TP1

Exercice n°1 : Tortues

(a) Charger le jeu de données `PaintedTurtles.txt` contenu dans le dossier data et le décrire.

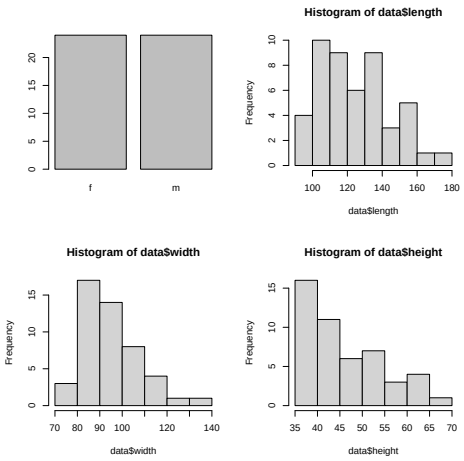
Solution:

```
data=read.table("data/PaintedTurtles.txt",header = T)
dim(data) ## 48 individus stats (des tortues)
## [1] 48  4
## 4 variables
```

(b) Décrire chaque variable du jeu de données et proposer un modèle probabiliste possible pour chaque variable (sans aller jusqu'à tester l'adéquation des données à une loi donnée).

Solution:

```
table(data$sex)
##
##  f  m
## 24 24
par(mfrow=c(2,2))
barplot(table(data$sex))
hist(data$length)
hist(data$width)
hist(data$height)
```

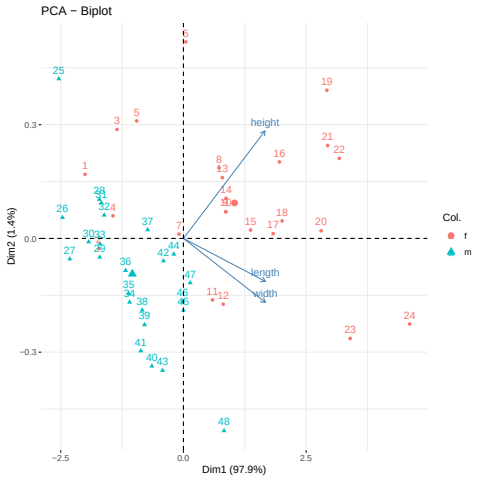


Une variable qualitative (sexe : F ou H), que l'on peut modéliser par une loi de Bernoulli. Trois variables quantitatives, modélisables par des lois normales (la variable `height` est cependant très asymétrique)

(c) Réaliser une ACP et interpréter les résultats de cette visualisation.

Solution:

```
library(FactoMineR)
resPCA <- PCA(data,scale.unit = TRUE,quali.sup = 1,graph = FALSE)
## la variable sexe n'est pas incluse dans l'ACP
## on spécifie qu'il s'agit d'une variable supplémentaire
library(factoextra,verbose = FALSE)
## Loading required package: ggplot2
## Welcome! Want to learn more? See two factoextra-related books at https://goo.g
fviz_pca_biplot(resPCA,col.ind = data$sex)
```



On remarque que l'axe 1 capture quasiment toute la variabilité du jeu de données initial. Les trois dimensions se résume en une variable qui indique si la tortue est grande ou petite. En coloriant les points en fonction de la variable sexe, on constate que les tortues femelles sont grandes et les mâles petits.

Exercice n°2 : Visualisations des données transcriptomiques `T-cell`

Ces données sont tirées du livre "Modern Statistics for Modern Biology", Susan Holmes et Wolfgang Huber. Il s'agit de données transcriptomiques issus de l'article suivant Holmes, Susan, Michael He, Tong Xu, and Peter P Lee. 2005. *"Memory T Cells Have Gene Expression Patterns Intermediate Between Naive and Effector."* PNAS 102 (15) : 5519–23. Les profils d'expression de gènes ont été mesurés chez 10 individus pour trois types de cellules.

```
load("data/Msig3transp.RData")
```

(a) Décrire le jeu de données.

Solution:

```
load("data/Msig3transp.RData")
dim(Msig3transp) ## 30 lignes, 156 colonnes
## [1]  30 156
rownames(Msig3transp)
##  [1] "HEA26_EFFE_1" "HEA26_MEM_1"  "HEA26_NAI_1"  "MEL36_EFFE_1" "MEL36_MEM_1"
##  [6] "MEL36_NAI_1"  "HEA31_EFFE_2" "HEA31_MEM_2"  "HEA31_NAI_2"  "MEL39_EFFE_2
## [11] "MEL39_MEM_2"  "MEL39_NAI_2"  "HEA25_EFFE_3" "HEA25_MEM_3"  "HEA25_NAI_3"
## [16] "MEL53_EFFE_3" "MEL53_NAI_3b" "MEL53_NAI_3"  "HEA55_EFFE_4" "HEA55_MEM_4"
## [21] "HEA55_NAI_4"  "MEL67_EFFE_4" "MEL67_MEM_4"  "MEL67_NAI_4"  "HEA59_EFFE_5
## [26] "HEA59_MEM_5"  "HEA59_NAI_5"  "MEL51_EFFE_5" "MEL51_MEM_5"  "MEL51_NAI_5"
round(Msig3transp,2)[1:5, 1:6]
##              X3968 X14831 X13492 X5108 X16348  X585
## HEA26_EFFE_1 -2.61  -1.19  -0.06 -0.15   0.52 -0.02
## HEA26_MEM_1  -2.26  -0.47   0.28  0.54  -0.37  0.11
## HEA26_NAI_1  -0.27   0.82   0.81  0.72  -0.90  0.75
## MEL36_EFFE_1 -2.24  -1.08  -0.24 -0.18   0.64  0.01
## MEL36_MEM_1  -2.68  -0.15   0.25  0.95  -0.20  0.17
```

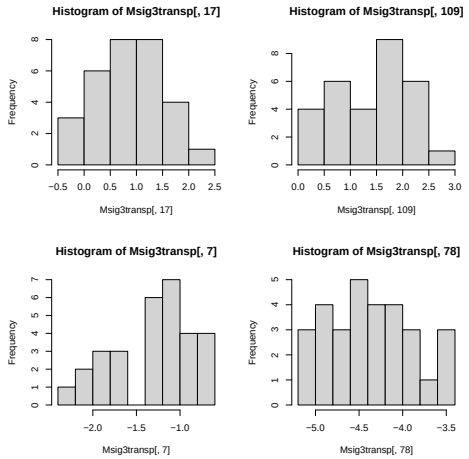
Les échantillons correspondent aux 30 lignes, 10 personnes pour 3 types cellulaires. Les 156 colonnes correspondent aux gènes. Les informations sur les échantillons sont contenus dans le nom des lignes. On peut récupérer ces infos.

```
## la fonction strsplit coupe une chaine de caractère
strsplit(rownames(Msig3transp)[1],"_")
## [[1]]
## [1] "HEA26" "EFFE"  "1"
## la fonction apply permet d'appliquer
## une opération à chaque élément d'une liste
## la fonction sapply permet de "simplifier"
## (transformer la sortie de apply qui est une liste)
## en un simple vecteur
info1 <- sapply(1:30, function(x) strsplit(rownames(Msig3transp)[x],"_")[[1]][1])
info2 <- sapply(1:30, function(x) strsplit(rownames(Msig3transp)[x],"_")[[1]][2])
info3 <- sapply(1:30, function(x) strsplit(rownames(Msig3transp)[x],"_")[[1]][3])
table(info1) ## contient une info que l'on devine etre l'indice de l'individu
## info1
## HEA25 HEA26 HEA31 HEA55 HEA59 MEL36 MEL39 MEL51 MEL53 MEL67
##     3     3     3     3     3     3     3     3     3     3
table(info2) ## contient une info que l'on devine être le type cellulaire
## info2
## EFFE  MEM  NAI
##   10    9   11
table(info3) ## info difficile à décrypter ... batch ? labo ?
## info3
##  1  2  3 3b  4  5
##  6  6  5  1  6  6
```

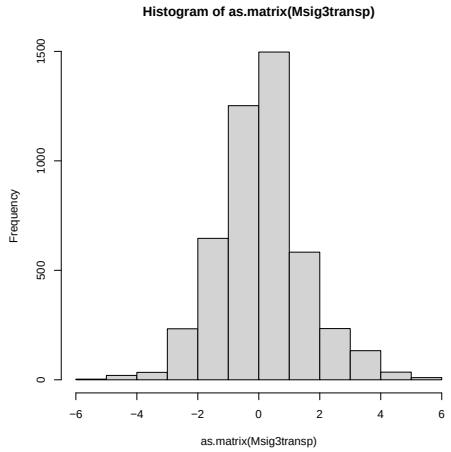
(b) Décrire chaque variable du jeu de données et proposer un modèle probabiliste possible pour chaque variable (sans aller jusqu'à tester l'adéquation des données à une loi donnée).

Solution: Il y a 156 variables (156 gènes).

```
par(mfrow=c(2,2))
hist(Msig3transp[,17])
hist(Msig3transp[,109])
hist(Msig3transp[,7])
hist(Msig3transp[,78])
```




```
range(Msig3transp)
## [1] -5.046580  5.905102
par(mfrow=c(1,1))
hist(as.matrix(Msig3transp))
```

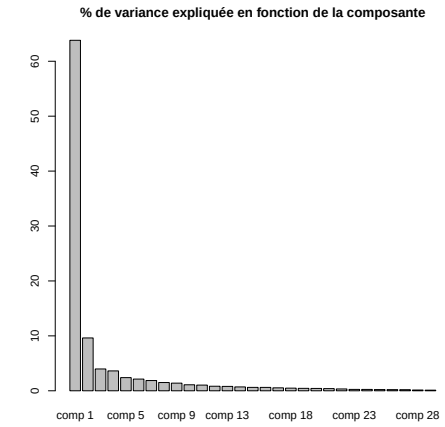


Il est difficile de visualiser la distribution de 156 variables, on peut cependant proposer une loi normale pour chaque variable.

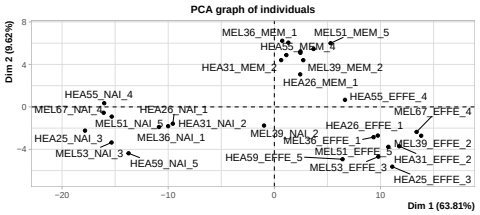
(c) Réaliser une ACP et interpréter les résultats de cette visualisation.

Solution:

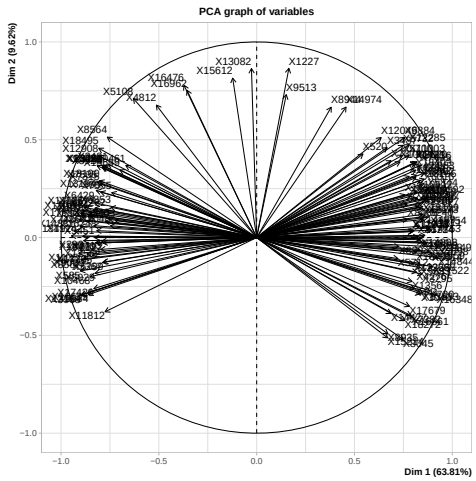
```
library(FactoMineR)
info1 <- factor(info1);info2 <- factor(info2)
info3 <- factor(info3)
resPCA <- PCA(Msig3transp,scale.unit = TRUE,graph=FALSE)
barplot(resPCA$eig[,2],main="% de variance expliquée en fonction de la composante
```



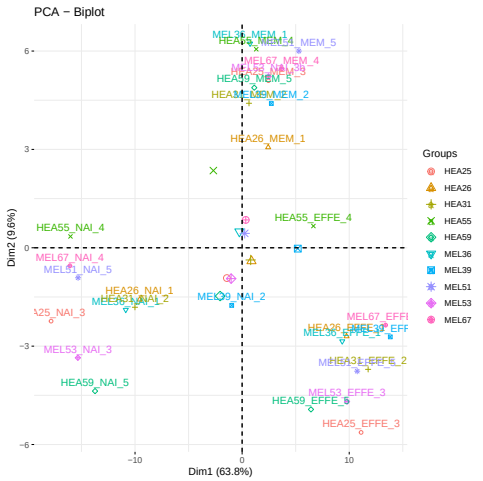

```
## on garde les 2 premières composantes
par(mfrow=c(1,2))
plot(resPCA,choix = "ind")
## Warning: ggrepel: 4 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```



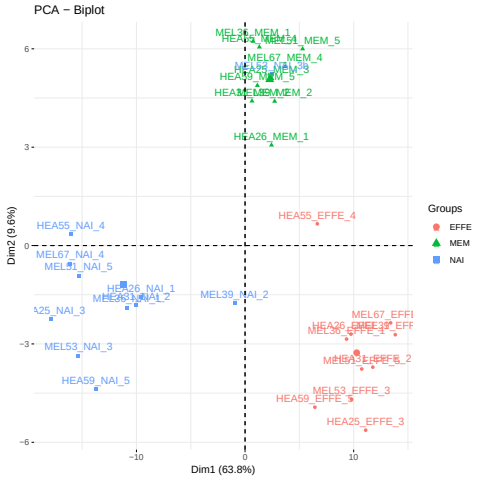

```
plot(resPCA,choix = "var")
```



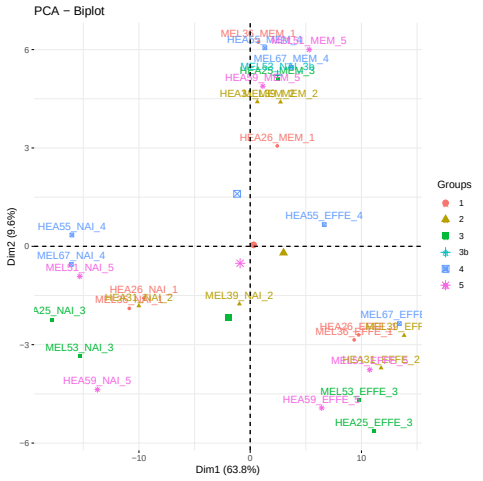

```
library(factoextra,verbose = FALSE)
par(mfrow=c(1,2))
fviz_pca_biplot(resPCA,habillage = info1,invisible = "var")
```




```
fviz_pca_biplot(resPCA,habillage = info2,invisible = "var")
```




```
fviz_pca_biplot(resPCA,habillage = info3,invisible = "var")
```



Ici, on remarque que le pourcentage de variance projeté est assez élevé, l'info contenue dans les 156 variables est donc redonantes. On remarque également que le type cellulaire structure le jeu de données. Les échantillons se ressemblent entre eux en fonction du type cellulaire. Cependant, on remarque un type cellulaire anormal dans le groupement des types cellulaires "MEM". Il semblerait qu'un échantillon "NAI" se soient glissés dans le type cellulaire "MEM". En réalité, il s'agit probablement d'une erreur d'étiquetage. Pour l'individu "MEL53", l'échantillon "MEM" a été mal étiqueté ("NAI" au lieu de "MEM").

```
table(info2,info1)
##       info1
## info2  HEA25 HEA26 HEA31 HEA55 HEA59 MEL36 MEL39 MEL51 MEL53 MEL67
##   EFFE     1     1     1     1     1     1     1     1     1     1
##   MEM      1     1     1     1     1     1     1     1     0     1
##   NAI      1     1     1     1     1     1     1     1     2     1
```

Exercice n°3 : Données d'expression de gènes

Les données suivantes sont extraites de cet article `https://pubmed.ncbi.nlm.nih.gov/21455293/`. L'étude visait à comparer l'expression des gènes à l'aide de la technologie RNA-seq chez deux lignées de souris. Les données sont téléchargables depuis les adresses `https://bowtie-bio.sourceforge.net/recount/countTables/bottomly_count_table.txt` et `https://bowtie-bio.sourceforge.net/recount/phenotypeTables/bottomly_phenodata.txt`.

(a) Charger les données suivantes. Décrire le jeu de données.

Solution:

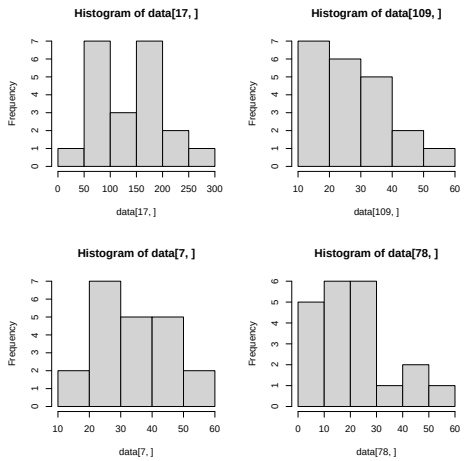
```
monurl <- "https://bowtie-bio.sourceforge.net/recount/countTables/bottomly_count_
data <- read.table(monurl,
                   header=TRUE,row.names = 1)
dim(data)
## [1] 36536    21
monurl2 <- "https://bowtie-bio.sourceforge.net/recount/phenotypeTables/bottomly_p
phenotype <- read.csv2(monurl2,sep=" ")
phenotype
##    sample.id num.tech.reps   strain experiment.number lane.number
## 1  SRX033480             1 C57BL/6J                 6           1
## 2  SRX033488             1 C57BL/6J                 7           1
## 3  SRX033481             1 C57BL/6J                 6           2
## 4  SRX033489             1 C57BL/6J                 7           2
## 5  SRX033482             1 C57BL/6J                 6           3
## 6  SRX033490             1 C57BL/6J                 7           3
## 7  SRX033483             1 C57BL/6J                 6           5
## 8  SRX033476             1 C57BL/6J                 4           6
## 9  SRX033478             1 C57BL/6J                 4           7
## 10 SRX033479             1 C57BL/6J                 4           8
## 11 SRX033472             1   DBA/2J                 4           1
## 12 SRX033473             1   DBA/2J                 4           2
## 13 SRX033474             1   DBA/2J                 4           3
## 14 SRX033475             1   DBA/2J                 4           5
## 15 SRX033491             1   DBA/2J                 7           5
## 16 SRX033484             1   DBA/2J                 6           6
## 17 SRX033492             1   DBA/2J                 7           6
## 18 SRX033485             1   DBA/2J                 6           7
## 19 SRX033493             1   DBA/2J                 7           7
## 20 SRX033486             1   DBA/2J                 6           8
## 21 SRX033494             1   DBA/2J                 7           8
dim(phenotype)
## [1] 21  5
table(phenotype$strain)
##
## C57BL/6J   DBA/2J
##       10       11
table(phenotype$experiment.number)
##
## 4 6 7
## 7 7 7
table(phenotype$lane.number)
##
## 1 2 3 5 6 7 8
## 3 3 3 3 3 3 3
```

On a 36536 gènes et 21 échantillons. On a 5 variables descriptives des échantillons. La variable strain indique la lignée d'appartenance des 21 souris.

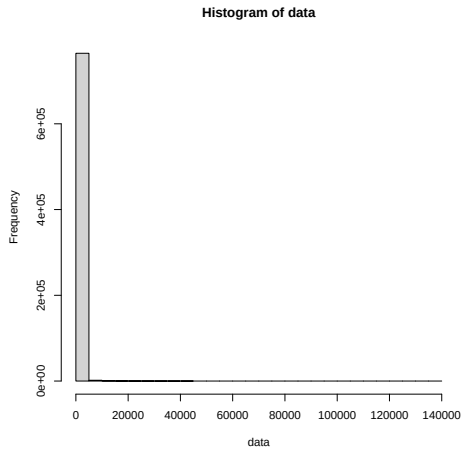
(b) Décrire chaque variable du jeu de données et proposer un modèle probabiliste possible pour chaque variable (sans aller jusqu'à tester l'adéquation des données à une loi donnée).

Solution: Il y a 36536 variables (36536 gènes).

```
data <- as.matrix(data)
par(mfrow=c(2,2))
hist(data[17,])
hist(data[109,])
hist(data[7,])
hist(data[78,])
```



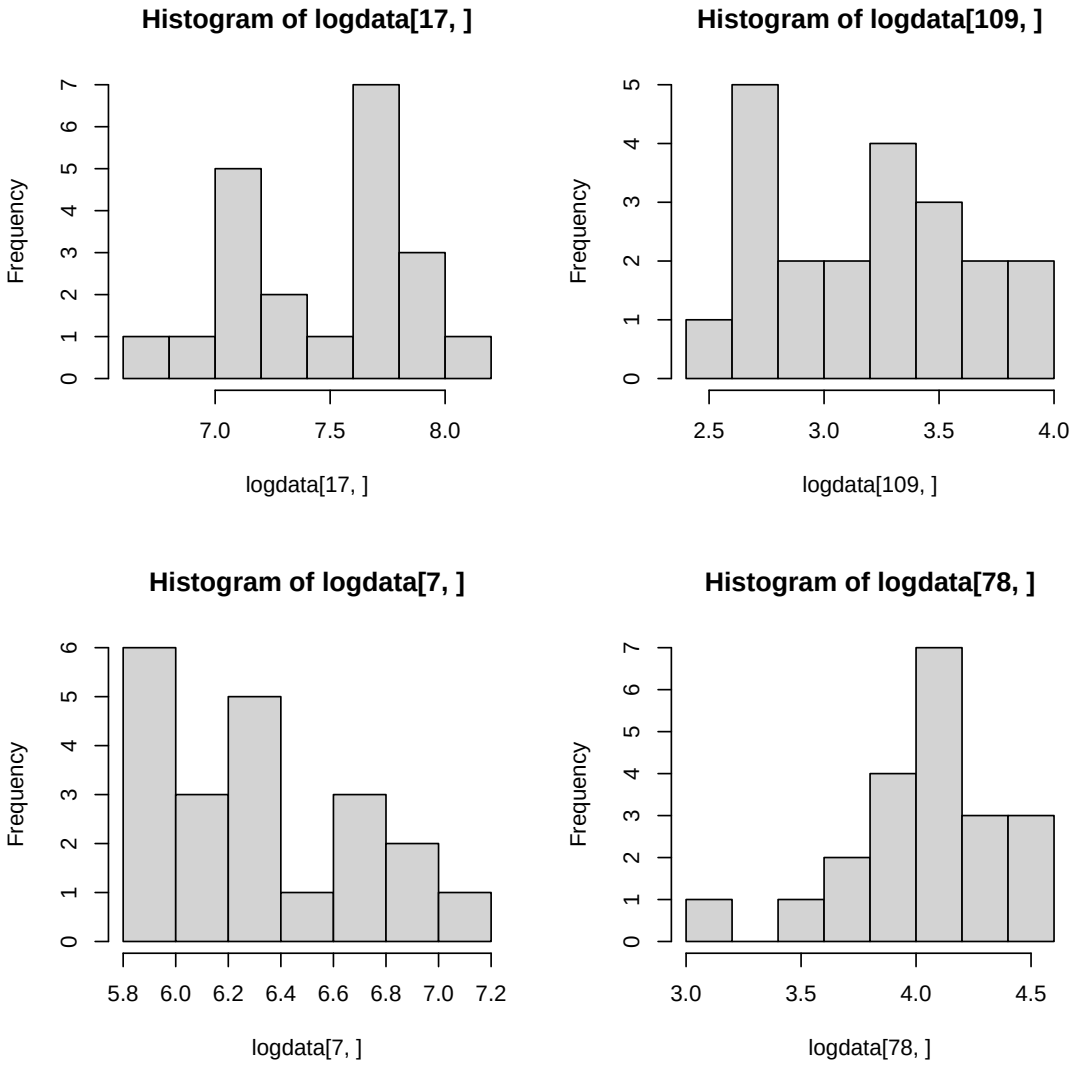

```
range(data)
## [1]      0 136998
par(mfrow=c(1,1))
hist(data)
```



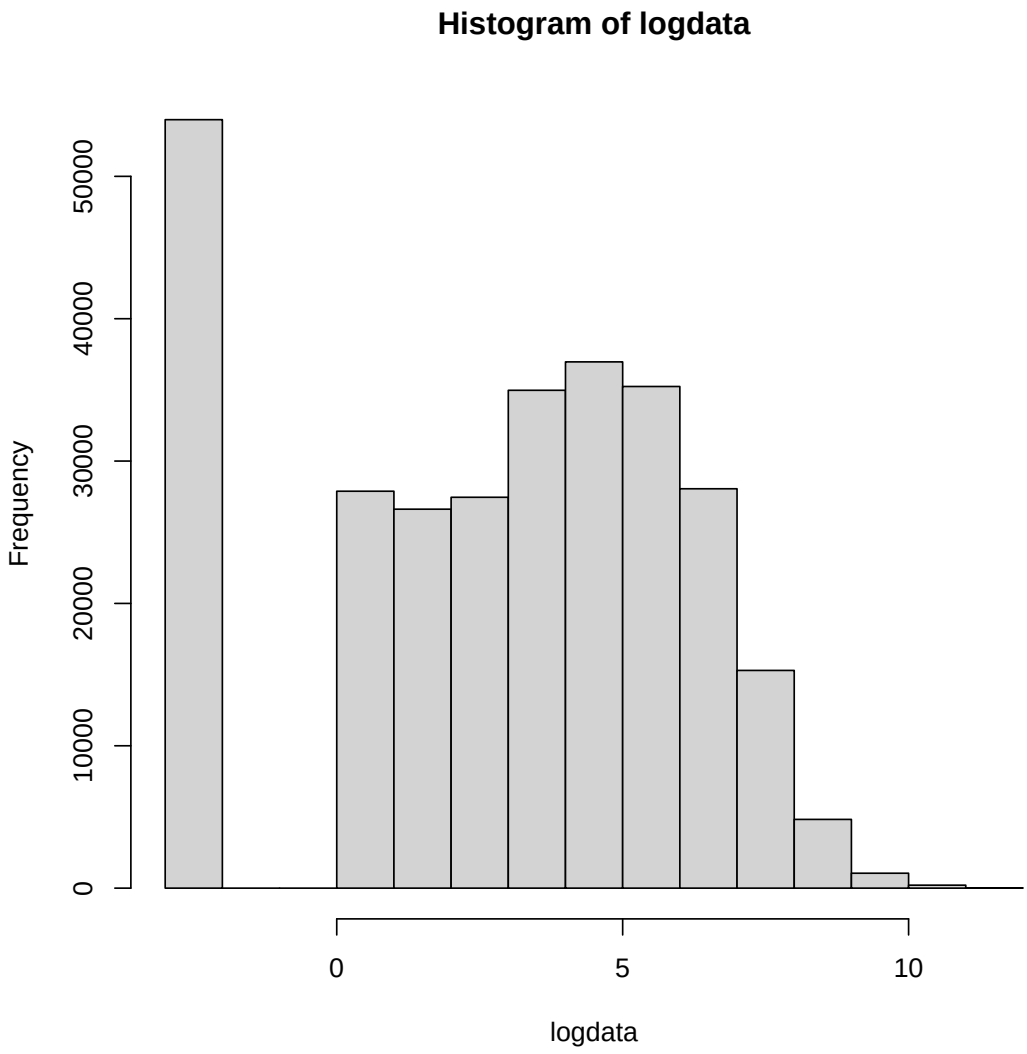

```
table(data==0)
##
##   FALSE    TRUE
## 238591 528665
```

On ne peut pas supposer que chaque variable peut être modéliser par une loi normale. Les données varient entre 0 et un nombre élevé entier. Toutes les valeurs sont entières. On peut penser à une loi discrètre type binomiale ou Poisson, voir négative binomiale.

```r
table(rowMeans(data)==0) ## uniquement 13932 variables non nulles
##
## FALSE  TRUE
## 13932 22604
## on log les données
logdata <- log(data[rowMeans(data)!=0,]+0.1)
par(mfrow=c(2,2))
hist(logdata[17,])
hist(logdata[109,])
hist(logdata[7,])
hist(logdata[78,])
```




```r
range(logdata)
## [1] -2.302585 11.827722
par(mfrow=c(1,1))
hist(logdata)
```

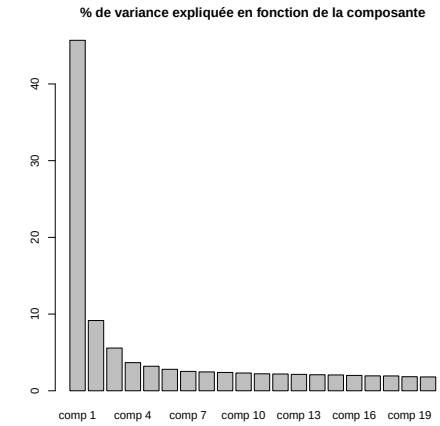


On peut proposer raisonnablement de modéliser le log des données par un loi normale, même si cela est discutable.

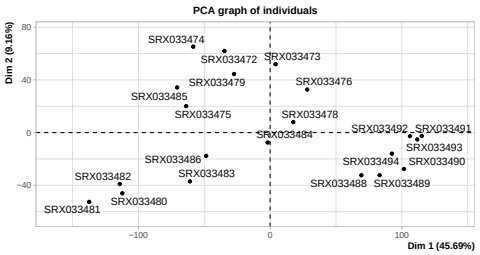
(c) Réaliser une ACP et interpréter les résultats de cette visualisation.

Solution:

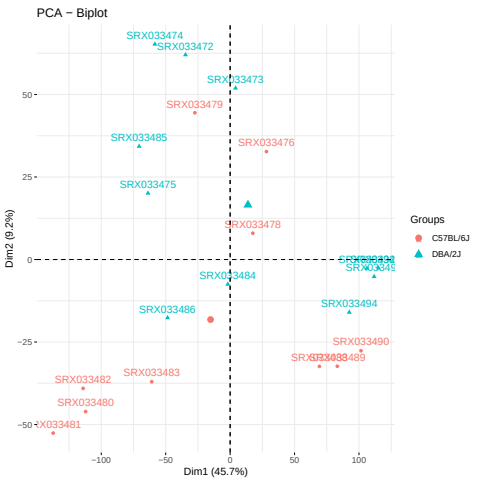
```
library(FactoMineR)
resPCA <- PCA(t(logdata),scale.unit = TRUE,graph=FALSE)
barplot(resPCA$eig[,2],main="% de variance expliquée en fonction de la composante
```



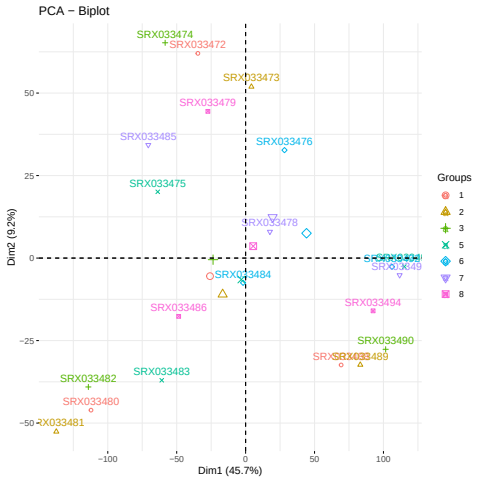

```
## on garde les 2 premières composantes
par(mfrow=c(1,2))
plot(resPCA,choix = "ind")
```



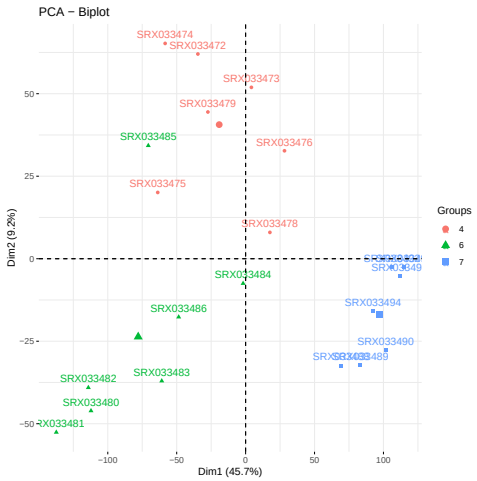

```
#plot(resPCA,choix = "var") graphique des variable peu lisible...
library(factoextra,verbose = FALSE)
par(mfrow=c(1,2))
lignee <- factor(phenotype$strain)
lane.number <- factor(phenotype$lane.number)
exp.number <- factor(phenotype$experiment.number)
fviz_pca_biplot(resPCA,habillage = lignee, invisible = "var")
```



```
fviz_pca_biplot(resPCA,habillage = lane.number,invisible = "var")
```




```
fviz_pca_biplot(resPCA,habillage = exp.number,invisible = "var")
```



On voit que les échantillons biologiques ne se regroupent pas totalement en fonction de la variable "lignée" mais également en fonction du numéro de l'expérience "expérimental number".

Exercice n°4 : Abondance de bactéries

(a) Télécharger et installer la package `phyloseq`.

```
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("phyloseq")
```

(b) Extraire les données suivantes et comprendre la structure des données à l'aide des commandes suivantes.

```
library(phyloseq)
data("GlobalPatterns", package = "phyloseq")
GPOTUs = as.matrix(t(phyloseq::otu_table(GlobalPatterns)))
#GPOTUs[1:4, 6:13]
#help(GlobalPatterns)
```

Solution:

```
dim(GPOTUs)
## [1]    26 19216
str(GPOTUs)
## Formal class 'otu_table' [package "phyloseq"] with 2 slots
##   ..@ .Data        : num [1:26, 1:19216] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:26] "CL3" "CC1" "SV1" "M31Fcsw" ...
##   .. .. ..$ : chr [1:19216] "549322" "522457" "951" "244423" ...
##   ..@ taxa_are_rows: logi FALSE
##   ..$ dim     : int [1:2] 26 19216
##   ..$ dimnames:List of 2
##   .. ..$ : chr [1:26] "CL3" "CC1" "SV1" "M31Fcsw" ...
##   .. ..$ : chr [1:19216] "549322" "522457" "951" "244423" ...
GPOTUs[1:4, 6:13]
## OTU Table:          [8 taxa and 4 samples]
##                      taxa are columns
##         246140 143239 244960 255340 144887 141782 215972 31759
## CL3          0      7      0    153      3      9      0     0
## CC1          0      1      0    194      5     35      3     1
## SV1          0      0      0      0      0      0      0     0
## M31Fcsw      0      0      0      0      0      0      0     0
rownames(GPOTUs)
##  [1] "CL3"      "CC1"      "SV1"      "M31Fcsw"  "M11Fcsw"  "M31Plmr"
##  [7] "M11Plmr"  "F21Plmr"  "M31Tong"  "M11Tong"  "LMEpi24M" "SLEpi20M"
## [13] "AQC1cm"   "AQC4cm"   "AQC7cm"   "NP2"      "NP3"      "NP5"
## [19] "TRRsed1"  "TRRsed2"  "TRRsed3"  "TS28"     "TS29"     "Even1"
## [25] "Even2"    "Even3"
min(GPOTUs);max(GPOTUs)
## [1] 0
## [1] 1177685
table(colSums(GPOTUs)==0)
##
## FALSE  TRUE
## 18988   228
```

On a 19216 variables possibles (19219 bactéries, ici détecté à l'aide de l'ARN 16S, un gène présent uniquement chez les bactéries) mesurées pour 26 échantillons différents (environnement différents).

(c) Décrire chaque variable du jeu de données et proposer un modèle probabiliste possible pour chaque variable (sans aller jusqu'à tester l'adéquation des données à une loi donnée).

Solution:

```
logGPOTUs <- log(GPOTUs+1)
```

Même problème que pour le cas précédent, on est face à des données de comptages. Il n'est pas possible de les modéliser à l'aide d'une loi normale. On peut penser à des lois discrètes comme la loi binomiale ou la loi de Poisson. On peut prendre les log des données.

(d) Réaliser une ACP et interpréter les résultats de cette visualisation.

Solution:

```
## on a bcp de variables, on travaille uniquement
## a partir des bactéries les plus "variables"
myvar <- apply(GPOTUs,2,var)
resPCA <- PCA(log(GPOTUs[,order(myvar,decreasing = TRUE)[1:400]]+1),graph=FALSE)
par(mfrow=c(1,1))
fviz_pca_ind(resPCA)
```