Université de Perpignan Via Domitia
Dpt STID (Carcassonne)
Domaine Universitaire d'Auriac
11000 Carcassonne

Inra Toulouse Midi-Pyrénées
Laboratoire de Génétique Cellulaire
24 chemin de Borde Rouge
31326 Castanet-Tolosan cedex

Étude de la cinétique des réponses au stress chez le cochon

Arthur Gomez

Stage de fin d'études - DUT STID Carcassonne
2012/2013

Tutrice de stage : Elena Terenina & Nathalie Villa-Vialaneix
Tuteur pédagogique : Mouna Kamel

Université de Perpignan Via Domitia
Dpt STID (Carcassonne)
Domaine Universitaire d'Auriac
11000 Carcassonne

Inra Toulouse Midi-Pyrénées
Laboratoire de Génétique Cellulaire
24 chemin de Borde Rouge
31326 Castanet-Tolosan cedex

Étude de la cinétique des réponses au stress chez le cochon

Arthur Gomez

Stage de fin d'études - DUT STID Carcassonne
2012/2013

Tutrice de stage : Elena Terenina & Nathalie Villa-Vialaneix
Tuteur pédagogique : Mouna Kamel

Table des matières

Résumé :
Mon stage de fin de DUT a été effectué dans l'organisme de recherche publique de l'INRA (Institut National de la Recherche Agricole), plus précisément dans l'unité mixte de recherche LGC (Laboratoire de Génétique Cellulaire). Les études menées par ce laboratoire concernent les animaux d'élevage et sont utilisées pour l'alimentation au travers d'analyses sur les phénotypes, et bien sûr d'analyses génétiques.

Les données sur lesquelles j'ai travaillé font partie d'un projet interne au laboratoire et appelé « Stress Express » : il vise à étudier l'impact du stress chez le porc au niveau des expressions de gènes et de diverses mesures phénotypiques. Dans le cadre de ce projet, mon travail a été d'analyser des mesures transcriptomiques et phénotypiques sur des porcs à plusieurs temps après une injection d'hormone ACTH (hormone chez le porc liée au stress). Le projet s'inscrit dans une démarche de sélection génétique pour permettre de sélectionner des animaux plus performants au niveau de l'élevage.

En premier lieu, le but de mon stage était d'extraire et d'analyser des gènes d'intérêt, en les regroupant selon leur cinétique d'expression au cours du temps après injection d'ACTH. Le but de ce travail était de permettre aux biologistes de se focaliser sur un nombre très restreint de gènes pour leur appliquer une démarche expérimentale plus fine sur une population plus large. Ensuite, mes analyses se sont portées sur l'analyse des liens entre les valeurs phénotypiques et les expressions des gènes sélectionnés.

J'ai donc au cours de ce stage assimilé les notions de biologie nécessaires pour débuter les analyses, puis au travers du logiciel R j'ai pu appliquer des méthodes vues en cours et enfin, découvrir de nouvelles analyses plus spécifiques aux problématiques que j'ai dû aborder. Les résultats de mon stage ont permis aux biologistes de disposer d'ensembles de gènes d'intérêts plus restreints sur lesquels focaliser leur travail futur.

Abstract:

My internship took place during 10 weeks in a genetic laboratory based at the INRA centre in Toulouse. INRA is a national public institute of research concerned mainly with agronomy. The role of the genetic laboratory, where I was incorporated, is to measure transcriptomic data in domestic animal species. These data are studied and analysed in order to increase the genetic knowledge about these species.

I was asked to study data coming from an internal project called "Stress Express". This project's goal was to study the effects of stress on pigs. To do so, pigs received a stimulation by means of the injection of an hormone, ACTH, that is related to the answer to stress. Transcriptomic and phenotypic data were acquired at different time steps after the injection.

First, I had to extract important genes which had an expression that was highly changing after the injection. These genes could be used for pig selection and also as a basis for other more specific studies in which more animals can be involved. Second, I studied the relationships existing phenotypic data and gene expression data.

During this internship, I had to understand some notions of Biology, needed to understand the data. I performed the analyses with the R software environment. What I had learned during the two last years was useful to do this work but I also had to understand and perform new methods that were more specific the issues raised by this project.

Première partie .

Introduction

À la fin de ma seconde année de dut STID, j'ai eu la possibilité d'effectuer un stage de fin d'études de 10 semaines (du 02/04 au 07/06) afin de valider la dernière partie de mon diplôme. Ayant fait un stage de première année orienté uniquement en informatique, mais aussi intéressé par le fait d'appliquer les méthodes statistiques acquises en milieu professionnel, je cherchais de préférence un stage orienté vers l'analyse de données. Le domaine de la biologie, sur lequel je n'avais pas encore travaillé est apparu comme particulièrement intéressant, en particulier dans le cadre d'un projet de recherche. J'ai donc été incorporé au Laboratoire de Génétique Cellulaire (LGC), unité du centre de recherche de l'INRA à Toulouse. Ce laboratoire a actuellement plusieurs missions : études du comportement des animaux d'élevages, nouveaux critères de sélection, qualité des viandes pour la consommation, le tout appuyé par des données et analyses génétiques.

En ce qui concerne mon stage, j'ai travaillé sur les données d'un projet visant à étudier l'impact du stress chez le porc ; le nom du projet (porté par le laboratoire dans lequel j'étais intégré) était Stress Express. Les analyses effectuées consistaient à étudier l'évolution des expressions de gènes après injection d'une hormone liée au stress, l'ACTH, et également de comprendre les relations entre ces expressions de gènes et des données phénotypiques.

En premier lieu, je présenterai l'institut INRA, et plus spécifiquement le Laboratoire de Génétique Cellulaire puis les données utilisées et le projet qui a permis leur production. Ensuite, je décrirai les outils utilisés pour mener à bien cette analyse, et la méthode de travail. Enfin, j'exposerai les différents travaux effectués pendant ce stage.

Deuxième partie .

Description de l’entreprise d’accueil

1. L'INRA

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

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

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

Les quatre priorités de recherche de l'INRA sont :

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

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

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

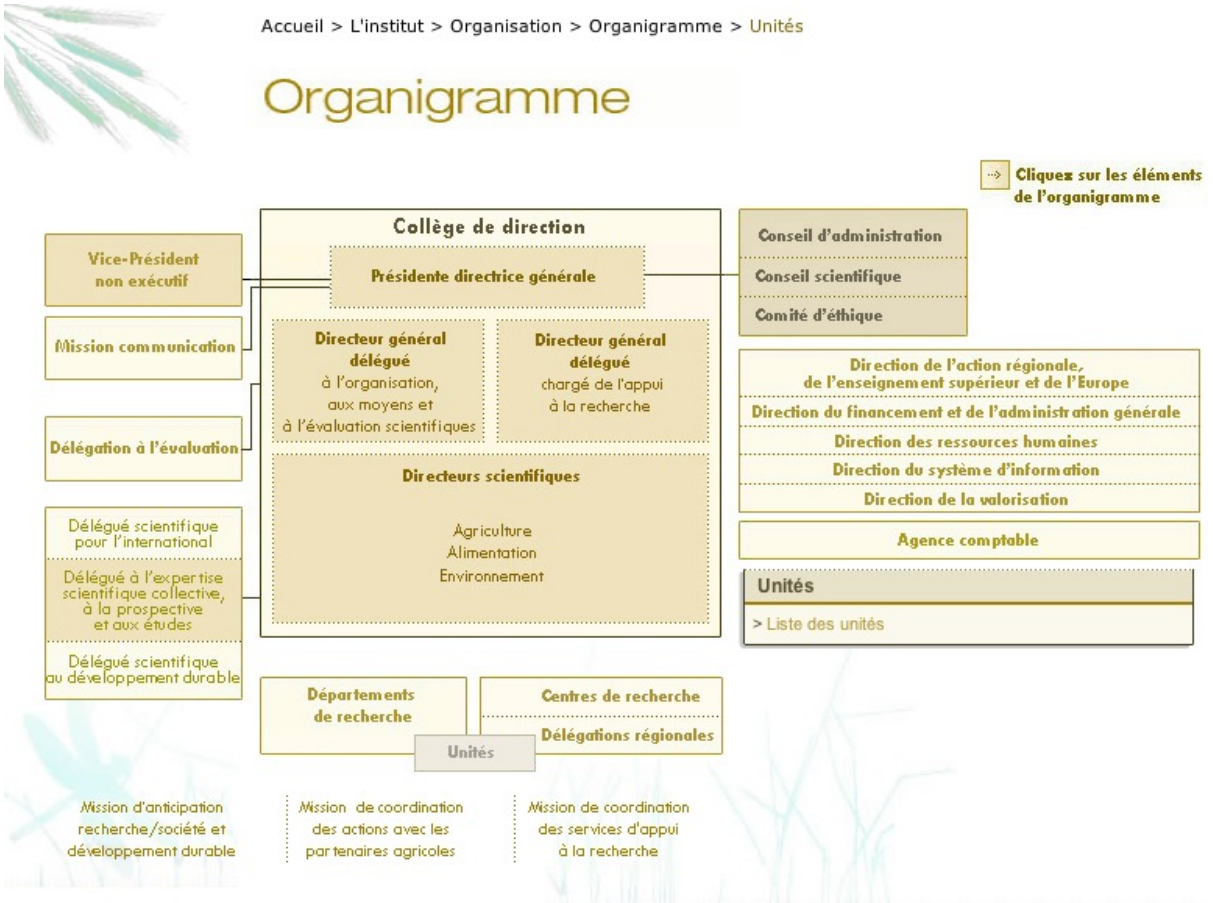


FIGURE 1.1.: Organigramme de l'INRA

2. Le Laboratoire de Génétique Cellulaire

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

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

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

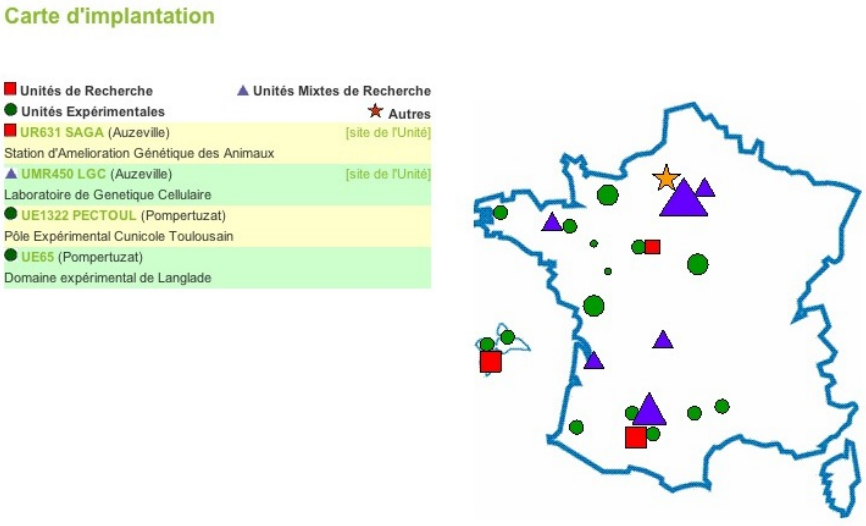


FIGURE 2.1.: Carte de l'implantation du département Génétique Animale

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

J'ai effectué mon stage dans le cadre d'un projet interne au laboratoire, « Stress Express », encadré, côté biologie, par Elena Terenina (LGC, INRA de Toulouse) et Nathalie Villa-Vialaneix côté statistique (MIAT, INRA de toulouse).

Au cours de ce stage, j'ai eu l'occasion de présenter mes travaux lors d'une réunion du groupe de travail « biopuces », qui s'intéresse à l'analyse de données biologiques de grande dimension. Cela m'a permis de présenter mes analyses, de suggérer des améliorations puis de définir les prochains travaux à accomplir du point de vue des biologistes et des statisticiens.

Troisième partie .

Description du stage

3. Problématique

3.1. Notions élémentaires de biologie

Afin de bien comprendre les données sur lesquelles j'ai travaillé durant mon stage, quelques connaissances et rappels de biologie sont nécessaires.

3.1.1. Le stress chez le porc

Le cortisol est une hormone du porc liée au stress et aux facultés d'adaptations de l'animal. Cette hormone a un effet plutôt négatif sur les caractères de production d'un élevage :

- Diminution de la vitesse de croissance
- Diminution de l'efficacité alimentaire
- Diminution de la qualité des carcasses (Protéines/Lipides)

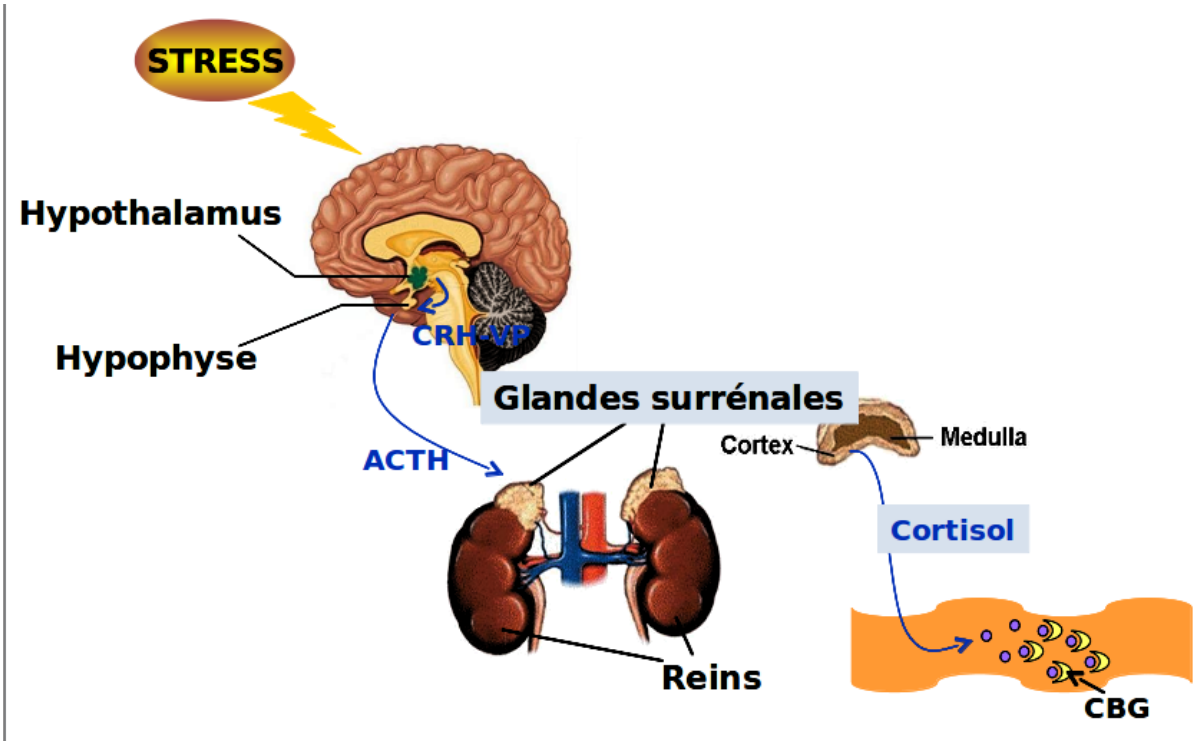


FIGURE 3.1.: Sécrétion du cortisol chez le porc

Cependant, il a aussi des effets positifs au niveau des caractères d'adaptation :

- Résistance au stress

- Résistance aux maladies
- Viabilité des nouveaux-nés

Le cortisol a donc été pendant longtemps un élément de sélection : les animaux ayant une activité corticoïde faible étaient favorisés pour avoir une exploitation plus rentable.

Le processus biologique de sécrétion de cette hormone est schématisé dans la figure 3.1[1]. Suite à un stimulus, l'hypothalamus sécrète la corticolibérine (CRH) et la vasopressine (VP). Ces deux hormones contrôlent la sécrétion par l'hypophyse de l'hormone adrénocorticotrope (ACTH). Le cortex surrénalien, en réponse à l'ACTH, synthétise l'hormone glucocorticoïde, le cortisol (chez le porc). Le cortisol est une hormone synthétisée à partir du cholestérol et véhiculée dans le sang par une protéine de transport, la transcortine ou corticosteroid binding globulin (CBG). De cette manière, il régule l'expression de plus de 500 gènes.

3.1.2. ADN

L'acide désoxyribonucléique (ADN, voir schéma dans la figure 3.2[2]), situé dans le noyau de chaque cellule, porte l'information génétique. De ce fait, l'ADN est directement lié au développement de tout organisme vivant, en particulier par l'intermédiaire de la synthèse des protéines.

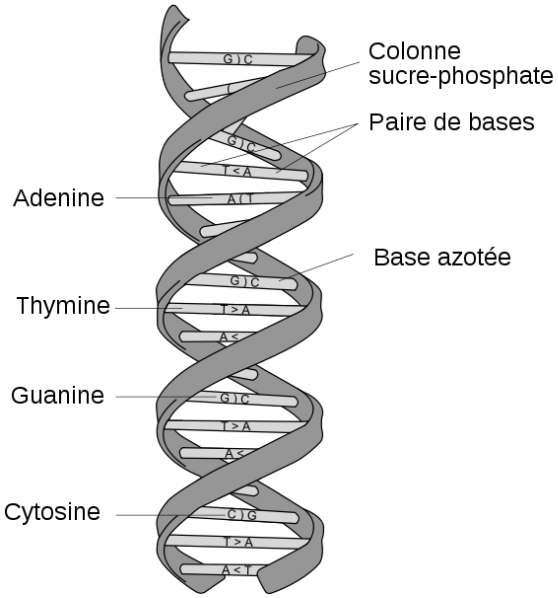


FIGURE 3.2.: Schéma composition de l'ADN

L'ADN est donc composée de 2 chaines nucléotidiques. Chaque nucléotide comprend :

- un groupe phosphate
- un sucre (désoxyribose)
- une des 4 différentes bases azotées présente dans l'ADN parmi la Thymine (T), l'Adénine (A), la Cytosine (C) et la Guanine (G).

[1] Le schéma est attribuable à Pierre MORMEDE LGC INRA Toulouse
[2] L'image est attribuable à Dosto.

La Thymine et l'Adénine sont complémentaires et sont donc toujours associés, tandis que la Cytosine l'est avec la Guanine.

3.1.3. ARN messager et mesure du transcriptome

L'ARN messager est une copie des régions codantes d'un brin d'ADN crée lors de la transcription (voir schéma dans la figure 3.3[3]). L'ARN messager ne contient pas la base azotée la Thymine (T) mais la base Uracile (U). L'ARN messager est ensuite utilisé par les ribosomes afin de synthétiser les protéines lors de la traduction.

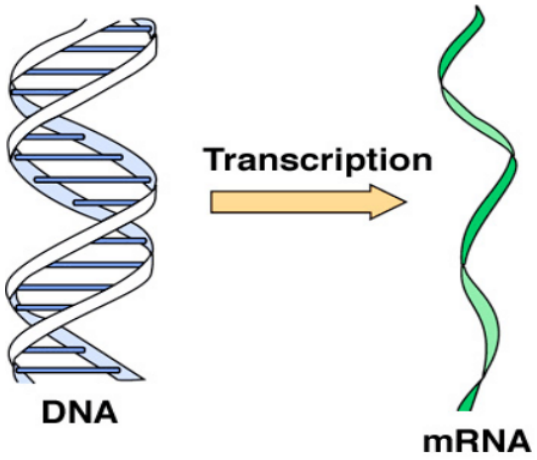


FIGURE 3.3.: ARN

Le transcriptome est l'ensemble des ARN messagers dans une cellule à un moment donné. L'expression d'un gène peut donc être mesurée par la quantité d'ARN messager associé. Afin de mesurer l'expression des gènes, l'analyse transcriptomique se fait de la manière suivante

1. à partir de l'ARN messager extrait des tissus (dans mon cas le sang), il faut obtenir l'ADN complémentaire ;
2. ensuite, il faut déposer l'ADN obtenu sur une plaque : sur cette plaque, appelée « biopuce » (ou « microarray » en anglais), sont déposés plusieurs milliers de gènes transcrits ;
3. enfin, après hybridation, on peut mesurer l'expression d'un gène à partir de sa fluorescence, comme dans la figure 3.4 [4]
4. un controle de qualité est effectué sur les plaques (par exemple pour les sous-puces 1-1 ; 1-2 ; 2-1) pour ne conserver que les mesures de bonne qualité.

[3]L'image provient de Wikimedia Commons et est attribuable à MesserWoland avec des modifications de Dosto.
[4]L'image provient de la plateforme GeT-TRiX de Toulouse

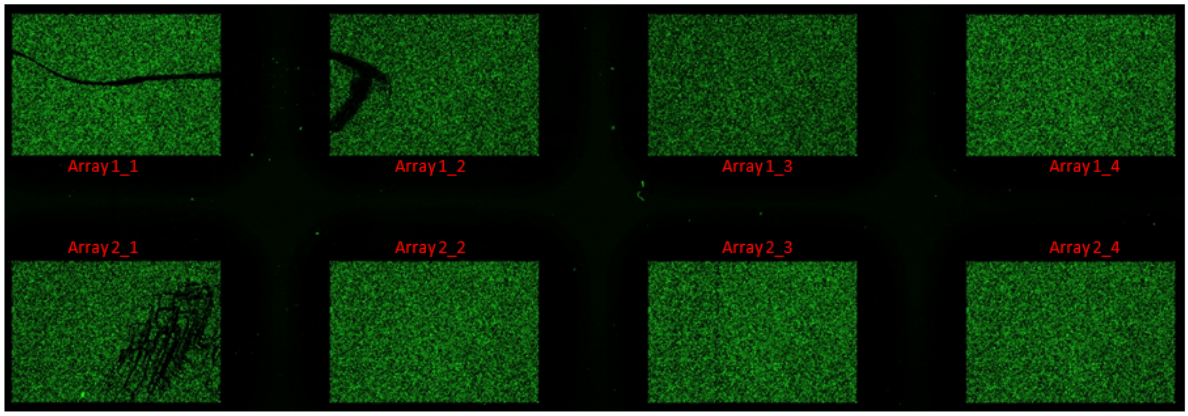


FIGURE 3.4.: Lame constituée de 8 sub-arrays

3.2. Présentation du projet

Le projet dans lequel j'ai été intégré a pour but de recueillir des informations sur les composantes de l'activité biologique du cortisol qui pourraient être utilisées pour la sélection d'animaux plus robustes.

La méthodologie qui a été employée est la suivante :

Le sang de 8 porcs mâles entiers de type charcutier a été prélevé aux temps 0, +1h, +4h, +24 après l'injection d'ACTH (hormone corticotrope, qui stimule exclusivement la sécrétion de cortisol par le cortex surrénalien, schématisé en 3.1). Pour les 30 prélèvements (2 valeurs manquantes), il y aura la mesure de 13 phénotypes différents, puis pour trouver les expressions des gènes, une étude sur le transcriptome.

Les ARN messagers ont été extraits et utilisés avec la puce d'expression Agilent 60K. Après normalisation, l'expression de 43287 gènes (soit 72% des gènes présentes sur la puce) a été mesurée. Bien sur toutes les expressions n'appartiennent pas à des gènes différents. Les gènes représentés plusieurs fois sont appelés doublons ou clones.

Ces gènes ont été testés pour détecter des expressions différentielles au cours de la cinétique (modèles linéaires mixtes).

Une correction de test multiples a été utilisée pour éviter les risques de faux-positifs (celle de Benjamini et Hochberg à 1%). Ainsi nous avons trouvé que l'expression de 110 gènes était significativement différente selon le moment de la cinétique post-injection d'ACTH. Parmi ces gènes il y avait 62 gènes différents.

Les données qui m'ont été fournies pour ce projet étaient donc :

- l'expression de 43286 gènes sur 30 expériences (table `genes`) ;
- les p-values pour un modèle mixte avec effet fixe le temps et effet aléatoire le cochon, des 110 gènes ayant une expression différentielle selon le temps (table `pvalues`) ;
- l'expression des 110 gènes sur 30 expériences (table `sign.genes`) ;
- la valeur de 13 mesures phénotypiques sur 30 expériences (table `phenotypes`) ;
- la description des 30 expériences : identifiant du porc, temps de prélèvement après injection (table `indiv`) ;
- la liste des doublons de gènes sous la forme « identifiant du gène | identifiant du doublon » (au format CSV).

L'objectif du présent projet est d'étudier la réponse du porc à l'hormone corticotrope (ACTH, qui stimule exclusivement la sécrétion de cortisol par le cortex surrénalien). Il s'agit donc de rechercher les groupes de gènes co-exprimés dans ces conditions puis de relier ces résultats aux mesures phénotypiques

Ceci pourra être utilisé pour définir des critères de sélection chez le porc, ou des démarches expérimentales plus fines, sur des populations plus importantes.

3.3. Méthodologie de travail

3.3.1. Logiciel rstudio

R est un logiciel de statistique libre et a été l'outil le plus utilisé durant mon stage. En général, il est constitué visuellement d'une fenêtre pour les commandes, et d'une pour les sorties. Un des avantages majeur est la réutilisation de code R, en particulier au travers de la création de fonctions réutilisables. De plus, comme le logiciel est libre, de nombreux packages et extensions ont été créés par des utilisateurs pour plusieurs problèmes statistiques spécifiques : R met donc à la disposition du statisticien une grande variété de méthodes développées par les chercheurs. C'est donc un outil largement répandu, particulièrement dans le milieu de la recherche publique et il est donc couramment utilisé par les chercheurs de l'INRA.

Une de ces créations est l'environnement de programmation rstudio, qui permet de faciliter l'éxécution des codes et la visualisation des résultats. C'est par l'intermédiaire de cette plateforme que j'ai utilisé R durant mon stage. La capture d'écran de la figure 3.5 permet de visualiser les différentes fenêtres disponibles : pour le code et les commandes, pour les sorties numériques, les sorties graphiques ou les aides, l'espace de travail ou l'historique des commandes.

Enfin, plusieurs packages ont été utilisés au cours de mon stage pour des taches spécifiques :

- **nlme** : qui donne accès aux modèles linéaires mixtes, utilisés lors des calculs de pentes pour les expressions de gènes au cours du temps ;
- **igraph** : utilisé lors des représentations graphiques des classifications ;
- **imputation** : qui permet d'effectuer l'imputation des valeurs manquantes (nous avons utilisé l'approche par plus proches voisins) ;
- **mixOmics** : qui est un package d'analyse multivariée spécifiquement orienté vers les données 'omiques. Nous l'avons utilisé pour effectuer une projection PLS (Partial Least Squares).

3.3.2. Partage des fichiers avec git

Afin d'avoir un meilleur suivi de mon travail, les fichiers et les consignes ont été corrigés par mes encadrants avec l'outil git[5], qui est un outil libre de gestion de version couramment utilisé par les professionnels du développement informatique (et qui est une alternative plus récente à subversion). Cet outil permet à plusieurs utilisateurs d'un serveur de partager des données et des fichiers dans un répertoire, de garder l'historique des

[5] http://git-scm.com/

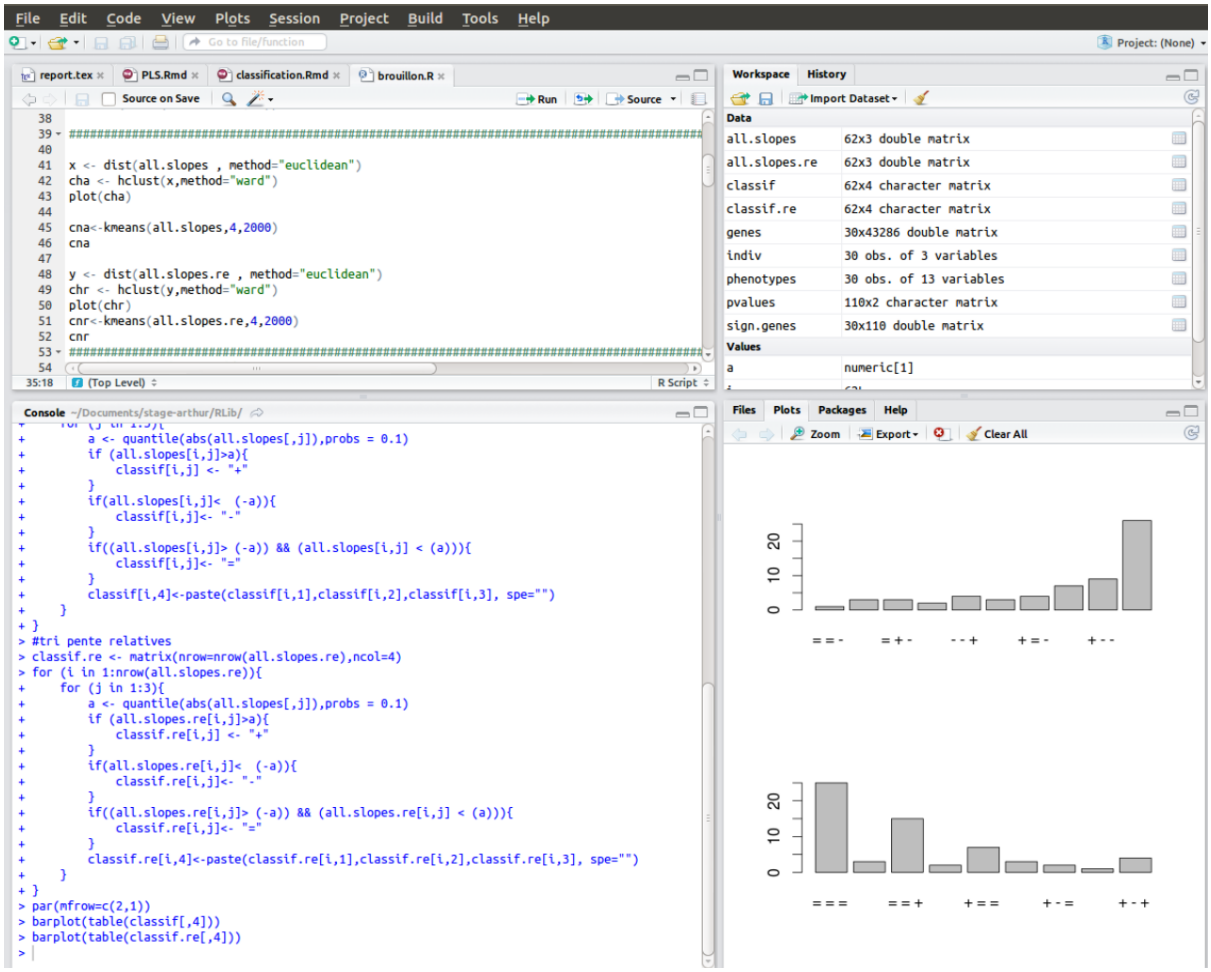


FIGURE 3.5.: Capture d'écran de rstudio

modifications et d'étudier sur plusieurs branches parallèles des développements distincts (je n'ai pas utilisé le système de branches de git durant mon stage). Un dépôt distant, contenant les fichiers et leur historique est créé sur un serveur ainsi que divers dépôts locaux sur lesquels les divers utilisateurs agissent.

Si un utilisateur veut travailler sur un des fichiers du dépôt, il doit procéder de la manière suivante :

- il apporte les modifications souhaitées sur le fichier en question ;
- il effectue ensuite un `git commit`, ce qui a pour effet d'enregistrer la modification dans son dépôt local ;
- quand il veut faire partager ses modifications avec les autres utilisateurs, il effectue un `git push`, la mise à jour est envoyée sur le serveur ;
- les autres utilisateurs du serveur peuvent donc utiliser la nouvelle version du fichier après avoir executé un `git pull`.

Dans le cas où 2 utilisateurs modifient le même fichier, il peut y avoir conflit. Si le fichier est modifié à des endroits différents, la fusion est faite automatiquement. Cependant, si deux utilisateurs ont effectué une modification conflictuelle, git émet une alerte (`unmerged file`) et l'utilisateur doit alors régler manuellement le conflit en choisissant quelle version garder aux endroits où le conflit est apparu.

Il est aussi possible de visualiser et de revenir sur l'historique des modifications effectuées par les utilisateurs ayant accès au serveur, comme sur la capture d'écran de la figure 3.6.

3.3.3. Présentation en Markdown

Le Markdown est un type de syntaxe utilisé pour produire des documents de présentation de manière simple et rapide. Le but de la syntaxe Markdown est d'offrir une syntaxe facile à lire et à écrire : un document formaté selon Markdown devrait pouvoir être publié comme tel, en texte, sans donner l'impression qu'il a été marqué par des balises ou des instructions de formatage.

Avec le package **knitr**, il est possible de produire des documents Markdown incluant du code R et ses résultats : un document (d'extension `.Rmd`) est créé incluant des bouts de code R (appelés « chunks », voir figure 3.7) et, après compilation par **knitr**, un document Markdown est obtenu, compilable en HTML ou en PDF contenant à la fois le code R mais aussi les sorties de celui-ci.

Pouvoir associer sorties graphiques, numériques, code R et commentaires dans un seul fichier simplifie grandement la présentation des résultats. De plus, comme le fichier compile tout le code R à chaque fois, il est possible de demander de conserver dans un dossier cache les résultats du code R pour les opérations nécessitant beaucoup de temps de calcul.

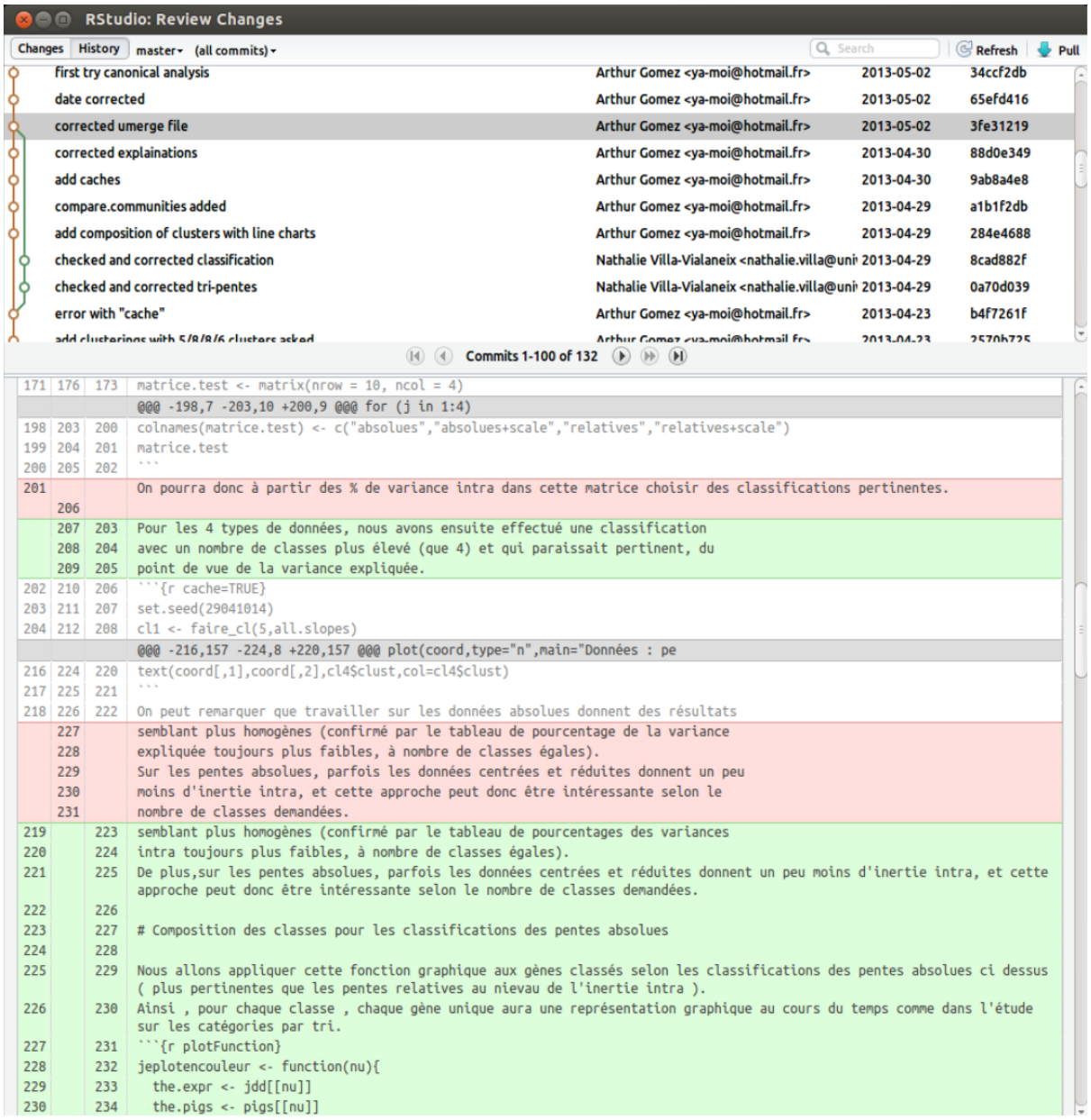


FIGURE 3.6.: Capture d'écran de l'interface de visualisation du dépôt git incluse dans rstudio

FIGURE 3.7.: Capture d'écran d'un fichier Rmd

4. Travail effectué

4.1. Classification des dynamiques d'expression

Comme indiqué dans la problématique, le travail à effectuer était de repérer certains gènes d'intérêt pour des études futures sur une population plus large.

Dans cette première partie de mon stage, l'objectif était de faire des groupes de gènes dont la cinétique d'expression (ou profil d'expression) après injection d'ACTH était similaire. Pour ce faire, on a d'abord modélisé les cinétiques au travers de pentes puis un tri empirique, basé sur des seuils, a été effectué et enfin, une classification automatique, a été également effecutée et comparée à la méthode empirique.

4.1.1. Profils d'expression : utilisation d'un modèle linéaire mixte

Le but de la première partie de mon projet était de classer les gènes selon leurs profils d'expression au cours du temps. Pour ce faire, il a fallu préalablement trouver une méthode pour représenter ce qu'était un profil d'expression. Le document HTML produit au cours du stage pour ce travail est disponible en annexe A.1 et contient notamment les scripts utilisés pour cette analyse. Comme nous avions seulement 4 temps de mesure : 0 (avant l'injection) ; 1h ; 4h ; 24h, nous nous sommes orientés vers un simple calcul de pentes entre deux temps consécutifs mais ce calcul a été effectué en incluant une composante aléatoire par un modèle mixte :

$$\texttt{expr} = a \times \texttt{temps} + b + v_{\texttt{porc}\times\texttt{doublon}} + \epsilon.$$

Le principe de ce modèle linéaire est d'expliquer l'expression `expr` d'un gène par :

- d'une part, un effet déterministe, dû au temps après l'injection `temps` ;
- d'une autre part, un effet aléatoire, dû à une combinaison de l'identité du porc `porc` et du numéro du doublon du gène considéré `doublon` ;
- une erreur Gaussienne centrée ϵ.

Les paramètres du modèles sont donc :

- a, $b \in \mathbb{R}$: deux nombres réels ;
- $(v_{\texttt{porc}\times\texttt{doublon}})$: des variables aléatoires Gaussiennes dont on estime la variance.

De cette manière, au lieu de trier puis classer les gènes directement selon leurs valeurs d'expression, il est possible de trier et classer leur évolution au cours du temps par le biais de 3 pentes : entre 0 et +1h, entre 1h et 4h et enfin entre 4h et 24h.

Afin de prendre en compte un éventuel effet du niveau initial d'expression du gène, il a été calculé pour chaque gène 2 types de pentes, pour les 3 intervalles de temps décrits ci-dessus. Le premier type de pente correspondait au a du modèle (appelé plus tard « pentes

absolues »). Le second type de pentes correspondait au rapport du a de la pente et de l'ordonnée à l'origine au temps 0 (appelé par la suite « pentes relatives »).

Les représentations graphiques pour les 2 types de pentes sont disponibles sous forme de boîtes à moustaches (ou boîtes de dispersion) parallèles pour chaque pas de temps décrit ci-dessus. Elles permettent d'avoir un premier aperçu de la distribution de ces pentes. On peut voir sur les figures 4.1 et 4.2 que les pentes sont globalement plus fortes au premier pas de temps (0 à 1h), puis plus faible au second pas de temps (1h à 4h) et à nouveau plus fortes sur le dernier pas de temps (4h à 24h). Ceci tend à se confirmer comme l'indiquent les sorties numériques 4.1 et 4.2 avec toujours une médiane sur le pas de temps 2 négative (contrairement aux 2 autres).

	Pente 1	Pente 2	Pente 3
Min. :	-0.4496	-3.3243	-0.7848
1st Qu. :	0.1008	-1.2333	-0.1940
Median :	0.5278	-0.7767	0.1783
Mean :	0.5512	-0.8431	0.3656
3rd Qu. :	0.8525	-0.3759	0.6841
Max. :	1.9688	0.6856	3.7981

TABLE 4.1.: Données numériques pentes absolues

	Pente 1	Pente 2	Pente 3
Min. :	-0.05551	-0.38009	-0.17651
1st Qu. :	0.01083	-0.17049	-0.01853
Median :	0.06209	-0.09311	0.02530
Mean :	0.08142	-0.11075	0.03881
3rd Qu. :	0.10242	-0.04013	0.08284
Max. :	0.44280	0.09191	0.42247

TABLE 4.2.: Données numériques pentes relatives

Cependant le but de cette étude est de définir des groupes de gènes en fonction de la cinétique de leur expression au cours du temps. Ces données ont donc été utilisées par la suite pour effectuer un tri selon un seuil empirique, puis des classifications automatiques.

4.1.2. Tri par seuils

Une des méthodes pour classer les gènes par profil d'expression a été basée sur un tri « ad hoc » des profils d'expression, demandé par les biologistes.

Chaque pente a été affectée à une des catégories suivantes : croissante (+), décroissante (-) et stable (=). De cette manière, il était possible de définir un profil pour un gène à l'aide des 3 pentes. Une annexe est disponible pour voir le travail effectué sur ce tri en A.2.

Le seuil à partir duquel on a pu considérer les pentes comme stables a été défini de manière empirique comme étant égal au premier décile des valeurs absolues des pentes. Cependant, ce seuil s'est avéré trop faible et il a été réhaussé au second décile des valeurs absolues des pentes. Une pente dont la valeur absolue était comprise entre ce seuil et son opposé était donc considérée comme constante. Le fait de réhausser le seuil au second

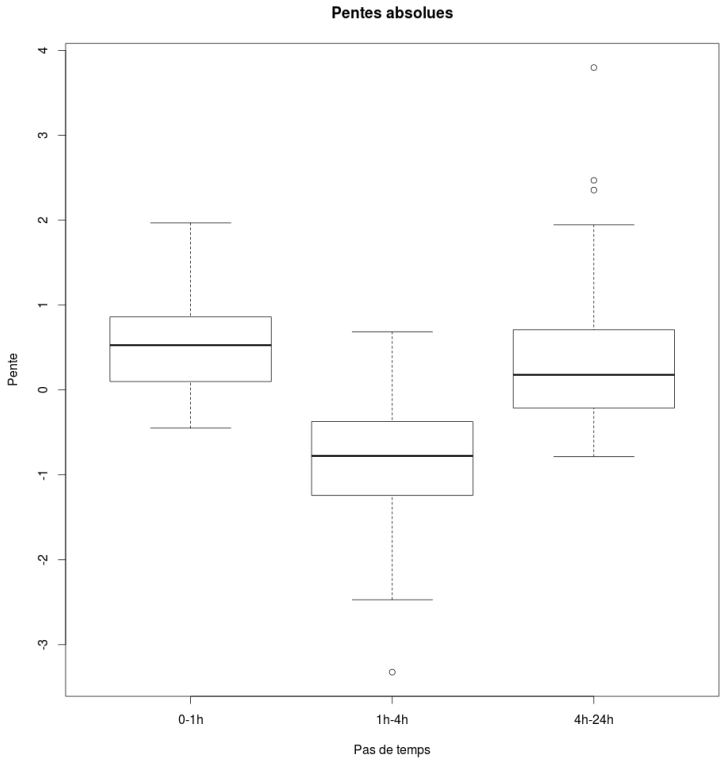


FIGURE 4.1.: Boites à moustaches absolues

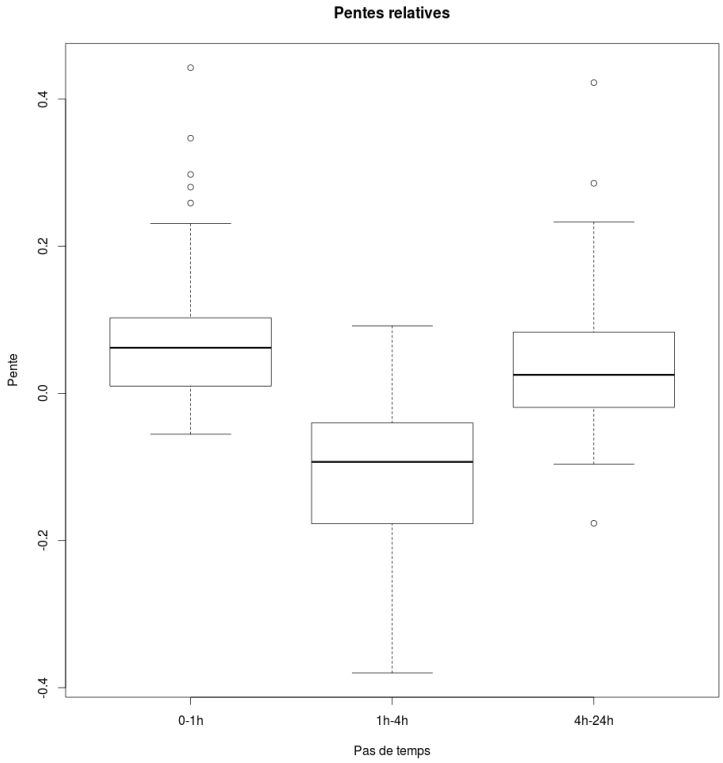


FIGURE 4.2.: Boites à moustaches des pentes relatives

décile a permis d'avoir des catégories plus élevées et des effectifs un peu plus homogènes, comme l'illustrent les tableaux d'effectifs 4.3 et 4.4.

Cette analyse a donc permis d'avoir une première vue des données avant d'exécuter des classifications avec d'autres méthodes. Les tableaux d'effectifs des divers types de profils calculés sur les pentes relatives sont disponibles dans l'annexe A.1.

==-	=-+	=+-	-=+	−+	-+-	+=-	+-=	+−	+-+
1	3	3	2	4	3	4	7	9	26

TABLE 4.3.: Tableau d'effectifs des divers types de profils avec seuil au premier décile (pentes absolues)

==-	==+	=-+	=+-	-=+	−+	-+=	-+-	+==	+=-	+-=	+−	+-+
2	2	7	2	2	2	1	2	1	6	11	4	20

TABLE 4.4.: Tableau d'effectifs des divers types de profils avec seuil au second décile (pentes absolues)

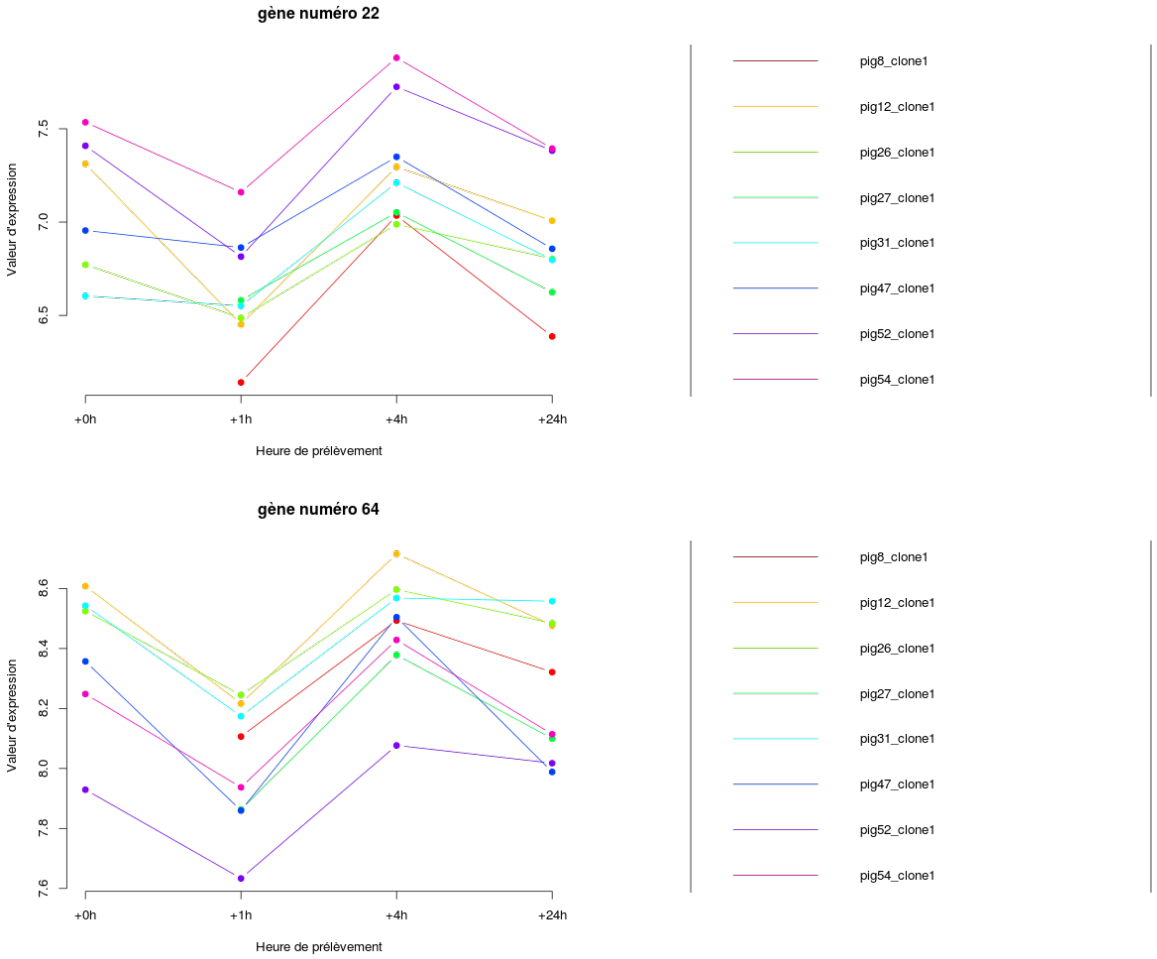


FIGURE 4.3.: Courbes des gènes 22 et 64

Dans l'exemple du tableau 4.4, la catégorie avec une première pente décroissante (-) , une seconde croissante (+) et une dernière décroissante (-) est notée -+- (ici définie sur les pentes absolues) et elle contient seulement 2 gènes : le numéro 22 et le numéro 64 dont les courbes sont disponibles dans la figure 4.3. Pour chaque gène, on a, en abscisse, le temps de prélèvement et, en ordonnée, l'expression du gène. Chaque courbe représente l'évolution de l'expression du gène observé sur un cochon pour un clône. Ces graphiques, organisés par profil, on été réalisés pour chaque gène. Leur nombre étant important, ils ne sont pas produits dans ce rapport mais disponibles dans l'annexe sur CD-Rom.

4.1.3. Classifications

Dans un second temps, l'utilisation d'une méthode de classification automatique a été appliquée sur les pentes. Ces travaux de classification ont été conduits encore une fois dans le but de trouver certains gènes d'intérêts et donc de définir des groupes de gènes ayant des cinétiques similaires. Nous utiliserons donc les pentes créées avec le modèle linéaire mixte. Ainsi, pour chaque gène, nous aurons des valeurs de 3 variables. Les classifications permettront donc de définir, en fonction des similarités entre les pentes des gènes, des groupes de gènes de profils similaires. La suite de cette section s'interesse donc aux diverses classifications qui ont été effectuées. L'intégralité des classifications est disponible dans le fichier en annexe A.3. La composition des classes est, quant à elle, trop volumineuse, mais cependant disponible en annexe numérique.

Les classifications ont été conduites sur les pentes absolues et relatives (3 pentes par individu pour chaque type de classification) avec les données brutes ou avec les données centrées et réduites. Au total, 4 classifications ont donc été effectuées pour chaque méthode utilisée.

Classifications hiérarchiques

En premier lieu, une classification ascendante hiérarchique a été utilisée pour repérer le nombre de classes pertinents. Elles sont illustrées par les dendrogrammes de la figure 4.4. Ces classifications se font à partir des distances euclidiennes et utilisent la méthode de Ward.

On s'aperçoit que pour 3 types de classification sur les 4 étudiées, 4 classes seraient un nombre pertinent. La première étude s'appuiera donc sur une comparaison entre ces classifications, à nombre de classes égales.

Ensuite, en recherchant un nombre de classes plus élevés (le tri « manuel » décrit en section 4.1.2, considérait environ une dizaine de catégories), nous effectuerons une nouvelle étude avec un nombre de classes supérieur à 4 paraissant le plus pertinent. Le nombre de classes retenues pour cette seconde étude sera donc :

- 5 pour les pentes absolues brutes
- 8 pour les pentes aboslues centrées et réduites
- 8 pour les pentes relatives
- 6 pour les pentes relatives centrées et réduites

La section suivante, étudiera donc, pour les 4 types de classification automatique effectuée (pentes absolues ou relatives, données brutes ou centrées réduites), à la fois un

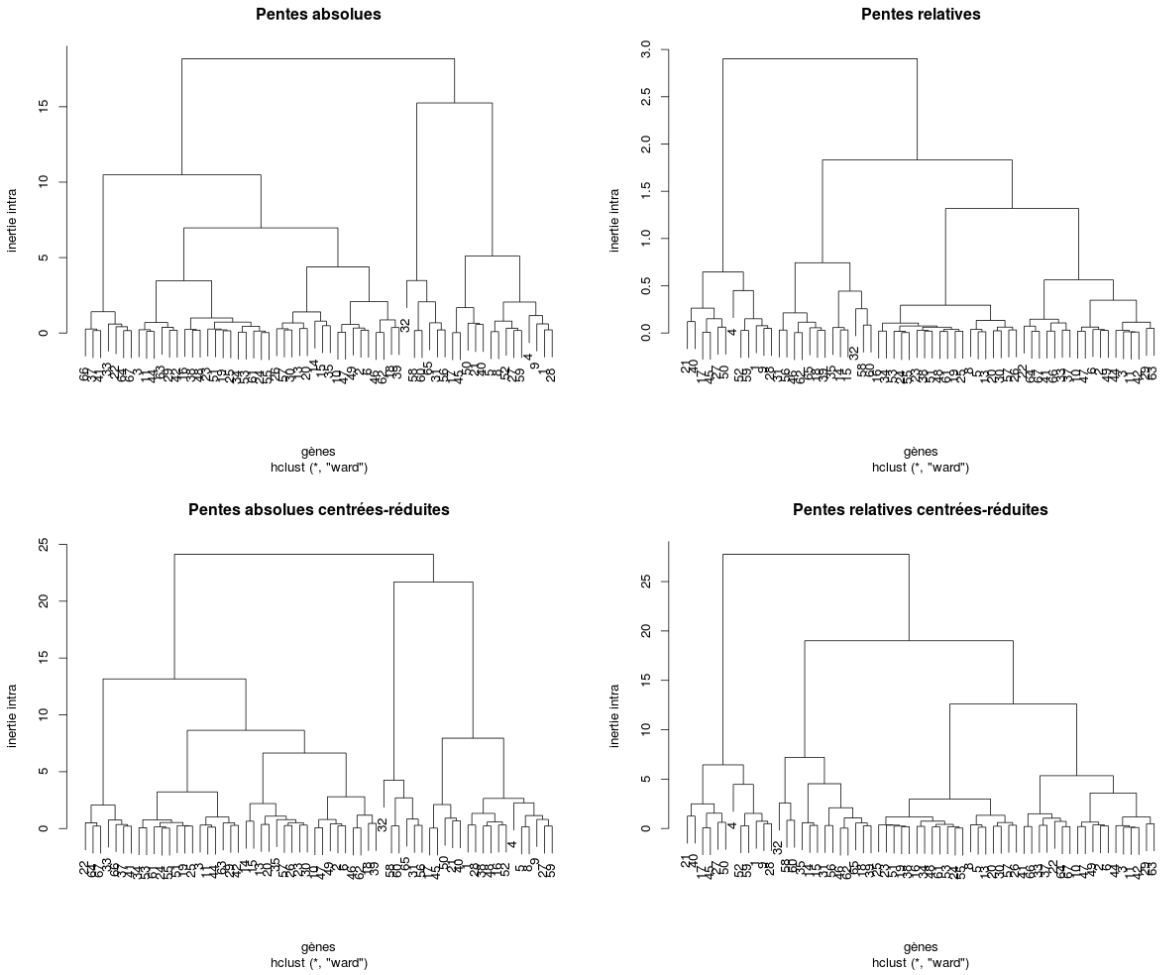


FIGURE 4.4.: Classifications hiérarchiques : dendrogrammes

nombre de classes égal à 4 et un nombre de classes plus élevé. Au total, nous avons donc étudié et comparé 8 types de classification avec la méthode k-means.

Classifications avec méthode des k-means

La méthode des k-means définit des centres de classe (le nombre de centres est choisi a priori par l'utilisateur) comme les valeurs moyennes des individus classés dans la classe correspondante. Les centres sont initialisés de manière aléatoire et, alternativement, chaque individu est relié au centre le plus proche (par exemple le centre 1 sur un des nuages de points en figure 4.6 et les centres sont remis à jour. L'algorithme est stoppé à stabilisation de la classification.

Les k-means souffrent d'un problème de stabilité : comme l'initialisation est aléatoire, les résultats obtenus dépendent de celle-ci et peuvent varier lorsque l'algorithme est répété. Pour pallier cette difficulté, nous avons effectué plusieurs itérations de l'algorithme (2000) et conservé le résultat donnant la plus faible variance intra-groupes.

Le tableau 4.5 permet de visualiser le pourcentage d'inertie intra-groupes (par rapport à la variance de l'échantillon total) pour chaque type de classification avec un nombre de classes allant de 1 à 10. Bien sûr, ce pourcentage diminue lorsque le nombre de classes augmente mais ces résultats permettent de voir que, de manière générale, les pentes

absolues produisent des classifications ayant une inertie intra-groupes un peu plus faible. On s'aperçoit aussi que les données centrées et réduites produisent des classification avec une inertie intra-groupes un peu plus élevée lorsque l'on a un nombre de classes élevé, en comparaison avec les classifications obtenues à partir des données brutes.

Nb classes	absolues	centré/réduit	relatives	centré/réduit
1	100.00	100.00	100.00	100.00
2	58.54	63.68	58.75	60.78
3	37.89	37.49	38.57	39.08
4	27.92	28.18	29.44	29.99
5	21.61	21.67	23.61	24.07
6	17.23	17.36	18.87	18.87
7	13.13	13.82	14.93	14.88
8	10.56	10.62	11.24	11.40
9	9.05	9.26	9.30	9.56
10	7.77	8.06	7.95	8.27

TABLE 4.5.: Comparaison du pourcentage d'inertie intra pour les classifications

Pour visualiser les classifications, nous avons tout d'abord utilisé une approche MDS (Multi-Dimensional Scaling) pour déterminer les coordonnées des observations dans un espace de dimension 2 (le MDS permet de projeter des individus décrits dans un espace multi-dimensionnel dans un espace de dimension plus faible en minimisant la distorsion des distances entre les deux espaces, l'espace initial et l'espace de projection ; voir [1]). La figure 4.5 permet de voir la projection des individus selon cette approche. Celle-ci permet d'avoir la même représentation graphique pour toutes les classifications et donne donc une méthode simple pour voir leurs différences.

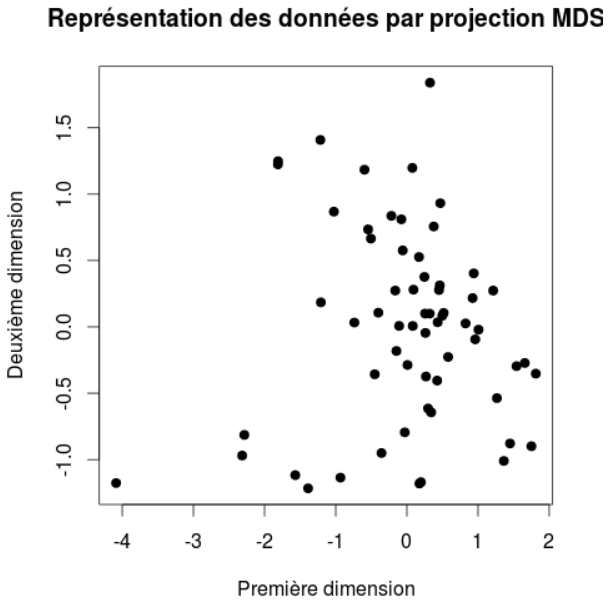


FIGURE 4.5.: Représentation graphique mds

La figure 4.6 donne les classifications obtenues à partir des 4 types de données initiales

pour une classification avec 4 classes (les classes sont représentées avec des couleurs et des symboles différents). On peut voir que ces 4 classifications semblent donner des résultats plutôt cohérents entre eux, même si la classification avec les pentes absolues centrées et réduites est un peu différente des autres. Changer le type de données change la classe de quelques gènes seulement.

Les classifications avec un nombre supérieur de classes sont disponibles sur la figure 4.7. Leur composition exacte, au niveau des gènes, est disponible dans l'annexe numérique puisque trop volumineuse.

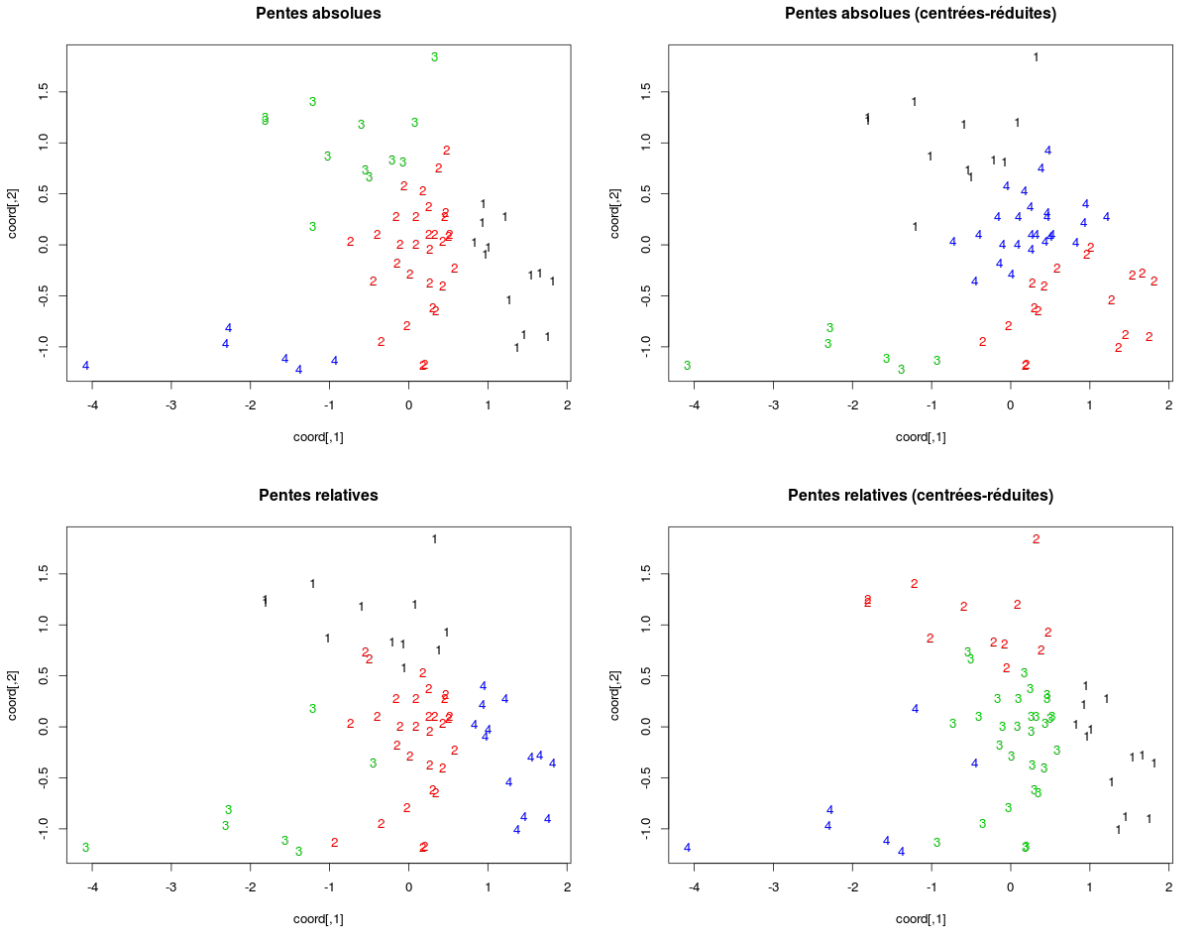


FIGURE 4.6.: Nuages de points de classifications avec 4 classes pour les 4 analyses

Les classifications les plus intéressantes, à savoir celle à 5 classes sur les pentes absolues et celle à 8 classes sur les mêmes pentes mais centrées et réduites ont été comparées à l'aide d'un tableau de contingence 4.6 et d'un indice de comparaison de classification appelé « information mutuelle normalisée » (voir [2]). Cet indice, variant entre 0 et 1, est égal à 1 lorsque les deux classifications comparées sont identiques. L'information mutuelle normalisée peut être obtenue directement à partir de la fonction `compare.communities` du package **igraph** de R. Dans notre cas, la valeur de cet indice était 0,597, qui indique que les deux classifications se ressemble de manière modérée (mais on doit tout de même noter que le nombre de classes est différent dans l'interprétation de cette valeur).

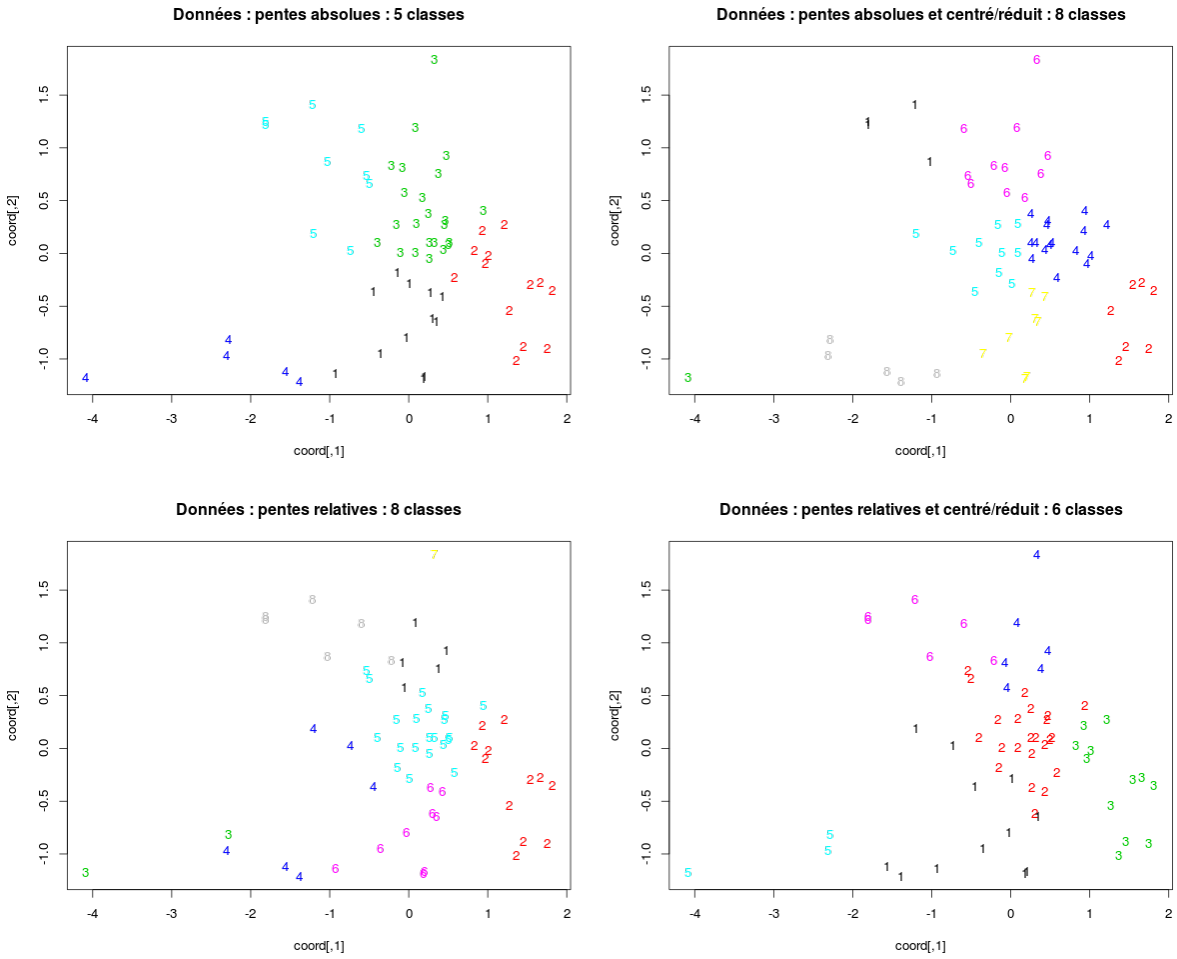


FIGURE 4.7.: Nuages de points de classifications avec nombre de classes plus élevé

	1	2	3	4	5	6	7	8
1	0	0	0	0	3	0	8	1
2	0	7	0	6	0	0	0	0
3	0	0	0	10	5	8	0	0
4	0	0	1	0	0	0	0	4
5	4	0	0	0	2	3	0	0

TABLE 4.6.: Tableau de contingence entre les classes (5 et 8 classes) sur les pentes absolues

4.2. PLS

Cette dernière analyse, Partial Least Square (PLS) vise à relier l'expression de certains gènes avec certains phénotypes. Au niveau de cette étude, il faut voir l'individu comme le prélèvement sur un animal à un temps donné après injection d'ACTH. Ainsi, pour chaque individu, c'est-à-dire pour chaque expérience, nous aurons des mesures phénotypiques ainsi que les expression des 110 gènes qui ont été sélectionnés. Nous avons donc 2 jeux de données portant sur les mêmes individus et la PLS va nous permettre de trouver les plus fortes corrélations entre ces deux jeux de données.

4.2.1. Description de la méthode

La PLS (Partial Least Square) est une méthode qui vise à relier 2 groupes de données. Son principe est de créer des variables latentes, qui sont en fait des combinaisons linéaires des variables d'origine. Les variables latentes du premier axe sont les combinaisons linéaires des deux groupes de variables d'origine qui maximisent leurs corrélations. On procède ensuite itérativement par déflation (c'est-à-dire par soustraction des composants créés) puis par estimation de la variable latente suivante.

Il existe un type spécifique de PLS permettant de limiter le nombre de variables impliquées dans la construction d'une combinaison linéaire donnée : c'est la « sparse PLS » (ou PLS parcimonieuse). Cette méthode est particulièrement bien adaptée au cas où il y a de nombreuses variables et peu d'individus et c'est celle qui est utilisée lors des prochaines analyses.

4.2.2. Préparation des données pour la PLS

Afin de pouvoir exécuter une PLS, il a fallu préparer les données. Nous avons donc supprimé les individus avec trop de valeurs manquantes et imputé le reste des valeurs manquantes.

Les données phénotypiques contenaient le plus de valeurs manquantes. Les individus n'ayant aucune mesure phénotypique ont, en premier lieu, été retirés. Ensuite, l'imputation des valeurs manquantes a été effectuée : pour ce faire, une approche par « plus proches voisins » a été utilisée. La moyenne des 3 plus proches voisins (les observations pour lesquelles les observations sur les variables non manquantes sont les plus proches) a été utilisée pour remplacer les valeurs manquantes.

La préparation du jeu de données au niveau des expressions de gènes s'est résumée à retirer les individus qui n'ont pas été retenus lors de la préparation des phénotypes, et à imputer une valeur sur un individu (une valeur manquante sur 110 gènes pour cet individu).

Les diverses distributions des phénotypes (valeurs centrées réduites) avant et après l'imputation sont disponibles sur la figure 4.8. En comparant la distribution des phénotypes avant et après imputation, on aperçoit l'apparition de plusieurs valeurs très faibles par rapport aux autres (-4). Elles sont environ au nombre de 8, ce qui correspond au nombre de valeurs qui ont été données par imputation à une expérience (8 valeurs manquantes sur 13 phénotypes étudiés pour le cochon 47 au temps 1). Lors de la projection de cet individu sur les axes de la PLS, il est apparu comme étant atypique. Nous avons donc choisi, dans un deuxième temps de le retirer également de l'étude.

FIGURE 4.8.: Boites à moustaches lors de l'imputation des phénotypes

Dans la suite, les données que nous utiliserons donc sont une matrice de phénotypes pour 27 (ou 26) individus (13 variables) et une autre d'expression de gènes pour 27 (ou 26) individus (110 variables).

4.2.3. Analyses PLS

Plusieurs analyses ont été faites par la méthode PLS. La fonction `spls` du package **mixOmics** a été utilisée au cours de cette étude. Le document produit pour ce travail est dis-

ponible en annexe A.4 contenant le code utilisé, les sorties statistiques et représentations graphiques ainsi que les commentaires sur ces analyses.

Création des dimensions

Plusieurs paramètres sont à fournir par l'utilisateur pour effectuer une PLS avec la méthode « sparse PLS » du package **mixOmics** : le nombre de dimensions pour la projection ainsi que le nombre de variables retenu pour chacun des deux jeux de données pour construire les variables latentes des différents axes. Dans notre cas, le nombre de dimensions a été choisi égal à 2, pour avoir des représentations graphiques plus simples à interpréter. Le nombre de variables permettant de définir les variables latentes dans chaque jeu de données a été défini à 15, 10 et 20 variables pour chaque axe, en ce qui concernait les gènes et à 13 variables pour chaque axe (le maximum possible) pour les phénotypes.

Interprétation des résultats

La figure 4.9 représente les corrélations des variables avec les variables latentes des deux axes de la PLS (en haut) et la projection des individus (en bas), pour 10, 15 ou 20 variables sélectionnées.

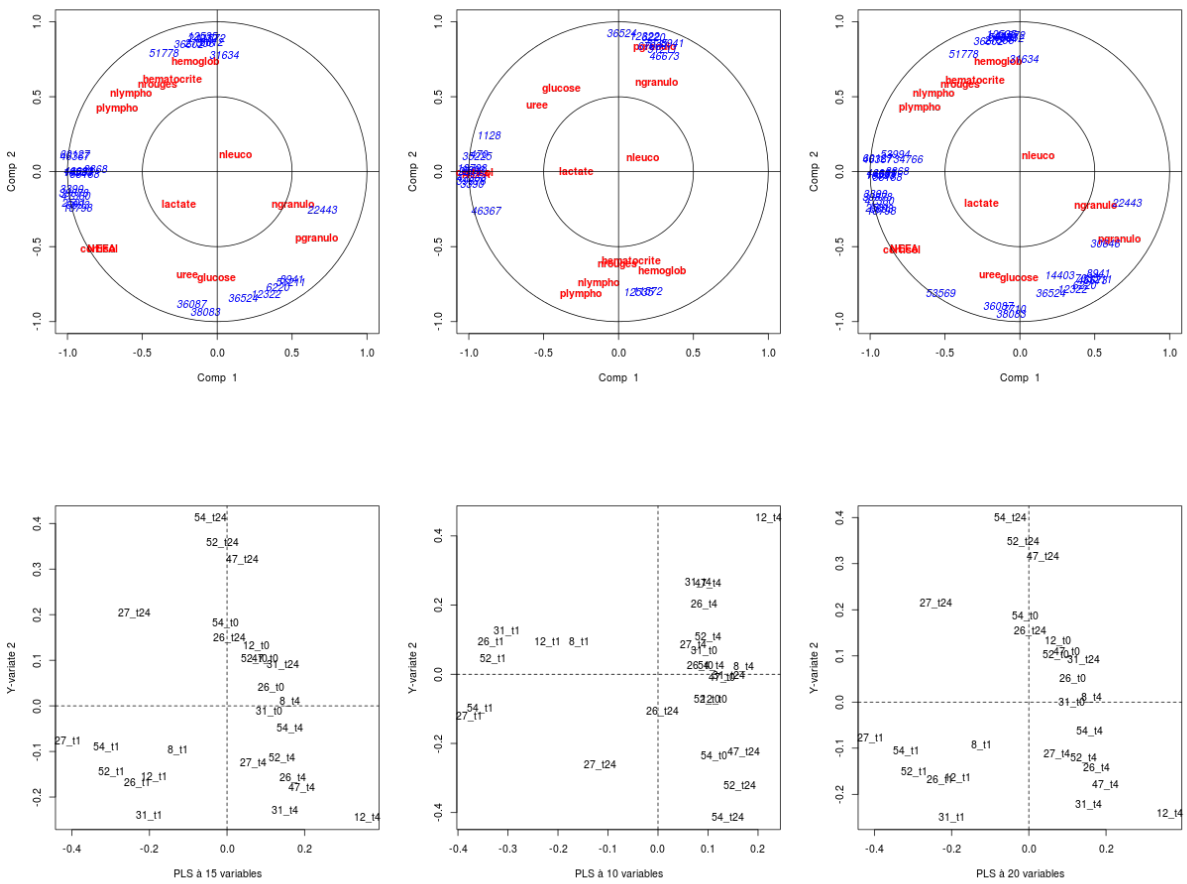


FIGURE 4.9.: Analyses PLS à 15, 10 puis 20 variables

Pour interpréter chaque axe plus facilement, nous avons choisi de conserver les variables dont la corrélation avec la variable latente correspondante avait une valeur absolue supérieure à 0,6. De cette manière, nous pouvons retrouver des ensembles de variables fortement corrélées, comprenant phénotypes et expressions de gènes, sur les 2 dimensions choisies. Les analyses à 15 et 20 variables sont très proches. L'analyse à seulement 10 variables elle est un peu différente au niveau du second axe mais présente quelques similitudes.

On peut donc voir l'axe 1 comme une opposition entre d'un coté les gènes numéro :

16593 ; 3868 ; 58188 ; 14531 ; 46083 ; 6893 ; 3390 ; 46367 ; 47360 ; 39878 ; 18798 ; 2501

et les phénotypes :

cortisol ; NEFA ; plympho

qui ont des valeurs fortes pour les individus au temps t1 (1h après injection), et de l'autre

le phénotype pgranulo et le gène 22443

qui ont des valeurs fortes sur les individus aux temps t4 majoritairement mais aussi un peu les temps t0 et t24 (les valeurs sont un peu plus faibles).

Le second axe lui oppose

les gènes 8941 ; 51211 ; 38083 ; 36087 ; 12322 ; 36524 ; 6220

et les phénotypes glucose ; urée

qui ont des valeurs fortes sur les individus aux temps t1 et t4, aux variables

génétiques : 11572 ; 12535 ; 27556 ; 36602 ; 8612 ; 31634 ; 24932

et phénotypiques : hematocrite ; hemoglob

qui ont des valeurs élevées aux temps t0 et t24.

Au niveau de la distribution des individus, on peut voir que les expériences à 1 heure et 4 heures après injection ont respectivement des positions très spécifiques sur les 2 axes. Les expériences à temps 0 et 24 sont plus proches, hormis certains individus à 24h. Cependant, de nombreux phénotypes à 24h étaient manquants, et il a fallu supprimer ou imputer ces individus. Les résultats pour ces derniers sont donc à prendre avec précaution.

Quatrième partie .

Conclusion

Au début de mon stage, j'ai en premier lieu dû acquérir certaines notions de biologie afin de comprendre les données qui m'ont été fournies. Ensuite, j'ai appris à utiliser les outils informatiques mis à ma disposition afin de pouvoir débuter les analyses.

J'ai de cette manière pu utiliser un modèle linéaire mixte, afin de pouvoir étudier la cinétique d'expression des gènes après une stimulation liée à une injection d'hormone ACTH. Ensuite, les gènes ont été regroupés selon les coefficients trouvés grâce à ce modèle, au travers de divers tris empiriques et méthodes de classification automatique. Enfin, les expressions de gènes ont été reliées aux valeurs phénotypiques par des analyses selon la méthode PLS. Ces résultats sont interprétés par des biologistes du Laboratoire de Génétique Cellulaire à Toulouse et vont être utilisés pour d'autres expérimentations, sur une population plus large avec un nombre plus restreint de gènes cibles. Le but final du projet est de comprendre le fonctionnement génétique lié au stress dans un objectif de sélection chez le porc.

Au niveau professionnel, j'ai eu l'occasion d'effectuer plusieurs analyses sur des données transcriptomiques et phénotypiques. J'ai utilisé plusieurs méthodes d'analyse statistique que je connaissais déjà, mais aussi certaines que je n'avais jamais vues, en particulier la PLS sur laquelle j'ai dû me documenter. J'ai aussi utilisé certains nouveaux outils comme le serveur git (outil de gestion de versions), les documents en syntaxe Markdown (présentation de résultats) et le langage LaTeX(pour la création de documents de rapport et de présentation).

D'un point de vue plus personnel, j'ai trouvé très valorisant d'avoir eu l'occasion de participer à un projet de recherche en biologie. La dimension donnée à la statistique dans des domaines très spécifiques comme la génétique est selon moi très intéressante. J'ai par ailleurs beaucoup apprécié d'avoir pu présenter mes résultats lors de la réunion « biopuces » et d'avoir pu discuter des modifications à apporter ou des analyses à rajouter dans mon travail au cours du stage.

Bibliographie

[1] Cox T.F., Cox M.A.A. (2001) *Multidimensional Scaling*. Chapman and Hall.

[2] Danon L., Diaz-Guilera A., Duch J., Arenas A. (2005) Comparing community structure identification. *Journal of Statistical Mechanics*, P09008.

Cinquième partie .

Annexes

A. Fichiers markdown produits

Les annexes sont composés des documents html produits par l'intermédiaire des fichiers markdown. Ces fichiers sont les analyses et les résultats qui ont été envoyés aux biologistes. Certains documents font plusieurs centaines de pages, et ne sont donc pas publiés ici en entier. Il s'agit en particulier de la composition des classes et catégories obtenues lors des tris de gènes, avec une illustration graphique pour chaque gène. Ils restent cependant disponibles dans l'annexe numérique.

A.1. Calcul des pentes

Etude des expressions des gènes uniques par régression linéaire mixte

Arthur Gomez, dernière version du fichier : 9 avril 2013

Préparation des données

En premier lieu, nous avons donc créé pour chaque gène unique, un vecteur, rassemblant les valeurs d'expression pour chaque expérience où l'on a des observations pour ce gène. Les gènes avec plusieurs représentants ont eux un vecteur de taille plus élevée (avec toutes les valeurs d'expression pour chaque expérience, de chacun des réprensentants), de manière à utiliser toutes les données disponibles lors de la regression. Les temps de prélèvement sont eux sauvegardés dans le vecteur `time.steps`, et les numéros des cochons dans `pigs`.

```
# This file makes a list of vectors Each element of the list
corresponds
# to one unique gene and is a vector having a number of elements
equal to
# the number of clones multiplied by the number of experiments

# Two other lists are constructed: pigs gives the number of the
pig and
# times the time step
load("../data/acth-data.rda")
duplicates <- read.table("../data/duplicates.csv", sep = ";",
header = TRUE)
duplicates$N.gene <- as.factor(duplicates$N.gene)

jdd <- list()
pigs <- list()
time.steps <- list()
for (i in seq_along(unique(duplicates$N.gene))) {
    the.clones <- which(duplicates$N.gene ==
unique(duplicates$N.gene)[i])
    jdd[[as.character(unique(duplicates$N.gene)[i])]] <-
as.vector(sign.genes[,
        the.clones])
    pigs[[as.character(unique(duplicates$N.gene)[i])]] <-
rep(indiv$porcexp,
        length(the.clones))
    time.steps[[as.character(unique(duplicates$N.gene)[i])]] <-
rep(indiv$temps,
        length(the.clones))
}
```

Calcul des pentes

Ensuite, nous avons exécuté une régression linéaire mixte, en fonction du temps de prélèvement et où le facteur aléatoire est composé du cochon et du numéro de doublon (pour les gènes ayant plusieurs représentants)

$$\mathrm{expr} = a \times \mathrm{temps} + b + v_{\mathrm{cochon} \times \mathrm{doublon}} + \epsilon.$$

Il y a donc pour chaque gène unique 3 pentes absolues (entre 0 et 1h, entre 1h et 4h et enfin entre 4h et 24h) calculées à partir des valeurs des différentes expériences où l'on a les observations.

Finalement, en divisant le coefficient de la pente par l'ordonnée à l'origine au temps 0 a/b, on pourra obtenir les pentes relatives dans une autre matrice.

```
library(nlme)
slope.calculation <- function(the.expr, sel.factor, sel.time) {
    obs <- !is.na(the.expr)
    g <- the.expr[obs]
    times <- sel.time[obs]
    tfind <- factor(sel.factor)[obs]
    tdata <- data.frame(g, times, tfind)
    reslm <- lme(g ~ times, random = ~1 | tfind, data = tdata)
    reslm$coefficients$fixed[2]
}
slope.calculation2 <- function(the.expr, sel.factor, sel.time) {
    obs <- !is.na(the.expr)
    g <- the.expr[obs]
    times <- sel.time[obs]
    tfind <- factor(sel.factor)[obs]
    tdata <- data.frame(g, times, tfind)
    reslm <- lme(g ~ times, random = ~1 | tfind, data = tdata)
    reslm$coefficients$fixed[1]
}

super.slopes <- function(nu) {

    the.expr <- jdd[[nu]]
    the.pigs <- pigs[[nu]]
    the.times <- time.steps[[nu]]
    tnt <- time.steps[[nu]]

    nb.clones <- round(length(the.pigs)/30)
    the.factor <- factor(paste("pig", the.pigs, "_clone",
rep(1:nb.clones, rep(30,
        nb.clones)), sep = ""))

    tnt <- rep(1, length(the.times))
    tnt[the.times == 1] <- 2
    tnt[the.times == 4] <- 3
    tnt[the.times == 24] <- 4

    all.slopes <- vector(length = 4)
    for (sel.t in 1:3) {
        sel.obs <- which(tnt == sel.t | tnt == (sel.t + 1))
        sel.factor <- the.factor[sel.obs]
        sel.times <- tnt[sel.obs]
        all.slopes[sel.t] <- slope.calculation(the.expr[sel.obs],
sel.factor,
            sel.times)
        if (sel.t == 1) {
            all.slopes[4] <- slope.calculation2(the.expr[sel.obs],
```

```
sel.factor,
                sel.times)
        }
    }
    all.slopes
}
all.slopes <- t(sapply(names(jdd), super.slopes))
all.slopes.re <- sweep(all.slopes[, 1:3], 1, all.slopes[, 4], "/")
all.slopes <- all.slopes[, 1:3]
save(all.slopes, file = "../results/all_slopes.rda")
save(all.slopes.re, file = "../results/all_slopes_re.rda")
```

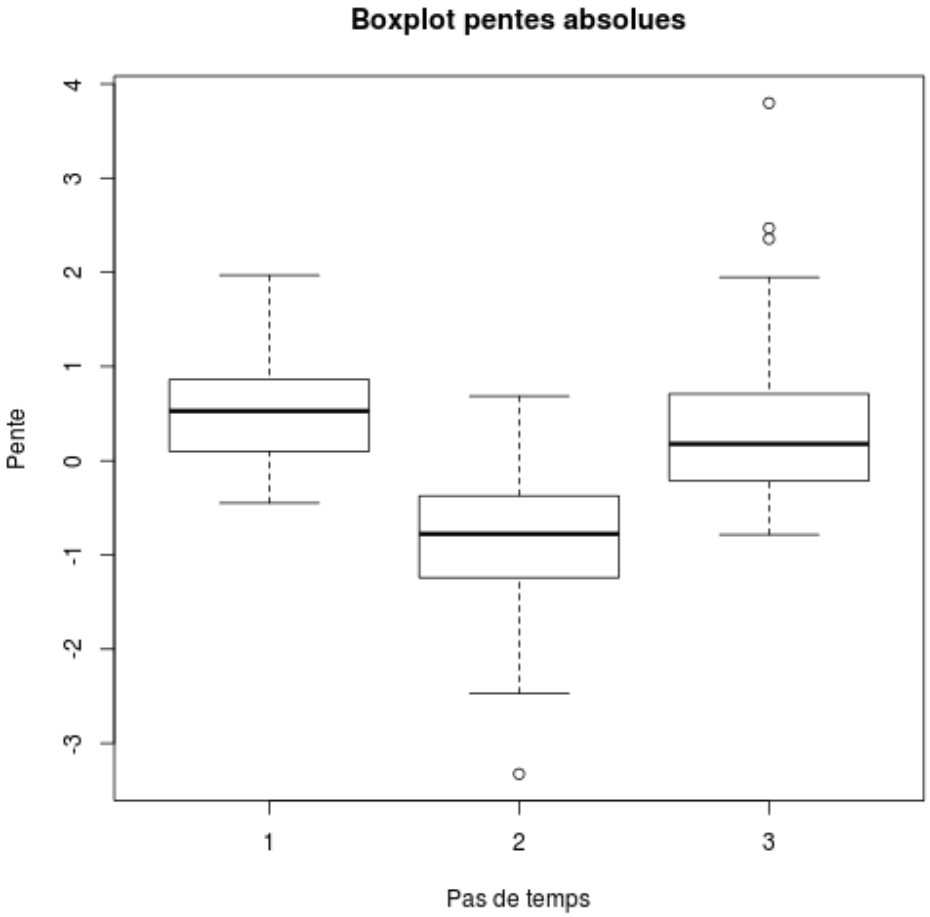
Brève analyse des pentes

Voici donc un premier aperçu des données pour les 2 types de pente :

```
summary(all.slopes)
```

```
##        V1                V2                V3          
##  Min.   :-0.450   Min.   :-3.324   Min.   :-0.785  
##  1st Qu.: 0.101   1st Qu.:-1.233   1st Qu.:-0.194  
##  Median : 0.528   Median :-0.777   Median : 0.178  
##  Mean   : 0.551   Mean   :-0.843   Mean   : 0.366  
##  3rd Qu.: 0.853   3rd Qu.:-0.376   3rd Qu.: 0.684  
##  Max.   : 1.969   Max.   : 0.686   Max.   : 3.798
```

```
boxplot(all.slopes, xlab = "Pas de temps", ylab = "Pente", main =
"Boxplot pentes absolues")
```

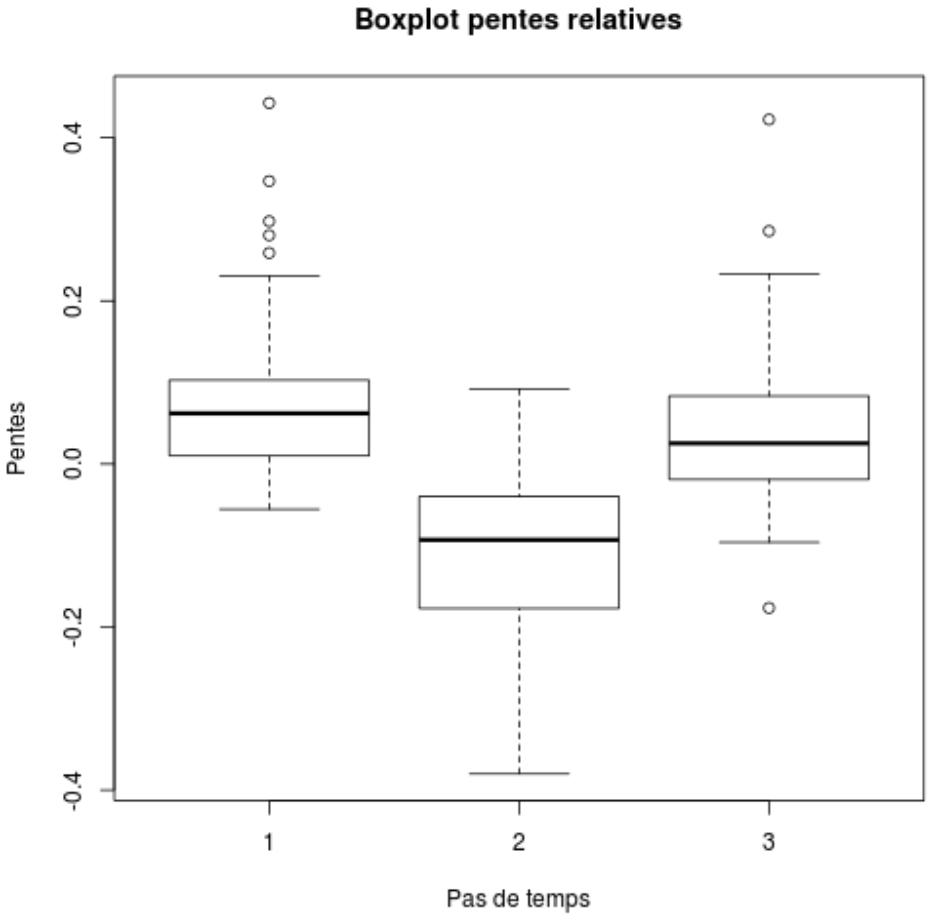


D'une manière générale, les pentes entre les temps 1 et 2 ($t = 0$ et $t = 1$) et les pentes entre les temps 3 et 4 (($t = 4$ et $t = 24$) sont plus fortes que les pentes entre les temps 2 et 3.

```
summary(all.slopes.re)
```

```
##        V1                 V2                 V3          
##  Min.   :-0.0555   Min.   :-0.3801   Min.   :-0.1765  
##  1st Qu.: 0.0108   1st Qu.:-0.1705   1st Qu.:-0.0185  
##  Median : 0.0621   Median :-0.0931   Median : 0.0253  
##  Mean   : 0.0814   Mean   :-0.1107   Mean   : 0.0388  
##  3rd Qu.: 0.1024   3rd Qu.:-0.0401   3rd Qu.: 0.0828  
##  Max.   : 0.4428   Max.   : 0.0919   Max.   : 0.4225
```

```
boxplot(all.slopes.re, xlab = "Pas de temps", main = "Boxplot
pentes relatives",
    ylab = "Pentes")
```



Nous allons ensuite utiliser les données de ces pentes pour essayer de classer les différents gènes significativement exprimés afin de définir des profils pour chaque gène, illustrant l'évolution de leur expression au cours du temps.

A.2. Tri empirique des profils d'expression

Tri des gènes selon des profils de pente

Arthur Gomez, dernière version du fichier : 23 avril 2013

Description du travail et de la méthodologie

Nous avons commencé donc à définir 3 types de pente :

- les pentes nulles ou quasi nulles (dont la valeur absolue est inférieure au premier décile des pentes absolues) : “ = ”
- les pentes positives (supérieures au premier décile): “ + ”
- les pentes négatives (inférieures à 0, dont la valeur absolue est supérieure au premier décile) : “ - ”

Ainsi, pour chaque gène, nous définierons un profil qui prend en compte ses 3 pentes : par exemple “+-+” pour définir une première pente postive, une seconde négative et une dernière à nouveau positive.

Une première étude a montré qu'il y avait beaucoup de pentes peu élevées considérées comme croissantes, ou décroissantes, ce qui entrainait des profils aux effectifs très hétérogènes. Afin de pallier ce problème, le seuil pour définir les pentes nulles a été relevé, à 15% des pentes les plus faibles (en valeur absolue), puis au second décile des valeurs absolues des pentes.

Cette étude a donc été effectuée en premier lieu pour les pentes absolues :

```
load("../data/acth-data.rda")
duplicates <- read.table("../data/duplicates.csv", sep = ";",
header = TRUE)
duplicates$N.gene <- as.factor(duplicates$N.gene)

jdd <- list()
pigs <- list()
time.steps <- list()
for (i in seq_along(unique(duplicates$N.gene))) {
    the.clones <- which(duplicates$N.gene ==
unique(duplicates$N.gene)[i])
    jdd[[as.character(unique(duplicates$N.gene)[i])]] <-
as.vector(sign.genes[,
        the.clones])
    pigs[[as.character(unique(duplicates$N.gene)[i])]] <-
rep(indiv$porcexp,
        length(the.clones))
    time.steps[[as.character(unique(duplicates$N.gene)[i])]] <-
rep(indiv$temps,
        length(the.clones))
}
```

```
load("../results/all_slopes.rda")
classif <- matrix(nrow = nrow(all.slopes), ncol = 4)
for (i in 1:nrow(all.slopes)) {
    for (j in 1:3) {
        a <- quantile(abs(all.slopes[, j]), probs = 0.1)
        if (all.slopes[i, j] >= a) {
            classif[i, j] <- "+"
        }
        if (all.slopes[i, j] <= (-a)) {
            classif[i, j] <- "-"
        }
        if ((all.slopes[i, j] > (-a)) && (all.slopes[i, j] < (a)))
{
            classif[i, j] <- "="
        }
        classif[i, 4] <- paste(classif[i, 1], classif[i, 2],
classif[i, 3],
            sep = "")
    }
}
table(classif[, 4])
```

```
##
## ==- =-+ =+- -=+ --+ -+- +=- +-= +-- +-+
##   1   3   3   2   4   3   4   7   9  26
```

La classification ainsi obtenue étant très déséquilibrée et ayant donné des classes dont les pentes notées “+” avaient une variabilité importante, nous avons reproduit l'étude en seuillant au deuxième décile.

Seuillage au deuxième décile

```
classif <- matrix(nrow = nrow(all.slopes), ncol = 4)
for (i in 1:nrow(all.slopes)) {
    for (j in 1:3) {
        a <- quantile(abs(all.slopes[, j]), probs = 0.2)
        if (all.slopes[i, j] >= a) {
            classif[i, j] <- "+"
        }
        if (all.slopes[i, j] <= (-a)) {
            classif[i, j] <- "-"
        }
        if ((all.slopes[i, j] > (-a)) && (all.slopes[i, j] < (a)))
{
            classif[i, j] <- "="
        }
        classif[i, 4] <- paste(classif[i, 1], classif[i, 2],
classif[i, 3],
            sep = "")
    }
}
table(classif[, 4])
```

```
##
## ==- ==+ =-+ =+- -=+ --+ -+= -+- +== +=- +-= +-- +-+
##   2   2   7   2   2   2   1   2   1   6  11   4  20
```

La classification est beaucoup plus équilibrée. Nous l'analysons en représentant sur un même graphique, tous les clônes de tous les animaux correspondant à un même gène. Les graphiques sont triés par profil d'évolution.

```
jeplotencouleur <- function(nu) {
    the.expr <- jdd[[nu]]
    the.pigs <- pigs[[nu]]
    the.times <- time.steps[[nu]]
    tnt <- time.steps[[nu]]

    nb.clones <- round(length(the.pigs)/30)
    the.factor <- factor(paste("pig", the.pigs, "_clone",
rep(1:nb.clones, rep(30,
        nb.clones)), sep = ""))

    tnt <- rep(1, length(the.times))
    tnt[the.times == 1] <- 2
    tnt[the.times == 4] <- 3
    tnt[the.times == 24] <- 4

    the.colors <- rainbow(nlevels(the.factor))

    plot(NA, type = "n", xlim = c(1, 4), ylim = range(the.expr,
na.rm = TRUE),
        axes = FALSE, xlab = "Heure de prélèvement", ylab =
"Valeur d'expression",
        main = paste("gène numéro", names(jdd)[nu], sep = " "))
    axis(2)
    axis(1, at = c(1, 2, 3, 4), labels = c("+0h", "+1h", "+4h",
"+24h"))

    for (ind in seq_along(unique(the.factor))) {
        the.ind <- unique(the.factor)[ind]
        pig.gene <- the.expr[the.factor == the.ind]
        x.value <- tnt[the.factor == the.ind]
        # order.time <- match(c(0, 1, 4, 24), tnt[the.factor ==
the.ind])
        lines(x.value, pig.gene, col = the.colors[ind], type =
"b", pch = 19)
    }
    plot.new()
    legend("center", lty = 1, col = the.colors, legend =
unique(the.factor))
}
```

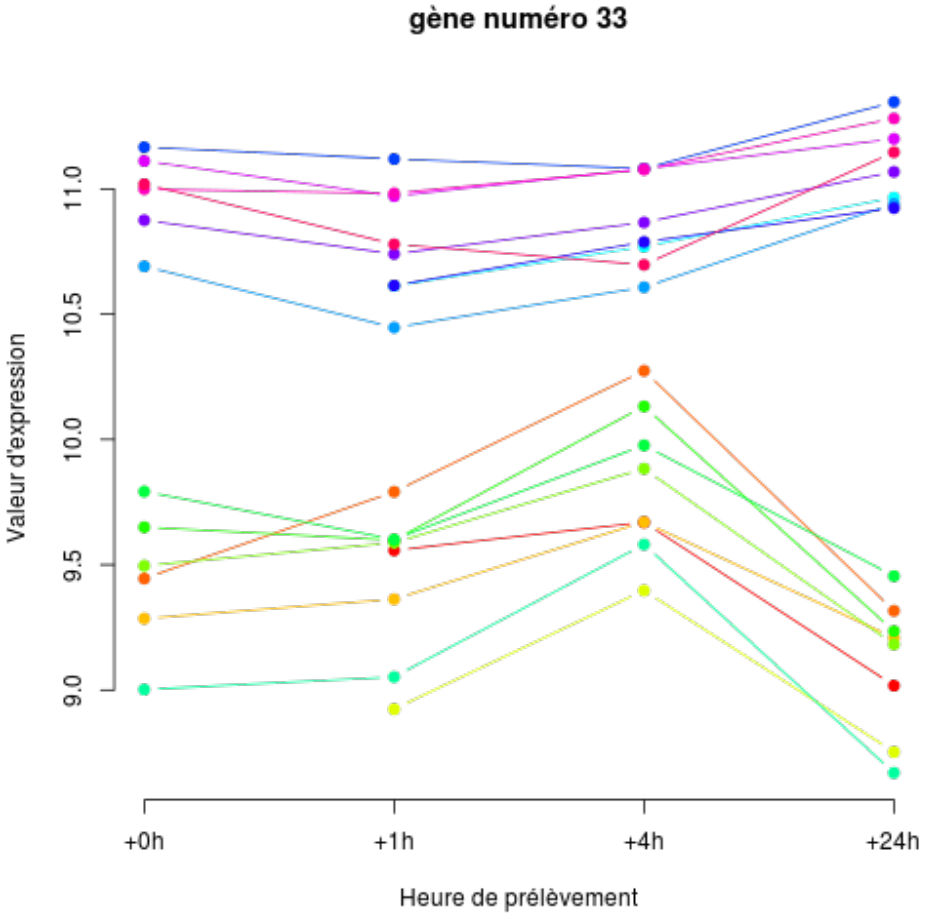
Gènes composant chaque profil (pentes absolues) :

==-

```
liste <- which(classif[, 4] == "==-")
names(jdd[liste])
```

```
## [1] "33" "37"
```

```
for (ind in liste) jeplotencouleur(ind)
```



A.3. Classification automatique des profils d'expression

Classification des gènes selon leur profil pentes

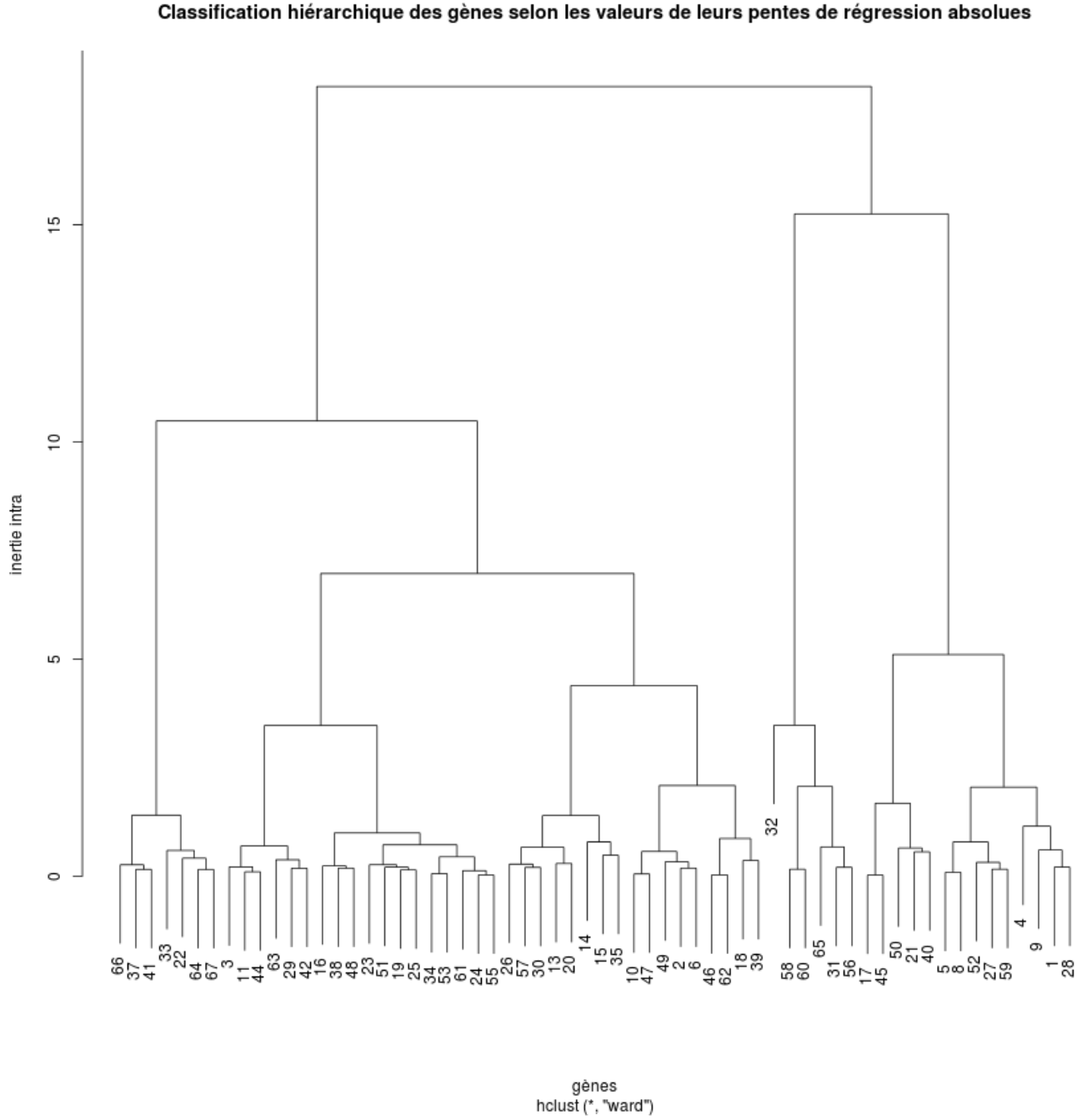
Arthur Gomez, dernière version du fichier : 2 mai 2013

Nous allons donc utiliser les pentes sur les expressions des gènes aux différents temps, pour essayer de définir plusieurs profils, classes de gènes. Nous commencerons donc par les données brutes, puis par la suite avec les pentes centrées et réduites.

Phase exploratoire : classification ascendante hiérarchique

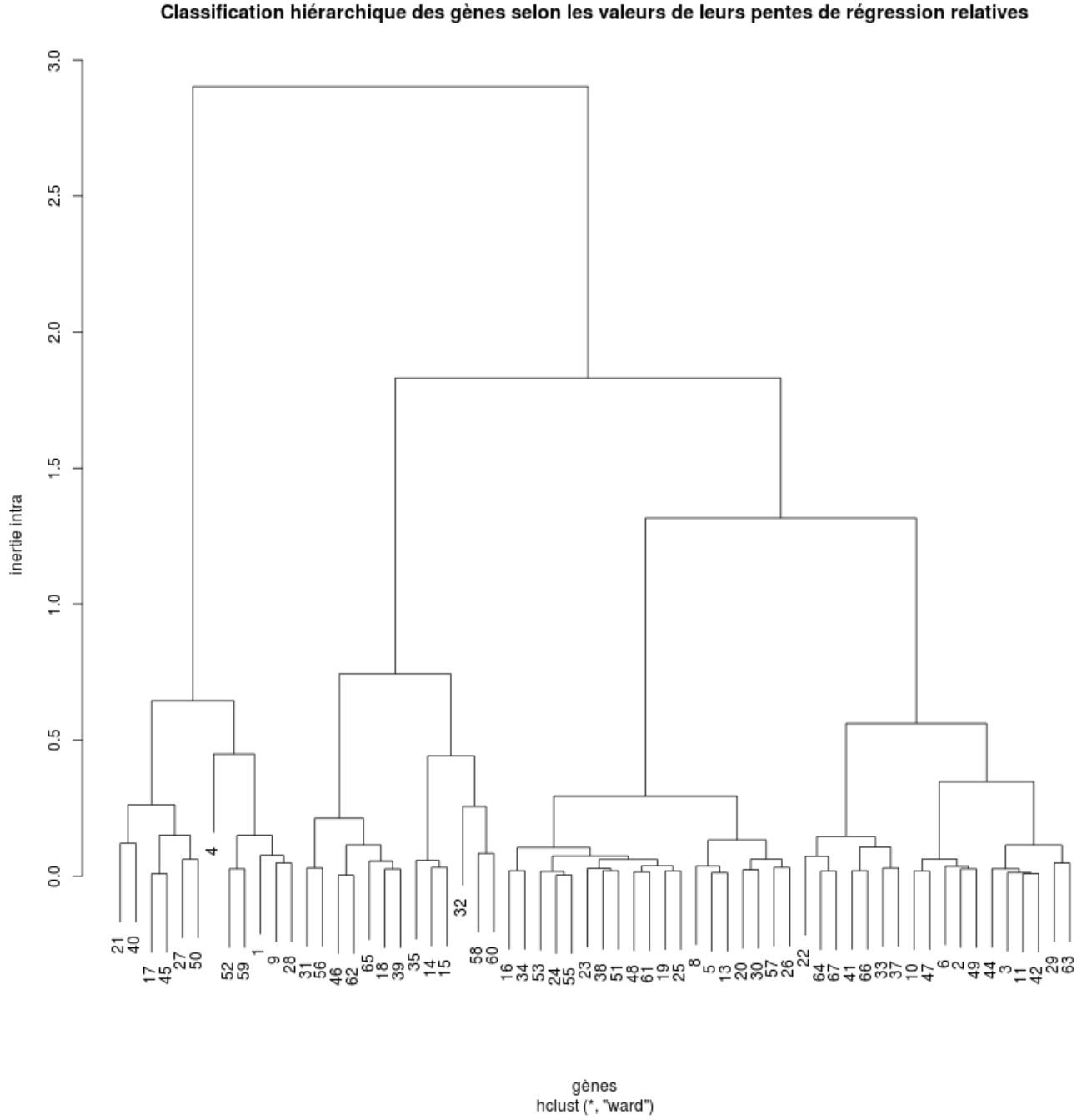
En premier lieu, nous auront une approche par classification ascendante hiérarchique, afin d'avoir une idée du nombre de classes à définir.

```
load("../results/all_slopes.rda")
x <- dist(all.slopes, method = "euclidean")
cha <- hclust(x, method = "ward")
plot(cha, main = "Classification hiérarchique des gènes selon les
valeurs de leurs pentes de régression absolues",
    xlab = "gènes", ylab = "inertie intra")
```



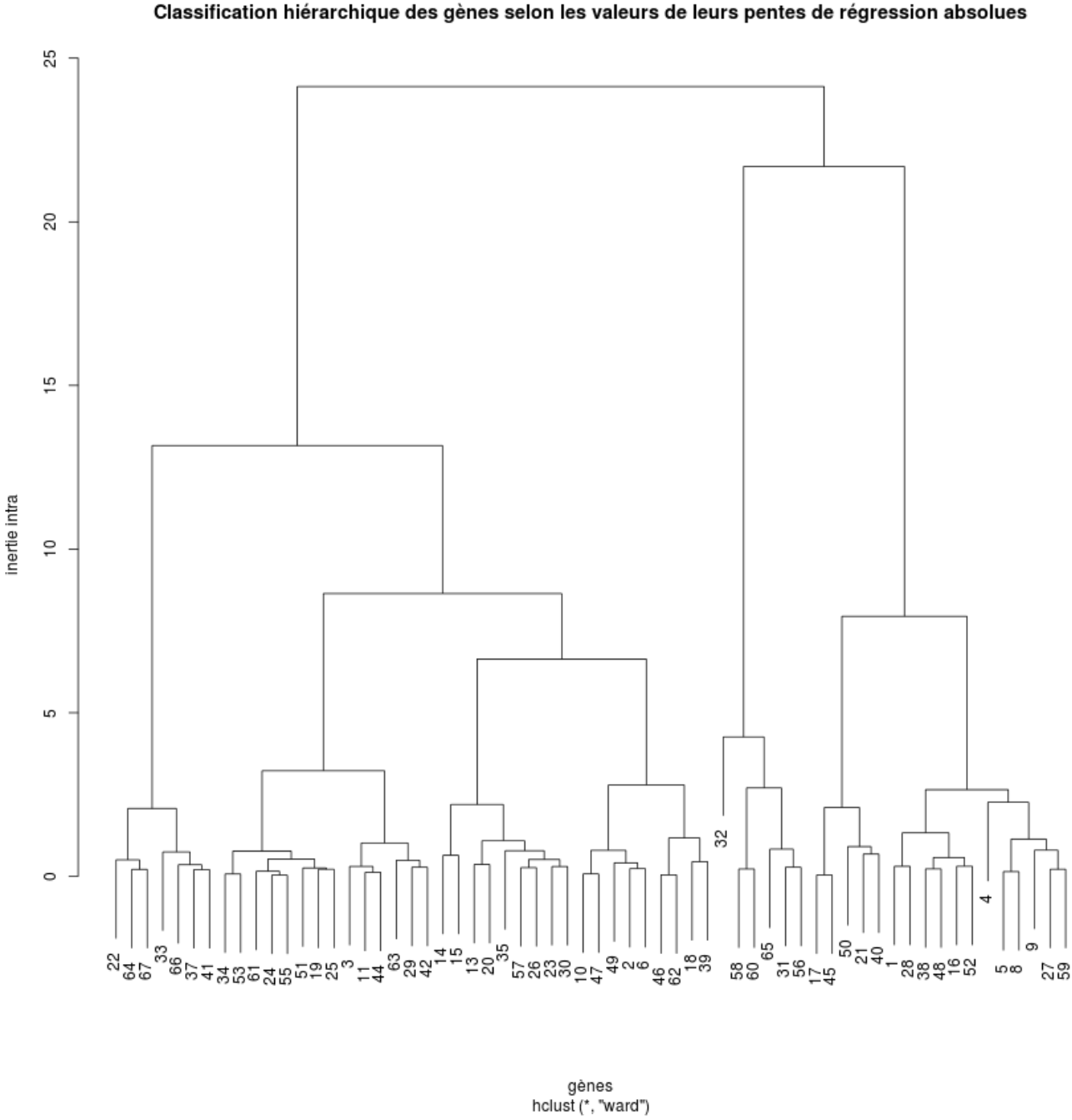
Cette classification se fait à partir de la distance euclidienne et utilise la méthode de Ward. On peut donc penser à partir de ce dendrogramme que 5 classes peut être un choix pertinent pour une étude à partir des k-means sur les pentes absolues.

```
load("../results/all_slopes_re.rda")
y <- dist(all.slopes.re, method = "euclidean")
chr <- hclust(y, method = "ward")
plot(chr, main = "Classification hiérarchique des gènes selon les
valeurs de leurs pentes de régression relatives",
    xlab = "gènes", ylab = "inertie intra")
```



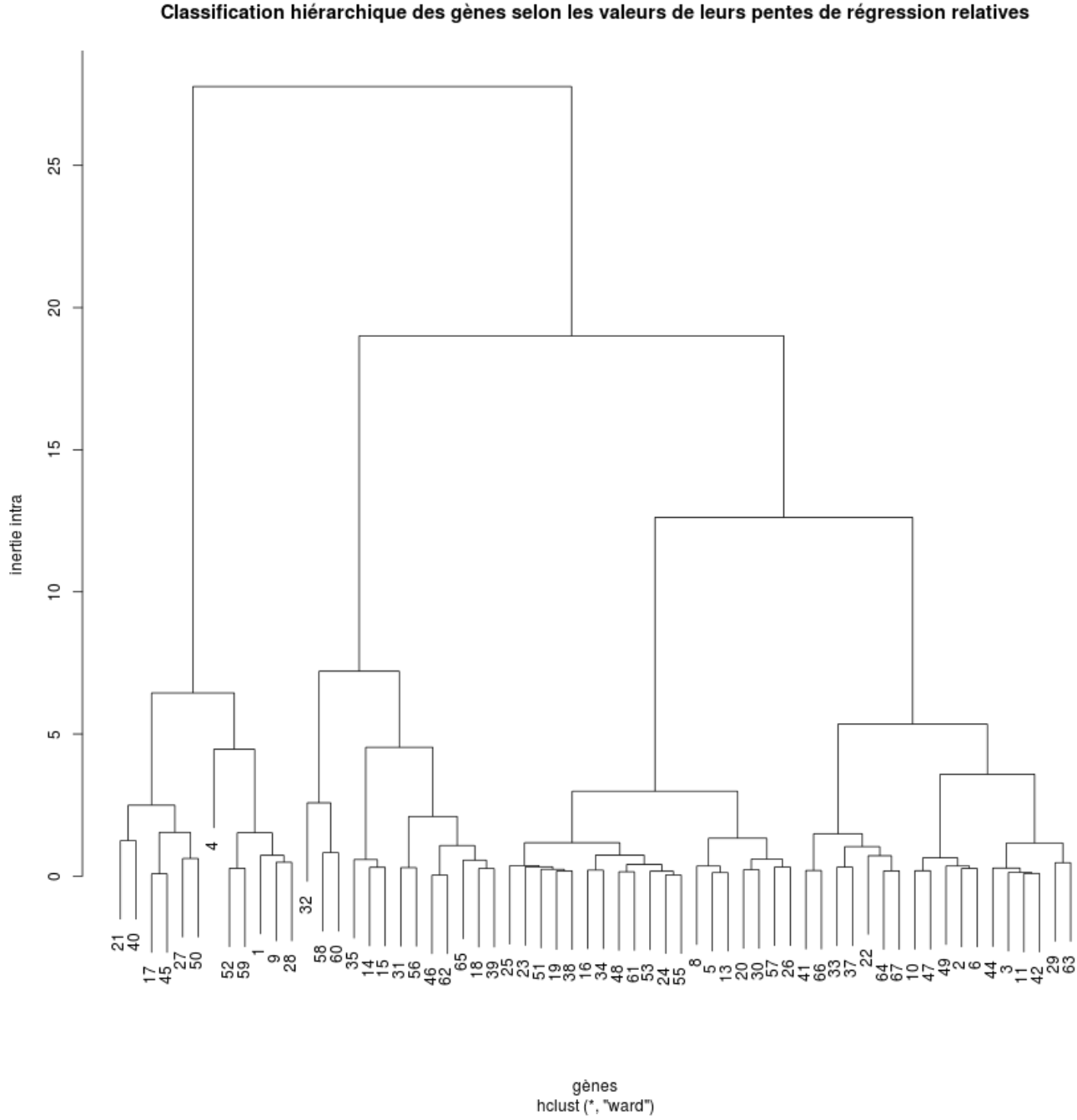
Sur les pentes relatives, on peut penser que le nombre de 4 classes est pertinent pour une approche par k-means Ensuite, nous allons passer à une classification sur les pentes qui seront centrées, et réduites.

```
penta <- scale(all.slopes)
x2 <- dist(penta, method = "euclidean")
cha2 <- hclust(x2, method = "ward")
plot(cha2, main = "Classification hiérarchique des gènes selon les
valeurs de leurs pentes de régression absolues",
    xlab = "gènes", ylab = "inertie intra")
```



Là encore, on choisira de retenir 4 classes.

```
pentr <- scale(all.slopes.re)
y2 <- dist(pentr, method = "euclidean")
chr2 <- hclust(y2, method = "ward")
plot(chr2, main = "Classification hiérarchique des gènes selon les
valeurs de leurs pentes de régression relatives",
    xlab = "gènes", ylab = "inertie intra")
```



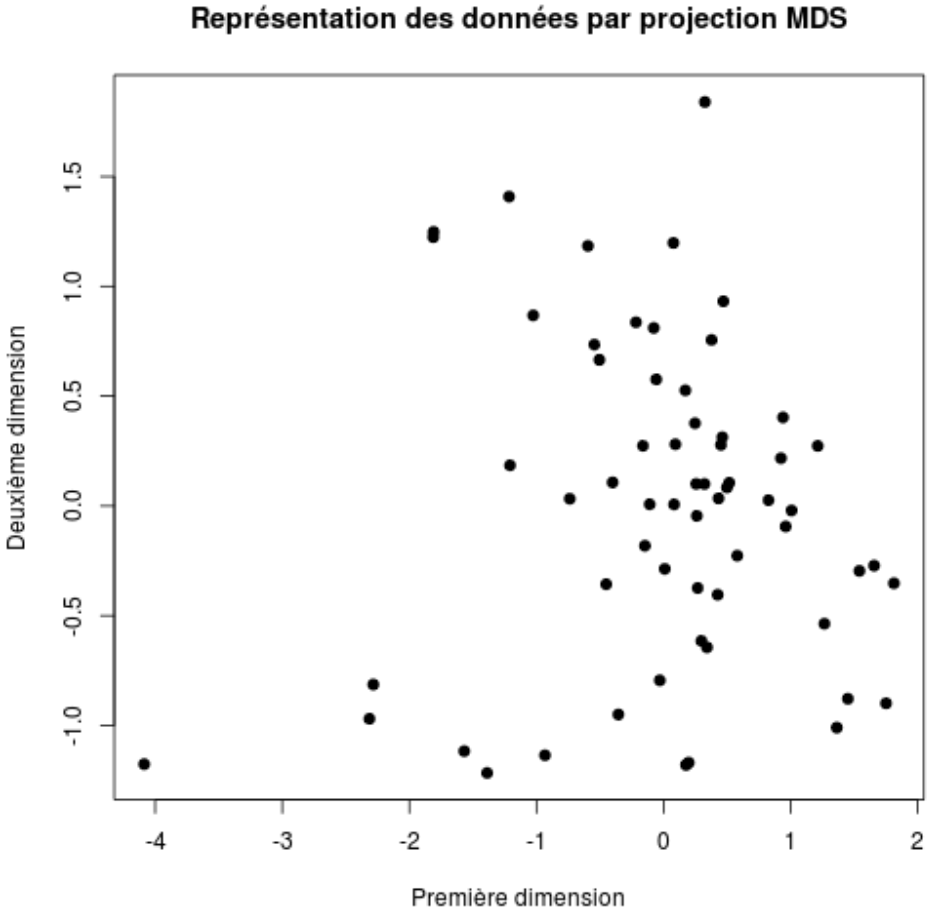
Enfin, sur cette dernière classfication hiérarchique sur les pentes relatives (préalablement centrées et réduites), un découpage en 4 classes semble être adapté pour une étude selon les k-means.

Représentation des données

Les individus (gènes) ont 3 dimensions (trois pentes). Pour les représenter en deux dimensions, on utilise une méthode appelée MDS (Multi-Dimensional Scaling) : cette méthode projette les individus sur un plan de manière à miniser la distorsion des distances deux à deux entre individus (cf Cox & Cox, 2001). Nous nous servirons

ensuite de cette représentation pour comparer les classifications obtenues.

```
coord <- cmdscale(dist(all.slopes))
plot(coord, pch = 19, xlab = "Première dimension", ylab =
"Deuxième dimension",
    main = "Représentation des données par projection MDS")
```



Classification basée sur la méthode des k-means

La méthode des k-means exécute une classification selon des centres moyens (dont le nombre est un paramètre à choisir). Le nombre d'exécutions est défini par le paramètre itération, où l'on recherche à chaque fois d'autres classes/centre moyens. Cette méthode de classification est aléatoire, ce qui peut expliquer des résultats différents lors de certaines exécutions. Comme pour la majorité des classifications hiérarchiques nous avons retenu 4 classes, nous utiliserons 4 comme paramètre de centre-moyens pour les classifications non hiérarchiques de manière à pouvoir comparer les différentes classifications.

k-means pour les pentes absolues

Nous allons donc effectuer une classification pour les pentes absolues, puis sur les données préalablement centrées et réduites, afin de comparer ces classifications.

```
faire_cl <- function(nb.classes, data) {
    cl <- kmeans(data, nb.classes, 200)
    var.tot <- as.numeric(cl[3])
    var.intra <- as.numeric(cl[5])
    best.score <- var.intra/var.tot * 100

    for (i in 1:2000) {
        classif <- kmeans(data, nb.classes, 20)
        var.tot <- as.numeric(classif[3])
        var.intra <- as.numeric(classif[5])
        score <- var.intra/var.tot * 100

        if (best.score > score) {
            cl <- classif
            best.score <- score
        }
    }
    return(cl)
}
```

```
load("../results/all_slopes.rda")
set.seed(11041222)
cna <- faire_cl(4, all.slopes)
```

```
load("../results/all_slopes.rda")
set.seed(11041222)
cna2 <- faire_cl(4, scale(all.slopes))
```

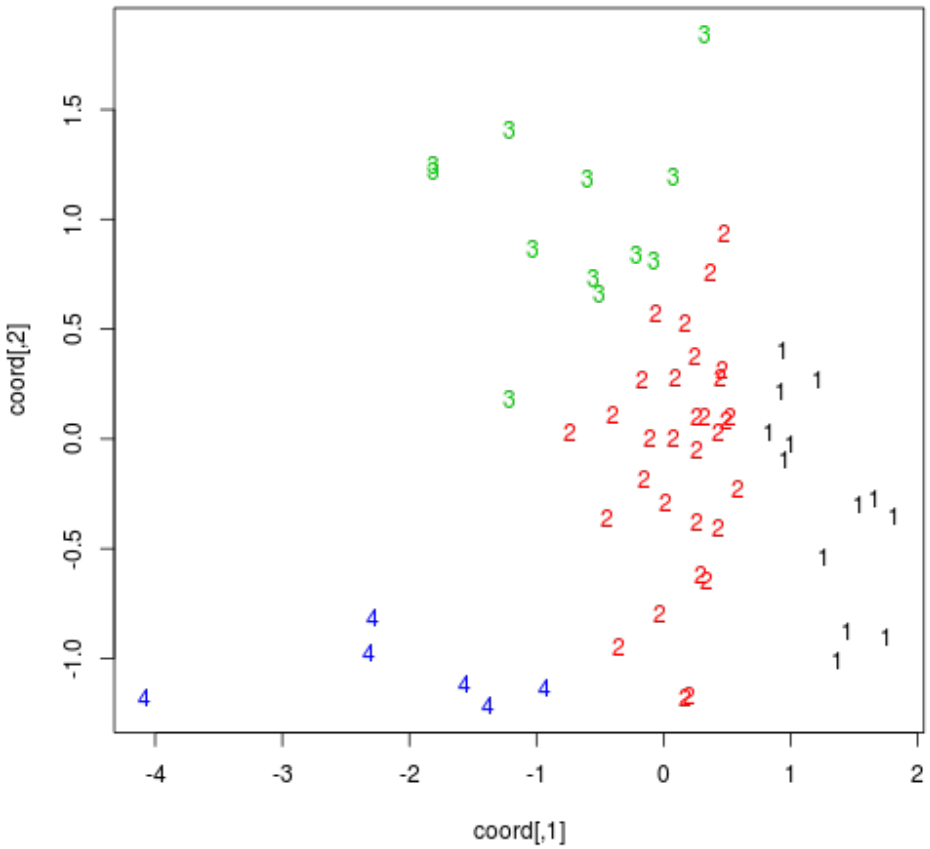
```
library(igraph)
compare.communities(cna$clust, cna2$clust, method = "nmi")
```

```
## [1] 0.6733
```

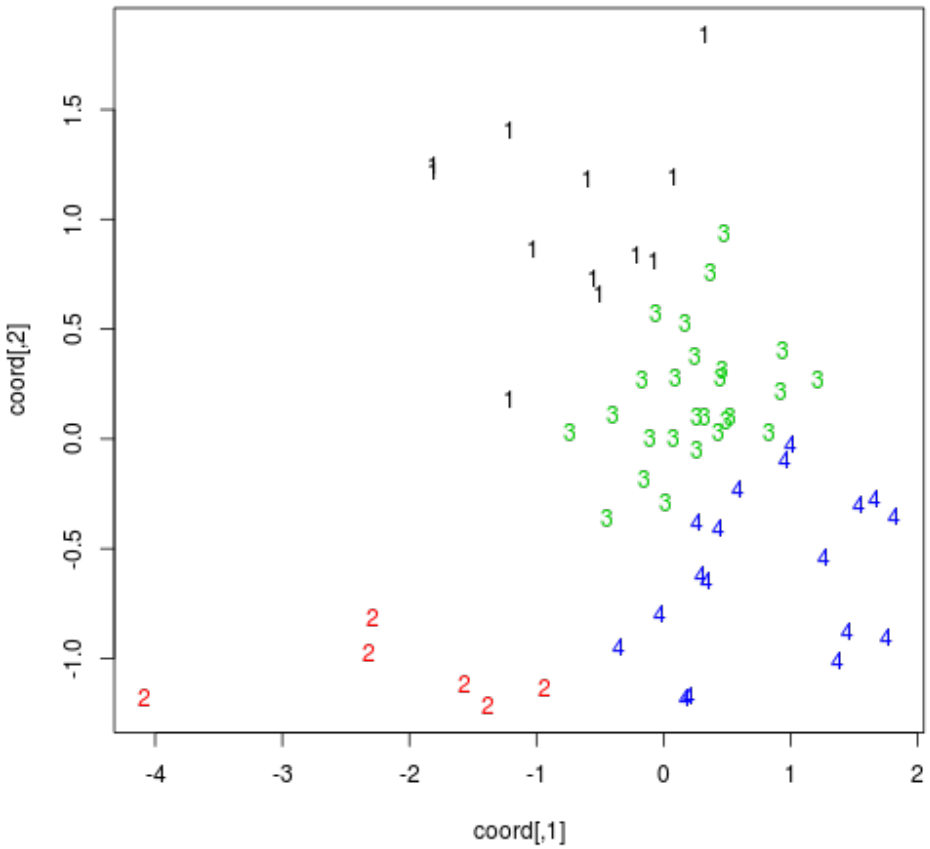
```
table(cna$clust, cna2$clust)
```

```
##
##       1  2  3  4
##   1   0  0  4  9
##   2   0  0 22  9
##   3  12  0  0  0
##   4   0  6  0  0
```

```
plot(coord, type = "n")
text(coord[, 1], coord[, 2], cna$clust, col = cna$clust)
```




```
plot(coord, type = "n")
text(coord[, 1], coord[, 2], cna2$clust, col = cna2$clust)
```



k-means pour les pentes relatives

De la même manière que pour les pentes absolues, nous allons comparer les classifications obtenues par la méthode des k-means selon que les données sont centrées-réduites ou non.

```
load("../results/all_slopes_re.rda")
set.seed(11041222)
cnr <- faire_cl(4, all.slopes.re)
```

```
load("../results/all_slopes_re.rda")
set.seed(11041222)
cnr2 <- faire_cl(4, scale(all.slopes.re))
```

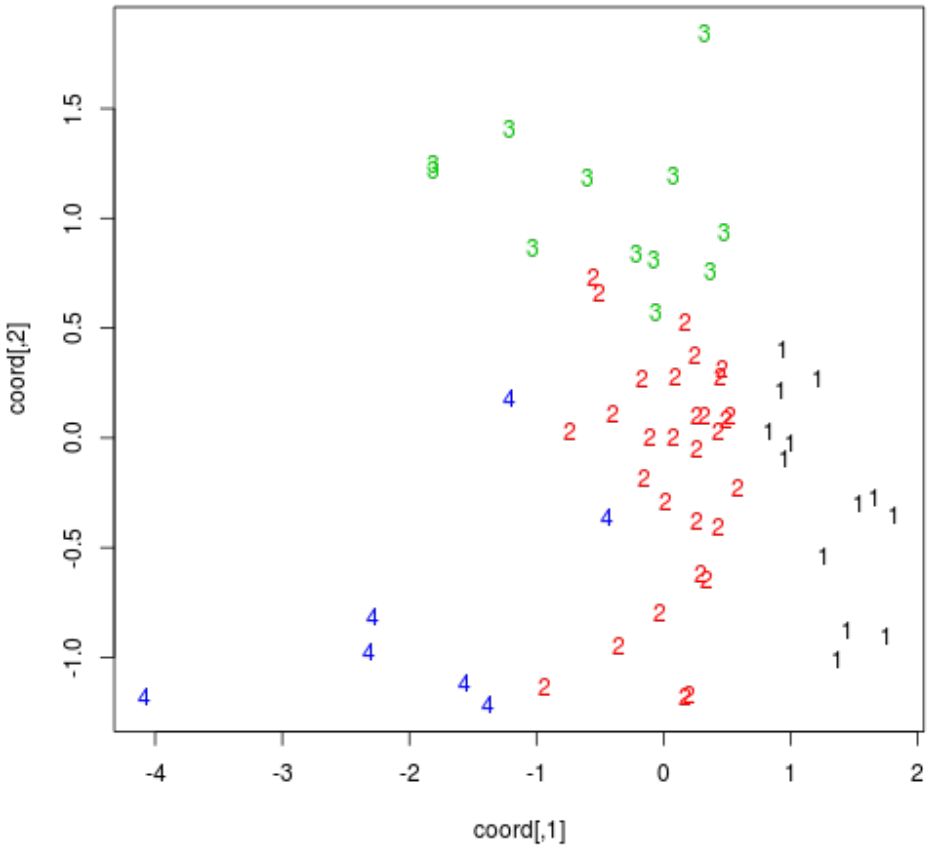
```
table(cnr$clust, cnr2$clust)
```

```
##
##       1  2  3  4
##   1 13  0  0  0
##   2  0 30  0  0
##   3  0  0 12  0
##   4  0  0  0  7
```

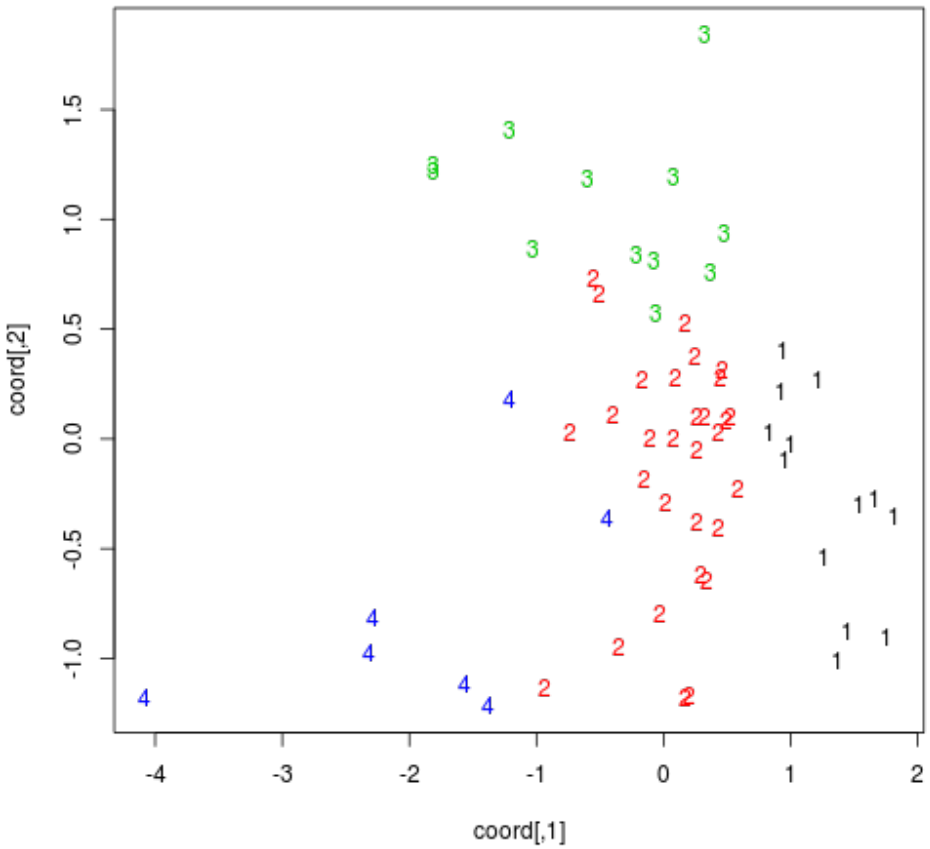
```
library(igraph)
compare.communities(cnr$clust, cnr2$clust, method = "nmi")
```

```
## [1] 1
```

```
plot(coord, type = "n")
text(coord[, 1], coord[, 2], cnr$clust, col = cnr$clust)
```




```
plot(coord, type = "n")
text(coord[, 1], coord[, 2], cnr2$clust, col = cnr2$clust)
```



Essai de classifications avec un nombre de classes plus élevé

La fonction suivante permet de calculer, pour un nombre de classes donné, le pourcentage de variance expliquée obtenue par la meilleure classification sur 2000 classifications obtenues par la méthode des k-means :

```
faire_clv2 <- function(nb.classes, data) {
    cl <- kmeans(data, nb.classes, 200)
    var.tot <- as.numeric(cl[3])
    var.intra <- as.numeric(cl[5])
    best.score <- var.intra/var.tot * 100

    for (i in 1:2000) {
        classif <- kmeans(data, nb.classes, 200)
        var.tot <- as.numeric(classif[3])
        var.intra <- as.numeric(classif[5])
        score <- var.intra/var.tot * 100

        if (best.score > score) {
            cl <- classif
            best.score <- score
        }
    }
    best.score
}
```

On refait la classification en augmentant le nombre de classes : pour trouver un nombre de classes pertinent, on commence par calculer, pour un nombre de classes variant de 1 à 10, le pourcentage de variance on expliquée par le nombre de classes :

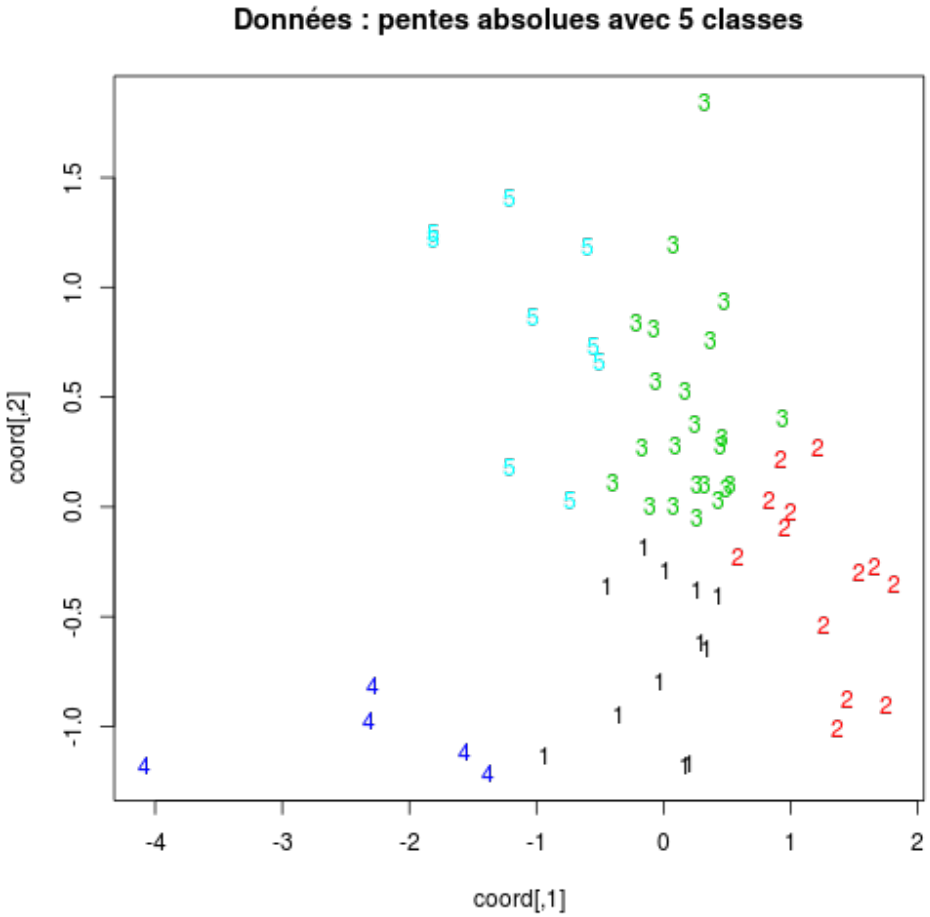
```
set.seed(29121032)
matrice.test <- matrix(nrow = 10, ncol = 4)
for (j in 1:4) {
    if (j == 1) {
        data <- all.slopes
        for (i in 1:10) {
            matrice.test[i, j] <- faire_clv2(i, data)
        }
    }
    if (j == 2) {
        data <- scale(all.slopes)
        for (i in 1:10) {
            matrice.test[i, j] <- faire_clv2(i, data)
        }
    }
    if (j == 3) {
        data <- all.slopes.re
        for (i in 1:10) {
            matrice.test[i, j] <- faire_clv2(i, data)
        }
    }
    if (j == 4) {
        data <- scale(all.slopes.re)
        for (i in 1:10) {
            matrice.test[i, j] <- faire_clv2(i, data)
        }
    }
}
colnames(matrice.test) <- c("absolues", "absolues+scale",
"relatives", "relatives+scale")
matrice.test
```

```
##       absolues absolues+scale relatives relatives+scale
##  [1,]  100.000        100.000   100.000         100.000
##  [2,]   58.545         63.677    58.753          60.780
##  [3,]   37.890         37.489    38.567          39.080
##  [4,]   27.918         28.176    29.436          29.988
##  [5,]   21.612         21.671    23.608          24.066
##  [6,]   17.228         17.357    18.874          18.873
##  [7,]   13.129         13.816    14.928          14.877
##  [8,]   10.557         10.618    11.243          11.401
##  [9,]    9.051          9.262     9.303           9.555
## [10,]    7.771          8.060     7.949           8.273
```

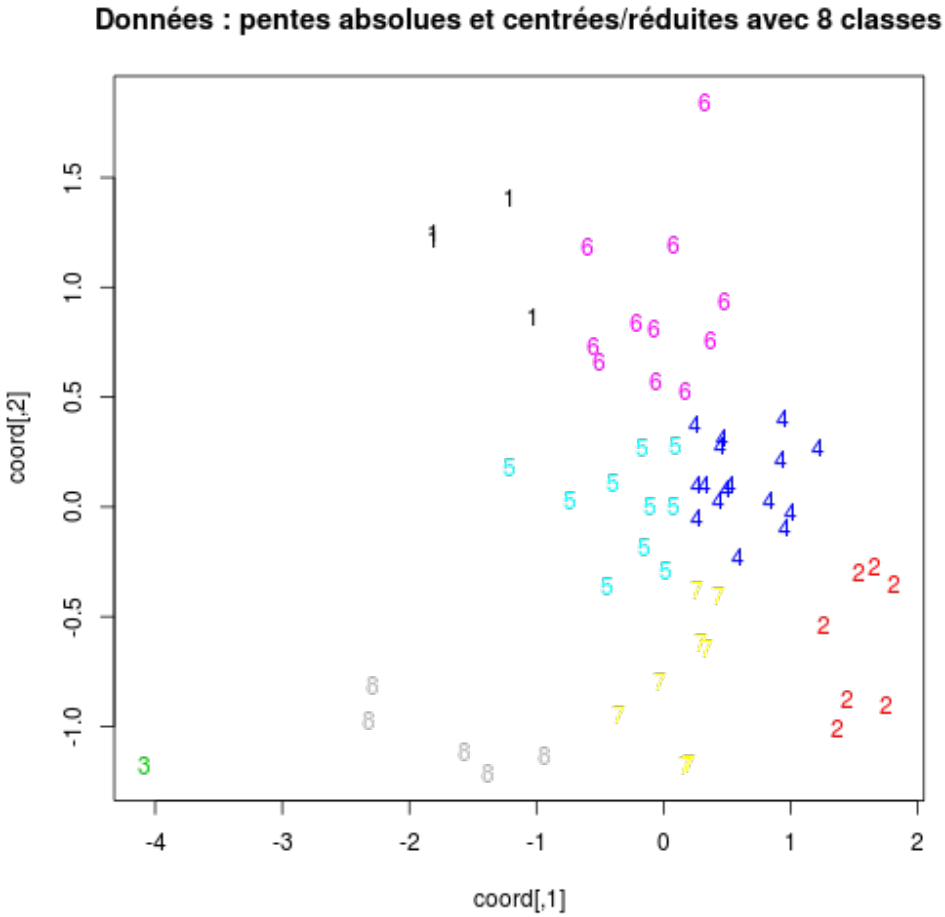
Pour les 4 types de données, nous avons ensuite effectué une classification avec un nombre de classes plus élevé (que 4) et qui paraissait pertinent, du point de vue de la

variance expliquée.

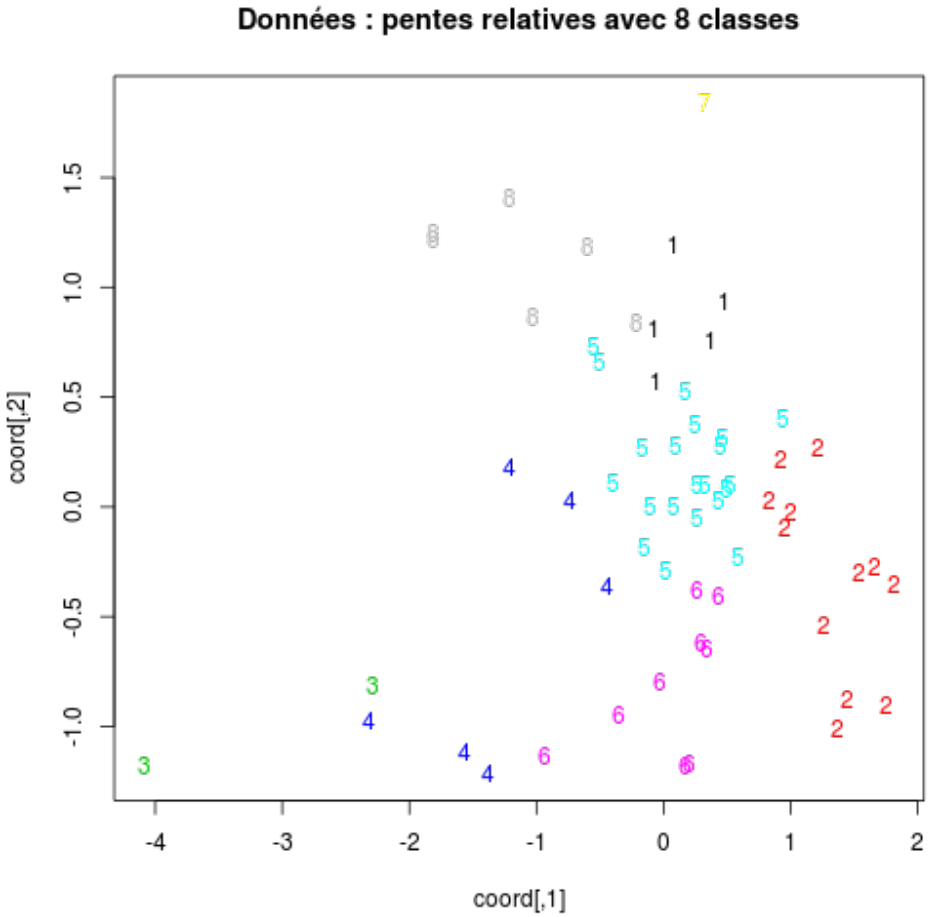
```
set.seed(29041014)
cl1 <- faire_cl(5, all.slopes)
cl2 <- faire_cl(8, scale(all.slopes))
cl3 <- faire_cl(8, all.slopes.re)
cl4 <- faire_cl(6, scale(all.slopes.re))

plot(coord, type = "n", main = "Données : pentes absolues avec 5
classes")
text(coord[, 1], coord[, 2], cl1$clust, col = cl1$clust)
```



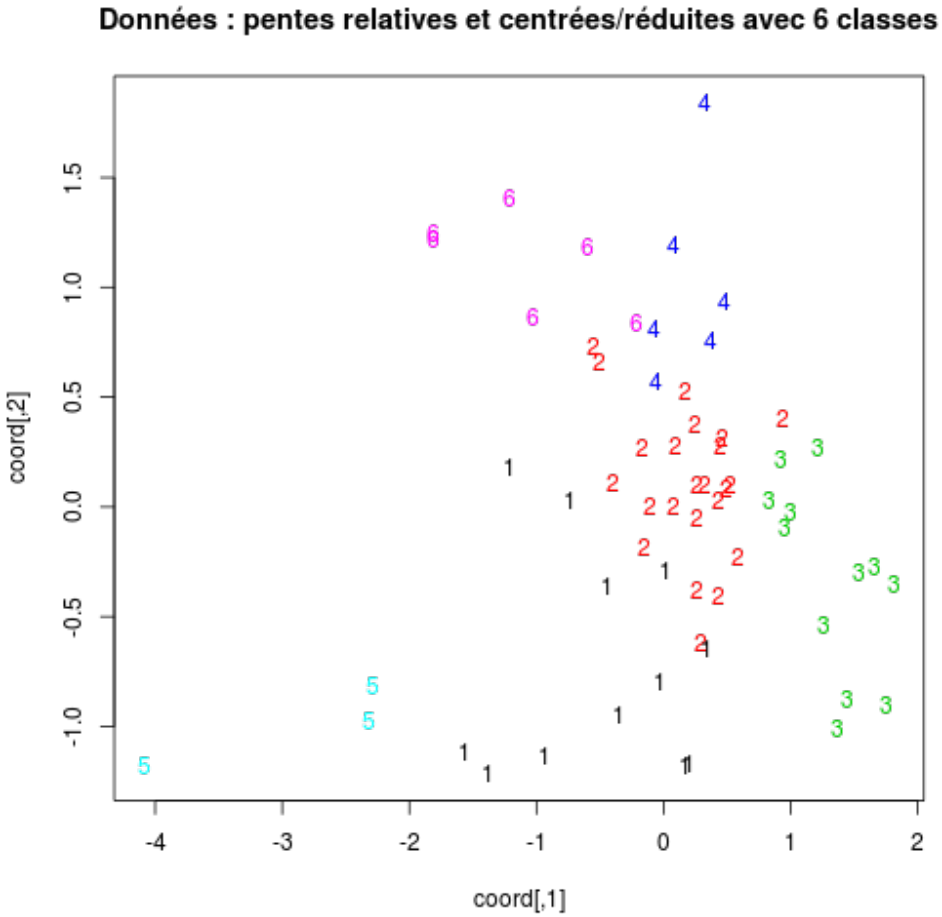

```
plot(coord, type = "n", main = "Données : pentes absolues et
centrées/réduites avec 8 classes")
text(coord[, 1], coord[, 2], cl2$clust, col = cl2$clust)
```




```
plot(coord, type = "n", main = "Données : pentes relatives avec 8
classes")
text(coord[, 1], coord[, 2], cl3$clust, col = cl3$clust)
```




```
plot(coord, type = "n", main = "Données : pentes relatives et
centrées/réduites avec 6 classes")
text(coord[, 1], coord[, 2], cl4$clust, col = cl4$clust)
```



On peut remarquer que travailler sur les données absolues donnent des résultats semblant plus homogènes (confirmé par le tableau de pourcentages des variances intra toujours plus faibles, à nombre de classes égales). De plus,sur les pentes absolues, parfois les données centrées et réduites donnent un peu moins d'inertie intra, et cette approche peut donc être intéressante selon le nombre de classes demandées.

Composition des classes pour les classifications des pentes absolues

Nous allons appliquer cette fonction graphique aux gènes classés selon les classifications des pentes absolues ci-dessus (plus pertinentes que les pentes relatives au nievau de l'inertie intra). Ainsi, pour chaque classe, chaque gène unique aura une représentation graphique au cours du temps comme dans l'étude sur les catégories par tri.

```
jeplotencouleur <- function(nu) {
    the.expr <- jdd[[nu]]
    the.pigs <- pigs[[nu]]
    the.times <- time.steps[[nu]]
    tnt <- time.steps[[nu]]

    nb.clones <- round(length(the.pigs)/30)
    the.factor <- factor(paste("pig", the.pigs, "_clone",
rep(1:nb.clones, rep(30,
        nb.clones)), sep = ""))
    tnt <- rep(1, length(the.times))
    tnt[the.times == 1] <- 2
    tnt[the.times == 4] <- 3
    tnt[the.times == 24] <- 4

    the.colors <- rainbow(nlevels(the.factor))

    plot(NA, type = "n", xlim = c(1, 4), ylim = range(the.expr,
na.rm = TRUE),
        axes = FALSE, xlab = "Heure de prélèvement", ylab =
"Valeur d'expression",
        main = paste("gène numéro", names(jdd)[nu], sep = " "))
    axis(2)
    axis(1, at = c(1, 2, 3, 4), labels = c("+0h", "+1h", "+4h",
"+24h"))

    for (ind in seq_along(unique(the.factor))) {
        the.ind <- unique(the.factor)[ind]
        pig.gene <- the.expr[the.factor == the.ind]
        x.value <- tnt[the.factor == the.ind]
        # order.time <- match(c(0, 1, 4, 24), tnt[the.factor ==
the.ind])
        lines(x.value, pig.gene, col = the.colors[ind], type =
"b", pch = 19)
    }
    plot.new()
    legend("center", lty = 1, col = the.colors, legend =
unique(the.factor))
}
```

```
# This file makes a list of vectors Each element of the list
corresponds
# to one unique gene and is a vector having a number of elements
equal to
# the number of clones multiplied by the number of experiments

# Two other lists are constructed: pigs gives the number of the
pig and
# times the time step
load("../data/acth-data.rda")
duplicates <- read.table("../data/duplicates.csv", sep = ";",
header = TRUE)
duplicates$N.gene <- as.factor(duplicates$N.gene)

jdd <- list()
pigs <- list()
time.steps <- list()
for (i in seq_along(unique(duplicates$N.gene))) {
    the.clones <- which(duplicates$N.gene ==
unique(duplicates$N.gene)[i])
    jdd[[as.character(unique(duplicates$N.gene)[i])]] <-
as.vector(sign.genes[,
        the.clones])
    pigs[[as.character(unique(duplicates$N.gene)[i])]] <-
rep(indiv$porcexp,
        length(the.clones))
    time.steps[[as.character(unique(duplicates$N.gene)[i])]] <-
rep(indiv$temps,
        length(the.clones))
}
```

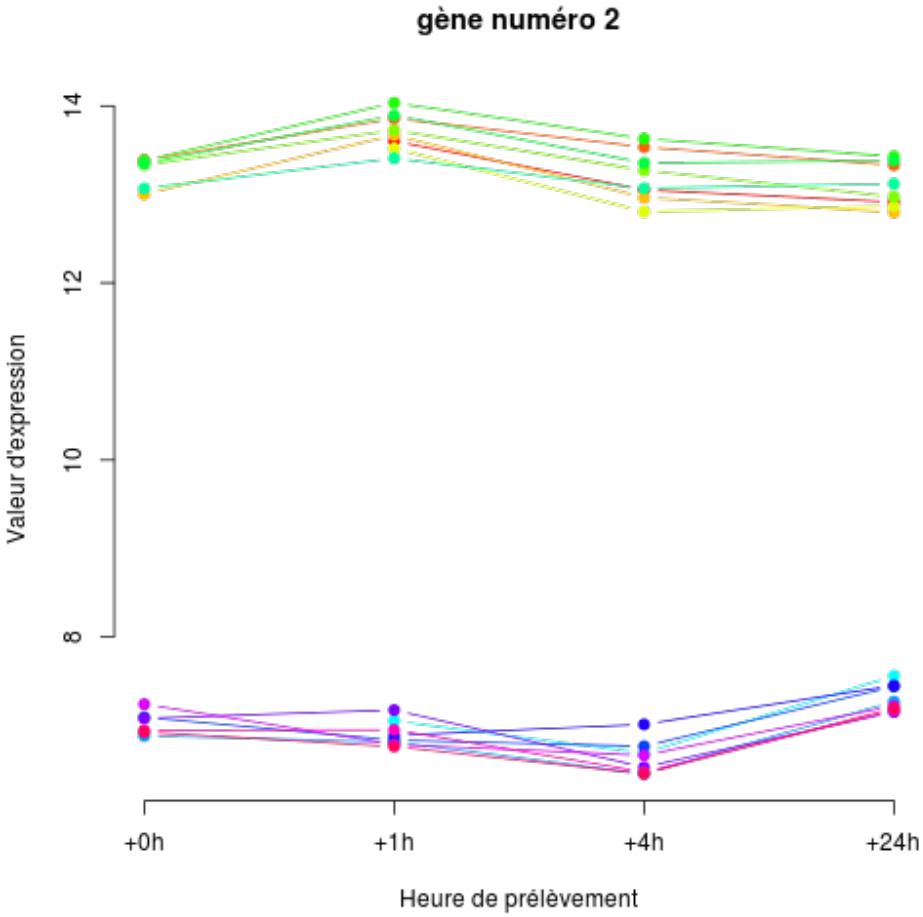
Composition des classes (classification en 5 classes)

Classe 1 :

```
liste <- which(cl1$clust == "1")
names(jdd[liste])
```

```
##  [1] "2"  "6"  "10" "57" "18" "26" "35" "39" "46" "47" "62" "65"
```

```
for (ind in liste) jeplotencouleur(ind)
```


gène numéro 2
Valeur d'expression
14
12
10
8
+0h
+1h
+4h
+24h
Heure de prélèvement

A.4. Etudes PLS

Analyse PLS entre les expressions des gènes et les phénotypes mesurés

Arthur Gomez, dernière version du fichier : 23 mai 2013

Cette analyse a pour but de visualiser un lien entre phénotypes et expression de gènes. Ainsi, l'individu n'est plus le gène seul, mais un prélèvement sur un animal, à un temps après injection. De cette manière, on a normalement pour chaque expérience, des données phénotypiques, mais aussi l'analyse du transcryptome.

Nous utiliserons dons la table `phenotypes` (13 phenotypes x 30 individus) et la table `sign.genes` (110 gènes x 30 individus) contenant les expressions des 110 gènes avec une cinétique significative au cours du temps sur les 30 expériences.

Analyse descriptive simple des phénotypes

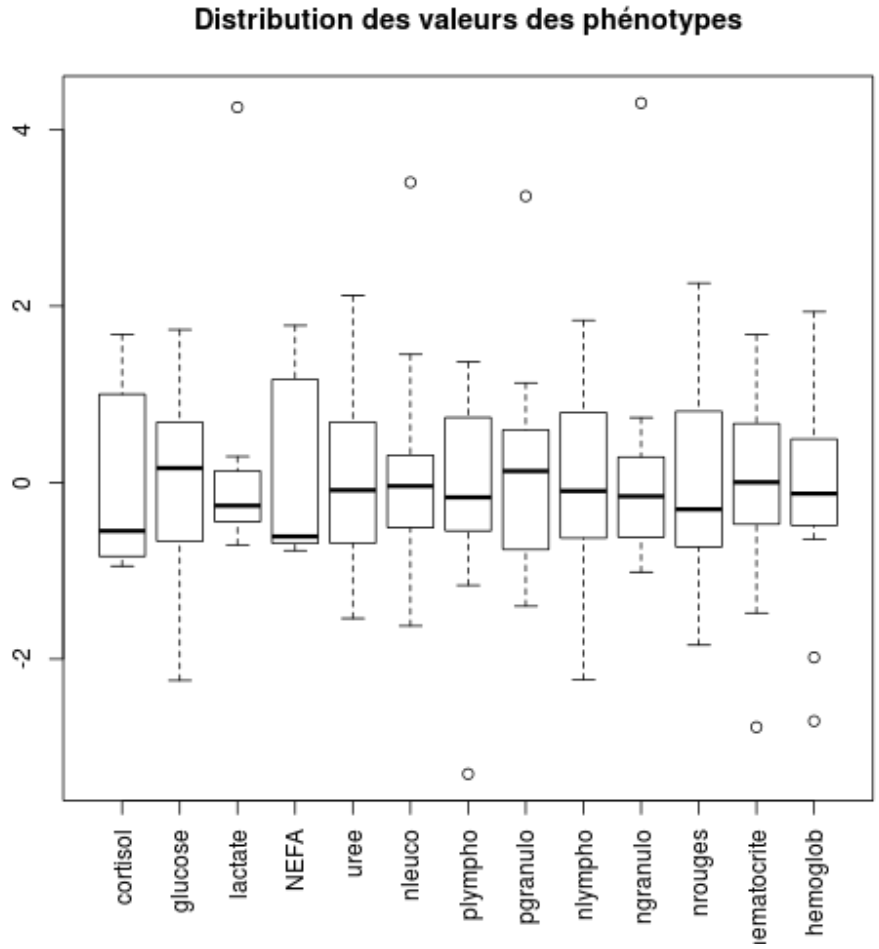
Etant donné la nature des données, (taux, decomptes, proportions) il est interessant de centrer et réduire les données pour la représentation graphique. Sur cette distribution, on peut remarquer une certaine assymétrie pour les variables Cortisol, NEFA, et nrouge indiquant des valeurs atypiques fortes. On peut aussi remarquer une dispersion relativement plus forte au niveau des variables glucose, uree, nlympho et nrouges.

Analyse exploratoire

```
load("../data/acth-data.rda")
summary(phenotypes)
```

```
##     cortisol        glucose        lactate          NEFA
##  Min.   :  4.8   Min.   : 809   Min.   : 556   Min.   :  27.9
##  1st Qu.: 10.4   1st Qu.: 922   1st Qu.: 767   1st Qu.:  91.2
##  Median : 23.5   Median : 981   Median : 911   Median : 149.3
##  Mean   : 49.3   Mean   : 969   Mean   :1122   Mean   : 611.8
##  3rd Qu.: 95.7   3rd Qu.:1013   3rd Qu.:1225   3rd Qu.:1488.7
##  Max.   :128.1   Max.   :1092   Max.   :4529   Max.   :1958.9
##  NA's   :8       NA's   :8      NA's   :8      NA's   :8
##       uree          nleuco         plympho         pgranulo
nlympho
##  Min.   :176   Min.   :17.1   Min.   :30.4   Min.   :31.9
Min.   : 8.0
##  1st Qu.:206   1st Qu.:20.8   1st Qu.:51.4   1st Qu.:36.9   1st
Qu.:11.2
##  Median :228   Median :22.3   Median :54.3   Median :43.7
Median :12.2
##  Mean   :231   Mean   :22.4   Mean   :55.6   Mean   :42.7
Mean   :12.4
##  3rd Qu.:252   3rd Qu.:23.4   3rd Qu.:61.2   3rd Qu.:47.3   3rd
Qu.:13.9
##  Max.   :306   Max.   :33.6   Max.   :66.0   Max.   :67.7
Max.   :16.0
##  NA's   :8     NA's   :3      NA's   :3      NA's   :3
NA's   :3
##     ngranulo        nrouges      hematocrite      hemoglob
##  Min.   : 6.60   Min.   :5.50   Min.   :26.0   Min.   : 7.40
##  1st Qu.: 7.80   1st Qu.:6.15   1st Qu.:32.0   1st Qu.: 9.55
##  Median : 9.20   Median :6.40   Median :33.3   Median : 9.90
##  Mean   : 9.67   Mean   :6.58   Mean   :33.3   Mean   :10.02
##  3rd Qu.:10.55   3rd Qu.:7.05   3rd Qu.:35.0   3rd Qu.:10.50
##  Max.   :22.70   Max.   :7.90   Max.   :37.7   Max.   :11.90
##  NA's   :3       NA's   :3      NA's   :3      NA's   :3
```

```
boxplot(scale(phenotypes), main = "Distribution des valeurs des
phénotypes",
    las = 3)
```



Analyse et correction des valeurs manquantes

Le nombre de valeurs manquantes par colonne et par ligne est donné par les lignes de commandes suivantes :

```
apply(phenotypes, 2, function(x) sum(is.na(x)))
```

```
##    cortisol     glucose     lactate        NEFA
uree      nleuco
##           8           8           8           8
8           3
##     plympho    pgranulo     nlympho    ngranulo     nrouges
hematocrite
##           3           3           3           3
3           3
##    hemoglob
##           3
```

```
missing.in.rows <- apply(phenotypes, 1, function(x) sum(is.na(x)))
missing.in.rows
```

```
##  5  6  7 12 13 14 15 17 18 19 20 22 23 24 25 26 27 28 29 30 31
32 35 36 37
##  0  0 13  0  0  0 13  0  0  0  5  0  0  5  0  0  0  5  0  8  0
5  0  0  0
## 38 39 40 41 42
##  5  0  0  0  5
```

Puis les deux lignes pour lesquelles toutes les mesures phénotypiques sont manquantes sont supprimées du fichier de données des phénotypes ainsi que du fichier des expressions de gènes :

```
all.missing <- which(missing.in.rows == ncol(phenotypes))
all.missing
```

```
##  7 15
##  3  7
```

```
no.allmissing.pheno <- phenotypes[-all.missing, ]
kept.genes <- sign.genes[-all.missing, ]
```

Ensuite, le package `imputation` est utilisé pour imputer les valeurs manquantes par une méthode des plus proches voisins : les 3 plus proches voisins sont utilisés pour imputer les valeurs manquantes (les observations pour lesquelles la distances, calculées sur les valeurs des variables non manquantes, sont les plus proches).

```
imputed.pheno <- kNNImpute(no.allmissing.pheno, 3, verbose =
TRUE)$x
```

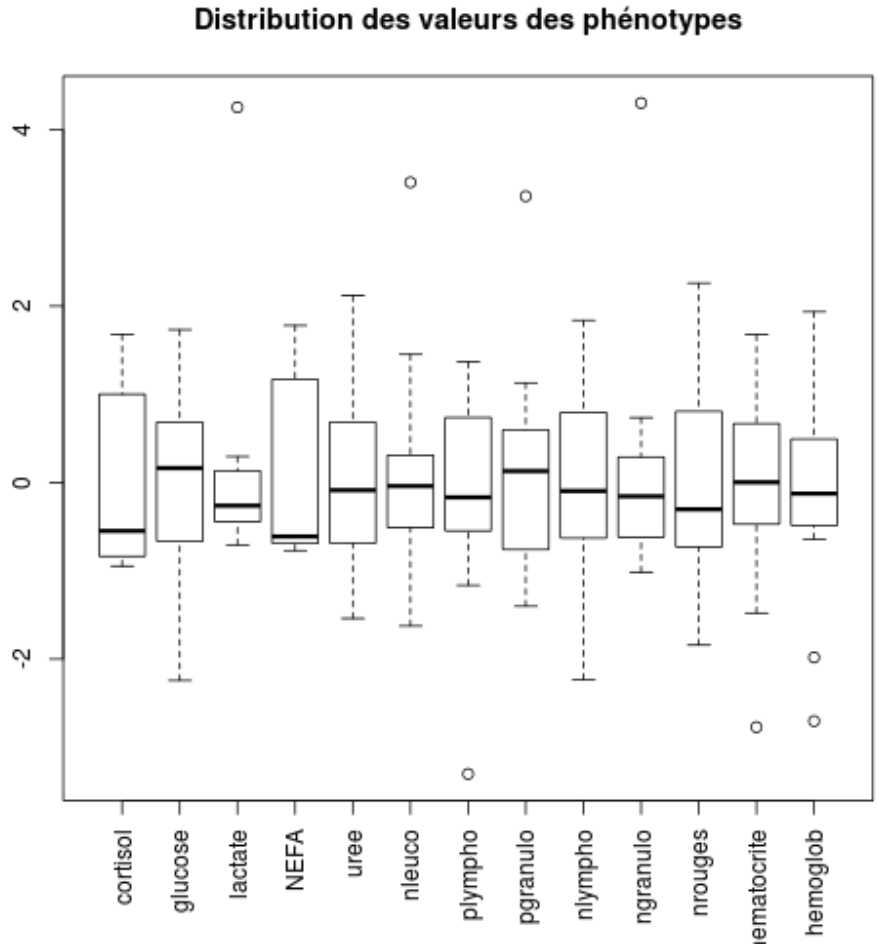
```
## [1] "Computing distance matrix..."
## [1] "Distance matrix complete"
## [1] "Imputing row 9"
## [1] "Imputing row 12"
## [1] "Imputing row 16"
## [1] "Imputing row 18"
## [1] "Imputing row 20"
## [1] "Imputing row 24"
## [1] "Imputing row 28"
```

Enfin, la distribution des phénotypes, après correction des valeurs manquantes est donnée ci-dessous :

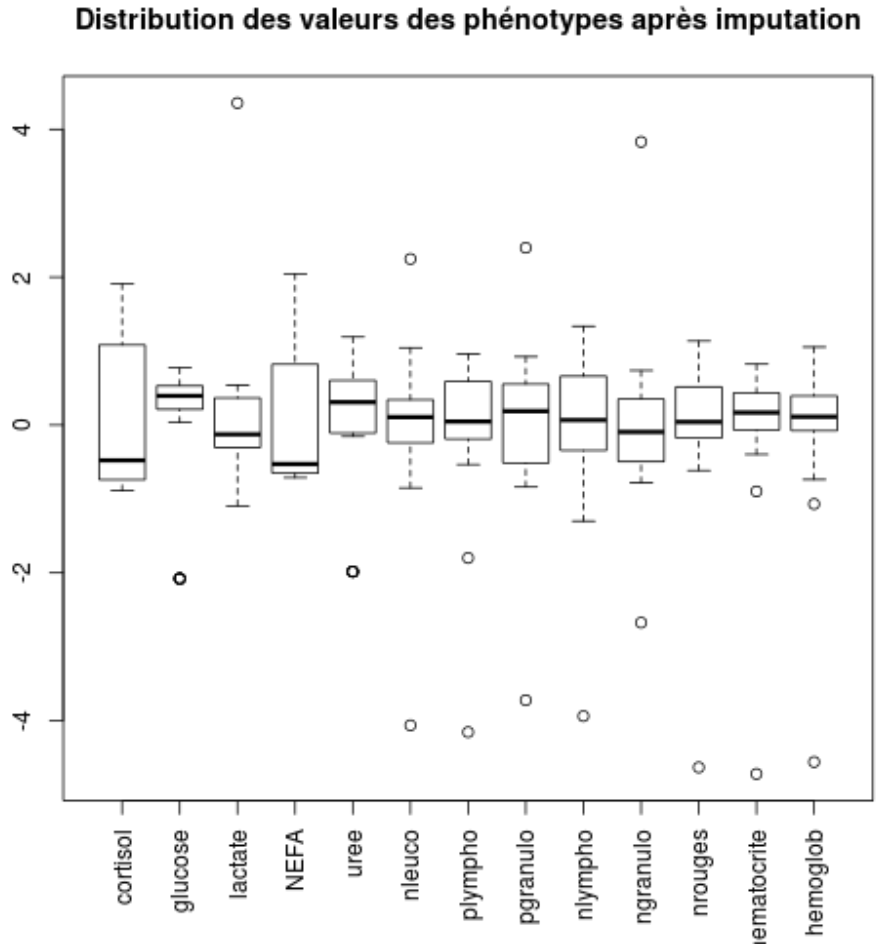
```
summary(imputed.pheno)
```

```
##     cortisol         glucose        lactate          NEFA
##  Min.   :  0.00   Min.   :   0   Min.   :   0   Min.   :   0.0
##  1st Qu.:  6.92   1st Qu.: 888   1st Qu.: 675   1st Qu.:  48.8
##  Median : 18.85   Median : 947   Median : 804   Median : 131.2
##  Mean   : 40.70   Mean   : 796   Mean   : 909   Mean   : 506.2
##  3rd Qu.: 88.25   3rd Qu.:1000   3rd Qu.:1210   3rd Qu.: 900.8
##  Max.   :128.10   Max.   :1092   Max.   :4529   Max.   :1958.9
##       uree          nleuco         plympho         pgranulo
nlympho
##  Min.   :  0   Min.   : 0.0   Min.   : 0.0   Min.   : 0.0
Min.   : 0.0
##  1st Qu.:181   1st Qu.:20.5   1st Qu.:51.2   1st Qu.:35.7   1st
Qu.:11.0
##  Median :221   Median :22.2   Median :54.2   Median :43.2
Median :12.2
##  Mean   :191   Mean   :21.6   Mean   :53.6   Mean   :41.2
Mean   :11.9
##  3rd Qu.:246   3rd Qu.:23.4   3rd Qu.:60.5   3rd Qu.:47.2   3rd
Qu.:13.9
##  Max.   :306   Max.   :33.6   Max.   :66.0   Max.   :67.7
Max.   :16.0
##     ngranulo        nrouges      hematocrite      hemoglob
##  Min.   : 0.00   Min.   :0.00   Min.   : 0.0   Min.   : 0.00
##  1st Qu.: 7.60   1st Qu.:6.10   1st Qu.:31.8   1st Qu.: 9.50
##  Median : 9.00   Median :6.40   Median :33.2   Median : 9.90
##  Mean   : 9.33   Mean   :6.34   Mean   :32.1   Mean   : 9.66
##  3rd Qu.:10.53   3rd Qu.:7.03   3rd Qu.:34.8   3rd Qu.:10.50
##  Max.   :22.70   Max.   :7.90   Max.   :37.7   Max.   :11.90
```

```
boxplot(scale(phenotypes), main = "Distribution des valeurs des
phénotypes",
    las = 3)
```




```
boxplot(scale(imputed.pheno), main = "Distribution des valeurs des
phénotypes après imputation",
    las = 3)
```



On peut voir certains changements dans la distribution, en particulier plus de valeurs atypiques. Certaines variables sont moins dispersées : Glucose ; Urée ; nleuco . On voit aussi apparaitre une certaine assymétrie due à des valeurs fortes sur l'urée, ce qui veut dire que la valeur imputée est peut être un peu trop forte comparée à la distribution, ou que cela est lié à la valeur faible qui est considérée comme atypique après imputation.

Les autres variables gardent des distributions plutôt similaires.

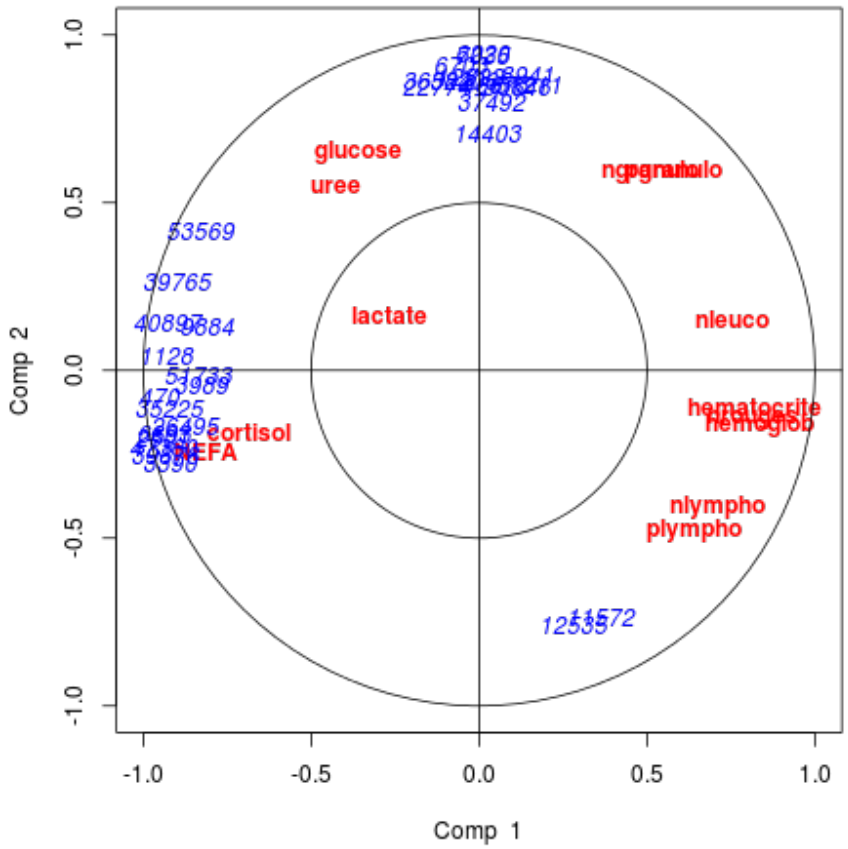
L'imputation est exécutée aussi au niveau de la table `sign.genes`. Les individus sans mesure de phénotype qui ont été enlevés de la table phenotypes sont aussi enlevés de la table `sign.genes`. Enfin, les valeurs manquantes des expressions de gènes sont aussi imputées de la même manière que pour les phénotypes.

```
where.miss <- which(apply(kept.genes, 1, function(x)
sum(is.na(x))) != 0)
nearest.nei <- names(sort(as.matrix(dist(kept.genes))[11, ])[2:4])
col.miss <- which(apply(kept.genes, 2, function(x) sum(is.na(x)))
!= 0)
imputed.genes <- kept.genes
imputed.genes[where.miss, col.miss] <-
mean(kept.genes[nearest.nei, col.miss])
```

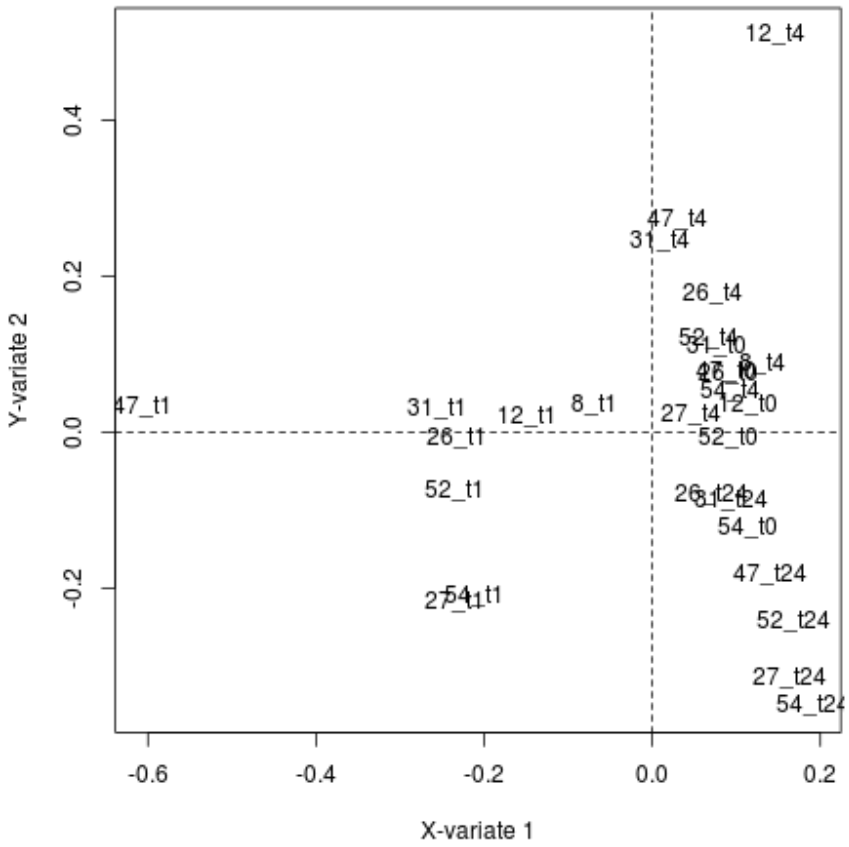
Partial Least Squares

Maintenant que les données sont prêtes, nous allons faire une PLS. Nous garderons pour cette première analyse 15 données d'expression sur deux dimensions (seulement 13 pour les phénotypes). Dans le nuage des individus, les individus sont identifiés par leur numéro d'expérience et par le pas de temps “t0”, “t1”, “t4”, “t24” après injection d'ACTH.

```
kept.indiv <- indiv[-all.missing, ]
names.indiv <- paste(kept.indiv[, 2], kept.indiv[, 3], sep = "_t")
rownames(imputed.pheno) <- names.indiv
rownames(imputed.genes) <- names.indiv
pls1 <- spls(imputed.pheno, imputed.genes, ncomp = 2, mode =
"canonical", keepX = rep(13,
    2), keepY = rep(15, 2))
var.coord <- plotVar(pls1, comp = 1:2, X.label = TRUE, Y.label =
TRUE)
```




```
plotIndiv(pls1, comp = 1:2, ind.names = TRUE, rep.space =
"XY-variate")
```


12_t4
47_t4
31_t4
26_t4
52_t4
54_t4
47_t1
31_t1
12_t1
8_t1
27_t4
12_t0
26_t1
52_t0
52_t1
54_t0
47_t24
52_t24
27_t24
54_t24
Y-variate 2
X-variate 1
0.4
0.2
0.0
-0.2
-0.6
-0.4
-0.2
0.0
0.2

```
seuil <- 0.6
axes.genes.pls1 <- data.frame(coord1 = factor(rep("peu
représenté", nrow(var.coord$coord.Y)),
    levels = c("peu représenté", "positif", "negatif")), coord2 =
factor(rep("peu représenté",
    nrow(var.coord$coord.Y)), levels = c("peu représenté",
"positif", "negatif")))
axes.genes.pls1[var.coord$coord.Y > seuil] <- "positif"
axes.genes.pls1[var.coord$coord.Y < -seuil] <- "negatif"
rownames(axes.genes.pls1) <- rownames(var.coord$coord.Y)

axes.pheno.pls1 <- data.frame(coord1 = factor(rep("peu
représenté", nrow(var.coord$coord.X)),
    levels = c("peu représenté", "positif", "negatif")), coord2 =
factor(rep("peu représenté",
    nrow(var.coord$coord.X)), levels = c("peu représenté",
"positif", "negatif")))
axes.pheno.pls1[var.coord$coord.X > seuil] <- "positif"
axes.pheno.pls1[var.coord$coord.X < -seuil] <- "negatif"
rownames(axes.pheno.pls1) <- rownames(var.coord$coord.X)

axes.pheno.pls1
```

```
##                    coord1         coord2
## cortisol           negatif peu représenté
## glucose     peu représenté        positif
## lactate     peu représenté peu représenté
## NEFA               negatif peu représenté
## uree        peu représenté peu représenté
## nleuco             positif peu représenté
## plympho            positif peu représenté
## pgranulo    peu représenté peu représenté
## nlympho            positif peu représenté
## ngranulo    peu représenté peu représenté
## nrouges            positif peu représenté
## hematocrite        positif peu représenté
## hemoglob           positif peu représenté
```

```
axes.genes.pls1
```

```
##               coord1         coord2
## 6703  peu représenté        positif
## 39765        negatif peu représenté
## 14403 peu représenté        positif
## 8941  peu représenté        positif
## 6893         negatif peu représenté
## 3989         negatif peu représenté
## 3390         negatif peu représenté
## 9884         negatif peu représenté
## 47360        negatif peu représenté
## 51211 peu représenté        positif
## 39878        negatif peu représenté
## 1128         negatif peu représenté
## 2501         negatif peu représenté
## 7035  peu représenté        positif
## 53569        negatif peu représenté
## 38206 peu représenté        positif
## 12322 peu représenté        positif
## 11572 peu représenté        negatif
## 40897        negatif peu représenté
## 36524 peu représenté        positif
## 12535 peu représenté        negatif
## 6220  peu représenté        positif
## 26495        negatif peu représenté
## 35225        negatif peu représenté
## 30646 peu représenté        positif
## 51733        negatif peu représenté
## 22774 peu représenté        positif
## 470          negatif peu représenté
## 46673 peu représenté        positif
## 37492 peu représenté        positif
```

L'individu “47_t1” (cochon d'identifiant 47 au temps 1) est très éloigné des autres. De plus, il est composé de 8 valeurs liées à l'imputation sur 13 (8 valeurs manquantes dans les données d'origine).

```
rownames(no.allmissing.pheno) <- names.indiv
apply(no.allmissing.pheno, 1, function(x) sum(is.na(x)))
```

```
##   8_t1   8_t4  12_t0  12_t1  12_t4  26_t0  26_t1  26_t4 26_t24
27_t1
##      0      0      0      0      0      0      0      0
5      0
##  27_t4 27_t24  31_t0  31_t1  31_t4 31_t24  47_t0  47_t1  47_t4
47_t24
##      0      5      0      0      0      5      0      8
0      5
##  52_t0  52_t1  52_t4 52_t24  54_t0  54_t1  54_t4 54_t24
##      0      0      0      5      0      0      0      5
```

Afin d'avoir une analyse plus fine, plus pertinente, nous n'allons pas le prendre en compte dans la suite de l'étude.

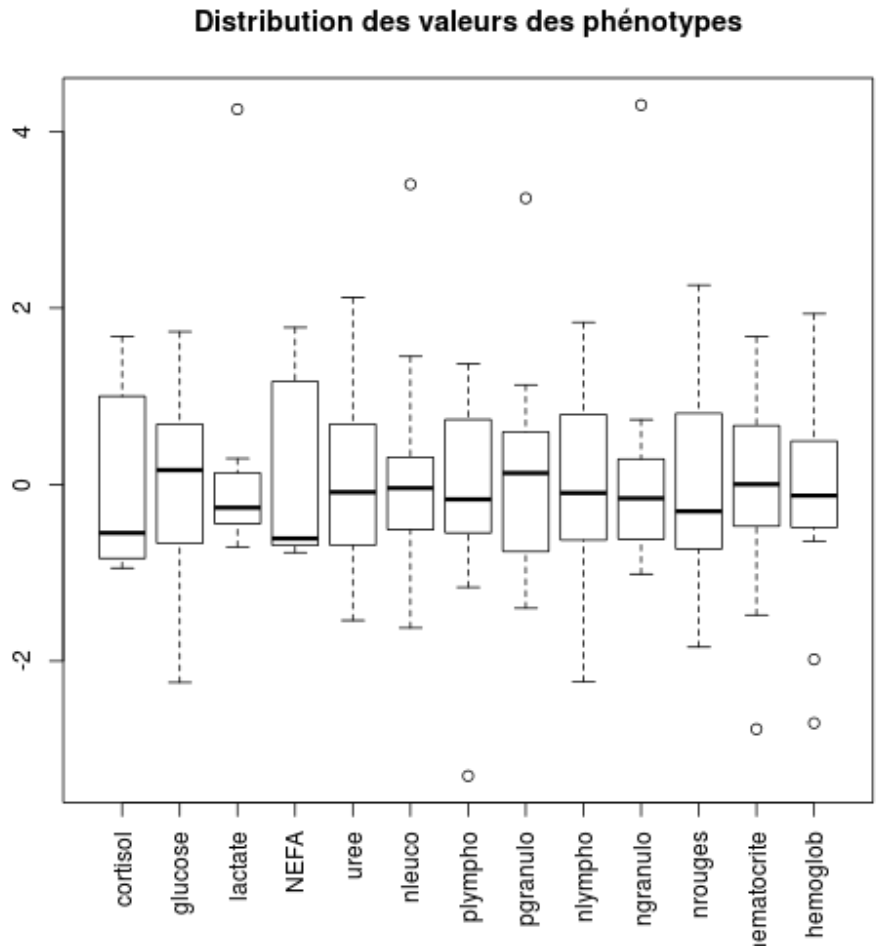
PLS sans l'individu “47_t1”

Nous allons donc refaire une imputation de manière à enlever l'individu “47_t1” (ici tous les individus avec plus de 7 phénotypes manquants sont exclus de l'analyse).

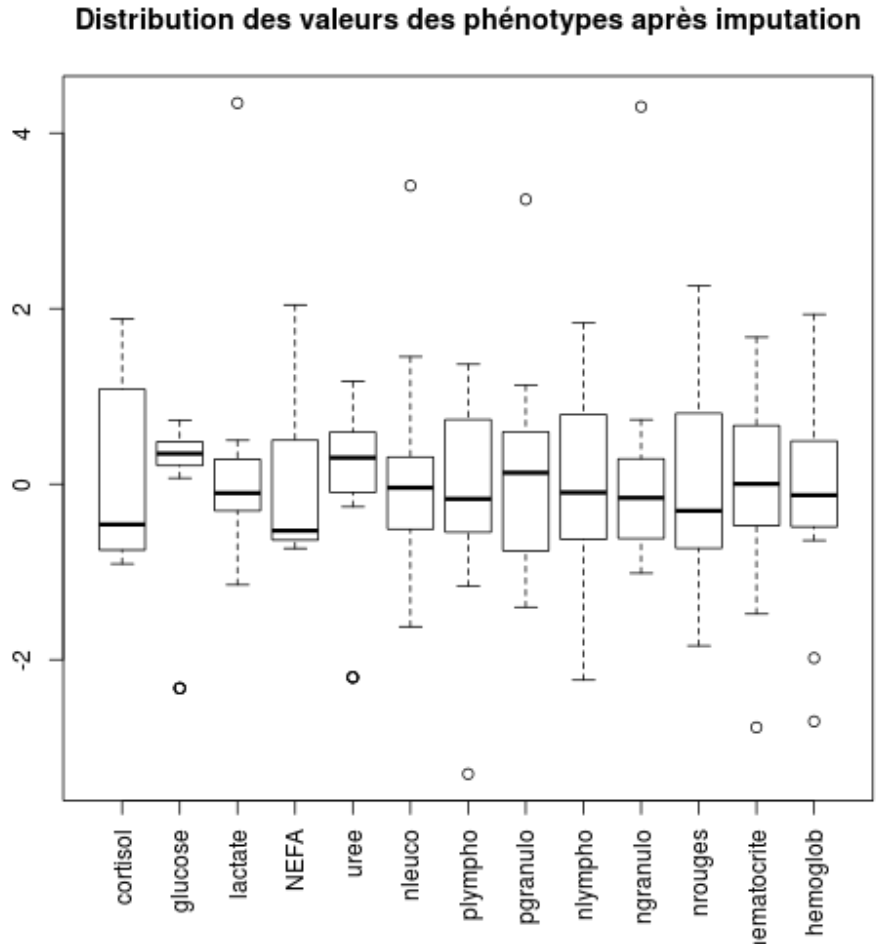
```
all.missing <- which(missing.in.rows > 7)
no.allmissing.pheno <- phenotypes[-all.missing, ]
kept.genes <- sign.genes[-all.missing, ]
kept.indiv <- indiv[-all.missing, ]
names.indiv <- paste(kept.indiv[, 2], kept.indiv[, 3], sep = "_t")

imputed.pheno <- kNNImpute(no.allmissing.pheno, 3, verbose =
TRUE)$x
```

```
## [1] "Computing distance matrix..."
## [1] "Distance matrix complete"
## [1] "Imputing row 9"
## [1] "Imputing row 12"
## [1] "Imputing row 16"
## [1] "Imputing row 19"
## [1] "Imputing row 23"
## [1] "Imputing row 27"
```

```
where.miss <- which(apply(kept.genes, 1, function(x)
sum(is.na(x))) != 0)
nearest.nei <- names(sort(as.matrix(dist(kept.genes))[11, ])[2:4])
col.miss <- which(apply(kept.genes, 2, function(x) sum(is.na(x)))
!= 0)
imputed.genes <- kept.genes
imputed.genes[where.miss, col.miss] <-
mean(kept.genes[nearest.nei, col.miss])
boxplot(scale(phenotypes), main = "Distribution des valeurs des
phénotypes",
    las = 3)
```

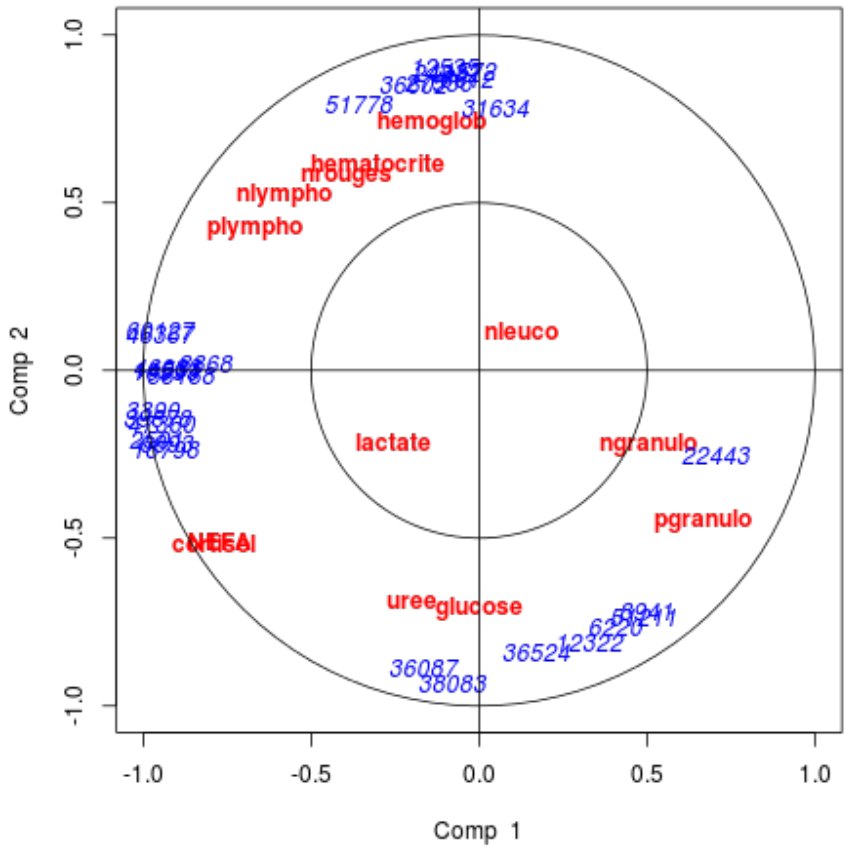



```
boxplot(scale(imputed.pheno), main = "Distribution des valeurs des
phénotypes après imputation",
    las = 3)
```

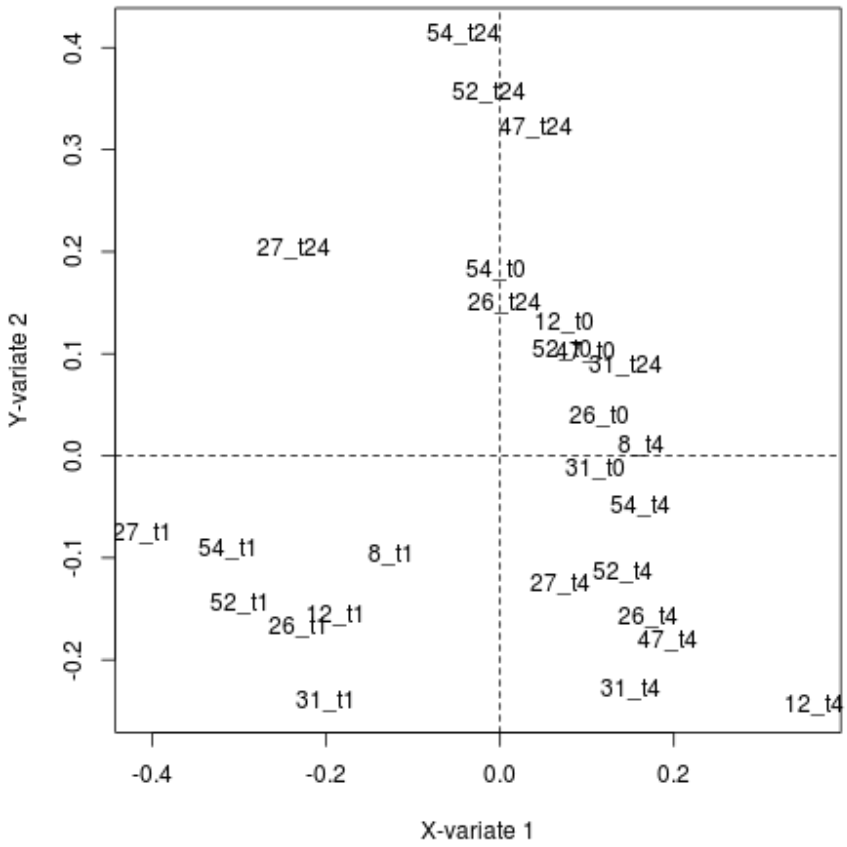


On s'apperçoit que sans l'individu 47_t1, les distributions avant et après imputation se ressemblent plus. En effet, ses valeurs très faibles n'apparaissent plus. Ensuite nous ré-éxécutons la PLS sans celui-ci dans cette commande :

```
pls2 <- spls(imputed.pheno, imputed.genes, ncomp = 2, mode =
"canonical", keepX = rep(13,
    2), keepY = rep(15, 2))
var.coord <- plotVar(pls2, comp = 1:2, rad.in = 0.5, X.label =
TRUE, Y.label = TRUE,
    pch = NULL, cex = NULL, col = NULL, font = NULL)
```




```
plotIndiv(pls2, comp = 1:2, ind.names = names.indiv, rep.space =
"XY-variate")
```


54_t24
52_t24
47_t24
27_t24
54_t0
26_t24
12_t0
31_t24
26_t0
8_t4
31_t0
54_t4
27_t1
54_t1
8_t1
52_t4
27_t4
52_t1
12_t1
26_t1
26_t4
47_t4
31_t4
31_t1
12_t4
0.4
0.3
0.2
0.1
0.0
-0.1
-0.2
-0.4
-0.2
0.0
0.2
Y-variate 2
X-variate 1

```
seuil <- 0.6
axes.genes.pls2 <- data.frame(coord1 = factor(rep("peu
représenté", nrow(var.coord$coord.Y)),
    levels = c("peu représenté", "positif", "negatif")), coord2 =
factor(rep("peu représenté",
    nrow(var.coord$coord.Y)), levels = c("peu représenté",
"positif", "negatif")))
axes.genes.pls2[var.coord$coord.Y > seuil] <- "positif"
axes.genes.pls2[var.coord$coord.Y < -seuil] <- "negatif"
rownames(axes.genes.pls2) <- rownames(var.coord$coord.Y)

axes.pheno.pls2 <- data.frame(coord1 = factor(rep("peu
représenté", nrow(var.coord$coord.X)),
    levels = c("peu représenté", "positif", "negatif")), coord2 =
factor(rep("peu représenté",
    nrow(var.coord$coord.X)), levels = c("peu représenté",
"positif", "negatif")))
axes.pheno.pls2[var.coord$coord.X > seuil] <- "positif"
axes.pheno.pls2[var.coord$coord.X < -seuil] <- "negatif"
rownames(axes.pheno.pls2) <- rownames(var.coord$coord.X)

axes.pheno.pls2
```

```
##                    coord1         coord2
## cortisol           negatif peu représenté
## glucose     peu représenté        negatif
## lactate     peu représenté peu représenté
## NEFA               negatif peu représenté
## uree        peu représenté        negatif
## nleuco      peu représenté peu représenté
## plympho            negatif peu représenté
## pgranulo           positif peu représenté
## nlympho     peu représenté peu représenté
## ngranulo    peu représenté peu représenté
## nrouges     peu représenté peu représenté
## hematocrite peu représenté        positif
## hemoglob    peu représenté        positif
```

```
axes.genes.pls2
```

```
##                coord1         coord2
## 16593         negatif peu représenté
## 8941  peu représenté         negatif
## 6893          negatif peu représenté
## 3390          negatif peu représenté
## 51778 peu représenté         positif
## 46367         negatif peu représenté
## 47360         negatif peu représenté
## 51211 peu représenté         negatif
## 39878         negatif peu représenté
## 3868          negatif peu représenté
## 18798         negatif peu représenté
## 2501          negatif peu représenté
## 38083 peu représenté         negatif
## 36087 peu représenté         negatif
## 58188         negatif peu représenté
## 14531         negatif peu représenté
## 46083         negatif peu représenté
## 12322 peu représenté         negatif
## 11572 peu représenté         positif
## 36524 peu représenté         negatif
## 60127         negatif peu représenté
## 12535 peu représenté         positif
## 27556 peu représenté         positif
## 36602 peu représenté         positif
## 6220  peu représenté         negatif
## 8612  peu représenté         positif
## 31634 peu représenté         positif
## 24932 peu représenté         positif
## 22443         positif peu représenté
```

```
p2.neg.axe1 <- list()
p2.neg.axe1[[1]] <- which(as.matrix(axes.genes.pls2)[, 1] ==
"negatif")
p2.neg.axe1[[2]] <- which(as.matrix(axes.pheno.pls2)[, 1] ==
"negatif")
p2.pos.axe1 <- list()
p2.pos.axe1[[1]] <- which(as.matrix(axes.genes.pls2)[, 1] ==
"positif")
p2.pos.axe1[[2]] <- which(as.matrix(axes.pheno.pls2)[, 1] ==
"positif")
p2.neg.axe2 <- list()
p2.neg.axe2[[1]] <- which(as.matrix(axes.genes.pls2)[, 2] ==
"negatif")
p2.neg.axe2[[2]] <- which(as.matrix(axes.pheno.pls2)[, 2] ==
"negatif")
p2.pos.axe2 <- list()
p2.pos.axe2[[1]] <- which(as.matrix(axes.genes.pls2)[, 2] ==
"positif")
p2.pos.axe2[[2]] <- which(as.matrix(axes.pheno.pls2)[, 2] ==
"positif")
```

L'analyse effectuée ici retient 15 gènes sur 2 dimensions. Elle ne prend pas en compte l'individu 47_t1 contrairement au premier essai. En premier lieu nous donnerons une interprétation des axes de cette PLS, puis nous essaierons de la comparer à la premier analyse pour voir quel était l'impact de l'individu 47_t1.

L'axe 1 permet donc de regrouper les variables suivantes du côté négatif

```
p2.neg.axe1
```

```
## [[1]]
## 16593  6893  3390 46367 47360 39878  3868 18798  2501 58188
14531 46083
##     1     3     4     6     7     9    10    11    12    15
16    17
## 60127
##    21
##
## [[2]]
## cortisol     NEFA  plympho
##        1        4        7
```

Ces variables s'opposent à :

```
p2.pos.axe1
```

```
## [[1]]
## 22443
##    29
##
## [[2]]
## pgranulo
##        8
```

dont les valeurs fortes sont rencontrés chez les individus au temps t4 et les valeurs faibles chez les individus au temps t1. Les autres individus sont placés plutôt proche de l'axe des abscisses, souvent coté positif. Le second axe nous sert aussi à visualiser les rapprochements entre phénotypes et expressions de gènes. Voici donc les variables côté négatif de cet axe :

```
p2.neg.axe2
```

```
## [[1]]
##  8941 51211 38083 36087 12322 36524  6220
##     2     8    13    14    18    20    25
##
## [[2]]
## glucose    uree
##       2       5
```

Ces variables s'opposent aux variables

```
p2.pos.axe2
```

```
## [[1]]
## 51778 11572 12535 27556 36602  8612 31634 24932
##     5    19    22    23    24    26    27    28
##
## [[2]]
## hematocrite    hemoglob
##          12          13
```

Sur ce second axe les valeurs fortes se trouvent chez les individus au temps t24 et t0. Du coté négatif, on retrouve plutôt les valeurs des temps t4 et t1.

On notera que les individus les plus éloignés (en particulier au temps 24) avaient plusieurs phénotypes manquants et ont donc dû être imputés. Il faut donc prendre avec prudence les conclusions concernant ces individus.

On se sert des commandes suivantes pour trouver les éventuels phénotypes ou gènes qui n'auraient été représentés sur aucun des 2 axes :

```
which(as.matrix(axes.genes.pls2)[, 1] == "peu représenté" &
as.matrix(axes.genes.pls2)[,
    2] == "peu représenté")
```

```
## named integer(0)
```

```
which(as.matrix(axes.pheno.pls2)[, 1] == "peu représenté" &
as.matrix(axes.pheno.pls2)[,
    2] == "peu représenté")
```

```
##  lactate   nleuco  nlympho ngranulo  nrouges
##        3        6        9       10       11
```

Comparaison entre les 2 analyses (avec ou sans l'individu “47_t1”“)

Ici, nous allons prendre les gènes retenus lors de la première pls et voir si ils sont retrouvés dans la seconde analyse.

```
# just a comparaison for gene kepts is individual n°47_1 is kept
or not
comparaison <- matrix(nrow = nrow(axes.genes.pls1))
# creat a matrix, if the pls1 gene is found in the pls2 gene kept,
we keep
# the name of gene, if not,we say not found
for (s1 in 1:nrow(axes.genes.pls1)) {
    rep <- "non trouvé"
    for (s2 in 1:nrow(axes.genes.pls2)) {
        if (labels(axes.genes.pls1)[[1]][s1] ==
labels(axes.genes.pls2)[[1]][s2]) {
            rep <- labels(axes.genes.pls1)[[1]][s1]
        }
    }
    comparaison[s1] <- rep
}
comparaison[which(comparaison != "non trouvé")]
```

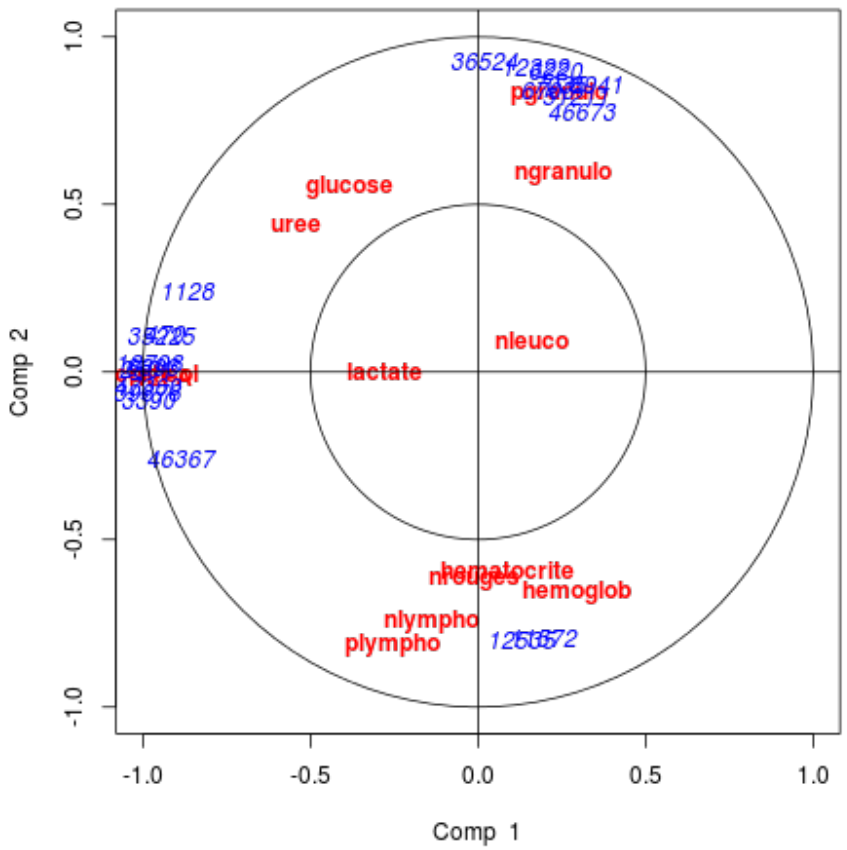
```
##  [1] "8941"  "6893"  "3390"  "47360" "51211" "39878" "2501"
"12322"
##  [9] "11572" "36524" "12535" "6220"
```

Sur les 30 gènes qui étaient sélectionnés sur une des 2 dimensions lors de la première analyse, on voit que 12 le sont encore sur l'analyse sans le numéro 47_1.

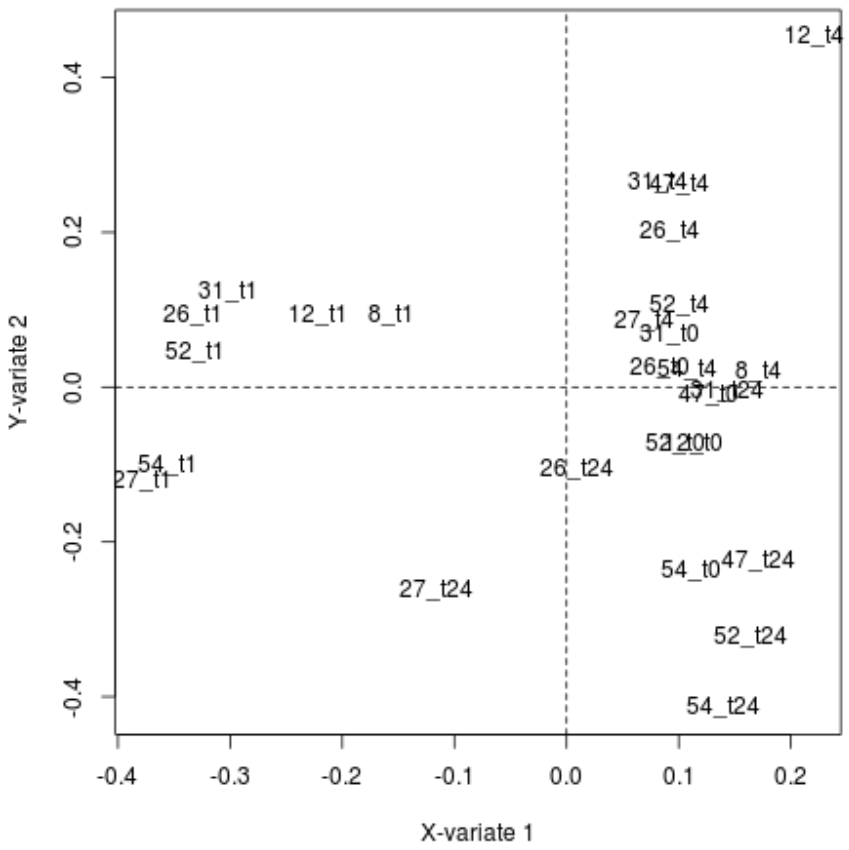
PLS avec 10 variables seulement retenues

Ici, nous allons sélectionner 10 gènes sur 2 dimensions au lieu des 15 précedemment.

```
pls3 <- spls(imputed.pheno, imputed.genes, ncomp = 2, mode =
"canonical", keepX = rep(10,
    2), keepY = rep(10, 2))

var.coord <- plotVar(pls3, comp = 1:2, rad.in = 0.5, X.label =
TRUE, Y.label = TRUE,
    pch = NULL, cex = NULL, col = NULL, font = NULL)
```



```
plotIndiv(pls3, comp = 1:2, ind.names = names.indiv, rep.space =
"XY-variate",
    x.label = NULL, y.label = NULL, col = "black", cex = 1, pch =
1)
```



```
seuil <- 0.6
axes.genes.pls3 <- data.frame(coord1 = factor(rep("peu
représenté", nrow(var.coord$coord.Y)),
    levels = c("peu représenté", "positif", "negatif")), coord2 =
factor(rep("peu représenté",
    nrow(var.coord$coord.Y)), levels = c("peu représenté",
"positif", "negatif")))
axes.genes.pls3[var.coord$coord.Y > seuil] <- "positif"
axes.genes.pls3[var.coord$coord.Y < -seuil] <- "negatif"
rownames(axes.genes.pls3) <- rownames(var.coord$coord.Y)

axes.pheno.pls3 <- data.frame(coord1 = factor(rep("peu
représenté", nrow(var.coord$coord.X)),
    levels = c("peu représenté", "positif", "negatif")), coord2 =
factor(rep("peu représenté",
    nrow(var.coord$coord.X)), levels = c("peu représenté",
"positif", "negatif")))
axes.pheno.pls3[var.coord$coord.X > seuil] <- "positif"
axes.pheno.pls3[var.coord$coord.X < -seuil] <- "negatif"
rownames(axes.pheno.pls3) <- rownames(var.coord$coord.X)

axes.pheno.pls3
```

```
##                    coord1         coord2
## cortisol           negatif peu représenté
## glucose     peu représenté peu représenté
## lactate     peu représenté peu représenté
## NEFA               negatif peu représenté
## uree        peu représenté peu représenté
## nleuco      peu représenté peu représenté
## plympho     peu représenté        negatif
## pgranulo    peu représenté        positif
## nlympho     peu représenté        negatif
## ngranulo    peu représenté peu représenté
## nrouges     peu représenté        negatif
## hematocrite peu représenté peu représenté
## hemoglob    peu représenté        negatif
```

```
axes.genes.pls3
```

```
##               coord1         coord2
## 6703  peu représenté        positif
## 8941  peu représenté        positif
## 6893         negatif peu représenté
## 3390         negatif peu représenté
## 46367        negatif peu représenté
## 47360        negatif peu représenté
## 51211 peu représenté        positif
## 39878        negatif peu représenté
## 18798        negatif peu représenté
## 1128         negatif peu représenté
## 2501         negatif peu représenté
## 7035  peu représenté        positif
## 12322 peu représenté        positif
## 11572 peu représenté        negatif
## 36524 peu représenté        positif
## 12535 peu représenté        negatif
## 6220  peu représenté        positif
## 35225        negatif peu représenté
## 470          negatif peu représenté
## 46673 peu représenté        positif
```

```
p3.neg.axe1 <- list()
p3.neg.axe1[[1]] <- which(as.matrix(axes.genes.pls3)[, 1] ==
"negatif")
p3.neg.axe1[[2]] <- which(as.matrix(axes.pheno.pls3)[, 1] ==
"negatif")
p3.pos.axe1 <- list()
p3.pos.axe1[[1]] <- which(as.matrix(axes.genes.pls3)[, 1] ==
"positif")
p3.pos.axe1[[2]] <- which(as.matrix(axes.pheno.pls3)[, 1] ==
"positif")
p3.neg.axe2 <- list()
p3.neg.axe2[[1]] <- which(as.matrix(axes.genes.pls3)[, 2] ==
"negatif")
p3.neg.axe2[[2]] <- which(as.matrix(axes.pheno.pls3)[, 2] ==
"negatif")
p3.pos.axe2 <- list()
p3.pos.axe2[[1]] <- which(as.matrix(axes.genes.pls3)[, 2] ==
"positif")
p3.pos.axe2[[2]] <- which(as.matrix(axes.pheno.pls3)[, 2] ==
"positif")
```

Cette PLS contient 10 gènes et 10 phénotypes sur chacune des 2 dimensions.

Sur l'axe 1, elle rassemble côté négatif les variables :

```
p3.neg.axe1
```

```
## [[1]]
##  6893  3390 46367 47360 39878 18798  1128  2501 35225   470
##     3     4     5     6     8     9    10    11    18    19
##
## [[2]]
## cortisol     NEFA
##        1        4
```

Et celle côté positif :

```
p3.pos.axe1
```

```
## [[1]]
## named integer(0)
##
## [[2]]
## named integer(0)
```

(aucune variable)

On peut remarquer au niveau de la position des individus sur cet axe que les faibles valeurs sont prises par les individus au temps t1. Au contraire, la majorité des auters individus ont des valeurs positives, même si peu élevées. Sur l'axe2, ce sont les variables suivantes qui sont rassemblées côté négatif

```
p3.neg.axe2
```

```
## [[1]]
## 11572 12535
##    14    16
##
## [[2]]
##  plympho  nlympho  nrouges hemoglob
##        7        9       11       13
```

et celle côté positif sont :

```
p3.pos.axe2
```

```
## [[1]]
##  6703  8941 51211  7035 12322 36524  6220 46673
##     1     2     7    12    13    15    17    20
##
## [[2]]
## pgranulo
##        8
```

Cet axe là permet une séparation au niveau des individus t4 et t24 ou t0. En effet, les individus au temps t4 ont des valeurs plus élevées, tandis que ceux au temps t0 ou t24 ont des valeurs faibles. On note que les individus 27_t24 ; 47_t24 ; 52_t24 ; 54_t24, qui peuvent paraître éloignés des autres, ont subi une imputation sur 5 variables phénotypes. Enfin, toutes les variables ont été gardées dans la PLS mais ne sont pas corrélées fortement aux axes 1 et 2 (d'abord pour les gènes, ensuite pour les phénotypes) :

```
which(as.matrix(axes.genes.pls3)[, 1] == "peu représenté" &
as.matrix(axes.genes.pls3)[,
    2] == "peu représenté")
```

```
## named integer(0)
```

```
which(as.matrix(axes.pheno.pls3)[, 1] == "peu représenté" &
as.matrix(axes.pheno.pls3)[,
    2] == "peu représenté")
```

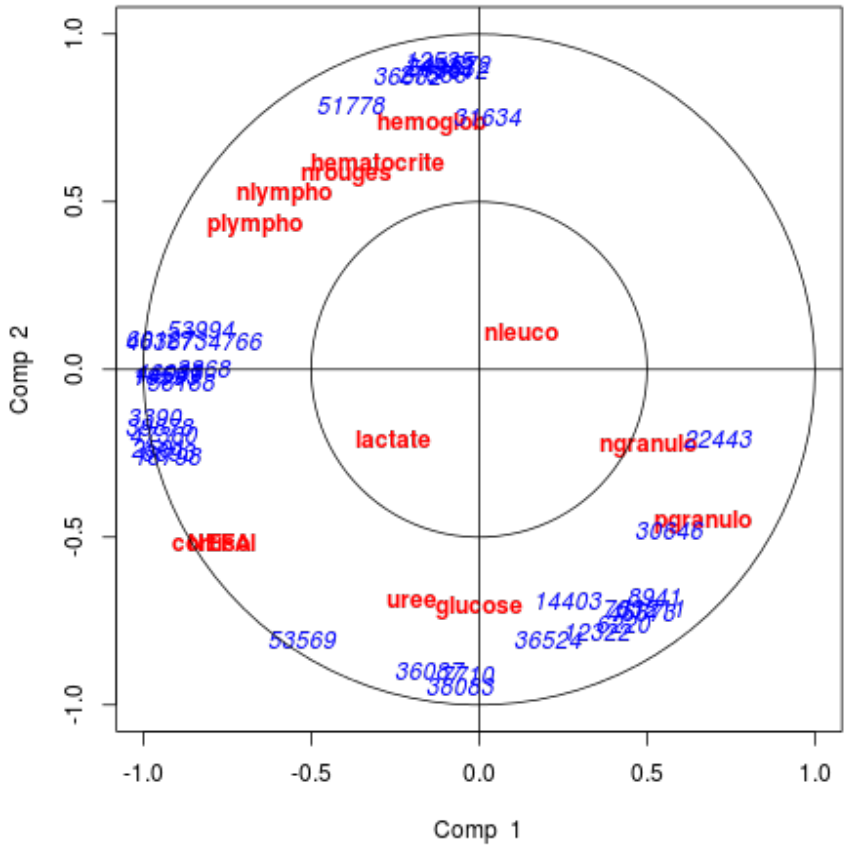
```
##     glucose     lactate        uree      nleuco    ngranulo
hematocrite
##           2           3           5           6
10          12
```

Certains phénotypes non corrélés sont les mêmes que lors de la pls2 (15 variables, sans l'individu numéro 47_t1).

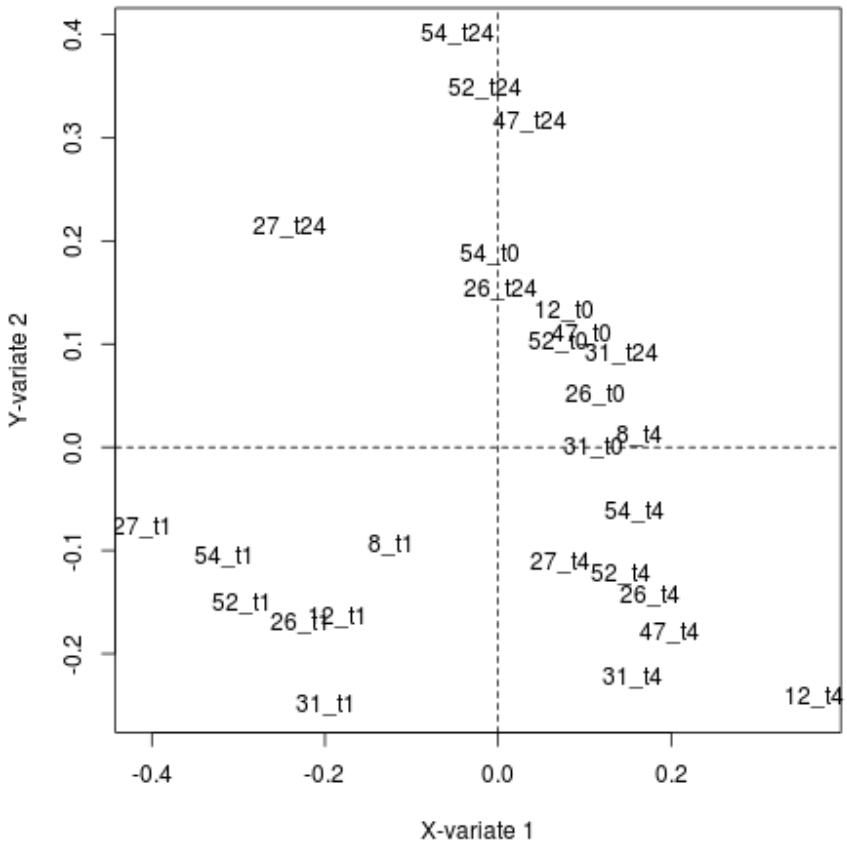
PLS avec 20 variables retenues

Ici nous allons faire une dernière PLS avec plus de variables retenues : 20 par dimension (bien sûr seulement 13 phénotypes sont disponibles).

```
pls4 <- spls(imputed.pheno, imputed.genes, ncomp = 2, mode =
"canonical", keepX = rep(13,
    2), keepY = rep(20, 2))
var.coord <- plotVar(pls4, comp = 1:2, rad.in = 0.5, X.label =
TRUE, Y.label = TRUE,
    pch = NULL, cex = NULL, col = NULL, font = NULL)
```




```
plotIndiv(pls4, comp = 1:2, ind.names = names.indiv, rep.space =
"XY-variate",
    x.label = NULL, y.label = NULL, col = "black", cex = 1, pch =
1)
```


54_t24
52_t24
47_t24
27_t24
54_t0
26_t24
12_t0
47_t0
52_t0
31_t24
26_t0
8_t4
31_t0
54_t4
27_t1
54_t1
8_t1
27_t4
52_t4
26_t4
52_t1
26_t1
12_t1
47_t4
31_t4
12_t4
31_t1
0.4
0.3
0.2
0.1
0.0
-0.1
-0.2
Y-variate 2
-0.4
-0.2
0.0
0.2
X-variate 1

```
seuil <- 0.6
axes.genes.pls4 <- data.frame(coord1 = factor(rep("peu
représenté", nrow(var.coord$coord.Y)),
    levels = c("peu représenté", "positif", "negatif")), coord2 =
factor(rep("peu représenté",
    nrow(var.coord$coord.Y)), levels = c("peu représenté",
"positif", "negatif")))
axes.genes.pls4[var.coord$coord.Y > seuil] <- "positif"
axes.genes.pls4[var.coord$coord.Y < -seuil] <- "negatif"
rownames(axes.genes.pls4) <- rownames(var.coord$coord.Y)

axes.pheno.pls4 <- data.frame(coord1 = factor(rep("peu
représenté", nrow(var.coord$coord.X)),
    levels = c("peu représenté", "positif", "negatif")), coord2 =
factor(rep("peu représenté",
    nrow(var.coord$coord.X)), levels = c("peu représenté",
"positif", "negatif")))
axes.pheno.pls4[var.coord$coord.X > seuil] <- "positif"
axes.pheno.pls4[var.coord$coord.X < -seuil] <- "negatif"
rownames(axes.pheno.pls4) <- rownames(var.coord$coord.X)

axes.pheno.pls4
```

```
##                    coord1         coord2
## cortisol           negatif peu représenté
## glucose     peu représenté        negatif
## lactate     peu représenté peu représenté
## NEFA               negatif peu représenté
## uree        peu représenté        negatif
## nleuco      peu représenté peu représenté
## plympho            negatif peu représenté
## pgranulo           positif peu représenté
## nlympho     peu représenté peu représenté
## ngranulo    peu représenté peu représenté
## nrouges     peu représenté peu représenté
## hematocrite peu représenté        positif
## hemoglob    peu représenté        positif
```

```
axes.genes.pls4
```

```
##               coord1         coord2
## 16593        negatif peu représenté
## 14403 peu représenté        negatif
## 58383 peu représenté        positif
## 8941  peu représenté        negatif
## 6893         negatif peu représenté
## 3390         negatif peu représenté
## 51778 peu représenté        positif
## 46367        negatif peu représenté
## 47360        negatif peu représenté
## 51211 peu représenté        negatif
## 39878        negatif peu représenté
## 3868         negatif peu représenté
## 18798        negatif peu représenté
## 2501         negatif peu représenté
## 38083 peu représenté        negatif
## 36087 peu représenté        negatif
## 58188        negatif peu représenté
## 34766        negatif peu représenté
## 7035  peu représenté        negatif
## 7710  peu représenté        negatif
## 53569 peu représenté        negatif
## 53994        negatif peu représenté
## 14531        negatif peu représenté
## 46083        negatif peu représenté
## 12322 peu représenté        negatif
## 11572 peu représenté        positif
## 36524 peu représenté        negatif
## 60127        negatif peu représenté
## 12535 peu représenté        positif
## 27556 peu représenté        positif
## 36602 peu représenté        positif
## 6220  peu représenté        negatif
## 8612  peu représenté        positif
## 30646 peu représenté peu représenté
## 31634 peu représenté        positif
## 24932 peu représenté        positif
## 46673 peu représenté        negatif
## 22443        positif peu représenté
```

```
p4.neg.axe1 <- list()
p4.neg.axe1[[1]] <- which(as.matrix(axes.genes.pls4)[, 1] ==
"negatif")
p4.neg.axe1[[2]] <- which(as.matrix(axes.pheno.pls4)[, 1] ==
"negatif")
p4.pos.axe1 <- list()
p4.pos.axe1[[1]] <- which(as.matrix(axes.genes.pls4)[, 1] ==
"positif")
p4.pos.axe1[[2]] <- which(as.matrix(axes.pheno.pls4)[, 1] ==
"positif")
p4.neg.axe2 <- list()
p4.neg.axe2[[1]] <- which(as.matrix(axes.genes.pls4)[, 2] ==
"negatif")
p4.neg.axe2[[2]] <- which(as.matrix(axes.pheno.pls4)[, 2] ==
"negatif")
p4.pos.axe2 <- list()
p4.pos.axe2[[1]] <- which(as.matrix(axes.genes.pls4)[, 2] ==
"positif")
p4.pos.axe2[[2]] <- which(as.matrix(axes.pheno.pls4)[, 2] ==
"positif")
```

Cette PLS avec 20 gènes et 13 phénotypes utilisés permet de rapprocher les variables suivantes au seuil de corrélation 0.6. On peut donc voir l'axe comme une opposition entre les variables côté négatif :

```
p4.neg.axe1
```

```
## [[1]]
## 16593  6893  3390 46367 47360 39878  3868 18798  2501 58188
34766 53994
##     1     5     6     8     9    11    12    13    14    17
18    22
## 14531 46083 60127
##    23    24    28
##
## [[2]]
## cortisol     NEFA  plympho
##        1        4        7
```

Et celle côté positif :

```
p4.pos.axe1
```

```
## [[1]]
## 22443
##    38
##
## [[2]]
## pgranulo
##        8
```

Au niveau des individus, on voit sur l'axe 1 des valeurs très faibles au temps t1. Les valeurs au temps t4 sont plutôt fortes sur cet axe, tandis que les individus à t0 et t24 ont des valeurs proches de 0 voir un peu positives (hormis l'individu 27_t24 qui est toujours excentré, possiblement à cause du processus d'imputation).

De la même manière, le second axe permet lui aussi de regrouper des variables entre elles, en particulier des phénotypes avec des gènes. Les variables côté négatif sont donc :

```
p4.neg.axe2
```

```
## [[1]]
## 14403  8941 51211 38083 36087  7035  7710 53569 12322 36524
6220 46673
##     2     4    10    15    16    19    20    21    25    27
32    37
##
## [[2]]
## glucose    uree
##       2       5
```

Tandis que celle côté positif sont :

```
p4.pos.axe2
```

```
## [[1]]
## 58383 51778 11572 12535 27556 36602  8612 31634 24932
##     3     7    26    29    30    31    33    35    36
##
## [[2]]
## hematocrite    hemoglob
##          12          13
```

On retrouve donc sur ce second axe les individus aux temps t1 et t4 qui ont des valeurs faibles, et de l'autre coté de l'axe les valeurs aux temps t0 et t24. Les 4 valeurs aux temps 24 qui sont très éloignées des autres sont à nouveau celle qui ont subi le

plus grand nombre d'imputation.

Enfin, les variables suivantes ont été gardées dans la PLS mais ne sont pas représentées significativement sur les dimensions 1 et 2.

```
which(as.matrix(axes.genes.pls4)[, 1] == "peu représenté" &
as.matrix(axes.genes.pls4)[,
    2] == "peu représenté")
```

```
## 30646
##    34
```

```
which(as.matrix(axes.pheno.pls4)[, 1] == "peu représenté" &
as.matrix(axes.pheno.pls4)[,
    2] == "peu représenté")
```

```
##  lactate   nleuco  nlympho ngranulo  nrouges
##        3        6        9       10       11
```

Ces phénotypes peu représentés reviennent souvent plusieurs fois au cours des différentes analyses.

Comparaison entre les PLS

Axe 1 négatif

```
p2.neg.axe1
```

```
## [[1]]
## 16593  6893  3390 46367 47360 39878  3868 18798  2501 58188
14531 46083
##     1     3     4     6     7     9    10    11    12    15
16    17
## 60127
##    21
##
## [[2]]
## cortisol     NEFA  plympho
##        1        4        7
```

```
p3.neg.axe1
```

```
## [[1]]
##  6893  3390 46367 47360 39878 18798  1128  2501 35225   470
##     3     4     5     6     8     9    10    11    18    19
##
## [[2]]
## cortisol     NEFA
##        1        4
```

```
p4.neg.axe1
```

```
## [[1]]
## 16593  6893  3390 46367 47360 39878  3868 18798  2501 58188
34766 53994
##     1     5     6     8     9    11    12    13    14    17
18    22
## 14531 46083 60127
##    23    24    28
##
## [[2]]
## cortisol     NEFA  plympho
##        1        4        7
```

Au niveau des phenotypes retrouvés ici, le cortisol est retrouvé 3 fois, ainsi que NEFA. La variable plympho est elle retrouvée 2 fois.

Les gènes suivants sont retrouvés plusieurs fois : 16593 ; 3868 ; 58188 ; 14531 ; 46083 (2 fois) et 6893 ; 3390 ; 46367 ; 47360 ; 39878 ; 18798 ; 2501 (3 fois)

Axe 1 positif

```
p2.pos.axe1
```

```
## [[1]]
## 22443
##    29
##
## [[2]]
## pgranulo
##        8
```

```
p3.pos.axe1
```

```
## [[1]]
## named integer(0)
##
## [[2]]
## named integer(0)
```

```
p4.pos.axe1
```

```
## [[1]]
## 22443
##    38
##
## [[2]]
## pgranulo
##        8
```

De l'autre côté de l'axe, on retrouve le phenotype pgranulo et seulement le gène 22443, tous les 2 lors des pls à 15 et 20 variables.

Axe 2 négatif

```
p2.neg.axe2
```

```
## [[1]]
##  8941 51211 38083 36087 12322 36524  6220
##     2     8    13    14    18    20    25
##
## [[2]]
## glucose    uree
##       2       5
```

```
p3.neg.axe2
```

```
## [[1]]
## 11572 12535
##    14    16
##
## [[2]]
##  plympho  nlympho  nrouges hemoglob
##        7        9       11       13
```

```
p4.neg.axe2
```

```
## [[1]]
## 14403  8941 51211 38083 36087  7035  7710 53569 12322 36524
6220 46673
##     2     4    10    15    16    19    20    21    25    27
32    37
##
## [[2]]
## glucose    uree
##       2       5
```

Le second axe semble différent pour la PLS à seulement 10 variables retenues. En effet, là où on retrouve les mêmes phénotypes (glucose et urée) sur les autres PLS, ils sont totalement différents sur la PLS à 10 variables. On voit le même phénomène pour les gènes (aucun en commun avec les autres PLS) alors que on retrouve tous les gènes de la PLS à 15 variables sur celle à 20 variables. On peut cependant noter que les gènes 11572 et 12535 de la PLS à 10 variables sont retrouvés de l'autre côté de l'axe pour les autres analyses.

Axe 2 positif

```
p2.pos.axe2
```

```
## [[1]]
## 51778 11572 12535 27556 36602  8612 31634 24932
##     5    19    22    23    24    26    27    28
##
## [[2]]
## hematocrite    hemoglob
##          12          13
```

```
p3.pos.axe2
```

```
## [[1]]
##  6703  8941 51211  7035 12322 36524  6220 46673
##     1     2     7    12    13    15    17    20
##
## [[2]]
## pgranulo
##        8
```

```
p4.pos.axe2
```

```
## [[1]]
## 58383 51778 11572 12535 27556 36602  8612 31634 24932
##     3     7    26    29    30    31    33    35    36
##
## [[2]]
## hematocrite    hemoglob
##          12          13
```

Enfin à nouveau ici tous les phénotypes sont les mêmes pour les PLS à 15 et à 20 variables. On retrouve les gènes : 11572 ; 12535 ; 27556 ; 36602 ; 8612 ; 31634 ; 24932 sur le côté positif de l'axe 2 lors des PLS à 15 et 20 variables. On peut noter que certains gènes de la PLS à 10 variables sont aussi présents côté négatif de ce même axe pour les autres PLS : 6220 ; 46673 ; 8941 ; 51211 et l'on observe de même pour le phénotype hemoglob.

De manière générale, l'axe 1 semble plus stable, même si l'analyse avec seulement 10 variables reste différente, ce qui est normal puisque c'est le premier axe à être estimé. L'axe 2 est donc très différent pour l'analyse à 10 variables même si l'on retrouve quelques gènes et phénotypes communs aux autres PLS. Notons que le signe de la corrélation par rapport à l'axe importe peu puisque les axes de la PLS sont définis à un signe près.

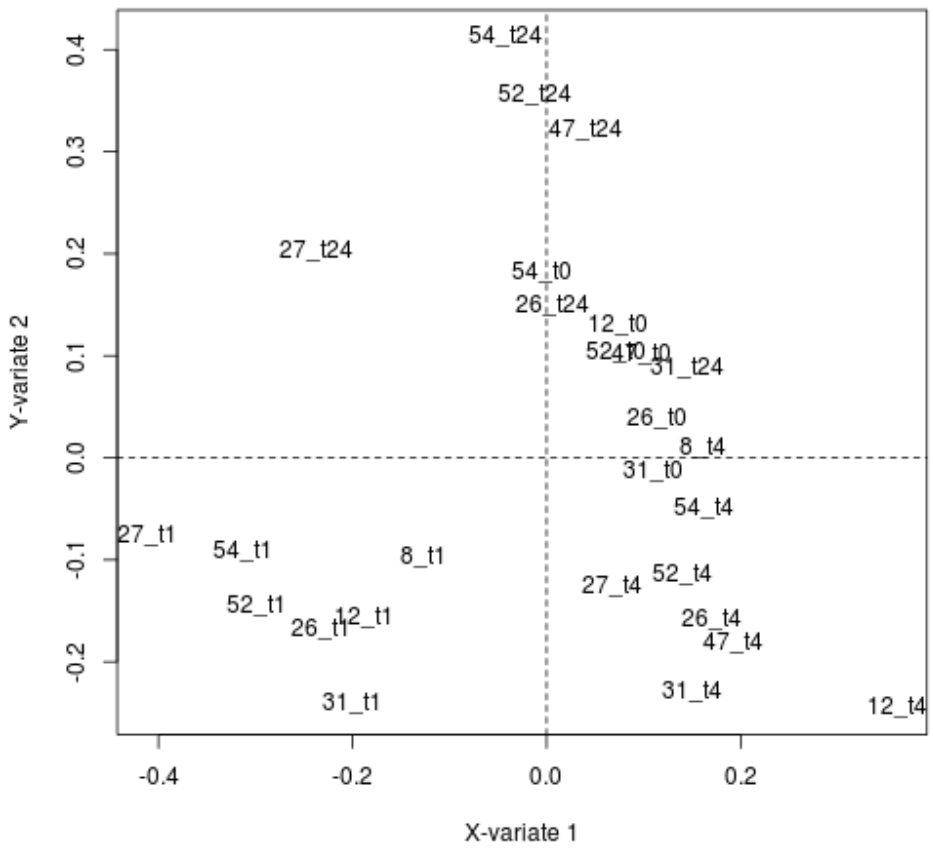
Placement des individus

On peut noter plusieurs similitudes sur les PLS comme illustrés dans les nuages ré-exécutés ci-dessous :

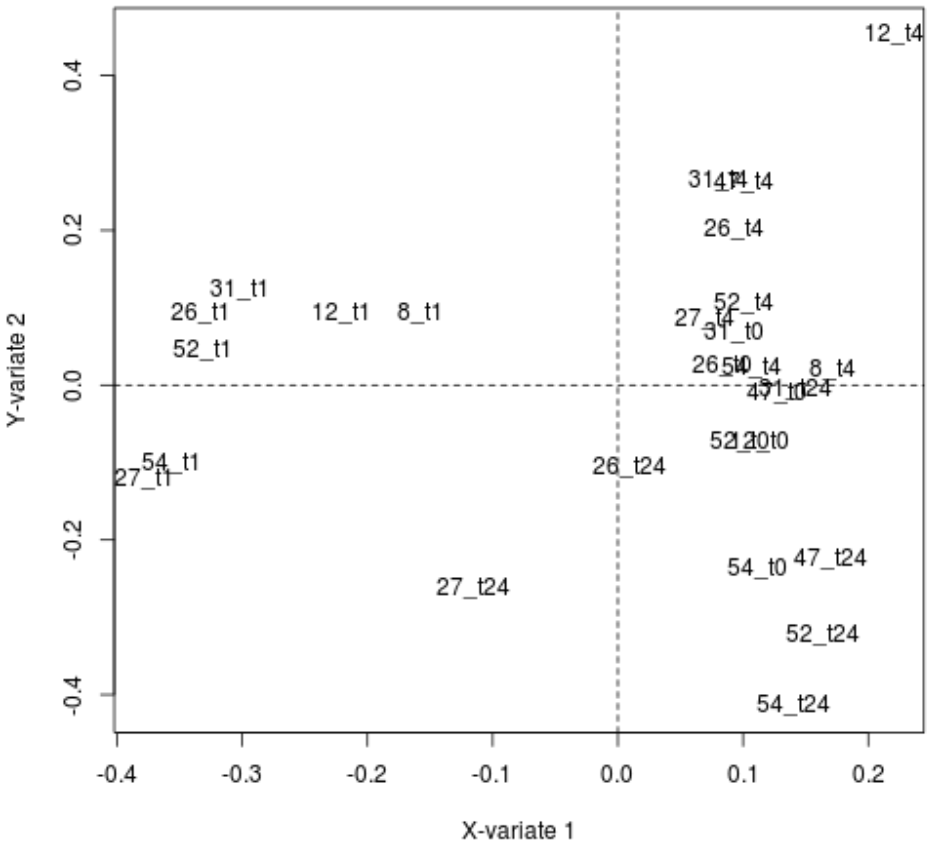
Au niveau de l'axe 1 qui est très ressemblant sur toutes les PLS, on retrouve à chaque fois des individus au temps t1 lorsque l'on cherche à qui appartiennent les valeurs faibles de cet axe. Les temps t0 et t24 sont souvent en position assez neutre sur cet axe, plutôt coté positif, tandis que le temps t4 a des valeurs plus élevées (surtout sur les PLS à 15/20 variables retenues).

Le second axe oppose les individus des temps t1 et t4 aux individus t0 et t24. (les valeurs sont opposées sur la PLS à 10 variables car l'axe est inversé).

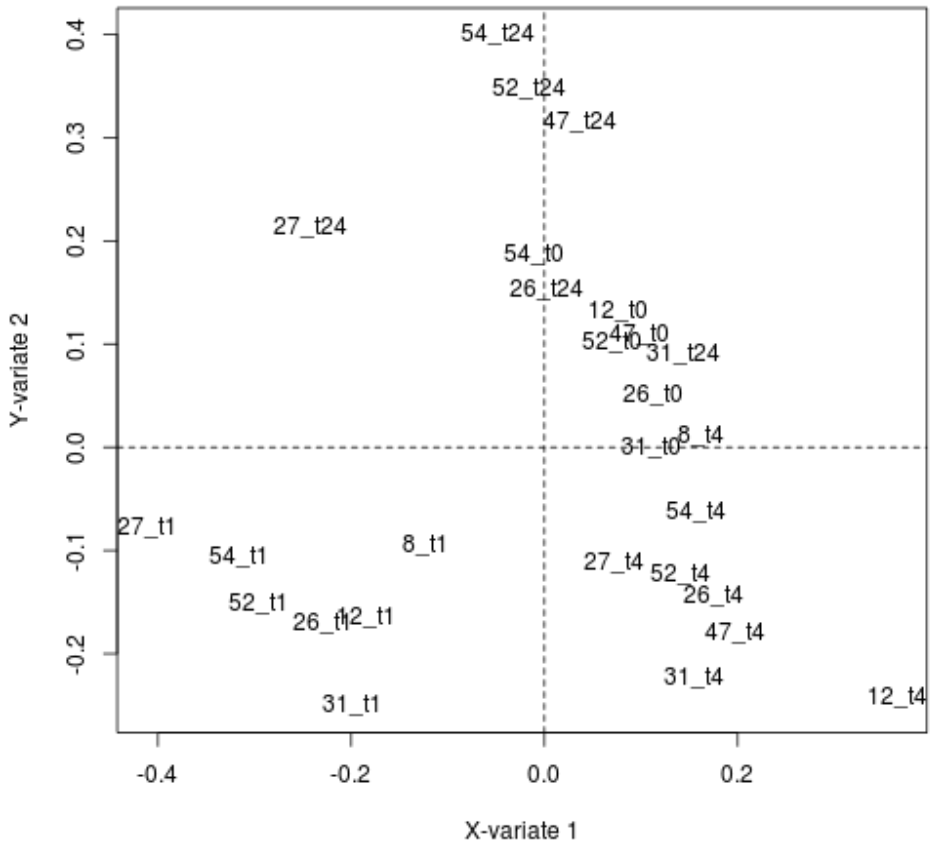
```
plotIndiv(pls2, comp = 1:2, ind.names = names.indiv, rep.space =
"XY-variate",
    x.label = NULL, y.label = NULL, col = "black", cex = 1, pch =
1)
```




```
plotIndiv(pls3, comp = 1:2, ind.names = names.indiv, rep.space =
"XY-variate",
    x.label = NULL, y.label = NULL, col = "black", cex = 1, pch =
1)
```




```
plotIndiv(pls4, comp = 1:2, ind.names = names.indiv, rep.space =
"XY-variate",
    x.label = NULL, y.label = NULL, col = "black", cex = 1, pch =
1)
```



Une remarque récurrente à quasiment toutes les analyses est que les individus ayant le plus de valeurs imputées ont une position atypique par rapport aux autres et qu'il convient donc d'interpréter les résultats les concernant avec prudence.

Rappel des valeurs manquantes sur les phénotypes :

```
rownames(no.allmissing.pheno) <- names.indiv
apply(no.allmissing.pheno, 1, function(x) sum(is.na(x)))
```

```
##   8_t1   8_t4  12_t0  12_t1  12_t4  26_t0  26_t1  26_t4 26_t24
27_t1
##      0      0      0      0      0      0      0      0
5      0
##  27_t4 27_t24  31_t0  31_t1  31_t4 31_t24  47_t0  47_t4 47_t24
52_t0
##      0      5      0      0      0      5      0      0
5      0
##  52_t1  52_t4 52_t24  54_t0  54_t1  54_t4 54_t24
##      0      0      5      0      0      0      5
```