

# TP 2 Importation et exportation de données

## Les outils graphiques de R

Consulter les données disponibles sur R

```
Consulter les données disponibles sur les packages chargées en mémoire data()
```

```
Consulter les données disponibles sur tous les packages du poste  
data(package = .packages(all.available = TRUE))
```

### Lire les données d'un fichier

R peut lire les données stockées dans des fichiers texte (ASCII) grâce, entre autres, aux fonctions suivantes `read.table()`, `scan()`.

Lorsque les données ont été sauvegardées sous le format propriétaire d'un logiciel statistique tiers, il est nécessaire de disposer d'outils permettant leur transfert vers le système R. La librairie `foreign` offre ces outils pour une sélection des logiciels statistiques les plus courants, à savoir SAS, SPSS et Stata. Par exemple, la fonction `read.spss` prend en charge les données enregistrées au moyen des commandes `save` et `export` de SPSS.

Par ailleurs, la fonction `read.xls` du package `gdata` fournit les outils pour lire des fichiers au format excel. Pour plus d'informations, je vous renvoie au document proposé par le «R development core team» disponible gratuitement sur internet à l'adresse suivante <http://www.r-project.org>.

Dans ce TP, nous nous limitons à la lecture des fichiers au format ASCII.

### Utilisation de la fonction `read.table()`

```
Importer dans un objet nommé A le jeu de données nommé auto2004_original.txt
```

```
A = read.table('chemin/auto2004_original.txt', header = TRUE, sep = '\t')
```

```
Importer dans un objet nommé B le jeu de données auto2004_sans_nom.txt
```

```
B = read.table('chemin/auto2004_sans_nom.txt', sep = '\t')
```

```
Importer dans un objet nommé C le jeu de données auto2004_virgule.txt
```

```
C = read.table('chemin/auto2004_virgule.txt', header = TRUE, sep = '\t',  
dec = ',')
```

```
Importer dans un objet nommé D le jeu de données auto_don_manquante.txt
```

```
D = read.table('chemin/auto2004_don_manquante.txt', header = TRUE, sep =  
\t')
```

```
Importer dans un objet nommé E le jeu de donnée auto_don_manquante(99999).txt
```

```
E = read.table('chemin/auto2004_don_manquante(99999).txt', header = TRUE,  
sep = '\t', na.strings = 99999)
```

Quel est le mode des objets créer par la fonction read.table() ?

Le mode des objets créés par la fonction read.table est la data.frame[]:  
is.data.frame(A)

Remarque[] on peut créer des objets d'un autre mode que data.frame en utilisant la fonction scan(). Nous ne traiterons pas cette situation dans le TP.

### Enregistrer des données

Créer la matrice suivante[] `matrice = matrix(1:25, 5, 5)`

Sauver la matrice sous le nom de 'matrice.txt' à une adresse valide. Que remarquez vous[]

```
write.table(matrice, 'chemin/matrice.txt')
```

On remarque que par défaut les lignes et les colonnes sont nommées

Ajouter des arguments à la commande précédente pour retirer des noms aux lignes et aux colonnes du fichier crée.


```
write.table(matrice, 'chemin/matrice.txt', row.names = F, col.names = F)
```

Sauvegarder la data.frame A dans le fichier nommé auto.txt

```
write.table(A, chemin/auto.txt')
```

Sauver les objets disponibles en mémoire vive à l'adresse '/chemin/Donnees.Rdata'



```
save(list = ls(all = TRUE), file = "/chemin/Donnees.Rdata")
```

Remarque[] Si on ne spécifie pas le chemin de sauvegarde,  sauve les données à l'adresse données par la fonction `getwd()`. On peut modifier le chemin par défaut grâce à la fonction `setwd()` Par exemple[] `(setwd("/Documents/donnees"))`.


Sauver les objets disponibles en mémoire vive à l'adresse "/chemin/" grâce à la commande `setwd()`

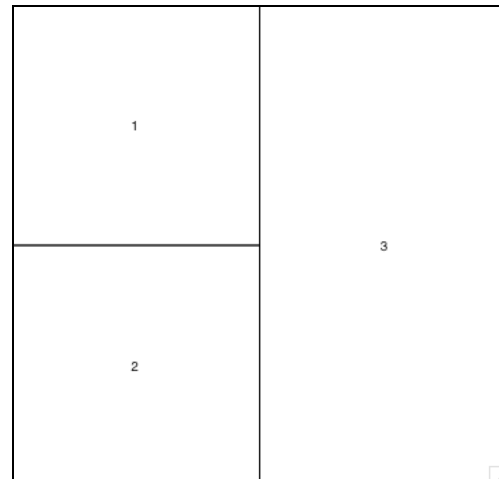
```
setwd("/chemin/")  
save.image()
```

## Les graphiques


 propose de nombreux outils graphiques pour l'analyse et la visualisation des données. On peut avoir un aperçu des possibilités graphiques de  grâce à la commande `demo(graphics)`.

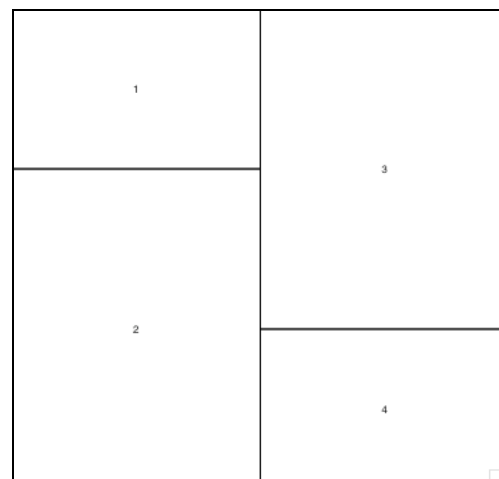
### Gestion des fenêtres graphiques

Créer la partition suivante 



```
mat1 = matrix(c(1, 2, 3, 3), 2, 2)
layout(mat1)
layout.show(3)
```

Créer la partition graphique suivante 



```
mat2 = matrix(c(1, 2, 2, 3, 3, 4), 3, 2)
layout(mat2)
layout.show(4)
```

## Utilisation de fonctions graphiques de

### Utilisation de la fonction `plot()`

#### Première exercice

Créer une matrice, `mat1`, composée de 100 lignes et deux colonnes—chacune des colonnes des vecteurs aléatoires de loi normale centrée et de variance 1.

```
mat1 = matrix(rnorm(200, 0, 1), 100, 2)
```


Créer une matrice, `mat2`, composée de 100 lignes et deux colonnes—chacune des colonnes étant des vecteurs aléatoires de loi normale de moyenne 1 et de variance 1.

```
mat2 = matrix(rnorm(200, mean = 1, sd = 0.5), 100, 2)
```

Tracer le graphe bivarié de la première colonne de `mat1` sur la deuxième colonne de `mat1`

```
plot(mat1)
```

Ajouter aux graphe précédent le graphe bivarié de la première colonne de `mat2` sur la deuxième colonne de `mat2`. Que constatez vous?

Remarque—On utilisera des couleurs différentes pour distinguer les points de `mat1` des points de `mat2`. Pour connaître la liste des couleurs disponible sur , tapez la commande `colors()`.

```
plot(mat1, col = "blue")  
points(mat2, col = "red")
```

On constate que les points de `mat2` ne sont afficher que dans le graphique définit par les bornes de `mat1`.

Utiliser les arguments `xlim` et `ylim` pour contourner cette problématique.

```
mat = rbind(mat1, mat2)  
plot(mat1, col = "blue",  
      xlim = range(mat[,1]),  
      ylim = range(mat[,2]),  
      main = "représentation d'un nuage de points",  
      xlab = "X1", ylab = "X2")  
points(mat2, col = "red")
```

Utiliser les options de la fonction `plot()` pour générer un graphique esthétique

Par exemple:

```
plot(1,  
     xlim = range(mat[,1]),  
     ylim = range(mat[,2]),  
     main = "représentation d'un nuage de points",  
     xlab = "X1", ylab = "X2",  
     bty = "l", tcl = -.25  
     )  
rect(-3, -3, 3, 3, col = "cornsilk")
```

```
points(mat1, col = "blue", pch = 22, bg = "red")
points(mat2, col = "red", pch = 25, bg = "yellow")
```

## Deuxième exercice

Dans le package datasets est disponible le jeu de données iris.

Charger ce jeu de données en mémoire

```
data(iris)
```

Quel est le mode de ce jeu de données?

```
Ce jeu de données est une data.frame
```

Convertir le jeu de données iris en une matrice nommé matrice\_iris

```
matrice_iris = as.matrix(iris[, 1:4])
```

Tapez la commande plot(matrice\_iris). Que construit la fonction plot() dans le cadre des matrices?

Dans le cadre de matrices la fonction plot() trace le graphe de la première colonne de la matrice sur la deuxième.

Construire le graphique composé de tous les graphiques bivariés du jeu de données Iris (à partir de matrice\_iris).

Ajouter une forme et une couleur aux points en fonction de l'espèce

Ajouter un titre et un label.

```
pairs(matrice_iris[,1:4],
      bg = c("red", "green3", "blue")[as.numeric(iris[, 5])],
      pch = c(21, 25, 24)[iris[, 5]],
      main = "Iris de Fisher",
      labels = c("Longueur\nSepale",
                "Largeur\nSepale",
                "Longueur\nPetale",
                "Largeur\nPetale"
               )
)
```

## fonction plot () dans le cadre d'une data.frame

### Troisième exercice

Charger en mémoire le jeu de données iris disponible dans le package datasets.

Ce jeu de données est une data.frame. Tapez la commande plot(iris). Que constatez-vous?

On constate que dans le cadre des data.frame la fonction plot trace toutes les combinaisons possible de graphes bivariés.

Faire un graphique esthétique du jeu de données iris (couleur, titre, forme, labels, etc...)

Par exemple :

```
plot(iris[,1:4],
     bg = c("red", "green3", "blue")[iris[,5]],
     pch = c(21, 25, 24)[iris[,5]],
     main = "Iris de Fisher",
     labels =
     c("Longueur\nSepale",
       "Largeur\nSepale",
       "Longueur\nPetale",
       "Largeur\nPetale"
     )
)
```

## Utilisation de la fonction `hist()`

### Quatrième exercice

Intéressons nous aux liens entre le temps d'attente entre deux éruptions et la durée des éruptions pour le «[Old faithful geyser](#)» du parc national du Yellowstone (Wyoming, États-Unis). Ce jeu de données nommé `faithful` est disponible dans le package `datasets`.

Lorsqu'on tape la commande `?faithful`, on obtient un descriptif du jeu de données.

Visualiser le jeu de données `faithful` par l'intermédiaire de la fonction `plot`.

Définir un seuil critique d'attente au delà duquel, la probabilité que la prochaine éruption soit longue, soit forte.

Posons le seuil critique arbitrairement à 63.

Construire un histogramme de la durée d'éruption. Ajouter un titre, labeliser l'axe des abscisses, colorer les barres de l'historgramme en vert, colorer les traits de l'historgramme en rouge et représenter l'historgramme en terme de fréquence plutôt qu'en termes d'effectifs.

```
hist(faithful$eruptions, col = "green", border = "red", proba = TRUE,
     main = "histogramme du temps des eruptions",
     xlab = "le old faithful geyser"
)
```

Augmenter la taille du pas de l'historgramme à 20 et ajuster une loi normale à cet histogramme. Que remarque t'on?

```
hist(faithful$eruptions, col = "green", border = "red", proba = TRUE,
     main = "histogramme du temps des eruptions",
     xlab = "le old faithful geyser", breaks = 20
)
```

```
x = seq(from = 1.5, to = 5, length = 500)
```

```
y = dnorm(x, mean = mean(faithful$eruptions),
          sd=sd(faithful$eruptions)
        )
```

```
lines(x, y, col = "brown")
mtext("Ajustement à une loi normale")
```

L'ajustement à une loi normale n'est pas adéquat.

### Utilisation des estimateurs locaux de la densité

Superposer à l'histogramme les estimateurs locaux de la densité associés aux paramètres d'ajustement 0.1, 0.3, 0.8 et 1.

```
hist(faithful$eruptions, col = "yellow", border = "green", proba = TRUE,
     main = "histogramme du temps des ruptions",
     xlab = "le old faithful geyser", breaks = 20
    )
lines(density(faithful$eruptions, adj = .1), type='l', col='red', lwd=1)
lines(density(faithful$eruptions, adj = .3), type='l', col='green', lwd=1)
lines(density(faithful$eruptions, adj = .8), type='l', col='blue', lwd=1)
lines(density(faithful$eruptions, adj = 1), type='l', col='black', lwd=1)
```

On s'aperçoit que l'estimateur local de la densité associé à un coefficient d'ajustement de 0.1 dépasse de la fenêtre graphique.

Imposer donc des contraintes à la taille de l'axe des ordonnées.

```
hist(faithful$eruptions, col = "yellow", border = "green", proba = TRUE,
     main = "histogramme du temps des ruptions",
     xlab = "le old faithful geyser", breaks = 20, ylim = c(0, 1)
    )
lines(density(faithful$eruptions, adj = .1), type='l', col='red', lwd=1)
lines(density(faithful$eruptions, adj = .3), type='l', col='green', lwd=1)
lines(density(faithful$eruptions, adj = .8), type='l', col='blue', lwd=1)
lines(density(faithful$eruptions, adj = 1), type='l', col='black', lwd=1)
```

## Utilisation de la fonction `boxplot()`

Partitionner votre graphique en deux dans le sens horizontal.

```
vecteur = t(1:2)
layout(vecteur)
```

Dans le premier cadre de la partition, tracer les boîtes à moustache associé aux durées d'éruption conditionnellement aux temps d'attente entre deux éruptions

```
boxplot(faithful$eruption ~ faithful$waiting)
```

Dans le second cadre de la partition, utiliser le seuil calculé précédemment pour construire deux boîtes à moustache associé aux durées d'éruption conditionnellement aux temps d'attente entre deux éruptions. L'une codant les séisme court et l'autre codant les séisme long.

Pour ce faire créer une variable égale à court si waiting est inférieur au seuil et à long si waiting est supérieur au seuil.

```
Y = character(272)
Y[faithful$waiting > 63] = "long"
Y[faithful$waiting <= 63] = "court"
Y = as.factor(Y)
boxplot(faithful$eruption ~ Y)
```

Décorer le graphique (Couleur, label, titre) et rajouter un indicateur de densité.

```
boxplot(faithful$eruption ~ Y, col = c("yellow", "red"),
        main = "boxplot en fonction de la duree du seisme",
        xlab = "court vs long"
        )
rug(faithful$eruption, side = 2)
```

## Utilisation de la fonction `pie()`

On souhaite visualiser, via les boîtes à moustache, la proportion de séisme dont la durré est supérieur à 3 minutes.

```
Y = numeric(272)
Y[faithful$eruption > 3] = 1
Y[faithful$eruption <= 3] = 0

Pourcentage_court = length(Y[Y == 0])/nrow(faithful)
Pourcentage_long = length(Y[Y == 1])/nrow(faithful)
Seisme = c(Pourcentage_court, Pourcentage_long)
names(Seisme) = c("seisme long", "seisme court")
pie(Seisme, col = c("yellow", "red"), main = "duree des seismes", border = NA)
```

## Utilisation des packages grid et lattice

Charger le package lattice en mémoire

```
library(lattice)
```

Nous allons illustrer l'utilisation des fonctions du package lattice à l'aide du jeu de données `bordeaux_R.txt`.

Il s'agit d'un jeu de données qui fournit la qualité de vin de bordeaux (QUALITE) en fonction de 4 facteurs (TEMPERAT, SOLEIL, CHALEUR, PLUIE)

Télécharger le jeu de données `Bordeaux_R.txt` à l'adresse suivante [☐](#)

Charger en mémoire ce jeu de données sur [R](#).

```
A = read.table('/chemin/bordeaux_R.txt', header = TRUE, sep = '\t')
```

TEMPERAT = somme des températures moyennes journalières (°C)

SOLEIL = Durée d'insolation (h)

CHALEUR = Nombre de jours de grande chaleur

PLUIE = hauteur de pluie en (mm)

QUALITÉ☐: 1 = BON, 2 = MOYEN et 3 = MÉDIOCRE

On souhaite évaluer l'impact des différentes variables sur la qualité des vins de bordeaux.

Tracer les estimateurs de la densité de chacune des variables conditionnellement à la qualité. Qu'en concluez-vous☐

```
densityplot(~ A$TEMPERAT | A$QUALITE, xlab = "Temperature", col = "red")
```

```
densityplot(~ A$SOLEIL | A$QUALITE, xlab = "Soleil", col = "Yellow")
```

```
densityplot(~ A$CHALEUR | A$QUALITE, xlab = "Chaleur" , col = "green")
```

```
densityplot(~ A$PLUIE | A$QUALITE, xlab = "Pluie", col = "blue")
```

On peut interpréter l'influence de chaque variable sur la qualité du produit.

Par exemple☐:

A l'aide du premier graphique sur la température, on peut remarquer que les meilleurs vins résultent d'années de forte température

Effectuer la même analyse à l'aide des boîtes à moustaches conditionnées

```
bwplot(~ A$TEMPERAT | A$QUALITE, xlab = "Temperature", col = "red")
```

```
bwplot (~ A$SOLEIL | A$QUALITE, xlab = "Soleil", col = "Yellow")
```

```
bwplot (~ A$CHALEUR | A$QUALITE, xlab = "Chaleur" , col = "green")
```

```
bwplot (~ A$PLUIE | A$QUALITE, xlab = "Pluie", col = "blue")
```

On peut interpréter l'influence de chaque variable sur la qualité du produit.

Par exemple☐:

A l'aide du graphique sur la pluie, on peut remarquer que les meilleurs vins résultent d'années plutôt sèche

Visualiser les relations entre couple de variables conditionnellement à la qualité. Conclure

```
splom(~A[ 2:5] | A$QUALITE, pscale = 0)
```

Par exemple :

On remarque qu'une année peu pluvieuse jumelée à de forte chaleur fournira des vins de bonne qualité; ce qui ne semble pas surprenant!!

Prochain T.P. Vendredi 17 juin

Thème Analyse statistique avec 