Compte-rendu Statistiques – TP2

Analyse en composantes principales et apprentissage

Université de Tours
Moreau Clément

05 January 2021

Les iris de Fischer

On considère le fichier `iris.csv` sur Celene répertoriant 150 individus fleurs d'iris. On donne la description suivante des colonnes:

Colonne	Description	Value
`sepal_length`	Longueur des sépales	Int
`sepal_width`	Largeur des sépales	Int
`petal_length`	Longueur des pétales	Int
`petal_width`	Largeur des pétales	Int
`species`	Espèce d'iris	{Versicolor, Virginica, Setosa}

Figure 1: Les iris de Fischer

1. Statistiques descriptives

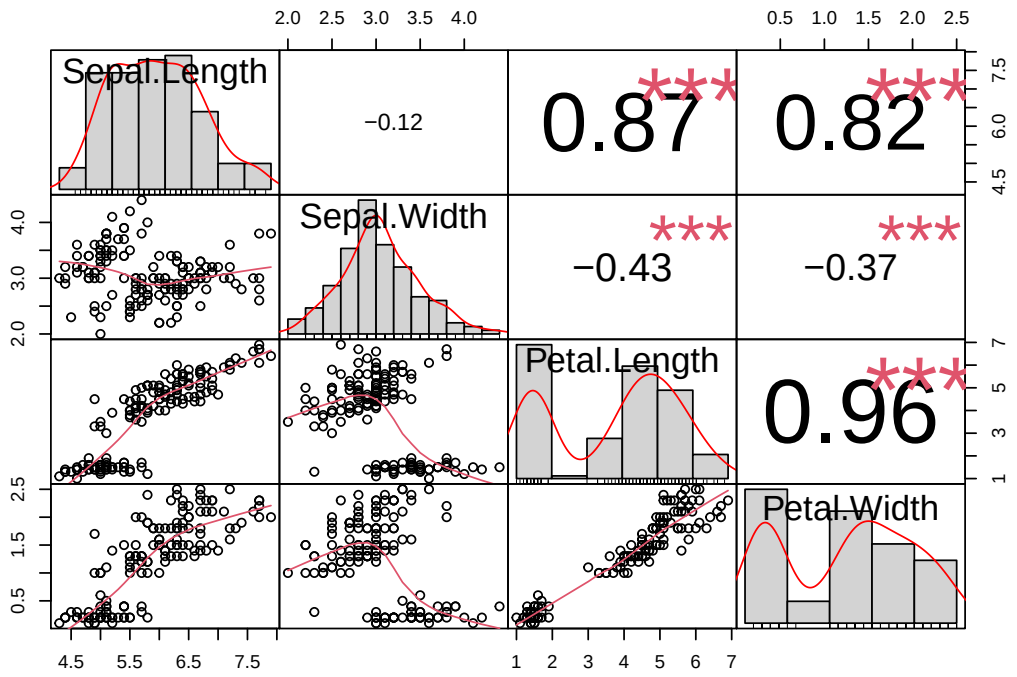
   (a) Proposer une analyse préliminaire par statistiques descriptives du jeu de données `iris`. Votre analyse[1] devra contenir notamment:

   - Distribution de chaque variable puis analyses synthétiques agrégées par espèce.
   - Corrélation entre les variables.

   On donne le graphique des corrélations et distributions suivant:

[1] Vous pourrez vous aider la fonction `chart.Correlation` de la librairie `PerformanceAnalytics`.

```
library(PerformanceAnalytics)
chart.Correlation(iris[, -5])
```

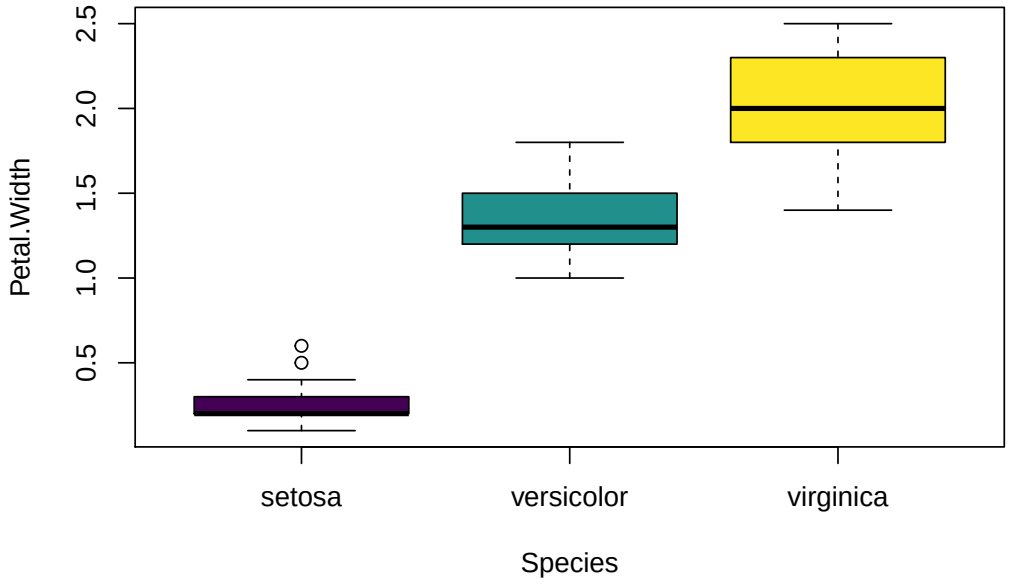


On constate sur ce graphique que les variables Petal.Length et Petal.Width sont extrêmement corrélées. À l'inverse, Sepal.Length et Sepal.Width sont très faiblement corrélées.

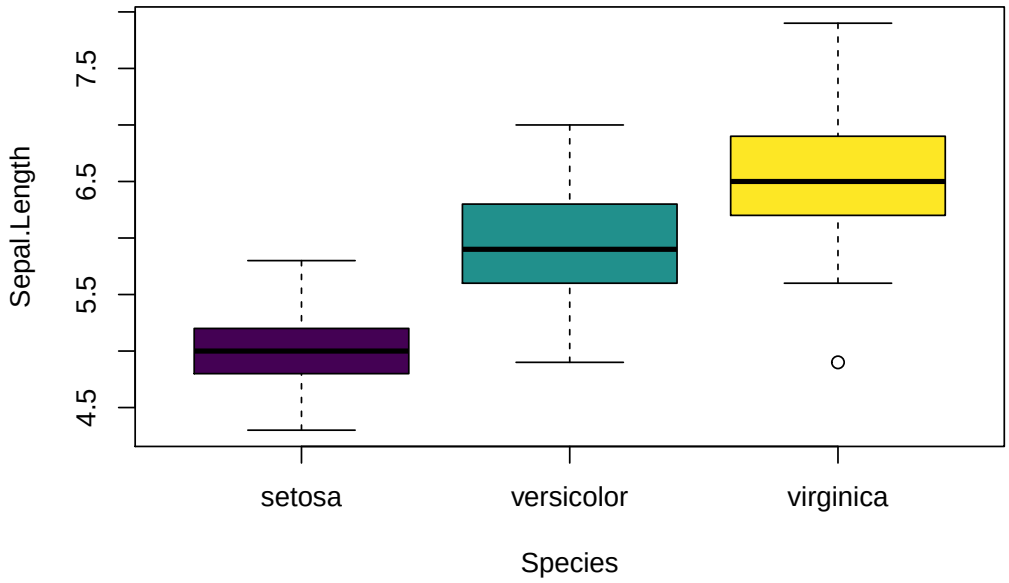
On remarque également certaines ruptures dans les distributions notemment au niveau des variables "Petal". Cette anomalie au niveau du premier intervalle peut laisser penser qu'elle est produite par une classe d'individus. Pour cela, on analyse les variables pour chaque classe, on retient pour l'analyse les variables les moins corrélées entre elles, soit:

- *Petal.Width*
- *Sepal.Width*
- *Petal.Length*

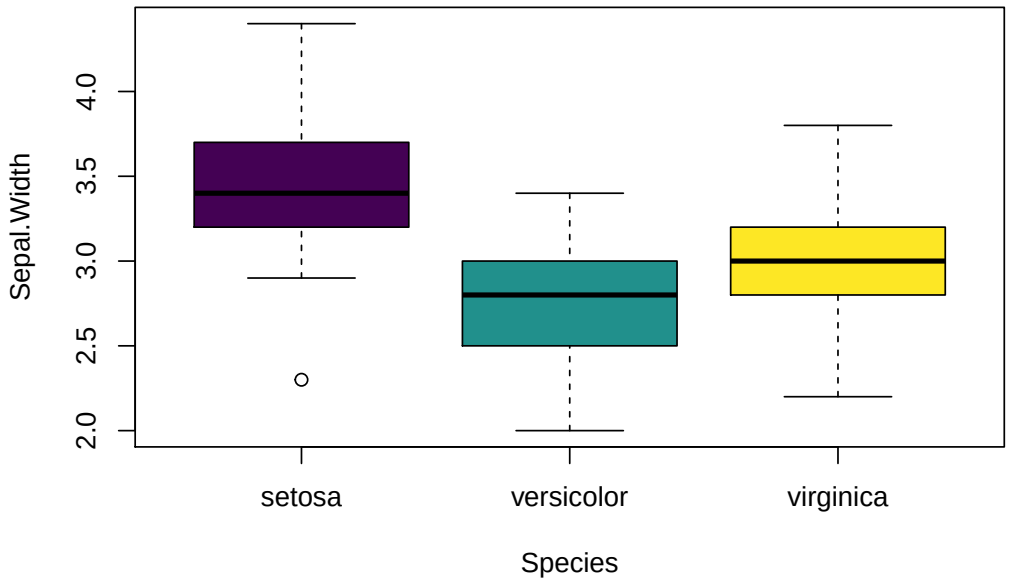
```
library(viridis)
boxplot(Petal.Width ~ Species, data = iris, col = viridis(3))
```




```
boxplot(Sepal.Length ~ Species, data = iris, col = viridis(3))
```




```
boxplot(Sepal.Width ~ Species, data = iris, col = viridis(3))
```



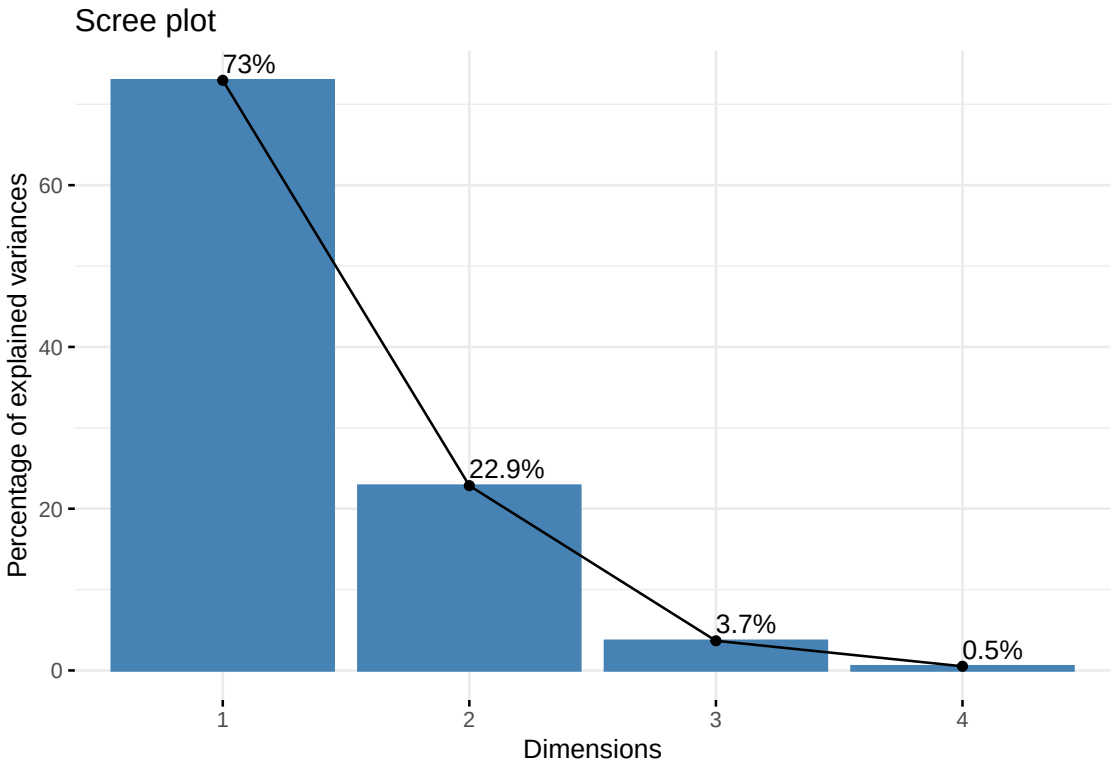
Ces trois graphiques montrent que la variété Setosa est très différente des deux autres pour chacun des trois critères étudiés. Les variétés Versicolor et Virginica semblent plus semblables entre elles, hormis au niveau de la variable ***`Petal.Width`****.*

(b) Sur la base de ces analyses, quelles variables vous semblent pertinentes pour l'ACP ?

On peut penser que les variables ***`Petal.Width`*** *et* ***`Sepal.Width`*** *sont les plus pertinentes pour l'ACP car elles discrimiment fortement les individus et sont plutôt faiblement corrélées entre elles (-0.37).*

2. Calculer les valeurs propres de la matrice des données `iris`. Combien d'axes proposez vous de retenir pour l'ACP ? Détaillez votre réponse.

```r
library("FactoMineR")
library("factoextra")
iris.pca <- PCA(iris[, -5], scale = TRUE, graph = FALSE)
fviz_eig(iris.pca, addlabels = TRUE)
```

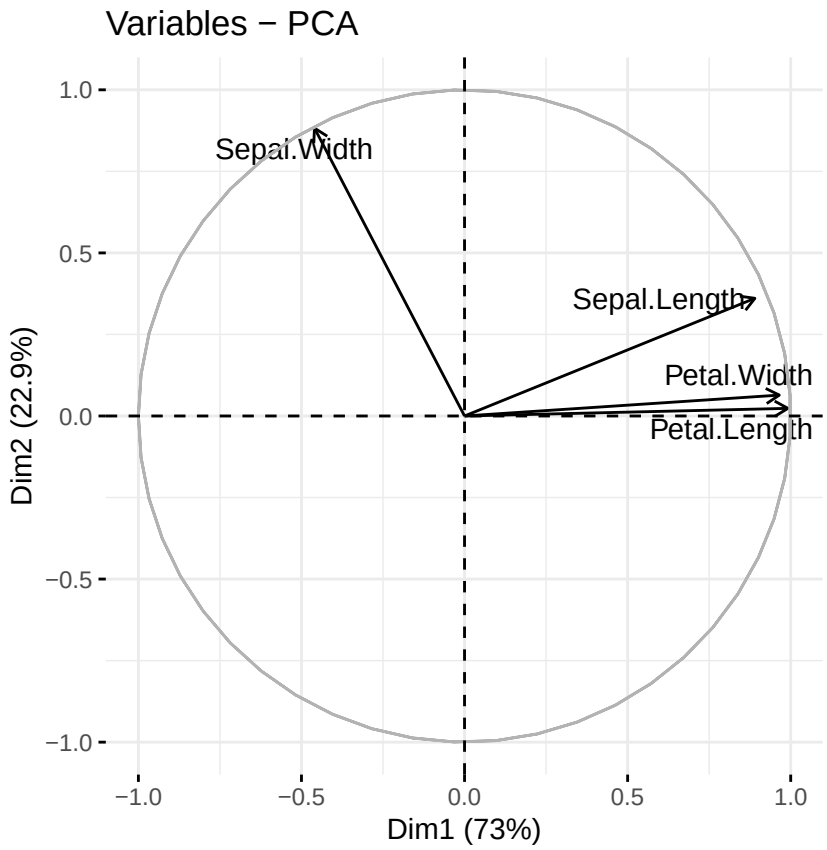


On explique 95.9% de la variance totale à l'aide des deux premiers axes factoriels. On retient les deux axes pour l'ACP.

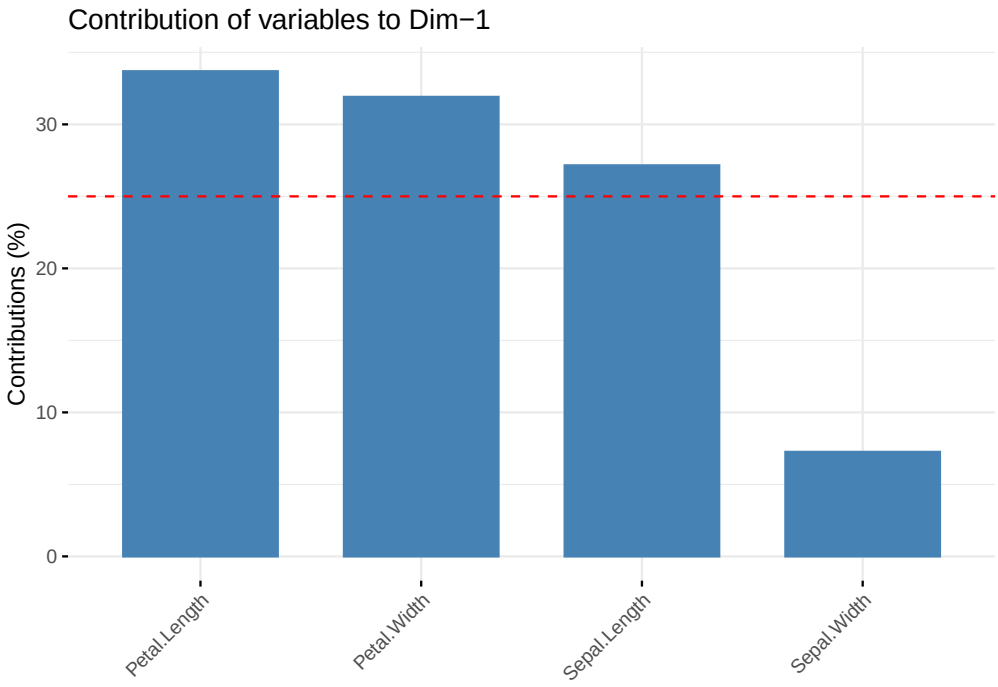
3. Analyse des variables

   (a) Dresser le cercle des corrélations de l'ACP. Commentez la qualité de représentation et la contribution de chaque variable quant aux axes retenus.

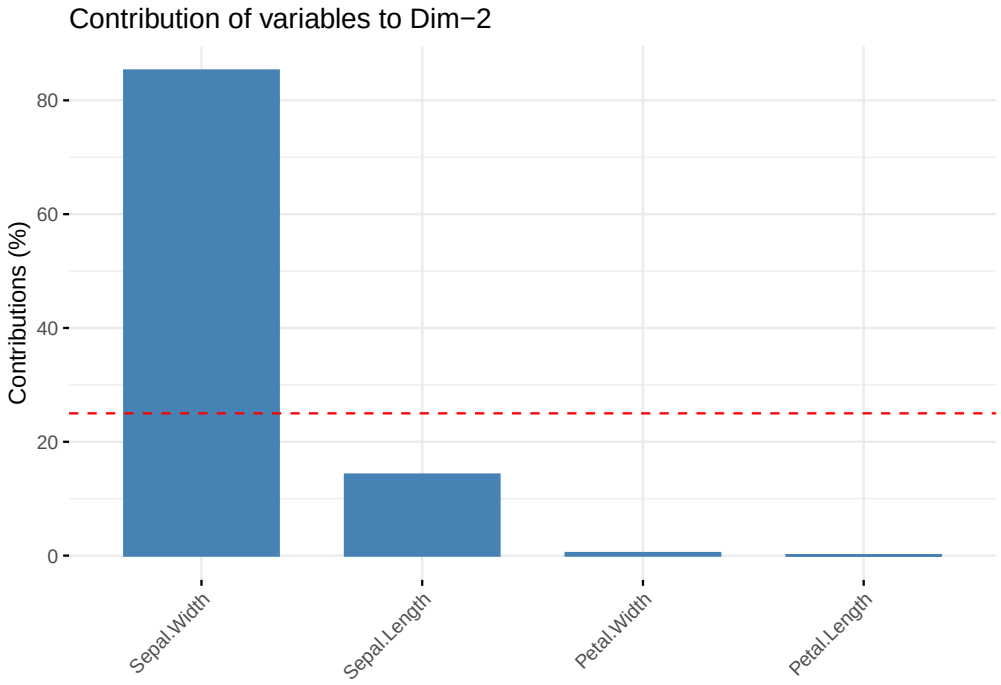
```
fviz_pca_var(iris.pca, repel = TRUE)
```




```
fviz_contrib(iris.pca, choice = "var", axes = 1)
```




```
fviz_contrib(iris.pca, choice = "var", axes = 2)
```



Les précédents graphiques montrent à la fois la très bonne contribution et la très bonne représentation des variables au sein du plan factoriel. Pour la contribution, on remarque sur le cercle des corrélations que ce sont les variables ***`Petal.Length`*** *et* ***`Petal.Width`*** *qui sont le plus corrélées à l'axe 1. Cette analyse est renforcée par le graphique ci-dessus qui montre une contribution à la dimension 1 supérieure à la moyenne pour ces deux variables tandis que c'est* ***`Sepal.Width`*** *qui contribue le plus sur l'axe de dimesion 2.*

```
round(get_pca_var(iris.pca)$cos2, 2)
```

```
##              Dim.1 Dim.2 Dim.3 Dim.4
## Sepal.Length  0.79  0.13  0.08  0.00
## Sepal.Width   0.21  0.78  0.01  0.00
## Petal.Length  0.98  0.00  0.00  0.01
## Petal.Width   0.93  0.00  0.06  0.01
```

Concernant la qualité de représentation, la norme des vecteurs sur le cercle des corrélations est très proche de 1 pour toutes les variables ce qui indique une très bonne représentation dans le plan factoriel. Cette hypothèse est renforcée par les résultats ci-dessus qui exhibent un cos2 cumulé excellent (> 95%) pour l'ensemble des variables ce qui indique que les variables sont effectivement très bien représentées par les deux axes.

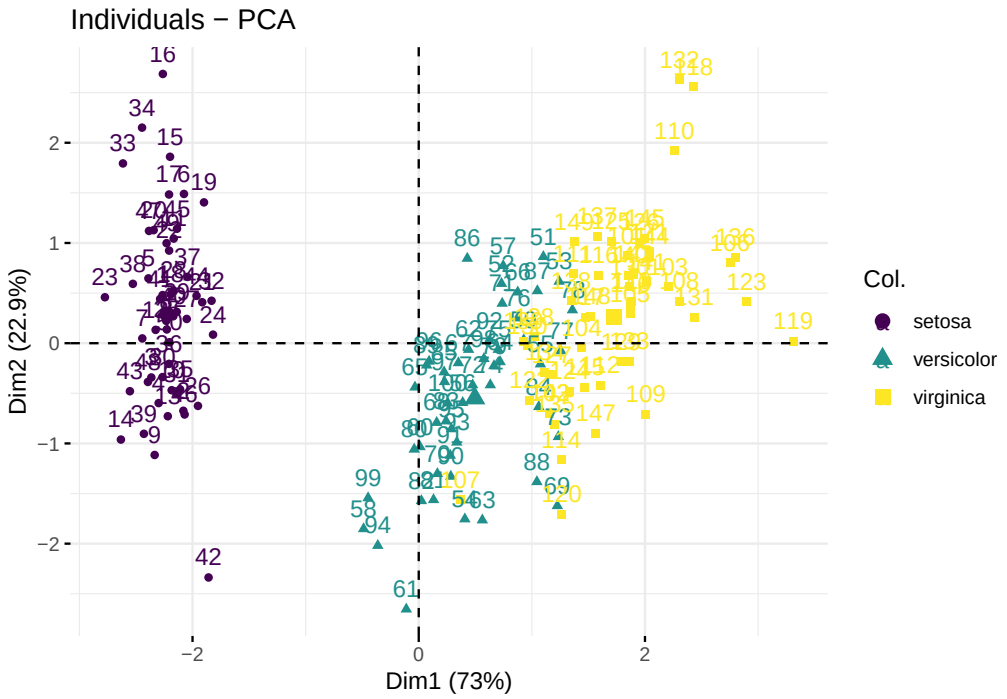
(b) Interpréter la signification des axes retenus. Vous pourrez vous aider de la contribution des variables aux axes factoriels.

Comme relevé question 1. (b) il semble que l'axe 1 représente une combinaison linéaire de ***`Petal.Width`*** *et* ***`Petal.Length`****. L'axe 2 représente principalement la varibale* ***`Sepal.Width`*** *et, en moindre mesure, 'Sepal.Length. Ainsi, l'axe de dimension 1 peut être interprêté comme celui des Pétales et l'axe de dimension 2 comme celui des Sépales.*

4. Analyse des individus

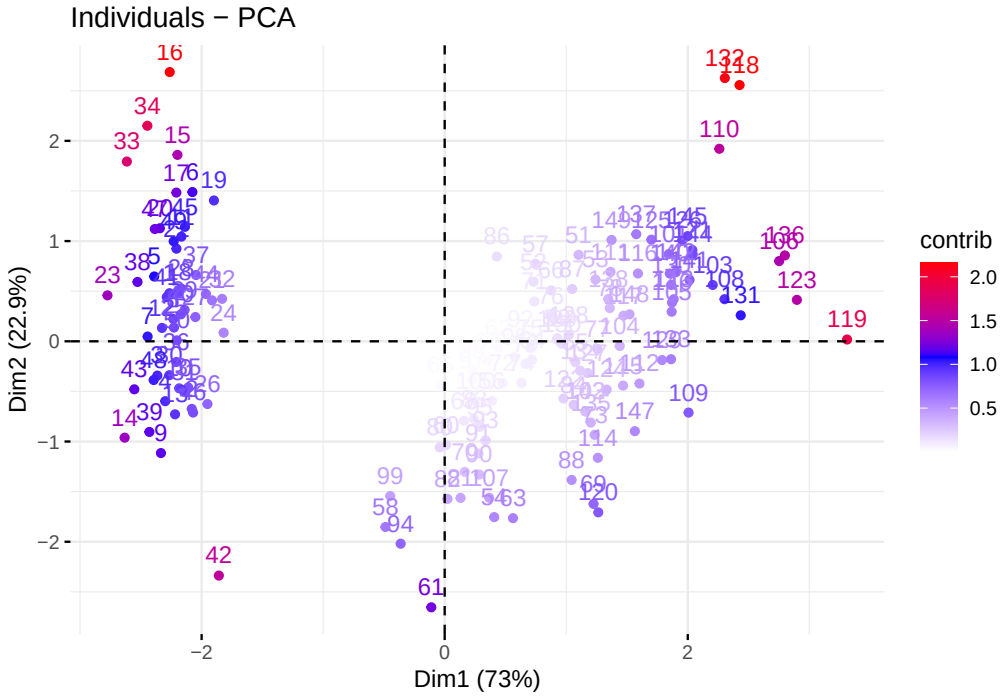
(a) Présenter la projection des indivus dans le plan factoriel. Vous colorerez dans un premier temps les points en fonction de l'espèce d'iris.

```
fviz_pca_ind(iris.pca, col.ind = iris$Species, palette = viridis(3))
```



(b) Colorer les individus en fonction de leur contribution aux axes factoriels. Que remarquez-vous ? Pouvez l'expliquer ?

```
fviz_pca_ind(iris.pca, col.ind = "contrib", gradient.cols = c("White", "Blue", "Red"))
```

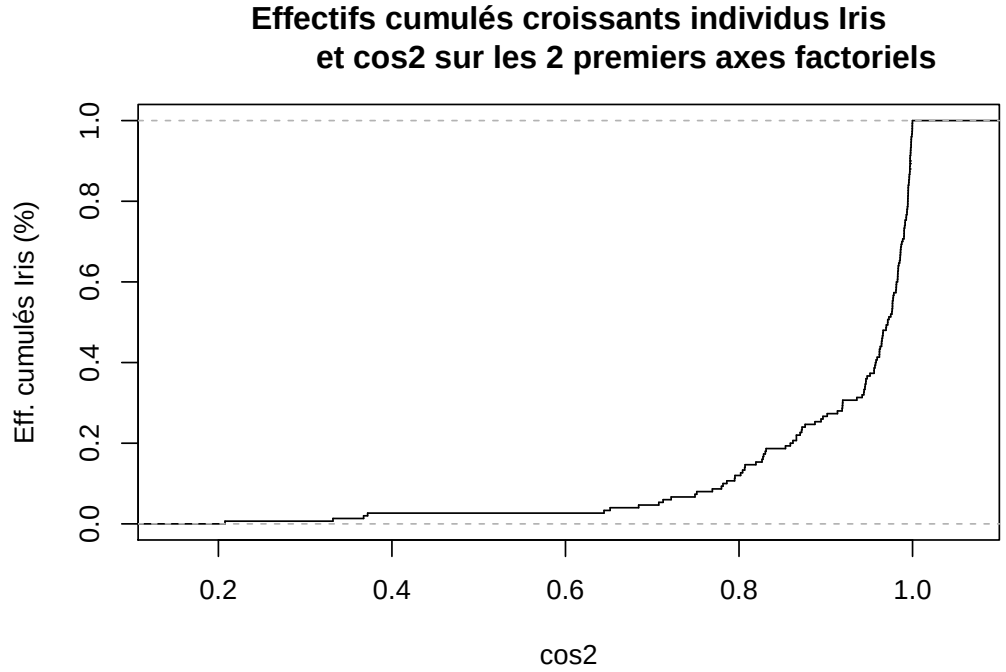


Les individus qui contribuent le moins aux axes sont principalement ceux près du centre du repère. Cette observation est en accord avec la formule de la contribution des

individus. Plus le projeté de l'individu sur les axes factoriels est proche de l'origine, plus la contribution est faible.

(c) Commenter la qualité de représentation des individus.

```
plot(ecdf(get_pca_ind(iris.pca)$cos2[,1] + get_pca_ind(iris.pca)$cos2[,2]),
     verticals = TRUE,
     do.points = FALSE,
     ylab = "Eff. cumulés Iris (%)",
     xlab = "cos2",
     main = "Effectifs cumulés croissants individus Iris
             et cos2 sur les 2 premiers axes factoriels")
```



La représentation des individus est très bonne. On peut voir sur la courbe des effectifs cumulés croissants que moins de 20% des individus ont un cos2 cumulé sur les deux axes factoriels < 0.8.

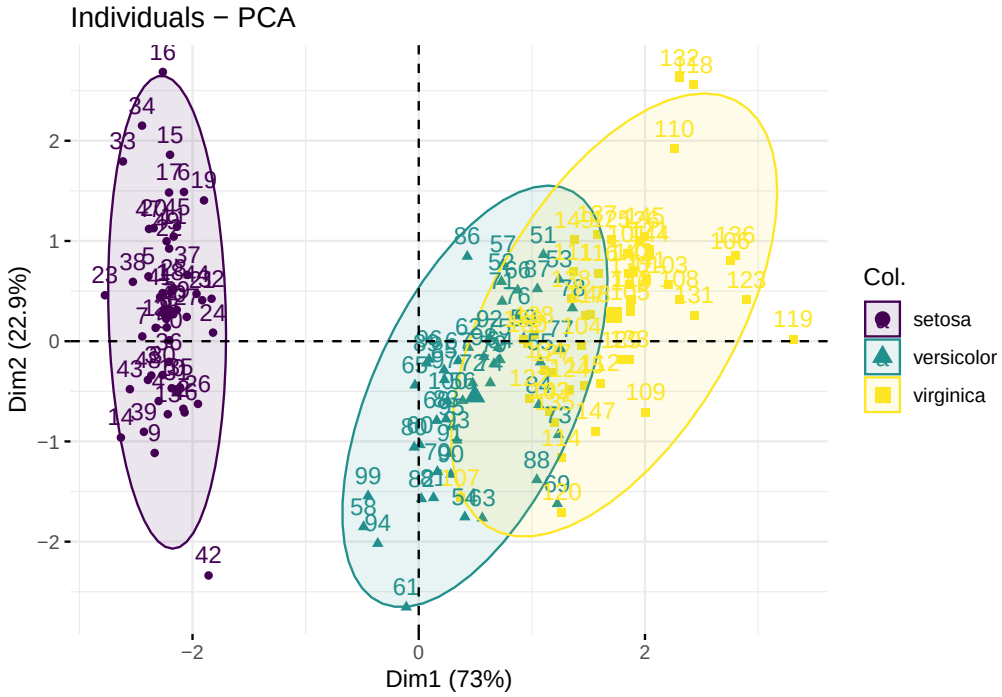
5. Apprentissage statistique

L'option `addEllipses=TRUE` de la fonction `fviz_pca_ind` permet de dessiner l'ellipse de confiance (covariance ellipse error) à 95%.

(a) Sous quelle condition la définition d'ellipses de confiance est-elle valable ? Est-ce le cas selon vous-ici ? Pourquoi ?

La définition d'ellipses de confiance est valable si la loi suivie par l'échantillon analysé est de type Normale. Plus simplement, on veillera à ce que la forme des nuages des individus soit convexe. Cette condition est ici vérifiée.

```
fviz_pca_ind(iris.pca, col.ind = iris$Species, addEllipses=TRUE, palette = viridis(3))
```



Une deuxième exigence pourrait être que les ellipses définies ne se superposent pas afin d'assurer une délimitation claire des frontières entre les classes d'individus. Cette condition ici n'est pas vérifiée ; les ellipses des espèces Versicolor et Virginica se chevauchent.

(b) Proposer un algorithme permettant de classifier automatiquement une nouvelle iris inconnue et ainsi déterminer son espèce. Vous évoquerez les limites de votre approche et possibilités pour pallier à ces effets.

On peut imaginer la méthode de classification suivante:

```
1. On calcule les coordonnées x, y du nouvel individu dans le plan factoriel.
2. On regarde où se situent les coordonnées.
- Si x, y sont dans une et une seule ellipse:
Prédire la classe de l'ellipse.
- Si x, y appartiennent à plusieurs ellipse:
Prédire la classe du centroid le plus proche.
- Si x, y n'appartiennet à aucune ellipse:
Prédire la classe de la frontière la plus proche.
```

Il existe de nombreux inconvénients à cette méthode naïve de classification, notamment dans le cas où l'individu tombe à l'intersection de plusieurs ellipses. Dans ce cas, il est difficile de pouvoir affirmer avec certitude la classe de l'individu. Une méthode pour pallier à cet effet pourrait être de faire figurer une probabilité d'appartenance aux classes prédites plutôt qu'une classe unique.

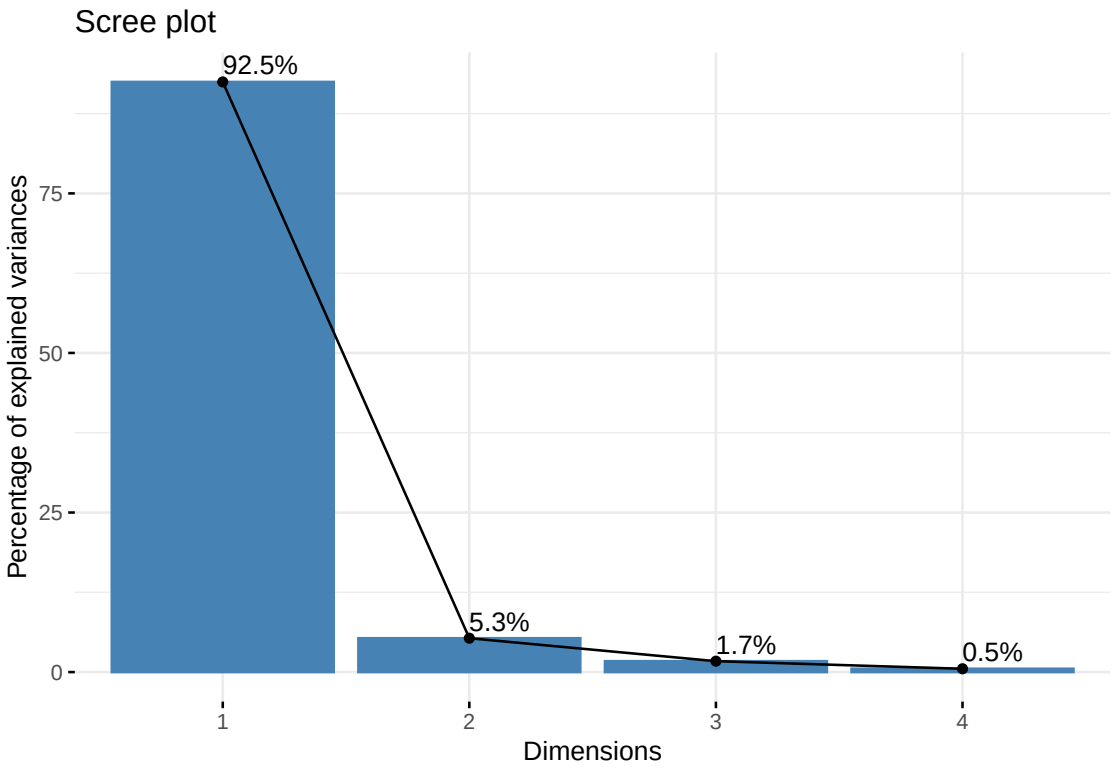
6. Reprendre l'analyse du jeu de données `iris` mais en effectuant ici une ACP **non réduite**. On appliquera pour ça l'option `scale = FALSE` lors de l'exécution de la fonction `PCA`.

   Que remarquez vous ? Quelle méthode semble finalement donner les meilleurs résultats ici ? Expliquer ces résultats.

```
iris$Sepal.Length <- iris$Sepal.Length - mean(iris$Sepal.Length)
iris$Petal.Length <- iris$Petal.Length - mean(iris$Petal.Length)
iris$Sepal.Width <- iris$Sepal.Width - mean(iris$Sepal.Width)
iris$Petal.Width <- iris$Petal.Width - mean(iris$Petal.Width)

iris.pca <- PCA(iris[, -5], scale = FALSE, graph = FALSE)

fviz_eig(iris.pca, addlabels = TRUE)
```



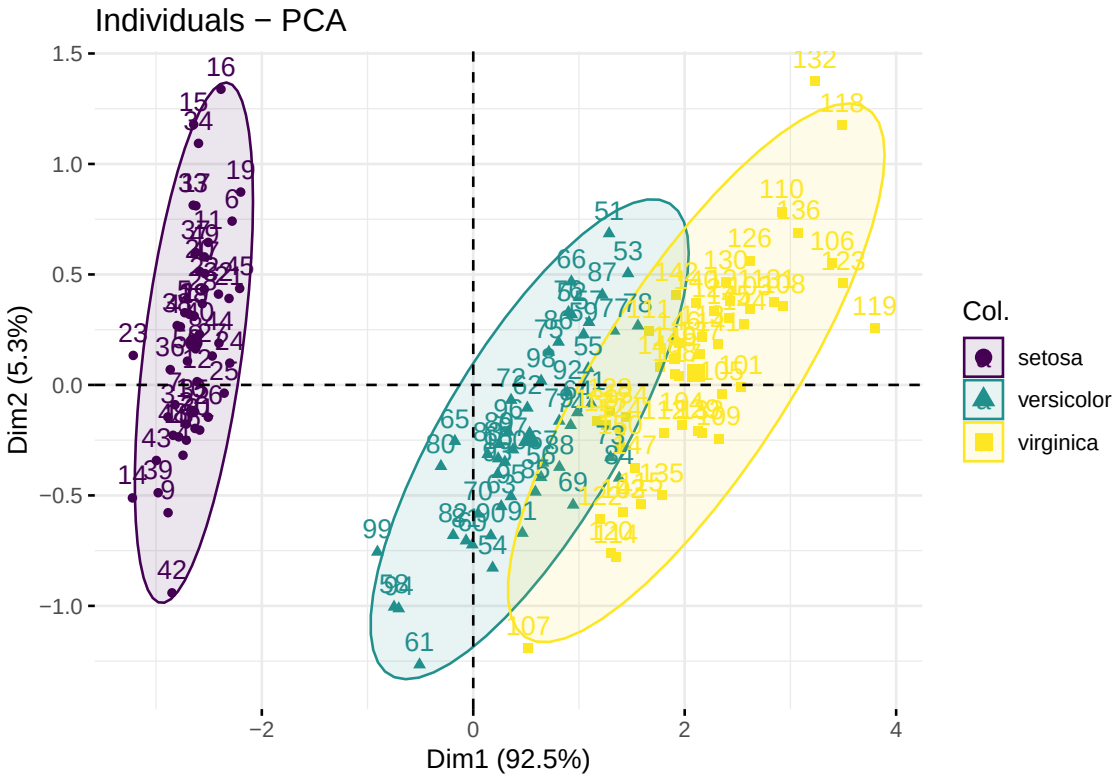
On remarque que l'ACP non réduite permet ici de représenter mieux l'espace à l'aide des deux premiers axes factoriels (97.8% vs 95.9%). Pour comprendre ce fait, on peut regarder la l'écart-type (ou la variance) des variables.

```
var(iris[,-5])
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    0.6856935  -0.0424340    1.2743154   0.5162707
## Sepal.Width    -0.0424340   0.1899794   -0.3296564  -0.1216394
## Petal.Length    1.2743154  -0.3296564    3.1162779   1.2956094
## Petal.Width     0.5162707  -0.1216394    1.2956094   0.5810063
```

On distingue dans la matrice des variances-covariance que la variable ayant la plus forte variance est ***`Petal.Length`****. Or, on a vu précédemment que cette variable était une contributrice majeure à l'axe factoriel de la dimension 1. Le fait de pratiquer une ACP non normée permet de donner un poids plus fort à 'Petal.Length et ainsi tirer l'effet de la réduction selon l'axe factoriel Dim1.*

```
fviz_pca_ind(iris.pca, col.ind = iris$Species,
addEllipses=TRUE, palette = viridis(3))
```



On voit sur le projeté dans le plan factoriel que les individus sont plus "serrés" selon l'axe factoriel Dim1. Il y'a moins de chevauchement entre les variétés Versicolor et Virginica. On peut dire ici que l'ACP non normée est de meilleure qualité.

Sommeil des mammifères

On considère le fichier `sleep.csv` sur Celene répertoriant les données de 70 espèces de mammifères concernant leur sommeil et quelques autres caractéristiques. On donne la description suivante des colonnes:

Colonne	Description	Value
`name`	Nom français vernaculaire de l'animal	String
`genus`	Genre, subdivion de la classification biologique	String
`vore`	Régime alimentaire de l'animal	String
`order`	Ordre, subdivion de la classification biologique	String
`sleep_total`	Durée (en h) de sommeil sur une journée	Double
`sleep_rem`	Durée (en h) de sommeil paradoxal	Double
`awake`	Durée (en h) où l'animal est éveillé	Double
`brain_wt`	Masse (en kg) moyenne du cerveau de l'animal	Double
`body_wt`	Masse (en kg) totale moyenne de l'animal	Double
`brain_body_ratio`	Ratio masse cerveau, masse totale $\frac{\texttt{brain\_wt}}{\texttt{body\_wt}}$	Double
`gest_day`	Période de gestation moyenne de l'animal	Int

1. Statistiques descriptives

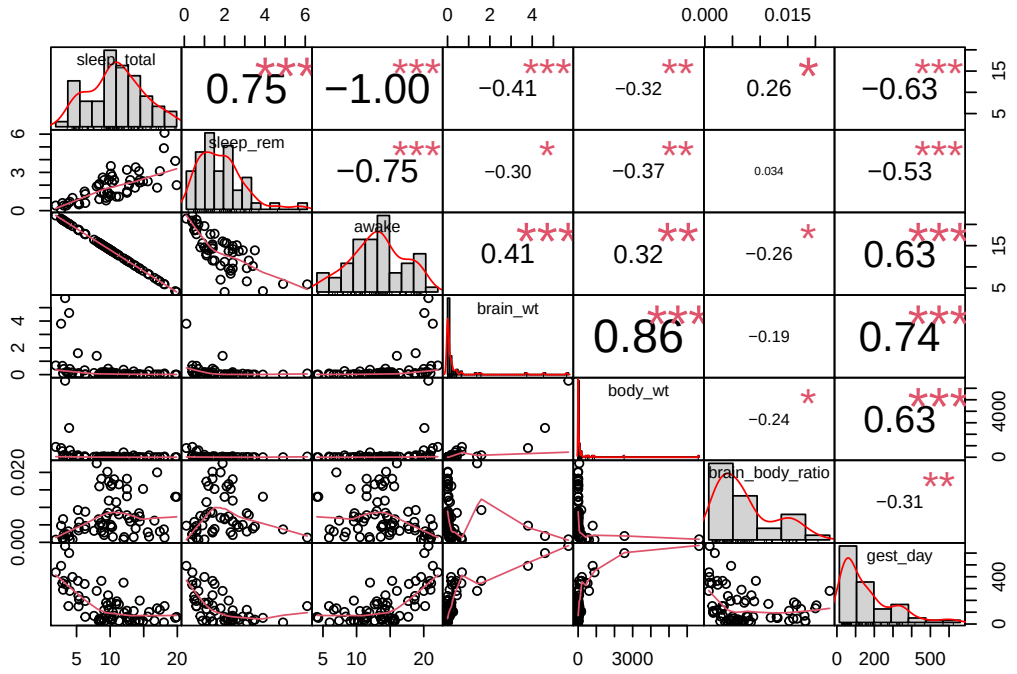
   (a) Proposer une analyse préliminaire par statistiques descriptives du jeu de données `sleep`. Votre analyse devra contenir notamment:

   - Distribution de chaque variable puis analyses synthétiques agrégées selon différentes

variables qualitatives.

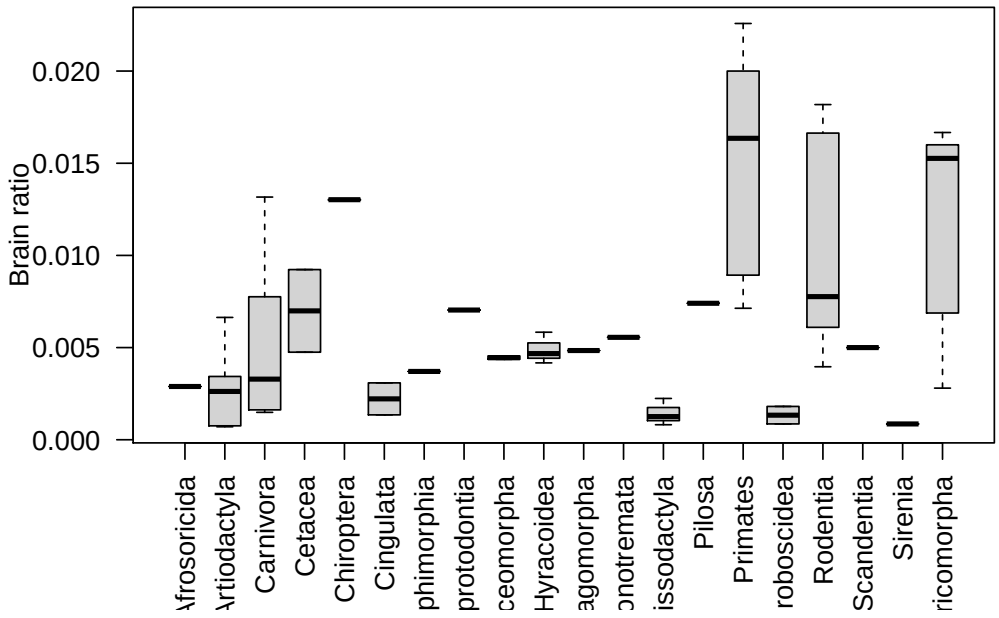
- Corrélation entre les variables.

```
chart.Correlation(sleep[, 5:ncol(sleep)])
```



Les temps de sommeil sont tous très corrélés sans grande surprise et globalement distribués selon une loi normale. Les variables ***brain_wt****,* ***body_wt*** *et* ***gest_day*** *sont également assez corrélées (> 0.6). Une dernière variable assez atypique,* ***brain_body_ratio****, n'est corrélée à aucune autre. La distribution ne ressemble pas un modèle précis de loi, on peut chercher à l'analyser en fonction de variables catégorielles. On commence par analyser selon l'ordre.*

```
boxplot(sleep$brain_body_ratio ~ sleep$order,
        xlab = "",
        ylab = "Brain ratio",
        las= 2)
```



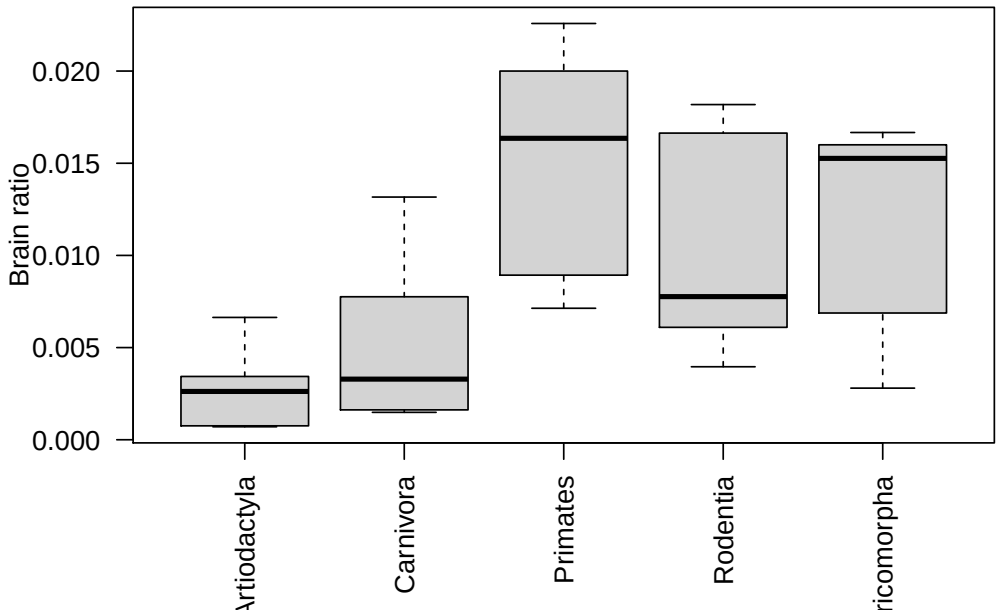
La variable order contient de nombreuses valeurs différentes et beaucoup d'entre elles n'enregistre qu'une unique occurence. On peut s'en persuader par la commande:

```
table(sleep$order)
```

```
##
##    Afrosoricida    Artiodactyla       Carnivora         Cetacea      Chiroptera
##               1               6              13               2               2
##       Cingulata Didelphimorphia   Diprotodontia  Erinaceomorpha      Hyracoidea
##               2               1               1               2               3
##      Lagomorpha     Monotremata  Perissodactyla          Pilosa        Primates
##               1               1               3               1              10
##     Proboscidea        Rodentia      Scandentia         Sirenia    Soricomorpha
##               2              12               1               1               5
```

Pour alléger la visualisation et avoir une puissance statistique suffisante, on retient uniquement les ordres avec au moins 5 individus.

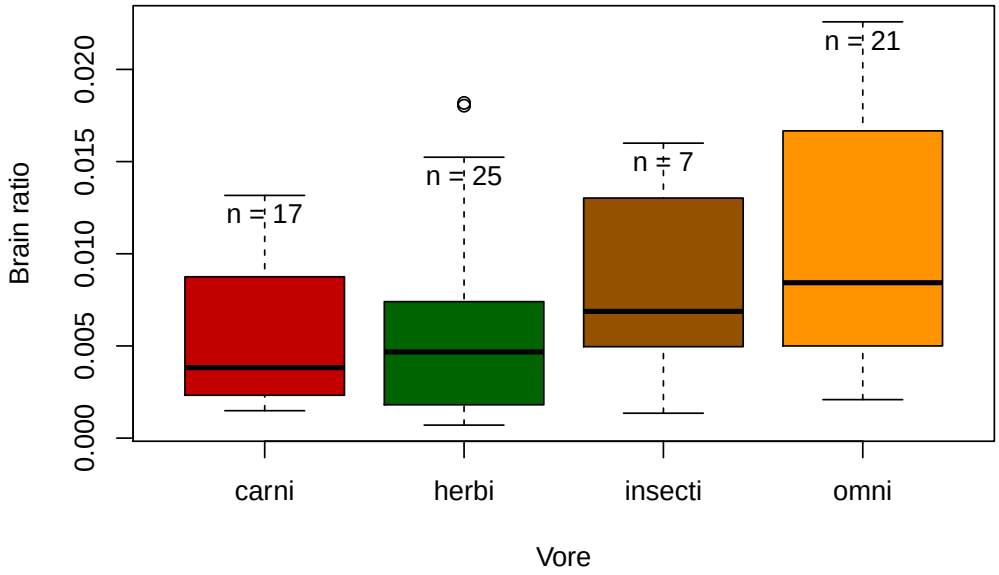
```
X <- table(sleep$order)
names_order <- names(X[X >= 5])
T <- subset(sleep, order %in% names_order)
boxplot(T$brain_body_ratio ~ T$order,
        xlab = "",
        ylab = "Brain ratio",
        las= 2)
```



On constate ici que l'ordre des primates possède le rapport masse corps-masse cerveau le plus élévé. Nous sommes suivis des Soricomorpha (musareignes, taupes) puis des Rodentia (souris, rats).

On peut faire une analyse similaire selon le régime alimentaire des animaux.

```
b <- boxplot(sleep$brain_body_ratio ~ sleep$vore,
             xlab = "Vore",
             ylab = "Brain ratio",
             col = c("#C20000", "#006500", "#945200", "#FF9300"))
nbGroup <- length(unique(sleep$vore))
text(x=c(1:nbGroup), y=b$stats[nrow(b$stats),] - 0.001,
paste("n = ", table(sleep$vore), sep=""))
```



On voit une légère supériorité des régimes Insectivore et Omnivore. Une analyse par table de contingence montre un facteur de confusion entre l'ordre et le régime alimentaire.

```r
table(sleep$order, sleep$vore)
```

```
##
##                   carni herbi insecti omni
##   Afrosoricida        0     0       0    1
##   Artiodactyla        0     5       0    1
##   Carnivora          13     0       0    0
##   Cetacea             2     0       0    0
##   Chiroptera          0     0       2    0
##   Cingulata           1     0       1    0
##   Didelphimorphia     0     0       0    1
##   Diprotodontia       0     0       0    1
##   Erinaceomorpha      0     0       1    1
##   Hyracoidea          0     2       0    1
##   Lagomorpha          0     1       0    0
##   Monotremata         0     0       1    0
##   Perissodactyla      0     3       0    0
##   Pilosa              0     1       0    0
##   Primates            1     0       0    9
##   Proboscidea         0     2       0    0
##   Rodentia            0    10       0    2
##   Scandentia          0     0       0    1
##   Sirenia             0     1       0    0
##   Soricomorpha        0     0       2    3
```

(b) Sur la base de ces analyses, quelles variables vous semblent pertinentes pour l'ACP ? Quelles variables explicatives proposez-vous ?

Les variables les plus pertinentes pour la constitution des axes factoriels sont les variables de "sommeil", celles de "masse" et `brain_body_ratio`*. On peut utiliser les variables* `order` *(avec un nombre réduit de valeurs) ou* `vore` *en tant que variables explicatives.*

2. On propose de compléter les données manquantes de la colonne `sleep_rem` en utilisant une technique de regression par *la méthode des moindres carrés*. Quelle valeur est estimée pour l'individu *Lamantin* ? Compléter les valeurs manquantes.

On cherche à calculer la regression linéaire par la méthode des moindres carrés afin de prédire les valeurs manquantes de la variable `sleep_rem`*. Algébriquement, on calcule l'hyperplan tel que:*

$$\hat{y} = \hat{\beta}_0 + \sum_{i=1}^{n} \hat{\beta}_i x_i + \hat{\epsilon}$$

où:

- $\hat{y}$ = `sleep_rem`*, la variable ajustée.*
- $\hat{\beta}_{i \in \{0,\dots,n\}}$*, le coefficient estimé par la méthode des moindres carrés.*
- x_i *les variables quantitatives utiles à la régression.*
- $\hat{\epsilon}$*, un bruit blanc.*

Pour se faire, on va utiliser la fonction `lm` *(linear model) de R tel que:*

```r
fit <- lm(sleep_rem ~ sleep_total +
            brain_wt +
            body_wt +
```

```
              brain_body_ratio +
              gest_day, data = sleep)

new.sleep <- subset(sleep, is.na(sleep_rem))
predict(fit, newdata=new.sleep)
```

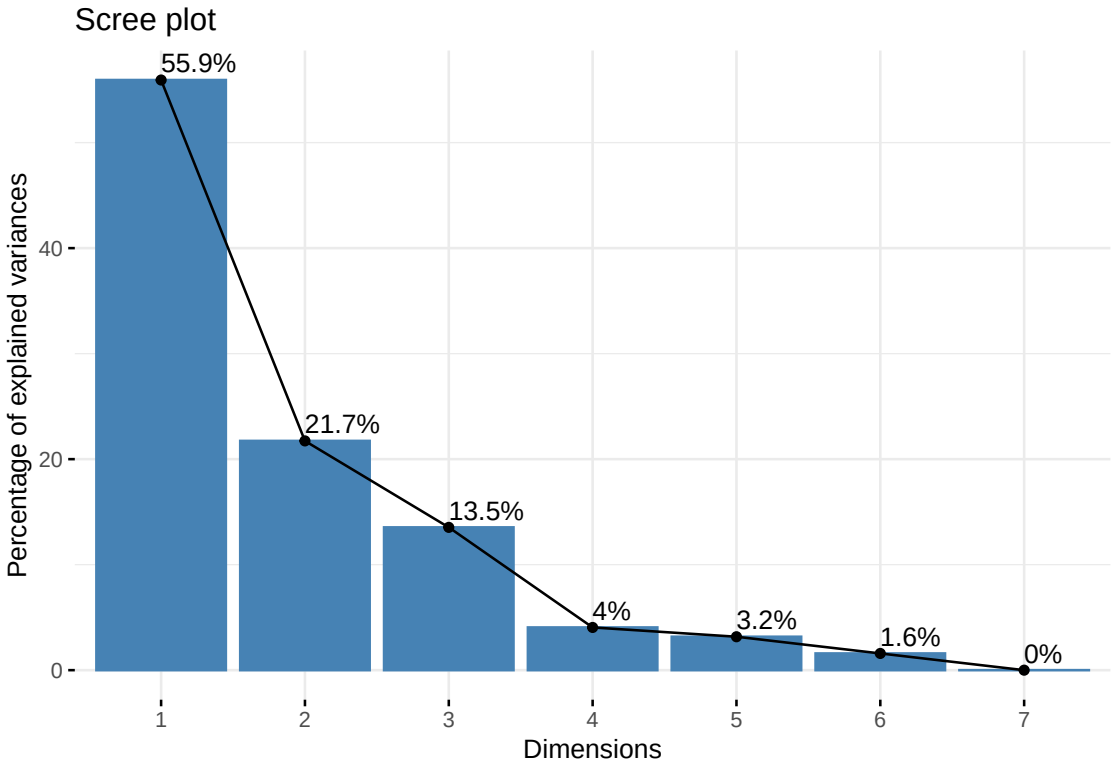
```
##         1         2         3         4         5         6         7         8
## 2.5893082 0.2476132 1.0478367 1.2843662 1.8343318 2.2525656 3.3290393 2.2651399
##         9        10        11        12        13        14        15        16
## 2.8713911 1.1978138 3.3261348 1.8105918 1.6744927 0.6847396 2.1496313 1.5590993
```

La valeur estimée de temps de sommeil paradoxal de l'individu Lamantin (6) est d'environ 2.25, soit environ 2h15.

```
# On remplace les valeurs manquantes.
values <- predict(fit, newdata=new.sleep)
new.sleep$sleep_rem <- values
sleep <- rbind(subset(sleep, !is.na(sleep_rem)), new.sleep)
```

3. Calculer les valeurs propres de la matrice des données `sleep`. Combien d'axes proposez vous de retenir pour l'ACP ? Détaillez votre réponse.

```
sleep.pca <- PCA(sleep[,5:ncol(sleep)], scale = TRUE, graph = FALSE)
fviz_eig(sleep.pca, addlabels = TRUE)
```

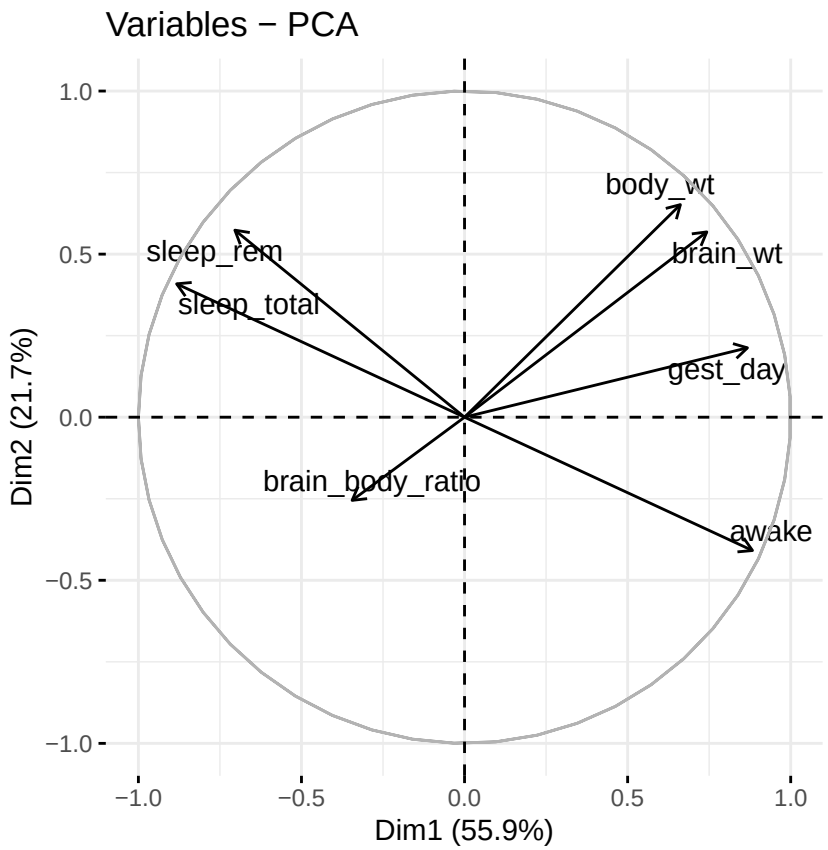


Le choix de retenir deux ou trois axes factoriels est pertinent ici. La part de variance expliquée avec deux axes, de 77.6 %, peut-être considérée comme suffisante. Néanmoins, la règle du coude et le fait que l'on puisse visualiser les données en 3D peut nous pousser à conserver trois axes. Plus loin, nous retenons trois axes pour appliquer un exemple de visualisation 3D.

4. Analyse des variables

(a) Commentez la qualité de représentation et la contribution de chaque variable quant aux axes retenus.

```
fviz_pca_var(sleep.pca, repel = TRUE)
```



On voit clairement sur le cercle des corrélations que la dimension 1 est expliquée par les variables de sommeil (`sleep_total`, `awake` et `sleep_rem`). La norme de ces vecteurs est proche de 1, elles sont donc bien représentées dans le plan factoriel. Les variables les mieux représentées sur le deuxième axe facoriel sont celles de masse (`brain_wt` et `body_wt`). La qualité de représentation est moins bonne. `gest_day` est correctement représentée par Dim1 et Dim2. La seule variable mal représentée est `brain_body_ratio`. On regarde la qualité de représentation des variables sur les autres axes factoriels.

```
round(get_pca_var(sleep.pca)$cos2, 2)
```

```
##                 Dim.1 Dim.2 Dim.3 Dim.4 Dim.5
## sleep_total      0.78  0.17  0.01  0.02  0.02
## sleep_rem        0.50  0.33  0.04  0.01  0.13
## awake            0.78  0.17  0.01  0.02  0.02
## brain_wt         0.55  0.32  0.06  0.00  0.00
## body_wt          0.44  0.42  0.03  0.05  0.02
## brain_body_ratio 0.12  0.07  0.80  0.00  0.02
## gest_day         0.75  0.05  0.00  0.18  0.01
```

On apprend ici que `brain_body_ratio` est représentée en grande partie par le troisième axe factoriel ce qui justifie la conservation de trois dimensions.

(b) Interpréter la signification des axes retenus. Vous pourrez vous aider de la contribution des

variables aux axes factoriels.

En inspectant la contributions aux axes, on remarque les variables qui contribuent le plus à la dimension 1 sont ***sleep_total****,* ***awake*** *et* ***gest_day****. Pour la dimension 2, ce sont les variables* ***brain_wt****,* ***body_wt****, et étonnement* ***sleep_rem****. La contribution des variables pour la dimension 3 est presque uniquement assurée par* ***brain_body_ratio****.*

5. Analyse des individus

   (a) Présenter la projection des indivus dans l'espace factoriel retenu. Vous colorerez dans un premier temps les points en fonction de la variable explicative retenue.

   Pour une projection 3D, on utilisera la commande `plot_ly(df, x = ~Dim.1, y = ~Dim.2, z = ~Dim.3)` de la librairie `plotly` où `df` est votre dataframe des coordonnées des individus et `Dim.k`, la colonne des coordonnées sur l'axe k.

```
library(plot3D)
library(plotly)

coord <- as.data.frame(get_pca_ind(sleep.pca)$coord)
plot_ly(coord, x = ~Dim.1, y = ~Dim.2, z = ~Dim.3,
        color = sleep$vore,
        colors = c("#C20000", "#006500", "#945200", "#FF9300"),
        name = sleep$name)
```

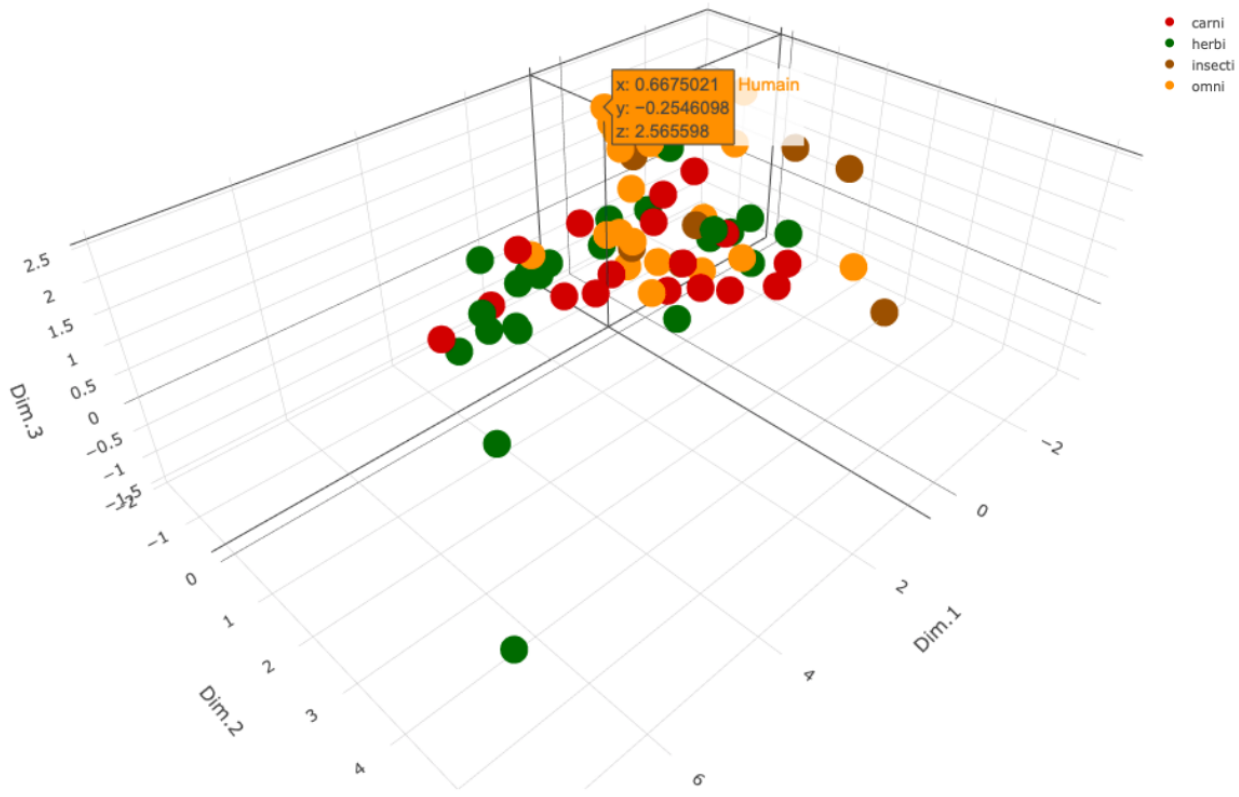


Figure 2: Projection ACP

   (b) Colorer les individus en fonction de leur qualité de réprésentation aux axes factoriels puis en

fonction de la contribution. Commentez ces résultats.

```
qual <- as.data.frame(get_pca_ind(sleep.pca)$cos2)
df <- as.data.frame(get_pca_ind(sleep.pca)$coord)
plot_ly(df, x = ~Dim.1, y = ~Dim.2, z = ~Dim.3,
      color = qual$Dim.1 + qual$Dim.2 + qual$Dim.3,
      name = sleep$name)
```

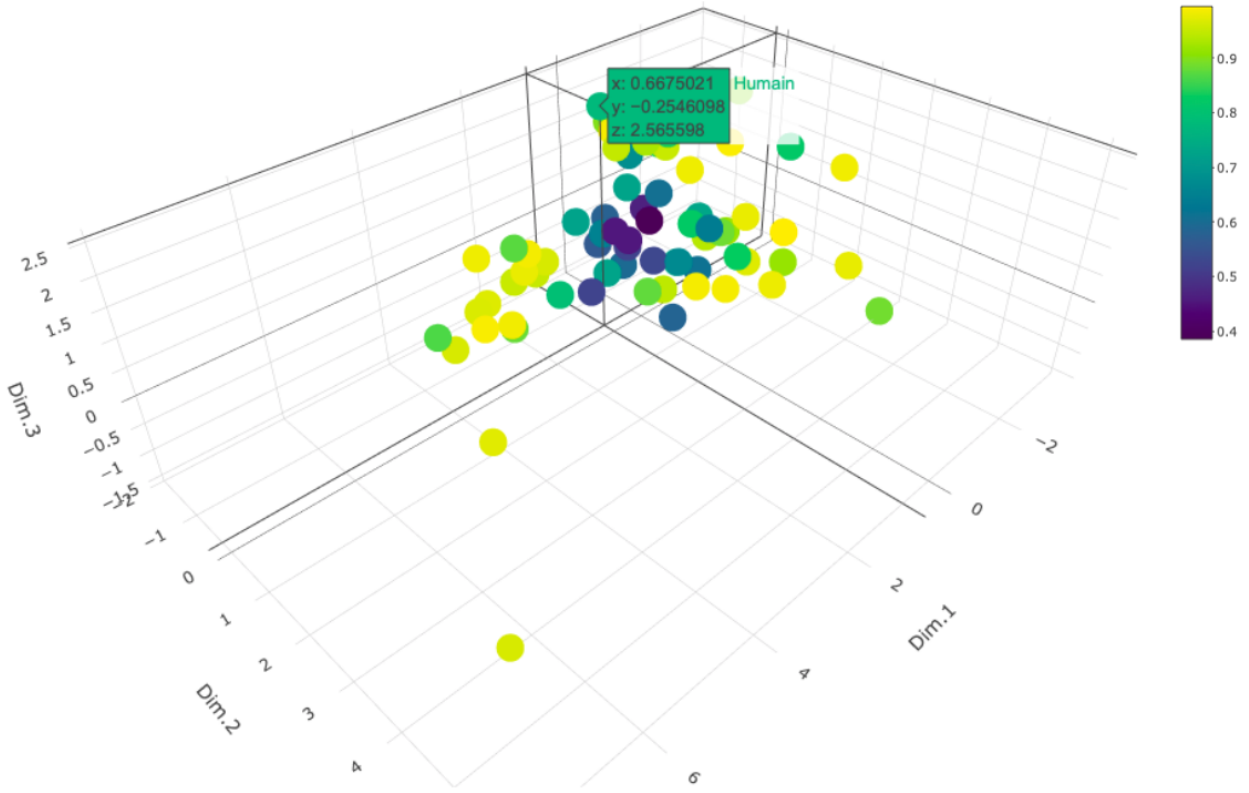


Figure 3: Projection ACP avec qualité de projection

De même la façon que vu précédemment, on constate que la qualité de représentation est la plus dégradée à l'origine.

Classiffication de caractères manuscrits

On considère le fichier `mnist.csv` sur Celene. Ces données proviennent de la base MNIST[2] sur laquelle des milliers de chercheurs ont travaillé. Elle est constituée initialement de 70.000 chiffres manuscrits au format 28 pixels par 28 pixels où chaque pixel est représenté par un niveau de gris allant de 0 à 255. Un chiffre manuscrit est vu comme un vecteur de $\{0, ..., 255\}^{28\times 28}$.

Pour limiter le temps de calcul et la mémoire nécessaire, nous ne considérons que les 20.000 premiers chiffres manuscrits de la base originale. On donne la description des colonnes suivantes:

- chaque ligne correspond à un chiffre manuscrit.
- la première colonne contient la *classe* (ou label) du caractère, c'est-à-dire le chiffre qu'il représente.

[2] http://yann.lecun.com/exdb/mnist/

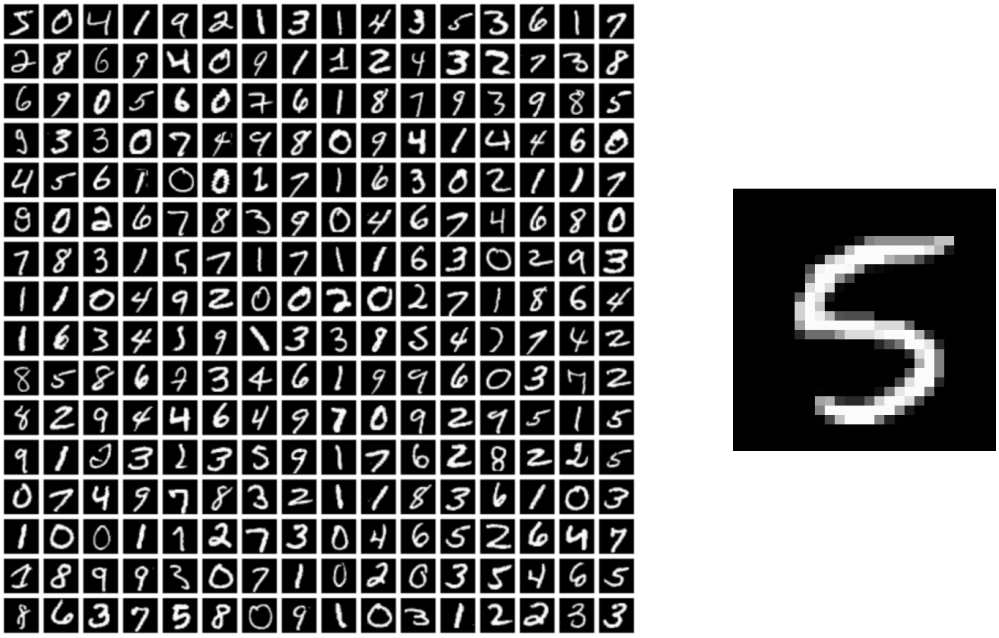


Figure 4: Exemple de caractères manuscrits. Le caractère manuscrit à droite fait partie de la classe '5'

- les colonnes suivantes, contiennent les valeurs des $28 \times 28 = 784$ pixels de l'image en commençant par le coin supérieur gauche et parcourant l'image ligne par ligne.

On donne la fonction de visualisation suivante:

```
img <- function(data, row_index){

    r <- as.numeric(data[row_index, 2:785])
    im <- matrix(nrow = 28, ncol = 28)
    j <- 1
    for(i in 28:1){
        im[,i] <- r[j:(j+27)]
        j <- j+28
    }
    image(x = 1:28,
          y = 1:28,
          z = im,
          col=gray((0:255)/255),
          main = paste("Number:", data[row_index, 1]))
}
```

L'appel `img(mnist, i)` retourne la figure correspondant au caractère manuscrit ligne i.

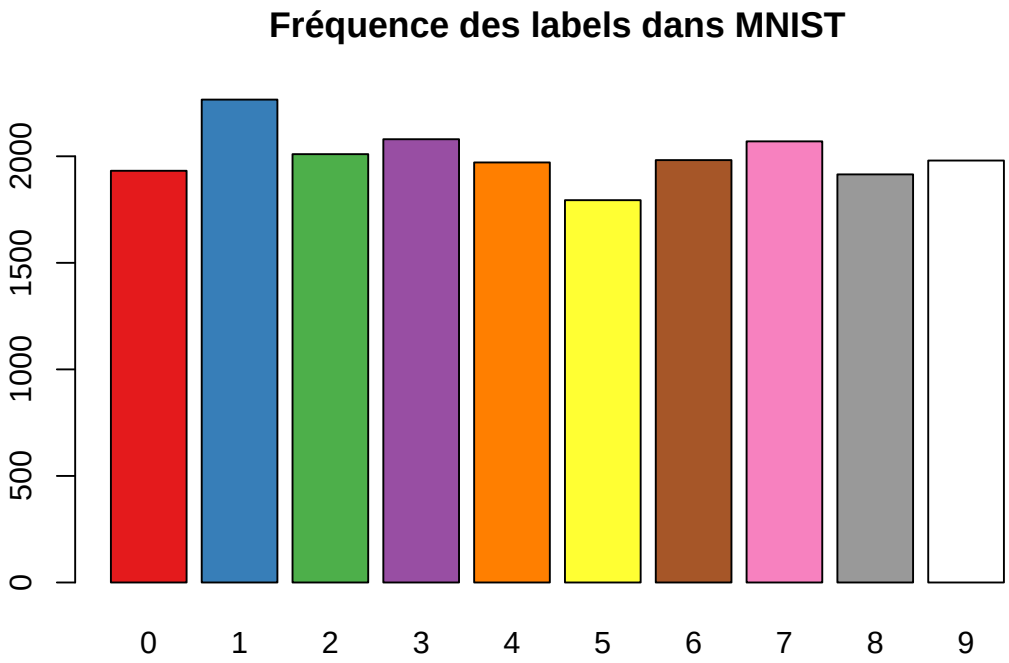
1. Statistiques descriptives

   (a) Proposer une analyse préliminaire par statistiques descriptives du jeu de données `mnist`. Votre analyse devra contenir notamment:

   - Nombre de caractères de chaque classe.
   - Des premiers indicateurs sur la proportion de gris par pixel, puis agrégé par classe de caractère.

```r
library(RColorBrewer)

barplot(table(mnist$label),
main = "Fréquence des labels dans MNIST",
col = c(brewer.pal(n = 9, name = "Set1"), "white"))
```



Les labels sont répartis de façon assez uniforme. On constate qu'il y'a un peu moins de chiffres 5 et un peu plus de 1. Une classe fortement déséquilibrée supposerait un traitement particulier. Ce n'est pas le cas ici.

Pour l'analyse pixel par pixel, la quantité importante (784 pixels) impose un pré-traitement afin de pouvoir visualiser de façon concise les données. Pour se faire, on peut utiliser une technique de zoning. Cette technique consiste à fragmenter l'image originelle en meta-pixel comme illustré dans la figure ci-dessous.

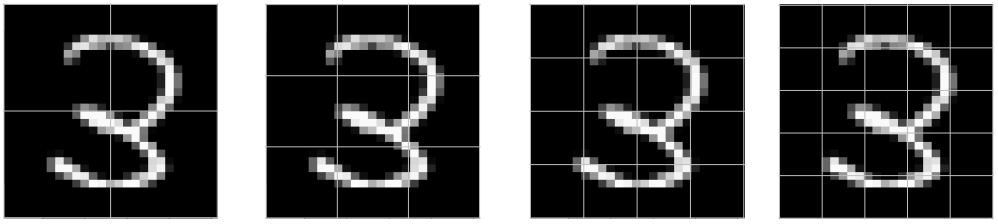


Figure 5: Zoning sur le caractère 140 de Mnist pour $n = 2, 3, 4$ et 5

Le meta-pixel correspond alors à la moyenne des niveaux de gris pour la zone considérée. La fonction suivante permet de calculer les meta-pixels pour une valeur n donnée:

```r
zoning <- function(n) {
  v <- rep(0, n * n)
  sep <- 28 / n
  # Fragmentation en zones
  for(k in 0:783) {
```

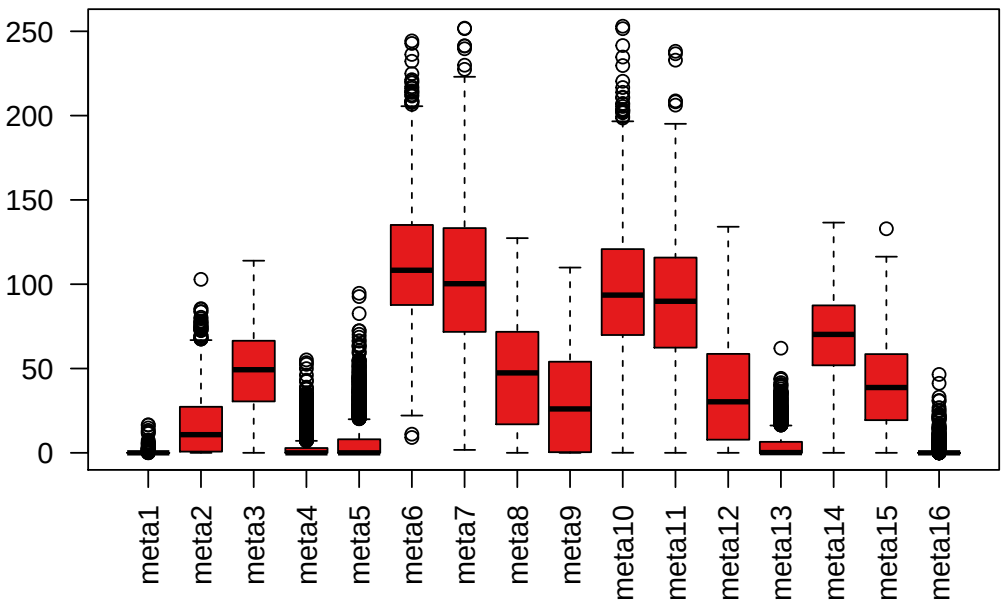
```
  l = as.integer(k / 28)
  c = k %% 28
  case = as.integer(l / sep) * n + as.integer(c / sep)
  v[case+1] = v[case+1] + mnist[, (k+2)]
  }
  # Création du dataframe
  meta_pixel <- as.data.frame(as.factor(mnist$label))
  names(meta_pixel) <- "label"
  for(k in 1:(n*n)) {
    meta_pixel <- cbind(meta_pixel, as.data.frame(v[k]))
    meta_pixel[ncol(meta_pixel)] <- meta_pixel[ncol(meta_pixel)] / ((28 / n)^2)
    names(meta_pixel)[ncol(meta_pixel)] <- paste("meta", k, sep = "")
  }
  return(meta_pixel)
}
```

On applique un zoning avec $n = 4$. Puis on dresse la série des boîtes à moustaches pour les chiffres 0 et 1.

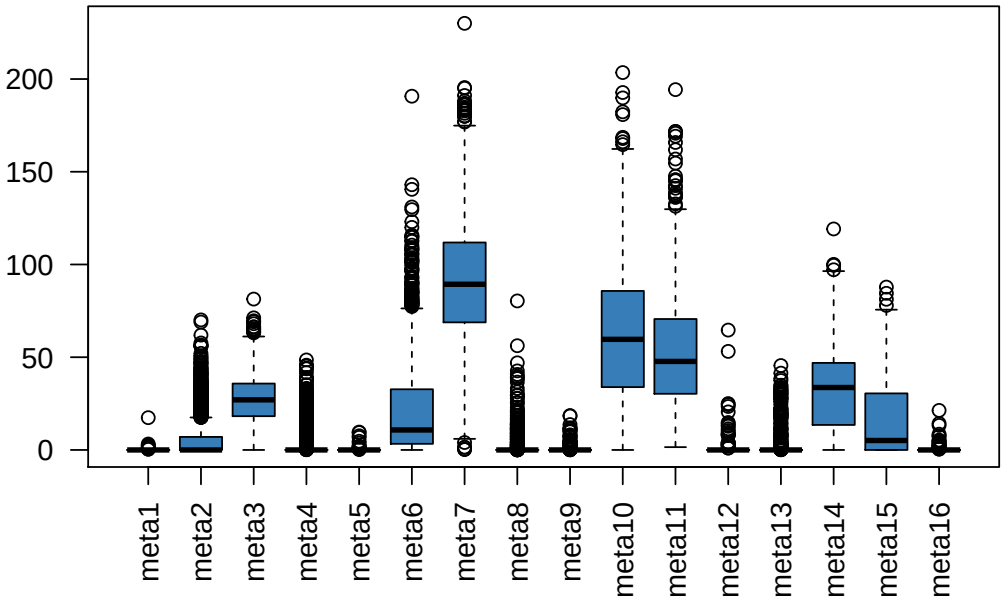
```
meta_pixel <- zoning(4)

meta_pixel0 <- subset(meta_pixel, label == 0)
meta_pixel1 <- subset(meta_pixel, label == 1)
boxplot(meta_pixel0[2:17], las = 2,
        main = "Niveau de gris des meta-pixels : 0",
        col = brewer.pal(n = 2, name = "Set1")[1])
```

```
## Warning in brewer.pal(n = 2, name = "Set1"): minimal value for n is 3, returning requested p
```




```
boxplot(meta_pixel1[2:17], las = 2,
        main = "Niveau de gris des meta-pixels : 0",
        col = brewer.pal(n = 3, name = "Set1")[2])
```



On voit distinctement ici que certains meta-pixels sont plus clairs pour les chiffres 0 et plus foncés pour les 1. Par exemple les meta-pixels 6, 8, 9 et 12.

(b) Sur la base de ces analyses, certaines zones de l'image vous semblent t-elles plus pertinentes pour l'analyse ? Lesquelles ? Pourquoi ?

Pour connaitre les zones de l'images les plus discriminantes. On calcule la variance (ou écart-type) par meta-pixel. Plus la variance est grande, plus on a une diversité importante des niveaux de gris dans les images. On peut émettre l'hypothèse que chaque classe de caractère suit, pour un meta-pixel donné, une loi statistique quant à son niveau de gris caractéristique ce qui permettrait de les différencier.

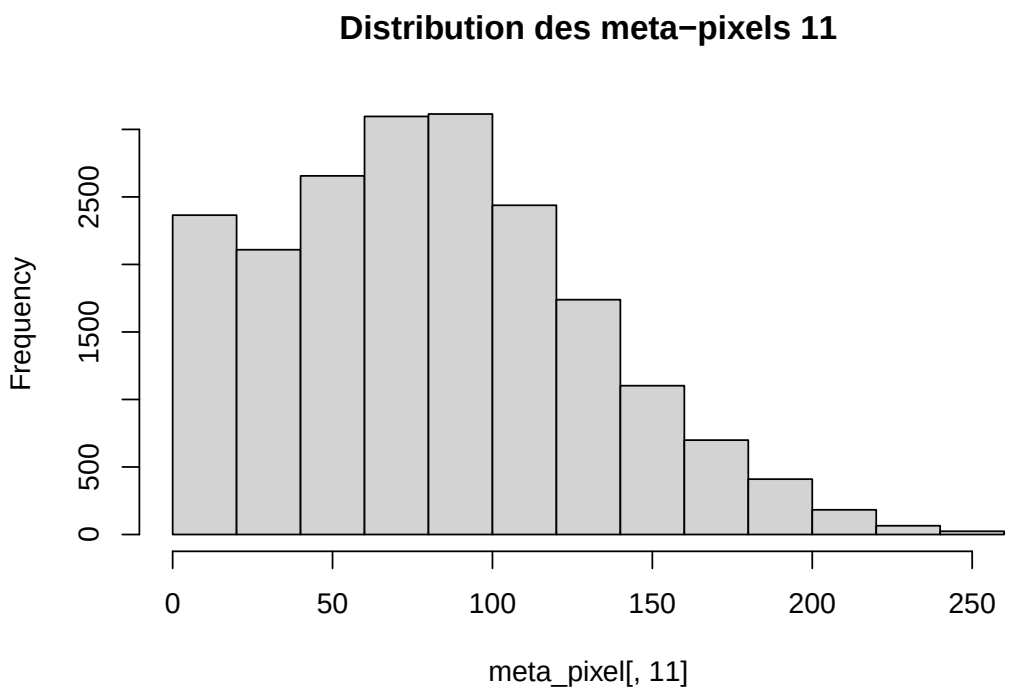
```
for (i in 2:ncol(meta_pixel)) {
  cat(i, ": ", sd(meta_pixel[,i]), "\n")
}
```

```
## 2 :  2.373936
## 3 :  22.18004
## 4 :  26.62249
## 5 :  7.583808
## 6 :  12.5006
## 7 :  44.46598
## 8 :  42.1045
## 9 :  22.4383
## 10 :  17.24568
## 11 :  48.99201
## 12 :  42.67061
## 13 :  21.50809
## 14 :  7.41112
## 15 :  32.42082
## 16 :  27.99066
## 17 :  5.220051
```

De façon prévisible on remarque que les pixels en périphérie de l'image ont un écart-type

faible. On peut analyser la distribution du meta-pixel 11 qui a l'écart-type le plus important.

```
hist(meta_pixel[, 11],
     main = "Distribution des meta-pixels 11")
```

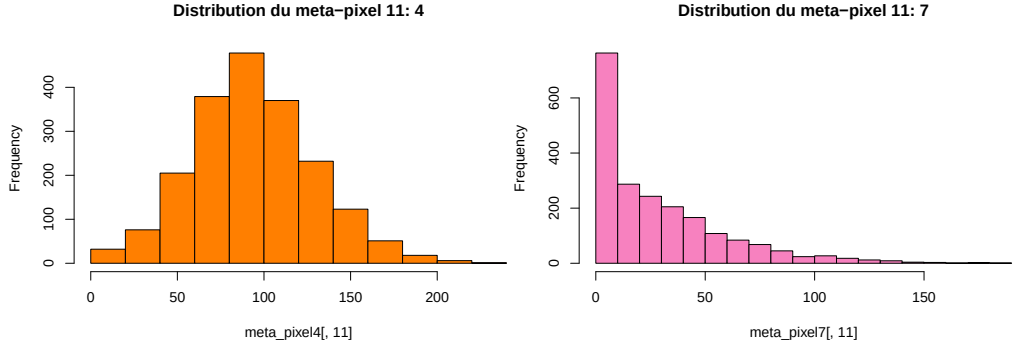


On remarque les valeurs de gris sont très étalées de 0 à 255 avec une prédominance du sombre (valeur proche de 0 et mode autour de 80). Pour se convaincre que les classes de caractères adoptent des distributions très différentes pour ce pixel, on trace les histogrammes pour les caractères 4 et 7 qui sont des exemples frappants.

```
meta_pixel4 <- subset(meta_pixel, label == 4)
meta_pixel7 <- subset(meta_pixel, label == 7)

hist(meta_pixel4[,11],
     col = brewer.pal(n = 5, name = "Set1")[5],
     main = "Distribution du meta-pixel 11: 4")

hist(meta_pixel7[,11],
     col = brewer.pal(n = 8, name = "Set1")[8],
     main = "Distribution du meta-pixel 11: 7")
```



La première distribution est clairement normale tandis que la seconde semble plus tirer de l'exponentielle. En conséquence les meta-pixels 11 des caractères 4 semblent plus clairs tandis que pour les 7 ceux-ci semblent très largement noirs.

2. Classification par l'algorithme des k plus proches voisins (kNN).

   L'algorithme des k proches voisins (k-Nearest Neightbors) est une méthode de prédiction qui, pour une base de données d'apprentissage, cherche à déterminer la classe d'une donnée inconnue.

   L'idée générale de cet algorithme est très simple. Pour une nouvelle donnée d'entrée x, on évalue sa distance à toutes les autres données connues de notre base d'apprentissage $\mathbf{X}$.

   On rappelle que la distance euclidienne entre deux éléments $x, y \in \mathbb{R}^p$ est définie telle que:

$$\|x - y\| = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2}$$

   On retient ensuite uniquement les k voisins $\mathbf{X_i}$ les plus proches de x. On regarde alors les classes $\mathbf{Y_i}$ de ces données $\mathbf{X_i}$, puis on prédit la classe la plus présente. Par défaut on utilisera $k = 1$.

   (a) En assumant que $\mathbf{X}$ est doté de n individus définis dans un espace de dimension p. Quelle est la complexité de l'algorithme des k-Nearest Neightbors pour $k = 1$.

   On a une complexité en $O(p)$ pour le calcul de la distance euclidienne et une complexité en $O(n)$ pour l'application à l'ensemble de données $\mathbf{X}$. Ainsi, kNN est de complexité $O(n \times p)$.

   (b) Diviser le jeu de données `mnist` en deux ensembles :

   - Un ensemble d'apprentissage (train set) qui contiendra 80% du jeu initial.
   - Un ensemble test (test set) qui contiendra le reste des données.

   On veillera à conserver les labels des deux ensembles dans un vecteur à part.

```
percentage <- 0.8

X_train <- mnist[1:(nrow(mnist)*percentage), -1]
Y_train <- mnist[1:(nrow(mnist)*percentage), 1]

X_test <- mnist[(nrow(mnist)*percentage+1):nrow(mnist), -1]
Y_test <- mnist[(nrow(mnist)*percentage+1):nrow(mnist), 1]
```

   (c) La commande `knn` du package `class` permet de réaliser une classification à l'aide de l'algorithme des k-Nearest Neightbors:

```
library(class)

knn(X_train, X_test, cl = Y_train_label, k = nb_neightbors)
```

   Appliquer l'algorithme kNN (avec $k = 1$) sur votre ensemble d'apprentissage et de test. On veillera à sauvegarder le résultat de la fonction dans une variable `prediction`:

```
prediction <- knn(...)
```

   Donner le temps d'exécution de l'algorithme.

```
library(class)
prediction <- knn(X_train, X_test, cl = Y_train$label, k = 1)
```

Sur un mac book air doté d'un processeur 2,2 GHz Intel Core i7, l'algorithme met 4min à s'exécuter.

(d) La commande `table(Y_test_label, prediction)` permet de dresser la *matrice de confusion* C de la classification effectuée. Le nombre c_{ij} représente le nombre d'éléments de la classe i classifiés en tant que j.

Quel est le pourcentage de caractères manuscrits de l'ensemble de test qui ont été mal classés ? Cet algorithme vous semble t-il efficace ? Quel critique peut-on lui faire ?

```
C <- table(Y_test$label, prediction)
C
```

```
##    prediction
##       0   1   2   3   4   5   6   7   8   9
##   0 394   0   0   0   0   0   2   0   1   0
##   1   0 446   1   0   1   1   0   1   1   1
##   2   1   7 383   3   0   0   0   5   2   0
##   3   2   0   3 418   0  11   0   3   5   3
##   4   0   6   0   0 379   0   1   1   1  11
##   5   1   0   1  10   2 309   7   1   2   3
##   6   2   1   0   1   0   2 381   0   1   0
##   7   0   9   4   1   1   0   0 400   0   2
##   8   0   1   3   9   1  10   2   2 340   5
##   9   3   0   1   0   7   3   0   6   2 370
```

On peut calculer le score de misclassification qui est égale à:

$$misclassification = 1 - \frac{1}{N}\sum_{k=1}^{n} c_{kk}$$

où N est la taille du jeu de test `X_train` est n, le nombre de classes.

```
misclassification <- function(C) {
  return(1 - 1/sum(C) * sum(diag(C)))
}

cat("Misclassification = ", misclassification(C)*100, "%")
```

```
## Misclassification =  4.5 %
```

On a seulement une erreur de 4,5%, ce qui est un excellent score au vue de la simplicité de l'algorithme. Son seul défaut est une lenteur relative.

(e) Pour chaque classe, identifier un exemple de caractère mal classé par l'algorithme. Vous illustrerez ces caractères à l'aide de la fonction `img` donnée plus haut et ferez figurer la classe prédite et réelle des caractères.

La fonction suivante permet de retourner, la liste des id des caractères i ayant été confondus avec un caractère j:

```
erreur <- function(i, j) {
  for(k in 1:nrow(Y_test)) {
    if(Y_test$label[k] == i & prediction[k] == j) {
      cat(k + nrow(Y_train), "\n")
    }
  }
```

```
}

erreur(0, 6)
```

```
## 18899
## 19572
```

Par exemple, on voit que les caractères 18899 et 19572, qui sont originellement des 0 ont été confondus avec des 6. La figures ci-dessous montrent des exemples d'erreurs commises.

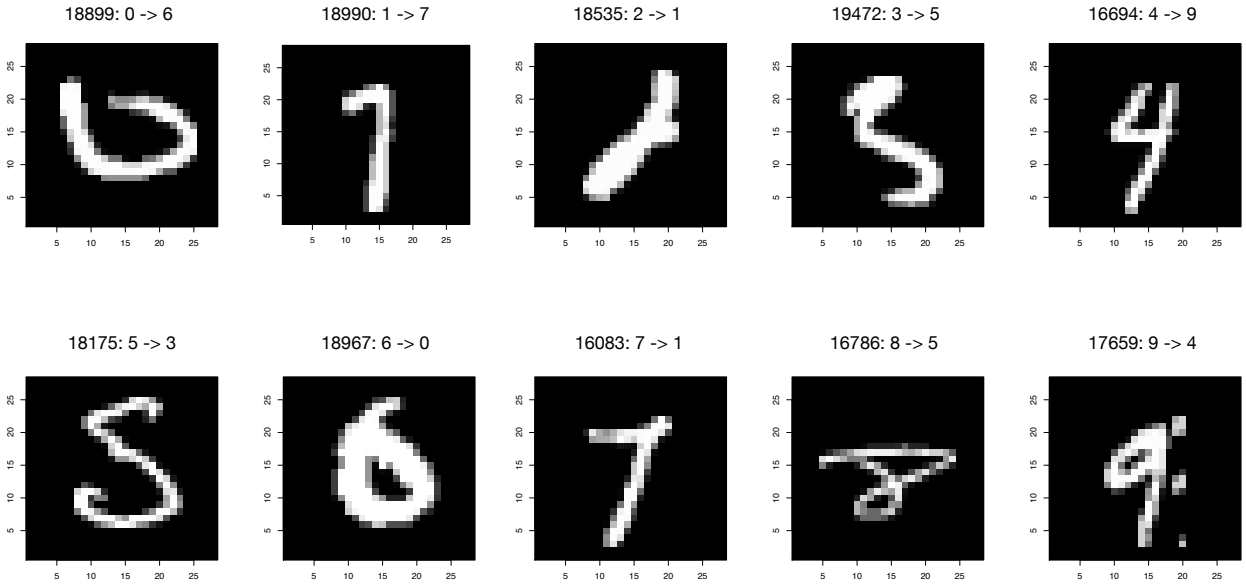


Figure 6: Exemple de caractères manuscrits mal classés: *id*: i -> j

3. Prétraitement-compression des données par ACP

   (a) Effectuer une ACP du jeu `mnist` et analyser la série des valeurs propres. Combien de composantes doivent être conservées pour avoir plus de 95% de l'inertie.

```
mnist.pca <- PCA(mnist[, -1], scale = TRUE, graph = FALSE)
```

```
eig.val <- get_eigenvalue(mnist.pca)
round(eig.val,2)
```

On constate qu'il faut conserver 303 dimensions pour expliquer 95% de la variance totale du jeu de données.

   (b) Appliquer à nouveau l'algorithme kNN mais ici vous utiliserez comme jeu initial la projection via ACP réalisée à la question précédente. Que constatez-vous ?

```
mnist.pca <- PCA(mnist[, -1], scale = TRUE, graph = FALSE, ncp = 303)
acp_coor <- as.data.frame(get_pca_ind(mnist.pca)$coord)
X_train_acp <- acp_coor[1:(nrow(acp_coor)*percentage), ]
X_test_acp <- acp_coor[(nrow(acp_coor)*percentage+1):nrow(acp_coor), ]
```

```
prediction <- knn(X_train_acp, X_test_acp, cl = Y_train$label, k = 1)
```

Le temps d'exécution est moins long, l'algorithme prend cette fois-ci 1.10min d'exécution, soit environ un gain de rapidité $\times 4$.

(c) Dresser la nouvelle matrice de confusion à l'issu de la classification précédente. Comparer ces résultats avec la matrice de la question 2. (d). Que peut-on dire ?

```
C <- table(Y_test$label, prediction)
C
```

```
##    prediction
##       0   1   2   3   4   5   6   7   8   9
##   0 389   1   2   0   0   0   4   0   0   1
##   1   0 444   1   0   1   1   1   2   1   1
##   2   2   3 369   9   1   0   3   5   5   4
##   3   5   0   8 393   0  21   1   5   8   4
##   4   1   3   3   0 361   0   3   2   3  23
##   5   2   1   2  12   1 303   8   1   3   3
##   6   4   0   2   1   1   2 377   0   1   0
##   7   0  10   3   3   4   0   0 387   0  10
##   8   2   6   5   9   3  15   2   4 320   7
##   9   1   0   3   1  17   1   0  14   2 353
```

```
cat("Misclassification = ", misclassification(C)*100, "%")
```

```
## Misclassification =  7.6 %
```

On constate que l'algorithme a fait plus d'erreurs que précédemment, ce qui est normal car en procédant à une ACP on a détruit de l'information. Néanmoins, on peut estimer cette faible dégradation de performances (de 3.1%) valable au vue du gain de rapidité de l'algorithme.