges.inter2)
            tn2<-cbind(tn2,  (n2*(n2-1))/2 - no.edges.graph2 - no.
               edges.tg2 +  no.edges.inter2 )
            fn2<-cbind(fn2, no.edges.tg2 - no.edges.inter2)
            rownames(tp2)<-"True Positives c2"
            rownames(fp2)<-"False Positives c2"
            rownames(tn2)<-"True Negatives c2"
            rownames(fn2)<-"False Negatives c2"
            colnames(tp2)<-seq(1, ncol(tp2))
            colnames(fp2)<-seq(1, ncol(fp2))
            colnames(tn2)<-seq(1, ncol(tn2))
            colnames(fn2)<-seq(1, ncol(fn2))
            # Comparaison inter-condition
            true.intercond<-cbind(true.intercond, length(E(graph.
               intersection(simus[[ind]]$tg1, simus[[ind]]$tg2)))
               )
            simul.intercond<-cbind(simul.intercond, length(E(graph
               .intersection(all.graphs[[ind]]$graph1, all.graphs
               [[ind]]$graph2))) )
        }
        precision1<-tp1/(tp1+fp1)
        rownames(precision1)<-"Precision c1"
        recall1<-tp1/(tp1+fn1)
        rownames(recall1)<-"Recall c1"
        precision2<-tp2/(tp2+fp2)
        rownames(precision2)<-"Precision c2"
        recall2<-tp2/(tp2+fn2)
        rownames(recall2)<-"Recall c2"

        rownames(true.intercond)<-"Intersection of conditions in
           simulated graphs"
        rownames(simul.intercond)<-"Intersection of conditions in '
           true' graphs"
        colnames(true.intercond)<-seq(1, ncol(true.intercond))
        colnames(simul.intercond)<-seq(1, ncol(simul.intercond))

        #graphs
        points(recall1, precision1, pch="o", col=col.method)
        points(recall2, precision2, pch="x", col=col.method)

        # creating a dataframe with all the info for cond1, cond2 and
           1+2
        diff.simul2true1<-rbind(tp1, fp1, tn1, fn1, round(precision1
           ,4), round(recall1,4))
        diff.simul2true2<-rbind(tp2, fp2, tn2, fn2, round(precision2
           ,4), round(recall2,4))
        diff.simul2true.a<-rbind(tp1+tp2, fp1+fp2, tn1+tn2, fn1+fn2,
           round((precision1+precision2)/2, 4), round((recall1+recall2
           )/2, 4))

        # Display Venn's diagram for the first simulation only: to be
           improved
```

```
        # plot.new()
        # draw.pairwise.venn(fp1[1]+tp1[1], fn1[1]+tp1[1], tp1[1],
           category=c("Tests Positive", "Is positive"))
        # dev.print(png, file=paste(dir.out, "display_one_vennDiag-",
           dataset,"-",method,"-simul2true.png",sep=""), width=600)

        all.info.method<-t(rbind(diff.simul2true1,diff.simul2true2,
           diff.simul2true.a, true.intercond, simul.intercond))
        all.info.method
}

compare.methods.simul2true<-function(dir.graphs, dir.truth, dir.out,
   dataset, methods) {
        plot(0,0, xlim=c(0,1), ylim=c(0,1), type="n", xlab="TPR/Recall
            (pct of correctly identified true positives out of all
           tested positives)", ylab="PPR/Precision (pct of correctly
           identified true positives out of all identified positives)"
           , main=paste("Precision-Recall scatterplot for ", dataset,
           " simulations \n obtained by multiple approaches", sep=""))

        #no : number of
        no.methods<-length(methods)
        #defining colors for each method
        colmethod<-brewer.pal(max(c(no.methods,3)), "Dark2")
        # the height of the legend may need accomodation if the number
            of datasets increases significantly
        legend("topleft", methods, col=colmethod, pch=19)
        legend("bottomright", c("condition 1", "condition 2"), col="
           black", pch=c("x","o"))

        all.res<-NULL
        for (i in 1:no.methods) {
                res<-gen.compare(method=methods[i], dir.graphs=dir.
                   graphs, dataset=dataset, dir.truth=dir.truth, dir.
                   out=dir.out, col.method=colmethod[i])
                colnames(res)<-paste(methods[i], colnames(res), sep="-
                   ")
                all.res<-cbind(all.res, res)
        }
        dev.print(png, file=paste(dir.out, "display_precision_recall-"
           ,dataset,"-methods-simul2true.png",sep=""), width=600)
        write.csv(all.res, file=paste(dir.out, "compare-",dataset,"-
           methods-simul2true.csv",sep=""))
}

########## Function calls ##############################

library(igraph)
library(RColorBrewer)
# library(VennDiagram)

# compare.methods.simul2true(dir.out="../Results/comparison/", dir.
   graphs="../Results/simulations/", dir.truth="../Data/simulations/",
    dataset="nv2", methods=c("genenet-indep", "simone-friedman-bic", "
   simone-intertwined","therese-friedman-bic-d5-a0dot5", "therese-
   friedman-bic-d10-a0dot5"))
# compare.methods.simul2true(dir.out="../Results/comparison/", dir.
   graphs="../Results/simulations/", dir.truth="../Data/simulations/",
```

```
    dataset="ll1", methods=c("genenet-indep", "simone-friedman-bic", "
   simone-intertwined"))
# compare.methods.simul2true(dir.out="../Results/comparison/", dir.
   graphs="../Results/simulations/", dir.truth="../Data/simulations/",
    dataset="nv1", methods=c("genenet-indep", "simone-friedman-bic", "
   simone-intertwined"))
# compare.methods.simul2true(dir.out="../Results/comparison/therese/",
    dir.graphs="../Results/simulations/", dir.truth="../Data/
   simulations/", dataset="nv2", methods=c("therese-friedman-bic-d5-
   a0dot25", "therese-friedman-bic-d10-a0dot25", "therese-friedman-bic
   -d5-a0dot5", "therese-friedman-bic-d10-a0dot75", "therese-friedman-
   bic-d5-a0dot5", "therese-friedman-bic-d10-a0dot75"))
compare.methods.simul2true(dir.out="../Results/comparison/nv2-
   simulations/simone-indep/", dir.graphs="../Results/simulations/",
   dir.truth="../Data/simulations/", dataset="nv2", methods=c("simone-
   friedman-bic", "simone-friedman-bic-d5", "simone-mbor-bic-d5","
   simone-mband-bic-d5"))
compare.methods.simul2true(dir.out="../Results/comparison/nv2-
   simulations/simone-intertwined/", dir.graphs="../Results/
   simulations/", dir.truth="../Data/simulations/", dataset="nv2",
   methods=c("simone-intertwined", "simone-intertwined-d5","simone-
   intertwined-d10"))
compare.methods.simul2true(dir.out="../Results/comparison/nv2-
   simulations/glasso/", dir.graphs="../Results/simulations/", dir.
   truth="../Data/simulations/", dataset="nv2", methods=c("glasso","
   glasso-pen0dot2"))
compare.methods.simul2true(dir.out="../Results/comparison/nv2-
   simulations/density5/", dir.graphs="../Results/simulations/", dir.
   truth="../Data/simulations/", dataset="nv2", methods=c("genenet-
   indep","glasso-pen0dot2","simone-friedman-bic", "simone-intertwined
   -d5","therese-friedman-bic-d5-a0dot5"))
```

A.3. Documentation

La présente documentation a été rédigée en anglais. Son objectif est dans un premier temps, de servir à l'utilisateur final de ce script, et dans un second temps de montrer comment cet ensemble de scripts fonctionne, pour qu'un utilisateur avancé puisse se le réapproprier si besoin est.

```
Readme

-------------------------------------------------------------------------------

Table of contents :

i.   General Description
ii.  Typical workflow for real data
iii. Typical workflow for simulated data
iv.  Programs description

        1. Import
        2. Sim Creation/Import
        3. Export
        5. Run Export
        4. Parallel Export
        5. Analysis
```

```
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------

--------------------------------------------------------------------------------



i.   General Description
   ------------------------------------------------------------------

        This set of tools uses existing R packages (simone, GeneNet)
           to automate the inference and comparison of gene co-
           expression networks across two experimental conditions.

        Prerequisites for understanding everything : reading my report
           .


ii.  Typical Workflow
   ------------------------------------------------------------------


        The typical workflow has three key steps : importing the data
           so that it conforms to specifications (described later on),
            inferring the graph structure for both conditions using a
           choice of methods, and finally analyzing (and, if needed,
           visualizing) the resulting graphs.

        A typical R run is shown below, for data called "nv5" in the (
           joined) simone "Intertwined"  approach, with a target edge
           count of 122.
#------------------------------Sourcing of all required functions
# Loading required libraries
library(igraph)
library(simone)
# Loading required files
source("preprocess_export_graph-simone-joined.r")
source("export_graph-simone-joined-bic.R")
source("show_graphs-cond.R")
source("create_layout.R")
#------------------------------Import
# Loading data
source("import_nv5.R")
#------------------------------Inference
# Inferring graph
res<-creategraph.simone.joined(dataset="nv5", method="intertwined",
   max.edges=122, cor=T)
graph1<-res$graph1
# Adding vertex information
V(graph1)$nom<-names(cond1)
V(graph1)$name<-names(cond1)
#Writing out all graph files
write.graph(graph1, file="../Results/data-nv/Joined/graph-nv5-cond1-
   simone-intertwined-bic-d10-wpcor.graphml", format="graphml")
write.table(get.edgelist(graph1), file="../Results/data-nv/Joined/
   edgelist-nv5-cond1-simone-intertwined-bic-d10-wpcor.txt", row.names
```

```
    =F, col.names=F)
graph2<-res$graph2
V(graph2)$nom<-names(cond2)
V(graph2)$name<-names(cond2)
write.graph(graph2, file="../Results/data-nv/Joined/graph-nv5-cond2-
    simone-intertwined-bic-d10-wpcor.graphml", format="graphml")
write.table(get.edgelist(graph2), file="../Results/data-nv/Joined/
    edgelist-nv5-cond2-simone-intertwined-bic-d10-wpcor.txt", row.names
    =F, col.names=F)
save(graph1, graph2, file="../Results/data-nv/Joined/graph-nv5-cond12-
    simone-intertwined-bic-d10-wpcor.RData")
#------------------------------Visualization
# Creating layout files
create.layout("nv5","simone-intertwined-bic-d10-wpcor","../Results/
    data-nv/Joined/","../Results/data-nv/Joined/", "kk")
# Exporting visualization
show.graph("../Results/data-nv/Joined/", "nv5", "simone-intertwined-
    bic-d10-wpcor",layout="cond1-simone-intertwined-bic-d10-wpcor-
    igraph-kk", vshape=c(rep("square", 38), rep("circle", 12)), col=c(
    rep("pink", 38), rep("lightgreen", 12)), show.labels=T, appart.leg=
    T)


        In the next part, the process will be described in further
           detail, if one wished to get his hands in the code.

        Step 1 : Import
        The data has to be organized as follows :
        - two R objects called "cond1" and "cond2" have to be created,
            whose rows should reference individuals and whose columns
           should reference genes.
        - for simulated data, an R list named "simus" must be created,
            each element of the list containing 4 objects : 2 "true"
           graphs, from which simulated data has been created named "
           tg1" and "tg2" for comparison purposes, and 2 simulated
           data sets, "cond1" and "cond2". Examples of import files
           are available in the folder (having dataset names:"ll1", "
           ll2", "nv1, "nv2", etc).
        Note : for joined approaches, a function within the inference
           function must be called with a dataset name. In this case,
           the file containing the import script has to be named like
           so : ("import_", dataset, ".r"), 'dataset' being the
           dataset's name; the script has to import the data in
           dataframes named 'cond1' and 'cond2'.

        Step 2 : Inference
        -for "real" data, independent approaches, one has to reference
            the import files so that the two R dataframes are created.
            One call has to be made per condition, and the resulting
           graph (igraph object) is the function's result.
        -for "real" data, joined approaches, one has to write an R
           import file (following conventions set in the above note),
           place it in the same directory as the inference script's
           file, then enter the dataset name in the function. both
           resulting graphs are inside the function object :
res<-[function call]; res$graph1, res$graph2
        -for simulated data, one sources the import function to import
            simulation data as described above.
```

```
        Inference Options :
                simone-based inferences
        condition : the dataset name, if not
        method : in "mbor", "mband", "friedman"
        cor : boolean : keep partial correlation (-B, as of report
           notations) as graph edge attributes in E(graph1)$weight
        min.pen : numeric, in [0,1] : the minimal penalty value (as a
           multiple of the smallest penalty value resulting in an
           empty graph) for which one wants to infer a network.
        max.edges : the maximal about of edges past which inference
           stops and the last network is saved (will have about 'max.
           edges' edges)
        use.bic : boolean : whether to use the Bayesian Information
           Criterion to select the best graph




        Step 3 : Visualization/Analysis
        -For "real" data, it is done by means of show_graphs-cond.R,



iv. Programs description
   ------------------------------------------------------------------


Requests/add_annot_info.R
        add.annotations : This function reads the specified CSV file,
           establishes a correspondence between spot names and
           annotations, and attributes the annotations to the vertices
            of the input graph objects (either in the R environment or
            specified with a dataset and a filename). It also has the
           option of saving a '.graphml' file alongside, for each
           condition.

compare.simul2true.R
        This script allows one to compare "true" known data, with data
            simulated from it.
This function takes 5 atomic arguments
dataset, the dataset for which one wants to compare simulated data to
   true data ("ll1", "nv1", and "nv2" currenty have the requisite
   files)
method, the method for which one wants to compare the simulation to
   the real data
dir.graphs, the directory in which the simulated graphs are
dir.truth, the directory in which the true graphs are
dir.out, the directory in which the results should be placed.
(of which dir.graphs, dir.truth, and dir.out have defaults)
        The function calculates the amount of [no. True Positives], [
           no. False Positives], [no. True Negatives], [no. False
           Negatives], [Precision], [Recall] for each condition of the
            dataset, and then the average/sum where appropriate. This
           generates 3 data tables, which are written to dir.out as .
           CSV files. It also traces the precision-recall curve, in
           which condition1 and condition 2 are blue and red
           respectively, and saves it in dir.out as a .png
```

```
compare.methods.simul2true.R (same as above, for several methods)
        This script is meant to compare simulated networks to the real
            network from which the simulations were built, in both
           quantitative and visual ways. This script takes 5 arguments
            as input :
- dir.graphs, dir.truth and dir.out (three directories in which are
   placed respectively the simulated graphs, the "true" graph from
   which the simulations were generated, and the directory where the
   output should be placed), as well as
- a vector of methods used to create the graph, and
- a dataset name. This function, for each graph,each condition,and
   each method passed to it, will calculate the amount of [no. True
   Positives], [no. False Positives], [no. True Negatives], [no. False
    Negatives], [Precision], [Recall], the average/sum where
   appropriate, and for each condition it will generate. This
   generates 3 data tables, which are written to dir.out as .CSV files
   . It also traces the precision-recall curve, in which condition1
   and condition 2 are crosses "x" and circles "o", respectively, each
    method is color-coded, and saves the resulting graph in dir.out as
    a .png.


clean_all_graphs.R
        This script contains a function. When it is called, it takes
           as input a R list named "all.graphs". If that list contains
            NULL elements, it will delete them and move all the
           following items in the list so that there is no NULL
           element. The function will return the resulting list.


compare_conditions.R **sourceable
        This file outputs comparisons between both conditions (number
           of edges in condition1, condition2, and the number of edges
            both conditions have in common) for a specified dataset on
            all (specified) methods
TODO : arguments

compare_methods.R  **sourceable
        This file outputs comparisons between different graph methods
           :
for a specified dataset and condition, and for each pair of methods,
   it outputs the number of common edges.
** arguments : dir.in, dir.out="../Results/comparison/", dataset,
   condition, methods

export_graph-glasso-indep.R **should be sourceable
        this file (program) sources the three above files, and
           produces for each one (each dataset) 2 graphs using the MB
           method with the "or" policy and both igraph and glasso
           packages, and outputs 5 files per dataset : .Rdata file (
           whose name is :"graph-'dataset'-'condition'-glasso-'method'
           .Rdata" and is placed in "Results/dataset-'dataset'/
           Independent/") containing the graphs for both conditions (
           labeled "graph1", and "graph2", respectively), two text
           files ("edgelist-'dataset'-'condition'-glasso-'method'.txt)
            which are plain edge lists and 2 .graphml files (named "
           graph-'dataset'-'condition'-glasso-'method'.graphml" and
```

```
            placed at the same location as the .RData file)
**Possible values for :
'dataset' : ll1, ll2, nv
'condition' : cond1, cond2
'method' : mbor

export_graph-simone-indep-mbor-bic.R **sourceable, if one has a week
   to spare
        This program sources the "import" files, and runs simone (on
           each condition of each dataset), to create the best graph,
           selected by BIC, by methods : MB "or" policy, MB "and"
           policy and Friedman, then outputs to the console the basic
           graph statistics (# of vertices, edges, density and
           connectedness), and outputs, for each dataset,
2 graphs, 2 edge lists (one per condition) and one .Rdata file with
   two objects, graph1 and graph2 (for each condition)
**Possible values for :
'condition' : cond1, cond2
'method' : mbor, mband, friedman
**note : not all output of this file is present : due to dataset ll2's
    size, we were unable to calculate successfully all data (friedman'
   s method)

export_graph-genenet-indep.R **sourceable
        This program sources the import file, and runs Genenet on each
            condition for each dataset, and creates a graph (using the
            approach documented in : Opgen-Rhein and Strimmer (2006a,b
           ) and Schafer and Strimmer (2005). It then exports the
           standard info (graph objects for R, .graphml and .txt edge
           lists. Further information is located at the beginning of
           the file)
**Possible values for : 'condition' : cond1, cond2


export_graph-simone-joined.R
TODO

export_graph-simone-joined-bic.r **sourceable, will take a _very_ long
    time to execute
        This script sources "preprocess_export_graph-simone-joined.r"
           to create the right kind of objects for simone multitask,
           executes simone multitask mode with 2 conditions, chooses
           the best pair of graphs by optimal BIC (don't ask me how,
           ask J. Chiquet, Genopole  http://stat.genopole.cnrs.fr/
           members/jchiquet/welcome, who co-wrote Simone.)
** arguments    : dataset : dataset to be used ("ll1", "ll2", "nv1", "
   nv2")
                : method : method to be used ("coopLasso", "groupLasso
                   " or "intertwined")
                : [min.pen] : minimal penalty value for generating
                   networks. If not specified, simone will take care
                   of determining the value, and may go out of
                   convergence or hang on huge datasets)


import_simu-ll1.r
        This script loads the simulation data included in ../Data/simu
           -ll1.Rdata, which contains a single list named "simus", of
```

```
            100 items. Each item contains 4 objects, two graphs tg1 and
             tg2, and two datasets, cond1 and cond2.

import_simu-nv1.r
        This script loads the simulation data included in ../Data/simu
           -nv1.Rdata, which contains a single list named "simus", of
           100 items. Each item contains 4 objects, two graphs tg1 and
            tg2, and two datasets, cond1 and cond2.

import_simu-nv2.r
        This script loads the simulation data included in ../Data/simu
           -nv2.Rdata, which contains a single list named "simus", of
           100 items. Each item contains 4 objects, two graphs tg1 and
            tg2, and two datasets, cond1 and cond2.

import_ll1.r
        this file (program) loads the data included in Data/data-ll/
           data2.Rdata and produces two objects: cond1 (LW pigs) and
           cond2 (Landrace pigs)

import_ll2.r
        this file (program) loads the data included in Data/data-ll/
           data3.Rdata and produces two objects: cond1 (LW pigs) and
           cond2 (Landrace pigs)

import_nv1.r
        this file (program) loads the data included in Data/data-nv/
           norm_gene_exp-S1.txt and Data/data-nv/norm_gene_exp-S2.txt
           and produces two objects: cond1 (before diet) and cond2 (
           after diet)

import_nv2.r
        this file (program) loads the data included in Data/data-nv/
           selVar-S1.txt and Data/data-nv/selVar-S2.txt and produces
           two objects: cond1 (before diet) and cond2 (after diet)

import_simu-nv1.r
        this script loads the simulation data included in ../Data/simu
           -ll1.Rdata, which contains a single list named "simus", of
           100 items. Each item contains 4 objects, two graphs tg1 and
            tg2, and two datasets, cond1 and cond2.

preprocess_export_graph-simone-joined.r
        This script can be used for preprocessing of data prior to
           joined simone runs. It concatenates (while keeping) both
           conditions, and creates a vector (called tasks) that allows
            us to be able to still distinguish both conditions.
** argument : dataset ("ll1", "ll2", "nv1", "nv2")


parallel-export_simone-joined.R
        This script does some preprocessing to input simulation data (
           like preprocess_export_graph-simone-joined.r, but for each
           simulation run), then calls simone with a specified "joined
           " approach ('intertwined', 'grouplasso', 'cooplasso' ) with
            a specified minimal penalty value.
```

```
parallel-export_simone-indep.R
        This script sources export_graph-simone-indep-bic.R and
           parallelizes the function, (said function is better
           described in the sourced script's readme).
The name of the .RData file containing required R objects (must be a
   list of simus containing each $cond1 and $cond2) must be specified
   before each function call, as must the name of each exported .RData
    files (containing all.graphs, a list as long as simus, each
   occurrence containing $graph1 and $graph2), after the function call
   .

parallel-export_glasso-indep.R
        This script sources export_graph-glasso-indep-bic.R and
           parallelizes the function, (said function is better
           described in the sourced script's readme).
The name of the .RData file containing required R objects (must be a
   list of simus containing each $cond1 and $cond2) must be specified
   before each function call, as must the name of each exported .RData
    files (containing all.graphs, a list as long as simus, each
   occurrence containing $graph1 and $graph2), after the function call
   .

parallel-genenet-indep.R
        This script sources export_graph-genenet-indep.R and
           parallelizes the function, (said function is better
           described in the sourced script's readme).
The name of the .RData file containing required R objects (must be a
   list of simus containing each $cond1 and $cond2) must be specified
   before each function call, as must the name of each exported .RData
    files (containing all.graphs, a list as long as simus, each
   occurrence containing $graph1 and $graph2), after the function call
   .

generate_random_network.R


rhoselect_cv.R
        this file (function) is a function with
                inputs: d (data set); lambdas (list of lambda); fold (
                   number of fold for the CV, default 10)
                outputs: a vector with cv.mse for all
** example:
library(glasso)
source("import_ll1.r")
source("rhoselect_cv.R")
lambdalist <- c(0.15,0.2,0.4,0.6)
mse.path <- rhoselect.cv(cond1,lambdalist,5)
plot(lambdalist,mse.path$mse,type="b",xlab="lambda",ylab="cv-mse")

show_graphs-cond.R
        This script takes three required arguments, and up to 8
           arguments (described below), Imports a data file and a
           layout file, extracts edgelists from data, creates a
           comparative plot for both conditions, (unique/common edges
           are respectively red/blue) using the provided layout file,
           and writes some basic graph information to the console,
           some on the plot. Finally, it exports both graphs as ".png"
```

```
            files in "dir" : "displaygraphs-big-",dataset,"-",method,"
           .png" and "displaygraphs-",dataset,"-",method,".png",
           respectively 1400x1000 pixels and 700x500 pixels.
** examples :   -show.graph("../Results/data-nv/Independent/", "nv1",
   "glasso-mbor")
               -show.graph("../Results/data-nv/Independent/", "nv2",
                  "glasso-mbor",layout="cond1-simone-mbor-bic",
                  vshape=c(rep("square", 23), rep("circle", 16)), col
                  =c(rep("pink", 23), rep("lightgreen", 16)), show.
                  labels=T)
**Arguments : (bracketed arguments "[..]" are optional)
# dir :  the directory where data is located (relative to current R-
   lib folder) ("../Results/data-nv/Independent/")
# dataset<-"nv2" (typical values : "ll1", "ll2", "nv1")
# method : the approach whose graphs we want to display ("simone-
   friedman-bic", "glasso-mbor", "simone-mband-bic", "simone-mbor-bic
   ", "genenet")
# [mode]= "condition" (unused)
# [layout] : the layout file to be used "cond1-simone-mbor-bic", "
   cond1-glasso-mbor"
# [show.labels] (boolean, defaults to FALSE) : whether to show the
   names (they have to be in the graph object as a vertex attribute,
   and called 'name' to be displayed) of the vertices on the graph or
   not.
# [vshape]="circle", vector of shapes for vertices, same length as V(
   graph)
# [col]="orange", vector of colors for vertices, same length as V(
   graph)

tune_glassopath-indep.R **definitely not sourceable, erroneous lambda
   values lead to R crashing; to be used only with ll1 data
        this file (program) sources the three above files, and
           produces for each one (each dataset) one image (to Results/
           data-ll/glassopath-'dataset'-'condition'-density.png)
           containing 6 line plots representing the values for graph
           density and graph transitivity (as well as graph
           connectedness) for different values of the sparse L1
           penalization factor lambda (also called rho). Three pairs
           of line plots are shown per image file : graphs done with
           the following methods : Meinshausen and Buhlmann "and"
           policy, Meinshausen and Buhlmann "or" policy, and Friedman
           et al.
Each pair contains a line plot of graph density (and points indicating
    the graph connectedness), and a line plot of graph transitivity,
   each for different values of lambda.
```

B. Article présenté aux Rencontres R BoRdeaux 2012

Le travail effectué durant mon stage sera présenté aux 1ères Rencontres R de Bordeaux[1], le 3 juillet 2012. L'article résumant l'objectif du travail est inclus ci-dessous dans le rapport.

[1] http://r2012.bordeaux.inria.fr

Comparison of network inference packages and methods for multiple network inference

Nathalie Villa-Vialaneix[a], **Nicolas A. Edwards**[b], **Laurence Liaubet**[b]
Nathalie Viguerie[c], **Magali SanCristobal**[b]

[a]Laboratoire SAMM - Université Paris 1 (Panthéon-Sorbonne)
90 rue de Tolbiac, 75013 Paris - France
nathalie.villa@univ-paris1.fr

[b]INRA, UMR444 - Laboratoire de Génétique Cellulaire
F-31326 Castanet Tolosan cedex, France
nicolas.ae@free.fr, {laurence.liaubet,magali.san-cristobal}@toulouse.inra.fr

[c]Inserm UMR1048, Obesity Research Laboratory
I2MC, Institute of Metabolic and Cardiovascular Diseases
CHU Rangueil, Toulouse
nathalie.viguerie@inserm.fr

Keywords: network inference, transcriptomic data, gene co-expression network, Gaussian graphical model, multiple graphical structure

Integrative and systems biology is a very promising tool for deciphering the biological and genetic mechanisms underlying complex traits. In particular, gene networks are used to model interactions between genes of interest. They can be defined in various ways, but a standard approach is to infer a co-expression network from genes expression measured by means of sequencing techniques (for example, microarrays). Among methods used to perform the inference, **Gaussian graphical models** (GGM) are based on the assumption that the gene expressions are distributed as Gaussian variables, and Σ is their covariance matrix. Non-zero partial correlations between two genes are modeled by network edges, and are directly obtained from the inverse of Σ. But it turns out that estimating the inverse of Σ leads to an ill posed problem, since this kind of data leads to a number of observations (typically less than one hundred) that is usually much smaller than the number of variables (the number of genes/nodes in the network can range from a few hundred to several thousands). To overcome this difficulty, the seminal papers [8, 9] were the basis for the R package **GeneNet**, in which the partial correlation is estimated either by means of a bootstrap approach (not available in the package anymore) or of a shrinkage approach. More recently, the ability to handle genomic longitudinal data was also added as described in [7]. Then, [6] and later [3] introduced sparse approaches, both implemented in the R package **glasso** (graphical LASSO). Similarly, [4] describes the methods implemented in the package **parcor** that provides several regularization frameworks (PLS, ridge, LASSO...) to infer networks by means of Gaussian graphical models. Finally, [2, 1] describe several extensions of the Gaussian graphical model implemented in the package **simone** such as latent variable models and time-course transcriptomic data.

In systems biology, an interesting issue is to link gene functioning to an external factor. Thus, transcriptomic data are often collected in different experimental conditions. One must then understand which genes are correlated *independently* from the condition and which ones are correlated *depending* on the condition, under the plausible biological assumption that a common functioning should exist regardless of said condition. A simple naive approach would be to infer a different network from each sample, and then to compare them. Alternative approaches

are described in [2, 1] and implemented in **simone**: the log-likelihood can be penalized by a modified group-LASSO penalty or the empirical covariance matrix can be modified by adding a component depending on all samples. The purpose of this communication is to present a full comparative case study of this problem on two real data sets.
The first dataset has been collected during the DiOGenes project[1]: a few hundreds human obese individuals were submitted to a 8 weeks low calorie diet. The expressions of pre-selected genes as well as physiological variables (age, weight, waist size...) were collected *before* and *after* the diet (see [5] for further information). The underlying issue is to understand how the diet has affected the correlations between all these variables. The second data set has been collected during the Delisus project[2]: the expression of several thousands genes were collected from 84 pigs (in both *Landrace* and *Large White* breeds). The underlying issue is to understand how the breed affects the correlations between a set of selected genes which were found to be differentially expressed for the breed.
The comparison is lead by using independent inference from the packages **GeneNet**, **glasso** and **simone** or by using the different joint models included in **simone** or even by proposing new joint approaches based on the aforementionned packages. Networks are inferred from the previously described real datasets or from simulated datasets that mimic the real ones. The proximity between networks inferred from different methods or from different conditions is assessed by means of common edge counts, or, when available, by the accuracy of the inferred network when compared to the true one. A biological discussion about the relevance of the inferred networks will also be provided.

References

[1] J. Chiquet, Y. Grandvalet, and C. Ambroise. Inferring multiple graphical structures. *Statistics and Computing*, 21(4):537–553, 2011.

[2] J. Chiquet, A. Smith, G. Grasseau, C. Matias, and C. Ambroise. SIMoNe: Statistical Inference for MOdular NEtworks. *Bioinformatics*, 25(3):417–418, 2009.

[3] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[4] N. Kraemer, J. Schaefer, and A.L. Boulesteix. Regularized estimation of large-scale gene regulatory networks using Gaussian Graphical models. *BMC Bioinformatics*, 10:384, 2009.

[5] T.M. Larsen, S.M. Dalskov, M. van Baak, S.A. Jebb, A. Papadaki, A.F.H. Pfeiffer, J.A. Martinez, T. Handjieva-Darlenska, M. Kunešovà, M. Pihlsgård, S. Stender, C. Holst, W.H.M. Saris, and A. Astrup. Diets with high or low protein content and glycemic index for weight-loss maintenance. *New England Journal of Medicine*, 363:2102–2113, 2010.

[6] N. Meinshausen and P. Bühlmann. High dimensional graphs and variable selection with the lasso. *Annals of Statistic*, 34(3):1436–1462, 2006.

[7] R. Opgen-Rhein and K. Strimmer. Inferring gene dependency networks from genomic longitudinal data: a functional data approach. *REVSTAT*, 4:53–65, 2006.

[8] J. Schäfer and K. Strimmer. An empirical bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 21(6):754–764, 2005.

[9] J. Schäfer and K. Strimmer. A shrinkage approach to large-scale covariance matrix estimation and implication for functional genomics. *Statistical Applications in Genetics and Molecular Biology*, 4:1–32, 2005.

[1] supported by funding from the European Communities (DiOGenes, FP6-513946, MolPAGE, LSHG-CT-2004-512066 and ADAPT, HEALTH-F2-2008-2011 00), Fondation pour la Recherche Médicale and Région Midi-Pyrénées `http://www.diogenes-eu.org`

[2] funded by the ANR, `http://www.inra.fr/les_partenariats/programmes_anr/genomique/genanimal/appel_a_projets_2007/delisus`

Bibliographie

[Bastian et al., 2011] Bastian, M., Heymann, S., and Jacomy, M. (2011). Gephi : an open source software for exploring and manipulating networks. In Adar, E. e. a., editor, *Proceedings of the Third International AAAI Conference on Weblogs and Social Media*, pages 361–362. Menlo Park : AAAI Press, 2009.

[Chiquet et al., 2011] Chiquet, J., Grandvalet, Y., and Ambroise, C. (2011). Inferring multiple graphical structures. *Statistics and Computing*, 21(4) :537–553.

[Edwards, 1995] Edwards, D. (1995). *Introduction to Graphical Modelling*. Springer, New York.

[Friedman et al., 2008] Friedman, J., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3) :432–441.

[Fruchterman and Reingold, 1991] Fruchterman, T. and Reingold, B. (1991). Graph drawing by force-directed placement. *Software-Practice and Experience*, 21 :1129–1164.

[Meinshausen and Bühlmann, 2006] Meinshausen, N. and Bühlmann, P. (2006). High dimensional graphs and variable selection with the lasso. *Annals of Statistic*, 34(3) :1436–1462.

[R Development Core Team, 2012] R Development Core Team (2012). *R : A Language and Environment for Statistical Computing*. Vienna, Austria. ISBN 3-900051-07-0.

[Schäfer and Strimmer, 2005] Schäfer, J. and Strimmer, K. (2005). An empirical bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 21(6) :754–764.

[Verzelen, 2012] Verzelen, N. (2012). Minimax risks for sparse regressions : ultra-high-dimensional phenomenons. Preprint.

[Viguerie et al., 2012] Viguerie, N., Montastier, E., Maoret, J., Roussel, B., Combes, M., Valle, C., Villa-Vialaneix, N., Iacovoni, J., Martinez, J., Holst, C., Astrup, A., Vidal, H., Clément, K., Hager, J., Saris, W., and Langin, D. (2012). Determinants of human adipose tissue gene expression : impact of diet, sex, metabolic status and *cis* genetic regulation. Submitted to PLoS Genetics.

[Villa-Vialaneix et al., 2012] Villa-Vialaneix, N., Edwards, N., Liaubet, L., and Viguerie, N. (2012). Comparison of network inference packages and methods for multiple network inference. In *1ères Rencontres R BoRdeaux*, BoRdeaux, France. To appear.

C. Travail préliminaire d'analyse des données

Lors de mon projet de deuxième année de DUT STID, j'avais commencé à travailler sur des données similaires a été mêné en binôme avec Étienne Auclair. En utilisant des techniques statistiques usuelles (analyse univariée, ANOVA, ACP) et nouvelles (Forêts aléatoire), le but était de repérer quels gènes expliquaient une variable quantitative ou qualitative d'intérêt. Le rapport complet de ce projet est disponible ci-dessous.

Etienne AUCLAIR
Nicolas EDWARDS

Projet de statistique :

Analyse de données transcriptomiques en relation avec des phénotypes

2ème année STID - 2011-2012

Nathalie VILLA-VIALANEIX

Table des matières

Introduction

La biologie des systèmes, notamment la génétique est un secteur de recherche relativement nouveau, seules les technologies modernes permettent des mesures suffisamment précises pour faire évoluer ce domaine. L'amélioration de ces technologies au fil du temps permet une expansion de ce domaine de recherche, et ainsi, de comprendre de mieux en mieux le vivant. Par exemple, les puces à ADN, appelées aussi « microarray » ou « biopuces » apparues il y a une quinzaine d'années, permettent de mesurer massivement l'expression de dizaine de milliers de gènes à la fois. Cette technique s'est généralisée grâce au développement de la robotique qui permet d'obtenir de forts rendements et une haute précision dans les manipulations. Malgré cela, l'immense diversité des espèces vivantes est telle que le génome, ainsi que toutes les activités biologiques liées (transcription, mutations...) de la majorité des espèces n'est que peu connu, voire inconnu.

Dans ce projet, nous travaillerons sur des données récoltées par L'INRA de Toulouse (laboratoire LGC (génétique cellulaire), département GA (génétique animale), projet DeLiSus financé par l'ANR[1]), qui a mis en œuvre une grande expérience afin de mieux comprendre les comportements génétiques de différentes races de cochons d'élevage et le lien entre les spécificités génétiques de ces races et des phénotypes d'intérêt relatifs à la qualité ou la quantité de viande produite : 111 cochons (tous des mâles), de 5 races différentes, ont été élevés en station d'élevage expérimentale, en différents lots appelés « bandes ». Les conditions d'élevage dans les différentes bandes étant, au maximum, calibrées de manière identique, afin de l'imiter l'impact environnemental sur les relevés de l'expérience.

Après abattage des cochons, on a relevé, pour chacun d'eux, de multiples mesures. Une partie de ces données a donné lieu au jeu de données sur lequel nous travaillons dans ce projet et qui comprend, pour chaque cochon :

- ***La race des animaux (variable qualitative nominale)***. 5 races sont présentes dans cette étude : Duroc, Landrace, Large White F (LWF, cochon élevé pour ses qualités maternelles), Large White M (LWM, cochon élevé pour la qualité de sa viande) et Dchn.
- ***La « bande » d'élevage des animaux (variable qualitative nominale)***. Tous les animaux n'ont pas tous été élevés en même temps, un système de lots de cochons élevés en même temps, avec les mêmes conditions, est là pour limiter, ou au moins contrôler l'impact environnemental sur les relevés biologiques. Il y a 8 bandes au total.
- ***2 phénotypes d'intérêt, relatifs à la taille de la carcasse (Variables quantitatives continues)***.
- ***L'expression de 4674 gènes (variables quantitatives continues).*** Ces mesures sont obtenues grâce à un système de biopuce (expliqué en début d'introduction, et détaillé en annexe). Une expression de gène élevé traduit une forte expression de ce gène dans le tissu étudié (ici le muscle). Les données d'expression ont été préalablement nettoyées (filtrage, transformation logarithmique, mise à l'échelle, normalisation entre les diverses expériences par Magali SanCristobal, biostatisticienne à l'INRA) et sélectionnées (seuls les gènes dont la variabilité d'expression était supérieure à un seuil donné ont été conservés sur la dizaine de milliers de gènes que comportait le jeu de données initial).

Toutes ces mesures donnent le jeu de données final contenant 111 individus pour 4678 variables, dont l'individu statistique est le cochon. À partir de ces données, on cherche à connaître les gènes dont l'expression est influencée par la race, et aussi, les gènes influant sur la taille des carcasses des bêtes.

Une analyse exploratoire sera tout d'abord nécessaire, afin de connaître le comportement global des données, par des analyses univariées et bivariées, ainsi qu'une analyse en composantes principales des données d'expression. Par la suite le principal travail consistera en la sélection des

1 Agence Nationale de la Recherche : http://www.agence-nationale-recherche.fr/

gènes différentiellement exprimés selon la race, et les phénotypes d'intérêt, par le biais de méthodes telles les ANOVA ou les forêts aléatoires.
L'analyse exploratoire consistera dans un premier temps en une analyse rapide des expressions des gènes, avant d'analyser par des méthodes uni et bivariées les facteurs race et bande, puis les deux mesures du phénotype d'intérêt. Dans une deuxième temps, une analyse multivariée sera réalisée à partir de toutes les expressions gènes.
Par la suite, par le biais de forêts aléatoires, et après une brève présentation de cette méthode ,seront effectués des prédictions de la race, puis du phénotype, expliqué par les expressions des gènes, dans le but d'en relever les gènes importants.

I - Étude exploratoire

1) Étude des expressions de gènes

L'objet de l'étude est principalement les expressions des gènes, qu'il convient donc d'étudier par des méthodes de statistique descriptive. Pour se faire, il convient de considérer, pour cette partie seulement, non pas les individus en tant que tels, mais des résultats de statistique descriptive de chaque expression de gène mesuré, comme la moyenne des expressions des gènes, montrant si un gène est plutôt exprimé ou peu exprimé chez la grande majorité des individus, et aussi l'écart-type, montrant les gènes dont l'expression est très variable ou plutôt constante. La Figure 1 donne la répartition des 4674 expressions moyennes de gènes (sur les 111 individus).

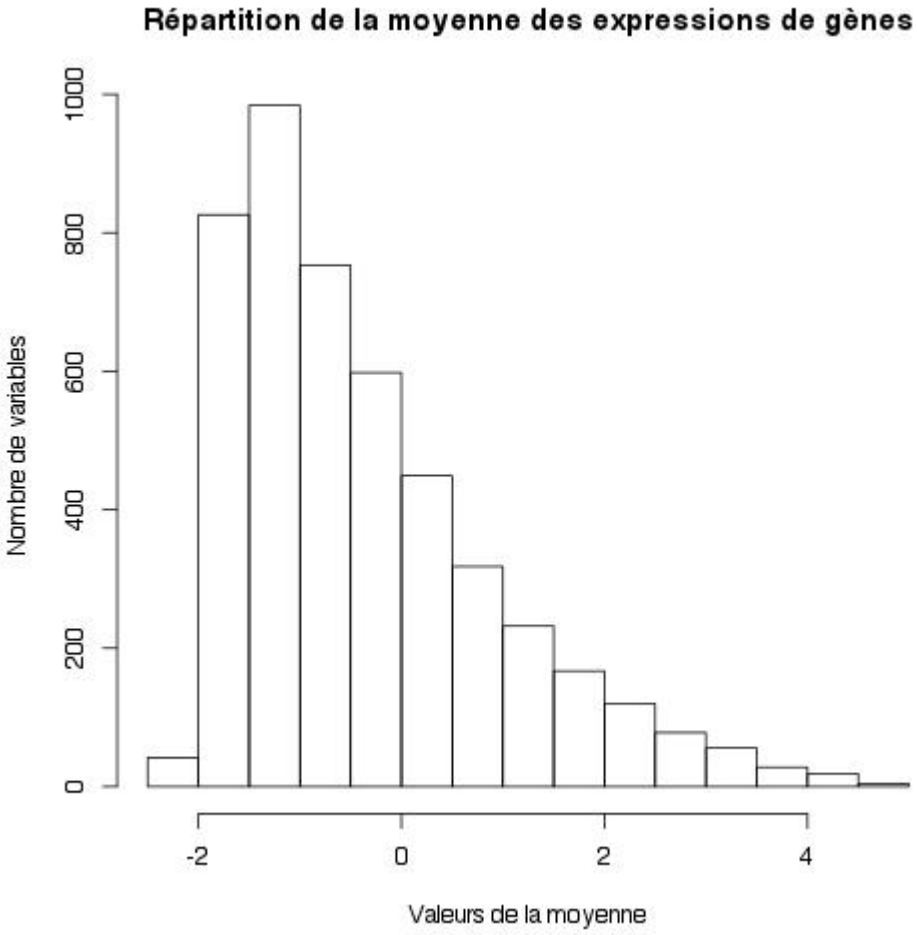


Figure 1 Répartition de la moyenne des expressions de gènes

On remarque que la distribution des moyennes des expressions des gènes est très dissymétrique : il y a peu de gènes de très faible expression, mais la plupart des gènes sont d'expression assez faible (entre -2 et -0,5). Le nombre de gènes décroit très vite en fonction de l'expression moyenne : il n'y a finalement que peu de gènes très fortement exprimés en moyenne. Le coefficient d'aplatissement (Kurtosis) est d'environ 0,919, c'est à dire qu'il y a un pic assez prononcé. Cela est dû au fait qu'un grand nombre de gènes ont une expression moyenne comprise entre -2 et -0,5. Le coefficient d'asymétrie, lui (Skewness) est d'environ 1,123, signe d'une distribution asymétrique, avec un regroupement vers les valeurs négatives, traduisant l'étalement des valeurs à droite. Ce phénomène est sûrement dû au fait que les gènes ont été prélevés dans des muscles, où les cellules sont très spécialisées, ayant une fonction essentiellement

motrice. La plupart des gènes ne s'expriment que très peu dans cette partie du corps, et de moins en moins de gènes s'expriment plus fortement.
Le Tableau 1 donne une liste de gênes atypiques négativement (dont l'expression moyenne est inférieure à -2) et le Tableau 2 donne une liste de gênes atypique positivement (dont l'expression moyenne est supérieure à 4).

X6425	X6528	X6921	X7004	X8718	X10164	X10593	X10755	X12285
X13135	X13964	X14079	X14508	X15082	X15745	X16270	X16473	X16804
X16861	X17501	X17529	X17576	X17609	X17746	X18172	X19329	X21128
X21919	X24625	X25918	X26319	X26997	X27052	X27610	X27840	X27999
X28593	X31254	X31406	X33923	X34736	X36686			

Tableau 1: Gènes d'expression moyenne faible

X309	X756	X1467	X2430	X3142	X5013	X5693	X6641	X6787	X10412
X13788	X20140	X25314	X28517	X29462	X30387	X33530	X37762	X39616	X39947
X40486	X42514	X44761							

Tableau 2 : Gènes d'expression moyenne élevée

La Figure 2 donne l'histogramme de la répartition de l'écart type des expressions des gènes.

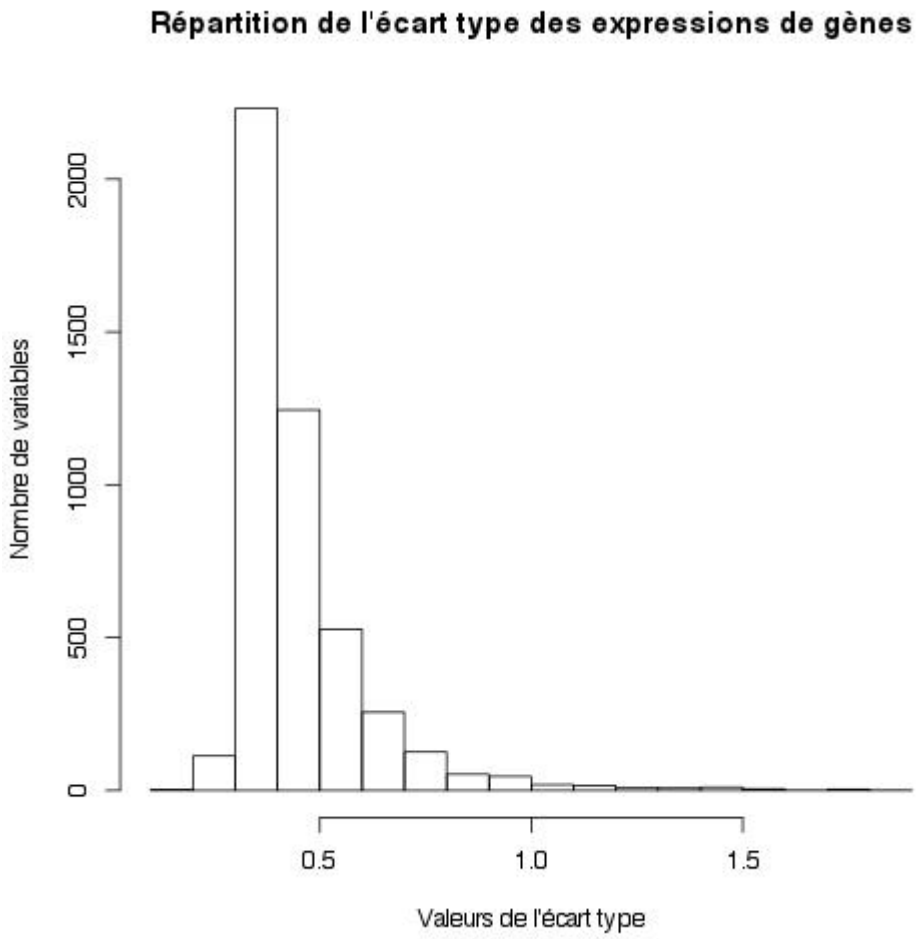


Figure 2 : Répartition de l'écart type des expressions des gènes

Ici, le phénomène est équivalent à celui retrouvé dans l'étude de la répartition de la moyenne, mais de façon

beaucoup plus marquée : une écrasante majorité (plus de la moitié) des gènes ont une expression d'écart-type compris entre 0,3 et 0,4, il y a donc un pic très important, traduit par un Kurtosis très élevé (environ 13,423). Les expressions des gènes ayant des écarts types plus élevés sont très rapidement minoritaires, traduit par un coefficient d'asymétrie également élevé (environ 3,004).
Alors que les gènes ayant un écart-type de l'expression faible sont probablement des gènes commun à tous les individus, donc codant pour l'état de cochon, voire de mammifère de l'individu, les gènes dont l'écart-type de l'expression est plus fort, donc aux valeurs plus distinctes entre chaque individu, sont susceptibles d'être spécifiques à la race, voire à l'individu lui même.
Le Tableau 3 donne une liste de gènes « extrêmes » négativement (écart-type de l'expression inférieur à 0,25) et le Tableau 4 une liste de gènes dont l'expression a une variabilité très forte (écart type supérieur à 1,5)

X839	X1115	X1860	X4164	X6964	X7533	X8489	X10858
X18429	X18574	X24342	X24741	X38047	X40666	X42245	X43177

Tableau 3: Liste de gènes d'écart type très faible

X3404	X5499	X6097	X9208	X9797	X10266	X10620	X12685	X18492	X26143	X29114

Tableau 4: Liste de gènes de très forte variabilité d'expression

On remarque qu'aucun gène extrême (positivement ou négativement) du point de vue de la moyenne ne l'est du point de vue de l'écart-type, et réciproquement.

2) Étude des races et des bandes

Chaque cochon se distinguant par une race et une bande, il convient d'étudier ces deux paramètres, qui peuvent avoir une influence importante sur le reste des données. Le Tableau 5 donne les effectifs d'individus dans chaque race, ainsi que la Figure 3.

Race	Dchn	Duroc	Landrace	LWF	LWM
Effectif	3	24	33	41	10

Tableau 5: Répartition des diverses races de cochon

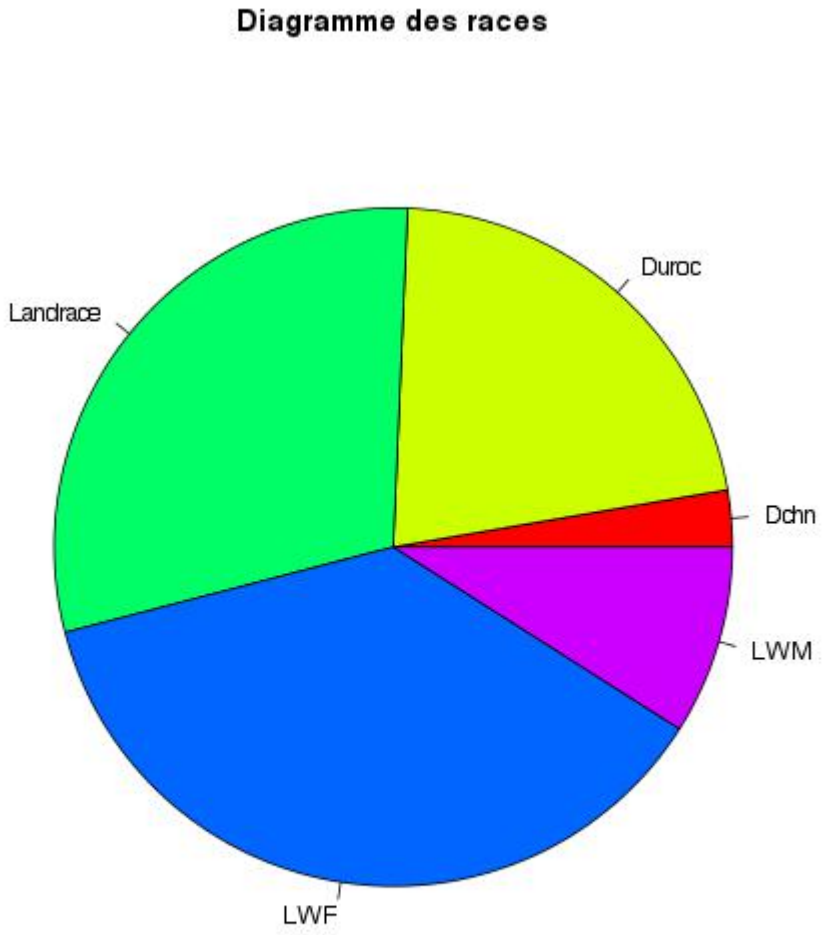


Figure 3 : Distribution des races dans l'échantillon

On remarque deux races majoritaires : LWF et Landrace, comprenant plus de la moitié des individus, et, dans une moindre mesure, les Duroc, également assez présents. LWN et Dchn sont deux races très peu présentes, notamment Dchn, qui ne comprend que 3 individus. L'étude se voulant avoir un souci d'homogénéité entre les individus, une telle disparité pourrait fausser les résultat, cela montre que les données ne sont pas idéales d'un point de vue statistique.

Chaque cochon a été élevé dans une « bande » donnée, ensemble de cochons élevés en même temps dans un même environnement (tous les animaux ne pouvaient pas être élevés en même temps pour des questions de capacité de la station d'élevage). Cela a pour but de limiter l'impact environnemental sur le phénotype. Le Tableau 6 donne la répartition des effectifs dans les différentes bandes, de même que la Figure 4.

Id de la bande	70602	70604	70606	70608	70610	80604	80606	80610
Effectif	7	13	12	27	18	6	9	19

Tableau 6: Distribution des différentes bandes

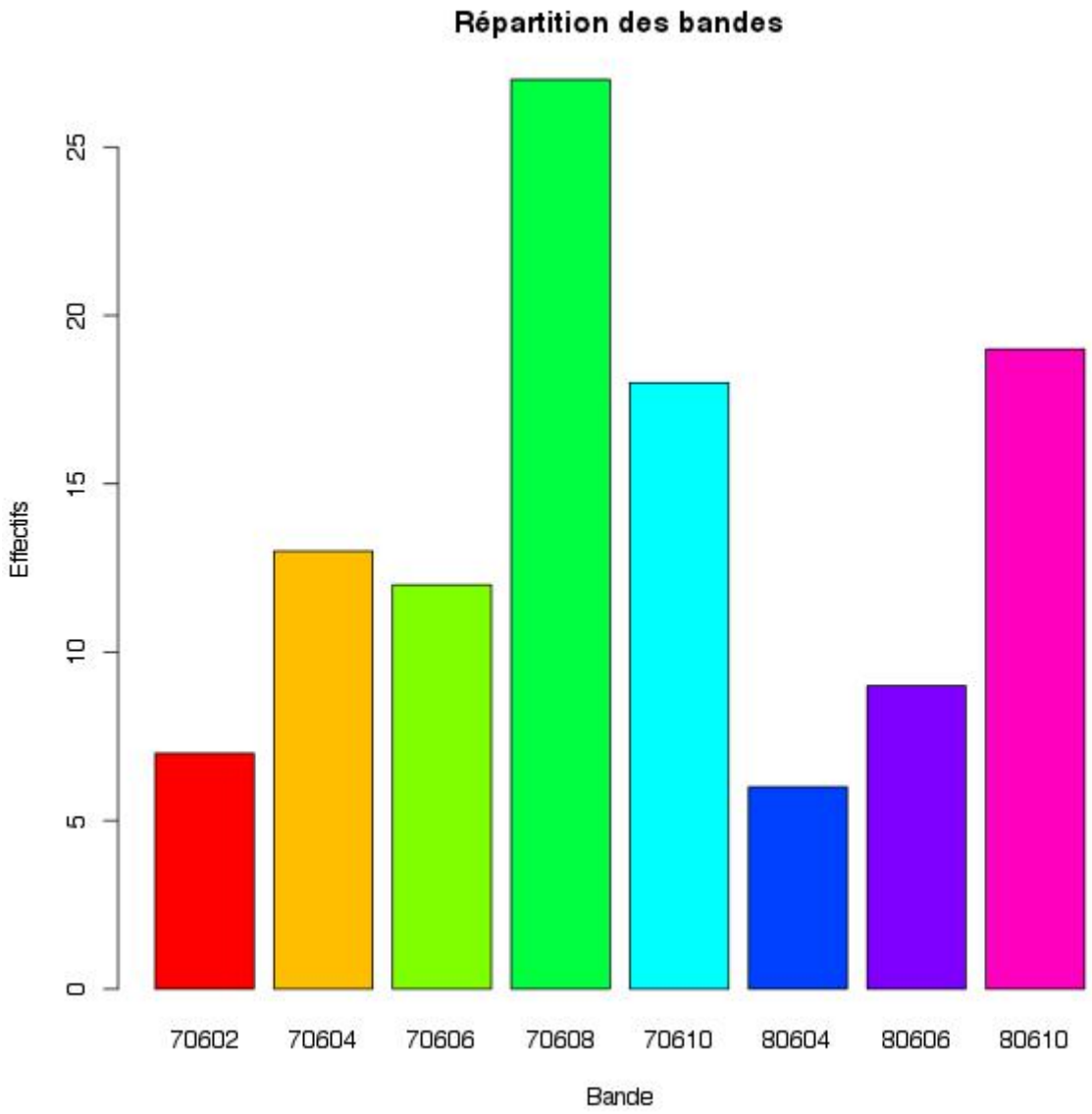


Figure 4 : Distribution des diverses bandes

Tout comme pour la race, il y a une grande hétérogénéité dans les effectifs des bandes. En effet, une majorité des cochons proviennent de la bande 70608, les bandes 70610 et 80610 contiennent également un nombre de cochons assez importants (une vingtaine), alors que certaines bandes contiennent moins de 10 cochons : les bandes 70602, 80604 et 80606. Une telle hétérogénéité montre également que les données ne sont pas parfaites, l'homogénéité n'étant pas bien respectée, et qu'il est nécessaire de prendre ces disparités en compte lors de l'étude statistique.

L'impact de la bande et de la race peut être bien plus important s'il se trouve que ces deux variables sont liées car on peut avoir du mal à différencier si c'est la race ou la bande qui influence l'expression d'un gène donné. Le Tableau 7 et la Tableau 5 donnent la table de contingence croisement entre race et bande.

	70602	70604	70606	70608	70610	80604	80606	80610	Total
Dchn	0	0	0	0	0	0	0	3	3
Duroc	0	4	0	0	4	0	0	16	24
Landrace	1	3	4	12	0	6	7	0	33
LWF	0	6	8	15	10	0	2	0	41
LWM	6	0	0	0	4	0	0	0	10
Total	7	13	12	27	18	6	9	19	111

Tableau 7: Table de contingence entre « race » et « bande »

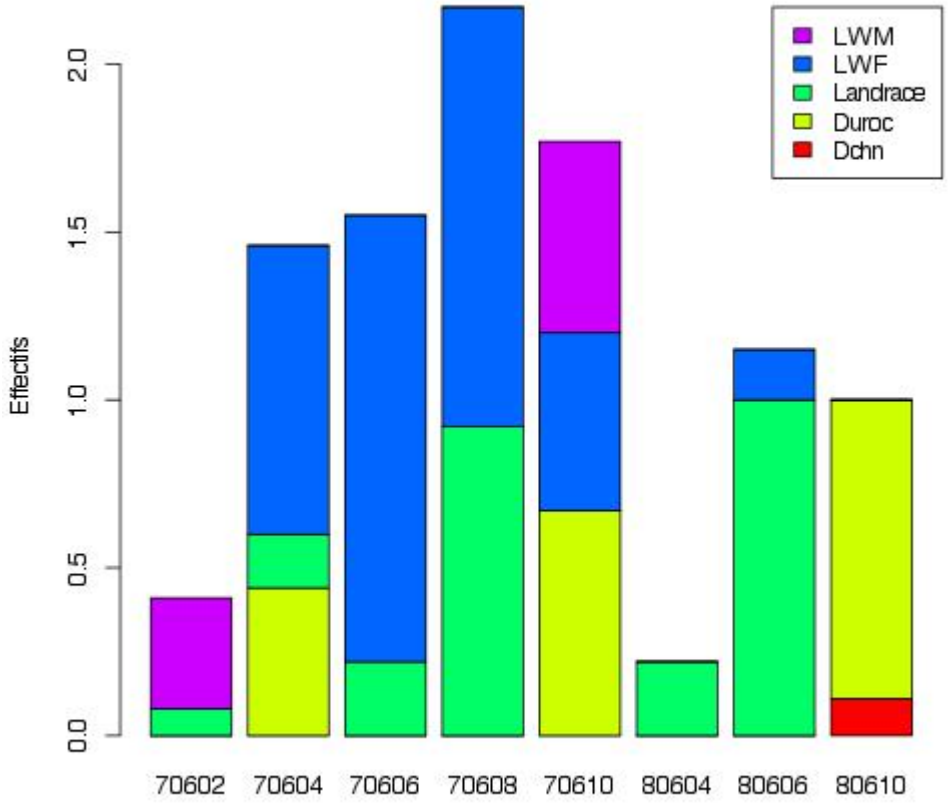


Figure 5 : Distribution conjointe entre race et bande

On remarque des phénomènes semblant montrer un certain lien entre la bande et la race. En effet, il existe des bandes ne contenant presque exclusivement qu'une race, comme la bande 80604, qui, en plus d'être de petite taille, ne contient presque que des cochons de race Landrace. Un autre phénomène notable est le fait qu'une race, Dchn, n'est présente que dans une seule bande : la bande 80610, on n'a donc aucune information sur l'impact environnemental pour cette race. Le lien supposé semble se confirmer avec le test du χ^2 de l'hypothèse

H_0 : « Les variables **race** et **bande** sont indépendantes »

La p-value est égale à $2{,}2\times 10^{-16}$, montrant un lien plutôt fort entre la race et la bande. Ce résultat est discutable, étant donné que les conditions d'effectifs théoriques minimum par case (de Cochran) ne sont pas remplies. Le chiffre peut donc également s'expliquer par ces conditions non remplies et non par un lien réel. Cependant, les résultats obtenus par le tableau de contingence montrent encore une fois que les données ne

sont pas idéales statistiquement. Ainsi pour certaines analyses plus poussées, un traitement supplémentaire sur les données pourrait être nécessaire pour corriger l'effet de la corrélation entre bande et race.

3) Étude du phénotype d'intérêt

Le but de cette étude est également de connaître les gènes agissant sur la longueur de la carcasse du cochon. C'est donc cette dernière variable qui est appelée « phénotype d'intérêt ». Celle-ci est obtenue à partir de deux mesures (lgr.carc, la longueur de la carcasse et lgr.carc.r cette longueur corrigée de l'effet bande sur un échantillon plus grand d'animaux), toutes deux en relation avec le poids de la carcasse. Nous ne connaissons pas le protocole expérimental précis de la mesure de ces deux variables, donc de leur différence quant à la mesure du poids de la carcasse.
Le Tableau 8 donne quelques mesures de statistique descriptive pour le premier phénotype d'intérêt (lgr.carc) dont la distribution est représentée en Figure 6.

Minimum	1er Quart	Médiane	Moyenne	3eme Quart	Maximum	Skewness	Kurtosis
950	990	1010	1013	1030	1100	0.475	-0.156

Tableau 8: Statistiques élémentaires sur la longeur de la carcasse (lgr.carc)

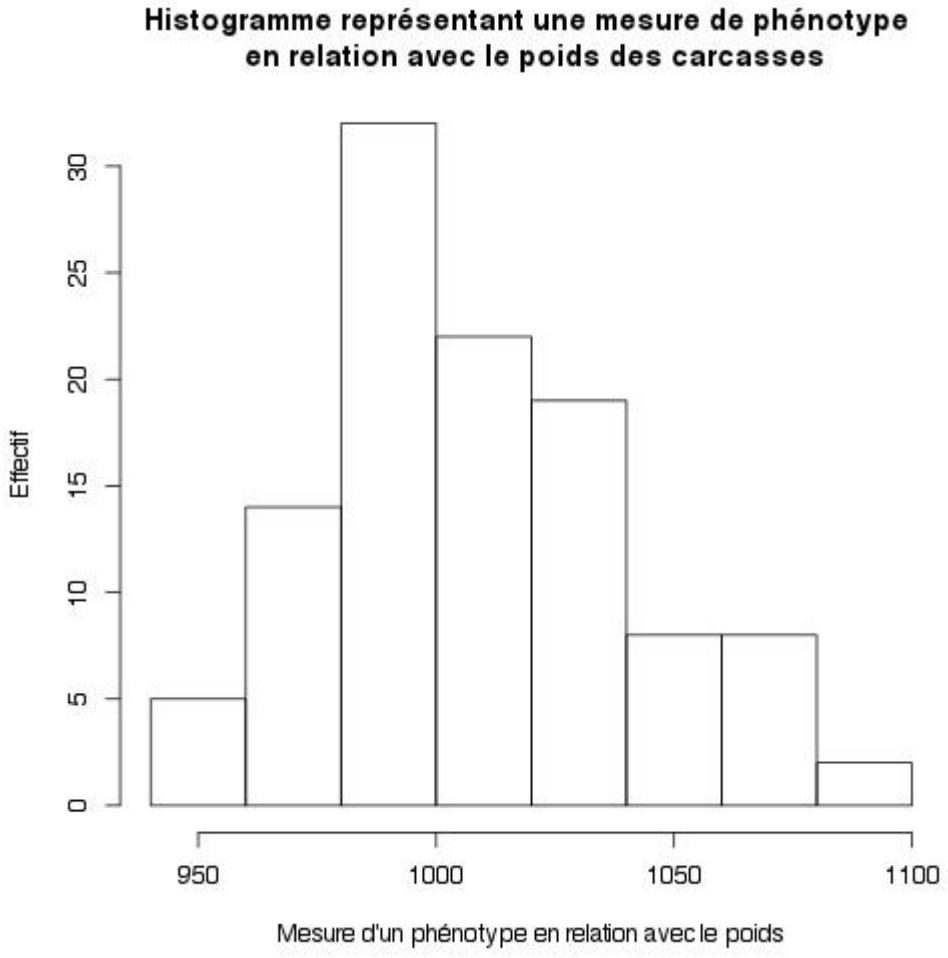


Figure 6 : Distribution de la longueur de carcasse (lgr.carc)

Cette mesure du phénotype montre des résultats variants entre 950 et 1100, plutôt symétrique, et un pic entre 980 et 1000.
Le Tableau 9 donne quelques mesures de statistique descriptive pour l'autre mesure du phénotype d'intérêt (lgr.carc.r) dont la distribution est représentée en Figure 7.

Minimum	1er Quart	Médiane	Moyenne	3eme Quart	Maximum	Skewness	Kurtosis
-60.510	-19.980	-1.738	1.582	22.360	72.410	0.130	-0.554

Tableau 9: Statistiques élémentaires sur la longueur de la carcasse (lgr.carc.r)

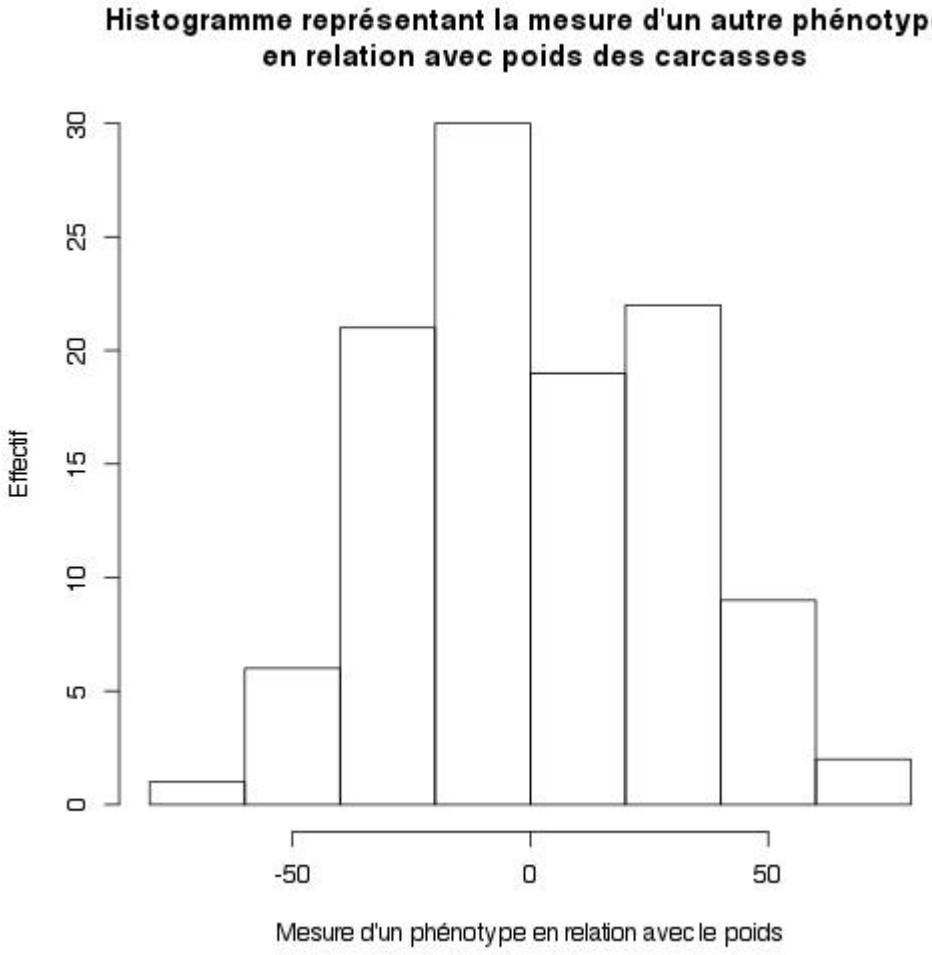


Figure 7 : Distribution de la longueur de carcasse (lgr.carc.r)

Cette mesure du phénotype montre des résultats variants entre -60,510 et 72,410, avec une distribution relativement symétrique, et un pic entre -20 et 0, et un autre, moins marqué, entre 20 et 40.

Ces deux mesures du phénotype montrent des différences, surtout au niveau des ordres de grandeurs. Mais peut-on considérer que l'une dépend directement de l'autre, et que ça n'est qu'une mesure "normalisée" ? Pour cela, considérons un test de corrélation linéaire entre ces deux mesures de phénotypes.
Il est attendu que ces deux mesures soient corrélées, elles sont en effet toutes deux en relation avec le poids, et mesurées sur le même échantillon ; pour être plus précis, lgr.carc.r est calculée à partir de lgr.carc par une « normalisation » effectuée à l'aide d'un échantillon plus grand d'animaux et qui avait pour but de corriger les effets de « bandes » (et autres effets environnementaux). Le test de Pearson permet de tester l'hypothèse

H0 : « Cor(lgr.car;r,lgr.car) = 0 »

où Cor désigne le coefficient de corrélation linéaire. La p-value obtenue est égale à $2{,}2\times10^{16}$, et on conclut donc qu'il existe une corrélation significative entre les deux mesures de phénotype. D'ailleurs, le coefficient de corrélation linéaire entre les deux variables a une valeur égale à 0,779, qui est effectivement le signe d'une corrélation assez forte même si celles-ci contiennent des informations légèrement différentes.
Voici ci-dessous le nuage de points de ces deux mesures, illustrant cette conclusion.

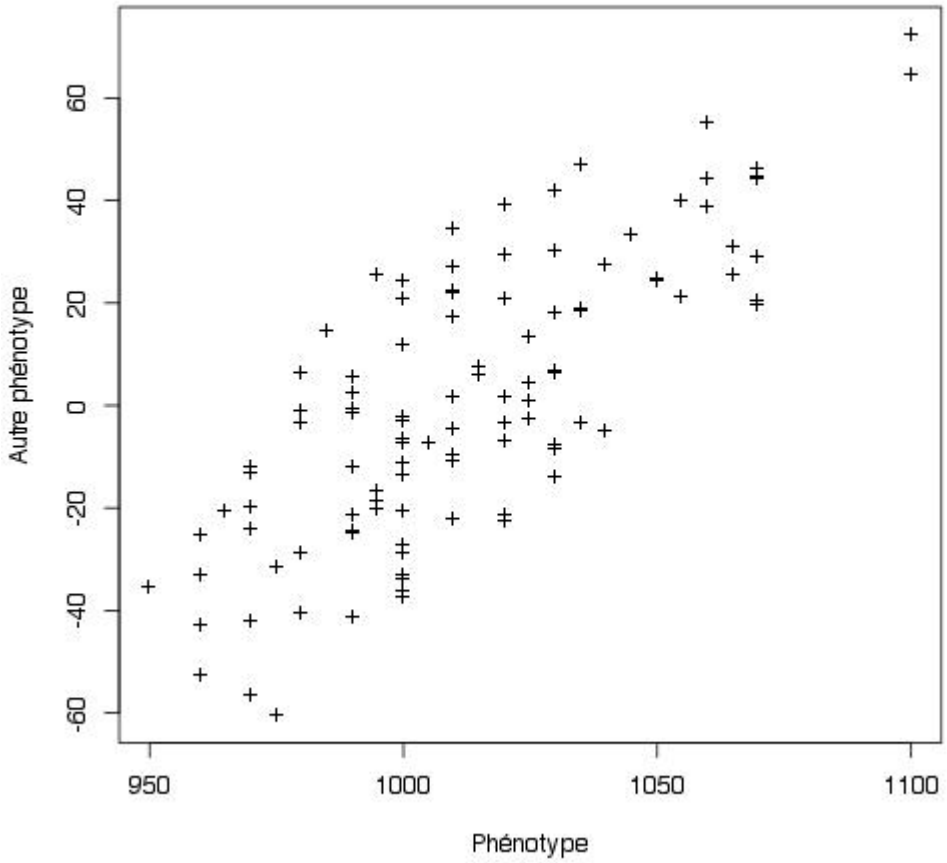


Figure 8: Nuage de points des deux mesures du phénotype

Ce nuage est en forme de fuseau et présente un bon ajustement linéaire même si celui-ci n'est pas parfait.
Pour les prochaines analyses, on considérera les deux mesures du phénotypes séparément, quand cela sera possible, puisque leur signification n'est pas strictement identique.

Il est attendu que ce phénotype soit influencé par la race du cochon. Mais l'est-il également par la bande ? Un test de normalité par race et par bande, préalable à une ANOVA, permettra de vérifier cela, avec, en premier lieu, l'influence de la race sur les phénotypes.
Comme vu précédemment, la race Dchn pose problème. En effet, en plus de n'avoir que peu d'individus de cette race, elle n'est présente que dans une seule bande, faussant tout l'intérêt de la bande, qui est de mesurer, afin de limiter, l'impact environnemental sur le phénotype, et risque de fausser les tests. Ainsi, pour cette ANOVA, les individus de race Dchn ont été écartés.
Au préalable, il faut vérifier que, au sein d'une race, les phénotypes des individus sont distribués normalement, par le biais de tests de normalité de Shapiro-Wilk, appliqués à chaque race. Dans ce test, l'hypothèse testé, pour chacune des sous-populations définies par les animaux d'une même race, est :

H_0 : « La distribution de la longueur de carcasse est gaussienne »

Voici donc un tableau présentant les p-values associées à chacun de ces tests.

Race	Duroc	Landrace	LWF	LWM
Valeur de la statistique de test	0.9655	0.9582	0.9605	0.884
Valeur de la p-value	0.5823	0.2294	0.1643	0.1451

Dans tous les cas, la valeur des p-values associées à ces tests est supérieure aux seuils de 5%, ainsi, on ne peut pas rejeter l'hypothèse de normalité au niveau 5% pour chacune de ces races. On admettra donc que le phénotype d'intérêt est distribué normalement au sein de chacune des races étudiées. Grâce à cette hypothèse, on peut effectuer le test de Bartlett d'homogénéité des variances du phénotype par race. Ce test permet de tester l'hypothèse

H_0 : « Les variances de la variable **lgr.carc** sont identiques dans les différentes races. »

contre l'hypothèse alternative

H_1 : « Pour au moins une des races, la variance de la variable **lgr.carc** est différente des variances pour les autres races. »

La statistique de test est égale à 4,2607, avec une p-value associée de 0,2346, supérieure au seuil de 5%. On ne peut donc pas rejeter l'hypothèse d'homogénéité des variances du phénotype entre les races au niveau 5%.

Les hypothèses de normalité et d'homogénéité des variances entre les races étant admises, on peut effectuer l'ANOVA. Ce test permet de tester l'hypothèse

H_0 : « Les moyennes de la variable **lgr.carc** sont identiques dans les différentes races. »

contre l'hypothèse alternative

H_1 : « Pour au moins une des races, la moyenne de la variable **lgr.carc** est différente des moyennes pour les autres races. »

La p-value obtenue est environ égale à 10^{-5}, de ce fait, on rejettera l'hypothèse d'égalité des longueurs de carcasse moyennes entre races. La race influe donc significativement sur la longueur de la carcasse. Ce résultat, bien qu'attendu, est illustré par les boîtes à moustaches de ce phénotype par races.

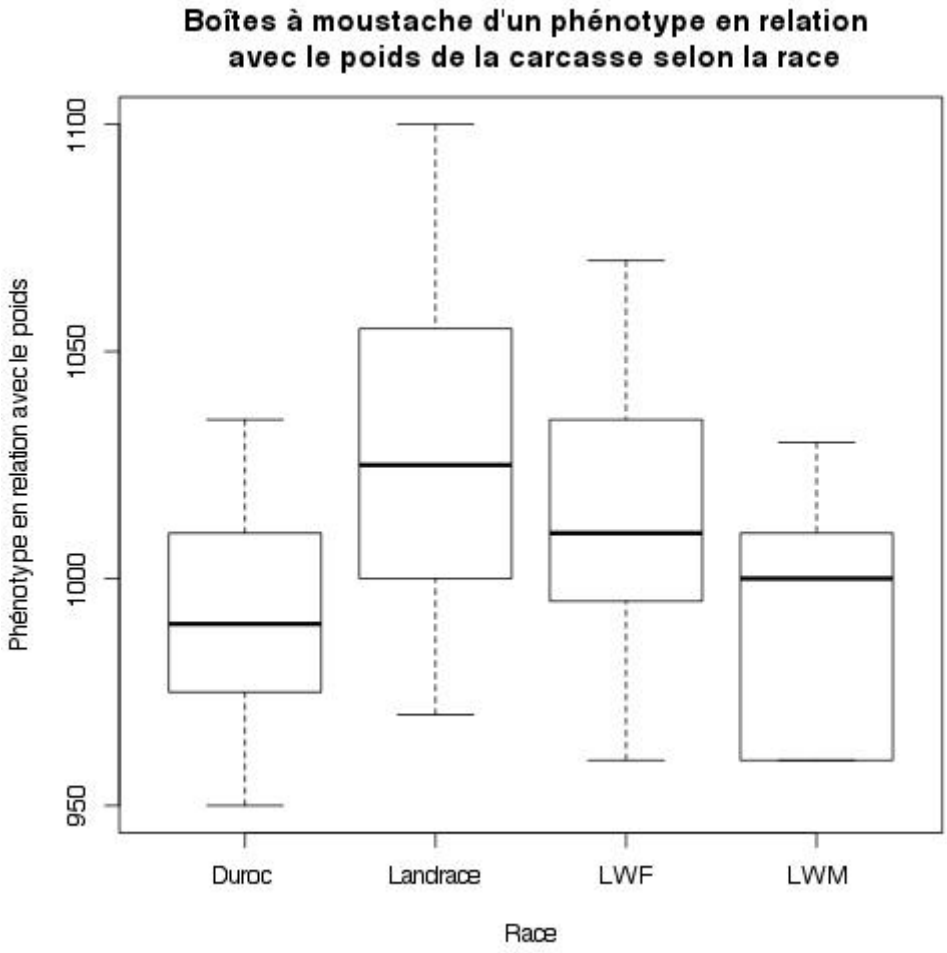


Figure 9: Distribution de la longueur de carcasse corrigée selon la race

La même démarche devra être effectuée pour l'autre mesure du phénotype. Voici donc, comme précédemment, un tableau présentant les résultats du test de Shapiro-Wilk relatif à la deuxième mesure du phénotype d'intérêt pour chaque race (lgr.carc.r).

Race	Duroc	Landrace	LWF	LWM
Valeur de la statistique de test	0.9749	0.9684	0.9767	0.795
Valeur de la p-value	0.8049	0.438	0.5528	0.01259

Ici, cependant, l'hypothèse de normalité est rejetée pour les individus de race LWM, l'ANOVA n'est donc pas valable comme test pour vérifier l'égalité des valeurs moyennes de ce phénotype entre races. Cependant, une visualisation graphique, présentant les boîtes à moustache de ce phénotype selon les races, permet d'avoir une idée intuitive de l'influence de la race sur ce phénotype d'intérêt.

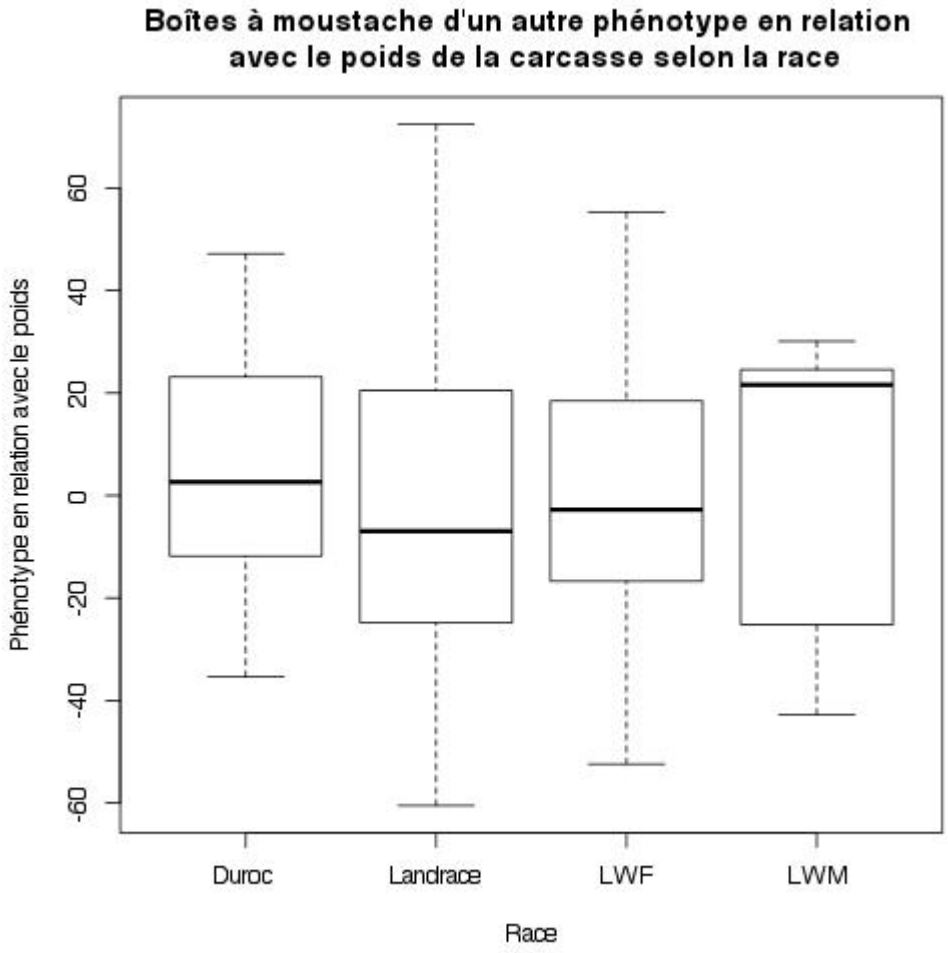


Figure 10: Distribution d'une autre mesure du phénotype selon la race

Si l'on ne tient pas compte de la distribution très dissymétrique de ce phénotype pour la race LWM, la race semble avoir moins d'influence sur la moyenne de ce phénotype en comparaison avec l'autre.

Si l'influence de la race sur la longueur de carcasse est attendue, qu'en est-il de l'influence de la bande sur ce phénotype ? Si la bande influence la longueur de la carcasse, cela montrerait l'importance des facteurs environnementaux sur le phénotype, mais serait assez problématique pour l'étude qui devra, pour continuer, tenir compte de ce phénomène, vérifiable par ANOVA.

Comme pour les précédents tests effectués, des tests de Shapiro-Wilk devront être effectués pour chaque bande, afin de mesurer la normalité des phénotypes dans chaque sous-population. Voici donc un tableau présentant les p-values associés à chacun de ces tests pour le premier phénotype d'intérêt (lgr.carc).

Bande	70602	70604	70606	70608	70610	80604	80606	80610
Valeur de la statistique de test	0,9298	0,9343	0,9334	0,9419	0,9817	0,8964	0,923	0,9709
Valeur de la p-value	0,5493	0,3873	0,417	0,1358	0,9658	0,3529	0,4178	0,8706

L'hypothèse de normalité n'est rejetée pour aucune des bandes, on admettra donc la distribution normale de ce phénotype au sein de chaque bande. Le test de Bartlett d'homogénéité des variances peut donc être effectué. Celui-ci donnant une statistique de test de 8,7491, avec une p-value associée de 0,2712, on ne trouve pas de différence significative de la variance de la mesure du phénotype entre les bandes. La p-value associée au test d'ANOVA est de 0,673. De ce fait, on ne peut pas rejeter l'hypothèse d'égalité des variances de la mesure de ce phénotype entre les races. La bande ne semble donc pas influencer significativement le phénotype, même si les boîtes à moustaches de cette mesure du phénotype par race présentées ci-dessous semblent illustrer le phénomène inverse.

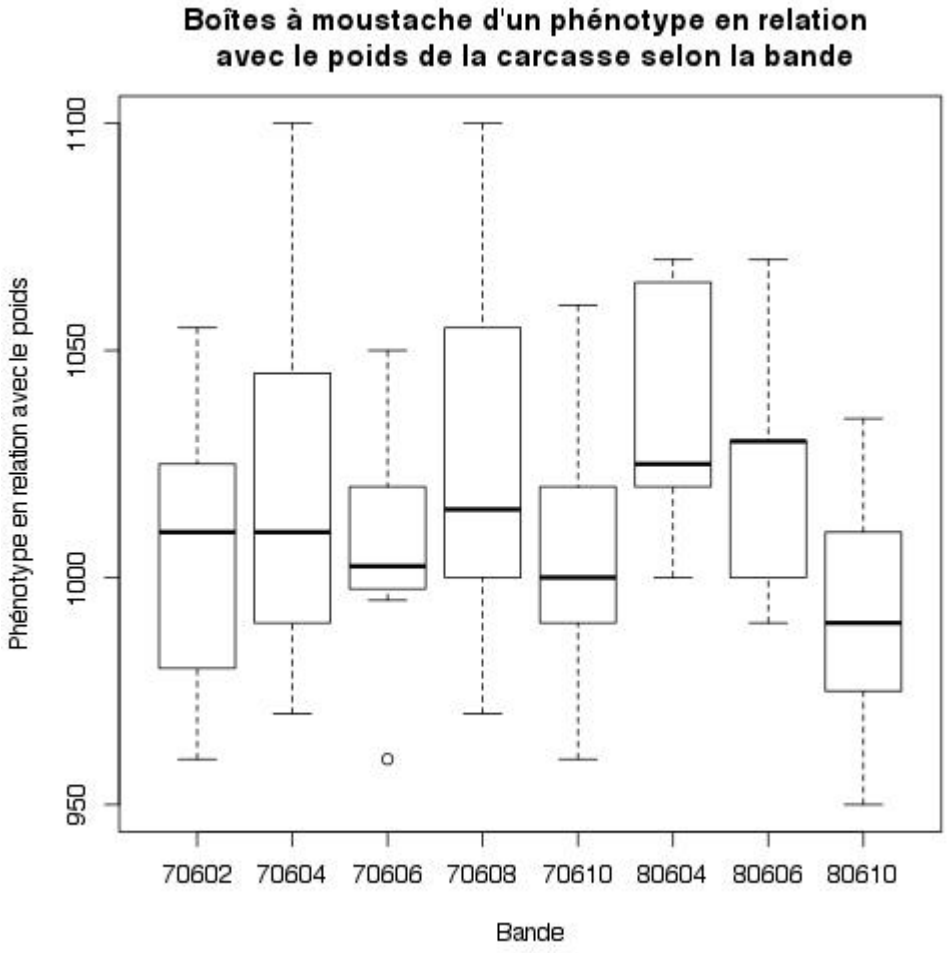


Figure 11: Distributions d'une mesure du phénotype selon la race

La même démarche devra être effectuée pour l'autre mesure du phénotype. Voici donc, comme précédemment, un tableau présentant les résultats du test de Shapiro-Wilk relatif à la deuxième mesure du phénotype d'intérêt pour chaque bande (lgr.carc.r).

Bande	70602	70604	70606	70608	70610	80604	80606	80610
Valeur de la statistique de test	0.7246	0.9379	0.9671	0.9616	0.9782	0.892	0.9215	0.4049
Valeur de la p-value	0.006834	0.4304	0.8781	0.4017	0.9301	0.3291	0.4049	0.8617

La normalité n'est pas vérifiée pour la bande 70602. Ainsi, on ne peut pas continuer l'ANOVA sans risque de biaiser ce test. Cependant, une visualisation graphique, présentant les boîtes à moustaches de ce phénotype selon les races, permet d'avoir une idée de l'influence de la race sur ce phénotype d'intérêt.

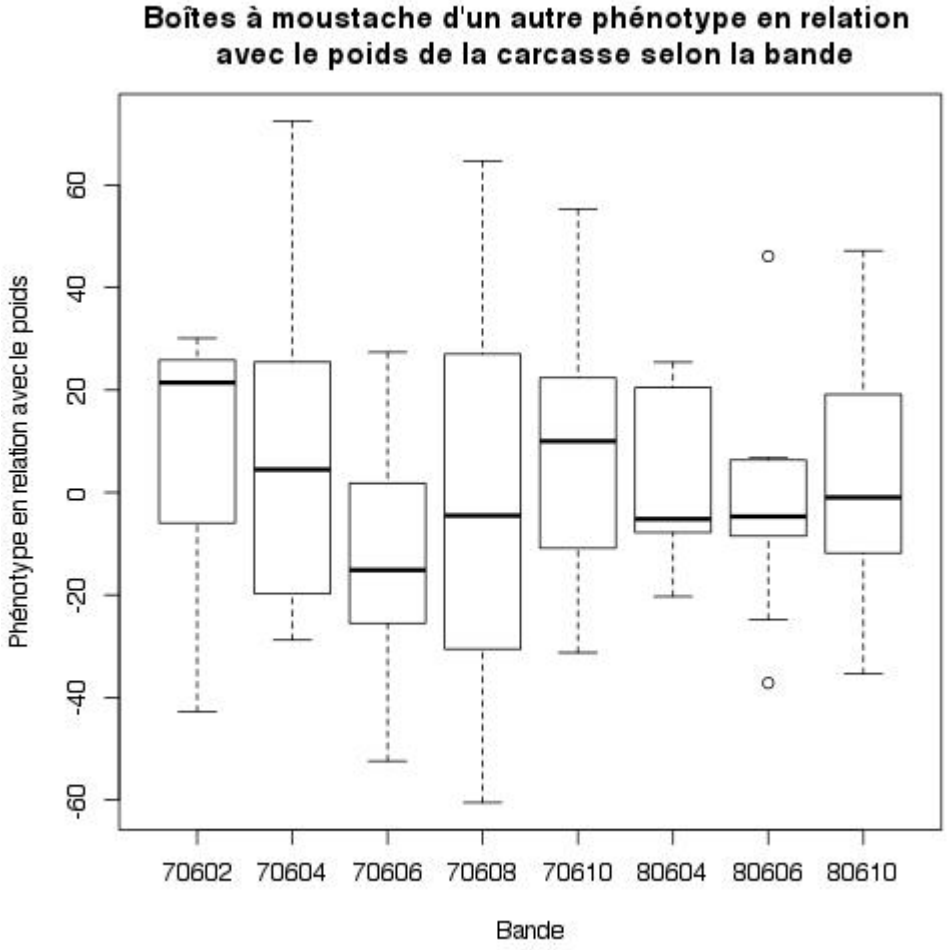


Figure 12: Distributions d'une autre mesure du phénotype selon la race

L'effet de la bande semble être encore moindre sur cette mesure du phénotype. Mais comme vu précédemment, cette visualisation peut être trompeuse, et l'on ne peut pas conclure précisément du fait de la distribution non normale de la bande 70602.

4) Analyse multivariée des expressions

La variabilité de l'expression des gènes contient une information importante: les gènes dont l'expression est peu variable d'un individu à l'autre pourraient être des gènes de fonctionnement commun à l'ensemble de l'espèce (le cochon) voire, au delà, des gènes de fonctionnement commun

aux mammifères ou au règne animal dans son ensemble. À l'inverse, les gènes dont l'expression est fortement variable peuvent être spécifique de l'individu ou, de manière plus intéressante pour le biologiste, peuvent être de bons candidats de gènes spécifiques à la race ou en relation avec un phénotype d'intérêt. Par ailleurs, dans une dimension de biologie intégrative, on peut aussi s'intéresser aux relations entre l'expression de l'ensemble des gènes : la réalisation d'une « heatmap », croisant une classification des expressions des gènes par similarité sur l'ensemble des individus et les individus, que nous avons ordonné par race, permettra de voir si il existe des groupes de gènes facilement identifiables ont l'expression est spécifique de la race, ou éventuellement repérer certains individus ou certains groupes de gènes ayant un comportement particulier.

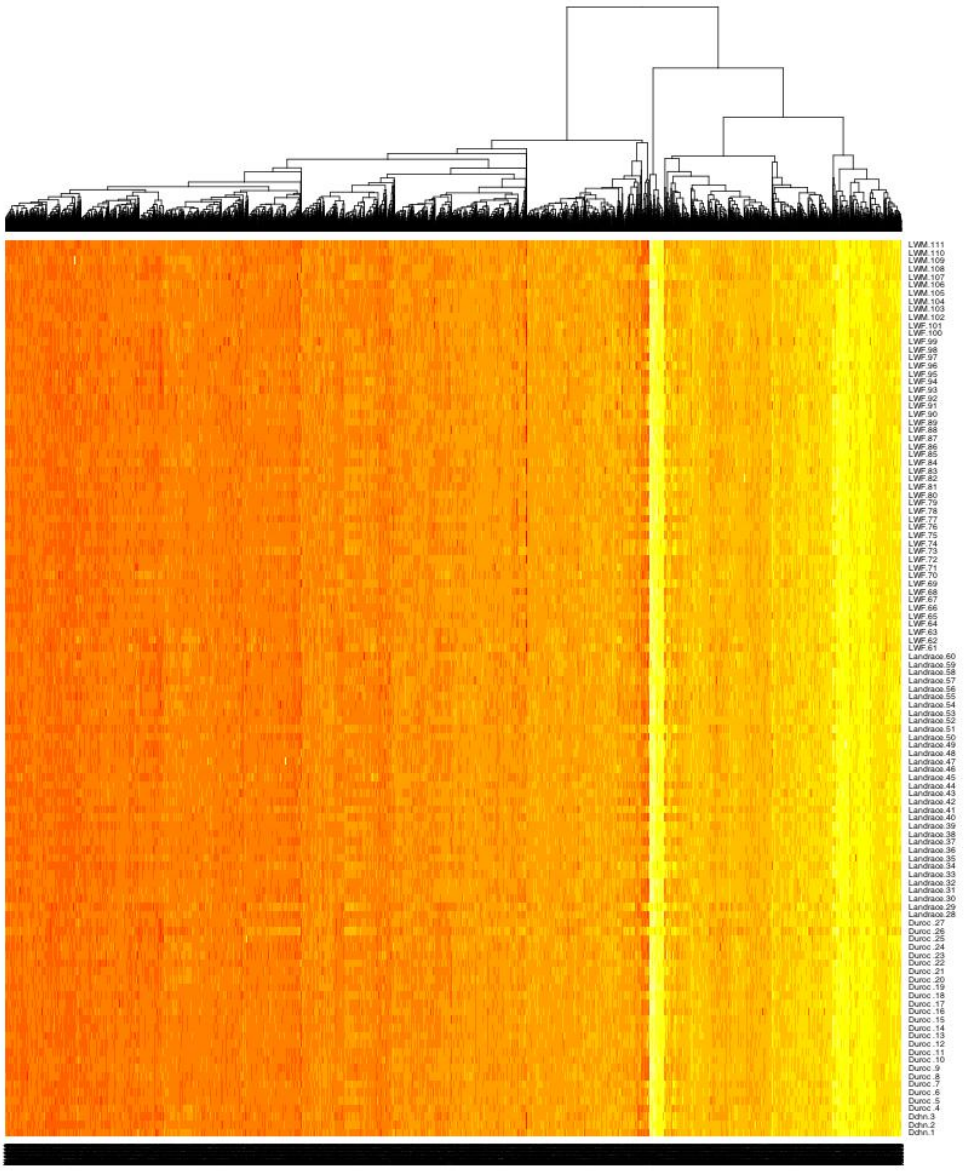


Figure 13: « Heatmap » croisant une classification des gènes et les individus triés par race : les gènes sont ordonnées d'après les résultats d'une classification hiérarchique ascendantes, les individus sont ordonnés par race et les couleurs montrent l'expression d'un gène donné pour un individu donné (rouge : gène fortement exprimé ; jaune : gène faiblement exprimé)

On remarque une évolution progressive de l'expression des gènes : les gènes apparaissent être de moins en moins exprimés de gauche à droite de la classification. La tendance de l'expression des gènes semble en tout cas se faire indifféremment des individus, et même des races. Ainsi, il est difficile de repérer simplement des groupes de gènes dont l'expression est spécifique d'une race.

Afin d'approfondir les relations multiples de l' expression des gènes et de les mettre en relation avec des facteurs extérieurs (bande, race, phénotype), on réalise une ACP à partir des expressions des gènes, afin de trouver des groupes d'individus semblables vis-à-vis de leur génome. La représentation des individus sur les premiers axes de l'ACP sera ensuite interprétée en relation avec leurs races, leurs bandes, ou la valeurs du phénotype étudié. Vu le nombre de gènes (donc de variables incluses dans l'ACP), on ne peut représenter de cercle de corrélations, on se limitera au nuage de points des individus et d'un cercle des corrélations sur lequel seules les variables les plus corrélées aux axes seront représentées. Voici tout d'abord l'éboulis des valeurs propres de cette ACP, et le tableau des valeurs propres et de l'inertie pour les 20 premiers axes.

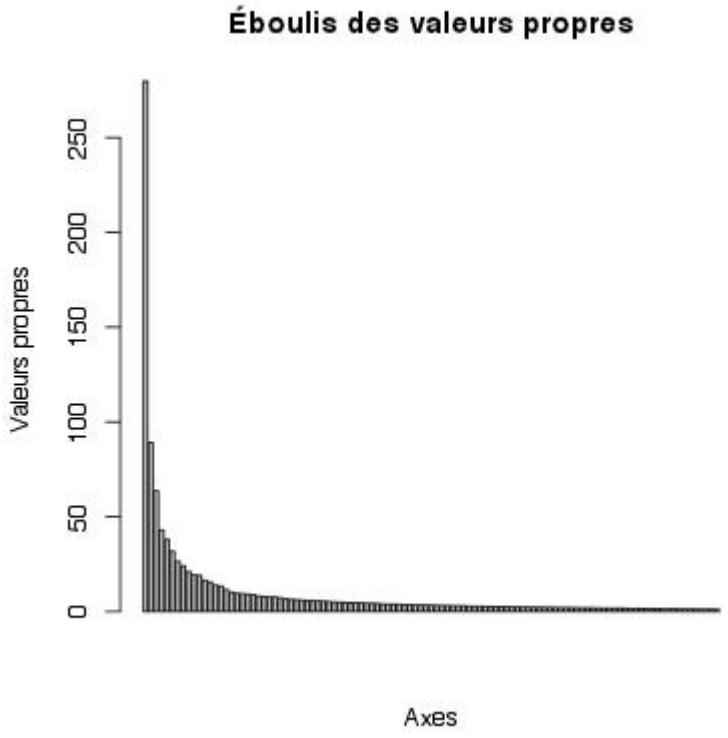


Figure 14: Éboulis des valeurs propres dans l'ACP de l'expression des gènes

Axe	Valeur propre	Inertie	Inertie cumulée
1	279,77	26,53	26,53
2	89,09	8,45	34,98
3	63,65	6,04	41,02
4	42,97	4,08	45,1
5	38,04	3,61	48,7
6	31,78	3,01	51,72
7	26,44	2,51	54,23
8	24,03	2,28	56,5
9	21,06	2	58,5
10	19,4	1,84	60,34
11	19,03	1,8	62,15
12	16,54	1,57	63,71
13	15,51	1,47	65,19
14	13,87	1,32	66,5
15	13,39	1,27	67,77
16	11,6	1,1	68,87
17	10,15	0,96	69,83
18	9,61	0,91	70,75
19	9,4	0,89	71,64
20	9,22	0,87	72,51

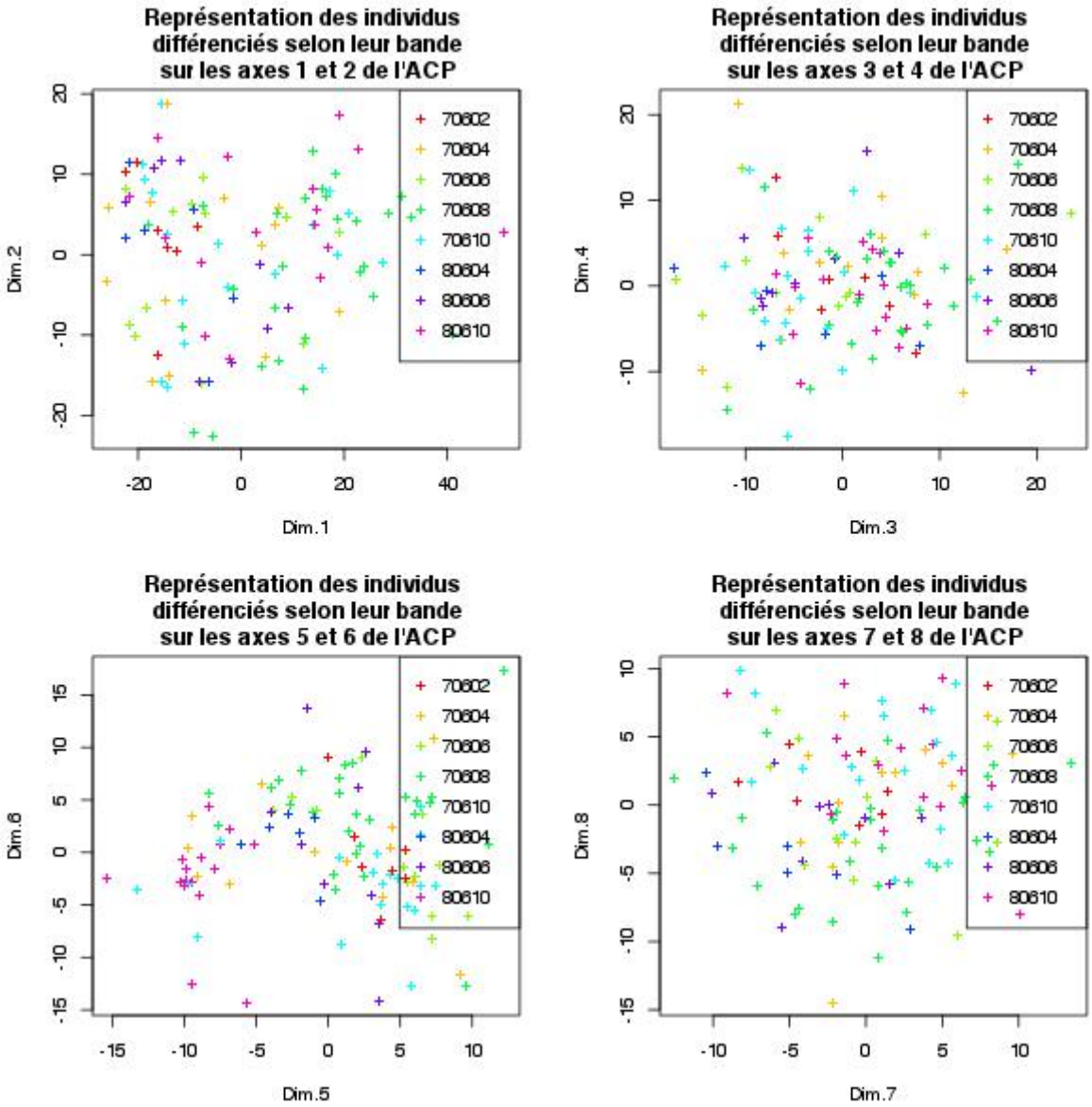
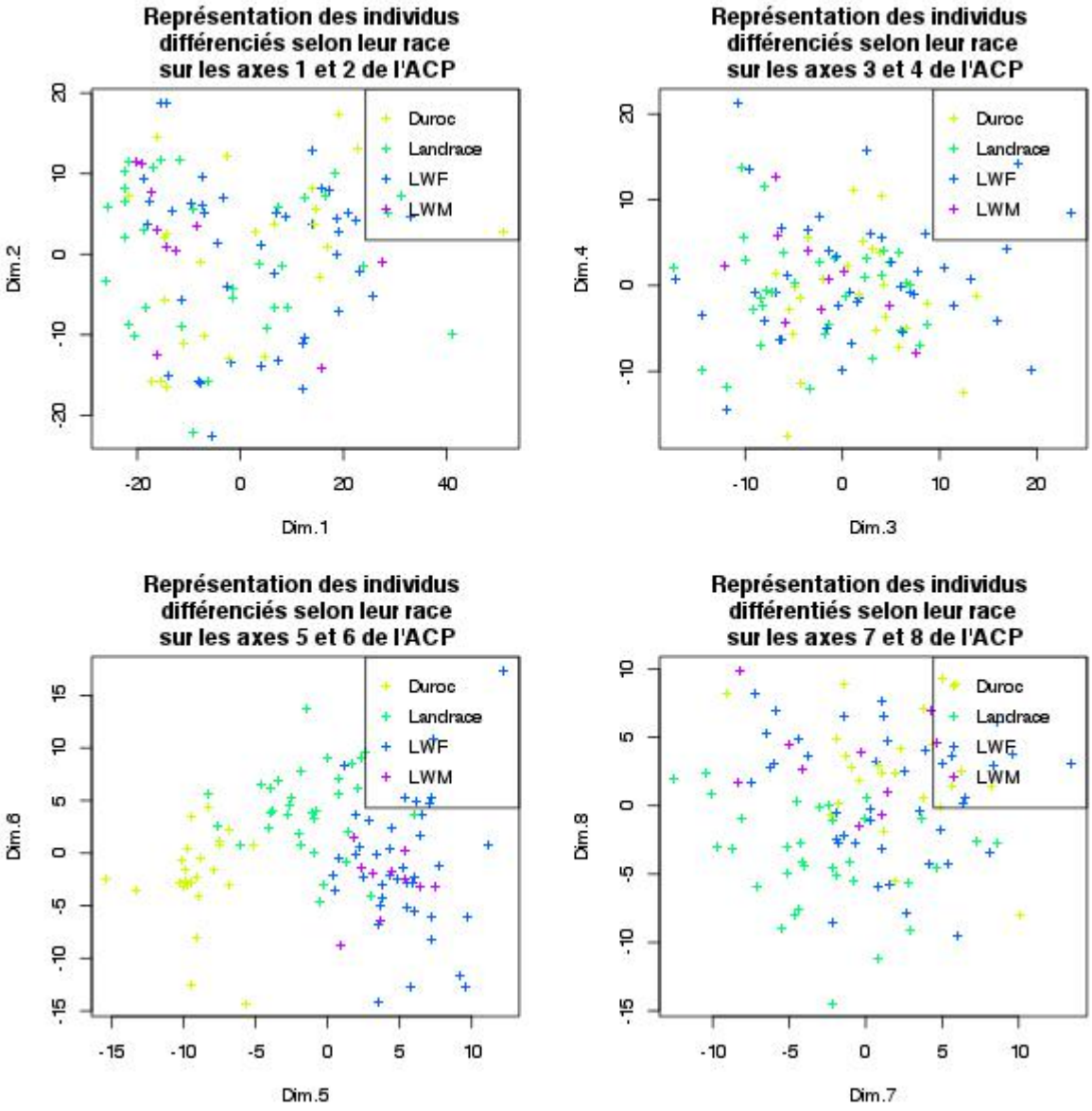


Figure 15: Représentation des individus différenciés selon leur bande sur les axes 1 à 8 de l'ACP

Vu le nombre d'axes (107), on ne peut utiliser les critères de Kaiser ou de l'éboulis, au risque de perdre énormément d'information (le critère de l'éboulis conduirait à choisir 2 axes à interpréter, ce qui, vu le nombre d'axes, est très nettement insuffisant ou à choisir trop d'axes à interpréter). Un compromis est trouvé en représentant les 8 premiers axes (cela permet de dépasser 50% d'inertie et correspond également à un léger coude, même si il n'est pas visible sur le scree-graph ci-dessus). Voici, en annexe, les coordonnées, les contributions et les cosinus carrés des 20 gènes les plus reconstitués sur les axes 1 et 2 (dont la somme des cosinus carrés des axes 1 et 2 sont les plus importantes), ainsi que les nuages de points des individus colorés selon certaines autres variables. Voici donc le nuage de points des individus de cette ACP pour les 8 premiers axes, colorés par bande.
L'ACP ne semble pas regrouper significativement les individus appartenant à la même bande. Dans ce jeu de données, la bande, donc le facteur environnemental ne semble avoir que peu d'impact sur le génome, ce qui est le résultat attendu puisque la bande est un facteur de nuisance

environnementale . On note tout de même une exception sur l'axe 5, où les individus appartenant à la bande 80610 sont regroupés dans les valeurs négatives sur l'axe 5 et ont une position centrale sur l'axe 6. Cela peut être expliqué par le fait que, après traitement, cette bande ne contient que des individus d'une seule race : Duroc. Voici, pour vérifier cela, le même nuage de points coloré par race.



Sur la plupart des axes, les races ne sont que peu ou pas regroupées entre elles. Cependant, l'axe 5 montre clairement un regroupement des individus de même race sur le nuage, ainsi que, dans une moindre mesure, l'axe 8. Les individus de race Duroc se placent de la même façon que la bande 80610, ce qui confirme l'hypothèse de regroupement de cette bande sur la coloration précédente. On remarque, par ce regroupement, que les Duroc semblent éloignés de la race LWF du point de vue du génome, alors que les deux races Large White se ressemblent à ce niveau.
Si l'axes 5 et 8 regroupent les races, l'étude concerne surtout le phénotype d'intérêt, peut-être celui ci apparaît-il sur cette ACP. Voyons pour cela ces mêmes représentations des individus sur l'ACP colorés selon la valeur de leur phénotype d'intérêt (lgr.carc), regroupé en classes (les classes sont définies selon les déciles de la distribution du phénotype).

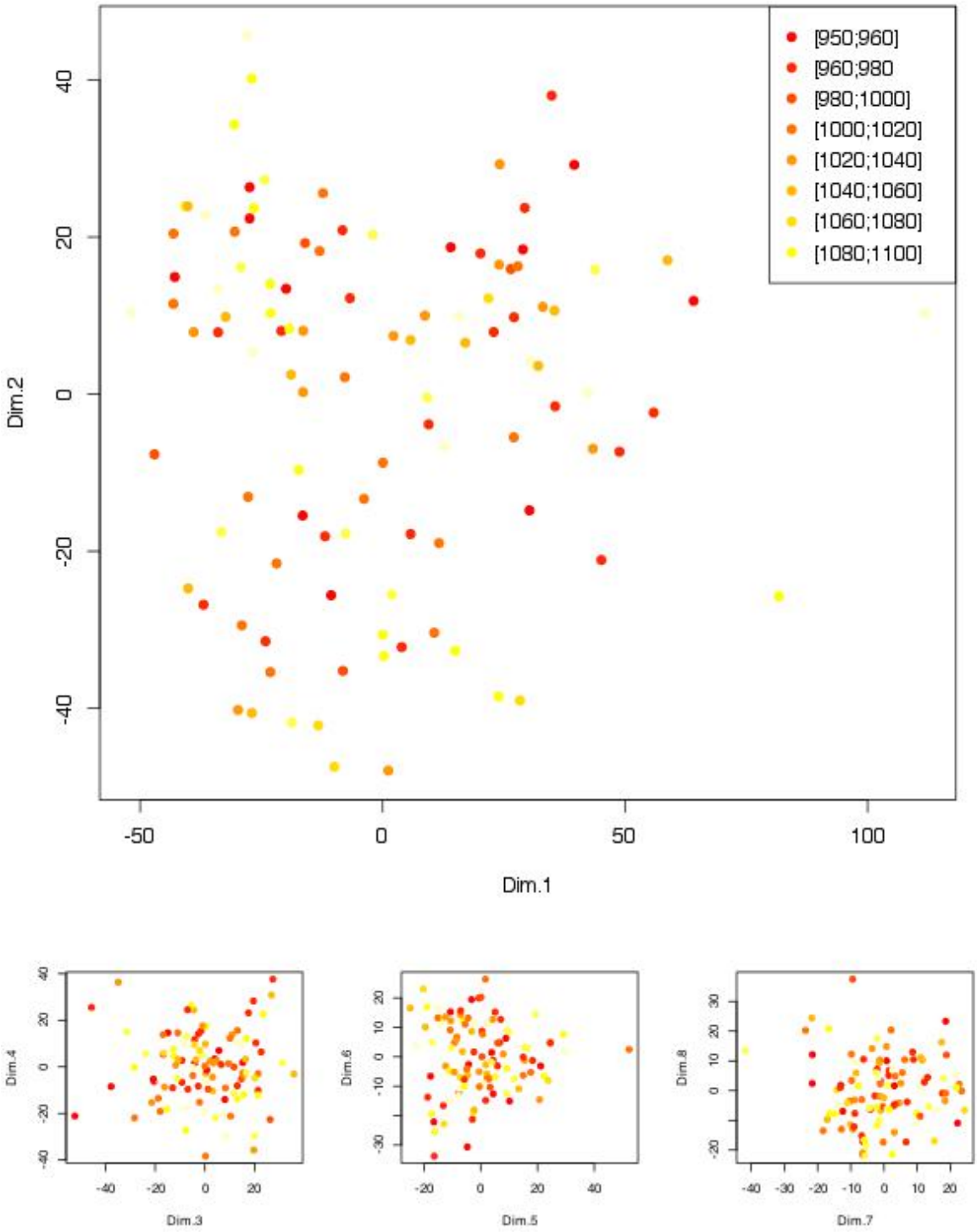


Figure 17: Représentation des individus selon la mesure du phénotype d'intérêt sur les axes 1 à 8 de l'ACP

On ne peut pas, à partir de cette ACP, regrouper les individus de valeur similaire du phénotype d'intérêt, en effet, aucune tendance précise ne se dessine. On ne peut donc trouver avec cette méthode un type de génome plus ou moins propice à une valeur élevée du phénotype. Cependant, l'ACP apporte tout de même une information sur les liens entre les gènes. En effet, cette analyse multivariée a pu regrouper certains gènes dont l'expression est corrélée. Pour vérifier cela, voici le cercle des corrélations des 8 premiers axes, pour les 20 gènes dont la somme des cosinus carré des

deux axes représentés sont les plus important.

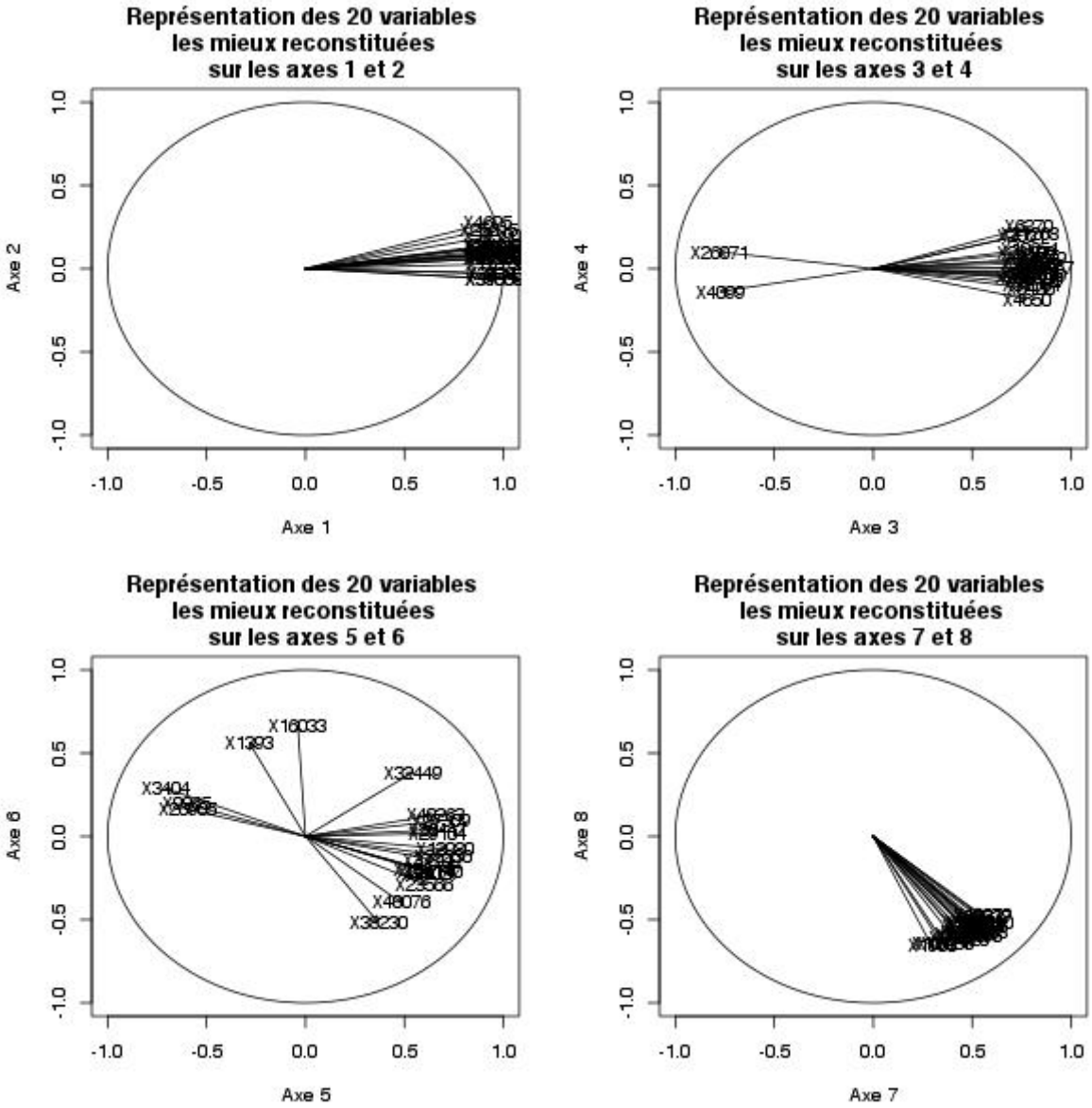


Figure 18: Cercles des corrélations pour les variables les mieux reconstituées sur les axes 1 à 8

On remarquera que les gènes dont les expressions sont les mieux reconstituées sont toutes corrélées positivement sur l'axe 1, qui semble être un axe d'échelle. Sur l'axe 3, les expressions des gènes les mieux reconstituées sur les axes 3 et 4 sont aussi corrélées, mais cet axe semble opposer les expressions des gènes X4099 et 26071 et les autres gènes les mieux reconstitués sur ces axes. Les axes 5 et 6 montrent moins de corrélations significatives. L'axe 5, le plus important du point de vue du classement des individus selon leur race, semble être un axe d'opposition entre des gènes tels que X3404, et des gènes tels que X46076, gènes semblant être influencés par la race du cochon. Les axes 7 et 8 semblent être tous deux des axes d'échelle, montrant de fortes corrélations entre les expressions des gènes les plus reconstituées sur ces axes. Un autre phénomène notable est que les gènes affichés sur chaque cercle des corrélations sont tous différents : aucun gène contribuant fortement sur 2 axes parmi les 8 affichés ne contribue fortement sur les 6 autres.

Les méthodes d'analyse exploratoire (classification non supervisée, ACP), si elles permettent de

percevoir les problèmes liées aux données (notamment la relation entre race et phénotype), et donnent des pistes de réflexions, ne permettent pas de répondre précisément à la question posée qui est de trouver des gènes spécifiques de la race ou d'un phénotype d'intérêt. Il faudra donc envisager une autre approche.

II-Prédictions par forêts aléatoires

1) Principe

On voudrait déterminer les gènes les plus influents sur la race, ainsi que sur le phénotype. Si l'on fait de la prédiction de ces deux variables, en les expliquant par les expressions des gènes, on pourra par la suite déterminer les variables contribuant le plus à expliquer la race ou le phénotype, et ainsi, trouver les gènes les plus influents pour ces deux variables.
La technique des forêts aléatoires est une méthode de prédiction non paramétrique supportant facilement un nombre important de variables explicatives. Elle est donc très adaptée au jeu de données en présence, comportant beaucoup de variables explicatives (les expressions des gènes) sans nécessité, contrairement aux méthodes de test paramétrique (ANOVA), d'hypothèse spécifique sur la loi des expressions.
L'algorithme CART (ou arbre de régression ou de classification) est une méthode de régression ou de discrimination simple. Il s'agit de séparer en 2 groupes pour lesquels les individus sont homogènes selon la variable expliquée, selon un critère par rapport à une variable explicative, et de répéter l'opération sur les groupes les moins homogènes jusqu'à une homogénéité suffisante entre les groupes.
La forêt aléatoire est un regroupement de plusieurs arbres, volontairement perturbés, car prenant en compte seulement une partie des variables, construit à partir de sous-échantillons dits bootstrap (un tirage aléatoire avec remise dans l'échantillon étudié, d'une taille fixée) dont on fera la moyenne de leurs fonctions de régression ou de leurs prédictions. Cela donnera une fonction finale, dont on calibrera 3 hyper-paramètres : le nombre d'arbres, le nombre d'individus dans les échantillons bootstrap et le nombre de variables par division des arbres ; prédisant la variable explicative (variable numérique ou qualitative). Les erreurs de prédictions, utiles pour connaître l'efficacité de la prédiction, se calculent à partir des erreurs dites « out of bag » (OOB), c'est à dire les erreurs calculées pour chaque arbre par les observations qui n'apparaissent pas dans le sous-échantillon bootstrap utilisé pour définir l'arbre : cela évite d'avoir une erreur de prédiction trop optimiste.
Bien que l'on ne connaisse pas l'expression analytique de la fonction de régression, on est capable de déterminer les variables explicatives contribuant le plus à cette dernière, en mesurant leur importance, obtenue en intervertissant aléatoirement les valeurs prises par les individus pour la variable à mesurer, et en calculant la chute de l'erreur de prédiction du modèle appris avec la variable perturbée. Dans le cas étudié, les variables importantes détermineront donc les gènes les plus influents quant à la race, et au phénotype.
Pour plus d'informations sur les forêts aléatoires et la notion d'importance d'une variable dans celles-ci, on renvoie le lecture à l'article (Breiman, 2001)[2]. Les simulations présentées ci-dessous ont été effectuées à l'aide du package randomForest de R.

2) Prédiction de la race

On va tout d'abord tenter d'expliquer par forêt aléatoire la variable race par les variables d'expression des gènes. Pour avoir un résultat dans lequel l'effet environnemental (effet « bande ») a été supprimé, nous avons utilisé des données qui avaient été préalablement corrigé de l'effet

2 (Breiman, 2001) Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5-32.

bande par un modèle mixte (avec effet bande aléatoire). Cette partie n'étant pas étudié en deuxième année de DUT STID, les données ont été corrigées par notre tutrice.
Par ailleurs, la biostatisticienne en charge des données Magali SanCristobal, avait déjà à sa disposition une liste de gènes significativement différentiels entre les races (utilisation d'un test basé sur un modèle mixte également). Cette liste nous a été fournie et contenait plus de 700 gènes avec les p-values associées (corrigées par un méthode de correction de tests multiples, Benjamini-Hochberg, non étudiée non plus en deuxième année).
Des forêts aléatoires avec les valeurs par défaut pour les hyper-paramètres ont donné de très mauvais résultats, où plus de 60% des individus ont été mal classés. Pour avoir des prédictions un peu meilleures, on a dû pousser ces paramètres. Plusieurs modèles ont donc été testés, en changeant pour chacun un des hyper-paramètres, afin de trouver un modèle suffisamment précis. Voici ci-dessous un tableau récapitulant les modèles testés. La colonne set.seed correspond à l'implantation de la "graine aléatoire" sous R, permettant de générer un pseudo-hasard et donc de rendre les résultats reproductibles à l'identique.

set.seed	ntree	mtry	sampsize	erreur
4004	10000	40	65	68,52
3244	5000	150	60	67,04
6432324	5000	1000	40	65,9
1145	2000	70	75	63,89
1103	10000	100	75	62,4
112358	10000	150	75	61,36
1012	5000	150	75	60,19
314159	5000	300	75	54,54
654356	5000	1000	60	50,11
47399955	5000	500	75	47,77
432412	5000	1200	75	44,31
4345812	5000	1000	75	43,18
55435643	10000	1000	75	43,18
47395	5000	750	75	37,04

Tableau 10: Récapitulatifs des modèles de forêts aléatoires utilisés

On retiendra le modèle donnant l'erreur la plus faible, à savoir un modèle avec 5000 arbres, 750 variables utilisés par division d'arbres, et une taille d'échantillon bootstrap de 75. Cette forêt aléatoire donne un taux de mauvais classement de 37,04% ce qui est finalement acceptable. Voici ci-dessous une représentation graphique du taux d'individus mal prédits par la forêt aléatoire pour chaque race, en fonction du nombre d'arbres.

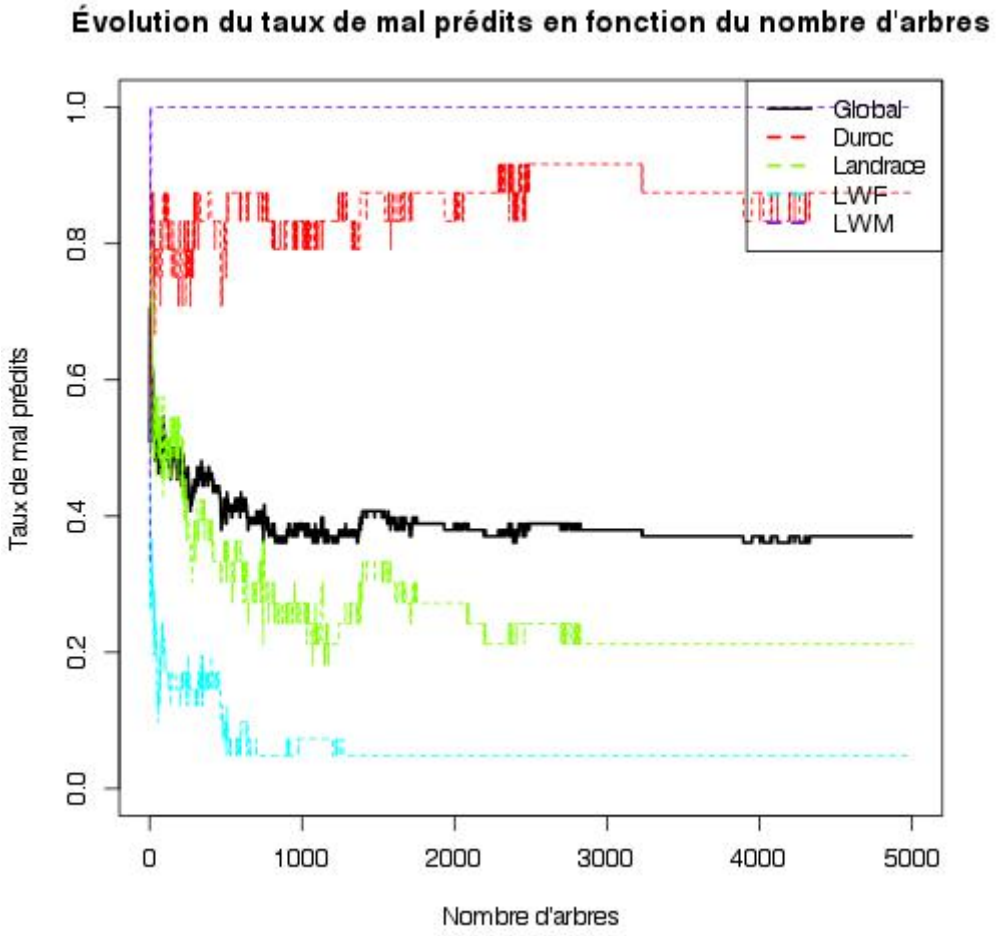


Figure 19: Evolution du taux de mauvais classement en fonction du nombre d'arbres

L'erreur s'est globalement stabilisée à partir de 3000 arbres dans la forêt. La race la mieux prédite par cette forêt est la race LWF. Les cochons de race Landrace sont également bien prédits par la forêt. Cependant, les Duroc et les LWM ont un très fort taux de mauvais classement. Le Tableau 11 ci dessous permettra de savoir dans quel sens s'effectuent les mauvais classement.

Effectif prédit / Effectif observé	Duroc	Landrace	LWF	LWM	class error
Duroc	3	10	11	0	0,88
Landrace	1	26	6	0	0,21
LWF	0	2	39	0	0,05
LWM	0	3	7	0	1

Tableau 11: Table de contingence croisant les effectifs prédits par la forêt aléatoire et les effectifs observés

On remarque que les individus de race LWM ne sont jamais prédits par la forêt aléatoire. Ils sont prédits en majorité dans la race LWF : en fait, ces deux types de cochons sont très difficiles à discriminer car ils font partie de la même race (Large White) avec des différence au niveau des critères de sélection ; comme l'effectif des LWM est très faible, ceux-ci sont systématiquement « oublié » lors de la prédiction. Il serait pertinent de réunir ces deux races pour l'apprentissage. Les

Duroc, eux, sont majoritairement prédits dans les races Landrace et LWF, sans préférence visible. Malgré les résultats de ces deux races, le modèle reste intéressant, surtout pour la prédiction des races Landrace et LWF, bien prédites par la forêt. Voici donc les 19 gènes les plus importants pour cette prédiction, au vue de 2 critères : la précision (accuracy) et le critère de Gini, ainsi qu'une représentation graphique de leur importance, correspondant à l'intersection des gènes les plus important pour les deux critères.

Nom du gène	Accuracy	Gini
X10620	0,012	1,37
X1849	0,011	1,18
X2623	0,008	1,05
X3404	0,007	0,9
X16273	0,006	0,8
X33796	0,004	0,76
X11151	0,004	0,53
X39008	0,004	0,7
X9645	0,003	0,45
X6787	0,002	0,37
X23314	0,002	0,5
X29840	0,001	0,33
X12450	0,001	0,33
X5693	0,001	0,25
X9435	0,001	0,24
X29666	0,001	0,24
X30357	0,001	0,31
X42955	0,001	0,26
X30185	0,001	0,22

Tableau 12: Gènes les plus importants

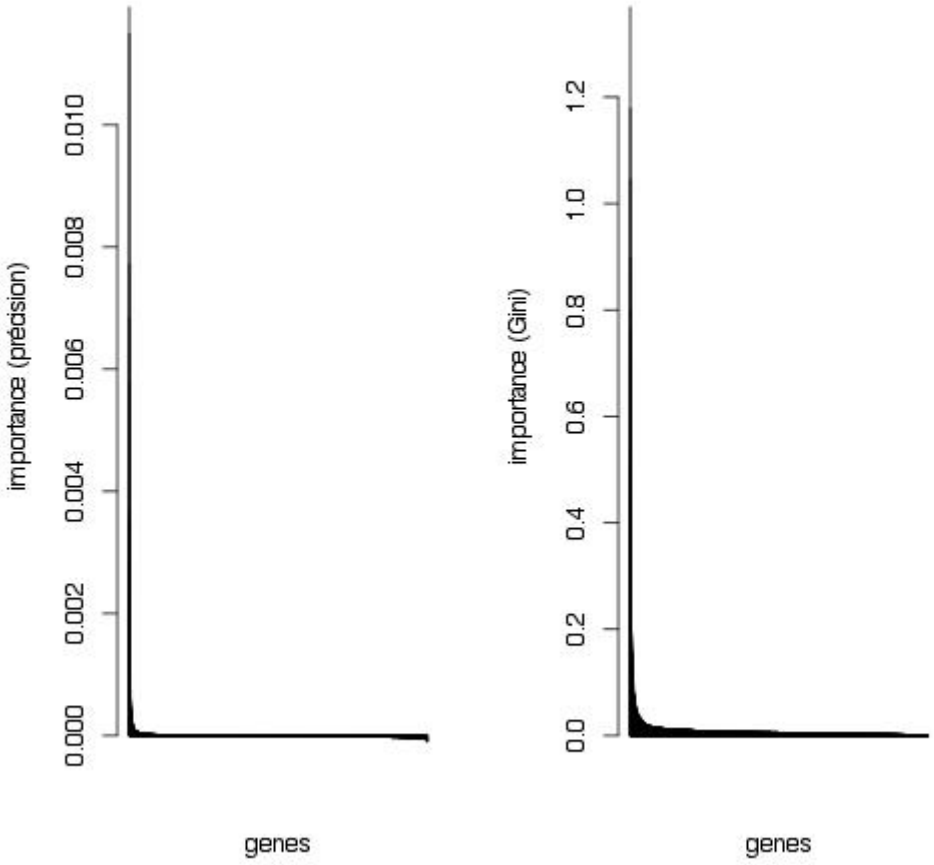


Figure 20: Importance des 19 gènes les plus important selon les critères de la précision et de Gini

On remarque les quelques gènes très importants selon les deux critères, formant visuellement un grand pic sur le graphique. Les gènes principalement recherchés sont les gènes exprimés distinctement selon les races. Afin de détecter ces gènes particulièrement, on a sélectionné les 20 premiers gènes selon les deux critères d'importance et on a gardé les gènes en commun dans les deux listes (19). La Figure 20 est une heatmap croisant la classification des gènes les plus importants (en colonne) et les individus triés par races (en ligne).

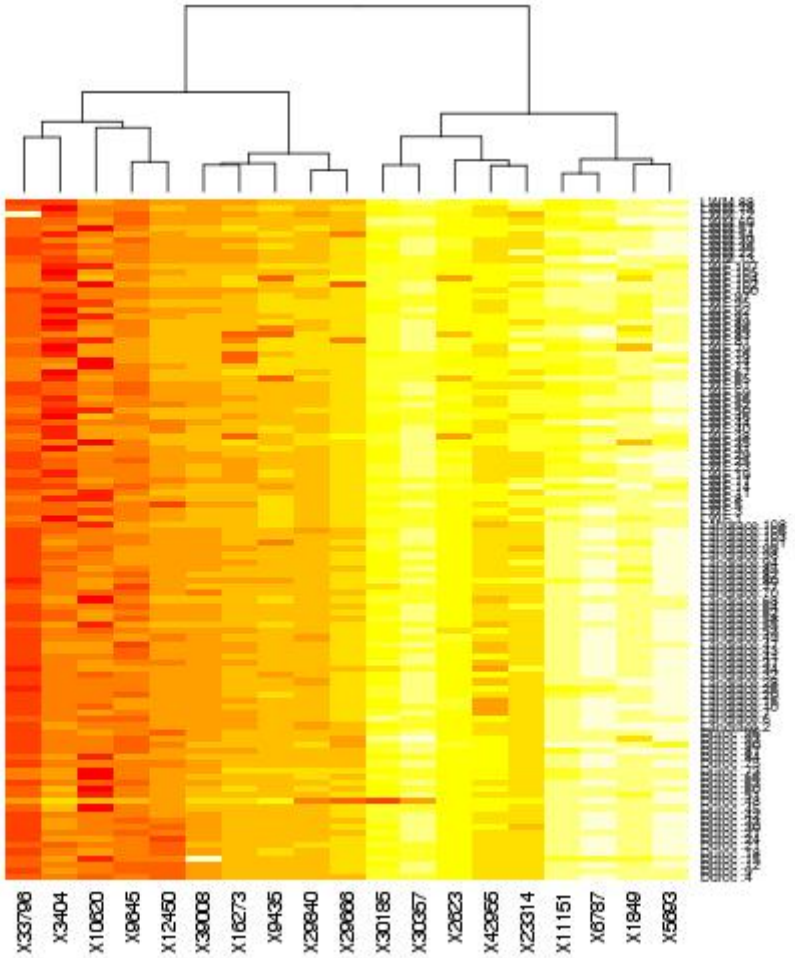


Figure 21: Heatmap croisant la classification des gènes les plus importants pour les deux critères et les individus triés par race

Peu de gènes ont clairement le profil recherché, cependant, le gène X3404 est un bon candidat, car il semble s'exprimer plus fortement chez les individus de race LWM et LWF que chez les individus de race landrace. Il semble encore moins exprimé chez les individus de race Duroc. Voici ci dessous les Boîtes à moustaches parallèles selon les races pour les 19 gènes les plus importants, apportant des informations plus précises.

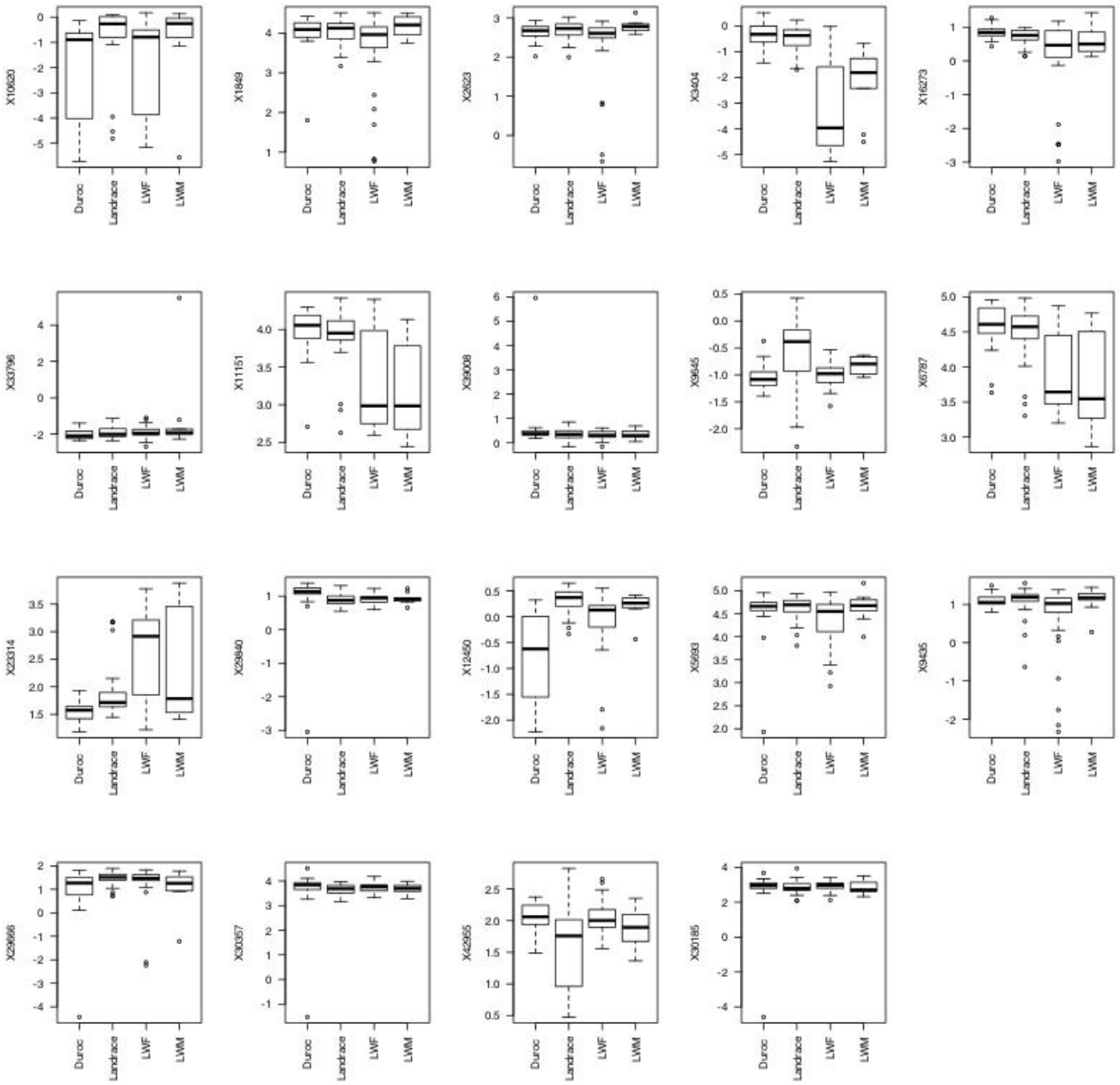


Figure 22: Boîtes à moustaches parallèles selon les races pour les 19 gènes les plus importants

On remarquera 2 types de phénomènes pour ces gènes, expliquant qu'ils soient considérés comme importants : Les gènes tels le gène X3404 déjà relevé pour la heatmap, montrant une influence claire de la race sur l'expression du gène, confirmé par une p-value du test ANOVA d'indépendance de la race sur l'expression de ce gène, hypothèse significativement rejetée avec une p-value de $2x10^{-16}$; et ceux, tels le gène X16273, montrant une influence exercée en grande majorité par des individus atypiques. Ceux-ci, n'influençant peut-être pas significativement la race, contribuent peut-être à faire baisser le pouvoir prédictif du modèle.

Il est clair que la forêt est mauvaise pour les deux races Large White, proches génétiquement, qui peuvent fausser les prédictions. Une nouvelle forêt aléatoire a été réalisée en regroupant ces deux races en une seule appelée LW. La même méthodologie que précédemment a été appliquée pour

trouver les hyper-paramètres donnant lieu au meilleur modèle au niveau de l'erreur de prédiction. Le Tableau 13 ci dessous indique les différents modèles testés.

set,seed()	ntree	mtry	sampsize	OOB error
14	1000	100	75	0,0381
34533	5000	100	75	0,0381
34533	1000	400	75	0,0382
3453	3000	100	75	0,0286
33	500	100	75	0,0095

Tableau 13: Modèles utilisés pour la forêt aléatoire avec la réunion des races LW

Le modèle retenu est un modèle à 500 arbres (le nombre d'arbres a été diminué car les modèles avec plus d'arbres avaient tendance à surapprendre, c'est à dire à appendre trop d'information, et intégrer dans le modèle les erreurs de mesure), 100 variables utilisés par division d'arbres, et des échantillons bootstrap contenant 75 individus. Ce modèle donne seulement 0,95% de mauvais classement, ce qui montre une très nette amélioration par rapport au modèle précédent, et des prédictions très solides.
La Figure 23 ci dessous montre l'évolution du taux de mauvais classement (ou erreur OutOfBag) en fonction du nombre d'arbres pour cette forêt aléatoire.

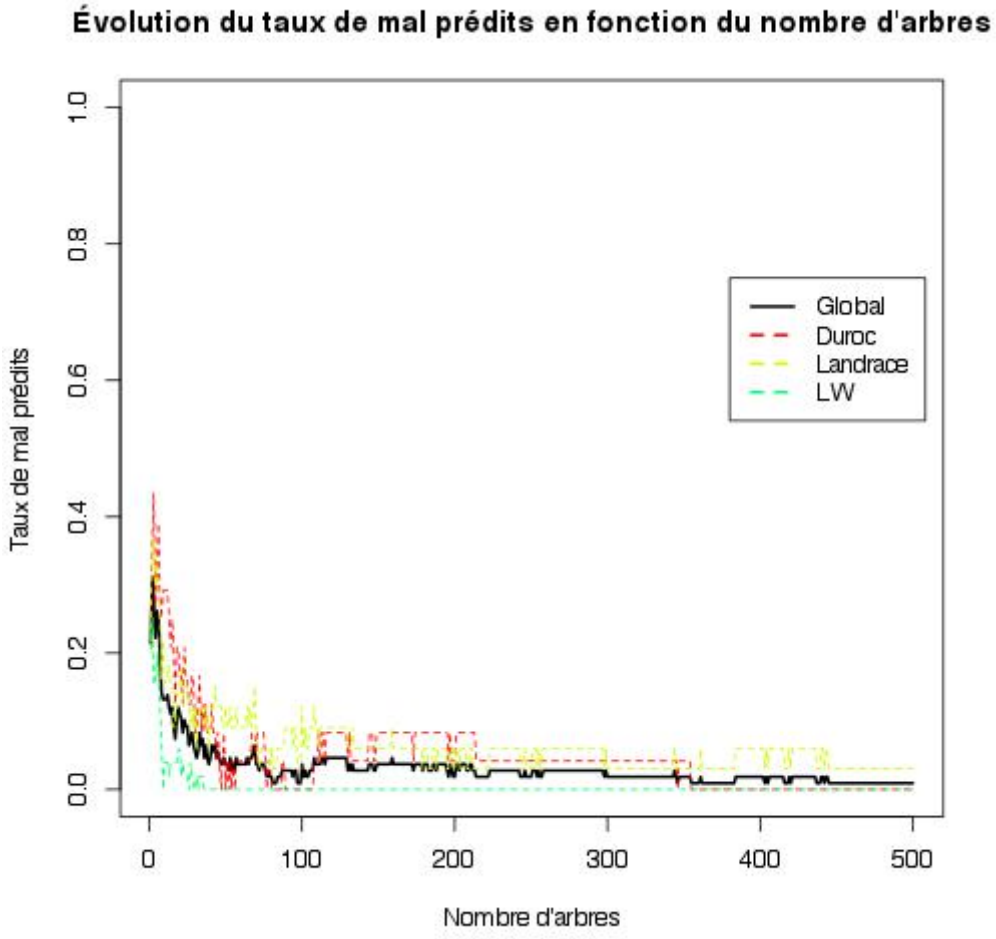


Figure 23: évolution de l'erreur OutOfBag en fonction du nombre d'arbres

Il est à noter que dans ce cas, l'erreur n'est pas totalement stabilisée à 500 arbres, en effet, les autres modèles ont montré que plus d'arbres finissaient par faire augmenter l'erreur, correspondant probablement à du surapprentissage. On remarque également que les erreurs sont similaires pour chaque race.

Le Tableau 14 ci dessous est une table de contingence croisant les valeurs prédites aux valeurs observées.

Valeurs Prédites / Valeurs Observées	Duroc	Landrace	LW	class,error
Duroc	24	0	0	0,0000
Landrace	0	32	1	0,0303
LW	0	0	51	0,0000

Tableau 14: Tableau de contingence des valeurs prédites croisées avec les valeurs observées

Le taux d'erreur est effectivement très faible, seul un individu de race Landrace a été prédit dans la race LW. Les résultat quant aux gènes contribuant à la race seront sans doute pertinents. Voici donc dans le Tableau 15 les 16 gènes les plus importants selon les critères de la précision et de Gini, correspondant à l'intersection des 20 gènes les plus importants selon ces deux critères. Ces gènes sont tous également inclus dans la liste des gènes du modèle mixte créé par les biologistes de l'INRA. Dans ce modèle mixte, des p-value du test de nullité des coefficients dans un modèle mixte (non étudié en seconde année).

N° de gène	Accuracy	Gini
X3404	0,010	0,732
X7762	0,007	0,523
X22274	0,007	0,453
X27309	0,006	0,536
X11878	0,006	0,508
X31039	0,006	0,519
X15632	0,006	0,414
X20447	0,006	0,469
X36089	0,005	0,480
X16335	0,005	0,363
X12647	0,005	0,401
X9965	0,005	0,383
X10512	0,005	0,421
X2093	0,004	0,420
X25459	0,004	0,456
X29104	0,004	0,344

Tableau 15: Gènes les plus importants pour la prédiction de la race

La précédente forêt aléatoire avait permis, tout comme cette dernière de repérer le gène X3404, qui est ici le plus important sur les deux critères. La Figure 24 ci dessous montre une représentation de la matrice de corrélation des expressions des gènes les plus importants.

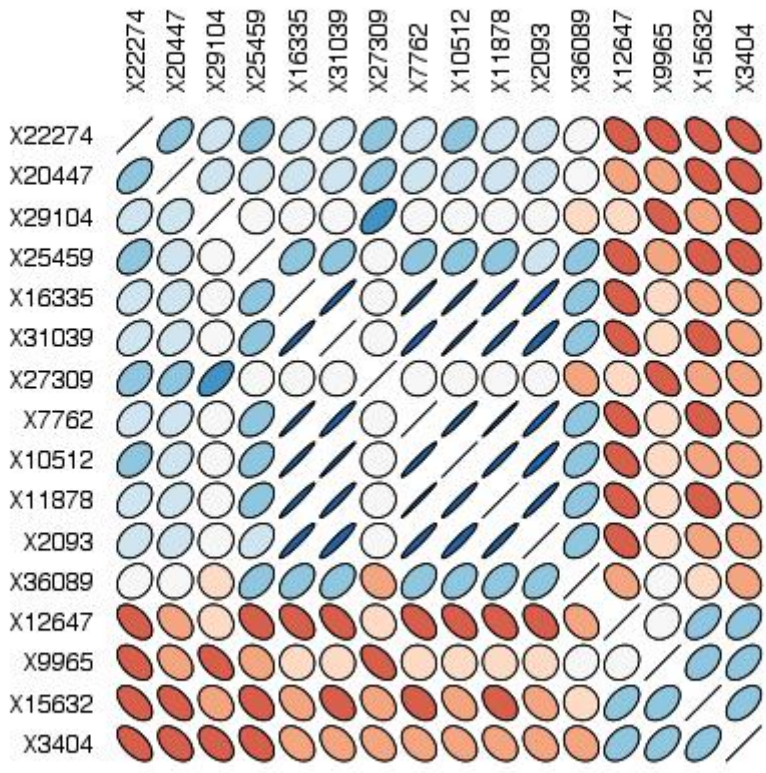


Figure 24: Corrélations des expressions des gènes les plus importants

On distinguera 3 groupes de croisements : certains, comme le croisement entre le gène X7762 et X11878, très fortement corrélés positivement (représentés par des ellipses très écrasés, colorée en bleu foncé), d'autres, comme le croisement entre le gène X10512 et X22274, peu corrélés positivement (représentés par des ellipses moins écrasées colorée plus clairement), et des croisements comme celui entre X3404 et X22274, assez corrélés négativement (représentés par des ellipses colorés en nuances de rouge).

La heatmap en Figure 25 croisant ces gènes aux individus triés par race, permettra de voir plus clairement les gènes différentiellement exprimés selon les races (cette heatmap est présentée à la verticalement contrairement à la précédente, pour une meilleure visibilité du phénomène).

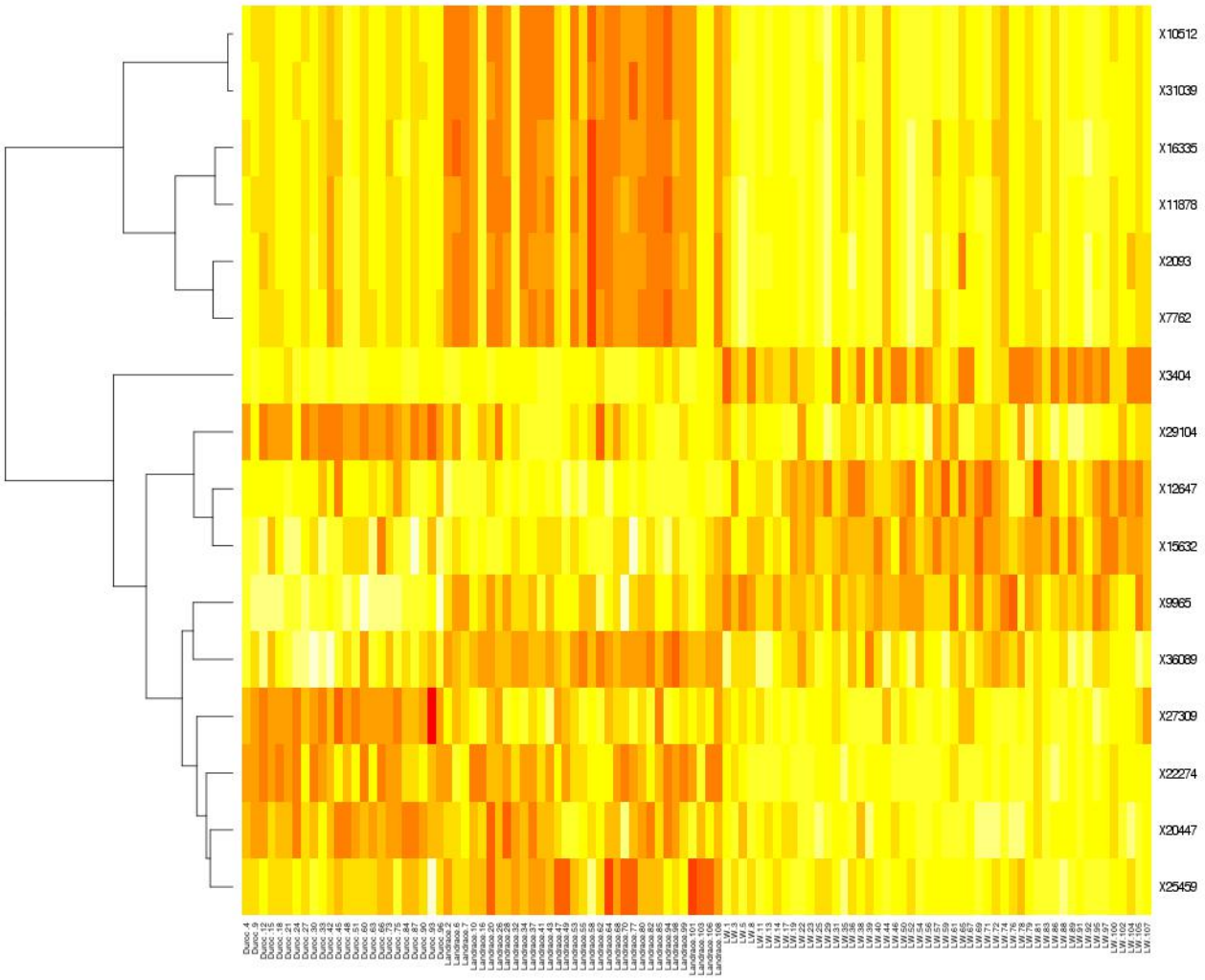


Figure 25: Heatmap croisant la classification des gènes les plus importants pour les deux critères et les individus triés par race

Les résultat sont assez clairs pour certains gènes, comme le gène X3404 très exprimé chez les LW, et moins exprimé chez les autres races, le gène X36089, s'exprimant surtout sur les individus de race landrace, et le gène X27309, davantage exprimé chez les duroc. La Figure 26 ci présente une version simplifiée de la heatmap en Figure 25, en présentant la moyenne des expressions des gènes importants par race, et non l'expression par individu. La tendance globale est écrite dans le Tableau 16 récapitulant les gènes importants selon leur spécificité à une race donnée. Un gène bien exprimé chez une race (comparativement aux autres) sera noté par un +, et un gène peu exprimé chez une race (comparativement aux autres) sera noté par un -. Ce tableau présente également, pour chaque gène, la valeur de la p-value du test d'effet du facteur races.
La Figure 27 présente les boîtes à moustaches parallèles pour chaque race de ces mêmes gènes importants.

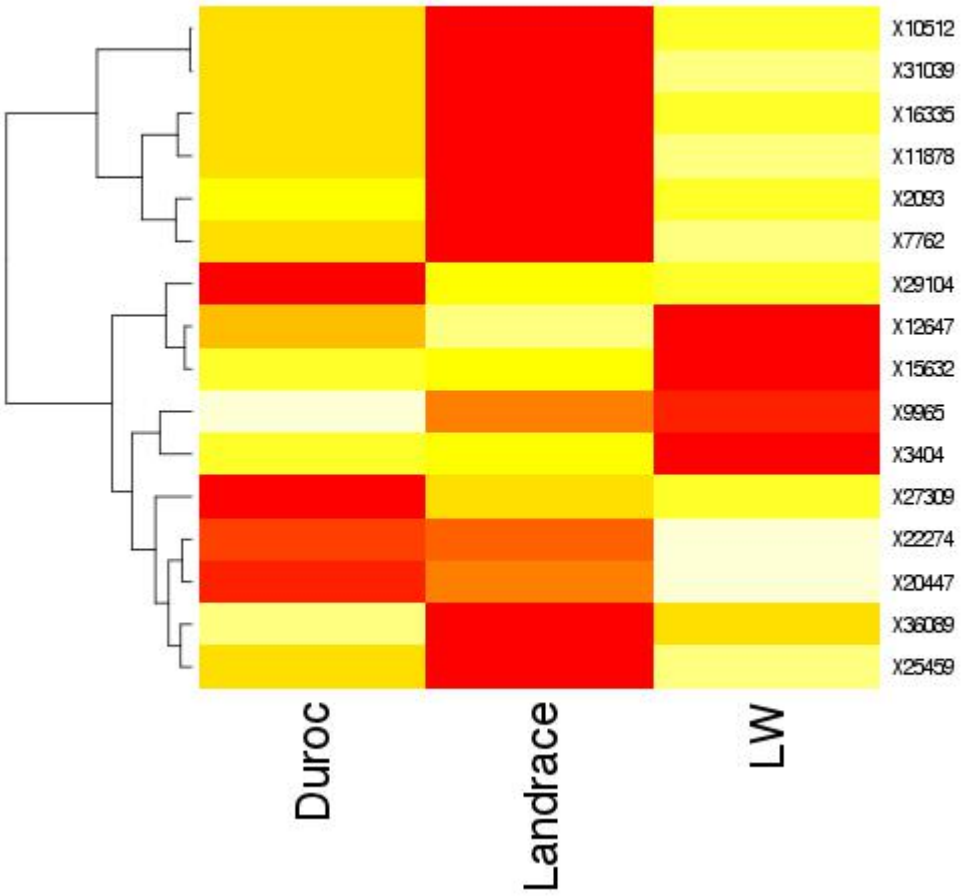


Figure 26: Heatmap croisant la classification de la moyenne des gènes importants par race, et les différentes races

Gène	LW	Landrace	Duroc	p-value
X25459	-	+	-	$2{,}91*10^{-16}$
X20447	-	+	+	$2*10^{-16}$
X22274	-	+	+	$2*10^{-16}$
X27309	-	-	+	$2*10^{-16}$
X36089	-	+	-	$2{,}76*10^{-16}$
X9965	+	+	-	$2*10^{-16}$
X15632	+	-	-	$8{,}64*10^{-16}$
X12647	+	-	+	$6{,}13*10^{-15}$
X29104	-	-	+	$4{,}16*10^{-16}$
X3404	+	-	-	$2*10^{-16}$
X7762	-	+	-	$2*10^{-16}$
X2093	-	+	-	$2*10^{-16}$
X11878	-	+	-	$2*10^{-16}$
X16335	-	+	-	$2*10^{-16}$
X31039	-	+	-	$2*10^{-16}$
X10512	-	+	-	$2*10^{-16}$

Tableau 16: Taux d'expression des gènes importants par race

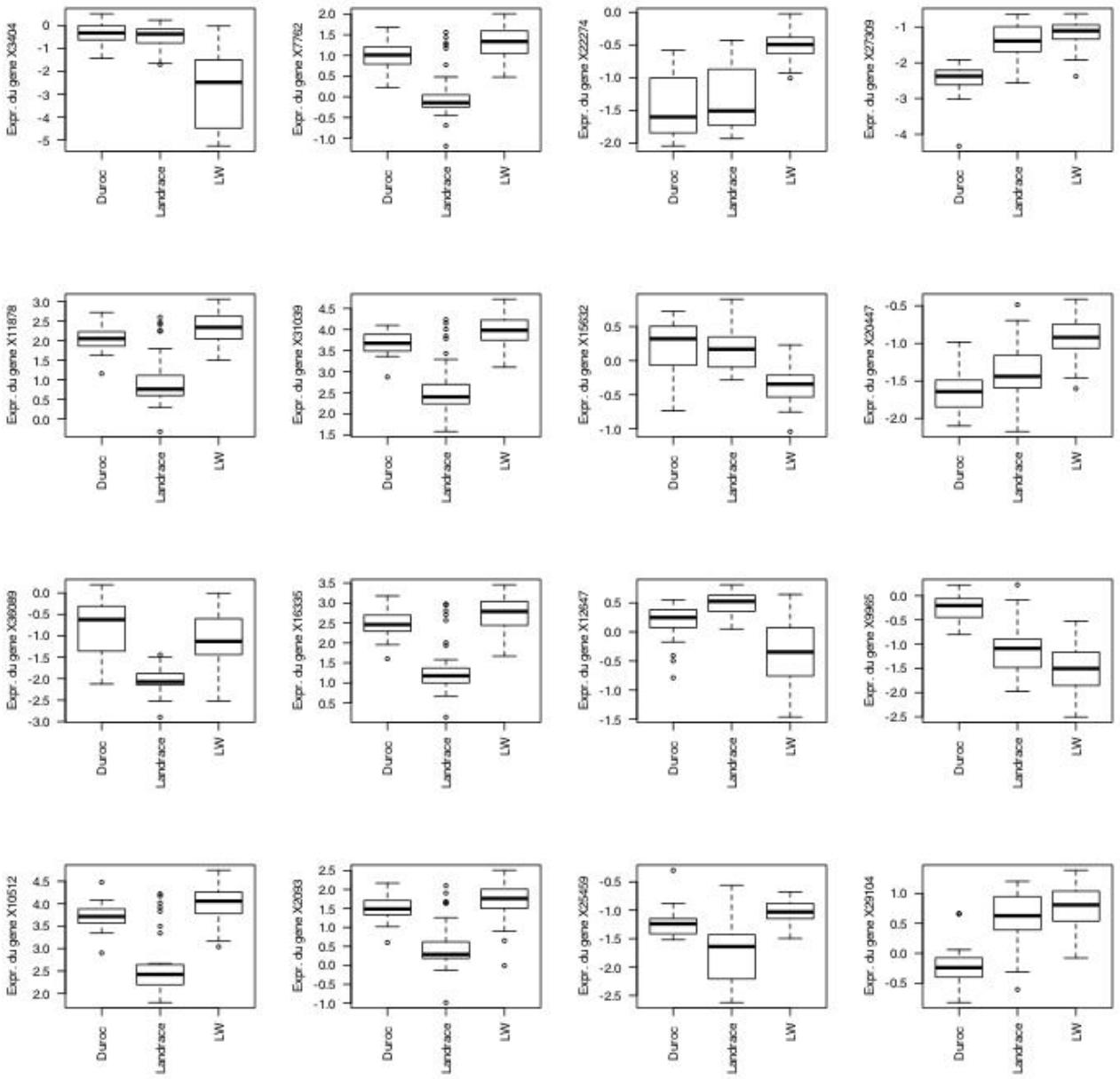


Figure 27: Boîtes à moustaches parallèles pour les races des expressions des gènes les plus importants

Chacun de ces gènes présente des différences notables selon les races, montré par des p-value du test d'égalité des variances pour le facteur race toutes significatives. Il s'agit donc de gènes codant certainement pour un caractère d'une race particulière, s'agissant ainsi de gènes déterminants pour la race. Certains gènes importants présentés ici se retrouvent également sur l'axe 5 de l'ACP, regroupant clairement les races entre elles. C'est le cas des gènes X3404, X29104 et X20447.
Il est à noter que les p-value affichées ne sont données qu'à titre indicatif, les résultats étant assez peu fiables, étant donné que les conditions de normalité et d'homogénéité des variances de Bartlett ne sont pas respectés pour toutes les races, d'autant plus que ces dernières peuvent être influencés par la bande, fortement liée à la race.

3) Prédiction du phénotype d'intérêt

On va maintenant tenter de prédire le phénotype d'intérêt par forêt aléatoire. Ce phénotype est traduit par 2 mesures. Pour cette prédiction, une seule a été utilisée. Ils s'agit de lgr.carc.r, qui a l'avantage d'être normalisé. Les prédictions n'étaient pas très bonnes, et il a fallu, comme pour la prédiction de la race, chercher des hyper-paramètres permettant le meilleur R^2 entre les valeurs prédites et les valeurs observées. Le Tableau 17 récapitule les modèles utilisés, précisant les hyper-paramètres et le R^2 relatif à chaque modèle.

set.seed	ntree	mtry	sampsize	R^2
34234	5000	400	75	0,0179
4324532	12500	400	75	0,0196
3234	12500	800	75	0,0206
5233	12500	100	75	0,0212
112358	7500	400	75	0,0222
897423	7500	800	75	0,0225
897423	7500	800	75	0,0225
34234	7500	200	75	0,0232
5234	5000	800	20	0,0241
52334	12500	800	50	0,0241
5233	20000	1000	50	0,0241
343234	7500	100	75	0,0262
2056	20000	50	75	0,0267
343234	7500	100	50	0,0271
52334	12500	100	50	0,0276
5234	5000	800	50	0,0296
5233	5000	100	50	0,0300

Tableau 17: Récapitulatifs des modèles de forêts aléatoires utilisés

On utilisera le modèle ayant le R^2 le plus grand, de 0,03. Il a été construit avec 5000 arbres, 100 variables par division d'arbres, et 50 individus dans les échantillons bootstrap. Cependant, le R^2, bien que meilleur que pour les autres forêts aléatoires, reste très faible, et la prédiction sera très mauvaise donc les gènes repérés comme « importants » peu fiables. Faire des forêts aléatoires en réunissant, comme pour la prédiction de la race, les deux races LW, n'a pas donné de résultats plus satisfaisant. Nous nous en tiendrons donc à celle-ci.
Voici ci dessous l'évolution de l'erreur Out Of Bag en fonction du nombre d'arbres pour cette forêt aléatoire.

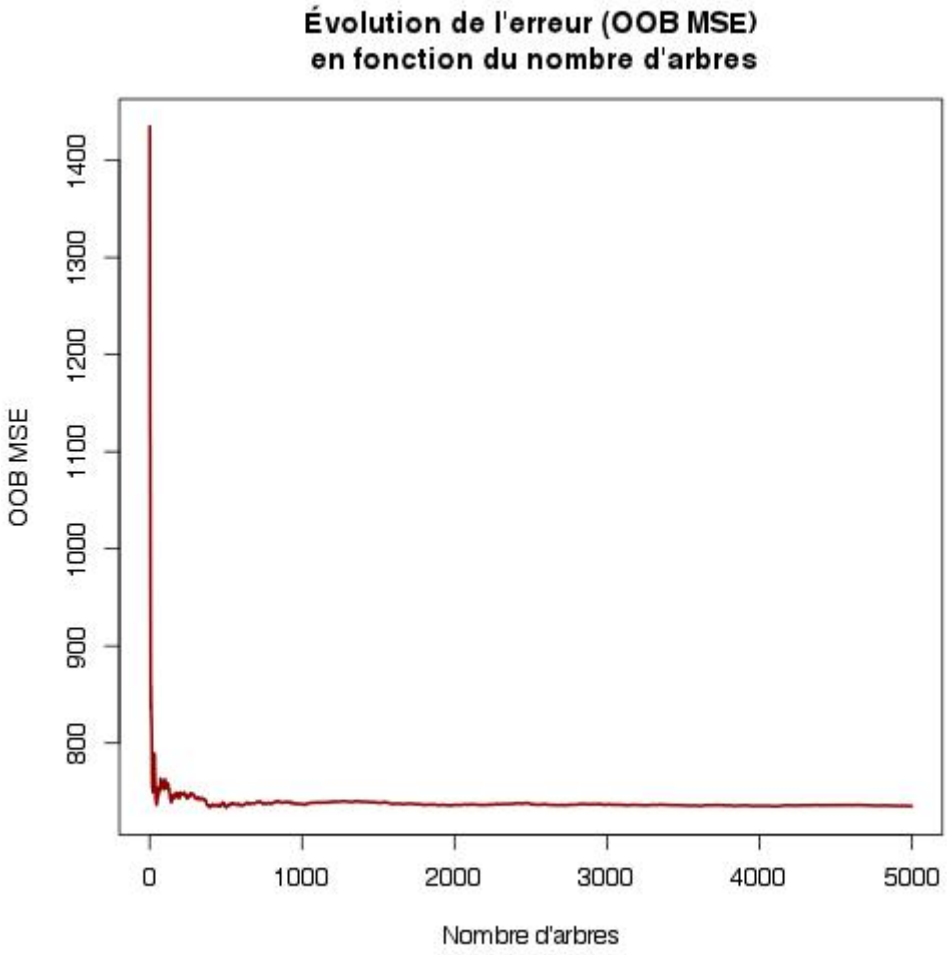


Figure 28: évolution de l'erreur Out Of Bag en fonction du nombre d'arbres

Ici, l'erreur s'est stabilisée à partir d'un milliers d'arbres. Voici en Figure 29 le nuage de points des valeurs prédites et observées.

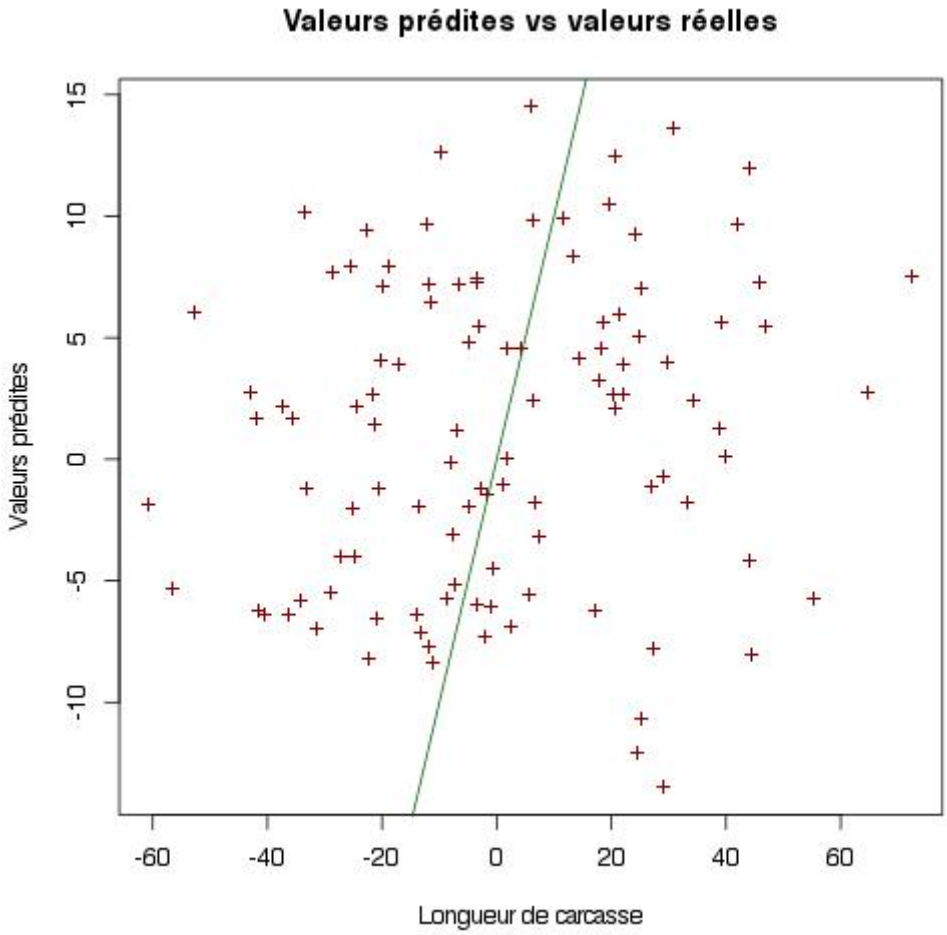
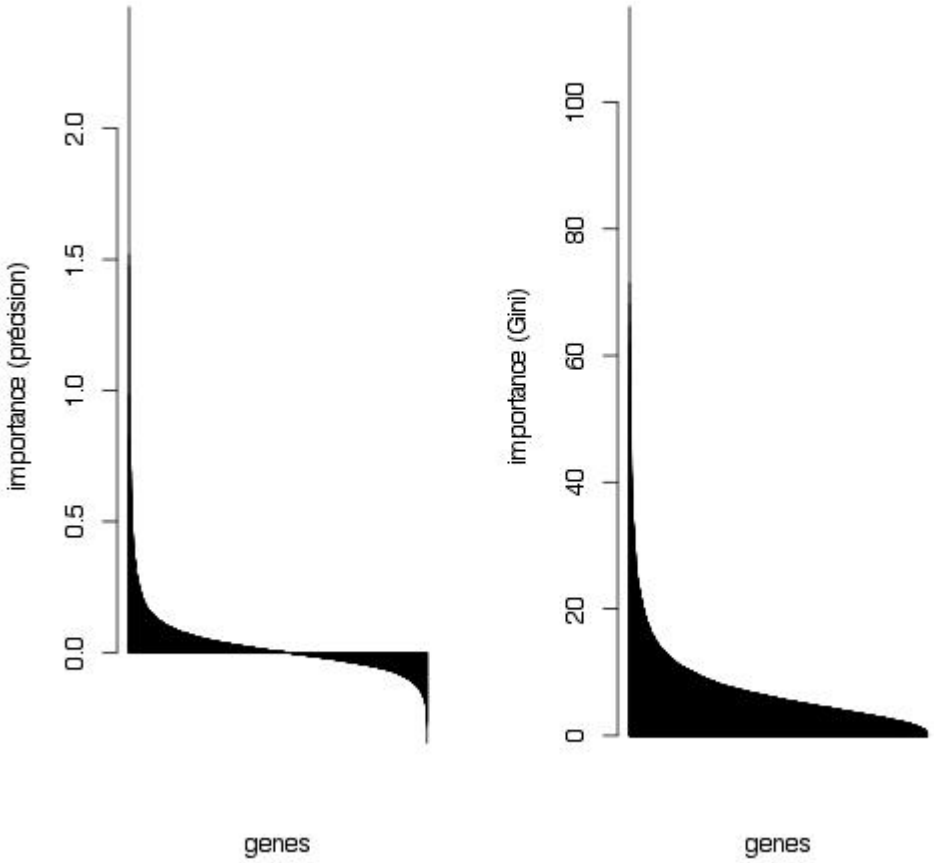


Figure 29: Nuage de point pour les valeurs prédites et les valeurs observées

On remarque que le nuage de points ne semble suivre aucune tendance particulière. Le modèle semble donc effectivement assez mauvais au niveau de la prédiction. Ceci semble traduire le fait que les phénomènes biologiques influant sur la taille de la carcasse sont sans doute liés en majeure partie à autre chose que le génome, comme la race, ou peut-être les facteurs environnementaux.
Les gènes les plus importants ont tout de même été relevés. Voici les 14 premiers gènes les plus importants selon les critères de la précision et le critère de Gini, ainsi que la représentation graphique de cette importance.

Nom du gène	Accuracy	Gini
X19892	2,46	115
X27016	1,52	65,05
X27224	1,47	61,52
X41791	0,98	68,11
X32751	0,83	53,38
X21749	0,81	62,1
X44990	0,81	46,48
X17513	0,78	48,71
X26483	0,77	50,31
X35018	0,75	57,13
X42065	0,74	71,42



Ici encore, quelques gènes sont très importants du point de vue des deux critères, et cette importance diminue vite pour les autres gènes, mais ce phénomène était bien plus marqué lors de la prédiction de la race. On recherche ici des gènes plus exprimés selon la valeur de la mesure du phénotype. Cela va se faire grâce à l'utilisation d'une Heatmap de la classification des gènes les plus importants de la forêt aléatoire croisés avec les individus ordonnés par leur valeur de lgr.carc.r.

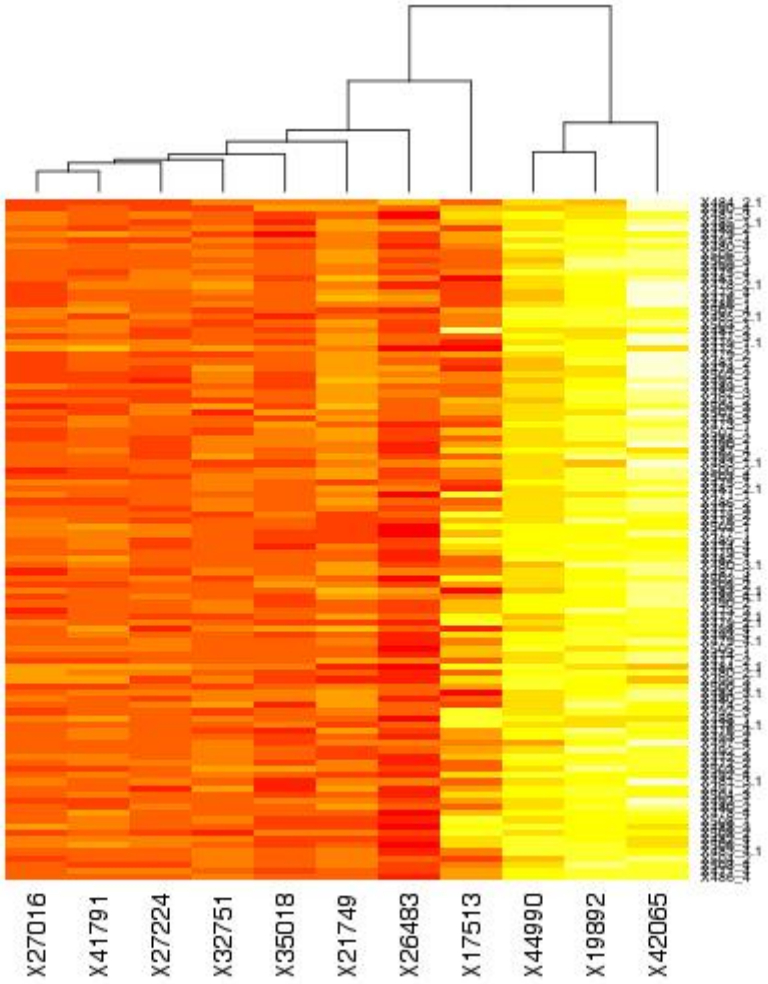


Figure 30: Heatmap de la classification des gènes les plus importants de la forêt aléatoire croisés avec les individus ordonnés par leur valeur de lgr.carc.r

Ici, les résultats sont peu clairs, cependant, parmi ces gènes, l'un des gènes se rapprochant le plus du comportement recherché est le gène X19892.
En effet, une régression linéaire de la variable à expliquer lgr-carc-r, et l'expression de ce gène donne une p-value du test de Fisher de dépendance entre les variances expliquées et non expliquées de $1{,}143 \times 10^{-5}$, traduisant le fait que ce gène explique bien la variation de la mesure du phénotype, avec un R^2 correct, de -0,41. Voici en Figure 31 le nuage de points de ces deux variables.

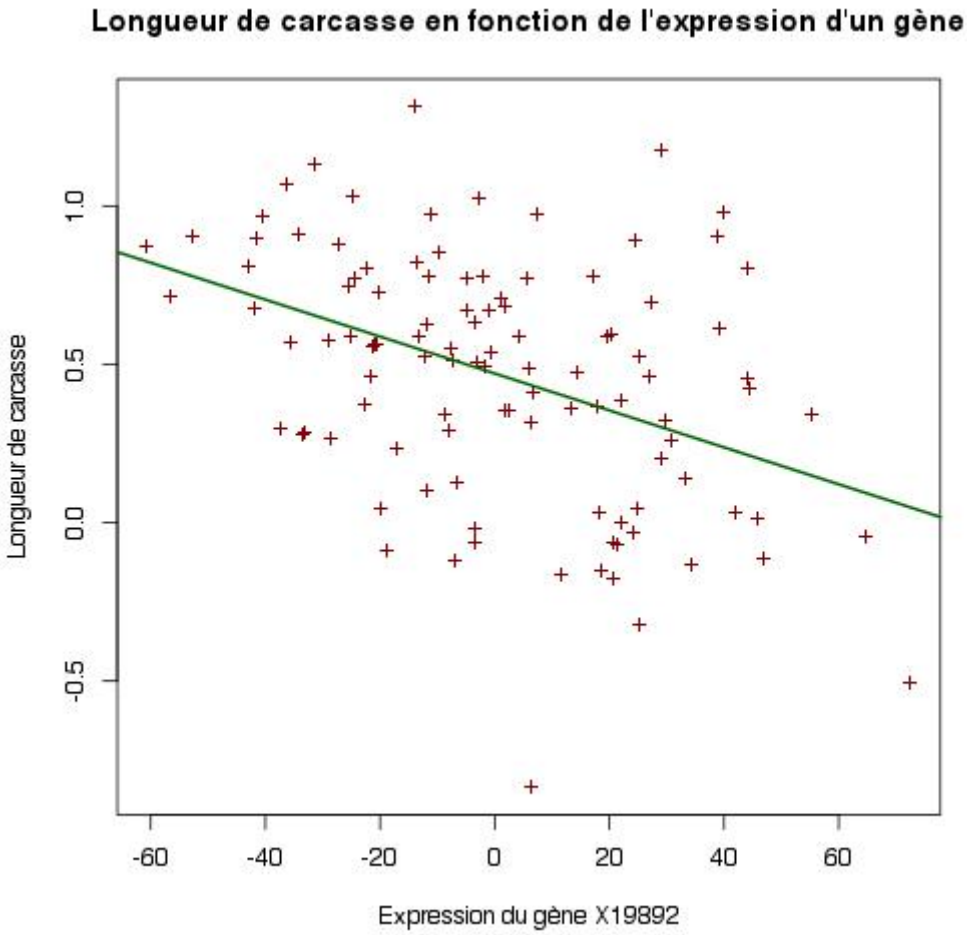


Figure 31: Nuage de points de la longueur de la carcasse en fonction de l'expression du gène X19892

La longueur de la carcasse semble effectivement dépendre de l'expression de ce gène, selon la droite d'équation y = -28,855 x + 14,499. Malgré le fait que le phénotype semble dépendre de nombreux facteurs extérieurs au génome, il semblerait qu'on puisse trouver des gènes capables de l'influencer significativement.

Conclusion

L'échantillon utilisé pour cette étude est très varié, de part la mixité des races des cochons, ainsi que leur conditions d'élevage, indiqué par la bande. Ces deux facteurs ont apporté une variabilité génétique dont il a fallu tenir compte, d'autant plus que leur distribution n'était pas homogène.

Cette étude avait pour but de chercher les gènes, à partir des mesures de leur expressions, déterminant la race et un phénotype d'intérêt des cochon, en rapport avec le poids, par le biais de méthodes exploratoires et prédictives. La plupart des gènes avaient une expression moyenne faible, voire très faible. Cependant, l'ACP a montré que des liens existaient entre l'expression des gènes et la race, qui a été confirmé par la prédiction par forêt aléatoire, très solide, montrant des gènes déterminants pour la race des cochons, le plus significatif étant le gène X3404, très exprimé chez les cochons de race Large White, et pas chez les autres. Prédire le phénotype, mesuré plus spécialement par la largeur de la carcasse, est bien plus délicat, ce phénotype étant fortement influencé par la race, et très peu par les expressions des gènes, d'autant plus qu'il peut très fortement varier,

notamment selon les conditions environnementales. Cependant, les forêts aléatoires, bien que donnant des prédictions très peu solides, ont permis de relever quelques gènes semblant influencer le phénotype, comme le gène X19892.

Un temps plus important aurait permis de réaliser de nouveaux modèles, afin de tenter de les améliorer, par exemple, la race influant fortement sur le phénotype, considérer des forêts aléatoires pour chaque race indépendamment, et chercher des gènes importants se retrouvant dans chacune des forêts aléatoires.

Annexe 1 : Résultats détaillés de l'ACP

	Dim.1	Dim.2		Dim.3	Dim.4		Dim.5	Dim.6		Dim.7	Dim.8
X31954	0,919	0,018	***X20957***	0,747	0,000	***X3404***	0,495	0,083	***X13198***	0,285	0,324
X40312	0,920	0,009	***X29019***	0,687	0,002	***X12980***	0,500	0,006	***X9386***	0,313	0,293
X28303	0,929	0,001	***X34949***	0,677	0,004	***X35236***	0,477	0,014	***X20495***	0,252	0,349
X3780	0,918	0,008	***X6270***	0,616	0,063	***X27309***	0,465	0,009	***X39307***	0,255	0,341
X12390	0,888	0,038	***X21268***	0,630	0,045	***X44140***	0,412	0,046	***X7525***	0,209	0,382
X4695	0,847	0,075	***X8804***	0,671	0,000	***X35027***	0,408	0,049	***X38710***	0,317	0,262
X21241	0,920	0,001	***X33100***	0,657	0,004	***X40263***	0,431	0,016	***X28241***	0,272	0,291
X20575	0,904	0,016	***X37564***	0,639	0,008	***X29104***	0,445	0,000	***X30666***	0,233	0,329
X36422	0,913	0,006	***X5405***	0,630	0,014	***X16033***	0,002	0,442	***X7163***	0,190	0,360
X31009	0,912	0,004	***X4650***	0,597	0,033	***X32449***	0,294	0,147	***X24180***	0,208	0,339
X38395	0,907	0,009	***X10654***	0,616	0,013	***X23566***	0,358	0,082	***X11761***	0,264	0,276
X20966	0,903	0,012	***X15511***	0,627	0,001	***X20447***	0,436	0,002	***X8939***	0,165	0,373
X27130	0,911	0,004	***X24216***	0,622	0,001	***X37709***	0,407	0,019	***X15986***	0,128	0,409
X36419	0,893	0,021	***X30208***	0,617	0,001	***X22274***	0,373	0,043	***X18051***	0,239	0,296
X39558	0,904	0,004	***X4099***	0,598	0,019	***X9965***	0,368	0,038	***X28237***	0,299	0,232
X31049	0,899	0,008	***X26071***	0,601	0,010	***X38230***	0,134	0,265	***X13121***	0,117	0,413
X25615	0,850	0,055	***X26109***	0,590	0,009	***X1393***	0,080	0,313	***X37201***	0,262	0,263
X27722	0,883	0,018	***X3062***	0,596	0,002	***X26905***	0,362	0,027	***X12079***	0,299	0,222
X17679	0,901	0,001	***X2001***	0,554	0,041	***X40076***	0,232	0,156	***X1908***	0,086	0,427
X43330	0,897	0,004	***X36031***	0,576	0,003	***X23314***	0,347	0,038	***X10064***	0,251	0,258

Annexe 2 : Script R des analyses

Importation et préparation des données

```
# Les données sont importées par :
x <- read.table("project-data.csv",sep=" ",header=TRUE,dec=".")

# Pour obtenir la moyenne et l'écart type de l'expression des gènes, on calcule
 ces deux statistiques pour chacun des gènes :
x.means <- apply(x[,1:4674],2,mean)
x.sds<-apply(x[,1:4674],2,sd)

# On construit des graphiques à partir de ces informations :
hist(x.means, main="Répartition de la moyenne des expressions de gènes",
 xlab="Valeurs de la moyenne", ylab="Nombre de variables")

hist(x.sds, main="Répartition de l'écart type des expressions de gènes",
 xlab="Valeurs de l'écart type", ylab="Nombre de variables")


# On extrait du jeu de données les expressions atypiques des gènes :
which(x.means>4)
which(x.means<(-2))

which(x.sds>1.5)
which(x.sds<0.25)
```

```
# Ensuite on calcule à l'aide du package e1071 les coefficients d'asymétrie et
 d'aplatissement :
library(e1071)

kurtosis(x.means)
skewness(x.means)

kurtosis(x.sds)
skewness(x.sds)
```

Analyses descriptives uni et bi-variées

```
## Analyse univariée de la race
#Graphique
pie(table(x$race), col=rainbow(5), main="Diagramme des races")
# Tableau d'effectifs
table(x$race)


## Analyse univariée de la bande
#Graphique
barplot(table(x$anstaba), ylab="Effectifs", xlab="Bande", main="Répartition des
 bandes", col=rainbow(8))
# Tableau d'effectifs
table(x$anstaba)


## Analyse bivariée race/bande
# Diagramme en tuyaux d'orgue de la distribution de la race conditionnellement à
 la bande
barplot(round((TCRA)/apply(TCRA, 2, sum),
 digits=2),col=ind.race.colors,legend.text=dimnames(TCRA)[[1]],
 args.legend=list(x="topright"), ylab="Effectifs")

# Table de contingence entre la race et la bande
TCRaceAnstaba<-table(x$race, x$anstaba)
TCRaceAnstaba<-cbind(TCRaceAnstaba, apply(TCRaceAnstaba,1,sum))
TCRaceAnstaba<-rbind(TCRaceAnstaba, apply(TCRaceAnstaba,2,sum))
dimnames(TCRaceAnstaba)[[1]][6]="Total"
dimnames(TCRaceAnstaba)[[2]][9]="Total"
TCRaceAnstaba

#Test du Chi-deux d'indépendance entre la race et la bande
TCRA<-TCRaceAnstaba[-nrow(TCRaceAnstaba), -ncol(TCRaceAnstaba)]
TCRA
chisq.test(TCRA)


## Phénotypes
# Statistiques univariées de base pour les deux variables de longueur de
 carcasse
summary(y$lgr.carc)
summary(y$lgr.carc.r)
skewness(y$lgr.carc)
skewness(y$lgr.carc.r)
kurtosis(y$lgr.carc)
kurtosis(y$lgr.carc.r)

# Histogrammes des deux phénotypes
```

```
hist(y$lgr.carc, main="Histogramme représentant une mesure de phénotype \n en
 relation avec la longueur des carcasses", ylab="Effectif", xlab="Mesure d'un
 phénotype en relation avec le poids")
hist(y$lgr.carc.r, main="Histogramme représentant la mesure d'un autre phénotype
 \n en relation la longueur des carcasses", ylab="Effectif", xlab="Mesure d'un
 phénotype en relation avec le poids")

#corrélation entre les deux phénotypes
cor(y.notDchn$lgr.carc,y.notDchn$lgr.carc.r)
cor.test(y.notDchn$lgr.carc,y.notDchn$lgr.carc.r)

#représentation graphique de la corrélation entre les deux phénotypes
plot(y.notDchn$lgr.carc,y.notDchn$lgr.carc.r, pch="+", xlab="Phénotype",
 ylab="Autre phénotype")

# Corrélation avec la race : ANOVA de la longueur de carcasse en fonction de la
 race
# Test de normalité de la distribution de la longueur de carcasse dans chacune
 des races (Shapiro-Wilk)
tapply(y.notDchn$lgr.carc,y.notDchn$race,shapiro.test)
# Test d'égalité des variances des longueurs de carcasse des chacunes des races
 (Bartlett)
bartlett.test(y.notDchn$lgr.carc,y.notDchn$race)
# ANOVA : Test d'égalité des longueurs de carcasse moyenne entre les races
anova1<-aov(lgr.carc ~ race, data=y.notDchn)
summary(anova1)
# Boîte à moustaches de la distribution de la longueur de carcasse selon la race
boxplot(y.notDchn$lgr.carc ~ y.notDchn$race, main="Boîtes à moustache d'un
 phénotype en relation \n avec le poids de la carcasse selon la race",
 xlab="Race", ylab="Phénotype en relation avec le poids")


# Corrélation avec la race : ANOVA de la longueur de carcasse corrigée en
 fonction de la race
tapply(y.notDchn$lgr.carc.r,y.notDchn$race,shapiro.test)
bartlett.test(y.notDchn$lgr.carc.r,y.notDchn$race)
anova2<-aov(lgr.carc.r ~ race, data=y.notDchn)
summary(anova2)

# Boîte à moustaches de la distribution de la longueur de carcasse corrigée
 selon la race
boxplot(y.notDchn$lgr.carc.r ~ y.notDchn$race, main="Boîtes à moustache d'un
 autre phénotype en relation \n avec le poids de la carcasse selon la race",
 xlab="Race", ylab="Phénotype en relation avec le poids")

# Corrélation avec la bande
# ANOVA par bande lgr.carc, la longueur de la carcasse
tapply(y.notDchn$lgr.carc,y.notDchn$anstaba,shapiro.test)
bartlett.test(y.notDchn$lgr.carc,y.notDchn$anstaba)
anova3<-aov(lgr.carc ~ anstaba, data=y.notDchn)
summary(anova3)
# Boîte à moustache de la distribution de la longueur de carcasse selon la bande
boxplot(y.notDchn$lgr.carc ~ y.notDchn$anstaba, main="Boîtes à moustache d'un
 phénotype en relation \n avec le poids de la carcasse selon la bande",
 xlab="Bande", ylab="Phénotype en relation avec le poids")
# ANOVA par bande lgr.carc.r, la longueur de la carcasse corrigée
tapply(y.notDchn$lgr.carc.r,y.notDchn$anstaba,shapiro.test)
kruskal.test(y.notDchn$lgr.carc.r,y.notDchn$race)
# Boîte à moustache de la distribution de la longueur de carcasse corrigée selon
 la bande
boxplot(y.notDchn$lgr.carc.r ~ y.notDchn$anstaba, main="Boîtes à moustache d'un
```

```
autre phénotype en relation \n avec le poids de la carcasse selon la bande",
xlab="Bande", ylab="Phénotype en relation avec le poids")
```

Analyse multivariée

```
# Heatmap
x.sort.race <-x[order(x$race),]
x.genes <- as.matrix(x.sort.race[,1:4674])
rownames(x.genes) <- paste(x.sort.race$race,1:111,sep=".")
heatmap(x.genes,Rowv=NA)

# ACP
xPCA<-PCA(x.notDchn[,1:4674],graph=F,ncp=10, scale.unit=F)


# Scree graph
barplot(xPCA$eig$eigenvalue, main="Éboulis des valeurs propres",
 ylab="Valeurs propres", xlab="Axes")

# Représentations des individus avec une couleur par race
ind.race.colors <- rep(rainbow(5)[2],length(x.notDchn$race))
ind.race.colors[x.notDchn$race=="Landrace"] <- rainbow(5)[3]
ind.race.colors[x.notDchn$race=="LWF"] <- rainbow(5)[4]
ind.race.colors[x.notDchn$race=="LWM"] <- rainbow(5)[5]
layout(matrix(c(1,2,3,1,4,5),ncol=3,byrow=T))
plot.new()
legend("center",pch="+",col=rainbow(5)
 [2:5],legend=c("Duroc","Landrace","LWF","LWM"))
plot(xPCA$ind$coord[,1:2],pch="+",col=ind.race.colors,
 main="Représentation des individus \n différenciés selon leur race
 \n sur les axes 1 et 2 de l'ACP", xlab="Axe 1", ylab="Axe 2")
plot(xPCA$ind$coord[,3:4],pch="+",col=ind.race.colors,
 main="Représentation des individus \n différenciés selon leur race
 \n sur les axes 3 et 4 de l'ACP", xlab="Axe 3", ylab="Axe 4")
plot(xPCA$ind$coord[,5:6],pch="+",col=ind.race.colors,
 main="Représentation des individus \n  différenciés selon leur
 race \n sur les axes 5 et 6 de l'ACP", xlab="Axe 5", ylab="Axe 6")
plot(xPCA$ind$coord[,7:8],pch="+",col=ind.race.colors,
 main="Représentation des individus \n différentiés selon leur race
 \n sur les axes 7 et 8 de l'ACP", xlab="Axe 7", ylab="Axe 8")


# Représentations des individus avec une couleur par bande
ind.bande.colors <- rep(rainbow(8)[1],length(x.notDchn$anstaba))
ind.bande.colors[x.notDchn$anstaba=="70604"] <-rainbow(8)[2]
ind.bande.colors[x.notDchn$anstaba=="70606"] <- rainbow(8)[3]
ind.bande.colors[x.notDchn$anstaba=="70608"] <- rainbow(8)[4]
ind.bande.colors[x.notDchn$anstaba=="70610"] <- rainbow(8)[5]
ind.bande.colors[x.notDchn$anstaba=="80604"] <- rainbow(8)[6]
ind.bande.colors[x.notDchn$anstaba=="80606"] <- rainbow(8)[7]
```

```
ind.bande.colors[x.notDchn$anstaba=="80610"] <- rainbow(8)[8]
layout(matrix(c(1,2,3,1,4,5),ncol=3,byrow=T))
plot.new()
legend("center",pch="+",col=rainbow(8),legend=c("70602","70604","7
 0606","70608","70610","80604","80606","80610"))
plot(xPCA$ind$coord[,1:2],pch="+",col=ind.bande.colors,
 main="Représentation des individus \n  différenciés selon leur
 bande \n sur les axes 1 et 2 de l'ACP", xlab="Axe 1", ylab="Axe
 2")
plot(xPCA$ind$coord[,3:4],pch="+",col=ind.bande.colors,
 main="Représentation des individus \n  différenciés selon leur
 bande \n sur les axes 3 et 4 de l'ACP", xlab="Axe 3", ylab="Axe
 4")
plot(xPCA$ind$coord[,5:6],pch="+",col=ind.bande.colors,
 main="Représentation des individus \n  différenciés selon leur
 bande \n sur les axes 5 et 6 de l'ACP", xlab="Axe 5", ylab="Axe
 6")
plot(xPCA$ind$coord[,7:8],pch="+",col=ind.bande.colors,
 main="Représentation des individus \n  différenciés selon leur
 bande \n sur les axes 7 et 8 de l'ACP", xlab="Axe 7", ylab="Axe
 8")


# Représentations des individus avec un niveau de couleurs pour le
 premier phénotype
classes <-
 cut(y.notDchn$lgr.carc,breaks=quantile(y.notDchn$lgr.carc,probs=se
 q(0,1,length=11)),labels=F)
ind.lgr.carc.col <- heat.colors(10)[classes]
quantile(y.notDchn$lgr.carc,probs=seq(0,1,length=11))

layout(matrix(c(1,2,3,1,4,5),ncol=3,byrow=T))
plot.new()
legend("center",pch=19,col=heat.colors(10)
 [1:10],legend=c("[950;970]","[970;990]","[990;995]","[995;1000]","
 [1000;1010]","[1010;1020]","[1020;1026]","[1026;1035]","[1035;1060
 ]","[1060;1100]"))
plot(xPCA$ind$coord[,1:2],pch=19,col=ind.lgr.carc.col)
plot(xPCA$ind$coord[,3:4],pch=19,col=ind.lgr.carc.col)
plot(xPCA$ind$coord[,5:6],pch=19,col=ind.lgr.carc.col)
plot(xPCA$ind$coord[,7:8],pch=19,col=ind.lgr.carc.col)


# Représentations des individus avec un niveau de couleurs pour le
 deuxième phénotype
classes <-
 cut(y.notDchn$lgr.carc.r,breaks=quantile(y.notDchn$lgr.carc.r,prob
 s=seq(0,1,length=11)),labels=F)
ind.lgr.carc.r.col <- heat.colors(10)[classes]
round(quantile(y.notDchn$lgr.carc.r,probs=seq(0,1,length=11)), 2)
layout(matrix(c(1,2,3,1,4,5),ncol=3,byrow=T))
```

```
plot.new()
legend("center",pch=19,col=heat.colors(10)[1:10],legend=c("[-
 60,5;-33,50]","]-33,5;-22,39]","]-22,39;-13,59]","]-13,59;-
 7,29]","]-7,29;-2,04]","]-
 2,04;,6,21]","]6,21;18,99]","[18,99;25,32]","]25,32;39,18]","]39,1
 8;72,41]"))
plot(xPCA$ind$coord[,1:2],pch=19,col=ind.lgr.carc.r.col, xlab="Axe
 1", ylab="Axe 2", main="Représentation des individus \n selon
 l'autre phénotype d'interêt \n sur les axes 1 et 2 de l'ACP")
plot(xPCA$ind$coord[,3:4],pch=19,col=ind.lgr.carc.r.col, xlab="Axe
 3", ylab="Axe 4", main="Représentation des individus \n selon
 l'autre phénotype d'interêt \n sur les axes 3 et 4 de l'ACP")
plot(xPCA$ind$coord[,5:6],pch=19,col=ind.lgr.carc.r.col, xlab="Axe
 5", ylab="Axe 6", main="Représentation des individus \n selon
 l'autre phénotype d'interêt \n sur les axes 5 et 6 de l'ACP")
plot(xPCA$ind$coord[,7:8],pch=19,col=ind.lgr.carc.r.col, xlab="Axe
 7", ylab="Axe 8", main="Représentation des individus \n selon
 l'autre phénotype d'interêt \n sur les axes 7 et 8 de l'ACP")

# Représentation des variables
par(mfrow=c(2,2))
#### Axes 1 et 2
# sélection
axes12.costot <- xPCA$var$cos2[,1]+xPCA$var$cos2[,2]
order.costot.12 <- order(axes12.costot,decreasing=T)
sel12 <- order.costot.12[1:20]

# représentation
plot(seq(-1,1,length=1000),sqrt(1-seq(-
 1,1,length=1000)^2),type="l",ylim=c(-1,1), xlab="Axe 1", ylab="Axe
 2", main="Représentation des 20 variables \n les mieux
 reconstituées \n sur les axes 1 et 2")
lines(seq(-1,1,length=1000),-sqrt(1-seq(-1,1,length=1000)^2))
arrows(0,0,xPCA$var$cor[sel12,1],xPCA$var$cor[sel12,2],length=0.05
 )
text(xPCA$var$cor[sel12,1],xPCA$var$cor[sel12,2],colnames(x)
 [sel12],cex=0.7,col="sienna")

#### Axes 3 et 4
# sélection
axes34.costot <- xPCA$var$cos2[,3]+xPCA$var$cos2[,4]
order.costot.34 <- order(axes34.costot,decreasing=T)
sel34 <- order.costot.34[1:20]
# représentation
plot(seq(-1,1,length=1000),sqrt(1-seq(-
 1,1,length=1000)^2),type="l",ylim=c(-1,1), xlab="Axe 3", ylab="Axe
 4", main="Représentation des 20 variables \n les mieux
 reconstituées \n sur les axes 3 et 4")
lines(seq(-1,1,length=1000),-sqrt(1-seq(-1,1,length=1000)^2))
arrows(0,0,xPCA$var$cor[sel34,3],xPCA$var$cor[sel34,4],length=0.05
 )
```

```
text(xPCA$var$cor[sel34,3],xPCA$var$cor[sel34,4],colnames(x)
 [sel34],cex=0.7,col="sienna")

#### Axes 5 et 6
# sélection
axes56.costot <- xPCA$var$cos2[,5]+xPCA$var$cos2[,6]
order.costot.56 <- order(axes56.costot,decreasing=T)
sel56 <- order.costot.56[1:20]
# représentation
plot(seq(-1,1,length=1000),sqrt(1-seq(-
 1,1,length=1000)^2),type="l",ylim=c(-1,1), xlab="Axe 5", ylab="Axe
 6", main="Représentation des 20 variables \n les mieux
 reconstituées \n sur les axes 5 et 6")
lines(seq(-1,1,length=1000),-sqrt(1-seq(-1,1,length=1000)^2))
arrows(0,0,xPCA$var$cor[sel56,5],xPCA$var$cor[sel56,6],length=0.05
 )
text(xPCA$var$cor[sel56,5],xPCA$var$cor[sel56,6],colnames(x)
 [sel56],cex=0.7,col="sienna")

#### Axes 7 et 8
# sélection
axes78.costot <- xPCA$var$cos2[,7]+xPCA$var$cos2[,8]
order.costot.78 <- order(axes78.costot,decreasing=T)
sel78 <- order.costot.78[1:20]
# représentation
plot(seq(-1,1,length=1000),sqrt(1-seq(-
 1,1,length=1000)^2),type="l",ylim=c(-1,1), xlab="Axe 7", ylab="Axe
 8", main="Représentation des 20 variables \n les mieux
 reconstituées \n sur les axes 7 et 8")
lines(seq(-1,1,length=1000),-sqrt(1-seq(-1,1,length=1000)^2))
arrows(0,0,xPCA$var$cor[sel78,7],xPCA$var$cor[sel78,8],length=0.05
 )
text(xPCA$var$cor[sel78,7],xPCA$var$cor[sel78,8],colnames(x)
 [sel78],cex=0.7,col="sienna")

#Intersections croisées des variables sélectionnées
intersect(rownames(xPCA$var$cos2)[sel12],rownames(xPCA$var$cos2)
 [sel34])
intersect(rownames(xPCA$var$cos2)[sel12],rownames(xPCA$var$cos2)
 [sel56])
intersect(rownames(xPCA$var$cos2)[sel12],rownames(xPCA$var$cos2)
 [sel78])
intersect(rownames(xPCA$var$cos2)[sel34],rownames(xPCA$var$cos2)
 [sel56])
intersect(rownames(xPCA$var$cos2)[sel34],rownames(xPCA$var$cos2)
 [sel78])
intersect(rownames(xPCA$var$cos2)[sel56],rownames(xPCA$var$cos2)
 [sel78])
```

Forêts Aléatoires

```
#chargement des données corrigées de l'effet bande
load("corrected.Rdata")

#### Forêt 1 de prédiction de la race (données initiales)
## Apprentissage de la forêt aléatoire
# Paramétrage de la graine aléatoire
set.seed(47395)
# Construction de la forêt
best.RF.race <-
randomForest(x.correct,x.notDchn$race,ntree=500,mtry=100,sampsize=
75,importance=T)

## Analyse des performances
# Table de contingence
TC <- best.RF.race$confusion
TC
# Taux d'erreur global
best.err <- 1-sum(diag(TC[,1:4]))/sum(TC[,1:4])
best.err
# Evolution de l'erreur
plot(1:nrow(best.RF.race$err.rate),best.RF.race$err.rate[,1],col="
 black", xlab="Nombre d'arbres", ylab="Taux de mal prédits",
 main="Évolution du taux de mal prédits en fonction du nombre
 d'arbres",lwd=2,type="l",ylim=c(0,1))
for (ind in 2:5)
 {
lines(1:nrow(best.RF.race$err.rate),best.RF.race$err.rate[,ind],co
 l=rainbow(5)[ind-1],lty=2)
 }
legend(x=3500,y=0.75,lwd=2,lty=c(1,2,2,2,2),col=c("black",rainbow(
 5)),legend=c("Global",rownames(TC)))

## Analyse des variables importantes
# Tri par ordre décroissant pour les deux critères d'importance
importanceord.Acc <- sort(best.RF.race$importance[,5],
 decreasing=T)
importanceord.Gini <- sort(best.RF.race$importance[,6],
 decreasing=T)
par(mfrow=c(1,2))
barplot(importanceord.Acc,names=rep(NA,length(importanceord.Acc)),
 ylab="importance (précision)",xlab="genes")
barplot(importanceord.Gini,names=rep(NA,length(importanceord.Gini)
 ),ylab="importance (Gini)",xlab="genes")
dev.print(jpeg, file="RF.race.importance.gini.precision1.jpg",
 width=500)
# Extraction des variables importantes
best.var.acc <- importanceord.Acc[1:20]
best.var.gini <- importanceord.Gini[1:20]
```

```
#Longueur de l'intersection
length(intersect(names(best.var.acc),names(best.var.gini)))
best.var <- intersect(names(best.var.acc),names(best.var.gini))
write.table(cbind(best.var,round(best.var.acc[best.var],3),round(b
 est.var.gini[best.var],3)),file="RFrace.19genes.plusimportants.csv
 ",row.names=F,col.names=F,sep=";",dec=",")

# JDD avec les 20 genes les plus importants
imp.and.race <- data.frame(x.notDchn$race,x.notDchn[,best.var])
rownames(imp.and.race) <- paste(x.notDchn$race,1:108,sep=".")
imp.and.race.ordered<-imp.and.race[order(imp.and.race[,1]),]
imp.and.race.ordered

#Heatmap des 20 genes les plus importants
heatmap(as.matrix(imp.and.race.ordered[,2:20]),Rowv=NA)

#Matrice de boxplots de l'expression des gènes les plus importants
selon la race
par(mfrow=c(4,5))
for (i in 1:19)
{
 boxplot(
x.data.best.var.race[,i]~x.notDchn$race,
ylab=dimnames(x.data.best.var.race)[[2]][i],
 las=2
 )
}

##Comparaisons avec les données issues du modèle mixte :
GenesMM<-read.table("projet-stat-etienne-nicolas/genes-de-
 races.csv", sep=";")
intersect(GenesMM[,1], best.var)



#### Forêt 2 de prédiction de la race (données avec LWM et LWF
regroupées)


###### Creation de x.LWFM, où l'on a, par rapport à x.notDchn,
« fusionné » les races LWF et LWM en LW

x.LWMF<-x.notDchn
x.LWMF$race<-as.character(x.LWMF$race)
x.LWMF$race[x.LWMF$race=="LWF"]<-"LW"
x.LWMF$race[x.LWMF$race=="LWM"]<-"LW"
x.LWMF$race<-as.factor(x.LWMF$race)

x.LWMF.genes<-x.LWMF[,1:4674]
```

```
## Forêt Aléatoire
set.seed(33)

RF.LWMF.race <-
randomForest(x.LWMF.genes,x.LWMF$race,ntree=500,mtry=100,sampsize=
75,importance=T)

## Analyse des performances
TC <- RF.LWMF.race$confusion
TC
# Taux d'erreur global
best.err <- 1-sum(diag(TC[,1:4]))/sum(TC[,1:4])
best.err
plot(1:nrow(RF.LWMF.race$err.rate),RF.LWMF.race$err.rate[,1],lwd=2
,type="l", xlab="Nombre d'arbres", ylab="Erreur")
# Evolution de l'erreur
plot(1:nrow(RF.LWMF.race$err.rate),RF.LWMF.race$err.rate[,1],col="
black", xlab="Nombre d'arbres", ylab="Taux de mal prédits",
main="Évolution du taux de mal prédits en fonction du nombre
d'arbres",lwd=2,type="l",ylim=c(0,1))
for (ind in 2:4)
 {
lines(1:nrow(RF.LWMF.race$err.rate),RF.LWMF.race$err.rate[,ind],co
 l=rainbow(5)[ind-1],lty=2)
 }
legend(x=3500,y=0.75,lwd=2,lty=c(1,2,2,2),col=c("black",rainbow(5)
 [1:4]),legend=c("Global",rownames(TC)))

##importance
importanceord.Acc <- sort(RF.LWMF.race$importance[,4],
 decreasing=T)
importanceord.Gini <- sort(RF.LWMF.race$importance[,5],
 decreasing=T)
par(mfrow=c(1,2))
barplot(importanceord.Acc,names=rep(NA,length(importanceord.Acc)),
 ylab="importance (précision)",xlab="genes")
barplot(importanceord.Gini,names=rep(NA,length(importanceord.Gini)
 ),ylab="importance (Gini)",xlab="genes")




#Sortie des variables les plus importantes
cbind(best.var,round(best.var.acc[best.var],3),round(best.var.gini
 [best.var],3))
```

```
###Comparaison avec les genes issus du modèle mixte
length(GenesMM[,1])
intersect(GenesMM[,1], best.var)
#Création d'un JDD contenant uniquement les gènes importants (et
 la race)
ImpRace.best.LWMF <- data.frame(x.LWMF$race,x.LWMF[,best.var])
rownames(ImpRace.best.LWMF) <- paste(x.LWMF$race,1:108,sep=".")
ImpRace.best.LWMF.ordered<-
 ImpRace.best.LWMF[order(ImpRace.best.LWMF[,1]),]

#heatmap
heatmap(as.matrix(ImpRace.best.LWMF.ordered[2:
 (ncol(ImpRace.best.LWMF.ordered))]), Rowv=NA)

# JDD des expressions moyennes des genes par race
mean.exp.ImpRace <-
matrix(ncol=3,data=unlist(by(ImpRace.best.LWMF[,2:17],ImpRace.best
.LWMF[,1],mean)))
colnames(mean.exp.ImpRace) <- c("Duroc","Landrace","LW")
rownames(mean.exp.ImpRace) <- colnames(ImpRace.best.LWMF[,2:17])
#heatmap
heatmap(mean.exp.ImpRace,Colv=NA)

#2e méthode de création d'un JDD de variables importantes
mostimport.x.LWMF<-x.LWMF[best.var]
dim(mostimport.x.LWMF)

# Corrélation entre les gènes
library(ellipse)
library(RColorBrewer)
cor.mat <- cor(mostimport.x.LWMF)
ord <- order(cor.mat[1,])
xc <- cor.mat[ord,ord]
colors <- brewer.pal(11,"RdBu")
colors
xc
plotcorr(xc,col=colors[5*xc + 6])

#Boucle de mini-barplots parallèles
par(mfrow=c(4,4))
for (i in 1:16)
{
 boxplot(mostimport.x.LWMF[,i]~x.LWMF$race,
 ylab=paste("Expr. du gene",dimnames(mostimport.x.LWMF)[[2]][i]),
 las=2
 )
}
```

```
#### Forêt 1 de prédiction du phénotype corrigé (données
initiales)
## Apprentissage de la forêt aléatoire
# Paramétrage de la graine aléatoire
set.seed(47395)
best.rf.carc <-
 randomForest(x.correct[ind.not.na,],x.notDchn$lgr.carc.r[ind.not.n
 a],ntree=5000,mtry=100,sampsize=50,importance=T)

## Analyse des performances
# pseudo-R²
ps.r.sq<- 1-mean((best.rf.carc$predicted-
 x.notDchn$lgr.carc.r[ind.not.na])^2)/var(x.notDchn$lgr.carc.r[ind.
 not.na])
ps.r.sq
# Évolution de l'erreur
plot(1:length(best.rf.carc$mse),best.rf.carc$mse,lwd=2,col="darkre
 d",main="Évolution de l'erreur (OOB MSE)\n en fonction du nombre
 d'arbres",xlab="Nombre d'arbres",ylab="OOB MSE",type="l")

# Valeurs prédites vs valeurs réelles
plot(x.notDchn$lgr.carc.r[ind.not.na],best.rf.carc$predicted,pch="
 +",lwd=2,col="darkred",main="Valeurs prédites vs valeurs
 réelles",xlab="Longueur de carcasse",ylab="Valeurs prédites")
abline(0,1,col="darkgreen")
# Étude des variables importantes
# Tri par ordre décroissant pour les deux critères d'importance
importanceord.Acc <- sort(best.rf.carc$importance[,1],
 decreasing=T)
importanceord.Gini <- sort(best.rf.carc$importance[,2],
 decreasing=T)
par(mfrow=c(1,2))
barplot(importanceord.Acc,names=rep(NA,length(importanceord.Acc)),
 ylab="importance (précision)",xlab="genes")
barplot(importanceord.Gini,names=rep(NA,length(importanceord.Gini)
 ),ylab="importance (Gini)",xlab="genes")
dev.print(jpeg, file="RF.carc.r.importance.gini.precision1.jpg",
 width=500)

# Extraction des variables importantes
best.var.acc <- importanceord.Acc[1:20]
best.var.gini <- importanceord.Gini[1:20]
length(intersect(names(best.var.acc),names(best.var.gini)))

best.var <- intersect(names(best.var.acc),names(best.var.gini))
write.table(cbind(best.var,round(best.var.acc[best.var],3),round(b
 est.var.gini[best.var],3)),file="RFcarc.14genes.plusimportants.csv
```

```
",row.names=F,col.names=F,sep=";",dec=",")

# JDD fait pour les expressions des genes les plus importants
imp.and.race <-
 data.frame(x.notDchn$lgr.carc.r[ind.not.na],x.notDchn[,best.var]
 [ind.not.na,])
imp.and.race.ordered<-imp.and.race[order(imp.and.race[,1]),]
imp.and.race.ordered
#Heatmap
heatmap(as.matrix(imp.and.race.ordered[,2:12]),Rowv=NA)

# Matrice de boxplots de l'expression des gènes les plus
 importants selon le phénotype corrigé
par(mfrow=c(3,4))
for (i in 1:11)
{
 plot(
x.notDchn$lgr.carc.r[ind.not.na],x.data.best.var.carc.r[,i],
ylab=paste("Gene",dimnames(x.data.best.var.carc.r)[[2]][i]),
xlab="Obs. du phénotype corrigé",
 pch="+")
}
```