1 Données

Dans ces travaux dirigés on va analyser des données de puces avec la classification et la discrimination. Telecharger les donnees:

```
> library(golubEsets)
```

I y a trois ensembles de donnees: Golub_Train, Golub_Test, and Golub_Merge dans une class speciale.

```
> data(Golub_Merge)
> ls()
```

Regardez les details des données, on utilisera Golub_Train comme ensemble d'apprentissage, et Golub_Test comme ensemble test. Golub_Merge pour faire l'analyse en validation croisée. et Golub_Merge pour la classification.

2 Normalization of data

On regarder la distribution des données. On peut observer les données aberrantes. Il y a des nombres negatifs aussi a cause d'un enlevement de fonds trop fort.

```
> hist(exprs(Golub_Merge))
```

Plus de details:

```
> hist(exprs(Golub_Merge),breaks=100)
```

ou

```
> plot(density(exprs(Golub_Merge)))
```

Information sur l'experience:

```
> sampleNames(Golub_Merge)
> phenoMerge <- pData(phenoData(Golub_Merge))
```

faites le meme pour les deux autres données (Golub_Train and Golub_Test) et comparer les resultats. Nous processons les données comme ont fait les auteurs de l'article, on modifie les valeurs extremees, les valeurs plus petites que 100 sont fixés a 100 et les valeurs plus grandes que 16,000 fixés a 16,000.

```
> gme <- exprs(Golub_Merge)
> gme[gme<100] <- 100
```

Faites pareils pour les valeurs grandes.

Maintenant prenons le log base 2 des données.

```
gme <- log2(gme)
```

Regardons les distributions de nouveau. Faites la meme chose avec les deux autres ensembles: on les appelera les matrices gtre et gtee a partir des données d'apprentissage et test.

Mainenant les données sont en bonne forme, on peut faire l'etape de filtrage.

3 Filtrage des genes

Une des des premieres taches consiste a enlever les genes qui ne seront pas interressants.

Moins de 40% des genes seront d'interet. On commence par enlever les genes qui ne sont pas exprimes et aussi les genes de maintenance qui sont exprimes de facon constant (housekeeping).

On utilise la bibliotheque genefilter.

```
>library(genefilter)
```

Les fonctions de filtrage lisent chaque ligne et produit une sortie booleenne qu'on peut utiliser pour prendre des sous ensembles de genes.

Essayer ce code:

```
mmfilt <- function(r = 5, d = 500, na.rm = TRUE){
function(x) {
minval <- min(2^x, na.rm = na.rm)
maxval <- max(2^x, na.rm = na.rm)
(maxval/minval > r) && (maxval - minval > d)
}
}
```

Que fait cette fonction?

```
>ffun <- mmfilt()
>sub<- genefilter(gtre, ffun)
>gtres <- gtre[sub, ]
>gtees <- gtee[sub, ]
>gmes <- gme[sub,]
```

Combien de genes restent -il?

3.1 Filtrage des genes

Il y a encore trop de genes qui restent?

Comme on veut etudier la difference de deux types de Lymphome on va se concentrer a cce sous ensemble. (ALL, AML).

On va garder les genes qui presentent une difference entre les deux groupes.

```
ttfun <- function(x){
  the.test <- t.test(x~phenoMerge$ALL.AML)
  if (the.test$p.value<10^(-4)) answer <- T
  else answer <- F
  return(answer)
}
```

La fonction ttfun utilise la fonction interne t.test pour faire le filtrage. On utilise le niveau $p = 1e - 04$. Nous utilisons l'information dans l'objet phenoMerge sur les types de leucemie.

Repeter ce que vous avez fait en utilisant la fonction ttfun (modifié pour utiliser les données phenoTrain et phenoTest) et les donnees gmes, gtres, et gtees. Appeler les resultats gmess, gtress, et gteess.

Combien de genes restent-ils?

Pourquoi n'utilise-t-on pas $p = 0.05$ comme la valeur de signification?

4 Location et Echelle pour standardization:

Standardization de location et echelle, on va centrer et standardiser les données: moyennes 0 et var 1.

```
star <- function(x) (x - mean(x))/sd(x)
merged <- t(apply(gmess, 1, star))
training <- t(apply(gtress, 1, star))
testing <- t(apply(gteess, 1, star))
```

5 Discrimination

On utilise les methodes supervisees pour les categories ALL.AML. On utilise les données d'apprentissage pour construire le modele, on peut utiliser les methodes des voisins les plus proches pour assigner la classe. library: class.

On peut aussi utiliser les voisins les plus proches $knn()$ pour classifier les patients Utiliser knn() avec les parametres $k = 1, 11, 22, 33$ pour faire des classifications separees.

Le parametre cl: est la verification veritable des données d'apprentissage: phenoMerge$ALL.AML.

Remarque: Nos matrices ont des patients comme colonnes et les variables(genes) comme lignes.

On doit transposer les donnees avant d'utiliser knn(). La fonction de transposition est t().

Quel est le taux de mauvais classement des differents essais? Quel est le meilleur choix de k?

```
knn(t(training),t(testing),phenoTrain$ALL.AML)  ==  phenoTest$ALL.AML
```

On utilise les valeurs d'expression dans les donnees assembles et la fonction knn.cv .

```
knn.cv(t(merged),phenoMerged$A,k=1)==phenoMerged$A
```

Quel est le taux d'erreur?

6 Cluster Analysis:

On veux faire un groupement de patients on peut utiliser les distances euclidiennes calculees sur la matrice d'ensemble:

```
> merged.dist <- dist(t(merged))
```

(We need to transpose the data matrix, because dist() computes distances between the rows of the matrix. What happens if you skip the t() ?

Le resultat est une matrice 72 by 72. D'autres types de distances se calculent avec d'autres arguments, utiliser help(dist).

Creer une classification hierarchique hierarchical clustering tree ou dendrogram.

```
> merged.ht <- hclust(merged.dist)
```

Faites help(hclust) pour voir les options.

L'object de sortie de hclust s'utilise de plusieurs facons. La sortie graphque standard s'obtient avec:

```
> plot(merged.ht)
```

(plclust donne quelquechose de semblable).

Ou couperez vous?

Couper chaque erreur en deux groupes en utilisant la fonction cutree. Utiliser les deux groupes pour classer les patients:

```
> merged.hc.cut <- cutree(merged.dist,k=2)
> merged.hc.cut
> cbind(merged.hc.cut,phenoMerged$A)
```

Quel est le taux d'erreur?
Faites le graphique de classification avec la fonction heatmap.

7 Fonction de discrimination linéaire de Fisher

La methode de discrimination lineaire (LDA) utilise les données d'apprentissage pour choisir les variables qui produisent une bonne "discrimination" des groupes.

Ensuite on peut essayer sur les nouvelles données.

```
library(MASS)
my.lda <- lda(t(training),groups=phenoTraining$A)
names(my.lda)
```

Maintenant on utilise la fonction de prediction sur les données tests:

```
my.predictions <- predict(my.lda,t(testing))
```

Quel est le taux d'erreur?

```
mean(my.predictions != phenoTesting$A)
```

Comparer a l'erreur de prediction estime par la methode de validation croisee:

```
ldawithcv <- lda(t(training),groups=phenoTraining$A,CV=TRUE)
```