INRA de Toulouse Midi-Pyrénées
24 Chemin de Borde Rouge
BP 52627
31326 Castanet Tolosan cedex

Rapport de stage en entreprise

Analyse exploratoire des caractéristiques du génome et du transcriptome de *Lactococcus lactis*

Wiklund Béatrice

Stage de validation de Licence Statistique et Informatique Décisionnelle
Effectué à l'INRA au sein de l'unité BIA du 2 avril au 2 juillet 2012

Référent de stage : Jérôme Farinas
Tutrices de stage : Annick Moisan, Christine Cierco, Christine Gaspin

Remerciements

Je remercie Annick Moisan, Christine Gaspin, Nathalie Villa-Vialaneix ainsi que Christine Cierco, mes tutrices de stage pour leur aide et leur gentillesse ainsi que pour le temps qu'elles m'ont consacré tout au long de cette période. Je aussi remercie Jérôme Farinas, mon référent de stage, ainsi que toute l'équipe du BIA pour son accueil chaleureux.

Introduction

Actuellement étudiante en troisième année de licence en Statistique et Informatique Décisionnelle, j'ai effectué mon stage de fin d'année à l'unité Biométrie et Intelligence Artificielle du département Mathématiques et Informatique Appliquées de l'INRA de Toulouse. Ces trois mois au sein du laboratoire m'ont permis d'obtenir une nouvelle expérience dans le domaine de la recherche, faisant ainsi un pont entre ma formation actuelle et le DUT Génie Biologique obtenu deux ans plus tôt.

Ayant effectué l'année précédente un stage en informatique dans le domaine agro-alimentaire, je souhaitais faire un stage plutôt orienté Statistique dans un domaine différent, mais toujours en rapport avec ma formation initiale. Le stage au sein de l'INRA a pu me faire découvrir un aspect de la recherche scientifique différent de la recherche en milieu médical (par opposition à mon stage de DUT Génie Biologique, effectué en milieu hospitalier), tout en approfondissant les connaissances acquises ces deux dernières années.

Le principal but de ce stage était de réaliser l'étude exploratoire d'un jeu de données d'origine génétique à l'aide du logiciel R. Cette analyse s'inscrit dans le projet DEGRADOMICS ayant pour but l'étude de la dégradation de l'ARN dans les bactéries lactiques. En plus de cette étude, j'ai réalisé un script Python ayant pour but de rassembler et traiter les informations de sorties d'outils logiciels de segmentation d'un signal.

Je présenterai d'abord dans ce rapport l'Institut National de Recherche Agronomique dans la partie présentation, puis l'étude statistique, avec la présentation du projet DEGRADOMICS ainsi que les méthodes utilisées au cours de l'étude dans la deuxième partie. Ensuite, nous verrons la partie programmation en Python et enfin un bilan de ces 12 semaines passées au sein de l'INRA.

Sommaire

1 Présentation du lieu de stage

1.1 L'Institut National de Recherche Agronomique

L'Institut National de Recherche Agronomique (INRA) est un organisme de recherche scientifique publique dépendant à la fois du ministère de l'Enseignement supérieur et de la Recherche et du ministère de l'Alimentation, de l'Agriculture et de la Pêche. Il est constitué de 14 départements scientifiques, répartis sur 19 centres de recherche régionaux.

Créé en 1946, l'INRA a accompagné les nombreuses mutations du monde agricole ainsi que des filières alimentaires et territoriales, ayant pour objectif de répondre aux attentes exprimées par la société, notamment la suffisance alimentaire française. Depuis la reconstruction nationale d'après-guerre, les défis scientifiques et sociétaux ont beaucoup évolués, se tournant entre autres vers la chimie verte, l'érosion de la biodiversité et les maladies émergentes.

L'institut a pour mission de :
- Produire et diffuser des connaissances scientifiques
- Concevoir des innovations et des savoir-faire pour la société
- Eclairer les décisions des acteurs publics et privés
- Développer la culture scientifique et technique ainsi que participer au débat science/société
- Former à la recherche et par la recherche

Pour accomplir ses objectifs, l'INRA a de nombreux partenariats avec des acteurs socio-économiques (entreprises, organisations collectives agricoles), les collectivités territoriales ainsi que les pouvoirs publics à qui il offre son expertise.

Historique

1946 : Création de l'INRA

1951 : Installation de la Station d'Agronomie et de Technologie Végatale à Toulouse

1970 : Implantation à Saint Martin du Touch et construction du site d'Auzeville

1981 : Création d'une équipe mixte avec le CNRS

1988 : Construction du pôle de biotechnologies végétales qui permet d'accueillir sur le campus des équipes de l'université (UPS/CNRS) et de l'école d'agronomie de Toulouse (ENSAT).

2004 : Installation du Centre National de Ressources en Génomique Végétale (CNRGV)

2006 : Labellisation d'une unité mixte technologique (UMT) pour renforcer le partenariat entre l'INRA et l'ENSAT (UMR AGIR) d'une part et le Cetiom d'autre part sur le thème de l'amélioration de la production d'huile à l'échelle des bassins de collecte.

1.2 Le centre de recherche

Le centre de recherche INRA de Toulouse Midi-Pyrénnées est l'un des 19 centres de l'INRA répartis sur l'ensemble du territoire national, et se situe à Castanet-Tolosan. Il se situe parmi les 5 plus grands centres en dehors de la région parisienne. Créé en 1970, il emploie maintenant 600 agents, dont 250 chercheurs et connaît une croissance soutenue depuis ces 15 dernières années.

Les recherches effectuées au sein du centre s'articulent autour de 19 unités de recherche (dont 14 en partenariat avec d'autres établissements d'enseignement supérieur et de recherche), 5 unités expérimentales, 3 unités d'appui à la recherche et une unité de service.
Le centre est structuré autour de 5 domaines de recherches :

- Génome et amélioration des productions
- Sécurité sanitaire des aliments
- Transformation des produits agricoles
- Économie de l'environnement et des marchés
- Environnement, territoire et société

1.3 Unité de Biométrie et d'Intelligence Artificielle

Unité BIA Toulouse
Biométrie et Intelligence Artificielle

L'UBIA a pour mission de mettre à la disposition de l'INRA des méthodes et compétences à jour en mathématiques et informatique appliquée, en particulier dans le cadre de collaborations inter-départements. Les compétences présentes dans l'unité couvrent un large spectre en Statistique, Probabilité, Algorithmique, Intelligence Artificielle et Sciences de la Décision. L'UBIA est divisée en deux équipes de recherche, SaAB (Statistique et Algorithmique pour la Biologie) et MAD (Modélisation des Agro-écosystèmes et Décision) ainsi que deux équipes plate-forme, Genotoul et Record. J'ai effectué mon stage au sein de l'équipe SaAB, qui a pour domaine de recherche

principal la bioinformatique.

Actuellement, les deux domaines principaux sur lesquels travaille l'équipe de l'unité de Biométrie et d'Intelligence Artificielle sont intitulés « Génome et Biotechnologie » et « Territoire et produits ». Mon stage s'inscrivait dans la première thématique, dont le but est l'étude de l'organisation et du fonctionnement de gènes d'animaux et de végétaux. Parallèlement à l'étude de ces gènes, l'UBIA développe des outils bioinformatique nécessaires à ces approches génomiques. Parmi ces outils informatiques, des logiciels à destination des chercheurs sont développés, tels que Toulbar2, RNAspace (plate-forme d'annotation d'ARN non codant) ou encore Eugène (prédiction de gènes) etc.

Organigramme de l'unité BIA

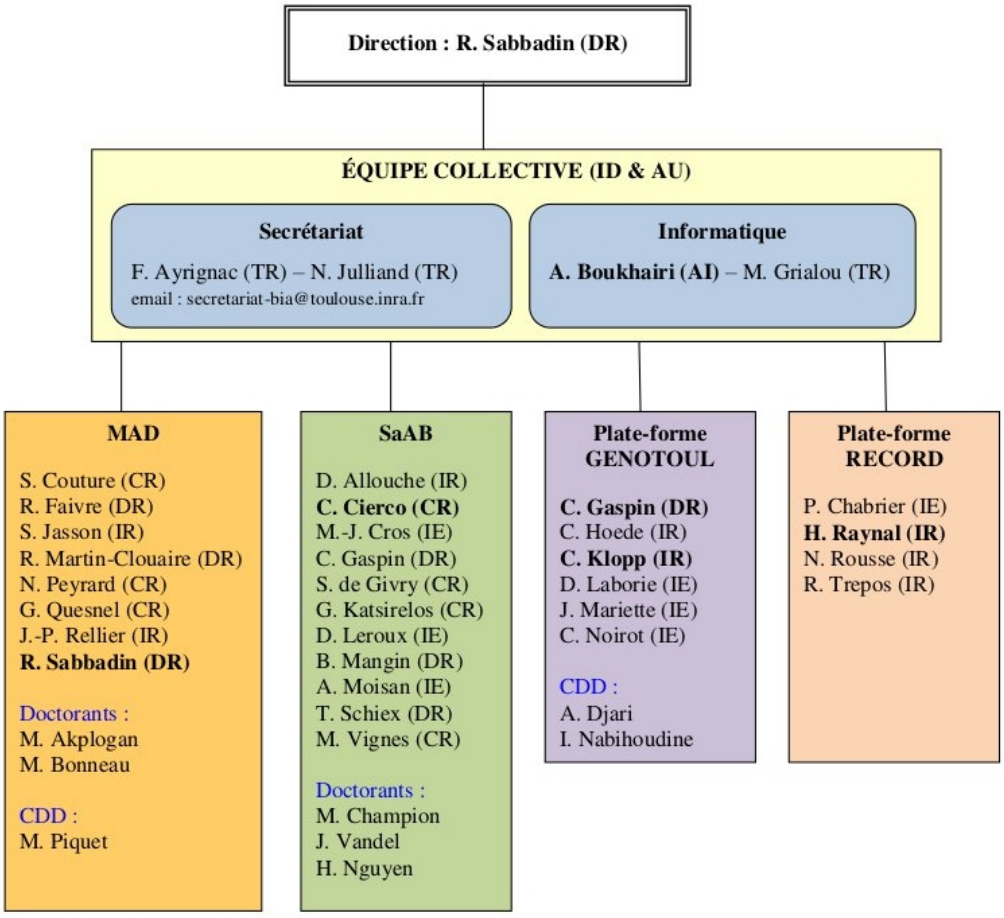


FIGURE 1.1 – Organigramme de l'unité de Biométrie et d'Intelligence Artificielle

2 Travail effectué lors du stage

2.1 Organisation du travail

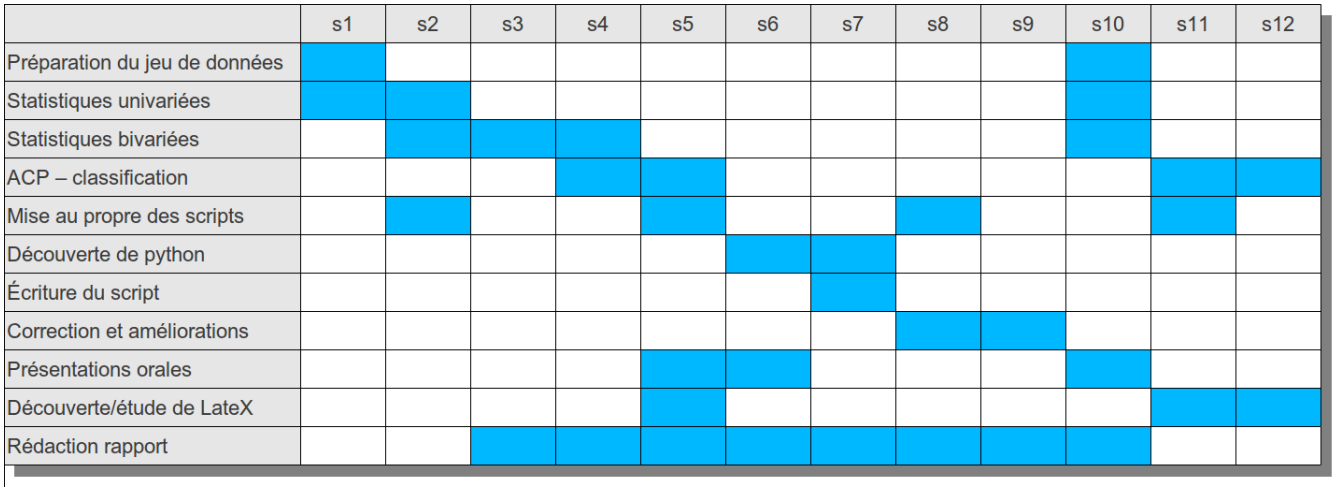


FIGURE 2.1 – Organisation du temps de travail

Les deux premières semaines, les données ont été regroupées. Elles sont arrivées en plusieurs jeux différents qui ont été regroupés puis étudiés. Des points réguliers ont été faits tout au long de l'étude statistique, notamment pour valider les transformations de variables et sélectionner les variables pertinentes à garder pour l'ACP. Régulièrement, les scripts ont été mis au propre et commentés pour pouvoir être réutilisés. Plusieurs présentations orales ont également eu lieu devant mes tutrices de stage, les biologistes de l'INSA ayant commandé l'étude ainsi que l'ensemble du département BIA lors d'une journée spéciale.

En attendant le retour des biologistes de l'INSA, j'ai étudié le langage Python puis conçu un script créant un fichier GFF d'après des sorties de séquenceurs, script qui a ensuite été amélioré. Le rapport a lui été rédigé tout au long du stage, tout d'abord sous LibreOffice, puis sous LaTeX, acquis progressivement.

2.2 Analyse d'un jeu de données

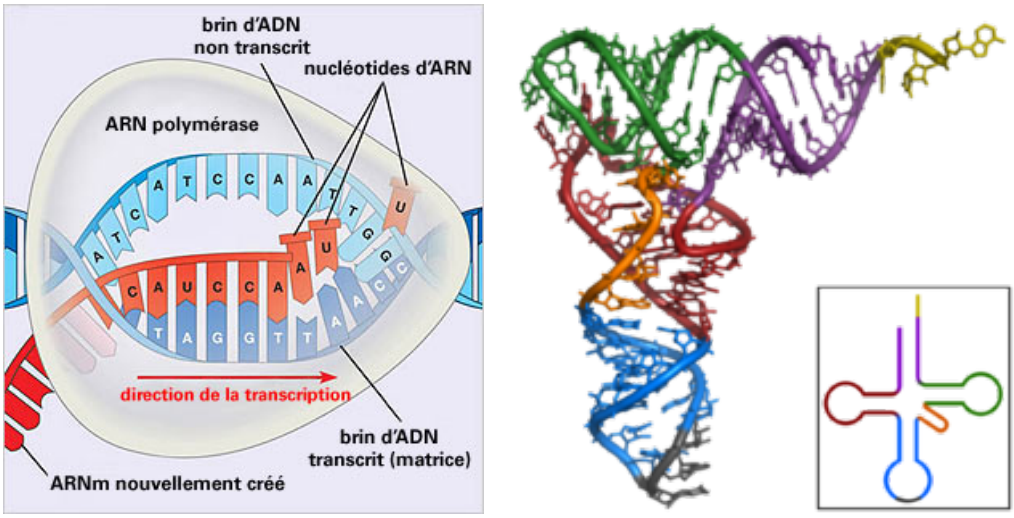
2.2.1 Présentation du projet DEGRADOMICS et du jeu de données

2.2.1.1 Notions de biologie

Gène

Un gène est un élément génétique correspondant à un segment d'ADN ou d'ARN (virus), situé à un endroit bien précis (locus) sur un chromosome. Chaque région de l'ADN qui produit une molécule d'ARN fonctionnelle est un gène, soit une unité d'hérédité contrôlant un caractère particulier.

ARN



(a) Transcription de l'ARN (b) Structure ARN de transfert

FIGURE 2.2 – L'ARN dans la cellule

Les ARN sont des molécules constituées par l'assemblage de ribonucléotides (adénine, cytosine, guanine, uracile) reliés entre eux par des liaisons nucléotidique et qui possèdent de très nombreuses fonctions dans la cellule.

L'ordre de cet enchaînement (structure primaire) est dicté par la séquence des désoxyribonucléotides portés par l'ADN dont il est issu. En effet, l'ARN provient de la transcription de l'ADN par une enzyme (l'ARN polymérase) qui recopie en quelque sorte la séquence.

Une manière de classer les ARN est de les séparer en ARN codants et ARN non codants :

Un ARN non codant (ou ARNnm pour ARN non-messager) est un ARN, issu de la transcription de l'ADN, qui ne sera pas traduit en protéine par les ribosomes. L'ARN codant est, à l'inverse, un ARN traduit en protéine.

Contrairement à l'ADN qui est la plupart du temps structuré en double hélice par l'appariement des deux brins complémentaire, l'ARN (qui est simple brin) peut adopter des conformations très différentes, étroitement liées à sa fonction. Ainsi, certaines molécules d'ARN sont simple brin, en tige boucle (grâce à un appariement des bases complémentaires), en feuille de trèfle (l'ARN de transfert), etc.

Il existe de nombreuses familles d'ARN (ARNr, ARNm, ARNt, ARNsi, ARNmi, snARN...), dont chacune possède une structure ou une fonction particulière :

- les ARN messagers (ARNm) serviront de matrice pour la synthèse des protéines ;
- les ARN ribosomiques (ARNr) entrent dans la composition des ribosomes, avec les protéines ribosomiques ;
- les ARN de transfert (ARNt) portent des acides aminés et permettent leur incorporation dans les protéines ;
- les ARN interférents (ARNsi, ARNmi...) régulent l'expression des gènes en ciblant la dégradation des ARN messagers spécifiques ou en inhibant la traduction des protéines.

Dans l'étude effectuée, il s'agit de l'ARN de transfert qui est étudié.

2.2.1.2 Le projet DEGRADOMICS

La dégradation des molécules d'ARN est l'un des mécanismes, avec la transcription des gènes, qui contribue à la régulation de la quantité d'ARN dans la cellule. C'est en cela un facteur de régulation de l'expression génique. Si les régulations transcriptionnelles sont très largement étudiées, un manque de connaissance du contrôle dégradationnel des ARN et de leur rôle *in vivo* dans l'adaptation des bactéries persiste.

Le projet Degradomic, coordonné par le Laboratoire de Biotechnologie-Bioprocédés de l'INSA de Toulouse, a pour objectif d'étudier la dégradation *in vivo* des ARN chez la bactérie lactique. Dans le but de ce projet, la contribution de l'UBIA consiste à développer des méthodes statistiques et informatiques afin d'intégrer l'ensemble des données de natures diverses en vue de l'inférence de réseaux de régulation. Le projet DEGRADOMICS s'incrit donc dans la recherche fondamentale et vise à répondre à deux objectifs :

- Développer une méthode générique pour étudier *in vivo* sur les mécanismes de dégradation de l'ARN ainsi que leur régulation
- Mesurer l'influence du processus de dégradation de l'ARN dans la résistance bactérienne

Pour cela, l'activité de la ribonucléase (ou ARNase, une nucléase qui catalyse la dégradation de l'ARN) sera analysée *in vivo* avec une nouvelle technique de marquage pouvant différencier les extrémités d'ARN dégradées et non dégradées. Puis ces résultats seront confirmés *in vitro* et différentes conditions de dégradation seront testées.

Trois mécanismes de régulation seront étudiés :

- Les mécanismes *in vivo* de protection du ribosome contre les ARNases
- Le rôle des structures secondaires de l'ARN messager
- Le rôle des ARN non codants

L'étape finale sera la contruction d'un modèle de régulation de l'ARN incluant les données de DEGRADOMICS permettant d'identifier les facteurs de stabilités de l'ARN. enfin, ces facteurs seront validés expérimentalement. Les bactéries utilisées pour ce projet sont des bactéries productrices d'acide lactique, communément utilisées dans le domaine alimentaire et de la santé.

Le sujet de mon stage se situe au début du projet DEGRADOMICS. Il a pour but l'intégration et l'analyse exploratoire de données de différentes natures provenant de bactéries dont l'ARN n'a pas

été dégradé. Il ne s'agit pas de répondre à une problématique mais de livrer une première analyse qui sera ensuite comparée par la suite aux résultats d'un autre jeu de données dont l'ARN aura été dégradé : l'évolution des liens entre variables pourra permettre d'expliquer leur importance dans la dégradation de l'ARN.

2.2.1.3 Présentation de la bactérie utilisée

Parmi les bactéries utilisées dans l'industrie agro-alimentaire, les bactéries productrices d'acide lactique ont un impact économique important sur la production de produits fermentés. En France, la production laitière est de 23.3 millions de tonnes et les industries fromagères sont autour de 600. La tendance actuelle est aux aliments incluant des probiotiques, ce que sont les bactéries homofermentaires. Certaines souches sont aussi employées aujourd'hui pour la production de protéines à haute valeur ajoutée, principalement dans le domaine de la santé.
De toutes ces bactéries, lactococcus lactis est considéré comme l'une des meilleures souches pour des applications alimentaires et liées à la santé.

FIGURE 2.3 – *Lactococcus lactis*

Lactococcus lactis est une bactérie très utilisée dans l'industrie laitière, notamment dans l'industrie fromagère. Elle se divise en deux sous-espèces, *Lactococcus lactis subsp. lactis* et *Lactococcus lactis subsp. Cremoris* qui peut être aussi appelée *Streptococcus lactis*. C'est une est une bactérie homofermentaire (fermentation produisant uniquement de l'acide lactique), hétérotrophe (ne produit pas lui-même tous ses constituants), anaérobie-facultatif. Sa température optimale de croissance est de 30 °C. Au microscope, elle se présente sous la forme de petites sphères non mobiles, souvent par paires (voir figure 2.3).

2.2.1.4 Présentation du jeu de données

Il s'agit de données extraites de *Lactococcus lactis* concernant l'ensemble des gènes protéiques de cette bactérie. Le jeu de données comporte 32 variables et 2126 observations, chaque observation correspondant à un gène différent. Il s'agit de données d'ARN non dégradé s'inscrivant dans la première phase du projet DEGRADOMICS, ce premier jeu de données ayant été fourni par le laboratoire de l'INSA. Elles seront ensuite comparées à d'autres données d'ARN ayant subit une dégaradation plus tard dans l'année.

Le but de cette étude est de trouver des liens entre les variables en vue de la comparaison avec des données de bactéries ayant subi une dégradation de l'ARN dans la cellule. Pour cela, j'ai fait d'abord des statistiques simples, puis bivariées, et enfin une ACP et une classification. Les variables qu'on cherche à étudier sont les CDS (moyennes et écart-types de l'expression du gène) ainsi que les CAI (Index d'Adaptation Carbone) et tAI (Index d'adaptation à l'ARN de transfert).

Le jeu de données comporte 32 variables dont 28 quantitatives et 4 qualitatives. Ces données sont de nature et de provenance diverses et sont liés à différents concepts :

– Codage des protéines et usage des codons : Nc, tAI, CAI, CAI.rib, CAI31, tAI(faux), Nc th/re, Nc.th.

- Caractéristiques des protéines codées : Cat, Localisatio, Aromaticité, Hydrophobie.
- ADN : longueur, GC, signe.
- Mesures de degrés de structuration de l'ARN (variables calculées) : DeltaG, énergie_moyenne, z_score moyen.
- Données de transcription : cdsCountMean, cdsCountSD, cdsBruteMean, cdsBruteSD

Le jeu de données a été étudié à l'aide du logiciel R, avec l'interface Rstudio.

Tableau des variables

Variables	Signification
Gene id	Nom du gène
Position 5'	Position du gène sur le brin ; correspond au début de la séquence codante
AGGAG	Nombre de motifs AGGAG présents dans la région promotrice du gène
Cat	Catégorie fonctionnelle du gène
Lgr CDS(bp)	Longueur du gène
Ttotal	Nombre de thymine
Ctotal	Nombre de cytosine
Atotal	Nombre d'adénine
Gtotal	Nombre de guanine
Nc	Nombre de codons
Nc théorique	Nombre de codons théoriques
Nc/th/réel	Quotient du nombre de codons
CAI31	Index de carbon adaptation
CAI rib	Index de carbon adaptation dans le ribosome
CAI	Correspond à la mesure de l'inverse de Nc
tAI	Index d'adaptation à l'ARN de transfert
tAI(FAUX)	Idem tAI mais avec un autre calcul
Localisation	Localisation dans la cellule de la protéine résultante
Hydrophobicity	Hydrophobie
Aromaticity	Aromaticité
DeltaGdow-24+30	Mesure du biais de composition nucléé en aval de la séquence du gène
DeltaGdow-24+100	Mesure du biais de composition nucléé en aval de la séquence du gène
DeltaGup-30+24	Mesure du biais de composition nucléé en amont de la séquence du gène
DeltaGup-100+24	Mesure du biais de composition nucléé en amont de la séquence du gène
DeltaGout	Somme des delta
Deb	Position de début du gène sur l'ADN
Fin	Position de fin du gène
signe	Brin sur lequel se trouve le gène
cdsBruteMean	Moyenne brute de l'expression du gène (non déplété)
cdsBruteSD	Ecart-type brut de l'expression du gène (non déplété)
cdsCountMean	Moyenne de l'expression du gène (déplété)
cdsCountSD	Ecart-type de l'expression du gène (déplété)

2.2.2 Méthodes et outils

2.2.2.1 Documents de référence et packages utilisés

Cette analyse exploratoire a été effectuée à l'aide du logiciel R pour linux, avec l'interface RStudio.

Les packages utilisés sont les suivants :

- Rcmdr
- FactomineR
- ellipse
- RcolorBrewer
- abind
- Gtools
- ClustOfVar
- Mixtools
- RColorBrewer

Les documents utilisés pour ce projet sont les suivants :

- Cours de data-mining de Sylvie Viguier-Pla
- Cours sur les tests statistique de Xavier Gendre
- Cours de manipulation de R de Nathalie Villa-Vialaneix, Jean-Noël Kien et Jérôme Gans
- www.wikipedia.org (définitions)
- Conférence Agilité et recherche du 13 juin 2012 - INRA
- http ://www.aubryconseil.com (site de présentation de la méthode SCRUM)

2.2.2.2 Démarche

Statistiques simples

Il s'agissait d'analyser chaque variable pour repérer les valeurs aberrantes, sa répartition, ainsi qu'une éventuelle loi suivie. Le but principal était de visualiser les variables, mais j'y ai ajouté quelques tests supplémentaires. Certaines variables étant calculées, une première vue a permis de déterminer si elles n'étaient pas pertinentes. Cette analyse systématique a permis de repérer quelques anomalies ainsi que des variables doublons. Pour les variables quantitative, cette analyse comporte les quantiles, moyenne, écart-type, test de Shapiro-Wilks ainsi que quatre graphiques : Un graphique simple, un boxplot, un histogramme et un QQplot comparant la courbe de la variable à la loi normale. Pour les variables qualitatives, la répartition des effectifs et des graphiques en camembert ont été sortis.

Statistiques bivariées

Croisement des variables qualitatives

Le but de cette analyse est de mettre en évidence les intéractions entre variables qualitatives.

Pour chacune des quatre variables qualitatives, trois diagrammes en barres ont été générés automatiquement en croisant la variable choisie avec toutes les autres. Cela permet de voir la répartition selon les effectifs et la deuxième variable en un seul coup d'œil. Un graphique en

barres empilées a aussi été envisagé mais abandonné pour des raison de visibilité (par ailleurs, des graphiques de ce type ont été générés pour l'analyse qualitative-quantitative). Pour compléter l'analyse des graphiques, j'ai ajouté un test du χ^2 d'ajustement et des tableaux de contingence en ligne et colonne à l'aide du package Rcmdr.

Ces derniers ont permis de voir les répartitions des modalités de manière plus précise lorsque le test du χ^2 était signficatif. Ces tableaux de contingence n'ont pas été automatisés. Une fonction faisant un test d'ajustement du χ^2 par modalité (fonction chi2) a ensuite été créé pour savoir quelles modalités étaient réellement influencées par les variables, ce qui c'est révélé utile notamment pour les catégories fonctionnelles et le signe.

Croisement des variables qualitatives et quantitatives

Le but de ce croisement est de mettre en évidence les interactions entre variables et plus particulièrement de repérer quelles variables sont influencées par la présence ou l'absence de la série AGGAG (vectrice de stabilité de l'ARN) ainsi que d'étudier l'influence de la catégorie fonctionnelle.

Tout d'abord, des graphiques ont été effectués : des boxplot de répartition de la variable quantitative en fonction de la variable qualitative pour observer les variations selon les modalités. Un script de test d'égalité des variances puis des moyennes a été créé puis abandonné. Un autre script testant la normalité puis l'égalité des variances puis des moyennes a été utilisé, mais les résultats des tests étaient toujours positifs (intéraction entre les variables), ce qui semblait douteux au vu des graphiques. Un test de Kruskal a ensuite été effectué avec le même résultat. Finalement un dendrogramme reliant les variables a été tracé à l'aide du package ClustOfVar.

Croisement des variables quantitatives

Il s'agissait ici de mettre en évidence les corrélations entre variables mais aussi de repérer quelles variables pouvaient être supprimer pour l'ACP. Il a été d'abord fait un graphique croisant toutes les variables deux à deux pour voir l'allure des courbes, puis une matrice de corrélation imagée avec ellipse et enfin, lorsque c'était pertinent, une régression linéaire simple.

Analyse en Composantes Principales

Une fois les analyses univariées et bivariées effectuées, plusieurs ACP ont été effectuées pour mettre en évidence les variables corrélées entre elles. Il s'agissait d'ACP réduites (variables de différentes natures avec des variances très éloignes). Une fois ces variables mises en forme, le nuage de point des individus a ensuite été colorié selon la variable catégorie fonctionnelle, puis selon AGGAG.

Une première ACP a d'abord été faite avec toutes les variables quantitatives pour voir les corrélations entre les variables. Il s'agissait d'un premier aperçu, c'est pourquoi l'ACP a été interprétée de manière visuelle et non pas de manière formelle. Le but, comme pour les analyse bivariées, était simplement de mettre en évidence des corrélations, des groupements de variables mais aussi des variables à supprimer. Puis, au vu des représentations sur les axes des variables et des individus, il a été décidé de transformer certaines variables et d'en retirer d'autres pour plus de clarté. Cette ACP finale a été interprétée à l'aide des contributions et des cosinus carrés. Elle a aussi été colorisée pour déterminer si les groupements de variables quantitatives étaient liés aux variables qualitatives étu-

diées. Ensuite, comme le nuage n'était pas vraiment lisible, chaque modalité de la variable qualitative a été mise en valeur sur un nuage grisé.

Une fois un résultat satisfaisant obtenu, un script propre et général a été créé pour permettre de refaire facilement des ACP à partir de n'importe quel jeu de données et coloriser selon n'importe quelle variable qualitative.

Classification

Une fois l'ACP effectuée, les gènes ont été regroupés sur critères de l'ACP par le biais d'une classification hiérarchique (ou CAH). Le but était l'étude des classes ainsi créées pour comparer les groupes de gènes et tracer des parallèles entre les variables de l'ACP et AGGAG ainsi que Catégories fonctionnelles.

Concept :

Le but de la classification hiérarchique (CAH) est de classer des individus en groupes ayant un comportement similaire sur un ensemble de variables. On commence par agréger les 2 individus les plus proches. Puis, on continue en agrégeant les éléments (individus ou groupes d'individus) les plus semblables. Ces agrégations sont effectuées 2 à 2. L'algorithme continue jusqu'à ce que l'ensemble des individus se retrouve dans une unique classe, regroupant donc les individus de façon hiérarchique. C'est parce que cette technique part du particulier pour remonter au général qu'elle est dite «ascendante». Chaque classe d'une partition est incluse dans une classe de la partition suivante. Le principal problème de cette méthode hiérarchique consiste à définir le bon critère de regroupement de deux classes, c'est-à-dire trouver une bonne méthode de calcul des distances entre classes.

Le nombre de classes se détermine à l'aide d'un dendrogramme où l'on cherche un compromis entre le nombre de classes, le nombre d'individus dans celles-ci et la variance expliquée. Le dendrogramme, ou arbre hiérarchique, montre les liaisons entre les classes, et la hauteur des branches nous indique leur niveau de proximité.

2.2.2.3 Scripts développés

Les scripts développés se présentent sous la forme de fichier R et ont pour but de pouvoir être utilisés sur d'autres jeu de données. Ils sont commentés de manière à comprendre leur utilisation.

Statistiques univariées

Ce script contient deux fonctions :

quanti : Prend comme paramètre le jeu de données à étudier et crée un autre jeu de données comportant uniquement les variables quantitatives.

graphique : Prend en paramètre le résultat de la fonction quanti (un tableau de données de variables quantitatives) et pour chaque variable enregistre une image contenant quatre graphiques : un graphique simple, un histogramme, un boxplot ainsi qu'un QQplot comparant la variable à la loi normale. En parallèle, les statistiques simples (quantiles, moyenne, écart-type, valeurs manquantes) sont sorties dans le terminal.

Statistiques bivariées

Ce script contient sept fonctions ayant pour but l'analyse bivariée du jeu de données :

quanti : Prend comme paramètre le jeu de données à étudier et crée un autre jeu de données comportant uniquement les variables quantitatives. (même fonction que dans le script univariée)

quali : Prend comme paramètre le jeu de données initial et crée un autre jeu de données comportant uniquement les variables qualitatives.

graphQUALI : Prend en paramètre le jeu de données initial. A partir de la deuxième colonne (pour éviter les identifiants), trace les graphiques croisés des variables qualitatives. Ce sont des diagrammes en barre dont la surface dépend de l'effectif des modalités, colorisés mais non enregistrés.

qualQuant : Prend en paramètre le jeu de données initial à partir de la deuxième colonne et trace les graphiques croisés des variables quantitatives en fonction des variables qualitatives. Les modalités sont représentés sous forme de boxplot. N'enregistre pas.

matrice de corrélation : Fonction prenant en paramètre un tableau de données quantitatives. Trace la matrice des corrélations avec ellipse, séparant celles-ci en cinq classes selon les bornes : -1,-0.7,-0.3,0.3,0.7,1. Il faut faire attention de ne pas avoir de valeurs manquantes dans le tableau passé en paramètre.

Cluster : Fonction prenant en paramètres un jeu de données quantitatif et un jeu de données qualitatif. Trace un dendrogramme. *Eviter de laisser une variable identifiant dans les tableaux d'entrée.*

ACP et classification

Ce script contient les fonction quanti et quali ainsi que

colorisation : Fonction de coloration de l'ACP selon une variable qualitative ; elle prend en paramètre la variable quali du jeu de données initial. Le nuage de l'ACP ensuite tracée sera colorisé selon les modalités de la variable qualitative.

colorisation sélective : Ces deux lignes sont à passer après la fonction de colorisation. La première fait disparaître la modalité passée en paramètre sur le nuage de l'ACP et la deuxième met en valeur la modalité choisie en grisant le nuage.

ACP : Cette fonction prend en paramètre la variable "tableau" créée par la fonction quanti et conçoit une ACP réduite à partir de cette dernière en se basant sur le nombre de variables du tableau. On peut obtenir les valeurs propres, les cosinus carrés, les contributions, et obtenir les graphiques des variables et des individus selon les axes. Couplé à la fonction colorisation, le nuage des individus sera colorisé.

colorTransp : Cette fonction vient après l'ACP. Elle prend en paramètres la variable qualitative voulue, et les deux axes choisis de l'ACP. Pour chaque modalité de la variable qualitative, cette variable est colorisée et le nuage grisé. Les graphiques sont enregistrés dans un dossier.

classification : Elle se base sur l'ACP. Permet de tracer le dendrogramme, et, une fois le nombre de classes déterminé (variable k), définit les classes de gènes et les trace sur le nuage de l'ACP. Permet aussi de séparer les gènes selon les classes et de tracer un graphique croisant une variable qualitative avec les classes.

2.2.3 Résultats

2.2.3.1 Statistiques simples

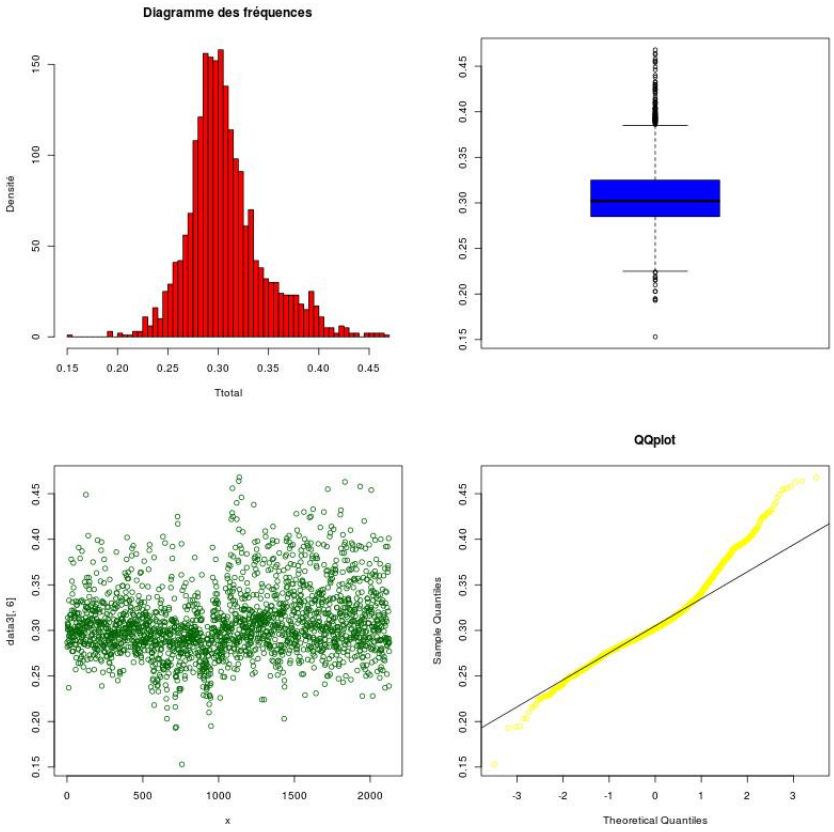


FIGURE 2.4 – Exemple de graphique de variable quantitative, ici Ttotal

Cette analyse systématique a permis une première approche du jeu de données. Elle a permis de :

- Mettre en évidence des variables doublon : Position 5' et Deb
- Repérer les variables à transformer. En effet, les variables CDS ont des valeurs extrêmes qui "écrasent" le boxplot. Il a été décidé à ce moment là qu'il serait sans doute nécessaire pour l'ACP de les transformer par un $\log_2(x+1)$. Pour certaines variables, la symétrie de la courbe (histogramme) est importante pour leur étude et ont été transformées dans ce but.
- Repérer les valeurs manquantes et de décider ou non de les laisser
- Repérer les variables présentant plusieurs pics telles que les variables delta ou l'hydrophobie, et de tracer une courbe de densité si nécessaire
- Repérer une variable ayant une allure anormale, ici tAI(FAUX)

Cette analyse a permis aussi de repérer d'autres singularités comme la variable Deb rangée par ordre croissant à partie de la moitié du fichier (figure 2.5. Cela est dû au fait qu'il s'agit de gènes encore inconnus qui ont donc été triés et nommés en commençant par y. Les gènes étant classés par ordre alphabétiques, les inconnus sont donc sortis en dernier laissant l'impression que la moitié du fichier était trié. Certains aspects des graphiques ont nécessité de l'aide pour leur interprétation, notamment les creux inattendus dans les histogrammes de position (début et fin). Ils sont dus à la nature des gènes étudiés, des gènes protéiques. N'étant pas les seuls gènes de l'ARN (gènes ribosomiques etc), ces gènes non étudiés prennent une place sur les brins qui correspondent à des « trous » dans les graphiques de position. Beaucoup de graphiques présentent des pics ou des creux ayant une explication relevant de la biologie cellulaire.

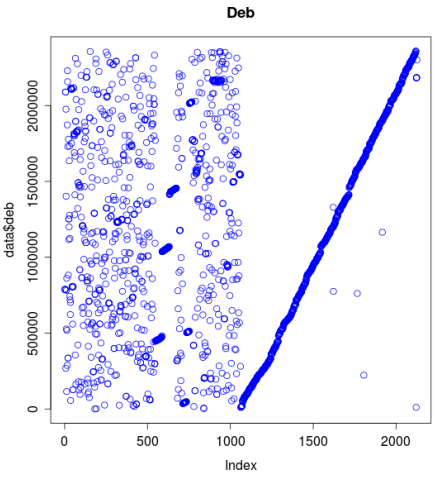


FIGURE 2.5 – Graphique de la variable Deb

2.2.3.2 Statistiques bivariées

Analyse croisée des variables qualitatives

L'analyse a permis de mettre en évidence quatre interactions entre les variables qualitatives. Il a été mis en évidence que les modalités de la variable AGGAG ont une certaine importance, notamment lorsque cette suite d'acides aminée est présente plus de une fois. La variable était tout d'abord prévue pour être recodée en qualitative « présence d'AGGAG » « absence d'AGGAG », mais les résultats de cette analyse ainsi que celle qualitative-quantitative ont amené à laisser la variable telle quelle. Lors de la classification, elle a été recodée en présence/absence, le nombre de séries dans le gène étant une information plus précise venant après la simple présence.

AGGAG - Signe *p-value = 0.006789*

On voit sur les graphiques et les tableaux que signe du brin est sans doute lié avec AGGAG. Les séries de 0,1,4,5 et 7 AGGAG (plus de 52%, jusqu'à 61% des effectifs pour la série de 4 AGGAG, sont plus présentes sur le signe +, tandis que les 2,3 et 6 sont plus présentes sur les brin -. En effectuant un test du χ^2 d'ajustement sur chacune des modalités, on s'aperçoit que les résultats dépendent beaucoup de celles-ci. Par exemple, sur le tableau ci-dessous, on voit que la répartition sur les brins dépend du nombre d'AGGAG (p-value inférieures à 5%).

```
> chi2(data$AGGAG,data$signe)
0   0.01099356
1   0.8784365
2   0.01525495
3   0.1506692
4   0.09880542
5   0.9295418
6   0.2118258
7   0.9579177
```

Au vu de ce test, on peut dire que seules les modalités 0 et 2 sont significativement différentes de ce qui est attendu à une échelle de 5%.

Cat - Signe *p-value = 0.0009225*

On peut voir sur les tableaux de contingence que le signe influence nettement les catégories fonctionnelles : AMI, CEL, COF, ENV, FAT, OTH, REG et REP sont majoritaires sur le brin +, tandis que INT,NRJ, PUR et TRD sont principalement sur le brin -.

L'autre tableau de contingence montre que les répartitions des catégories fonctionnelles sont légèrement différentes selon le signe du brin, l'énergie étant légèrement plus présente sur le brin moins (9.6%, second pourcentage sur le brin -, contre 11.3% OTH sur le brin +).

Un test du Khi2 d'ajustement à $\alpha = 5\%$ rejette l'hypothèse de la non-influence mutuelle des deux variables. Ici, le test du Khi2 sur chaque modalité des deux variables est intéressant : on peut y voir que toutes les catégories fonctionnelles ne sont pas influencées par le signe du brin d'ADN. Les catégories fonctionnelles significativement influencée par le signe sont : NRJ,REP et TRD, qui correspondent respectivement au métabolisme énergétique, la réplication ainsi que la translation. L'énergie et la translation se trouvent plutôt sur le brin - et la réplication sur le brin +. On remarque tout de même que l'enveloppe cellulaire (ENV) semble plutôt être sur le brin + même si sa significativité est légèrement supérieure à 5%. Enfin, le tableau du Khi2 dans l'autre sens confirme à alpha=5% que la répartition des catégories fonctionnelles sur l'ADN dépend du brin.

Cat - Localisation *p-value < 2.2e-16*

Quelque soit la catégorie fonctionnelle, la grande majorité des gènes se trouvent dans le cytoplasme à l'exception de TSP dont 65.4% des individus sont dans la membrane, 28.6% dans la paroi cellulaire/membrane et seulement 5.5% dans le cytoplasme. Le test du Khi-deux d'ajustement donne une p-value inférieure à 5% et confirme donc la présence d'un lien entre les deux variables.

On peut voir que la répartition de quasiment toutes les catégories fonctionnelles dépend de la Localisation, excepté pour CEL (processus cellulaires) et FAT ("Fatty acid"). Le résultat était prévisible et attendu : par exemple un gène ayant pour but la réplication de la cellule se trouvera plutôt à l'intérieur que sur la mebrane.

```
> chi2(data$Localisation, data$Cat)
Cellwall/Membrane   9.571904e-30
Cytoplasmic   1.81913e-142
Extracellular   1.046024e-10
Membrane   2.019078e-98
```

Il est visible sur cette sortie qu'à l'exception des valeurs manquantes, la composition des différentes localisations est liée aux catégories fonctionnelles.

Cat - AGGAG *p-value = 0.01001*

On s'aperçoit que le nombre de séries d'AGGAG varie selon les catégories fonctionnelles au delà de la simple présence/absence. Etant donné les pourcentages très proches, on peut regrouper les résultats de différentes manières :

Par deux : [0 -1] ou encore [1-0] : AMI, CEL, ENV, FAT, OTH, TSP, UNK. [1-2] : COF, INT, NRJ, PUR, REG, REP, TRD, TRS.

Par 3 : [0 - 1 - 2] : AMI, CEL, COF, ENV, FAT, OTH, REG, REP, TRS, TSP, UNK [1 - 2 - 3] : INT, NRJ, PUR, TRD.

Un test du Khi2 par modalités montre que seules quelques catégories fonctionnelles ont leur répartition dépendant de AGGAG : NRJ (énergie),OTH (autre),REG (régulation), et TSP (protéines de transport et de liaison). Inversement,les séries d'AGGAG se répartissant selon les catégories

fonctionnelles sont les séries 0,1 et 3. J'ai ensuite refait la manipulation en recodant AGGAG en qualitative binaire présence/absence : la présence de ou l'absence de la série AGGAG diffère selon la catégorie fonctionnelle.

Analyse croisée des variables qualitatives et quantitatives

Les graphiques sont générés sur quatre images différentes à raison de 9 diagrammes par image. On peut voir sur la figure 2.6 le graphique des deux variables qualitatives du début de jeu de données, puis les boîtes à moustache générées. Ces derniers permettent de voir les écarts de distribution et montrent l'influence plus ou moins marquée des qualitatives sur les autres variables. Le script conçu pour cette tâche pourrait-être amélioré notamment en trouvant le moyen de récupérer les coordonnées des variables qualitatives pour ne faire qu'une fonction et retirer les graphiques des variables qualitatives générés en même temps.

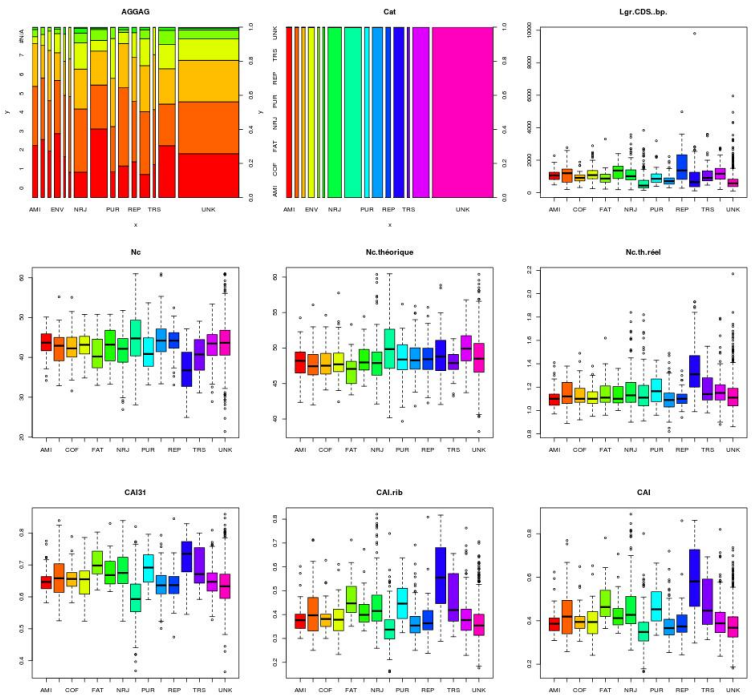


FIGURE 2.6 – Graphiques qualitatifs-quantitatifs des variables quantitatives non transformées en fonction de la catégorie fonctionnelle

En analysant les graphiques, les variables CDS étaient difficilement lisibles du fait de leurs très grandes valeurs extrêmes et ont donc été tranformées en $\log_2(x+1)$.
Cette analyse a aussi pu mettre en évidence des interactions a priori inattendues, notamment aux niveau des catégories fonctionnelles qui influent sur les résultats d'autres variables, comme cdsCountMean (variable à expliquer), même si les tests n'étaient pas probants. Cette constatation a amené à colorier l'ACP avec les catégories fonctionnelles pour mettre en valeur leur rôle.

Le Cluster, (figure 2.7), a permis de voir quelles variables se rapprochaient les unes des autres. On peut y lire des rapprochements attendus, comme l'énergie moyenne et GC (proportion de guanine

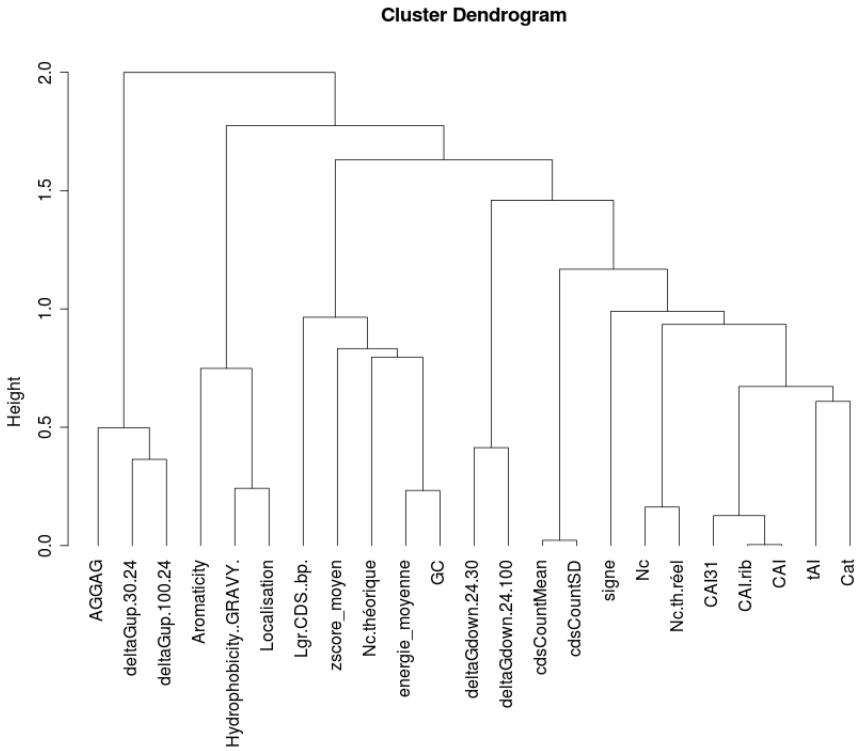


FIGURE 2.7 – Dendrogramme des variables qualitatives et quantitatives

et de cytosine), la localisation et l'hydrophobie, ou encore la variable AGGAG les deltaG, dépendant toutes de la guanine. On observe aussi un rapporchement des variables de même nature, comme les CAI et le tAI qui sont proches et concernent l'usage des codons. Plus remarquable, on observe une proximité de la catégorie fonctionnelle avec les variables liées à l'usage des codons, et, par extension, les variables liées à l'expression du gène (cdsCountMean, cdsCountSD).

Analyse croisée des variables quantitatives

Graphique croisé des variables quantitatives

Comme on peut le constater sur le graphique des variables quantitatives (figure 2.8), certaines variables semblent corrélées linéairement, telles que Nc et Nc th/réel. On voit par exemple que GC et tAI paraissent légèrement corrélées, ainsi que CAI et cdsCountMean (qu'on retrouvera dans l'ACP), mais que la plupart des autres variables forment plutôt un tas de points sans forme précise. D'autres telles que zscore_moyen sont constantes quasiment partout.

Matrice des corrélations

La matrice des corrélations étant relativement importante et difficile à lire, il est plus facile de la représenter sous forme de graphique avec le package ellipse. celui-ci est composé de ronds. Plus la corrélation est forte, plus l'ellipse est fine et penchée. Si elle est penchée vers la droite, il s'agit d'une corrélation positive, et dans le cas contraire, d'une corrélation négative.

Les classes sont les suivantes :

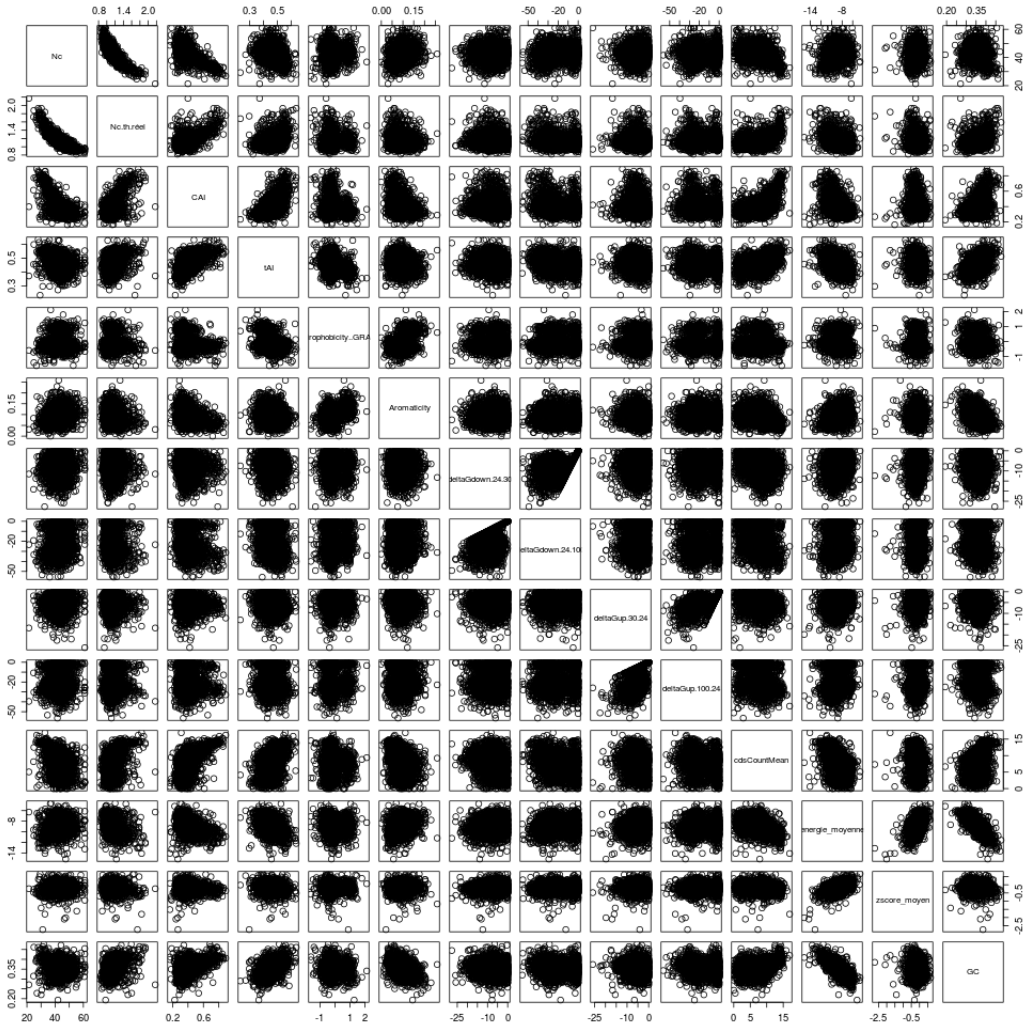


FIGURE 2.8 – Graphiques croisés des variables quantitatives

Ellipses penchées à gauche :
- Noir : -0 – -0.3
- Bleu clair : -0.3 – - 0.7
- Bleu foncé : -0.7 – -1

Ellipses penchées à droite :
- Noir : 0 – 0.3
- Orange : 0.3 – 0.7
- Rouge : 0.7 – 1

Nous pouvons voir sur la figure 2.9 que les variables à expliquer cdsCountMean et cdsBrutesMean sont très corrélées entre elles, tout comme les cdsBrutesSD et cdsCountSD. On remarque que les différents CAI sont aussi très liés, ainsi que les zscores. Les variables début et fin ne sont pas corrélées linéairement avec les autres, et on remarque une corrélation forte entre les différents Nc.

Cette analyse a permis par la suite de décider des variables à supprimer pour effectuer une ACP plus fine que la première, comme les variables cdsBruteSD et cdsMeanSD, ainsi que certains des CAI. Les variables début et fin ont elles été supprimées de l'analyse pour être comparées *a posteriori*. On remarque que les variables liées à l'usage des codons (CAI, Nc, tAI etc) sont corrélées avec cdsBrutesMean et cdsCountMean (moyennes d'expression du gène), ce qui était attendu.

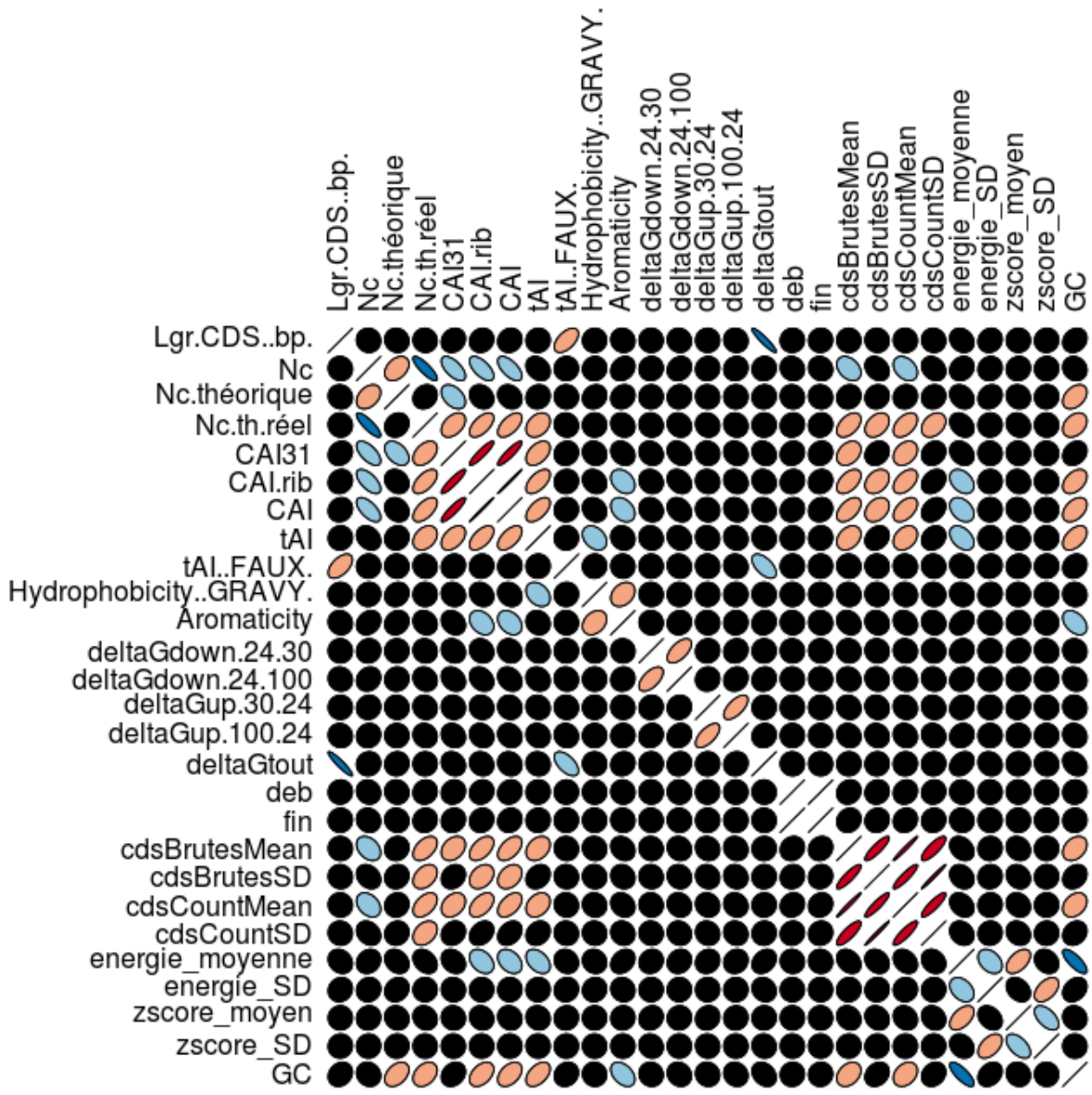


FIGURE 2.9 – Graphique des corrélations avec ellipse

2.2.3.3 Analyse en Composantes Principales

Première ACP

Pour cette ACP, il n'y a eu aucune transformation ni retrait de variable. Il s'agit d'une ACP réduite car les différentes variables ont des écarts-type très grands et des unités différentes. Elle a été effectuée avec le package FactoMineR qui en quelques lignes permet d'obtenir les graphiques et les calculs. Le but est de voir comment réagissent les variables.

On peut voir sur la figure 2.11 que beaucoup de variables sont confondues et que sur la gauche les noms de variables sont difficilement lisibles, dû à l'entassement des deltas. Comme la variance expliquée est ici très petite (8 axes pour 78% de la variance), et que beaucoup d'axes sont confondus, il a été décidé de retirer certaines variables et de transformer d'autres. On peut voir notamment sur ce graphique que les variables cds sont très proches voire confondues. Les variables deltas sont aussi proches les unes des autres, tout comme le CAI et le CAI ribosomique. Le tAI(faux) et la variable longueur semblent présents mais un peu à part. On remarque enfin que les variables début et fin sont mal représentées (et confondues) et que sur les trois variables NC (Nc, Nc théorique et le quotient Nc théorique sur réel) au moins l'une d'entre elles pourrait être supprimée. Le nuage des individus est marqué par la présence d'individus atypiques qui compliquent la lecture.

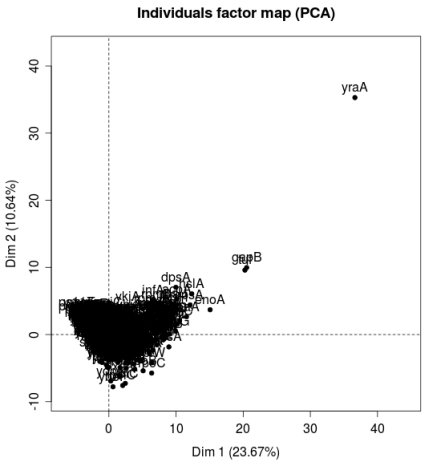


FIGURE 2.10 – Nuage d'individus de la première ACP

Plusieurs essais d'ACP ont permis de décider quelles variables étaient à supprimer et l'impact de leur suppression.

Variables retirées

A l'aide des analyses univariées et bivariées

LGR, deb, fin, Positions 5' : ce sont des variables de longueur et de position du gène sur le brin d'ADN. Elle seront utilisées pour une interprétation post-ACP. Position 5' est aussi un doublon de Deb.

tAI(faux) : il est en général mal rendu sur les axes et surtout n'est corrélé quasiment qu'avec deltaGtout ou les variables longueur. Probablement faux, comme son nom l'indique.

Après la première ACP

Une partie des variables éliminées ont été retirées car très liées avec d'autres variables sur l'ACP. Néanmoins, comme l'information était encore très floue, il a été décidé de retirer des variables en se basant sur le graphique des corrélations des variables quantitatives.

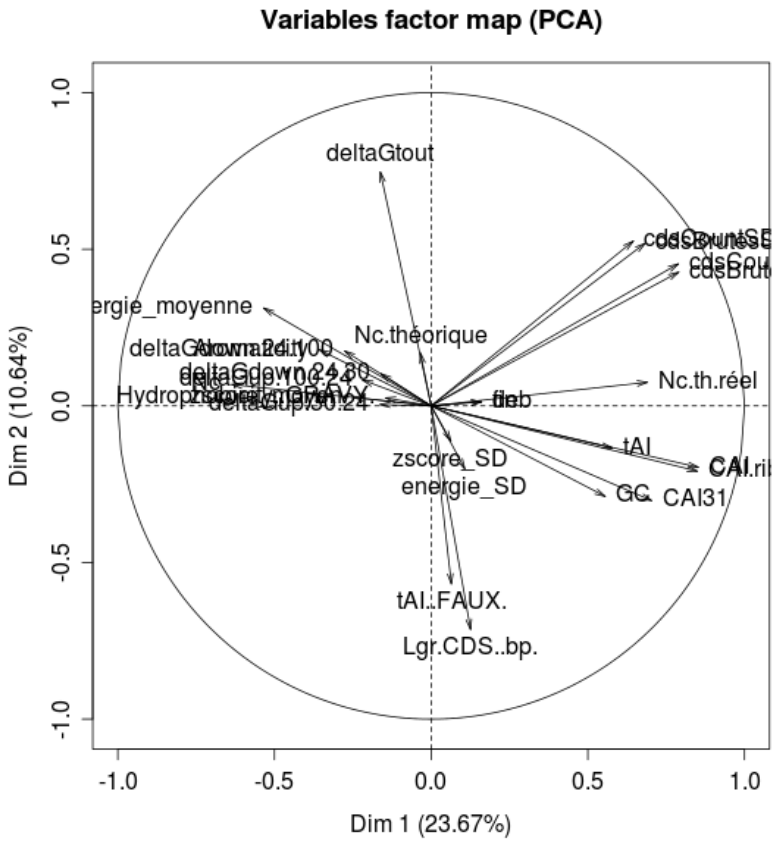


FIGURE 2.11 – Projection des variables sur les axes 1 et 2 de la première ACP

DeltaGtout : Cette variable correspond au total des delta et n'est pas vraiment corrélée avec le reste. Sans réelle importance.

CdsCountSD, CdsBrutesSD : Ces deux variables sont confondues avec les CDS moyenne. On a choisi de garder plutôt les moyennes.

CAI.31, CAI.rib : Ces deux variables sont très proches les unes des autres, CAI et CAI.31 étant même confondus sur les ACP.

CdsBrutesMean : Cette variable est corrélée linéairement ces CdsCountMean (voir étude dans l'annexe A)

ACP finale

Cette ACP a été effectuée avec beaucoup de variables supprimées et d'autres transformées. Elle a été interprétée de manière formelle, et les nuages de point des individus ont été coloriés selon la variable Cat (catégorie fonctionnelle) pour déterminer s'il y avait un lien entre la fonction du gène et les variables explicatives.

Variables utilisées :

- Gene.id
- Nc
- Nc.th.reel

- CAI
- tAI
- Hydrophobicity(GRAVY)
- Aromaticity
- DeltaGdown.24.30
- DeltaGdown.24.100
- DeltaGup.30.24
- DeltaGup.100.24
- CdsCountMean
- Energie_moyenne
- Zscore_moyen
- GC

CdsCountMean, comme toutes les variables cds a été transformée par $\log_2(x+1)$. Cette transformation est couramment utilisée dans le laboratoire pour les données génétiques.

Choix des axes

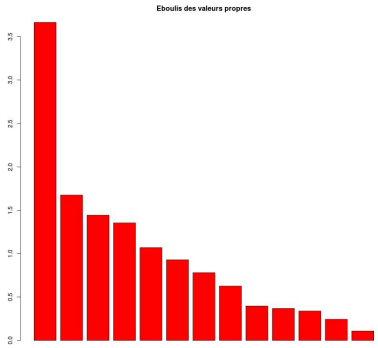


FIGURE 2.12 – Eboulis des valeurs propres

Valeurs propres et variance cumulée :

	eigenvalue	percentage of variance	cumulative percentage of variance
comp 1	3.98318666	28.4513333	28.45133
comp 2	1.69534173	12.1095838	40.56092
comp 3	1.52364537	10.8831812	51.44410
comp 4	1.43678254	10.2627324	61.70683
comp 5	1.29874441	9.2767458	70.98358
comp 6	0.94695923	6.7639945	77.74757
comp 7	0.88472657	6.3194755	84.06705
comp 8	0.64496282	4.6068773	88.67392
comp 9	0.39617146	2.8297961	91.50372
comp 10	0.36966879	2.6404913	94.14421
comp 11	0.33955404	2.4253860	96.56960
comp 12	0.24846665	1.7747618	98.34436
comp 13	0.16454599	1.1753285	99.51969
comp 14	0.06724375	0.4803125	100.00000

Règle de Kaiser : il s'agit de garder les axes ayant une valeur propre supérieure à 1. Elle donne 5 axes.

Règle de la part d'inertie : en observant la variabilité cumulée, on décide) combien de pourcents on souhaite étudier la variance. Pour expliquer 77% de la variance, il faut ici 6 axes.

Règle de l'éboulis : Les deux premiers axes étant automatiquement retenus, on stoppe l'interprétation des axes là où la différence entre les variances est maximum. Ici, cela donnerait 5 axes.

Étant donnée que la variance est ici très dispersée, on choisit ici 4 axes.

Contribution et cosinus carré :

Tableau des contributions :

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5
Nc	10.1050429	4.0427393	21.188270260	1.3610508	5.74337883
Nc.th.reel	12.3069462	2.8606702	10.923943691	1.1042723	10.05377515
CAI	16.7490463	2.2137081	3.352452848	0.1123828	0.20088365
tAI	10.7066554	0.9480619	0.617884429	0.1193599	5.45046046
Hydrophobicity.	0.6786244	4.0457585	0.006117216	1.5655745	43.44303857
Aromaticity	4.6732959	5.0620201	3.510938945	0.0295287	11.13826923
deltaGdown.24.30	1.4094429	15.1771264	3.633522994	27.7064404	1.71836229
deltaGdown.24.100	3.1976136	11.1981811	5.650761515	23.0192976	3.84625868
deltaGup.30.24	1.2313573	25.5737613	0.006958044	23.4184001	0.13176756
deltaGup.100.24	1.9458184	25.7823197	0.228831841	19.0424491	1.46757420
cdsCountMean	11.3645126	0.3959902	0.084874163	0.1545400	0.19760255
energie_moyenne	11.6270852	0.4989263	23.338535158	1.0387022	5.58039306
zscore_moyen	2.0822718	1.9503432	10.888402922	1.1115276	10.97392703
GC	11.9222871	0.2503936	16.568505973	0.2164740	0.05430874

Moyenne des contributions : 100/13=7,69. On garde les variables dont la valeur est supérieure à cette moyenne.

Tableau des cosinus carrés :

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5
Nc	0.40250272	0.068538247	3.228341e-01	0.019555340	0.07459181
Nc.th.reel	0.49020864	0.048498136	1.664422e-01	0.015865992	0.13057284
CAI	0.66714578	0.037529917	5.107949e-02	0.001614696	0.00260896
tAI	0.42646607	0.016072890	9.414367e-03	0.001714942	0.07078755
Hydrophobicity.	0.02703088	0.068589432	9.320468e-05	0.022493901	0.56421403
Aromaticity	0.18614610	0.085818540	5.349426e-02	0.000424263	0.14465764
deltaGdown.24.30	0.05614074	0.257304157	5.536200e-02	0.398081298	0.02231713
deltaGdown.24.100	0.12736692	0.189847438	8.609757e-02	0.330737248	0.04995306
deltaGup.30.24	0.04904726	0.433562647	1.060159e-04	0.336471483	0.00171132
deltaGup.100.24	0.07750558	0.437098425	3.486586e-03	0.273598584	0.01906003
cdsCountMean	0.45266975	0.006713388	1.293181e-03	0.002220403	0.00256635
energie_moyenne	0.46312851	0.008458506	3.555965e-01	0.014923892	0.07247504
zscore_moyen	0.08294077	0.033064982	1.659006e-01	0.015970234	0.14252326
GC	0.47488695	0.004245027	2.524453e-01	0.003110260	0.00070533

Pour savoir quelles variables sont à retenir et quelles variables sont trop faiblement représentées, on utilise la moyenne des cosinus carrés. Elle est de 1/nombre d'axes, soit ici 1/13=0.077. On garde les variables dont le cosinus carré est supérieur à cette valeur.

Graphique des variables

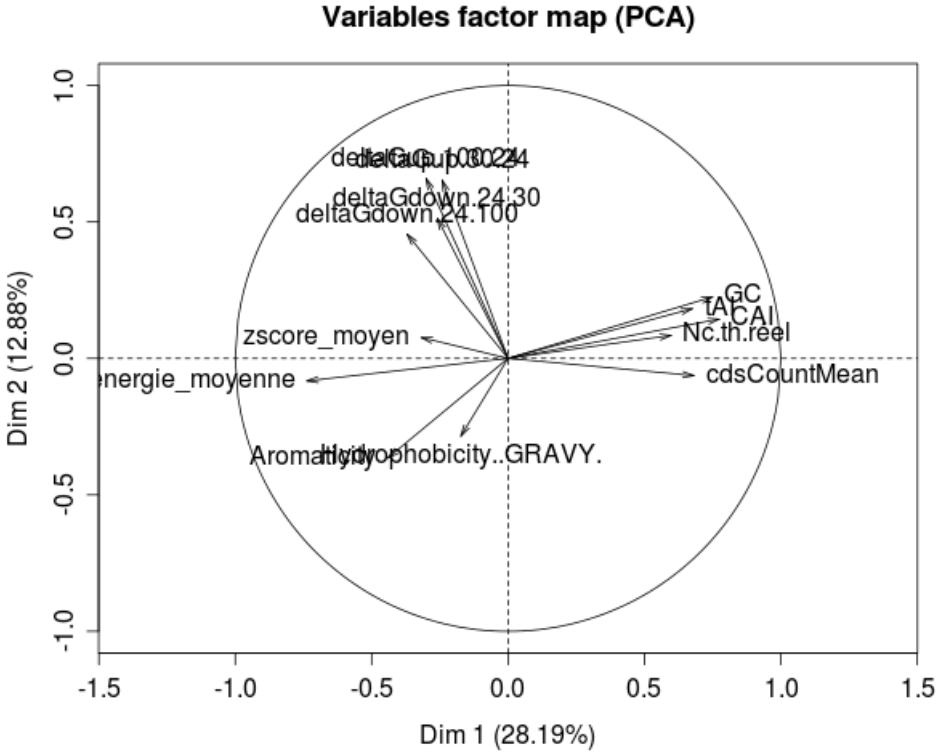


FIGURE 2.13 – Répartition des variables sur les axes 1 et 2

On peut voir sur le graphique (figure 2.13) que les variables forment trois paquets distincts : tous les deltas ensemble ; le groupe bien représenté sur l'Axe 1 positif ; les autres.

Les variables à expliquer (CAI, tAI et cdsCountMean) sont bien représentées sur l'axe 1. On remarque que zscore_moyen n'est pas très bien représenté et que les deltas sont très proches les uns des autres.

Tableau d'interprétation des axes :

	+	-
Axe 1	GC , tAI , CAI ; Nc.th.re , cdsCoutMean	Delta*100 , zcore_moyen , énergie_moyenne , Aromaticity , Nc
Axe 2	Delta*	Aromaticity
Axe 3	Nc , Gc	Énergie moyenne , zscore_moyen , Nc th/re
Axe 4	DeltaGdown24.30 , deltaGdown24.100	DeltaGup100.24 , deltaGup30.24

On observe sur ce tableau que les quatre axes sont clairement des axes d'opposition. L'axe 1 oppose les variables à expliquer aux Delta*100 , zcore_moyen , énergie_moyenne , Aromaticity , Nc. L'axe 2 sépare lui les variables delta de l'aromaticité. L'axe 3 oppose zscore_moyen, Nc th/re et énergie moyenne à Nc et GC. L'axe 4 oppose les deltas entre eux sans que les autres variables soient très représentées sur cet axe. L'opposition des deltas n'est cependant pas une surprise. Les variables à expliquer ne sont pas représentées sur les deux derniers axes (voir figure 2.15).

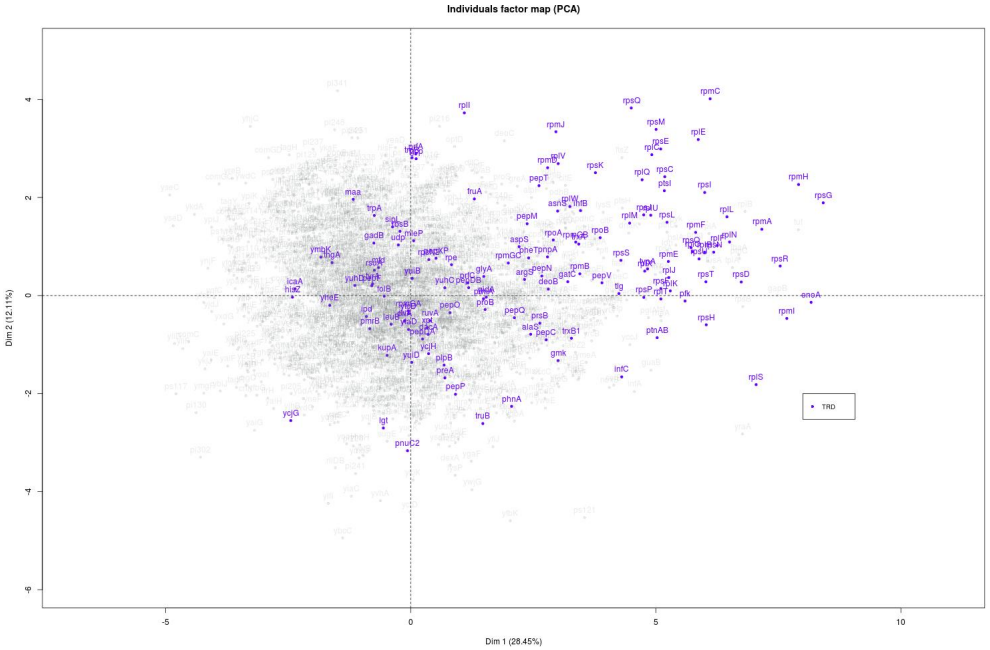


FIGURE 2.14 – Nuage de points de la modalité TRD sur le nuage grisé

Nuage des individus

On peut voir que le nuage de points (figure 2.16) est plutôt homogène, avec quelques valeurs à part sur l'axe 2. Il n'y a pas de valeurs extrêmes comme lors des premières ACP. On remarque aussi que les catégories fonctionnelles semblent se répartir selon la place sur l'ACP, notamment les TRD et UNK.

Le nuage de points sans la variable UNK (le groupe le plus important) a ensuite été tracé puis chaque modalité de Cat sur le nuage grisé comme sur la figure 2.14.

Synthèse des résultats : les corrélations trouvées

L'étude effectuée a pu mettre en évidence que les variables cdsCountMean et cdsBrutesMean sont corrélées linéairement entre elles. CdsBrutesMean adonc été de l'ACP.

L'ACP a pu montrer que les variables GC, tAI, CAI, Nc th/re et cdsCountMean sont corrélées entre elles sur l'axe 1. Elles s'opposent à l'énergie_moyenne, l'aromaticité, et le NC. Dans une moindre mesure, elles sont inversement corrélées aux deltas. Les deltas sont très proches les uns des autres, il faudra attendre l'axe 4 pour les voir s'opposer. L'axe 2 oppose les delta à l'aromaticité. L'axe 3 réunit les GC et Nc, par opposition à l'énergie moyenne, le zscore_moyen et le Nc th/re.

Les catégories fonctionnelles semblent être peu affectées par ce classement, sauf peut-être les modalités PUR, TRD et COF qu'il faudra approfondir sur l'axe 1.

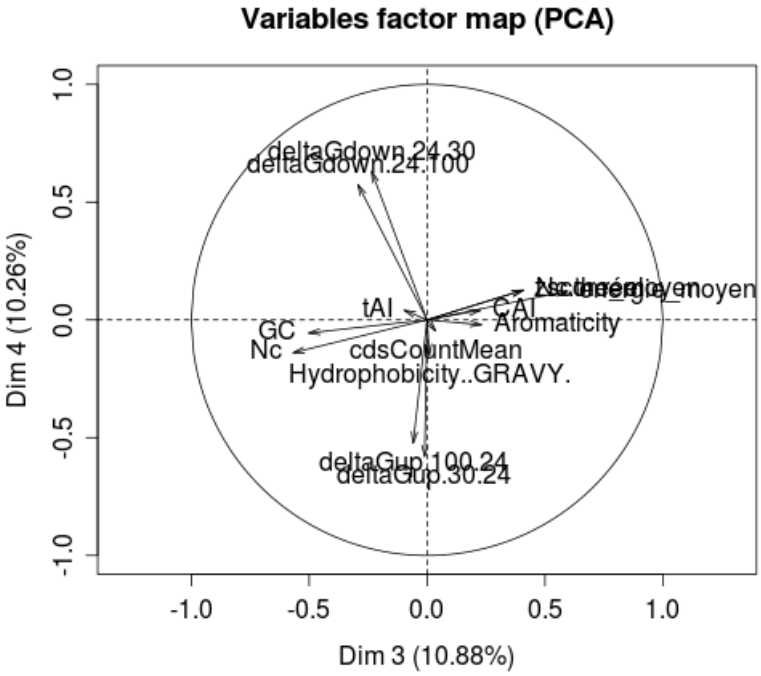


FIGURE 2.15 – Répartition des variables sur les axes 3 et 4

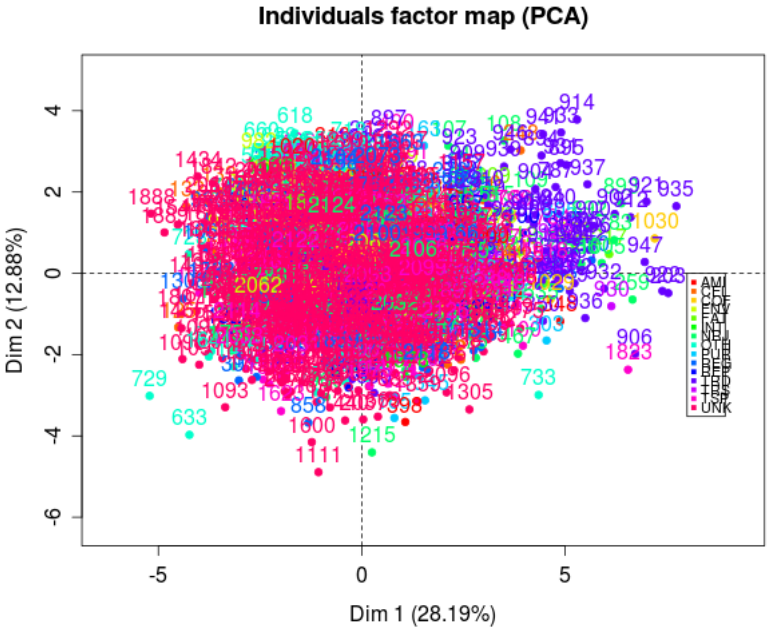


FIGURE 2.16 – Répartition des individus sur les axes 1 et 2

2.2.3.4 Classification

L'ACP effectuée, les gènes ont été rassemblés sur cette base pour créer des groupes ayant des points communs. Cette classification pourra être utile a posteriori, au moment de l'étude de la dégradation. La classification effectuée ici est une classification hiérarchique par la méthode de Ward.
Le dendrogramme (figure 2.17) a permis de choisir le nombre de classes, ici cinq, malgré la présence d'une classe avec un fort effectif. En effet, la variance supplémentaire interprétée n'était pas très grande et le groupe était découpé de manière déséquilibrée en choisissant six classes.

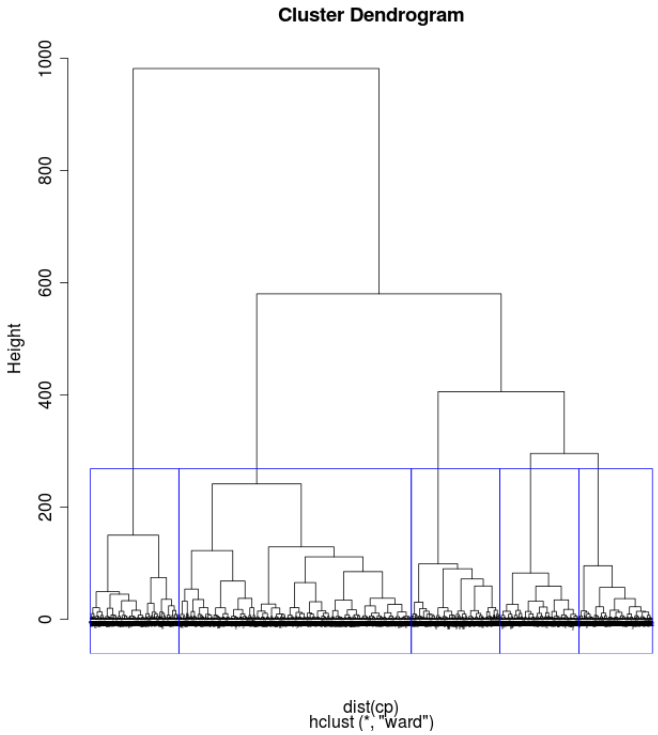


FIGURE 2.17 – Dendrogramme

On peut voir sur le dendrogramme ou sur le tableau ci-dessous que les classes sont de taille équivalentes à l'exception de la classe 2.

```
> table(classes)
classes
  1   2   3   4   5
299 879 335 278 335
```

Une fois les groupes déterminés, ceux-ci ont été étudiés. Comme on peut le voir sur le nuage de points représentant les groupes (figure 2.18), ceux-ci peuvent être interprétés selon les axes de l'ACP :

Groupe 1 ***(en rouge)*** **:** Groupe plutôt central, sans particularité visible au premier abord

Groupe 2 ***(en jaune)*** **:** Central, comme le groupe 1. Ce groupe semble comporter des individus ayant une Aromaticité et une Hydrophobie plutôt élevées.

Groupe 3 ***(en vert)*** **:** Ce groupe est plutôt sur la droite du nuage. Ce sont des gènes ayant un GC, Nc th/re, tAI, CAI, cdsCountMean plutôt élevés ainsi qu'une Aromaticité, énergie moyenne, z_score moyen, et delta plutôt faibles.

Groupe 4 ***(en bleu)*** **:** Situé en haut à gauche, les gènes de ce groupe ont plutôt des delta élevés.

Groupe 5 ***(en violet)*** **:** Ce groupe est sur la gauche du nuage et correspond à des gènes ayant une Aromaticité, énergie moyenne, z_score moyen, et delta plutôt élevés ainsi que des GC, Nc th/re, tAI, CAI, cdsCountMean plutôt faibles.

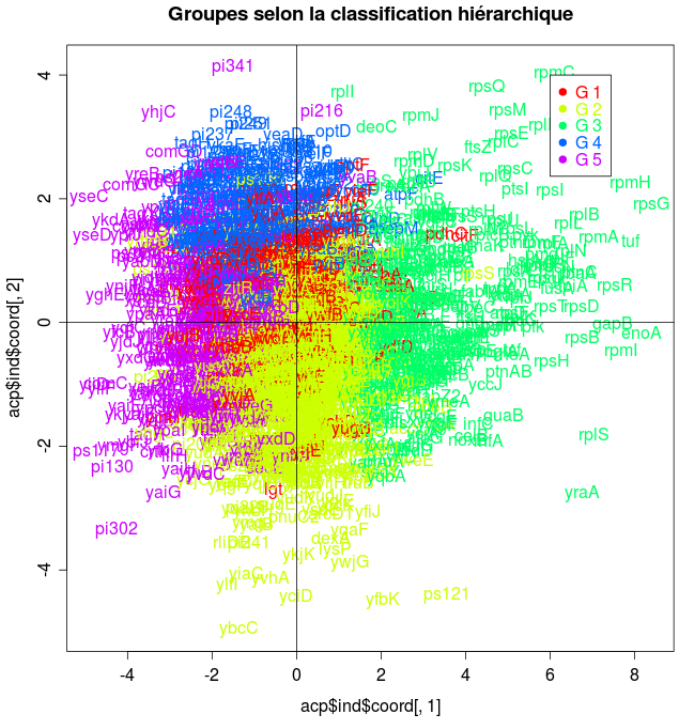


FIGURE 2.18 – Représentation des groupes dans le nuage d'individus de l'ACP

Les groupes ont ensuite été étudiés selon les variables AGGAG et Catégories fonctionnelles. Les tableaux de contingence ainsi que les tests du χ^2 et les graphiques (voir figure 2.19 et 2.20)) ont mis en valeur que ces deux variables sont placées différement selon les groupes, ce qui peut être intéressant par la suite. On voit par exemple sur la figure 2.14 que la modalité traduction de la variable catégorie fonctionnelle semble être sur-représentée dans le groupe 3, plutôt à droite.

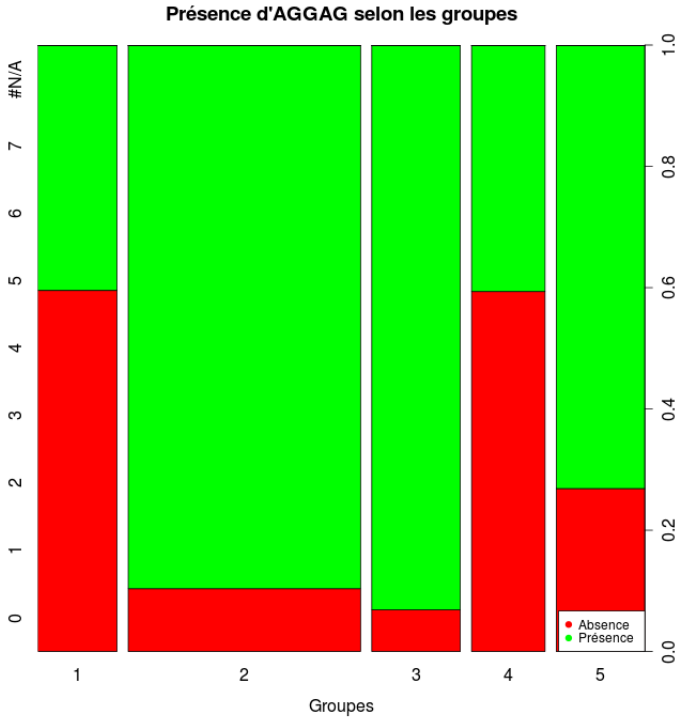


FIGURE 2.19 – Répartition des présences/absences d'AGGAG selon les groupes

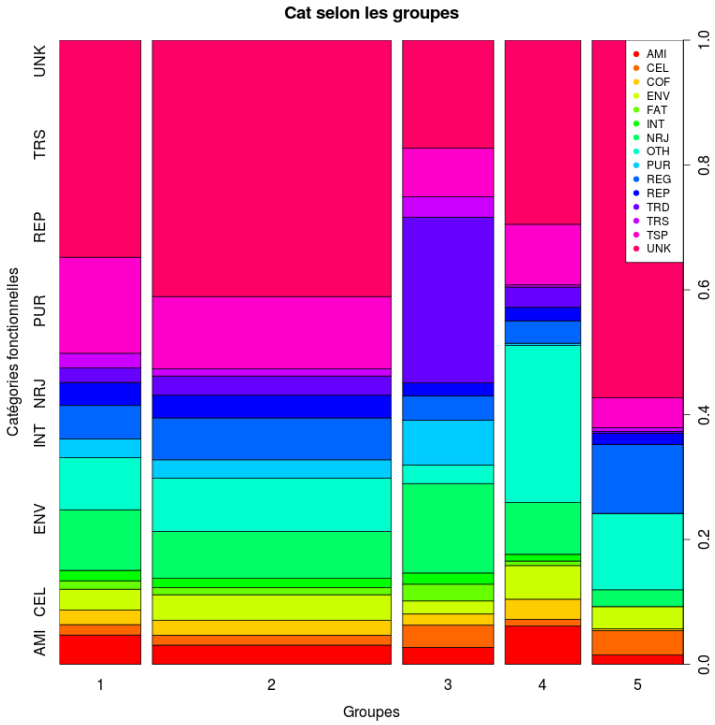


FIGURE 2.20 – Répartition des catégories fonctionnelles selon les groupes

2.3 Développement d'un outil de traitement de sortie segmentation

2.3.1 Contexte

La première partie de mon stage consistait à exploiter les données de transcriptome (ARN) au niveau des gènes de protéine du génome. Dans cette seconde partie, ces données sont exploitées au niveau nucléotidique (c'est-à-dire pour chaque position dans le génome). Les données sont donc un vecteur de valeurs de transcription (comptage) de chaque position. La taille du vecteur est donc de la taille du génome : cela revient à traiter un signal.

La problématique consiste à détecter des segments homogènes et des bornes de transition dans le signal, donc d'identifier des points de rupture. Pour aborder cette question, nous appliquons sur un même jeu de données (le vecteur de signal) plusieurs algorithmes de segmentation. Afin de comparer leurs résultats entre eux et les confronter à une référence (l'annotation), il est nécessaire de mettre en forme leurs sorties.

Dans le cadre du stage, l'objectif est de transformer les sorties brutes des segmenteurs pour produire des sorties dans le format standard GFF (utilisé dans le domaine de la bioinformatique).

2.3.2 Problématique

Nous disposons d'un fichier contenant, pour un brin donné (+ ou -) le vecteur du nombre de transcrits par nucléotide. La dimension du vecteur correspond à la longueur du génome. Certaines valeurs peuvent être nulles (pas de transcrit détecté à la position concernée). A partir de ce fichier du nombre de transcrits par position nous voulons identifier les objets biologiques transcrits, que l'on représentera sous forme de segments. Pour ce faire nous cherchons des segments (position début à position fin, brin + ou -) pour lesquels le nombre de transcrits est homogène.
Remarque : actuellement l'information du brin transcrit n'est pas disponible mais tout outil développé devrait prendre en compte la possibilité de distinguer les brins "+" et "-".

Pour définir les segments, il existe dans la littérature plusieurs méthodes de segmentation. Afin d'identifier la méthode la plus adaptée, les résultats des différentes méthodes peuvent être comparées entre elles, ainsi qu'à l'annotation du génome.

Les résultats des méthodes de segmentation sont donnés sous forme d'une liste de positions : les points de rupture (ChangePoints).

L'annotation est fournie au format GFF (ref : http://gmod.org/wiki/GFF)

2.3.3 Spécifications fonctionnelles

Le but du projet est de développer un script permettant de convertir une sortie de segmenteur (typiquement un vecteur de points de rupture dans le génome) en fichier de segments au format GFF (chaque ligne correspondant à un segment). Le premier segment commence par 1 et finit par la première position dans la liste. Le 2ème segment commence à la première position de la liste +1 et

finit à la 2ème position de la liste, et ainsi de suite jusqu'au dernier segment qui finit par la longueur du génome (voir l'exemple ci-dessous).

L'outil prendra en entrée un fichier liste de positions, les autres paramètres seront :

-n : indicateur NAME auquel il ajoute un nombre qui s'incrémente à chaque nouveau segment
-s : SOURCE
-t :TYPE
-b : STRAND : PLUS,MOINS
-c : nom du fichier de comptage par nucléotide (autant de valeurs que de nucléotides dans le génome)
-f : (facultatif) fichier contenant les coupures. Par défaut : jumps.txt.

La valeur SCORE sera calculée. Elle est égale à la moyenne du nombre de transcrits sur le segment c'est à dire c'est la somme des valeurs de la position_DEB à position_FIN du génome divisée par la longueur du segment (les valeurs sont données dans le fichier de comptage).

Le résultat devra être au format GFF où chaque champ est séparé par une tabulation.

Dans l'exemple ci-dessous les points de rupture sont dans le fichier jumps.txt et le fichier de comptage est deplog.txt.

Exemple :

A partir d'un fichier tel que « jumps.txt » :

3028
10818
12257
12618
16032
21416
...

La commande :

```
./cp2gff.py -n cptI -s binInf -t RNA -b PLUS -c deplog.txt jumps.txt
```

Retournera le résultat suivant :

#result of : « cp2gff.py -n cptI -s binInf -t RNA -b PLUS -c deplog.txt jumps.txt »
#NAME SOURCE TYPE START END SCORE STRAND FRAME
cptI1 binInf RNA 1 3028 3.405 + .
cptI2 binInf RNA 3029 10818 6.705 + .
cptI3 binInf RNA 10819 12257 1.765 + .
cptI4 binInf RNA 12258 12618 5.779 + .
cptI5 binInf RNA 12619 16032 2.705 + .
cptI6 binInf RNA 16033 21416 5.240 + .
...

Les vérifications à faire sur les paramètres et fichiers sont les suivantes :

- NAME, SOURCE et TYPE ne doivent pas contenir de tabulation ni de caractères spéciaux (voir spécifications des champs des fichiers GFF).
- Dans les fichiers, toute ligne commençant par « # » sera ignorée, et seules les valeurs numériques (entières pour les points de rupture) sont prises en compte (toute autre caractère est ignoré).
- Dans le fichier d'entrée, la dernière valeur ne doit pas être supérieure à la longueur du génome (nombre de valeurs dans le fichier de comptage).

2.3.4 Documents et références

Le script a été effectué avec la version 2.7+ pour linux de Python. Les documents utilisés pour la conception sont les suivants :

- Programmation Python, J.Delamarche
- Tutoriel de programmation Python, www.siteduzero.com
- http://gmod.org/wiki/GFF : spécifications détaillées du format GFF

Vocabulaire de base

Annotation du génome : Liste de segments des gènes sous forme de début et de fin

Backlog : Un "backlog" est une liste de fonctionnalités ou de tâches, jugées nécessaires et suffisantes pour la réalisation satisfaisante du projet

END : Fin de la séquence

Génome : C'est l'ensemble du matériel génétique d'un individu ou d'une espèce codé dans son ADN

NAME : Ici, il s'agit de l'identifiant de la séquence

Nucléotide : Molécule organique formant entre autres la base de l'ADN et de l'ARN

SCORE : Moyenne du nombre de transcrits par position

Séquenceur : Dispositif qui produit ou analyse une séquence. L'usage ici était de découper une séquence génétique en fonction de ruptures marquant le début et la fin de gènes.

SOURCE : Texte libre pour qualifier la procédure permettant de générer la sortie

Sprint : Itération temporelle utilisée dans la méthode Agile SCRUM

START : Début de la séquence

STRAND : Signe du brin : + ou -.

TYPE : Le type de la sortie (ici, ARN)

2.3.5 Organisation du développement

La méthode utilisée lors du développement est itérative et interactive et inspirée d'une méthode agile de type SCRUM.

L'organisation du développement s'est basée tout d'abord sur un cahier des charges précis, puis des points ont été organisés avec des retours et des suggestions d'amélioration. Avant chaque point, les nouveautés du script étaient testées (fichiers tests puis fichiers réels) pour vérifier le bon fonctionnement. Le but n'était pas de remplir précisément le cahier des charges mais de faire en sorte que le script soit le plus utile et le plus facile possible d'utilisation pour l'utilisateur. Un document d'utilisation était rempli et modifié au fur et à mesure du développement.

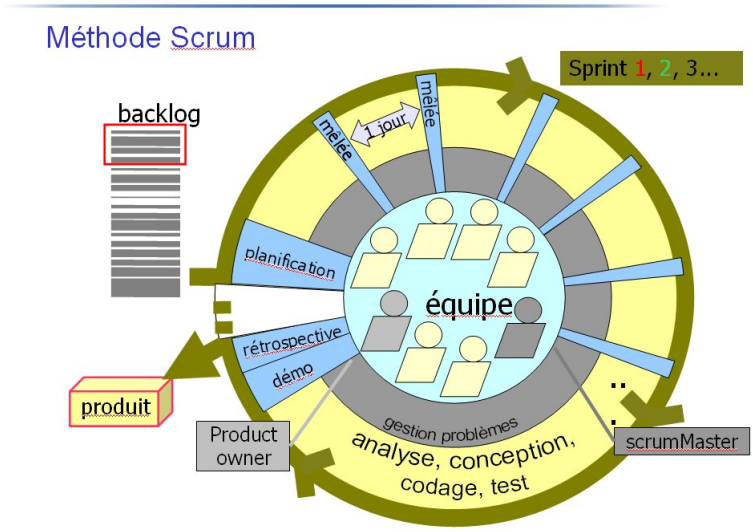
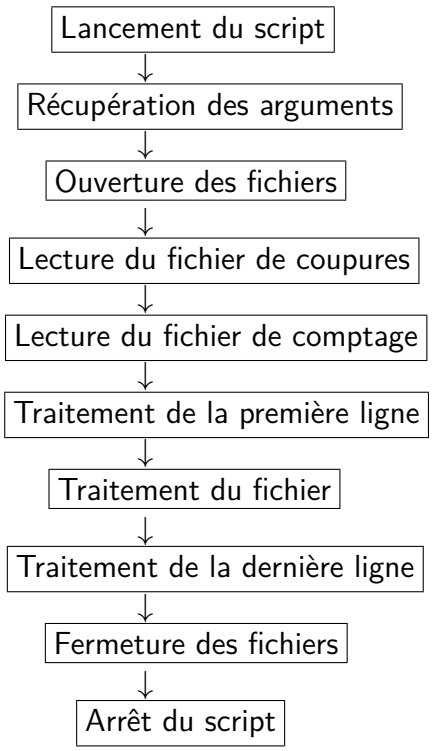


FIGURE 2.21 – Schéma général de la méthode SCRUM

La méthode SCRUM, qui est l'une des méthodes agile, a été présentée dans le cadre d'une conférence à l'INRA. Elle s'appuie sur le découpage d'un projet en incréments, nommés "sprint", ainsi que l'auto-organisation de l'équipe de développement. Le sprint se termine par une démonstration de ce qui a été achevé.

2.3.6 Conception

2.3.6.1 Conception générale



Après une première version du script, plusieurs améliorations ont été ajoutées :

Evolution 1 :

Ajouter un filtre pour éviter d'avoir des segments trop petits :
Il arrive souvent que le logiciel de segmentation détecte deux coupures très proche l'une de l'autre. Cet intervalle réduit n'est pas un nouveau gène mais une incertitude sur l'endroit de la jointure entre deux gènes. Un paramètre (par défaut à -1) qui prend une taille minimum de gène a donc été ajouté et l'élimine si trop petit. L'argument à ajouter dans la ligne de commande est *-k*.

Pouvoir modifier le nom de sortie :
Le nom du fichier de sortie par défaut est composé du nom et de la source passés en paramètres, ainsi que la date. Un argument *-o*, facultatif, a été ajouté pour pouvoir entrer son propre nom de sortie.

Evolution 2 :

Afficher les coupures retirées :
Dans le but de pouvoir comparer les résultats de différents algorithmes et comprendre d'éventuels décalages, les coupures non prises en compte par le filtre apparaissent dans le terminal.

Evolution 3 :

Insérer des classes :

Pour pouvoir être utilisé sous Appolo, les sorties doivent être organisées en classes. Sans cela, on ne vera à l'écran qu'une ligne pleine. La classe a été collée à la source dans le fichier sortie. Pour insérer les bornes des intervalles, deux solutions ont été envisagées : la première était l'ouverture d'un fichier texte dans lequel auraient été mises les classes. La deuxième, qui a été choisie, est de mettre les intervalles directement dans les arguments de la ligne de commande. La commande est *-p* et les bornes sont séparées par un underscore. Exemple : 1_2_3.

2.3.6.2 Algorithme

```
ligne python
encodage \\
import des packages \\
recuperation des options de la ligne de commande\\

Recuperation des classes:
        classe=split(classe)
        nl=longueur de classe

fonction de test des classes classeM(moy):
        c prend NF
        Pour chaque element de classe de 0 a nl:
                si moy superieure a classe(element):
                        alors message et sortie de boucle

                si moy inferieure a la classe
                        alors c prend classe
retour c


fonction de test des caracteres speciaux caracSpe(chaine):
        speciaux prend les caracteres speciaux
        retour prend FALSE
        si chaine contient un caractere special
                alors retour prend TRUE
retour

si CaracSpe(nom)==True alors exit
si CaracSpe(source)==True alors exit
si Caracspe(type)==True alors exit

si strand="PLUS" alors strd=+
elsif strand="MOINS" alors strd=i
else exit

a=ouverture premier fichier ou exit
b=ouverture deuxieme fichier ou exit


max=longueur(a)
si a[max]>longueur(b)
        alors exit

si -o different de none #parametre par defaut
        alors creation fichier sortie nomme parametre+.GFF, mode ecriture
```

```
sinon creation fichier sortie nomme source+type+.GFF, mode ecriture

ecriture de la ligne de commande en premiere ligne
ecriture de la ligne d'en-tete


total=0
numero de gene=1

si premiere entree de a -1 superieure a filtre alors
        pour k allant de 0 a la premiere entree de a
                suppression des commentaires
                total=total+b[k]

        moyenne=total/premiere entree de a
        cl=ClasseM(moy)
        si cl a la valeur NF
                alors sortie de script
        sinon ecriture dans le fichier de la ligne : source+type+classe+1+a
           [0]+moyenne+strd+.

        numero de gene ++

sinon print "a[0] elimine"

deb prend a[0] +1
i prend 1

Tant que i different de longueur de a

        Tant que i different de longueur de a et recherche de ligne
           commencant par # vraie
                Alors i++

        Si i>longueur(a) alors sortie de script

        Suppression d'une eventuelle regex commencant par diese
        tot prend 0
        FinI prend la partie entiere de a[i]

        Si (FinI - DebI)>filtre

                Pour j entre DebI et FinI
                        b[j]=string(b[j])
                        recherche de commentaire dans b[j]
                        total=total+b[j]

                moy= total/(DebI-FinI+1)
                cl=classeM(moy)

                Si cl = NF
                        Alors sortie de script "Classes superieures trop
                           petite"."
                ecriture dans fichier de sortie : nom, nogene, source,
                   classe,DEbI, FinI, moyenne arrondie a 5, strand
        nogene++
```

```
        Sinon
                print DebI, finI elimine

        i++
        DebI=FinI+1

#traitement derniere ligne
tot=0

Pour k allant de (DebI a longueur de B)
        suppression d'un commentaire dans b[k]
        total=total+b[k]

moy=tot/(longueur(b)-DebI)
cl=classeM(moy)
Si cl=NF
        sortie de script : "Classe superieure trop petite"
ecriture dans le fichier de sortie : nom, nogene, source, classe,DEbI, FinI,
    moyenne arrondie a 5, strand

fermeture du fichier de sortie
print : traitement termine
```

2.3.7 Tests effectués

Respecter le format GFF :

Pour cela, j'ai vérifié à l'aide d'une fiche de spécification détaillée (http://gmod.org/wiki/GFF) que le fichier créé respectait bien le format donné, notamment le nombre de colonnes, la présence de tabulation et l'absence de caractères spéciaux (pas d'espace ni de ">"). Le fichier a ensuite été passé dans le logiciel Appolo pour déterminer s'il pouvait l'utiliser ou non.

Vérifier l'absence de caractères spéciaux dans les variables NAME, SOURCE et TYPE :

Dans le script, une fonction a été ajoutée, renvoyant un booléen selon si la chaine de caractères entrée contient des caractères spéciaux tels que },£,ø,*,μ. Même si cela n'était pas dans le cahier des charges (ceux cités précédemment sont acceptés dans le format GFF), cette fonction été ajoutée car les arguments ne les contiennent normalement pas et que leur présence serait synonyme d'une faute de frappe. Les arguments NAME, SOURCE et TYPE sotn testés dans le script qui s'arrête avec un message d'erreur si un caractère de la liste apparait. Lors de la phase de test, chacun de ces arguments a été entré avec des caractères spéciaux pour vérifier que les messages d'erreur s'affichent bien. Il est à noter que certains caractères spéciaux tels que le $ ou les accolades entraînent le non-démarrage du script indépendamment du code.

Ignorer les lignes commençant par # :

Cette vérification a été la plus difficile. En effet, étant donné que le script utilise chaque coupure pour créer un intervalle, simplement rajouter $+1$ au compteur de lecture ne résout pas le problème. Exemple :

2682
3009
#Commentaire

4526

Le script va arriver sur 3009. L'intervalle créé sera [compteur-1 ; compteur], soit 2682-3009, et la moyenne des scores sera effectuée. Ensuite on arrivera sur la ligne de commentaire. L'intervalle [3009 - #Commentaire] renverra une erreur. Si on demande au compteur de s'incrémenter de un s'il rencontre une ligne de commentaire, le nouvel intervalle sera [#Commentaire - 4526] et arrêtera lui aussi le script. Pour obtenir l'intervalle [3009 - 4526], on peut passer par la création d'une variable curseur qui définit de combien on doit retirer au compteur pour définir l'intervalle. Par défaut cette valeur est à un mais en cas de ligne de commentaire, elle est à deux. Mais cette solution ne prend pas en compte un gros bloc de commentaires. Le plus simple dans ce cas de figure est donc la création d'une variable de début de gène et une variable de fin de gène, ce qui a été finalement fait.

Vérifier que la dernière valeur du fichier de comptage (fichier d'entrée) ne soit pas supérieure à la longueur du fichier jumps.txt (longueur du génome) :

Vérifier que la dernière valeur du fichier de comptage (fichier d'entrée) ne soit pas supérieure à la longueur du fichier jumps.txt (longueur du génome) Cette vérification se fait grâce à la comparaison des deux valeurs, et arrête le script si la condition n'est pas remplie. Des fichiers tests ont été utilisés pour vérifier le fonctionnement de cette fonction.

Filtrer les tailles de gène :

Pour cela, des essais avec un fichier test et différents arguments ont été essayés. Le fichier de sortie a aussi été comparé avec un fichier venant d'un autre traitement.

Mise en place des classes :

La vérification des classes est simple : il suffit de comparer la classe ainsi que la moyenne dont elle est issue. Ce test a mis en évidence que la présence d'un intervalle de fin trop petit pose problème et amené une correction avec un message d'erreur.

3 Bilan

L'étude statistique du jeu de données du projet DEGRADOMICS a permis de mettre en évidence des interactions entre variables. Par exemple, la présence ou l'absence d'AGGAG selon les groupes de classification des gènes et leur place dans l'ACP est significatif tant sur le plan statistique que biologique. Les gènes ayant pour fonction la traduction semblent avoir un gros pourcentage de guanine et cytosine, et une énergie moyenne assez faible.
Le but de cette étude, qui était de faire une analyse exploratoire en vue d'une comparaison ultérieure et laisser des scripts réutilisables, a été atteint. Les résultats, présentés devant les biologistes de l'INSA, ont amené une évolution des questions de départ ainsi que l'ajout de variables supplémentaires à intégrer au jeu de données.
Le script python créé remplit également sa fonction de traitement des fichiers et a subi plusieurs évolutions par rapport au cahier des charges. Sa conception s'inscrit dans le projet de segmentation en permettant de comparer entre eux ainsi qu'à une référence les résultats de différentes techniques de segmentation.

Ce stage de douze semaines effectué au sein de l'INRA m'a permis de découvrir la recherche en milieu non hospitalier. Ayant déjà effectué un stage orienté recherche au service de bactériologie du CHU hôpital nord d'Amiens, j'ai pu découvrir un autre aspect de cette filière, notamment en ce qui concerne l'application des statistiques dans ce domaine. Lors de ces trois mois, j'ai pu manipuler de manière plus approfondie le logiciel R, avec l'interface RStudio et de nombreux packages tels que Ellipse ou encore Heatmap. J'ai aussi beaucoup appris sur les graphiques sous R, ainsi que leur coloration, le réglage de la légende et leur automatisation. Enfin, j'ai pu utiliser mes connaissances acquises en statistiques cette année pour faire des tests sur les données et des régressions linéaires, ainsi que les commandes UNIX apprises au début du premier semestre. Les acquis en biologie de mon DUT Génie Biologique m'ont été utiles pour comprendre la problématique du sujet ainsi que la nature de certaines variables.

Lors de la deuxième partie du stage, j'ai découvert puis manipulé le langage Python, proche de Perl. La conception de ce script a nécessité des recherches pour comprendre comment fonctionnaient les arguments en ligne de commande ainsi que l'ouverture et l'écriture de fichier. J'ai aussi pu en savoir plus sur la Bioinformatique, les logiciels et les formats utilisés, mais aussi les problématiques abordées et le fonctionnement d'un laboratoire de recherche. Lors de la rédaction de mon rapport, je me suis intitée à LaTeX.

Enfin, j'ai aussi pu échanger avec de nombreuses personnes issues de parcours très différents, notamment sur la recherche en biologie, l'état du marché de l'emploi, ainsi que le fait de faire une thèse, les avantages, les inconvénients ainsi que la difficulté, me donnant des points de vue divers sur les possibilités qui me sont offertes après le master. Ayant trouvé plusieurs pistes intéressantes, telles qu'une spécialisation en base de données, ou encore me diriger vers la recherche, je compte sur le stage de master 1 ainsi que sur les cours à venir pour départager ces différentes possiblités.

4 Lexique

ADN : Acide désoxyribonucléique. C'est une molécule présente dans toutes les cellules vivantes renfermant l'ensemble des informations nécessaires au développement et au fonctionnement d'un organisme. C'est aussi le support de l'hérédité car il est transmis lors de la reproduction, de manière intégrale ou non. Il porte donc l'information génétique et constitue le génome des êtres vivants.

Aromaticité : Un composé organique est dit aromatique quand il satisfait aux conditions suivantes :

- présence d'un cycle comportant un système conjugué, formé de liaisons doubles et/ou de doublets non-liants
- chaque atome du cycle comporte une orbitale p
- les orbitales p se recouvrent (système conjugué), la molécule étant plane au niveau de ce composé cyclique
- la délocalisation des électrons du système entraîne une diminution de l'énergie de la molécule.

Cluster : Analyse de partitionnement de données. Méthode permettant de classer des groupes/documents en fonction de leur contenu.

Corrélation : Intensité de la liaison qui peut exister entre deux ou plusieurs variables.

Gène : Unité d'information génétique composée d'une séquence d'acide désoxyribonucléique

Dendrogramme : Diagramme fréquemment utilisé pour illustrer l'arrangement de groupes générés par un regroupement hiérarchique ou hiérarchisant.

Hydrophobie : Un composé est dit hydrophobe quand il repousse l'eau ou est repoussé par l'eau

Nucléase : Les nucléases sont des enzymes qui coupent les liaisons phosphodiesters des brins d'acide nucléique entre deux nucléotides.

QQ-plot : Quantile-quantile plot (traçage des quantiles d'une distribution versus les quantiles d'une autre distribution). Il est utilisé ici pour comparer les quantiles des variables avec la loi normale.

Variable qualitative : C'est une variable pour laquelle la valeur mesurée sur chaque individu (parfois qualifiée de catégorie ou de modalité) ne représente pas une quantité.

Variable quantitative : Variable représentant une notion de grandeur.

Annexe

A Etude de la corrélation entre CdsCountMean et CdsBruteMean

Le graphique de ces deux fonctions ainsi que leur proximité sur les ACP laisse penser que ces deux variables sont dépendantes l'une de l'autre et qu'il s'agit d'une corrélation linéaire. On peut notamment voir sur le graphique que les points semblent former une droite.

Test statistique de corrélation :

```
> cor.test(data$cdsCountMean,data$cdsBrutesMean)

        Pearson's product-moment correlation

data:  data$cdsCountMean and data$cdsBrutesMean
t = 231.8041, df = 2124, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.9791161 0.9823548
sample estimates:
     cor
0.980803
```

On observe que la p-value du test est largement inférieure à 5%, et que la corrélation estimée est de 98% . On va donc effectuer une régression linéaire sur ces deux variables avec les ligne de commandes suivante :

```
>Droite<-lm(cdsCountMean~cdsBrutesMean,data=data)
>summary(Droite)


Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.587410   0.026436   97.87   <2e-16 ***
cdsBrutesMean 1.166969   0.005034  231.80   <2e-16 ***

Residual standard error: 0.6219 on 2124 degrees of freedom
Multiple R-squared: 0.962,      Adjusted R-squared: 0.962
F-statistic: 5.373e+04 on 1 and 2124 DF,  p-value: < 2.2e-16
```

Nous obtenons donc une formule de régression et un modèle qu'on accepte à α=5%. Nous pouvons maintenant tracer le graphique avec la droite de régression. Les variables cdsCountMean et cdsBrutesMean étant corrélées entre elles, il n'y a pas de raison de garder les deux. J'ai donc retiré cdsBrutesMean.

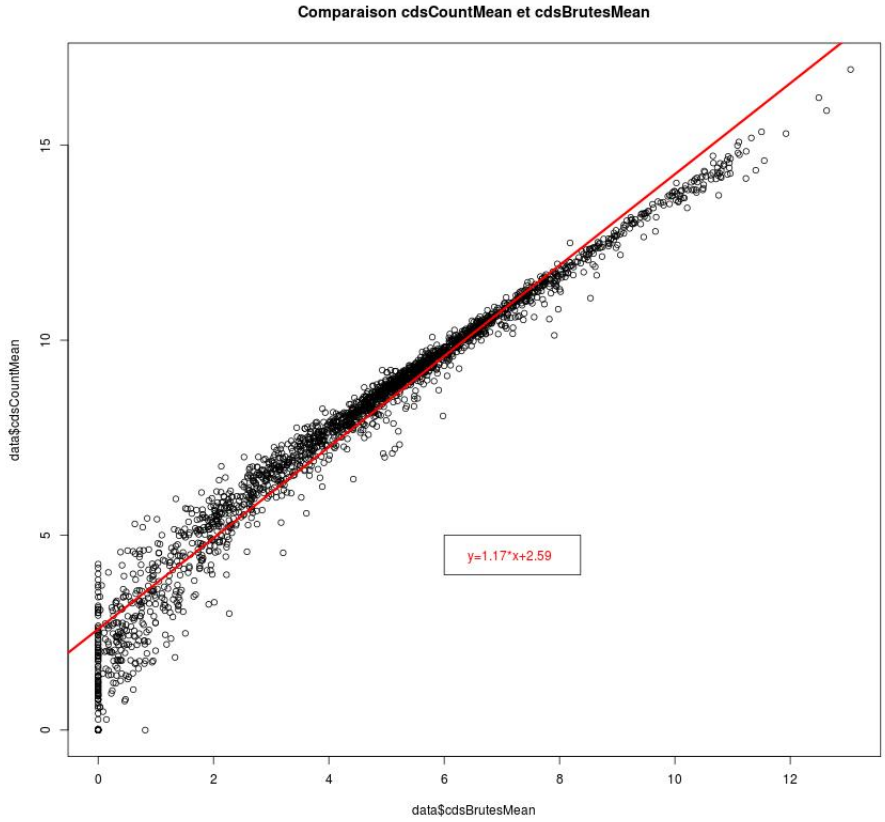


FIGURE A.1 – Régression linéaire des CDSmean

B Script analyse bivariée

```
#Analyse qualitative :

#Creer un jeu de donnees quanti
quanti <- function(d)
{
  num.index<- sapply(d[1,],is.numeric)
  tableau <- d[,num.index]
  tableau
}
tableau<-quanti(data)

#Creer un jeu de donnees quali
quali <- function(d)
{
  num.index<- sapply(d[1,],is.factor)
  tableau <- d[,num.index]
  tableau
}
qualitat<-quali(na.exclude(data))

##Graphiques quali-quali ; prend en parametre le jeu de donnees initial /!\
   part de la deuxieme colonne!! (a ameliorer)

graphQUALI<-function(d){
  par(mfrow=c(2,3))

  for (i in 2:ncol(d)){

    if (is.factor(d[,i])){
      for (j in 2:ncol(d))
        if (is.factor(d[,j])){
      name<-NULL
      name<-paste(colnames(d[j]),"en fonction de",colnames(d[i]), sep=" ")
      plot(d[,i],d[,j], main=name, col=rainbow(nlevels(d[,j])), cex.axis
         =0.4)
        }
    }
  }

}
graphQUALI(data)


#Graphique quali-quanti ; prend en parametre le jeu de donnees inital
qualQuant<-function(d){
  par(mfrow=c(2,2))

  for (i in 2:ncol(d)){
```

```
    if (is.factor(d[,i])==T){
      for (j in 2:ncol(d))
        name<-paste(colnames(d[j]),"en fonction de",colnames(d[i]), sep=" ")
      plot(d[,i],d[,j], main=name, col=rainbow(nlevels(d[,i])))

    }
  }

}
qualQuant(data)


#Matrice de correlation avec couleurs ordonnees : quanti-quanti
library(ellipse)
library(RColorBrewer)
par(mfrow=c(1,1))
xc <- cor(na.exclude(tableau))
tab.c <- cut(xc,breaks = c(-1,-0.6,-0.5,-0.4,0.4,0.5,0.6,1),labels=F)
pal.col <- brewer.pal(5,"RdBu")
#pal.col[4] <- "black" #cette ligne permet de mettre les correlations trop
   petites en noir au lieu de bleu
plotcorr(xc,col=pal.col[8-tab.c], main="Matrice des correlations")


#quali-quanti : le cluster
library("ClustOfVar")
require(ClustOfVar)
tree<-hclustvar(X.quanti=tableau,X.quali=qualitat)
plot(tree)
#quali-quali : test du Khi2 : retourne les pvalues
chi21<-function(data){


  for (i in 1:4){
    for (j in 2:4){
    a<-chisq.test(data[,i],data[,j])
    cat (colnames(data[i]),' ',colnames(data[j]), ' ', a$p.value, '\n')

  }
}
}
chi21(qualitat)
```

C Script ACP

```
data<-read.csv("/home/bwiklund/Documents/BW/donnees.csv", header=TRUE, sep="
    ;", dec=".")


#transformation des variables CDS
data$cdsBrutesMean<-log2(data$cdsBrutesMean+1)
data$cdsBrutesSD<-log2(data$cdsBrutesSD+1)
data$cdsCountMean<-log2(data$cdsCountMean+1)
data$cdsCountSD<-log2(data$cdsCountSD+1)


#fonction pour obtenir un jeu de donnees avec uniquement les variables
    quantitatives ; elle prend en parametre le nom du jeu de donnees initial
quanti <- function(d)
{
  num.index<- sapply(d[1,],is.numeric)
  tableau <- d[,num.index]
  tableau
}

#On cree le jeu de donnees quantitatif qu'on appelle "tableau"
tableau<-quanti(data)
#on renomme les lignes pour avoir les noms des genes sur l'ACP
rownames(tableau)<-data$Gene.id


#on y retire la longueur, la variable deb et fin etc. Pour n'avoir que les
    variables d'importance
tableau$Lgr.CDS..bp.<-NULL
tableau$deb<-NULL
tableau$fin<-NULL
tableau$tAI..FAUX.<-NULL


#(FACULTATIF)Fonction de coloration de l'ACP selon une variable qualitative
    ; elle prend en parametre la variable quali du jeu de donnees initial.
    Par exemple ici, le parametre est data$Cat.
colorisation<-function(variable){
  n<-nlevels(variable)
  colors <- rep(rainbow(n)[2],length(variable))
  for (i in 1:n){
    b<-levels(variable)[i]
    colors[variable==b]<-rainbow(n)[i]

  }
  colors
}

#On attribue a "a" la colorisation
a<-colorisation(data$Cat)
```

```
##Options de coloration :
#Colorisation selective manuelle ; l'etape precedente de coloration doit
   etre effectuee avant d'executer ces lignes et a chaque modification de
   modalite.

a[data$Cat=="UNK"] <-"#F1F7F600" #fait disparaitre la modalite choisie en la
    coloriant en blanc transparent
a[data$Cat!="AMI"] <-"#72797825"#grise toutes les modalites sauf celle
   choisie



#ACP
library("FactoMineR")
par(mfrow=c(1,1))
#ligne de lancement de l'ACP en elle-meme. Genere deux graphiques
   automatiquement
acp=PCA(tableau[,1:ncol(tableau)], scale.=T)

acp$eig #valeurs propres
acp$var$coord #coordonnees
acp$var$contrib #contributions
acp$var$cos2 #cosinus carres

#graphique des valeurs propres
barplot(acp$eig[,1], col="red", main="Eboulis des valeurs propres")

#Ligne de commande pour les graphiques de l'ACP. Le choix peut-etre "ind"
   pour les individus et "var" pour les variables. On fait varier les axes.
plot.PCA(acp, c(1,2), choix="ind", col.ind=a)

#legende en cas de coloration. Penser a changer la variable quali (deux fois
   ). On peut bouger la legende en modifiant x et y
legend(x=7,y=0, legend=levels(data$Cat),  pch="+", col=rainbow(nlevels(data$
   Cat)),text.col="black", cex=0.8)


#(FACULTATIF) Fonction colorisant uniquement une modalite a la fois, le
   reste du nuage etant grise. Elle prend en parametres la variable
   qualitative voulue, et les deux axes choisis pour le graphique. Dessine
   le graphique.
colorTransp<-function(variable,x,y){
n<-nlevels(variable)
par(mfrow=c(1,1))
for (i in 1:n){
        a<-colorisation(variable)
        a[variable!=levels(variable)[i]] <-"#72797830"
        plot.PCA(acp, c(x,y), choix="ind", col.ind=a, type="n")
        legend(x=8,y=-2, legend=levels(variable)[i],col=rainbow(nlevels(
           variable))[i],  pch=16,text.col="black", cex=0.8)
        #chemin d'enregistrement des graphiques
        filen<-paste("/home/bwiklund/Documents/colors/",levels(variable)[i],
           ".jpg",sep="")
dev.print(jpeg, filename=filen, width=1500,height=1000)
}
}
```

```
#exemple d'application
colorTransp(data$Cat,1,2)


#Classification

cp=acp$ind$coord[,1:4] #nombres d'axes choisis
hc.class=hclust(dist(cp),method="ward")
plot(hc.class, hang=-1, cex=0.2) #cluster
rect.hclust(hc.class, k=5, bord="blue") # k=nombre de classes
#classes=rect.hclust(hc.class,k=5)
classes=as.factor(cutree(hclust(dist(cp),"ward"),5)) #changer le nombre de
   classes

#Graphique classification

k=5 #nombre de classes
par(pty="s")
lo<-nrow(tableau)
plot(acp$ind$coord[,1],acp$ind$coord[,2], type="n", main="Groupes selon la
   classification hierarchique")
text(acp$ind$coord[1:lo,1],acp$ind$coord[1:lo,2],dimnames(acp$ind$coord)
   [[1]][1:lo],col=rainbow(k)[as.numeric(classes[1:lo])])
legend(x=6,y=4,paste("G",1:k), pch="+", col=rainbow(k),text.col=rainbow(k))

#voir les groupes en fonction d'une variable quali, ici Cat :
data2=cbind(data,classes)
plot(data2$classes,data2$Cat, col=rainbow(15), main="Cat selon les groupes")
legend("topright", legend=levels(data2$Cat),col=rainbow(nlevels(data2$Cat)),
     pch=16,text.col="black", cex=0.8)


#recuperer les genes des classes dans une liste
cl1<-data2[data2$classes==1,]$Gene.id
cl2<-data2[data2$classes==2,]$Gene.id
cl3<-data2[data2$classes==3,]$Gene.id
cl4<-data2[data2$classes==4,]$Gene.id
cl5<-data2[data2$classes==5,]$Gene.id
```

D Groupes de gènes

Groupe 1

accA accB adaA ahrC argE argF argH aroB atpH bar bglH bglS busAB butA [15] cbr cdd cdsA ceo choS citF citR clpC cmk cobQ copB cydD cysD dgkA [29] dltB dltD dnaC dnaH dxsA ecsB fhuG fmt fruA ftsQ ftsW1 ftsX gadB gidB [43] glgD glpF1 gltQ glyS hexB hisK hpt hslB icd kdgA kdgR kinA kinB kinC [57] lacZ lcnD lgt mapA menX metB1 metF mfd mleP mtsA mutS nadR napB ndrI [71] nrdG nusB oppC optF pabB pbpX pbuX pdhC pfs pi115 pi116 pi124 pi138 pi142 [85] pi143 pi203 pi218 pi246 pi318 pi324 pi333 pi348 pip pknB polA potC prfC prmA [99] proA ps104 ps105 ps209 ps215 ps219 ps301 ps308 ps310 pstC ptcC purE purN purQ [113] rbsB recD recJ ribC ribH rlrB rluB rluD rmaD rpmGA rpmGC rsuA scrK sdaA [127] serB sipL smpB tagD1 tenA thiE thrB topA tra1077B trmD trpA trpE tyrA udk [141] ung uvrB xynB yabE yaiB ybaG ybbC ybdC ybgD ybhB ybhE ybiD ybiH ybiK [155] yccB yccH ycdH ycfC ycgB ycgJ ychD ychG ycjM ydbA ydbF ydcD ydcF yddA [169] ydgE ydiD yeaE yebA yebE yeeB yeeD yeeF yeiG yejD yfaA yfcD yfcI yfhA [183] yfhK yfjH ygaB ygbB ygcA ygfA ygiK yhbF yhcI yheB yheG yhfD yhgA yhhB [197] yhhC yiaB yibF yieF yiiG yijE yjaI yjbC yjbF yjfJ yjgF yjhH yjiF yjjD [211] ykbA ykbE ykcC ykdB ykhH ykiD ykjH ylaG ylbB ylcC ylcF yleC yleE ylfC [225] ylfH yliA yljJ ymeB ymgG ymjE ynaC yncA yneE yngG yniC yniJ ynjD yohH [239] yoiB ypaD ypaG ypbG ypcD ypdA ypdD ypgB yphH ypiB yqaC yqbK yqcD yqeC [253] yqeI yqfA yrfB yrfE yrgI yriA yrjD yrjE ysaB yscA yseH ysfC yshB ysiB [267] ysjF ysxL ytbE ytcA ytdC yteC ytfA ytjA ytjF yucG yudD yueB yugB yuhJ [281] yujB yujD yvaD yvdE yvdG yveD yviA yvjA ywdF yweB ywfB ywiA ywiG ywjH [295] yxaF yxcE yxeB zitP zitS

Groupe 2

[1] accC accD acmA acmB acmC acmD acpD acpS add adhA ahpF aldB aldC aldR amtB amyL [17] ansB apbE apl apu araT arcA arcC1 arcC2 arcC3 arcD1 arcD2 arcT argG argR aroA aroD [33] aroE aroH arsC asd bacA bglA birA2 blt busAA busR butB cadA chiA choQ clsA clsB [49] coaA comX copA copR cpo cshA cstA ctsR cydA cydC cysE dacA dal dapA dcdA dexA [65] dexC dfpB dfrA dhaK dhaL dinF dinG dinP dnaA dnaB dnaD dnaE dnaG dnaN dukA dukB [81] dut enoB exoA ezrA fabD fbp feoA feoB fer fhuB fhuD folB folC ftsA ftsE ftsK [97] gadC galK gatC gidA glgA glgC glgP glnB glnP glnR glpF2 glpT gltD glyA gmk gntK [113] gntR gntZ gpdA gpo groES gshR hemH hemK hemN hexA hflX hisB hisC hisD hisH hly [129] holB hom hprT hrcA hsdM htrA ilvA ilvB ilvD ipd ispA kinF ksgA kupA kupB lacC [145] lacR lctO ldhB ldhX leuB leuC ligA llrB llrE llrF lmrA lmrP lnbA lplL lysA lysP [161] lysQ malQ menD menE menF metB2 mgtA miaA mleR mleS mraY mreC mscL mtlD mtlF murD [177] murE murF murG murI mutM mutX mutY mvaA mycA nah napC nifS nifZ noxE nrdE nth [193] nucA oppA oppD optC optS otcA pacA pacB pacL panE papL parA parC pbp1B pbp2A pbp2B [209] pcaC pcrA pdc pdp pepDA pepDB pepF pepO pepP pepXP pflA pgmB pgsA phnC phoL pi102 [225] pi103 pi104 pi105 pi106 pi107 pi113 pi125 pi127 pi132 pi133 pi139 pi144 pi205 pi208 pi217 pi225 [241] pi226 pi227 pi228 pi229 pi241 pi301 pi303 pi321 pi323 pi329 pi330 pi334 pi337 pi350 pi354 pi356 [257] pi357 pi358 plpA plpB plpC plpD pmpA pmrB pmsR pmsX pnuC2 polC ponA potA potD poxL [273] ppiA preA prfB priA proB proC ps106 ps110 ps112 ps113 ps115 ps116 ps118 ps120 ps121 ps122 [289] ps206 ps212 ps213 ps216 ps220 ps221 ps302 ps303 ps305 ps313 pspB pstA pstB pstE pstF ptbA [305] ptk ptpL ptsK purC purD purH purK purL purM purR pydA pyrB pyrC pyrF pyrZ qor [321] racD radA radC rbsR rdrA recM recN recQ relA rexB rgpA rgpB rgrA rgrB ribG rimM [337] rliA rliB rliDB rlrG rluC rmaA rmaB rmaG rmaH rmaI rmeC rmeD rnc rnhA rnhB rnpA [353] rodA rpe rpiA rpsN2 rpsS ruvA ruvB sdaB serC ssbA sugE sunL thdF thiD2 thiL thiM [369] thrA thrC thyA tpx tra904A tra905 tra983L trmH trpG truA truB trxB2 udp uvrC uxuA uxuB [385] uxuT vacB1 vacB2 xerD xpt xseA xylB xylT xylX xynD xynT yabC yacB yacG yafB yafC [401] yafD yafE yafF yafJ yagA yagB yagE yahB yahD yahI yaiI ybaA ybaB ybaC ybaD ybaF [417] ybaH ybaI ybbA ybbE ybcC ybcG ybcH ybdA ybdH ybdK ybeB ybeD ybeF ybeH ybfA ybfD [433] ybfE ybgB ybgE ybhA ybhC ybhD ybiB ybiE ybiI ybjA ybjD ybjK yccE yccI yccK yccL [449] ycdA ycdE ycdF ycfA ycfB ycfD ycfF ycfG ycfI ycgC ycgF ycgH ycgI ychE ychH yciA [465] yciD yciF ycjB ycjG ycjH ycjI ydaE ydaF ydaG ydbE ydbH ydcB ydcE ydcG yddC ydgB [481] ydgC ydgD ydgF ydgG ydgI ydhB ydhF ydiB ydiC ydiE ydiF ydiG ydjB yeaA yeaB yeaC [497] yeaF yeaG yeaH yebB yebF yecA yecD yedA yedE yeeA yeeE yeeG yeiE yejE yejH yfbI [513] yfbK yfbM yfcF yfcG yfdA yfdC yfdD yffA yffB yfgL yfgQ yfhB yfhL yfiE yfiG yfiJ [529] yfjA yfjB yfjC yfjD yfjF ygaF ygaI ygbE ygbF ygbG ygcC ygdD ygeC ygfE yggA yghB [545] yghC ygiH ygiI ygjB yhbE yhbH yhcA yhcB yhcE yhcK yheA yheD yhfA yhfB yhfC yhfF [561] yhgB yhgC yhgD yhhD yhhE yhhG yhjA yhjB yhjG yiaA yiaC yiaD yibB yibC yibD yibG [577] yicB yicE yidA yidB yidC yidE yieH yifA yigC yihF yiiB yiiH yiiI yijB yijC yijD [593] yijH yjaB yjaD yjaE yjaJ yjbB yjcA yjcD yjcE yjcF yjdA yjeF yjfG yjfI yjgB yjgD [609] yjhA yjhC yjhE yjhF yjiB yjjA yjjF yjjG ykaC ykbC ykbF ykcE ykcF ykcG ykhD ykhF [625] ykhG ykhK ykiE ykiF ykiG ykiH ykjB ykjE ykjF ykjI ykjK ylaD ylaE ylaF ylbA ylbD [641] ylcA ylcD yldC yldE yleB yleD yleF ylfA ylfB ylfD ylfF ylfI ylgB ylgC ylhB yliB [657] yliC yljB yljE yljI ylqL ymbD ymbG ymbJ ymcB ymdE ymfD ymfE ymgB ymgC ymgJ ymhG [673] ymjF ynaB ynaD ynaE ynbB ynbC yndF yndG yneB yneD yneF yneG yneH ynfC ynfH yngB [689] yngE ynhC ynhH yniG yniI ynjH yoaB yoaD yoaH yobA yobC yogJ yogM yoiA yoiC ypaI [705] ypbC ypbD ypcA ypcC ypcH ypdC ypfE ypfF ypgC ypgD yphC yphJ ypiA ypiL ypjA ypjB [721] ypjF ypjG ypjH ypjI yqaB yqaD yqaG yqbF yqbH yqbJ yqcA yqcC yqcF yqdA yqeA yqeB [737] yqfE yqfF yqgF yqgG yqiA yqjB yqjD yqjE yraB yraE yraF yrbA yrbB yrbD yrbI yrcA [753] yrdB yrgE yrgF yrgH yrhH yriB yriC yrjA yrjB yrjF yrjG ysaC ysbB ysbC ysbD yscE [769] ysdC ysdE yseA ysfD ysfG ysgA ysgB ysgC yshA ysiA ysiC ysiD ysiE ysjA ysjE ysjH [785] ytaD ytbB ytbC ytbD ytcB ytcD yteA yteB yteE ytfB ytgB ytgG ythB ythC ytiA ytjG [801] ytjH yuaA yuaB yuaC yuaD yuaE yucF yudA yudF yudG yudH yudJ yudK yudL yueA yueC [817] yueD yueE yueF yufA yugA yugC yuhB yuhC yuhI yuiB yuiD yuiE yujA yujC yujF yujG [833] yvaA yvdB yveA yveB yveC yveH yvfA yvgA yvhA yvhB yviI ywaF ywaG ywaI ywbA ywcC [849] ywdG yweA yweC yweD yweE ywfA ywfD ywfE ywfF ywfG ywfH ywgA ywhA ywhB ywiD ywjA [865] ywjB ywjD ywjE ywjG yxaB yxbA yxbC yxbF yxcA yxcB yxdB yxdC yxfC zitQ zitR

Groupe 3

[1] ackA1 ackA2 acpA adhE adk ahpC alaS als apt arcB argS aroC asnB asnH asnS aspC aspS atpA atpD atpE atpG [22] bcaT bmpA carA carB ccpA celB clpB clpE clpP codY cpsM cspE ctrA cysK cysM dapB ddl def deoB deoC deoD [43] dnaJ dnaK dpsA dtpT dxsB efp enoA eraL fabF fabG1 fabI fabZ1 fabZ2 fadD fbaA femD ffh fhs floL frdC frr [64] ftsH ftsY ftsZ fusA galE gapA gapB gatB glk glmS glmU glnA glnQ gltX gnd greA groEL grpE guaA guaB guaC [85] gyrA gyrB hasC hisS hmcM hslA ileS ilvC infA infB infC ldh lepA leuS llrA llrC lysS mae malE menB metK [106] metS msmK murB murC nadE nagA nagB ndrH nifJ nifU noxA noxB nrdF nusA

nusG obgL optA optB osmC parE pdhA [127] pdhB pdhD pepA pepC pepN pepQ pepT pepV pfk pfl pgiA pgk pheS pheT phnA phoU pi209 pi240 pi317 plsX pmg [148] pmi pnpA proS prsA prsB pta ptcA ptcB ptnAB ptnC ptnD ptsH ptsI purA purB purF pycA pydB pyk pyrE pyrG [169] pyrH pyrP pyrR queA rbfA recA rheA rmlA rmlB rmlC rplA rplB rplC rplE rplF rplI rplJ rplK rplL rplM rplN [190] rplO rplQ rplR rplS rplT rplU rplV rplW rplX rpmA rpmB rpmC rpmD rpmE rpmF rpmGB rpmH rpmI rpmJ rpoA rpoB [211] rpoC rpoD rpoE rpsA rpsB rpsC rpsD rpsE rpsF rpsG rpsH rpsI rpsJ rpsK rpsL rpsM rpsN rpsO rpsP rpsQ rpsR [232] rpsT rpsU secA secG secY serS sodA ssbB tgt thrS tig tkt tpiA trmU trpS trxA trxB1 trxH tsf tuf typA [253] tyrS upp usp45 valS xylA xylH yahA yahG ybdD ybeA ybeC ybfB ybjB ybjJ yccJ ycdB ycdG ycfH ycgD ycgE ycgG [274] yciC yciH ycjA ydbC ydjD yecE yedF yejC yfgG yfgH yfjE ygaJ ygdA yhfE yjaF yjgC yjhD ylaC ymeA ynbE yneC [295] yniH yohD ypcG yphL yqbA yqcG yqeL yqgA yqgE yqjA yraA yrcB yriD yscD yseF ysfB ytaA ytcC ytdA ytdB ytgE [316] ytgF ytgH ythA ytjD yudI yugD yuhE yuiC yvcA yvdD yvdF ywaB ywaH ywiE ywjC ywjF yxaC yxfA yyaL zwf

Groupe 4

[1] agl amyY argB argD argJ aroK aspB atpB atpF birA1 citB citC citD citE clpX cysS [17] dacB dexB dfpA dhaM dltC dnaQ ecsA fabG2 fadA fhuR folD folE folP galM galT gcp [33] gidC glpD glpK gltS hisA hisF hisG hisI hsdR hsdS icaC ilvN ispB kdgK kdtB kinD [49] kinE kinG leuD llrD maa mesJ mtsB mtsC murA1 murA2 noxC noxD nrdD oppB oppF optD [65] pabA pepM pheA phnB phnE pi108 pi109 pi111 pi114 pi118 pi119 pi120 pi123 pi129 pi135 pi137 [81] pi140 pi145 pi147 pi204 pi211 pi215 pi222 pi223 pi230 pi231 pi232 pi233 pi234 pi235 pi236 pi237 [97] pi238 pi239 pi242 pi243 pi244 pi245 pi247 pi248 pi249 pi251 pi307 pi308 pi310 pi319 pi320 pi322 [113] pi325 pi326 pi327 pi328 pi336 pi338 pi339 pi345 pi349 pi355 pmrA potB ppiB prfA ps101 ps102 [129] ps103 ps107 ps108 ps109 ps123 ps207 ps211 ps306 ps307 ps311 ps315 ps316 pth rbsA rbsC rbsD [145] rbsK rexA rgpC rgpE rgpF rheB ribA rmaC rmeA serA smc tagH tagZ thgA thiD1 tra981C [161] trpB trpC trpD trpF uvrA uxaC xylM yabA yacC yaiA ybbB ybeI ybiC ycbC ycbD ycbF [177] ycbH ycbI ycbJ yccF yccG ycdC yceJ ycgA ycjC ycjD ydgJ yeaD yeeC yeiF yejI yejJ [193] yfbB yfcC yfdE yffD yfhG ygaD ygfB ygiC ygiG yhcG yhcH yhdA yhdB yhgE yhjE yihD [209] yiiF yjbE yjdE yjfE yjgE yjiE yjjC ykaF ykiC ykjA ylcE ylcG yldA ylgG yliE yljF [225] yljG ylxQ ymbK ymcA ymcC ynaG ynaH ynbD yngF ynhI ynjE ynjI yogE yogG yohJ yojB [241] yojC ypdB yphI yphK yqcB yqgC yraC yraD yrbG yrbH yrfC yrfD yrgA yrjC yrjI ysdA [257] ysdB yseE yseI ysjC ysjD ysjG ytbA ytdF yteG ytgC yudC yuiA yujE yveF yvjB ywaD [273] ywaE ywbB ywfC yxbE yxcD yxfB

Groupe 5

[1] argC bglR cobC codZ coiA comC comEA comEC comFA comFC comGA comGB comGC comGD crtK dltE ftsW2 fur gadR gltA hisZ [22] icaA icaB lcnC llrG llrH lspA malF malG metA metE mreD mtlR ogt pi101 pi110 pi117 pi122 pi128 pi130 pi141 pi201 [43] pi202 pi207 pi210 pi213 pi216 pi224 pi302 pi316 pi331 pi341 pi343 pi347 pi353 pi359 pi360 ps111 ps114 ps117 ps119 ps201 ps202 [64] ps203 ps205 ps214 ps218 ps304 ps309 ps312 ps314 rarA rcfA rcfB rdrB rliC rlrA rlrC rlrD rluA rmaE rmaF rmaJ rmeB [85] sbcC sbcD secE sigX snf tag tagB tagD2 tagF tagL tagR tagX tagY umuC xylR yabB yabD yabF yacI yahC yaiE [106] yaiF yaiG yaiH yajB yajE yajF yajH ybdG ybdI ybdJ ybdL ybeM ybfC ybgA ybiG ybiJ ycaF ycaG ycbA ycbB yceA [127] yceD yceE yceG ychC yciG ydbD yddB yddD ydgH ydiA yeiD yfbG yfbJ yfcA yfcB yfdB yfeA yfgC yfgF yfhC yfhF [148] yfhH yfhI yfhJ yfiA yfiC yfiD yfiH yfiL yfjG ygaC ygaE ygbD ygdC ygdE ygdF ygeA ygeB ygeD ygfC yghD yghE [169] yghG ygiJ ygjD yhcC yhdC yheE yhhA yhjC yhjF yibE yicA yicC yihA yihB yiiD yiiE yijF yijG yjaH yjdB yjdI [190] yjdJ yjeA yjeD yjeG yjfB yjhB yjjB yjjE yjjH ykaE ykbB ykcA ykcB ykdA ykhE ykhI ykhJ ykiI ykjC ykjJ ylbE [211] yldB yliD yliF yliG yljC yljD yljH ymaB ymbC ymcF ymdC ymgF ymgH ymgI ymgK ymhA ymhC ymiA ynaA ynbA yncB [232] yndA yndB yndC yndD yndE ynfD ynfG yngA ynhA ynhD ynjB ynjC ynjF ynjG ynjJ yoaF yoaG yoaI yofM yogI yogL [253] yohC ypaA ypaC ypaH ypbB ypcB ypfD ypgH yphA ypiE ypiH ypiJ ypiK ypjC yqbI yqcE yqeD yqeH yqfC yqfD yqfG [274] yqhA yrbC yrbE yrbF yreA yreB yreC yreD yreE yrfA yrgG ysaA ysaD ysbA yscB yseB yseC yseD ysiG ysjB ytaB [295] ytcE ytdD yteD ytgA ytgD ytjE yudB yudE yufC yuhA yuhD yuhH yvaB yvcC yvdA yvdC yveE yveG yveI yvfB yviB [316] yviC yviH yviJ ywaC ywdA ywdB ywdC ywdD ywdE yweF ywiB ywiC ywiH ywiI yxbD yxdD yxdE yxdF yxdG yxeA

E Documentation script Python

Doc cp2gff.py

Le but de ce script est de convertir une sortie de segmenteur (typiquement un vecteur de points de rupture dans le génome) en fichier de segments au format GFF (chaque ligne correspondant a un segment). Le premier segment commence par 1 et finit par la première position dans la liste. Le 2e segment commence a la première position de la liste +1 et finit a la 2ème position de la liste, et ainsi de suite jusqu'au dernier segment qui finit par la longueur du génome.

L'outil prendra en entrée un fichier liste de coupures et un fichier de comptage par nucléotide. Les paramètres sont :

-n : indicateur NAME auquel il ajoute un nombre qui s'incrémente a chaque nouveau segment
-s : SOURCE ; la provenance de l'échantillon
-t : TYPE : ARN etc
-b : STRAND : PLUS,MOINS ; l'argument n'est pas sensible à la casse
-c : nom du fichier de comptage par nucléotide (autant de valeurs que de nucléotides dans le génome)
-f : (facultatif) fichier contenant les coupures. Par défaut : jumps.txt.
-o : (facultatif) le nom du fichier de sortie, sans extension. Sinon, le nom du fichier de sortie est créé à partir de la source et de la date.
-k : (facultatif) taille minimale du gène pour qu'il apparaisse dans les sorties. Par défaut, elle est à 5.
-p : classes. elles sont séparées par un underscore. ex : 1_2_3_4_10. Si la dernière classe n'est pas assez grande, une erreur est renvoyée. Ne pas mettre la borne minimale si c'est 0. Un message d'erreur est renvoyé si la borne maximale est trop petite. Les classes apparaissent accolées à la source.

Le script ignore les lignes commençant par "#" ou et les bouts de ligne commentés de la même manière. Si le dièse est en début de ligne, le script utilisera l'entrée avant et l'entrée après cette ligne. Score est la moyenne des comptages par nucléotides entre deux coupures. La valeur de début "1" est ajoutée en début de fichier. Un message "Traitement terminé" apparaît dans le terminal à la fin du script.

exemple : ./cp2gff.py -n cptI -s binInf -t RNA -b PLUS -c deplog.txt -f jumps.txt -o sortie1 -k 6 -p 1_2_3_5_6_9_12

Le résultat du script est un fichier sortie1.GFF contenant :

#./cp2gff.py -n cptI -s binInf -t RNA -b PLUS -c deplog.txt -f jumps.txt -o sortie1 -k 6 -p 1_2_3_5_6_20
#NAME SOURCE TYPE START END SCORE STRAND FRAME
cptI1 binInf1 RNA 1 358 0.77586 + .
cptI2 binInf20 RNA 359 1725 10.07584 + .
cptI3 binInf20 RNA 1726 1882 9.55634 + .
cptI4 binInf20 RNA 1883 3024 10.20612 + .
...

F Code script Python

```
#!/usr/bin/env python
# -*-coding:utf-8 -*
from optparse import OptionParser #pour les arguments
import sys, os
import re # pour la recherche de regex
from time import gmtime, strftime #pour recuperer la date

#ligne de test :
#./cp2gff_V2_1805.py -n cptI -s binInf -t RNA -b PLUS -c test2.txt -f test1.
   txt -o sortie2



#On recupere les parametres passes en ligne de commande
parser = OptionParser()
parser.add_option("-n", "--nom", dest="nom", help ="NAME : nom du segment ;
   s'incremente")
parser.add_option("-s", "--source", dest="source", help ="source")
parser.add_option("-f", "--fic", dest="fic", help ="nom du fichier contenant
    les coupures", default="jumps.txt")
parser.add_option("-t", "--type", dest="typeS", help ="type : ADN,ARN etc")
parser.add_option("-b", "--strand", dest="strand", help ="PLUS ou MOINS")
parser.add_option("-c", "--cpn", dest="cpn", help ="fichier de comptage par
   nucleotide")
parser.add_option("-o", "--out", dest="out", help ="nom du fichier de sortie
   ", default="none")
parser.add_option("-k", "--filtr", dest="filtr", help ="taille minimale du
   gene", default=5)


(options, args) = parser.parse_args()

fic = options.fic
nom = options.nom
source = options.source
typeS = options.typeS
strand = options.strand
cpn = options.cpn
out = options.out
filtre = options.filtr

'''#test boucle for
z=0
for n in range (2,6):
        z=z+1
        print str(n)+" "+str(z)
print ("Z/n= "+str((z/4)))'''
```

```
#fonction permettant de tester la presence de carateres speciaux retourne un
    booleen (a ameliorer)
def caracSpe (chaine):
        speciaux=r"&~#'{(-|e`[U+FFFD][U+FFFD][U+FFFD][U+FFFD]      /;?"
        retour=False
        for lettre in speciaux:
                if lettre in str(chaine):
                        retour=True

        return retour

#Test des parametres NAME, SOURCE et TYPE. Met fin au script avec un message
    d'erreur si la fonction caracSpe retourne True
if caracSpe(nom)==True:
        sys.exit("La variable NAME contient un caractere special")

if caracSpe(source)==True:
        sys.exit("La variable SOURCE contient un caractere special")

if caracSpe(typeS)==True:
        sys.exit("La variable TYPE contient un caractere special")

#Test du parametre signe. Autorise les minuscules et les majuscules.
if strand.upper()=='PLUS':
        strd='+'
elif strand.upper()=='MOINS':
        strd='-'
else:
        sys.exit("Erreur au niveau du signe : 'PLUS' ou 'MOINS' attendu")


#Ouverture du premier fichier (jointures) ; fin du script si echec.
try:
    a=open(fic,'r').readlines()

except :
    sys.exit("Echec a l'ouverture de "+str(fic)+". Verifiez le chemin et le
       nom de fichier.")

#Ouverture du deuxieme fichier (scores) ; fin du script si echec.
try:
    b=open(cpn,'r').readlines()

except:
    print("Echec a l'ouverture du fichier "+cpn+". Verifiez le chemin et le
       nom de fichier.")


#Test de la longueur : la derniere coupure ne doit pas[U+FFFD]tre plus grande que
    le nombre de nucleotides de l'autre fichier
max=len(a)
if int(a[max-1])>len(b):
        sys.exit("Le maximum du fichier de coupure est plus grand que la
           longueur du genome")


#Creation fichier de sortie avec nom de fichier automatique ou non.
```

```
if out!="none":
        sortie = open(out+'.GFF', 'w')
else:
        time=strftime("%d %b %Y", gmtime())
        sortie = open(source+typeS+'_'+time[0:]+'.GFF', 'w')

#recuperation de la ligne de commande pour la mettre dans le fichier de
   sortie
ligne="#"
for elements in sys.argv:
        ligne=ligne+" "+elements

#mise en place des deux premieres lignes
sortie.write(ligne + "\n")
sortie.write("#NAME     SOURCE  TYPE    START   END     SCORE   STRAND
   FRAME\n")


#premiere ligne : rajout du debut "1" et calcul a partir de cette valeur
#la liste commence avec l'element 0
tot=0

nogene=1

if ((int(a[0]))-1)>int(filtre):

        for k in range (0,int(a[0])+1):
                #suppression d'un eventuel commentaire
                b[k]=re.sub(r"\#.*","",b[k])
                tot=tot+float(b[k])

        moy=tot/float(a[0])

        sortie.write(nom+str(nogene)+ " "+str(source)+" "+str(typeS)+"  "+"1
           "+" "+a[0].rstrip()+"       "+str(round(moy,5))+"   "+strd+"
                   .\n")
        nogene=nogene+1


#Reste du fichier
#definition de la coupure "debut de gene" et du compteur
debI=int(a[0])+1
i=1

#debut de la boucle de lecture des coupures
while i!=len(a):

        #recherche d'une ligne commen[U+FFFD]ant par #
        while re.search(r"^\#", a[i]) and i!=len(a):
                i=i+1
        if i>=len(a): break

        #on elimine un eventuel commentaire en fin de ligne
        a[i]=re.sub(r"\#.*","",a[i])

        #remise a zero du total servant a calculer la moyenne
        tot=0
```

```
        #definition de la coupure "fin de gene"
        FinI=int(a[i])

        #boucle calculant la somme des scores ; filtre

        if (FinI-debI)>int(filtre):

                for j in range (debI,FinI+1):
                        b[j]=str(b[j])
                        b[j]=re.sub(r"\#.*","",b[j])
                        tot=tot+float(b[j])

                moy=tot/(FinI-debI+1)

                #ecriture dans le fichier
                sortie.write(nom+str(nogene)+ " "+str(source)+" "+str(typeS)
                   +"  "+str(debI)+"   "+str(FinI)+"   "+str(round(moy,5))+"
                      "+strd+"        .\n")
                nogene=nogene+1
        #incrementation du compteur et redefinition de la coupure de debut
           de gene
        i=i+1
        debI=FinI+1

sortie.close()
print "Traitement termine"
```

G Etude de nouvelles variables

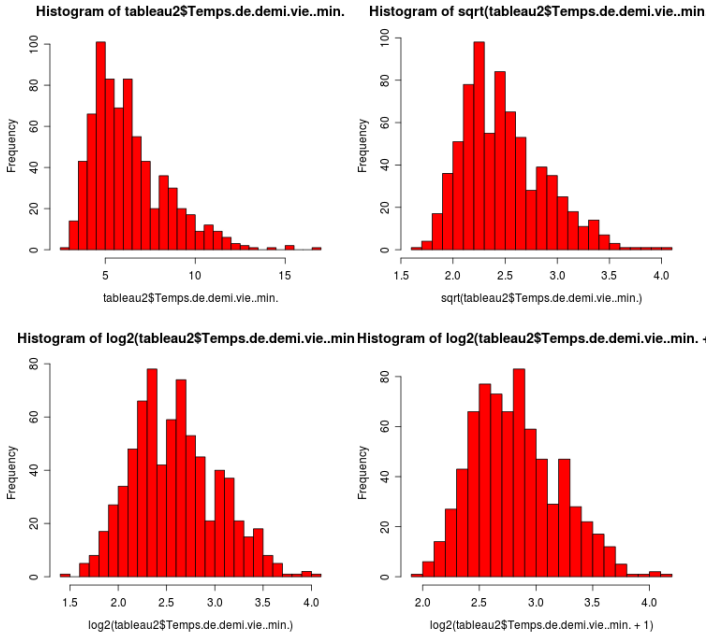


FIGURE G.1 – Différentes transformations de la variable demi-vie : recherche de la symétrie de la variable

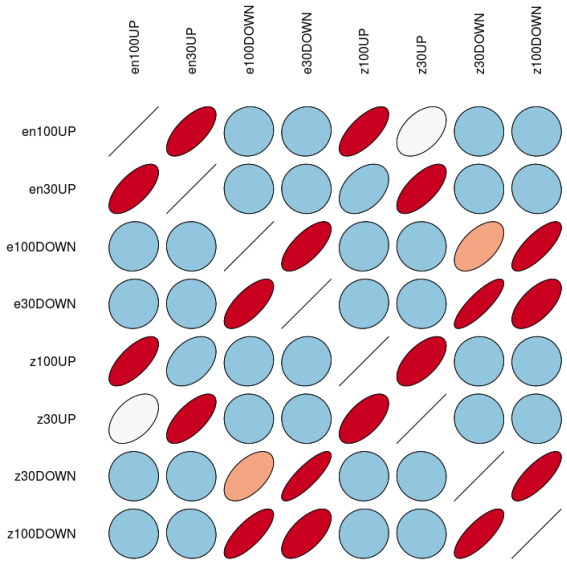


FIGURE G.2 – Matrice de corrélation des dernières variables

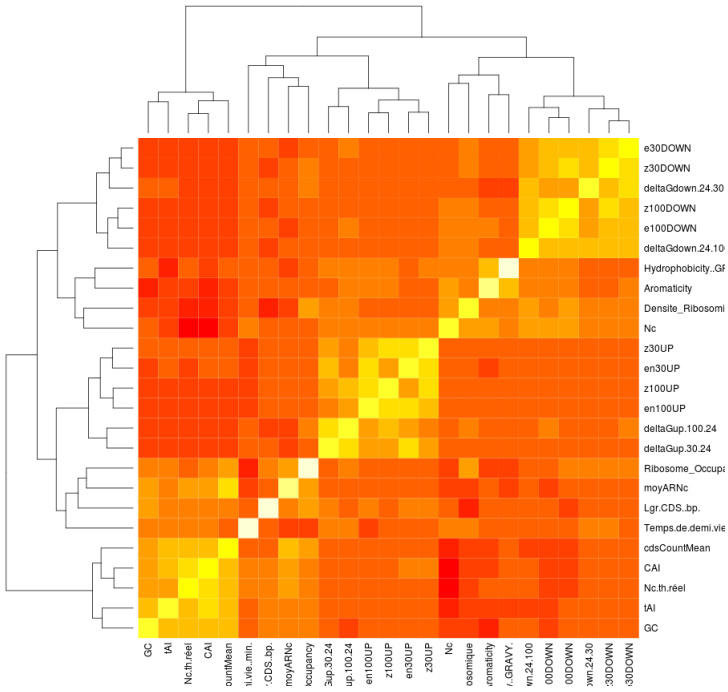


FIGURE G.3 – Heatmap effectuée sur la matrice des corrélations

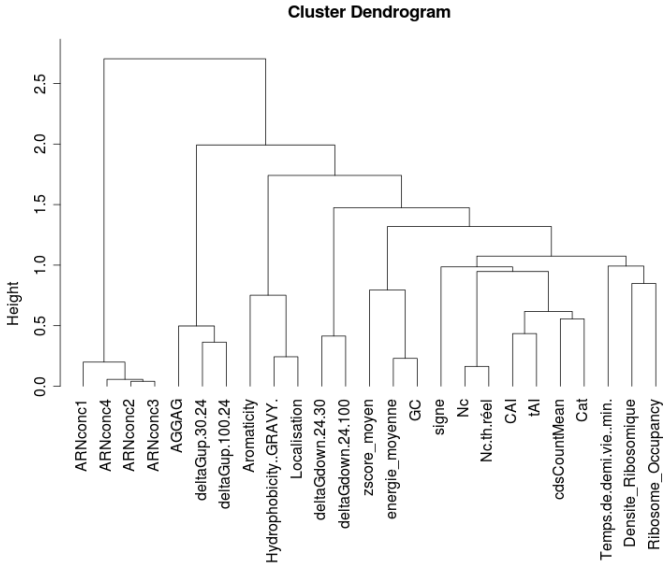


FIGURE G.4 – Cluster réunissant toutes les variables (non transformées)

Résumé

Ce document est un rapport de stage de douze semaines effectué à l'unité Biométrie et Intelligence Artificielle de l'INRA de Toulouse dans le cadre de la validation d'une licence Statistique et Informatique Décisionnelle. Le sujet du stage était l'analyse exploratoire du génome et du transcriptome de *Lactococcus lactis*. Cette problématique s'inscrit dans le projet DEGRADOMICS, ayant pour but l'étude de la dégradation de l'ARN dans les bactéries lactiques en vue de créer un modèle de régulation de l'ARN et d'étudier le rôle de cette régulation dans la résistance bactérienne. Le but du stage était d'intégrer des données de diverses natures dans le but d'effectuer un état des lieux et de laisser des scripts pouvant être réutilisés sur un autre jeu de données où l'ARN aura été dégradé. Parallèlement à cette étude, un script de traitement de sorties de segmenteur a été développé en Python. Il a pour but de créer un fichier au format GFF permettant ainsi de pouvoir comparer les sorties de segmentation entre elles ainsi qu'à une référence (l'annotation).

Abstract

This twelve weeks work placement was made in the Biometric and Artificial Intelligence Unit of the National Institute of Agronomic Research. This teaching practice's aim was to do an exploratory study of a genetic dataset from *Lactococcus lactis*. This work is included in the DEGRADOMICS project which has two objectives : the first objective of DEGRADOMICS is to develop a generic method to study in vivo the mechanisms of RNA degradation and their regulations. The second objective of DEGRADOMICS is to quantify the influence of the RNA degradation process in bacterial adaptation. The dataset study is included in the first part of the project and aims to give a first approach of the RNA's degradation issue. During the second part of the work period, I developed a script to compare the results of different algorithms of segmentation, as a continuation of the first part of my work placement. The goal of this second part was to create a GFF file working on Appolo.