Projet Statistiques

Jonas Issam Lucas Cassandra

2023-01-07

Contents

1 Introduction

On observe, pour G = 1615 gènes d'une plante modèle, les valeurs suivantes :

$$Y_{gtsr} = log_2(X_{gtsr} + 1) - log_2(X_{gt_0} + 1)$$

où :
- X_{gtsr} est la mesure du gène g pour le traitement t pour le réplicat r et au temps s.
- X_{ggt_0} est l'expression du gène g pour un traitement de référence.

On dit qu'un gène g est sur-exprimé si $Y_{gtsr} > 1$, sous-exprimé si $Y_{gtsr} < 1$ et non-exprimé sinon.
Le jeu de données comprend 36 variables quantitatives.

2 Analyse du jeu de données

Cette partie a été réalisée en Python dans la première partie du Jupyter notebook. ## Réduction des données :

Les données sont exprimées dans des ordres de grandeur différents. On normalise les données.

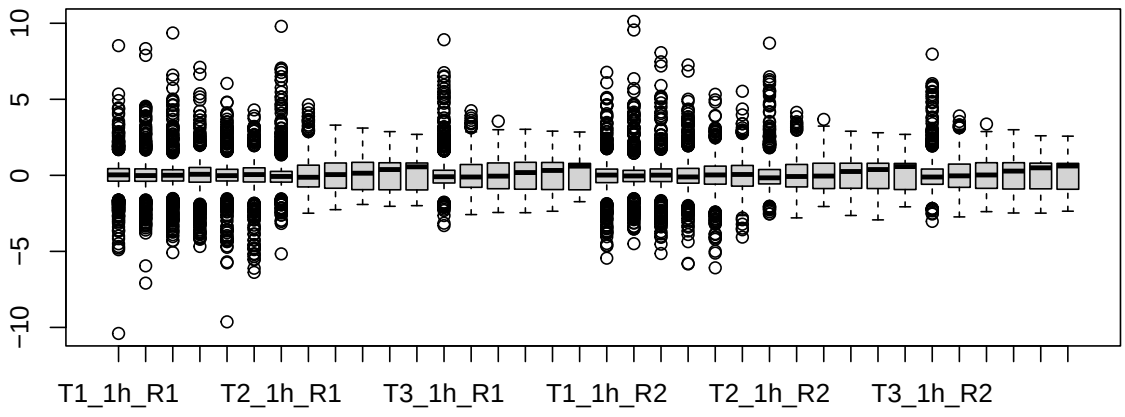


Figure 1: Boxplot des traitements

Malgré la transformation, on observe de nombreux outliers d'après la figure 1. En effet, dans la plupart des cas, les gènes ne sont pas exprimés, ainsi les valeurs sont concentrées autour de 0. Le boxplot est donc peu intéressant.

2.1 Analyse bi-dimensionnelle

D'après la figure 2, on observe que les réplicats 1 et 2 suivent les mêmes tendances (non/sous/sur-exprimé). Cependant, lorsque l'on calcule les coefficients de corrélation pour chaque traitement à chaque temps entre

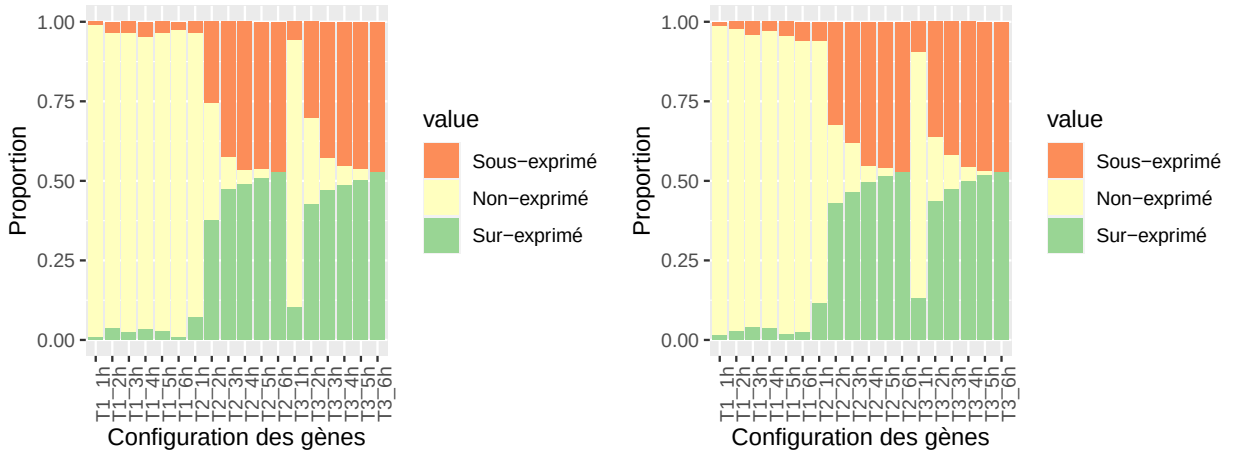


Figure 2: Etat des gènes selon le traitement et le temps pour les réplicats 1 et 2

les réplicats 1 et 2, on constate que les valeurs prises pour le traitement 1 possèdent une forte variabilité suivant le réplicat. Au contraire, pour les traitement 2 et 3, les valeurs obtenues sont très semblables.

```
                T1_1h     T1_2h     T1_3h     T1_4h     T1_5h     T1_6h
Cor(R1,R2) 0.3779514 0.5007921 0.7110046 0.4023233 0.4142479 0.4813625

                T2_1h     T2_2h     T2_3h     T2_4h     T2_5h     T2_6h
Cor(R1,R2) 0.8917355 0.9621318 0.9830982 0.9654878 0.9669259 0.9873128

                T3_1h     T3_2h     T3_3h     T3_4h     T3_5h     T3_6h
Cor(R1,R2) 0.871809 0.9703837 0.9857304 0.9826932 0.9720265 0.9864855
```

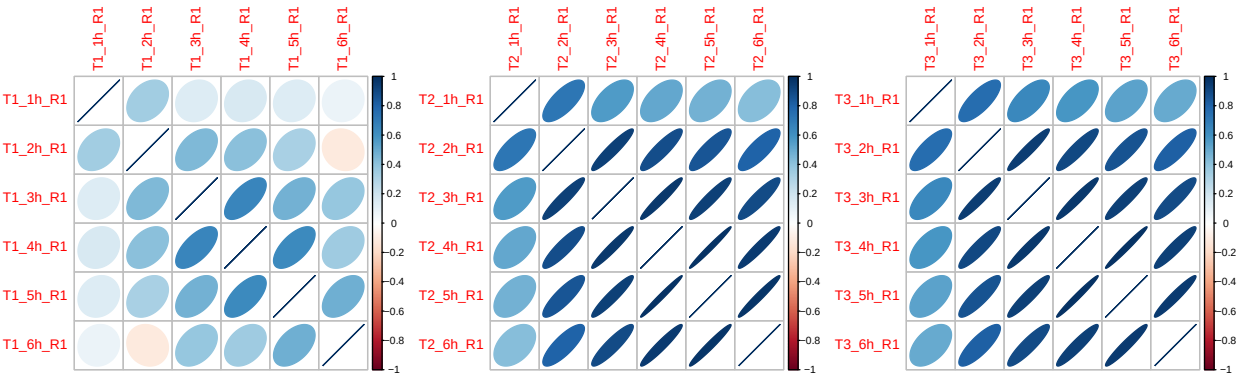


Figure 3: Cercles des corrélations au sein de chaque traitement

D'après figure 3, au sein des traitements 2 et 3, on observe une forte corrélation entre deux mesures successives (h et h+1).

2.2 Analyse en composantes principales

On fait une ACP afin de condenser l'information de notre jeu de données au travers de méta-variables.

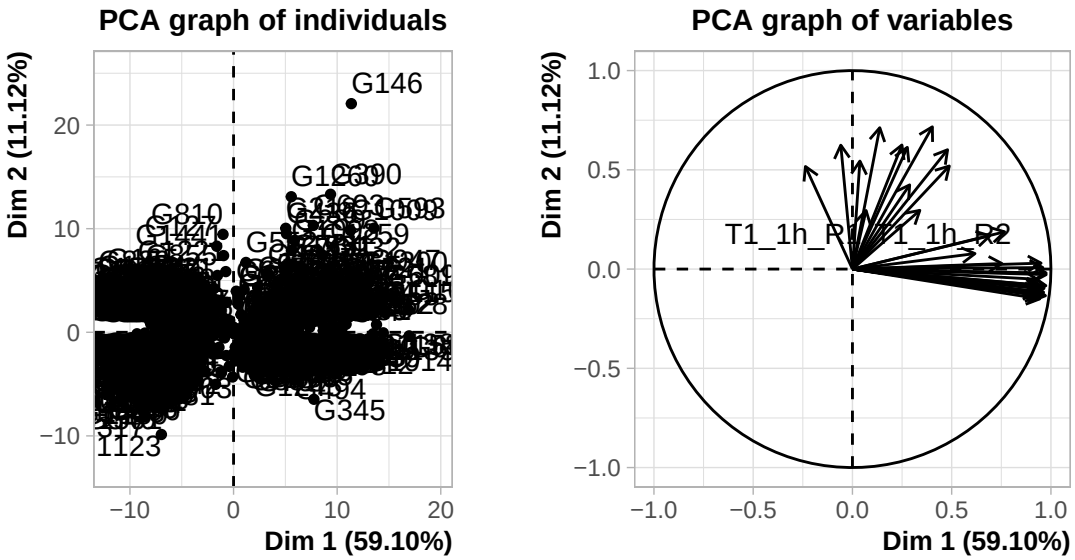


Figure 4: ACP des variables et des individus

D'après la figure 4, on constate deux clusters. L'analyse en cluster sera plus longuement détaillée par la suite.

Choix du nombres de composantes :

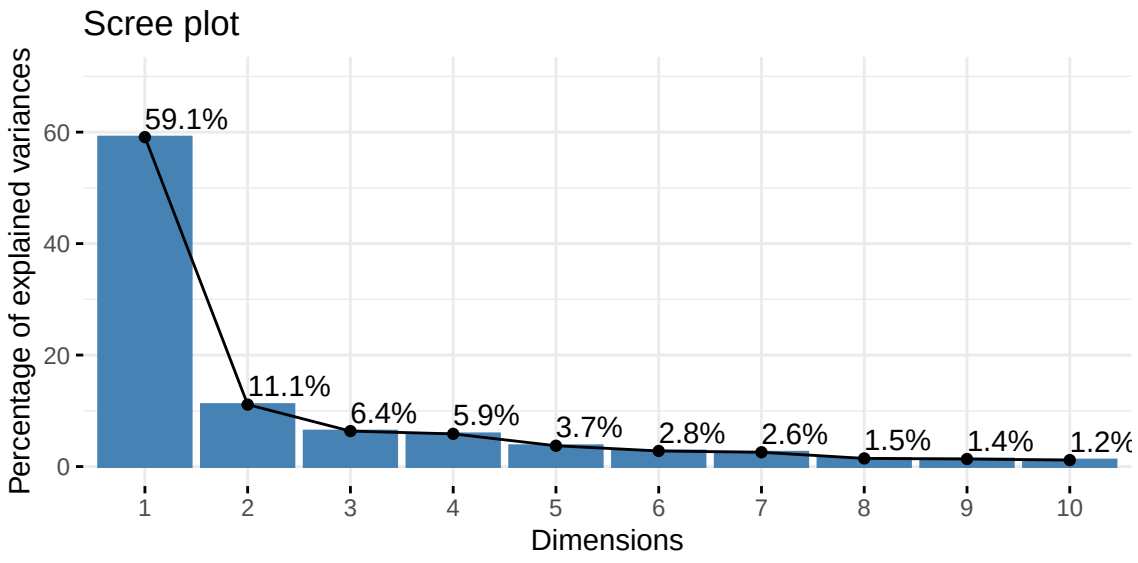


Figure 5: Pourcentage de variance expliquée en fonction de la dimension

On applique la méthode du coude à la figure 5 afin de déterminer le nombre de composantes que l'on garde. On conserve uniquement les deux premières composantes soit 70.2 % de la variance expliquée.
Cependant, par la suite, en expliquant les axes principaux, on constate que le 3ème semble porter une information pertinente. Ainsi, nous choisissons de garder les 3 premiers axes, soit 76.6% de la variance.
On exprime les trois premières dimensions en fonction des 36 variables.

D'après la figure 6, on remarque que les traitements T2 et T3 expliquent la composante 1 et ainsi le cluster de droite. En effet, les coefficients associés aux traitements T2 et T3 tendent vers 1 tandis que les coefficients

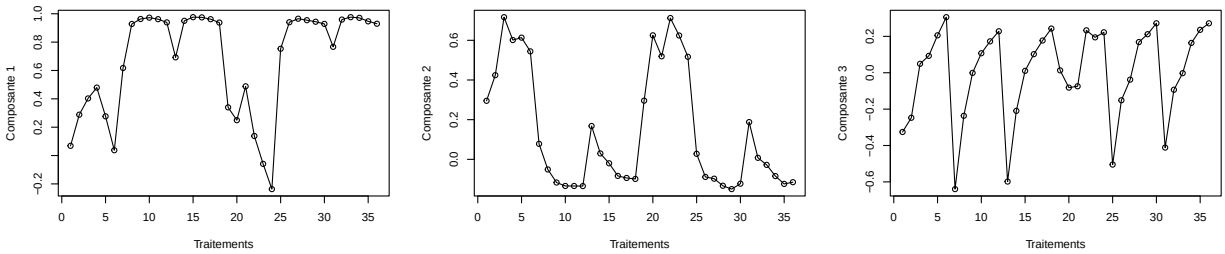


Figure 6: Coordonnées des composantes

associés au traitement T1 sont proches de 0. De la même manière, le traitement T1 explique la composante 2. Le graphique 3 est périodique de période 6.

3 Régression linéaire multiple

Cette partie n'a pas été réalisée en Python car aucun package nous permet de faire de la sélection de variables.

On réalise plusieurs sélections de variables avec différentes méthodes (forward et backward) pour lesquelles on applique différents critères (BIC, Cp de Mallows et AIC).

Les sélections de variables en backward/forward selon les critères BIC/Cp/AIC conduisent toutes au modèle initial. On en déduit donc que, pour le traitement 3 à 6h, toutes les heures du traitement 3 pour le réplicat 2 jouent un rôle significatif sur l'expression des gènes T3_6h_R1 .

3.1 T3_6h_R2 en fonction de T3_ih_R1 et T3_ih_R2

On considère maintenant le réplicat 1.

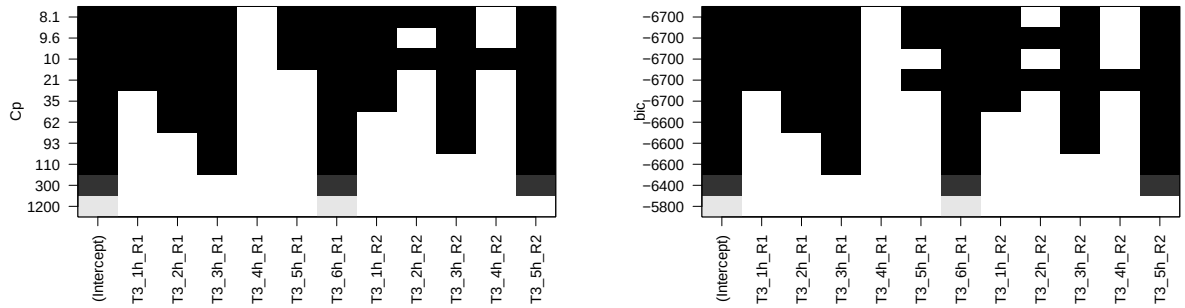


Figure 7: Séléction de variables : méthodes descendantes

D'après les figures 7 et 8, on obtient les deux sous-modèles suivant :
(M1) : $T3\_6h\_R2_i = \theta_0 + \theta_1 T3\_1h\_R1_i + \theta_2 T3\_2h\_R1_i + \theta_3 T3\_3h\_R1_i + \theta_4 T3\_5h\_R1_i + \theta_5 T3\_6h\_R2_i + \theta_6 T3\_1h\_R2_i + \theta_7 T3\_2h\_R2_i + \theta_8 T3\_3h\_R2_i + \theta_9 T3\_5h\_R2_i + \theta_{10} T3\_5h\_R2_i + \epsilon_i$

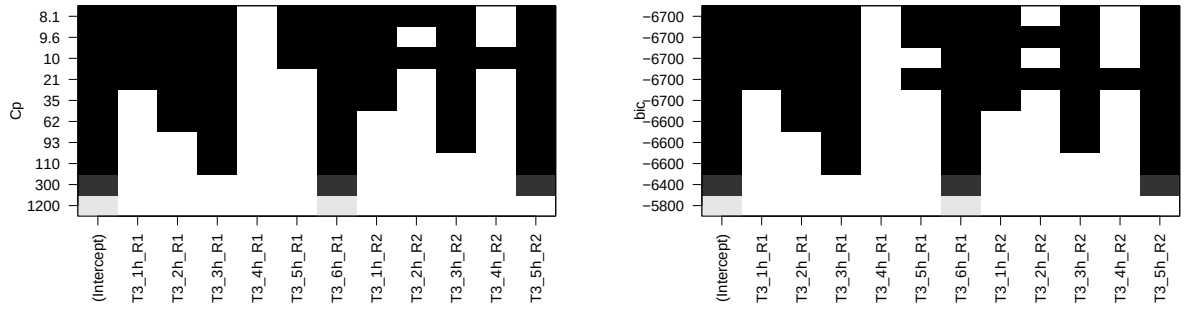


Figure 8: Séléction de variables : méthodes ascendantes

(M2) : $T3\_6h\_R2_i = \theta_0 + \theta_1 T3\_1h\_R1_i + \theta_2 T3\_2h\_R1_i + \theta_3 T3\_3h\_R1_i + \theta_4 T3\_5h\_R1_i + \theta_5 T3\_6h\_R2_i + \theta_6 T3\_1h\_R2_i + \theta_7 T3\_3h\_R2_i + \theta_8 T3\_5h\_R2_i + \theta_9 T3\_5h\_R2_i + \epsilon_i$
On réalise deux tests de Fisher afin de valider ou non les sous-modèles précédemment établis.

Le premier test renvoie une p-valeur égale à $0.93 > 0.05$. On accepte le sous-modèle (M1) au risque 5%.

Le second test renvoie un p-valeur égale à $0.31 > 0.05$. On accepte le sous-modèle (M2) au risque 5%.

3.2 T3_6h_R2 en fonction de Ti_jh_R2

A présent, nous étudions l'impact de l'expression des gènes sur le traitement T1 à 6h pour tous les traitements et pour tous les temps. On ne considère que le réplicat 2 pour cette partie.

3.2.1 Critère : BIC et Cp de Mallows

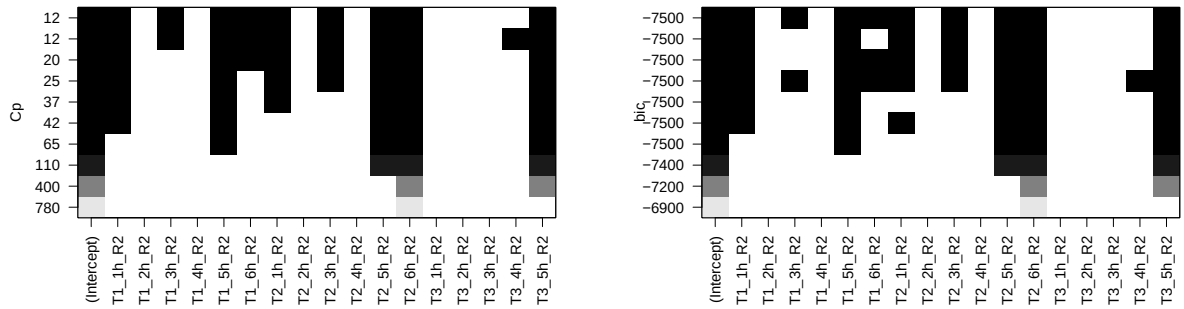


Figure 9: Séléction de variables : méthodes descendantes

On note (M1) le sous-modèle obtenue d'après la 9 avec les critères Cp et Bic en backward avec le critère Bic en forward.

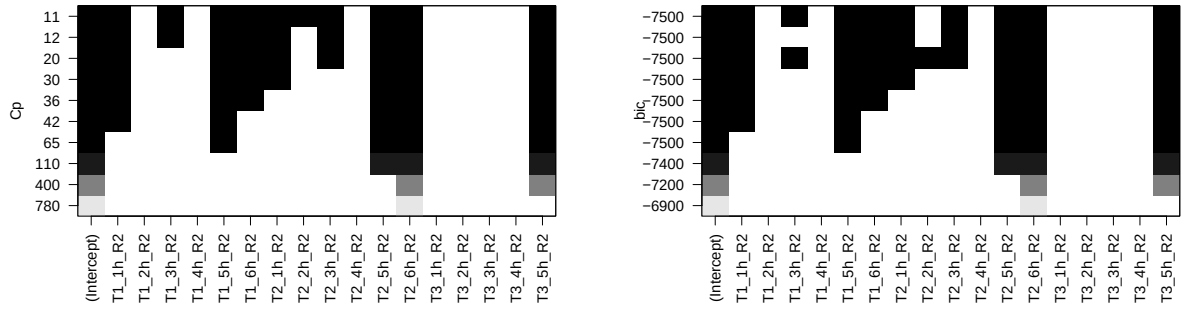


Figure 10: Séléction de variables : méthodes ascendantes

(M1) : $T3\_6h\_R2_i = \theta_0 + \theta_1 T1\_1h\_R2_i + \theta_2 T1\_3h\_R2_i + \theta_3 T1\_5h\_R2_i + \theta_4 T1\_6h\_R2_i + \theta_5 T2\_1h\_R2_i + \theta_6 T2\_3h\_R2_i + \theta_7 T2\_5h\_R2_i + \theta_8 T2\_6h\_R2_i + \theta_9 T3\_5h\_R2_i + \epsilon_i$

On note (M2) le sous-modèle obtenue d'après la 10 avec le critère Cp en forward.

(M2) : $T3\_6h\_R2_i = \theta_0 + \theta_1 T1\_1h\_R2_i + \theta_2 T1\_3h\_R2_i + \theta_3 T1\_5h\_R2_i + \theta_4 T1\_6h\_R2_i + \theta_5 T2\_1h\_R2_i + \theta_6 T2\_2h\_R2_i + \theta_7 T2\_3h\_R2_i + \theta_8 T2\_5h\_R2_i + \theta_9 T2\_6h\_R2_i + \theta_{10} T3\_5h\_R2_i + \epsilon_i$

On réalise deux tests de Fisher afin de valider ou non les sous-modèles précédemment obtenus :

Les deux tests de Fisher renvoient une pvaleur (respectivement 0.2798 et 0.4205) supérieure à 0.05. On ne rejette pas les sous-modèles au risque 5%.
Choisissons maintenant, le meilleur des deux sous-modèles. Le modèle (M1) étant inclu dans le modèle (M2), on peut réaliser de nouveau un test de sous-modèle de Fisher.

Le test de Fisher renvoie une pvaleur égale à $0.09932 > 0.05$. On conserve donc le modèle (M1)

3.2.2 Critère AIC

Réalisons maintenant la sélection de variables avec la méthode backward et le critère AIC (la méthode forward ne fonctionne pas avec le critère AIC).

La sélection de variables en backward avec le critère AIC renvoie le sous-modèle suivant :

(M3) : $T3\_6h\_R2_i = \theta_0 + \theta_1 T1\_1h\_R2_i + \theta_2 T1\_3h\_R2_i + \theta_3 T1\_5h\_R2_i + \theta_4 T1\_6h\_R2_i + \theta_5 T2\_1h\_R2_i + \theta_6 T2\_3h\_R2_i + \theta_7 T2\_5h\_R2_i + \theta_8 T2\_6h\_R2_i + \theta_9 T3\_3h\_R2_i + \theta_1 0 T3\_4h\_R2_i + \theta_1 1 T3\_5h\_R2_i + \epsilon_i$

On réalise un test de Fisher afin de valider ou non le sous-modèle précédemment obtenu :

Le test renvoie un pvaleur égale à $0.4887 > 0.05$. On valide le sous-modèle.
On ne peut pas faire un test de sous-modèle car ils ne sont pas sous-modèles l'un de l'autre. Les R2 ajustés des deux modèles étant égaux on compare la valeur du critère AIC.

3.3 T1_6h_R2 en fonction de T1_ih_R2

On réalise plusieurs sélections de variables avec différentes méthodes (forward et backward) pour lesquelles on applique différents critères (BIC, Cp de Mallows et AIC).

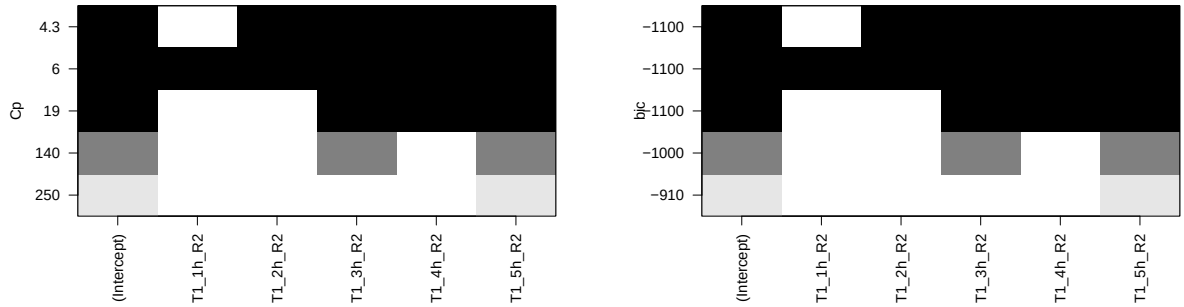


Figure 11: Séléction de variables : méthodes descendantes

D'après la figure 11, on obtient le sous-modèle suivant :
(M1) : $T1\_6h\_R2_i = \theta_0 + \theta_1 T1\_2h\_R2_i + \theta_2 T1\_3h\_R2_i + \theta_3 T1\_4h\_R2_i + \theta_4 T1\_5h\_R2_i$

La sélection de variables avec la méthode ascendante revoie le même sous-modèle.

On applique le critère AIC pour les sélections de variables en forward et backward :

La méthode backward avec le critère AIC propose le même sous-modèle. Vérifions si ce sous modèle est acceptable. On fait un test de sous-modèle de Fisher.

La pvaleur est largement supérieure à 0.05. On accepte le sous-modèle (M1) au risque 5%.
La méthode forward avec le critère AIC conserve toutes les variables.
Finalement, les temps affectant l'expression des gènes à 6h pour le traitement T1 fixé sont les heures 2, 3, 4 et 5.

3.4 T1_6h_R2 en fonction de T1_ih_R1 etT1_ih_R2

On considère maintenant le réplicat 1.

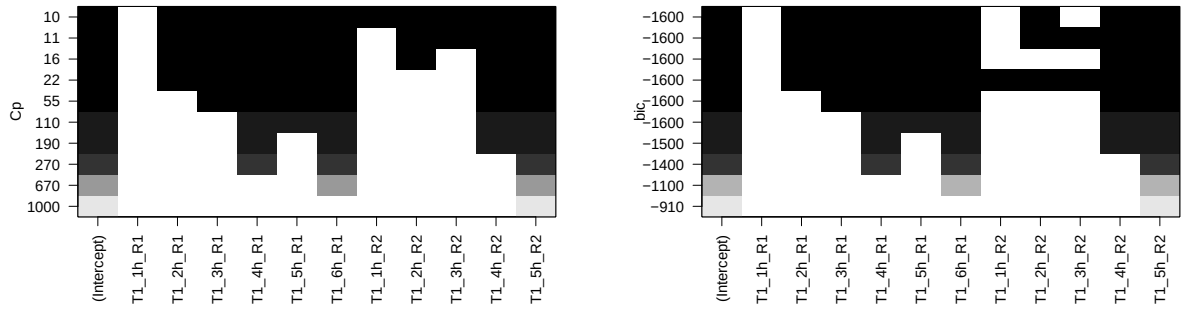


Figure 12: Séléction de variables : méthodes descendantes

D'après les figures 12 et 13, on obtient les deux sous-modèles suivant :
(M1) : $T1\_6h\_R2_i = \theta_0 + \theta_1 T1\_2h\_R1_i + \theta_2 T1\_3h\_R1_i + \theta_3 T1\_4h\_R1_i + \theta_4 T1\_5h\_R1_i +$

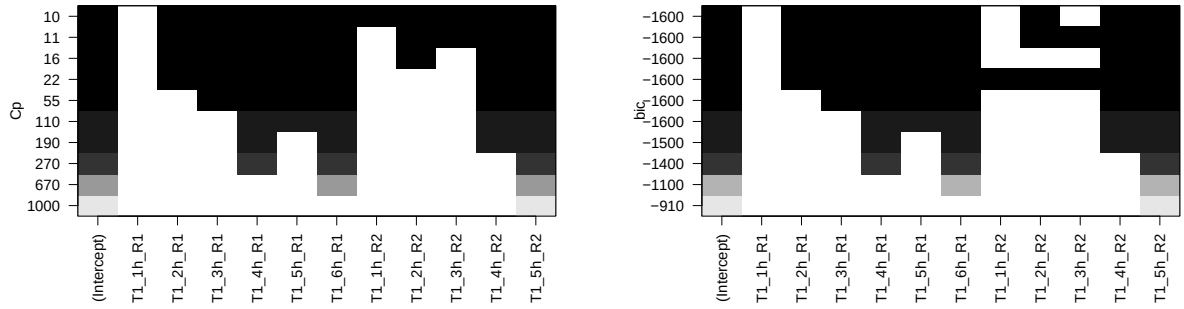


Figure 13: Séléction de variables : méthodes ascendantes

$\theta_5 T1\_6h\_R2_i + \theta_6 T1\_6h\_R1_i + \theta_7 T1\_1h\_R2_i + \theta_8 T1\_2h\_R2_i + \theta_9 T1\_3h\_R2_i + \theta_{10} T1\_4h\_R2_i + \theta_{11} T1\_5h\_R2_i + \epsilon_i$

(M2) : $T1\_6h\_R2_i = \theta_0 + \theta_1 T1\_2h\_R1_i + \theta_2 T1\_3h\_R1_i + \theta_3 T1\_4h\_R1_i + \theta_4 T1\_5h\_R1_i + \theta_5 T1\_6h\_R2_i + \theta_6 T1\_6h\_R1_i + \theta_7 T1\_2h\_R2_i + \theta_8 T1\_4h\_R2_i + \theta_9 T1\_5h\_R2_i + \epsilon_i$
On réalise deux tests de Fisher afin de valider ou non les sous-modèles précédemment établis.

Le test renvoie une p-valeur égale à $0.498 > 0.05$. On accepte le sous-modèle (M1) au risque 5%.

Le test renvoie une p-valeur égale à $0.21 < 0.05$. On rejette le sous-modèle (M2) au risque 5%.

3.5 T1_6h_R2 en fonction de Ti_jh_R2

Régression linéaire multiple : T3_6h_R2 en fonction de Ti_jh_R2.

3.5.1 Critère : BIC et Cp de Mallows

On fait une sélection de variables avec les méthodes forward/backward et les critères BIC/Cp de Mallows.

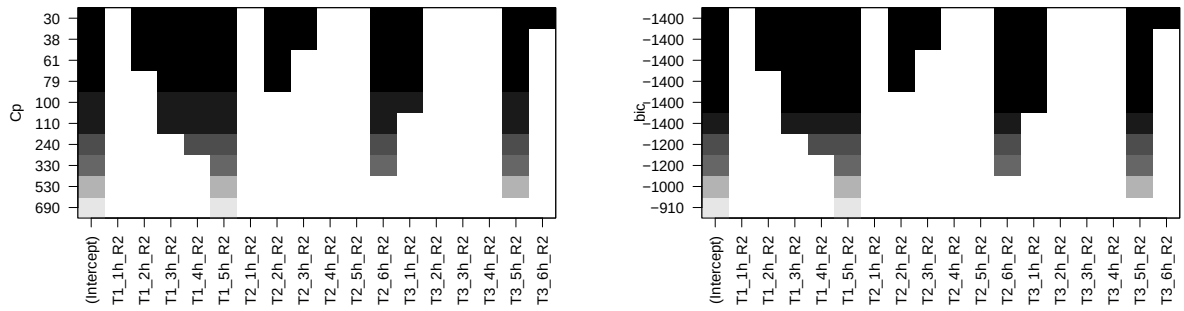


Figure 14: Séléction de variables : méthodes descendantes

Les deux algorithmes renvoient le même sous-modèle (figure 14).
On réalise un test de Fisher de sous-modèle :

Le test renvoie une pvaleur très inférieure à 0.05. On rejette le sous-modèle au risque 5%.

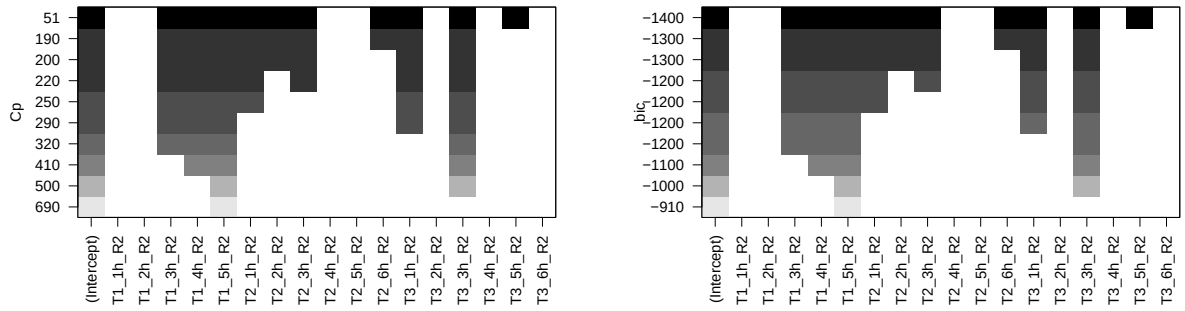


Figure 15: Séléction de variables : méthodes ascendantes

Les deux algorithmes renvoient le même sous-modèle (figure 15).
On réalise un test de Fisher de sous-modèle :

Le test renvoie une pvaleur très inférieure à 0.05. On rejette le sous-modle au risque 5%.

3.5.2 Critère AIC

Réalisons maintenant la sélection de variables avec la méthode backward avec le critère AIC.

On obtient le sous-modèle suivant : (M2) : $T1\_6h\_R2_i = \theta_0 + \theta_1 T1\_1h\_R2_i + \theta_2 T1\_2h\_R2_i + \theta_3 T1\_3h\_R2_i + \theta_4 T1\_4h\_R2_i + \theta_5 T1\_5h\_R2_i + \theta_6 T2\_1h\_R2_i + \theta_7 T2\_2h\_R2_i + \theta_8 T2\_3h\_R2_i + \theta_9 T2\_4h\_R2_i + \theta_{10} T2\_5h\_R2_i + \theta_{11} T2\_6h\_R2_i + \theta_{12} T3\_1h\_R2_i + \theta_{13} T3\_3h\_R2_i + \theta_{14} T3\_5h\_R2_i + \theta_{15} T3\_6h\_R2_i + \epsilon_i$

La pvaleur est supérieure à 0.05 donc on peut simplifier le modèle en enlevant le traitement 3 à 2h et 4h pour le réplicat 2 au risque 5%.

4 Modéle linéaire généralisé

4.1 Gène T3_6h_R2 :

Notre objectif est de déterminer les variables prédictives permettant de discriminer les gènes sur-exprimés des gènes sous-exprimés à 6h pour le traitement T3 (réplicat 2). Dans le but de mettre en place une régression logistique, nous faisons le choix de modifier les valeurs prises par la variable T3_6H_R2 afin d'obtenir une réponse binaire (i représentant le i-ème gène). On supprime les gènes non exprimés de notre dataframe. On a alors :

$$T3\_6h\_R2_i = \begin{cases} 1 & \text{si } T3\_6h\_R2_i > 1 \\ 0 & \text{si } T3\_6h\_R2_i < -1 \end{cases}$$

Nous étudions à présent le modèle linéaire généralisé suivant :

$$\begin{cases} T3\_6h\_R2_i & \sim \mathcal{B}(\pi_\rangle) \\ T3\_6h\_R2_i & indépendants \\ logit(\pi_i) & = ln(\frac{\pi_i}{1-\pi_i}) \\ & = \theta_0 + \theta_1 T3\_1h\_R2_i + \theta_2 T3\_2h\_R2_i + \theta_3 T3\_3h\_R2_i + \theta_4 T3\_4h\_R2_i + \theta_5 T3\_5h\_R2_i \end{cases}$$

A la vue des p-valeurs, ils semblent possible de simplifier le modèle en retirant certaines variables au risque 5%. Pour se faire, on met en place un algorithme de sélection de variables basé sur le critère BIC puis AIC (forward et backward) et donc sur une minimisation de la divergence de Kullback-Leibler.

Critère BIC :

```
Call:  glm(formula = y ~ ., family = family, data = Xi, weights = weights)

Coefficients:
(Intercept)     T3_2h_R2     T3_5h_R2
      1.063       -1.541        8.320

Degrees of Freedom: 1614 Total (i.e. Null);  1612 Residual
Null Deviance:      2234
Residual Deviance: 8.446    AIC: 14.45
```

Critère AIC, méthode descendante :

```
Call:  glm(formula = T3_6h_R2 ~ T3_2h_R2 + T3_5h_R2, family = binomial(link = "logit"),
    data = Data_T3)

Coefficients:
(Intercept)     T3_2h_R2     T3_5h_R2
      1.063       -1.541        8.320

Degrees of Freedom: 1614 Total (i.e. Null);  1612 Residual
Null Deviance:      2234
Residual Deviance: 8.446    AIC: 14.45
```

Les deux critères de sélection conduisent au même sous-modèle :

$$logit(\pi_i) = \theta_0 + \theta_1 T3\_2h\_R2_i + \theta_2 T3\_5h\_R2_i$$

Afin de valider la procédure de sélection de variables, on réalise un test de sous-modèle de Fisher. On vérifie en amont que le modèle obtenu après la sélection de variables est bien un sous-modèle du modèle initial.

Le test renvoie une p-valeur de 0.8555 : on ne rejette donc pas notre sous-modèle au risque 5%. Nous conservons alors le modèle obtenu après la sélection de variables.
Nous souhaitons à présent savoir quelle variable entre T3_2h_R2 et T3_5h_R2 a la plus grande influence sur le fait que le gène $T3\_6h\_R2_i$ soit sur-exprimé ou sous-exprimé. Pour cela on s'intéresse à la notion d'odds ratio. On définit deux individus x et $\tilde{x}$ qui différent uniquement sur la variable T3_5h_R2 de la façon suivante: $T3\_5h\_R2(\tilde{x}) = T3\_5h\_R2(x) + 1$. On a :

$$\begin{aligned} OR(\tilde{x}, x) &= \frac{odds(\tilde{x})}{odds(x)} \\ &= \frac{e^{\theta_0 + \theta_1 T3\_2h\_R2(\tilde{x}) + \theta_2 T3\_5h\_R2(\tilde{x})}}{e^{\theta_0 + \theta_1 T3\_2h\_R2(x) + \theta_2 T3\_5h\_R2(x)}} \\ &= e^{\theta_2 (T3\_2h\_R2(\tilde{x}) - T3\_5h\_R2(x))} \\ &= e^{\theta_2} \end{aligned}$$

```
 (Intercept)     T3_2h_R2     T3_5h_R2
   2.8961359    0.2141409 4104.1863597
```

On remarque donc que : $e^{\hat{\theta_2}^{Obs}} \approx 4104$.
Cela signifie, que lorsque l'on augmente d'une unité la mesure d'expression d'un gène pour le traitement 3 à 5 heure pour le réplicat 2, on multiplie le rapport de chance de sur-expression par 4104. Nous en déduisons donc, pour le traitement 3, que le phénomène de sur-expression du gène se fait dans les derniers instants du traitement.

Nous nous demandons à présent, quelle serait la réponse prédite pour un nouveau gène et ainsi si notre sous-modèle est bien ajusté.
Pour cela, nous rappelons que la probabilité pour un nouveau gène d'être sur-exprimé s'écrit de la façon suivante :

$$\hat{\pi_0} = \frac{e^{\theta_0+\theta_1 T3\_2h\_R2(x)+\theta_2 T3\_5h\_R2(x)}}{1+e^{\theta_0+\theta_1 T3\_2h\_R2(x)+\theta_2 T3\_5h\_R2(x)}}$$

La réponse prédite d'un nouveau gène s'écrit : $\mathbb{1}_{\hat{\pi_i}>0.5}$. Pour cela, nous étudions la table de contingence suivante :

```
   FALSE TRUE
0    762    2
1      0  851
```

Nous observons que seulement 2 individus ont été mal classés par notre sous-modèle sur l'ensemble des gènes, ce qui est très satisfaisant.

4.2 Gène T1_6h_R2 :

On cherche maintenant à déterminer les variables prédictives permettant de discriminer les gènes sur-exprimés des gènes sous-exprimés des gènes non-exprimés à 6h pour le traitement T1 (réplicat 2). Précédemment, nous nous sommes placés dans le cadre d'une régression logistique car la variable à expliquer ne prennait que deux valeurs. Ici, nous pouvons distinguer 3 plages de valeurs pour la variable T1_6h_R2.

Dans un premier temps, nous décidons de nous placer dans le cadre d'un modèle de régression polytomique ordonnée dont la dernière modalité est prise comme référence : le fait que le gène soit sur-exprimé. Pour rentrer dans ce cadre là, nous modifions les valeurs prises par la variable T1_6h_R2 de la façon suivante:

$$T1\_6h\_R2_i = \begin{cases} -1 & \text{si } T3\_6h\_R2_i < -1 \\ 0 & \text{si } T3\_6h\_R2_i \in [-1,1] \\ 1 & \text{si } T3\_6h\_R2_i > 1 \end{cases}$$

Le modèle s'écrit sous la forme :

$$ln(\frac{\pi_m(x_i)}{\pi_1(x_i)}) = x_i\theta^{(m)}$$

π_m est la probabilité de prendre la mobilité m pour les trois niveaux d'expression des génes. x_i est le vecteurs des valeurs observées pour le ième individu sur les variables explicatives $\theta^{(m)} \in \mathbb{R}^{12}$
Comme les modalités d'expression des génes ont un ordre naturel, il est préférable de considérer un modèle reliant les π_m pour deux modalités successives.

Le modéle ordonné s'écrit sous la forme : $ln[\frac{\pi_{m+1}(x_i)}{\pi_m(x_i)}]$
Ainsi, pour deux individus x et $\tilde{x}$ qui diffèrent d'une unité pour la variable j, on a :

$OR(Y = u_{m+1} \text{ vs } Y = u_m; x, \tilde{x}) = \exp\left[\theta_j^{(m)}\right]$
On regarde les valeurs des coefficients les plus élevés dans le summary du modèle ordonné.
Pour la variable T1_4h_R2, on a :
$OR(Y = 1 \text{ vs } Y = 0; x, \tilde{x}) = \exp\left[\theta_8^{(0)^{obs}}\right] = \exp(2.48761) = 12.03$. Ainsi, le rapport de chance que le gène T1_4h_R2 soit sur-exprimé que non-exprimé est multiplié par 12.03.
De même, on a pour la variable T1_5h_R2 :
$OR(Y = 0 \text{ vs } Y = -1; x, \tilde{x}) = \exp\left[\theta_9^{(-1)^{obs}}\right] = \exp(2.76819) = 15.92$. Ainsi, le rapport de chance que le gène T1_5h_R2 soit non-exprimé que sous-exprimé est multiplié par 15.92.

5 Clustering des gènes

Dans cette partie nous allons traiter de la partie sur le clustering des gènes. La même chose à été réalisé en Python dans la partie Clustering des données (gènes). Nous avons utilisé plusieurs méthodes pour réaliser cela (Kmeans, DBSCAN, classification hiérarchique, modèle de mélanges gaussiens). Pour l' algorithme des Kmeans nous avons utilisé des variantes de l'algorithme classique et pour la classification hiérarchique nous avons testé différentes mesures d'agrégation pour la distance euclidienne. Nous ne présentons pas toutes les méthodes utilisées mais vous pouvez vous référer au code pour les consulter.
Nous avons raisonné de la même manière pour tous les algorithmes. Dans un premier temps nous avons choisit le nombre de classes optimal avec différents critères (Inertie intra-classe, silhouette, Calinski-Harabasz). Nous sélectionnons le nombre de classe optimal puis nous réalisons la classification. Nous affichons la table de contingence puis nous calculons le ARI pour comparer nos méthodes.

5.1 Kmeans

Nous allons dans un premier temps nous intéresser à l'algorithme des Kmeans. Nous avons utilisé l'algorithme classique, l'algorithme des centres mobiles, des nuées dynamiques, PAM et fuzzy c-means.
Nous choisissons dans un premier temps le nombre de classes avec les critères d'inertie intra-classe et silhouette.

```
  cluster size ave.sil.width
1       1  773          0.48
2       2  842          0.47
```

Les deux critères nous permettent de choisir $K = 2$ classes. Avec le graphique obtenu (figure 16) avec le critère silhouette nous voyons que certains points sont assez éloignés de leur centre de gravité.

Nous réalisons maintenant un algorithme Kmeans (MacQueen) avec 2 classes.

Nous avons observé que tous les algorithmes donnaient le même résultat sauf l'algorithme PAM (même si nous avons presque la même classification).

Pour finir on regarde le nombre d'individus obtenu par cluster.

```
table(reskmeans$cluster)
```

```
  1   2
773 842
```

On remarque que les clusters ne sont pas disproportionnés, on obtient à peu près le même nombre d'individus dans les deux groupes.

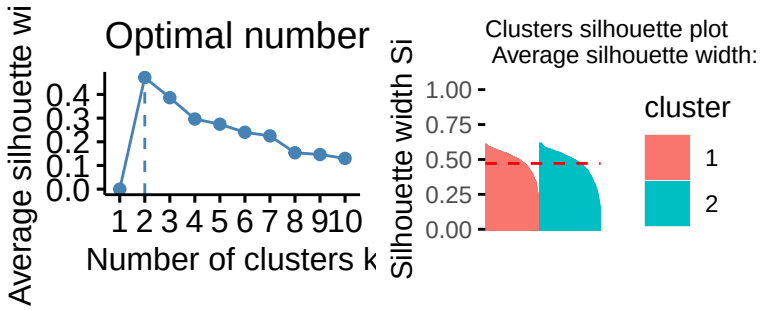


Figure 16: Critère Silhouette pour l'algorithme Kmeans

5.2 Méthodes hiérarchique

Nous allons maintenant voir dans un second temps le clustering des gènes avec les méthodes hérarchiques.

Nous avons réalisé une classification hiérarchique avec la distance euclidienne et les mesures d'agrégation single, complete, average et Ward:

Pour la classification hiérarchique le choix du nombre de classes peut aussi se faire avec les critères Inertie intra classe et Calinski-Halbraz. On réalise ces critères pour les 4 mesures d'agrégation précédemment utilisées.

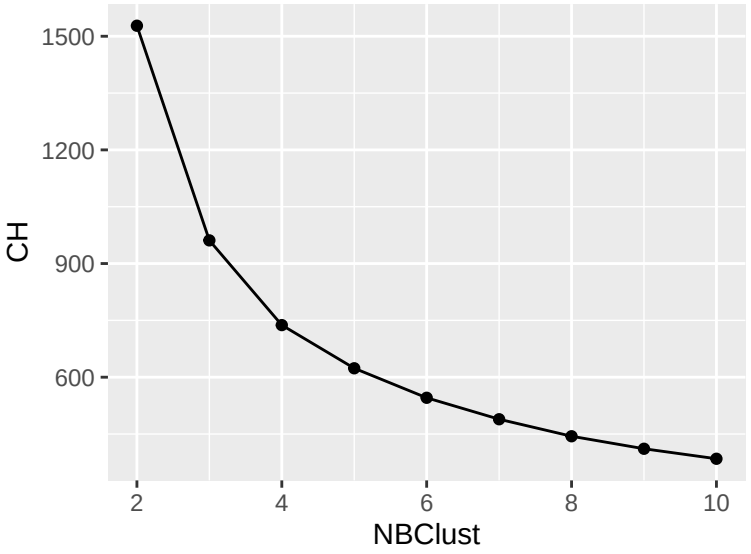


Figure 17: Inertie intra-classe pour la mesure d'agrégation Ward

Pour la mesure d'agrégation ward nous obtenons 2 classes pour chacun des critères.

```
table(Class_ward_2_classes)
```

```
Class_ward_2_classes
  1   2
856 759
```

Nous avons remarqué que les critères CH et Silhouette ne donnaient pas le même nombre de classe entre les mesures d'agrégation single, average et complete. De plus, en observant la table de contingence nous voyons que pour ces 3 mesures d'agrégations le nombre d'individus par classe est disproportionné. Par contre pour la mesure de Ward nous obtenons une classification en 2 classes et le nombre d'individus par classe est assez similaire.

```
adjustedRandIndex(Class_ward_2_classes, reskmeans$cluster)
```

```
[1] 0.9558874
```

Nous avons voulu confirmer le fait que la mesure d'agrégation de ward donne une meilleure classification. Pour cela, nous avons calculé le ARI entre chacune des mesures d'agrégation et aussi avec l'algorithme des Kmeans. Nous remarquons que la classification obtenue avec la mesure de Ward et l'algorithme des Kmeans est similaire. Par contre, pour les autres mesures nous avons une classification totalement différente.

5.3 DBSCAN

Nous allons maintenant utiliser un autre algorithme appelé DBSCAN. En Python, nous réalisons la classification sur les données après ACP en prenant les deux premières composantes, et nous obtenons la même classification.

On trace pour plusieurs valeurs de minPts et de epsilon le nombre de clusters obtenus en utilisant l'algorithme DBSCAN.

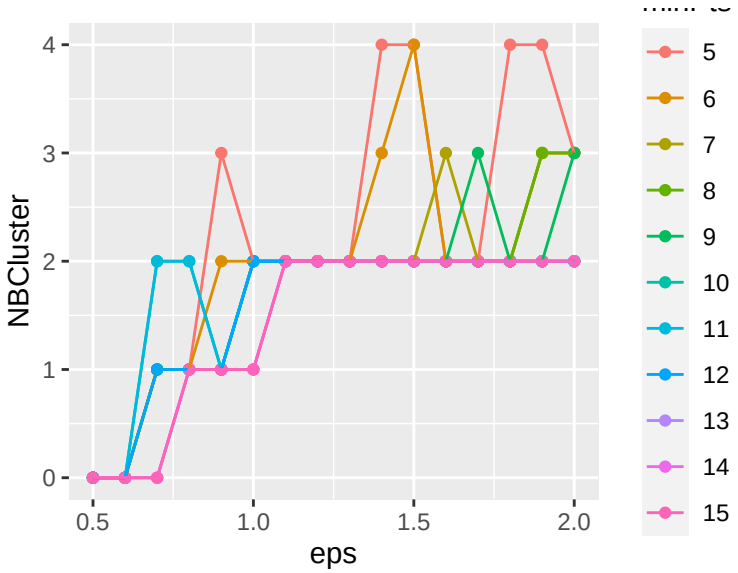


Figure 18: Epsilon en fonction du nombre de clusters

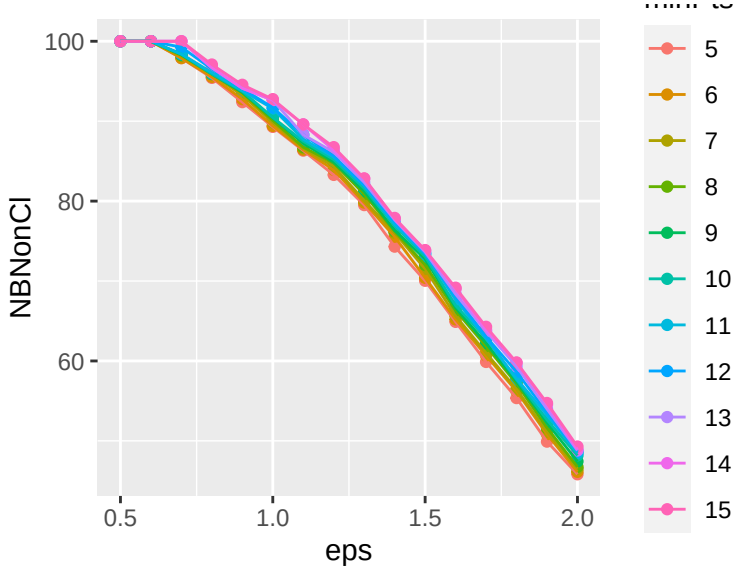


Figure 19: Epsilon en fonction du nombre de clusters

Avec ce graphique on voit que le nombre de cluster reste a peu près le même pour plusieurs valeurs de epsilon et minPts.

Nous réalisons maintenant l'algorithme DBSCAN avec MinPts=5 et eps=4. D'après le cours il faudrait prendre minPts tel que $K = ln(p)$ ou $K = p - 1$ avec p le nombre de colonnes. Nous prenons alors $minPts = 8$. Nous obtenons $eps = 5$ en traçant les 7-NN voisins (figure 20). En faisant varier le paramètre $minPts$ nous voyons que la classfification reste la même si $5 < minPts < 15$. Cependant, si on regarde la table de contingence on voit que les individus sont répartis dans un cluster et seulement 73 ne sont pas classés.

```
res.db <- dbscan::dbscan(Data_cr, eps = 5, minPts = 15)
table(res.db$cluster)
```

```
   0    1
  73 1542
```

Comparons nos résultats avec les 2 autres types d'algorithme utilisés avant (Kmeans et classification hiérarchique).

Avec la classification hiérarchique :

```
adjustedRandIndex(Class_average_2_classes, res.db$cluster)
```

```
[1] 0.04868968
```

Avec l'algorithme des Kmeans :

```
adjustedRandIndex(reskmeans$cluster, res.db$cluster)
```

```
[1] -0.001600826
```

Les classifications DBSCAN Kmeans et classification hiérarchique sont très différentes car le ARI est proche de 0.

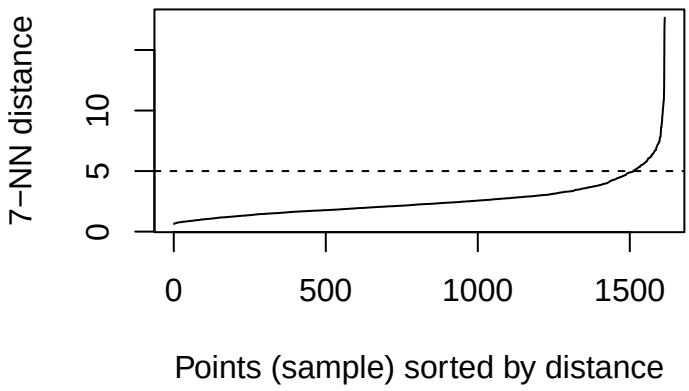
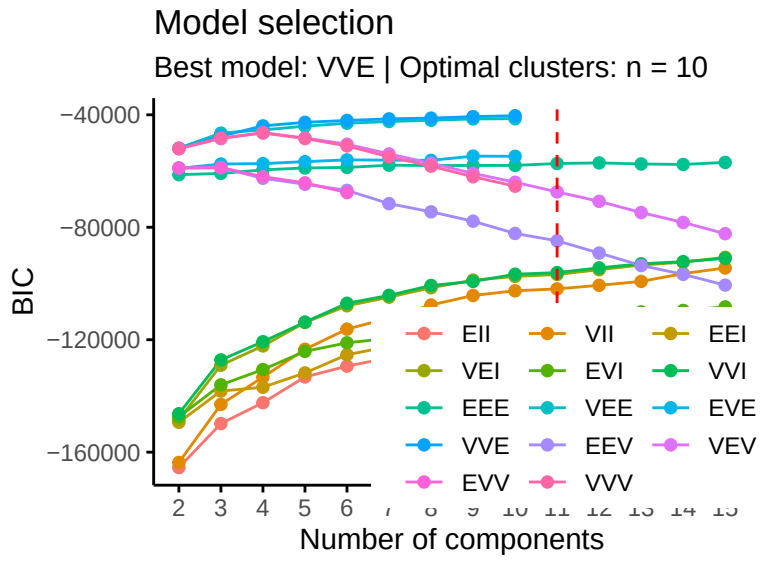


Figure 20: 7 plus proches voisins

5.4 Modèle de mélanges gaussiens

Nous allons maintenant voir dans un dernier temps l'algortihme de mélange gaussien. Ici, nous ne faisons pas l'algorithme sur les données centrées réduites contrairement aux autres algorithmes. En Python, pour cette partie nous avons en plus investigué la fusion des classes.

Les critères BIC et ICL nous permettent de choisir le nombre de cluster et la collection de modèle adaptée. Ils sélectionnent tous les deux la forme VVE avec 10 clusters.




```
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------
```

```
Mclust VVE (ellipsoidal, equal orientation) model with 10 components:

 log-likelihood    n   df       BIC       ICL
       -15142.6 1615 1359 -40324.26 -40475.51

Clustering table:
  1   2   3   4   5   6   7   8   9  10
224 304 230 211  97 130 121 133 102  63

  1   2   3   4   5   6   7   8   9  10
224 304 230 211  97 130 121 133 102  63
```

Analysons les probabilités a priori obtenues (figure 21) :

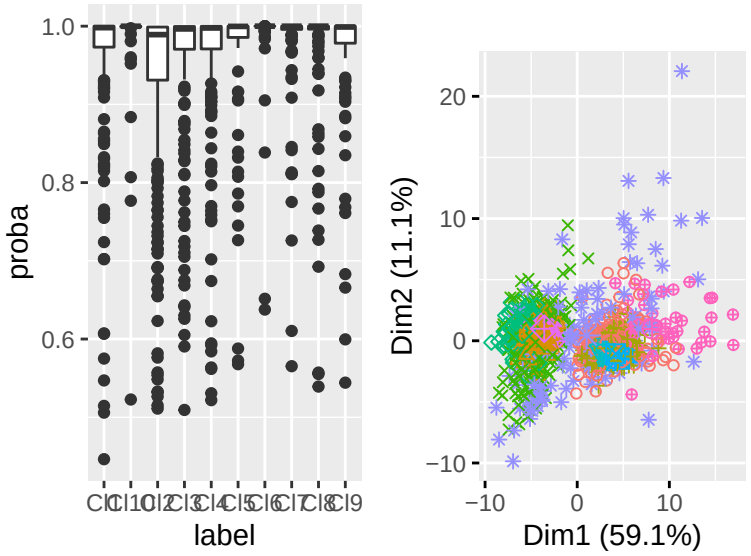


Figure 21: Probabilité à priori

Nous voyons que les probabilités a priori sont assez mauvaises. En effet, nous observons beaucoup d'outliers par classe ce qui se traduit par une mauvaise classification de ces points. En regardant, le graphique de droite de la figure 21 nous voyons que beaucoup de clusters sont mélangés et peuvent donc expliquer ces mauvaises probabilités à priori.

```
adjustedRandIndex(Modele_melange$classification, reskmeans$cluster)
```

```
[1] 0.2237161
```

En comparant le ARI avec les autres algorithmes utilisés nous voyons que la classification obtenue est aussi assez éloignée de celle obtenue avec les algorithmes Kmeans et classification hiérarchique avec mesure d'agrégation de Ward.

En conclusion, nous avons vu que les algorithmes Kmeans et classification hiérarchique avec mesure de Ward permettent d'obtenir la meilleure classification en 2 clusters.

5.5 Interpréation des clusters obtenus

Nous n'avons pas réussi à reproduire cette partie en R mais vous pouvez vous référer au Jupyter notebook sur la partie Interpretation des clusters (Kmeans). Nous remarquons que la moyenne d'expression des gènes du cluster 1 est quasiment toujours positive, et correspond à l'opposée de celle du cluster 2. On a donc des gènes sur-exprimés dans le cluster 1, et sous-exprimés dans cluster 2.

5.6 LDA

Dans cette partie nous allons, à l'aide d'une LDA, interpréter les 2 clusters obtenus en fonction des traitements.

La LDA permet de montrer que nos deux clusters sont bien séparés par le premier axe discriminant. Cependant, ils ne sont pas totalement séparés car il y a quelques individus du premier cluster qui viennent se mélanger aux individus du second.

Maitenant, nous séparons nos données en train et test. Ensuite, on réalise une LDA sur le dataset train avec les labels obtenus par l'algorithme des kmeans. Finalement, on prédit les classes d'appartenance du dataset test.

Cette prédiction permet dans un deuxième temps de calculer la correlation entre les variables Ti_jh_rk et l'axe discriminant pour déterminer quelles variables expliquent nos clusters.

Tous les poids supérieurs au poids moyen sont les variables qui expliquent l'axe discriminant. On en déduit que cet axe discriminant est expliqué par les variables T1_1h_R1, T2_2h_R1, T2_4h_R1, T2_5h_R1, T2_6h_R1, T3_1h_R1, T3_3h_R1, T3_4h_R1, T3_5h_R1, T3_6h_R1 puis exactement les mêmes variables pour le deuxième réplicat. Il semblerait donc que l'on sépare des gènes qui sont plutôt caractérisés par les traitements 2 à 2, 4, 5 et 6 heures puis 1, 4 4 et 5 heures pour le traitement 3 et 1 heure pour le traitement 1.

6 Clustering sur les traitements

Dans cette partie nous allons traiter du clustering sur les traitements. Cette partie à aussi été traitée en Python dans la partie Clustering des variables. Pour ce faire nous transposons la matrice centrée réduite de départ. La matrice possédant trop de colonnes, nous décidons de réaliser une ACP pour réduire les dimensions et faciliter le clustering. Nous décidons dans cette partie de n'utiliser que trois algorithmes (kmeans, classification hiérarchique et mélange gaussien).

Nous raisonnons de la même manière pour tout les algorithmes utilisés. Nous calculons d'abord le nombre de classes adéquat à chaque méthode. Dans beaucoup de cas nous obtenons des nombres de classes différents entre les critères, nous affichons donc la table de contingence des classifications obtenues avec les différents algorithmes pour sélectionner la meilleure classification et contrôler le nombre de classes. Nous réalisons ensuite le clustering avec les nombres de classes trouvés. Enfin nous calculons le ARI pour comparer les différents algorithmes. Nous décidons de ne pas afficher toutes les sorties mais vous retrouverez toutes ces étapes dans le code pour chacune des méthodes.

6.1 Kmeans

Nous réalisons dans un premier temps la sélection du nombre de classes par les critères silhouette et inertie intra-classe pour l'aglorithme Kmeans.

```
  cluster size ave.sil.width
1       1    7          0.18
```

```
2        2   24          0.63
3        3    5          0.58
```

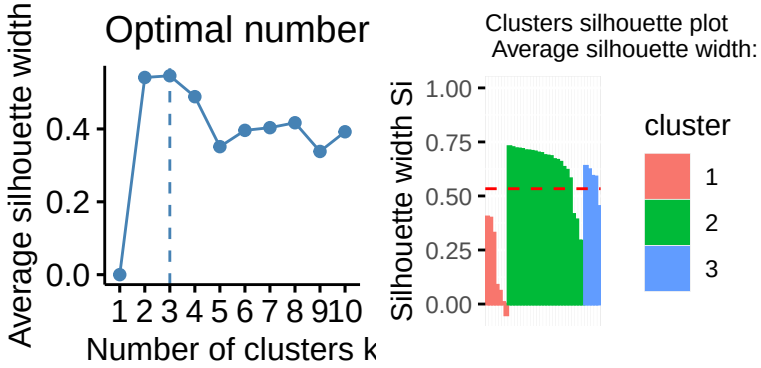


Figure 22: Sélection du nombre de classes avec le critère silhouette

On maximise le critère silhouette pour $K = 3$ classes. On remarque cependant sur la figure 22 que le premier cluster comporte des individus qui sont plutôt éloignés du centre de gravité de la classe.

Le résultat obtenu avec l'algorithme Kmeans et $K = 3$ classes est visible à la figure 23. Nous observons 3 clusters assez bien séparés.

6.2 Classification hiérarchique

Réalisons maintenant le clustering à l'aide de la classification hiérarchique. Nous faisons toutes les mesures d'agrégation avec la distance euclidienne. Nous avons dans un premier temps choisi le nombre de classes avec les critères Calinski-Halbraz et l'inertie intra-classe. Ensuite, comme les nombres de classes étaient différents, nous avons coupé l'arbre pour les deux valeurs de K obtenues et nous avons sélectionné la méthode dont le ARI était le meilleure en comparant avec l'algorithme Kmeans et en affichant la table de contingence.

D'après le critère du coude nous sélectionnons $K = 3$ classes comme le montre la figure 24.

```
adjustedRandIndex(Class_single_3, reskmeans_variables_1$cluster)
```

```
[1] 0.6837222
```

Les classifications single et Kmeans semblent similaires car le ARI est proche de 1. C'est avec cette mesure d'agrégation que nous obtenons le plus grand ARI en comparant avec l'algorithme des Kmeans.

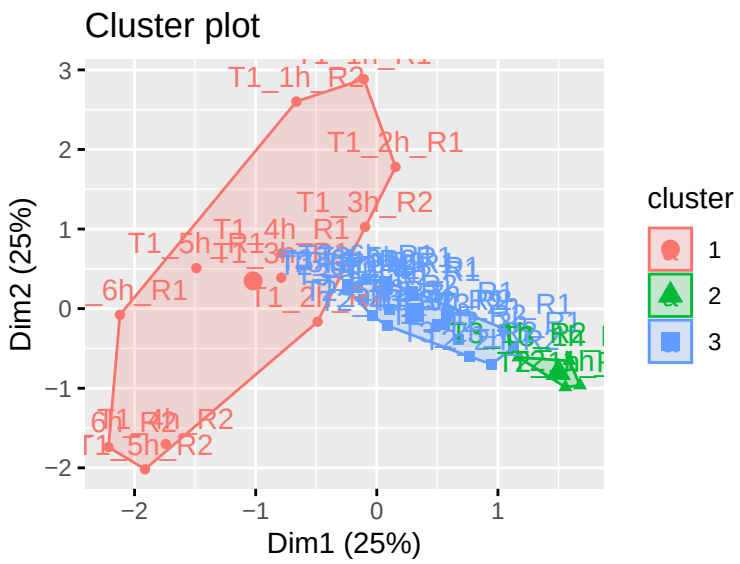


Figure 23: Classification à l'aide de l'algorithme Kmeans

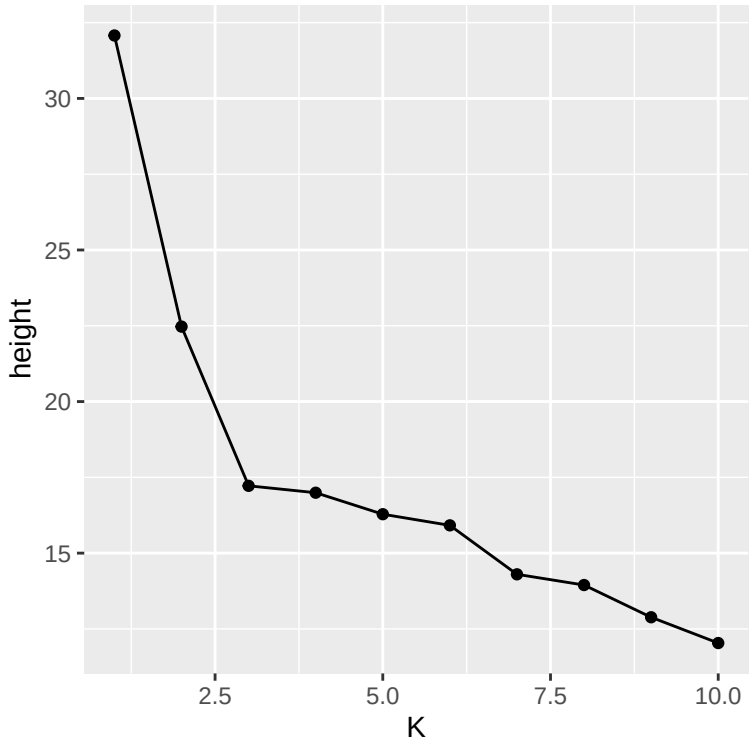


Figure 24: Inertie intra-classe pour la mesure d'agrégation single

6.3 Modèle de mélanges gaussiens

Dans une dernière partie nous allons utiliser le modèle de mélange gaussien pour réaliser notre classification.

Nous sélectionnons le nombre de cluster avec les critères BIC et ICL. Les deux critères donnent exactement le même résultat. Nous choisissons la forme VEV avec 4 clusters comme le montre la figure 25.

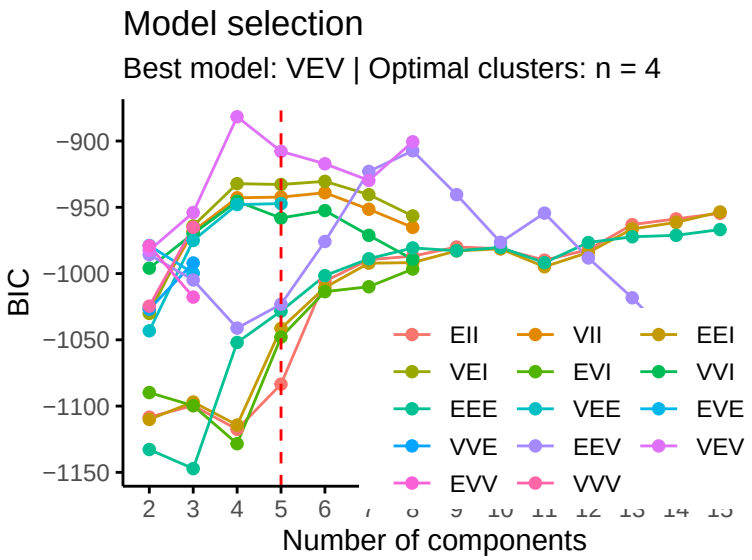


Figure 25: Sélection de modèle avec les mélanges gaussiens

```
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust VEV (ellipsoidal, equal shape) model with 4 components:

 log-likelihood  n df       BIC       ICL
      -351.2501 36 50 -881.6761 -881.7172

Clustering table:
 1  2  3  4
12  4  8 12
```

Nous observons à peu près les mêmes clusters qu'avec l'algorithme des Kmeans. Cependant, le cluster qui se situe tout à droite dans la figure 23 semble s'être divisé en deux pour le modèle de mélange gaussien comme on le voit dans la figure **??**.

Observons maintenant les probabilités associées de la figure 26.

On remarque qu'on obtient deux outliers pour le deuxième cluster. Les traitements semblent très bien affectés à chacun de leur groupe puisque toutes les probabilités sont entre 1 et 0.985.

Comparons les résultats avec les autres algorithmes.

Il semblerait que les classifications kmeans et modèle de mélange gaussien soit assez similaires car le ARI est proche de 1. Nous l'avions vu en affichant les clusters.

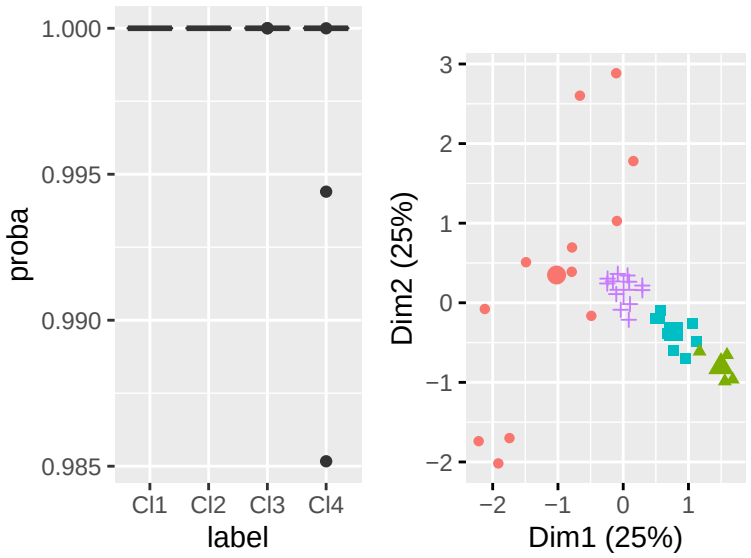


Figure 26: Probabilité à priori

```
adjustedRandIndex(Modele_melange_traitements$classification, reskmeans_variables_1$cluster)
```

```
[1] 0.6688858
```

Cependant, la classification ne semble pas être la même avec la classification hiérarchique.

```
adjustedRandIndex(Modele_melange_traitements$classification, Class_single_3)
```

```
[1] 0.3842502
```

6.4 Interprétation des clusters

Les schémas précédemment affichés avec les algorithmes de mélanges gaussiens ainsi que la classification hiérarchique permettent de diviser le dataset en 3 ou 4 clusters. Nous interprétons les résultats obtenus avec les algorithmes Kmeans et modèle de mélange car ils donnent à peu près la même classification contrairement à la classification hiérarchique. Pour l'algorithme des kmeans nous obtenons l'interprétation suivante : Le premier cluster correspond au traitement 1 à toutes les heures et tous les réplicats. Le second cluster correspond aux traitemetns 2 et 3 entre 2 et 6 heures pour les deux réplicats. Finalement le dernier cluster correspond aux traitements 2 et 3 à 1 heure pour les deux réplicats.

Pour l'algorithme des mélanges gaussiens nous avons exactement les 3 mêmes clusters que précédemment. Nous avons en plus un dernier cluster où nous avons les traitements 2 et 3 entre 4 et 6 heures pour les deux réplicats.

Nous interprétons maintenant les clusters selon la sur ou sous expression des gènes. Veuillez vous référer au code Python pour cette partie car nous n'avons pas réussi à reproduire cela en R (Partie Interpretation clusters de variables). On voit que les gènes du traitements 1, qui forment un cluster de variables, sont en moyenne centrées autour de 0, suivies en module par les gènes du traitement 2 et 3 de la première heure, qui marquent un début de sur-expression / sous-expression, sans pour autant s'éloigner des gènes du traitement 1 en moyenne d'expressions. Ces 4 variables constituent notre second cluster. Enfin, les gènes restantes (Traitements 2 et 3 de 2 à 6h), la sur-expression /sous-expression est clairement distinguable, car en moyenne, les valeurs sont en modules supérieures à 1. C'est notre 3e cluster de variables.

7 Conclusion

En conclusion, grâce aux statistiques descriptives nous avions vu que les traitements 2 et 3 étaient très correlés. Cette hypothèse a été vérifiée en faisant la classification des gènes mais avec une plus grande précision. En effet, grâce aux algorithmes de clustering nous avons vu que les traitements sont regroupés en 3 clusters correspondant respectivement au traitement 1 à toutes les heures et tout les réplicats, un deuxième correspondant aux traitements 2 et 3 entre 2 et 6 heures pour les deux réplicats et enfin un dernier qui regroupe les traitements 2 et 3 à 1 heure pour les deux réplicats. Le clustering sur les gènes nous a permis de voir qu'ils sont caractérisés en deux clusters. Un premier où ils sont sur-exprimés et un deuxième où ils sont sous-exprimés. Enfin, à l'aide de la partie sur le modèle linéaire généralisé, la sur-expression ou sous-expression des gènes est très impactée par les dernières heures du traitements (et particulièrement la 5 ème heure).