Scénario: classification de cinétiques de transcrits

Résumé

L'objectif de cette étude est de proposer une méthode permettant de regrouper les gènes selon leur profil d'expression au cours du temps. La classification ascendante hiérarchique appliquée sur les données brutes ne permet pas d'atteindre l'objectif fixé, car elle fournit une classification basée sur le niveau moyen d'expression des gènes. Le recours à une technique de lissage permet de modéliser la cinétique d'expression de chaque gène par une fonction mathématique, qui une fois dérivée, permet de regrouper les gènes selon leur profil d'évolution.

1 Les données

1.1 Expérimentation

Un total de 44 souris (11 cages * 4 souris/cage) ont été soumises à 11 différentes périodes de jeûne allant de 0 à 72h. Toutes les souris ont été placées dans des cages sans nourriture à 5h (2 heures avant le début de la phase éclairée de la journée). Comme les souris mangent essentiellement durant la nuit, cette expérience correspond à une jeûne post-prandial. À chaque temps sélectionné (0, 3, 6, 9, 12, 18, 24, 36, 48, 60 et 72 heures), 4 souris sont euthanasiées et l'expression de 130 gènes est mesurées par puce à ADN. Les données sont présentées sous la forme d'une matrice à 40 lignes (les données relatives à 4 souris n'étant pas exploitables, elles ont été supprimées du jeu de données ; elles concernent 2 mesures au temps 0, 1 au temps 12h et 1 au temps 24h) et 132 colonnes. Les deux dernières colonnes correspondent au génotype de la souris (colonne 131, ici uniquement WT) et au temps (0, 3, 6, 9, 12, 18, 24, 36, 48, 60 et 72 h).

1.2 Lecture des données

Les donnÃ©es sont dans le fichier `data_fasting_genes.csv` de l'URL `http://wikistat.fr/data/` ; commandes pour l'importation et un premier aperçu de la structure des données :

```
# Lire un fichier .csv
data.fasting = read.csv2("data_fasting_genes.csv",
   row.names=1)
dim(data.fasting)
head(data.fasting)
summary(data.fasting)
# temps de mesure
temps = data.fasting[,131]
```

Calcul de la valeur moyenne de chaque gÃ¨ne Ã chaque temps

```
data.fasting.moy = apply(data.fasting[,1:130],2,
  function(x){tapply(x,temps,mean)})
```

Suivi d'une première visualisation :

```
plot(temps,log(data.fasting[,1]),xlab="Time (h)",
     ylab="log (normalized intensity)",
     ylim=c(-0.5,2.6), col=1, pch="+")
abline(v=temps,lty=3)
for (i in 1:130)
{
  points(data.fasting[,131],data.fasting[,i],
    col=i,pch="+")
  lines(unique(temps),data.fasting.moy[,i],col=i)
}
```

Les données sont représentées par des "+" et la courbe relie la valeur moyenne du gène pour un temps donné.

L'objectif est de proposer une méthode permettant de regrouper les gènes selon leur profil d'expression au cours du temps.

2 Analyse préliminaire

2.1 Analyse en Composantes Principales

```
acp1 = prcomp(data.fasting[,1:130])
biplot(acp1)
plot(acp1$x,type="n")
text(acp1$x,labels=data.fasting$temps)
abline(h=0,v=0,lty=2)
```

Cette première ACP indique très clairement un effet "début" (à gauche) / "fin" de jeûne.

2.2 Classification ascendante hiérarchique

```
hc1 = hclust(dist(t(data.fasting[,1:130])),"ward")
plot(hc1,hang=-1)
plot(hc1$height[129:115],type="b")
```

Une classification ascendante hiérarchique indique une partition raisonnable en 4 ou 5 groupes.

```
# Recuperation du groupe de chaque gene
# en vue d'un codage couleur
coul = cutree(hc1,4)
plot(0,xlab="Time (h)",
  ylab="log (normalized intensity)",
  ylim=c(-0.5,2.6), xlim=c(0,72),type="n")
abline(v=unique(temps),lty=3)
for (i in 1:130)
   lines(unique(temps),data.fasting.moy[,i],
   col=coul[i])
```

La représentation des groupes indique que la classification s'est faite sur la base du niveau global d'expression mais pas selon le profil (croissant, décroissant, stationnaire) des gènes.

3 Lissage par spline

Plutôt que de calculer directement les moyennes des valeurs observées à chaque temps, nous envisageons une autre approche basée sur deux hypothèses :

- les valeurs acquises à chaque temps sont des obervations bruités d'une "vraie" valeur (inconnue) ;
- ce type de phénomène biologique devrait être une fonction du temps régulière et donc différentiable (i.e. sans singularités ni comportement chaotique). De plus, dans cette étude, le jeûne est un stimulus typiquement progressif qui implique la mise en place de changements métaboliques eux-aussi progressifs.

Pour atteindre le but recherché, nous allons passer par un lissage *spline* des données. Cette stratégie va permettre de représenter l'expression de chaque gène par une fonction mathématique assurant un compromis entre la proximité aux données ($\sum_{i;j}$ ...) et la régularité de la fonction ($\int_{t_1}^{t_J}$ ...) via un paramètre de lissage (λ) :

$$\min_{f \in H_1} \left\{ \sum_{i;j} \left[y_i^j - f(t_j) \right]^2 + \lambda \int_{t_1}^{t_J} [f''(u)]^2 du \right\} \quad (1)$$

3.1 Influence du paramètre de lissage

Nous illustrons l'influence du paramètre de lissage sur le gène CYP4A10.

```
plot(data.fasting$temps,data.fasting[,"CYP4A10"],
   pch="+")
```

Représentation des données relatives aux gènes CYP4A10.

```
mod.spline.0.8 = smooth.spline(data.fasting$temps,
  data.fasting[,"CYP4A10"],spar=0.8)
mod.spline.0.6 = smooth.spline(data.fasting$temps,
  data.fasting[,"CYP4A10"],spar=0.6)
mod.spline.0.4 = smooth.spline(data.fasting$temps,
  data.fasting[,"CYP4A10"],spar=0.4)
mod.spline.0.2 = smooth.spline(data.fasting$temps,
  data.fasting[,"CYP4A10"],spar=0.2)
```

Nous avons calulé ici 4 modèles pour le gène CYP4A10 correspondant à 4 valeurs du paramètres de lissage (0.8 - 0.6 - 0.4 - 0.2).

```
temps.discret = seq(0,72,l=50)
CYP4A10.0.8=predict(mod.spline.0.8,temps.discret)$y
CYP4A10.0.6=predict(mod.spline.0.6,temps.discret)$y
CYP4A10.0.4=predict(mod.spline.0.4,temps.discret)$y
CYP4A10.0.2=predict(mod.spline.0.2,temps.discret)$y
```

Le modèle sert ici à calculer les valeurs de la fonction en 50 points de discrétisation équi-répartis entre 0 et 72.

```
plot(0,xlim=c(0,72),ylim=c(-0.1,1.5),type="n",
  xlab="Time (h)",ylab="log (norm. intens.)")
points(data.fasting$temps,data.fasting[,"CYP4A10"],
   pch="+")
lines(temps.discret,CYP4A10.0.8,lwd=2,col=2)
lines(temps.discret,CYP4A10.0.6,lwd=2,col=3)
lines(temps.discret,CYP4A10.0.4,lwd=2,col=4)
lines(temps.discret,CYP4A10.0.2,lwd=2,col=5)
legend("bottomright",
  c(expression(lambda == 0.8),
    expression(lambda == 0.6),
    expression(lambda == 0.4),
    expression(lambda == 0.2)),
  lty=rep(1,4),lwd=rep(2,4),col=2:5)
```

3.2 Illustration de la contrainte de régularité

Afin de bien comprendre comment agit le paramètre de lissage sur la régularité de la fonction, nous calculons ici la dérivée seconde de la fonction pour les 4 valeurs testées précédemment. Il suffit pour cela de reprendre la fonction `predict` en précisant l'option `deriv=2`.

```
CYP4A10.0.8.d2=predict(mod.spline.0.8,
   temps.discret,deriv=2)$y
CYP4A10.0.6.d2=predict(mod.spline.0.6,
   temps.discret,deriv=2)$y
CYP4A10.0.4.d2=predict(mod.spline.0.4,
   temps.discret,deriv=2)$y
CYP4A10.0.2.d2=predict(mod.spline.0.2,
   temps.discret,deriv=2)$y
```

La représentation des fonctions dérivées secondes s'obtient de la même faÃ§on que les courbes lissées.

```
plot(0,xlim=c(0,72),ylim=c(-0.03,0.03),type="n",
  xlab="Time(h)",ylab="d2(log(norm. intens.))/dt2")
abline(h=0,lty=3)
lines(temps.discret,CYP4A10.0.8.d2,lwd=2,col=2)
lines(temps.discret,CYP4A10.0.6.d2,lwd=2,col=3)
lines(temps.discret,CYP4A10.0.4.d2,lwd=2,col=4)
lines(temps.discret,CYP4A10.0.2.d2,lwd=2,col=5)
legend("topright",
   c(expression(lambda == 0.8),
     expression(lambda == 0.6),
     expression(lambda == 0.4),
     expression(lambda == 0.2)),
       lty=rep(1,4),lwd=rep(2,4),col=2:5)
```

On constate les oscillations que subit la dérivée seconde pour les valeurs les plus faibles de λ. Dans le terme de droite de la quantité à minimiser 1, la dérivée seconde intervient au carré.

```
plot(0,xlim=c(0,72),ylim=c(0,0.001),type="n",
  xlab="Time(h)",ylab="d2(log(norm. intens.))/dt2")
abline(h=0,lty=3)
lines(temps.discret,CYP4A10.0.8.d2^2,col=2)
lines(temps.discret,CYP4A10.0.6.d2^2,col=3)
lines(temps.discret,CYP4A10.0.4.d2^2,col=4)
lines(temps.discret,CYP4A10.0.2.d2^2,col=5)
legend("topright",
   c(expression(lambda == 0.8),
     expression(lambda == 0.6),
     expression(lambda == 0.4),
     expression(lambda == 0.2)),
       lty=rep(1,4),lwd=rep(2,4),col=2:5)
```

Le terme intégral qui intervient dans la formule 1 correspondant à l'aire sous la courbe des fonctions tracées ainsi. On constate que seule une faible valeur de λ autorise une fonction de lissage dont la dérivée seconde oscille beaucoup (comme c'est le cas de la courbe turquoise). Avec un λ élevé, le prix à payer pour l'aire sous la courbe du carré de la dérivée seconde devient trop cher et les courbes retenues sont alors de plus en plus proche d'une droite (de dérivée seconde nulle).

3.3 Choix du paramètre de lissage

Des méthodes automatiques existent pour déterminer une valeur optimale du paramètre de lissage, mais des considérations biologiques peuvent également orienter le choix de ce paramètre. En notant par exemple que les oscillations observées dans l'expression de certains gène sont dues au rythme circadien, on peut considérer que ce phénomène n'est pas d'intérêt dans l'étude en cours et favoriser des valeurs de λ relativement élevée. En considérant également que la présence de plateau après une période de croissance ou de décroissance, la valeur de λ ne peut pas être non plus trop élevée pour ne pas manipuler que des profils quasi-rectilignes.

Ces considérations nous ont conduits à retenir la valeur $\lambda = 0.6$ pour l'ensemble des gènes. Une alternative aurait été d'adapter une valeur de λ à chaque gène mais nous n'avons pas fait ce choix ici.

Ce paramètre étant fixé, nous calculons la courbe lissée pour chaque paramètre.

```
param=0.6
discret=20
courbes.lisses.fasting = matrix(nr=130,nc=discret)

for (i in 1:130)
{
   tmp.spline=smooth.spline(data.fasting$temps,
      data.fasting[,i],spar=param)
   courbes.lisses.fasting[i,]=predict(tmp.spline,
      seq(0,72,l=discret))$y
}
```

3.4 Classification ascendante hiérarchique

Cela étant, mener la classification hiérarchique sur les courbes lissées n'améliore pas l'interprétation des résultats.

```
hc.lisse = hclust(dist(courbes.lisses.fasting),
  method="ward")
plot(hc.lisse,hang=-1)
cut.hc.lisse = cutree(hc.lisse,k=4)
plot(0,xlim=c(0,72),ylim=c(-0.5,3),type="n")
for (i in 1:130)
{
  points(data.fasting$temps,data.fasting[,i],
    col=cut.hc.lisse[i],lty=3,pch="+")
  lines(seq(0,72,l=discret),
    courbes.lisses.fasting[i,],col=cut.hc.lisse[i])
}
```

On retrouve, comme précédemment sur les donnés brutes, un regroupement des gènes selon le niveau global d'expression et non pas sur le profil.

4 Utilisation de la dérivée

Pour atteindre finalement notre objectif, nous allons utiliser la dérivée des courbes lissées qui contient l'information sur le profil d'expression : des gènes ayant un profil croissant (resp. décroissant) auront une dérivée positive (resp. négative) quel que soit leur niveau moyen d'expression.

4.1 Réprésentation des dérivées des courbes lissées

Nous calculons les dérivées premières des courbes lissées pour l'ensemble des gènes comme précédemment en initialisant une nouvelle matrice et en précisant l'option `deriv=1`.

```
deriv.courbes.lisses.fasting =
  matrix(nr=130,nc=discret)

for (i in 1:130)
{
```

```
   tmp.spline=smooth.spline(data.fasting$temps,
      data.fasting[,i],spar=param)
   deriv.courbes.lisses.fasting[i,]=
    predict(tmp.spline,
      seq(0,72,l=discret),deriv=1)$y
}
```

La représentation de ces dérivées n'est pas forcément très explicite.

```
plot(0,xlim=c(0,72),ylim=c(-0.05,0.06),type="n")
abline(h=0,v=temps,lty=3)
for (i in 1:130)
  lines(seq(0,72,l=discret),
    deriv.courbes.lisses.fasting[i,],col=i)
```

4.2 Classification ascendante hiérarchique

Nous travaillons maintenant sur les dérivées pour mettre en œuvre la classification hiérarchique. Le choix de 4 classes est discutable...

```
hc.deriv.courbes.lisses.fasting = hclust(
  dist(deriv.courbes.lisses.fasting),method="ward")
plot(hc.deriv.courbes.lisses.fasting, hang=-1)
cut.hc.deriv.courbes.lisses.fasting=
  cutree(hc.deriv.courbes.lisses.fasting,k=4)
```

La représentation des groupes de dérivées, n'est pas très explicite...

```
plot(0,xlim=c(0,72),ylim=c(-0.05,0.06),type="n")
abline(h=0,v=temps,lty=3)
for (i in 1:130)
  lines(seq(0,72,l=discret),
    deriv.courbes.lisses.fasting[i,],
     col=cut.hc.deriv.courbes.lisses.fasting[i])
```

... en revanche, si on projette les groupes obtenus sur les dérivées sur les courbes lissées, on obtient une représentation plus interprétable...

```
plot(0,xlim=c(0,72),ylim=c(-0.5,3),type="n")
for (i in 1:130)
  lines(seq(0,72,l=discret),
    courbes.lisses.fasting[i,],
     col=cut.hc.deriv.courbes.lisses.fasting[i])
```

... d'autant plus si on observe une représentation éclatée des groupes.

```
par(mfrow=c(1,4))
for (j in 1:4)
{
plot(0,xlim=c(0,72),ylim=c(-0.5,3),type="n")
for (i in
  which(cut.hc.deriv.courbes.lisses.fasting==j))
lines(seq(0,72,l=discret),
   courbes.lisses.fasting[i,],
   col=cut.hc.deriv.courbes.lisses.fasting[i])
}
```

Il apparaît ainsi clairement que les 4 groupes peuvent être caractérisés par :
- les profils stationnaires ;
- les profils faiblement croissants en début de jeûne puis stables ;
- les profils globalement décroissants ;
- 3 profils fortement croissants en début de jeûne puis stables.

Conclusion

Cette étude illustre l'utilisation d'un outil spécifique, le lissage *spline*, en vue d'atteindre un objectif précis fixé en termes biologiques : regrouper des gènes en fonction de leur profil d'expression. Si la classification apparaît clairement dans les méthodes à utiliser face à cet objectif, le recours au lissage n'est pas évident a priori.

Références

[1] S. Déjean, P. Martin, A. Baccini, P. Besse (2007) Clustering time series gene expression data using smoothing spline derivatives EURASIP Journal on Bioinformatics and Systems Biology, vol. 2007, article ID 70561