TP SPLEX

Séance N°2

Analyse d'expression différentielle avec le package "samr" ou le test t de Student associé à un ajustement des p-values brutes

1. **Exemple d'analyse d'expression différentielle à deux classes non appariées sur des données simulées:**

Le code suivant est proposé en exemple dans l'aide la fonction samr. Expliquer ce que fait chaque commande (vous pouvez utiliser la fonction help()).

```
library(samr)
set.seed(100)
x<-matrix(rnorm(1000*20),ncol=20)
dd<-sample(1:1000,size=100)

u<-matrix(2*rnorm(100),ncol=10,nrow=100)
x[dd,11:20]<-x[dd,11:20]+u
y<-c(rep(1,10),rep(2,10))

data=list(x=x,y=y,
geneid=as.character(1:nrow(x)),genenames=paste("g",as.character(1:nrow(x)),sep=""), logged2=TRUE)
samr.obj<-samr(data, resp.type="Two class unpaired", nperms=100)

delta=.4
samr.plot(samr.obj,delta)

delta.table <- samr.compute.delta.table(samr.obj)
siggenes.table<-samr.compute.siggenes.table(samr.obj,delta, data, delta.table)
```

2. **Réaliser la même analyse en utilisant un test t de Student et en corrigeant les p-values brutes obtenues en utilisant l'approche de correction type**

FDR de Benjamini & Yekutieli (2001)

Pour le test t de Student utiliser la fonction t.test().
Vous pouvez réaliser une boucle avec la commande suivante :
for (i in 1:n) {
code…
}

Pour l'ajustement des p-values brutes utiliser la fonction p.adjust(method="BY"); comparer les résultats obtenus si à la place de la correction de la FDR on utilise la méthode de Bonferroni i.e. p.adjust(method="bonferroni")

Comparer les résultats ainsi obtenus avec ceux obtenus précédemment en utilisant le package "samr" (seuil de significativité à 5%).

3. Exemple d'analyse d'expression differentielle à deux classes non appariées sur des données réelles:

Charger dans l'environnement le jeux de données "obèses/témoins" à partir du fichier text "obeses_temoins.txt" . Vous pouvez charger un fichier texte avec la fonction read.delim.

Réaliser une analyse d'expression differentielle en utilisant le package "samr" sur le modèle précédent en sachant que les patients obèses correspondent aux 25 premières colonnes du tableau alors que les témoins se trouvent dans les 10 dernières colonnes. Utiliser un FDR à environ 5%.

Sauvegarder les résultats pour réaliser ultérieurement une analyse fonctionnelle des gènes différentiellement exprimés.

TP SPLEX

Séance N°2

Clustering de données simulées

1. Exemple de clustering en utilisant l'algorithme kmeans

```
library(cluster)
set.seed(19)
data <- rbind(matrix(rnorm(100, sd = 0.3), ncol = 2),
matrix(rnorm(100, mean = 1, sd = 0.3), ncol = 2))
colnames(data) <- c("x", "y")
cl1 <- kmeans(data, 2)
plot(data, col = cl1$cluster)
points(cl1$centers, col = 1:2, pch = 8, cex=2)
```

2. Réaliser la même analyse en utilisant un algorithme de clustering hiérarchique

- Pour cela utiliser la fonction `hclust()` en imposant une découpe de l'arbre à 2 clusters: fonction `cutree()`
- Imprimer l'arbre de clusters en utilisant la fonction `plot()`

3. Calculer la silhouette des partitions ainsi obtenues et comparez les

- utilisez pour cela la fonction `silhouette(tree_cut, dist=distance_matrix)` du package `cluster`

Clustering de profils d'expression transcriptionnelle

Données d'expression issues de la comparaison des profils transcriptomiques du tissu adipeux chez des obeses massifs versus des sujets normopondéraux:

```
library(FunNet)
data(obese)
```

On peut lister les objects créés avec `objects()`. Les données à utiliser sont `down.frame` qui contient les gènes down-régulés, et `up.frame` qui contient les gènes up-régulés.

- Faire une ACP des données avec la librairie `FactoMineR` (fonction `PCA`). Que remarquez-vous sur les graphiques.
- Faire une analyse de clustering des patients en utilisant un algorithme de clustering hiérarchique et identifiez la partition optimale à l'aide d'un calcul itératif de la silhouette. Représenter cette partition sur le graphe PCA des individus.
- Faire de même avec un algorithme de k-means.

SPLEX TME 3

Mélange de Gaussiens et l'algorithme EM

Implementer l'algorithme EM et tester le sur les données "obeses_temoins" pour clusteriser les patients en deux classes. Comparez vous resultats avec ceux obtenus avec un package R, par exemple, le package R "mclust".

Mélange de Gaussiens

Si le événements sont issus d'une composition de N phénomènes, la densité p(X) prendra la forme d'une composition de lois normales. On peut l'approximer par une somme pondérée de densité normales, "mélange de Gaussiens"

$$p(x) = \sum_{m=1}^{M} \alpha_m \mathcal{N}(x; \mu_m, \Sigma_m), \tag{1}$$

ou

$$\mathcal{N}(x, \mu, \Sigma) = \frac{1}{(2\pi)^{D/2} \det(\Sigma)^{\frac{1}{2}}} \exp\Big(-1/2(x-\mu)^T \Sigma^{-1}(x-\mu)\Big). \tag{2}$$

. Dans le mĺange de Gaussiens il faut estimer trois paramètres

$$(\alpha, \mu, \Sigma). \tag{3}$$

Si toutes les μ et Σ etait fixe, on pourrait calculer les α_m directement. Mait ils sont libres et il faut donc les esimter par un processus iterative. Un tel processus est composé de deux étapes.

Algorithme EM

Soit une ensembe "training set" de N observation $X_{n\,n=1}^{N}$, $z^n \in \{0,1\}^m$ est l'indicateur de classe. La log-vraisemblence est

$$L(\alpha, \mu, \Sigma) = \sum_n \sum_m z_m^n \Big(\ln \alpha_m - \frac{1}{2} \ln |\Sigma_m| - \frac{1}{2} \mathrm{tr}(\Sigma_m^1 (x^n - \mu_m)(x^n - \mu_m)^T) \Big) \tag{4}$$

On fait une premier estimation des paramètres (α, μ, Σ) et puis on altern "Expectation" et "Maximisation" pendant plusieurs iterations t, tant que l'algorithme n'a pas convergé.

- "Expectation step": Faire une estimation des valeurs manquantes ("hidden")

$$z_m^n = p(z_m^n = 1|x^n, \alpha^{\text{old}}, \mu^{\text{old}}, \Sigma^{\text{old}}) = \frac{\alpha_m^t \mathcal{N}(x_n; \mu_m^t, \Sigma_m^t)}{\sum_{j=1}^M \alpha_j^t \mathcal{N}(x_n; \mu_j^t, \Sigma_j^t)}. \quad (5)$$

- "Maximisation step":

$$\alpha_m = \frac{1}{N} \sum_n z_m^n, \quad (6)$$

$$\mu_m = \frac{\sum_n z_m^n x^n}{\sum_n z_m^n}, \quad (7)$$

$$\Sigma_m = \frac{\sum_n z_m^n (x^n - \mu_m)(x^n - \mu_m)^T}{\sum_n z_m^n}. \quad (8)$$

SPLEX TME 4

La régression logistique

Implementer l'algorithme de la régression logistique et tester le sur les données simulées et les données "obeses_temoins" pour classifier les patients en deux classes. Comparez vous resultats avec ceux obtenus avec un package R.

1. Simulez un jeu de données

```
set.seed(666)
x1 = rnorm(1000)
x2 = rnorm(1000)
z = 1 + 2*x1 + 3*x2
pr = 1/(1+exp(-z))
y = rbinom(1000,1,pr)
```

et tester la régression logistique (la fonction *glm()*) sur ce jeu de données

```
df = data.frame(y=y,x1=x1,x2=x2)
glm( y~x1+x2,data=df,family="binomial")
```

2. La régression logistique binaire

Soit une ensemble "training set" de N observation $\{X_n, Y_n\}_{n=1}^N$. Dans le cadre de la régression logistique binaire, la variable Y prend deux modalités possibles $\{1, 0\}$. Les variables X sont exclusivement continues ou binaires.

La régression logistique est un modèle parametrique donc la log-vrasemblance est donnée par

$$\ell(Y|X;\theta) = -\sum_{i=n}^{N} \Big(y_n \theta^T x_n - \log(1 + \exp(\theta^T x_n)) \Big) \qquad (1)$$

où θ est le vecteur de paramètres à estimer.

Pour classifier une nouvelle observation X, on calcule et comapre les probabilités de classes sachant l'observation

$$p(Y = 1|X) = \frac{\exp \theta^T X}{1 + \exp \theta^T X} \tag{2}$$

$$p(Y = 0|X) = \frac{1}{1 + \exp \theta^T X}. \tag{3}$$

La méthode de Newton-Raphson

La méthode de Newton-Raphson qu'on utilise pour maximiser la log-vraisemblance et pour estimer les paramètres θ du modèle est une procedure itérative du gradient. Pour la régression logistique binaire la procedure est la suivante:

Initialize $\theta = (0, \ldots, 0)$

for $t = 1 : T$ // Faire plusieurs itérations ou jusqu'à la convergence **do**

Compute the first derivative
$\frac{\partial \ell(\theta)}{\partial \theta} = \sum_{n=1}^{N} x_n(y_n - p(y = 1|x_n))$ // dimension = 1×nb of parameters

Compute the Hessian matrix
$\frac{\partial^2 \ell(\theta)}{\partial \theta \partial \theta'} = \sum_{n=1}^{N} x_n x_n^t p(y = 1|x_n)(1 - p(y = 1|x_n))$
//dimension = nb of parameters × nb of parameters

Update the parameters
$\theta = \theta - \frac{\partial^2 \ell(\theta)}{\partial \theta \partial \theta'}^{-1} \frac{\partial \ell(\theta)}{\partial \theta}$
end for

TP SPLEX
Séance N°5

Analyse d'annotation fonctionnelle de deux listes de gènes

1. Exemple d'annotation fonctionnelle avec le package FunNet:

Charger dans l'environnement R le fichier RData issu de l'analyse d'expression différentielle du TP N°2.

```
load("obeses_temoins.RData")
```

Installer avec toutes les dépendances le package FunNet[1] et le charger ensuite dans R. Regarder le manuel pour l'explication des différents paramètres[2].

```
library(FunNet)
```

Créer deux variables contenant les noms des gènes (les Entrez GeneID) sur et sous exprimés en les transformant en data.frame.

Ajouter à chacune de ces data.frames 3 colonnes supplémentaires contenant que des valeurs nulles.

Appeler la fonction FunNet() en changent les paramètres suivants :

wd: le repertoire courant
org: "HS"
up.frame: la data frame transfromé contenant la liste des genes sur exprimés.
down.frame: la data frame transfromé contenant la liste des genes sous exprimés.
gene.frame et ref.list: NULL
two.lists: TRUE
restrict: FALSE
go.bp et kegg: TRUE
discriminant: TRUE
annot.method: "specificity"
fdr: NA
build.annot.net et estimate.th: FALSE

[1] http://cran.r-project.org/web/packages/FunNet/index.html
[2] http://cran.r-project.org/web/packages/FunNet/FunNet.pdf

```
level: 1
hard.th et soft.th: NA
topological: FALSE
coexp.method: "spearman"
keep.rdata: TRUE
zip: FALSE
```

Commenter les résultats obtenus dans les répertoires « images » et « html ».

2. Annotation fonctionnelle avec l'outil web FunNet:

Sauvegarder les deux listes de gènes sur et sous exprimés issu du TP n°2 dans des fichiers .txt et utiliser l'outil web FunNet[3] pour soumettre une analyse d'annotation fonctionnelle et utiliser les paramètres de l'exercice précédent.

Regarder le tutoriel[4] pour plus d'informations sur le format des fichiers et les paramètres. Faire cette analyse que pour le système d'annotation KEGG et comparer les résultats avec ceux obtenus dans l'exercice n°1.

3. Annotation fonctionnelle des données bypass:

Les données 'bypass' représentent des mesures d'expression des gènes du tissu adipeux sous-cutané après et avant une opération de chirurgie bariatrique. Le design de l'expérience est de type 1.

Nettoyer dans un premier temps l'environnement de travail de façon à voir les noms des variables chargés avec la commande :

```
rm(list=ls())
ls()
```

Charger les données de la librairie FunNet:

```
data(bypass)
```

Ensuite décrire les variables pour voir ce qu'elles contiennent.

Faire les modifications nécessaires pour adapter les variables comme dans l'exercice N°2 et lancer une analyse d'annotation avec l'outil web FunNet.

---

[3] Ce site se trouve à l'adresse www.funnet.ws ou www.funnet.info. Utiliser le premier.

[4] http://funnet.ws/index.php?menu=2

SPLEX TME 6

Support Vector Machines

1. Download the file *MetagenomicsExp.R* containing
   - A matrix of gut flora genes abundance, where each row corresponds to a patient and each column to a gene of gut flora.
   - A vector of classes: patients are stratified into two groups, a low gene count group and a high gene count group.
2. Install the *kernlab* R package
3. Here is a very well written tutorial on SVMs with *kernlab* in R

   `http://cbio.ensmp.fr/~jvert/svn/tutorials/practical/svmbasic/svmbasic_notes.pdf`
4. Train a linear SVM on the training (metagenomics) set
5. Use the learned model to predict the labels (you can do in on the same training set for the moment), and compute accuracy of your model
6. Implement a function to perform leave-one-out cross validation, i.e. you train your model on all patients excluding one patient, and you repeat this procedure for all patients.
7. Analyze the cross-validated accuracy
8. Consider the ROC curves
9. Repeat the same experiments with various non-linear kernels (rbfdot, polydot, etc.)
10. Test the effect of parameter C

TP SPLEX

Séance N°7

Classification supervisée et données puces

- **Télécharger le jeu de données contenu dans les fichiers suivants (utiliser read.table("mydata.txt", header=TRUE, row.names=1, sep=",") pour les fichiers de données) :**

  - data1_train.txt : contient les données d'apprentissage du jeu de données
  - data1_test.txt : contient les données de test du jeu de données
  - class_train.txt : contient les classes des exemples d'apprentissage
  - class_test.txt : contient les classes des exemples tests

Exercices :

- **Visualiser data1_train et data1_test en colorant les points en fonction de la classe**
- **Créer un classifieur sur les données d'apprentissage avec les méthodes LDA, QDA, KNN, svm, arbres de décision et boosting.**
- **Utiliser chaque classifieur pour prédire la classe des données d'apprentissage et des données de test**
- **Calculer le taux erreur de chaque méthode. Représenter ces résultats par un histogramme.**

Pour lda et qda, utilisez les fonctions `lda()` et `qda()` dans le package « MASS ».

Pour KNN vous utilisez la fonction `knn()` avec K=3 (package class).

Pour svm, utiliser la fonction `svm()` du package « e1071 ». Tester un noyau linéaire, un noyau polynomial et un noyau gaussien (utiliser les options par défaut). **Pour chaque classifieur, donnez le nombre de vecteurs de support.**

Pour les arbres de décision, utiliser le package rpart.

Pour le boosting, utilisez la fonction `adaboost.M1 ()` du package « adaboost ». **Donner le taux d'erreur en fonction du nombre d'itérations (du fait du caractère aléatoire du boosting, lancer les analyses plusieurs fois).**

NB : attention au format des données requit par chaque classifieur et par sa fonction `predict`

TP SPLEX N°8

Analyse transcriptionnelle de données expérimentales

Nous allons analyser des données expérimentales mesurant l'expression des gènes du tissu adipeux après et avant une opération bypass[1] sur des patients très obèses. Le design des puces est de type 1. Télécharger le jeu de données suivant et le charger en R. Transformer ces données de façon à ce que les geneids soient en nom de lignes.

1) before_after_bypass_100_SAM.csv

Exercice 1 – Analyse d'expression différentielle avec samr

Effectuer une analyse d'expression différentielle avec samr().

Sélectionner le delta de façon à ce qu'elle corresponde à la plus grande FDR pus petite que 0.05 (colonne "median FDR" de l'objet retourné de samr.compute.delta.table()).

Visualiser l'objet retourné par sam avec le delta sélectionné et le sauvegarder en pdf. Ensuite identifier les gènes significatifs et sélectionner que les 500 les plus significatifs pour des raisons de calcul.

Enfin extraire les profils d'expression pour les gènes significatifs et sauvegarder les fichiers (listes des gènes up, down, les profils d'expression up et down et le fichier de référence contenant tous les geneids dans le jeu). Avec la fonction plot.mat() du package WGCNA explorer visuellement les données.

Exercice 2 – Clustériser les patients

Explorer l'ensemble des données up et down en utilisant la fonction PCA() du package FactoMineR.Ensuite trouver la meilleure silhouette avec une classification kmeans et afficher le pca en utilisant les classes de kmeans comme couleurs (sachant que la silhouette n'a pas de sens sur des classifications triviales : un seul cluster ou un cluster par élément).

Exercice 3 – Annotation fonctionnelle

Effectuer une analyse d'annotation fonctionnelle avec FunNet sous R (discriminante et non discriminante) avec les fichiers produits dans l'exercice 1 (cf. TP 4 pour les détails de cette analyse).

[1] http://www.obesite-solution-chirurgie.com/

La même chose pour non discriminante mais avec le paramètre ci-dessous.

Exercice 4 – Analyse de co-expression

Utiliser les profils d'expression pour construire une matrice de corrélation. Avec un seuil de 0.8 transformer cette matrice en matrice d'adjacence. Créer ensuite un vecteur attribute.up2_down3 contenant des 2 les gènes up et des 3 pour les down.

En se basant sur la matrice d'adjacence filtrer les gènes qui n'ont aucun lien. Idem pour le vecteur attribute.up2_down3.

Créer un objet de type réseaux avec ces données et ajouter l'attribut attribute.up2_down3 à ce dernier.

Combien de nœuds et de liens a ce réseaux ?

Afficher le réseau en couleurs et le sauvegarder en pdf.

Avec la fonction deg() de la librairie sna calculer la connectivité du réseau (avec la matrice d'adjacence cette fois-ci) et afficher ensuite le graphe scale-free. Sauvegarder ce dernier en pdf.

Refaire cet exercice pour les seuils 0.75, 0.85, 0.9 et comparer les différents paramètres. Que peut-on conclure ? Quel est le meilleur seuil ?

Exercice 5 – Analyse fonctionnelle de co-expression

Avec les fichiers produits dans l'exercice 1, utiliser l'outil web FunNet pour estimer le seuil de co-expression. Effectuer ensuite une analyse discriminante en utilisant ce seuil et identifier la liste des 10 premiers gènes les plus connectés, intermédiaires, et fonctionnels[2].

Exercice 6 – Classification

Dans cet exercice nous allons créer un classifier qui classera les gènes en fonction de leur profils d'expression. Extraire dans un premier temps 100 gènes de façon aléatoire de la liste up et 100 de la liste down avec la fonction sample()[3] pour l'ensemble d'apprentissage. Les gènes restants les affecter à l'ensemble de test. Créer également deux vecteurs (un pour les données train et un pour test) contenant des A et des B en fonction de l'appartenance des données à la liste up ou down. Transformer ces derniers en type facteur. Construire un classifier lda et calculer le taux d'erreur sur l'ensemble de test. Refaire cet exercice mais cette fois-ci tester le classifier ci-dessus avec des données non significatives (FDR >5%). Pour cela recalculer samr.compute.siggenes.table() avec l'option all.genes=TRUE et extraire les données nécessaires. Discuter ces résultats.

---

[2] Ceci peut être fait avec Cytoscape ou bien sur R en chargeant d'abord le fichier texte contenant ces informations [système_annot]_genes_net_info.txt.

[3] Il faut tirer aléatoirement des indices (ces indices soient uniques). Utiliser l'opérateur %in% pour faire la correspondance.

SPLEX TME 9

Bayesian Networks

The **bnlearn** R package provides algorithms both for discrete and continuous data to estimate structure in data and to construct Bayesian networks.

1. Install the bnlearn package.
2. Load the data MetagenomicsExp.R. The file contains a matrix of gene abundance for gut flora. While construction the networks, keep in mind the question whether it were possible to cluster the genes based on obtained networks. How?
3. The bnlearn package disposes of several constraint-based and score-based algorithms to learn the structure of data. Test some of the approaches. Take into consideration that your data are continuous, therefore, be careful choosing (conditional) independent tests for a constraint-based algorithm and a network score for a score-based approach.
4. Construct several Bayesian networks using different approaches, visualize the networks, and compare them.
5. Consider the function modelstring() to see a learned graph as text.
6. Test the function arc.strength() (and boot.strength()) which measure the probabilistic relationships expressed by the arcs of a Bayesian network, and use model averaging to build a network containing only the significant arcs.
7. Based on the arcs strength, diminish the number of edges in your network, i.e., delete arcs whose probability is less than some threshold (e.g., < 0.5 or < 0.7).
8. Install the Cytoscape open source software

   `http://www.cytoscape.org/`

9. Visualize one of your networks with Cytoscape; make the nodes size be proportional to the number of adjacent vertices; make the width of edges be proportional to the strength of the arcs.

TP SPLEX

Séance N°10

Analyse des réseaux transcriptionnels (co-expression)

1. Exploration des réseaux de co-expression sous R

Installer et charger dans l'environnement R la librairie 'network' et 'WGCNA'. Le package 'network' contient des outils pour la création, l'accès et la modification des objets appartenant à la classe portant le même nom. L'avantage de tels objets par rapport à d'autres représentations plus classiques (les matrices d'adjacence) consiste à une manipulation plus performante en terme de calcul et à la possibilité d'abriter un grand nombre d'attributs. Le package WGCNA (Weighted Gene Co-Expression Network Analysis) contient un certain nombre de fonctions d'affichage dont nous allons nous servir dans ce TP.

Télécharger le jeux de données *obèse vs. témoin*[1] et charger sur R les deux fichiers "up_regulated.txt" et "down_regulated.txt". La première colonne qui contient les GeneID il faut la mettre en nom de lignes et ensuite la supprimer.

Joindre les deux variables en une seule matrice qui contient toutes les données.

Créer un vecteur de caractères qui contient "red" dans les 11 premières cases et "blue" dans les 28 suivantes.

Afficher la matrice avec les données d'expression en utilisant la fonction plot.mat() et en affectant le vecteur des couleurs à l'option ccols.

Sauvegarder l'image dans un fichier pdf.

Charger la librairie Hmisc (déjà installé car c'est une dépendance de FunNet).

Utiliser la fonction rcorr() de Hmisc pour calculer une matrice de coefficients de corrélation. Cette fonction permet de calculer en une seule fois toutes les corrélations possibles dans une matrice donnée

[1] http://funnet.ws/test_data.zip

et renvoie une liste avec plusieurs objets. Affecter le résultat contenant les coefficients de corrélation à la variable correlations.

Avec la fonction plot.cor() afficher l'image de cette matrice de coefficients de corrélation et la sauvegarder dans un fichier pdf. Que peut-on déduire de cette image ?

Transformer la matrice de corrélations dans une matrice d'adjacence (i.e. matrice qui contenant des 0 et des 1) en utilisant un seuil de 0.8. Cette matrice doit contenir des 0 dans sa diagonale (voir l'aide de la fonction diag()).

Créer un vecteur de même longueur que le nombre de gènes que l'on va appeler attribute.up2_down3 et qui va contenir des 2 pour les gènes up et des 3 pour les down.

Enlever les lignes et les colonnes de la matrice d'adjacence pour lesquels la somme correspondante est nulle. Utiliser un de ces indexes pour enlever les éléments correspondants du vecteur attribute.up2_down3.

Combien de gènes a-t-on filtrée ? Pourquoi on les enlève ces gènes ? Afficher à l'aide de la fonction heatmap() la matrice d'adjacence filtrée et la sauvegarder dans un fichier pdf. Comment peu-on commenter cette image ?

Transformer avec la fonction as.network.matrix() la matrice d'adjacence dans un objet network et affecter au paramètre vertex.names les GeneId. Attention, il s'agit d'un graphe non directionnel. Ajouter ensuite au réseaux l'attribut attribute.up2_down3 avec la fonction set.vertex.names().

Quelle est la taille du réseau ? Combien de liens y a-t-il ?

Afficher le réseau avec la fonction plot.network() en affectant au paramètre col le vecteur attribute.up2_down3. Enregistrer l'image dans un fichier pdf. Comment peut-on commenter cette image ? Quelles relations avec les autres images qu'on a obtenues avant ?

Extraire les gènes qui corrèlent avec le gène 196740 (aussi appelés gènes du voisinage) en utilisant la fonction get.neighborhood().

Maintenant nous allons utiliser le package sna issu des sciences sociales pour étudier les propriétés du réseau de co-expression. Le package sna reconnaît les matrices d'adjacence. Calculer la connectivité et l'intermédiarité à l'aide des fonctions degree() et betweenness(). Affecter aux valeurs résultantes les noms des gènes correspondants.

Afficher les distributions[2] de ces deux vecteurs et sauvegarder les

[2] Pour rappel ploter la distribution d'un vecteur consiste à afficher dans un graphique les valeurs triées par ordre décroissant de ce même vecteur.

images en pdf.

Utiliser la fonction scaleFreePlot() du package WGCNA pour tester si ce réseau est scale-free.

Enfin extraire les 10 gènes les plus connectés et les 10 les plus intermédiaires. Croiser ces deux listes et investiguer les gènes en commun sur le site de NCBI[3].

Refaire cette analyse pour d'autres seuils O.7, 0.75, 0.85, 0.9 et comparer les différents paramètres : nombre de nœuds, liens, voisins, modularité, etc.

2. Analyse des réseaux de co-expression avec FunNet

Utiliser les fichiers "up_regulated.txt" et "down_regulated.txt" pour effectuer une analyse de co-expression discriminante avec FunNet. Utiliser les systèmes d'annotation KEGG et GO biological process avec le niveau 5 et un seuil *hard* de 0.8. Une fois les résultats obtenus, les charger sur Cytoscape l'aide du plugin FunNetViz. Explorer les mesures de centralité et sélectionner les 10 gènes les plus importants (avec les trois mesures). Utiliser les plugin *NeighborNodeSelection* pour sélectionner les voisins des gènes les plus centraux.

Explorer ensuite les réseaux fonctionnels et filtrer les liens inter et intra modulaire. Que peut-on dire ? Explorer d'autre plugins de Cytoscape tels que MiMi, CluGO, BINGO, etc.

[3] http://www.ncbi.nlm.nih.gov/gene