

# Programmation R – Cours 1

Gaëlle LELANDAIS



Version du document : 26/11/2018, ce cours a été conçu  
avec Leslie REGAD

# Présentation de R

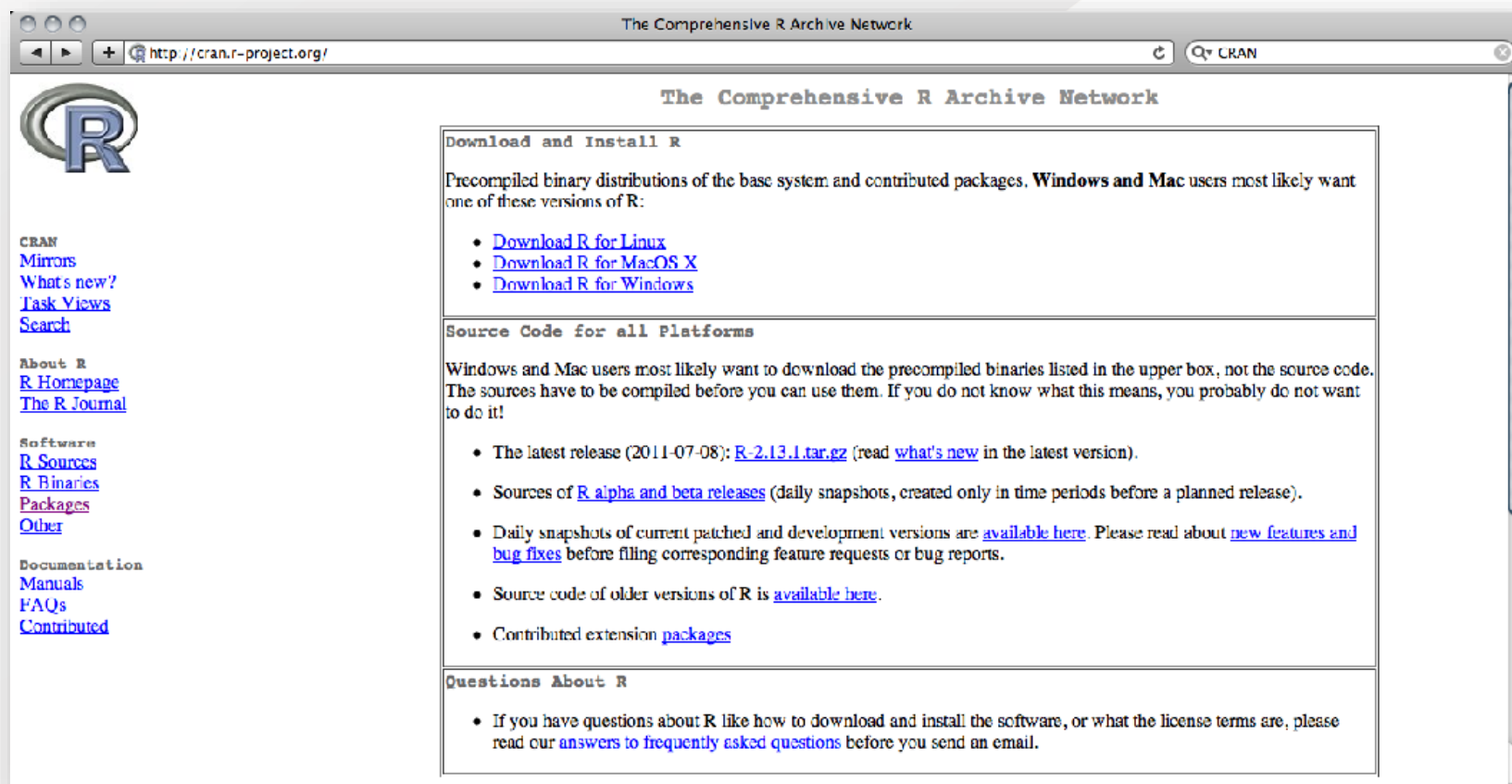
2

- ◉ Projet gratuit et participatif
- ◉ Fonctions statistiques et graphiques avancées
  - > Calculs sur des variables, tests statistiques, représentations histogrammes, etc.
- ◉ Librairies complémentaires
  - > Bio-informatique
  - > Data-mining
  - > Bases de données, etc.
- ◉ Interfaces possibles avec d'autres langages

# Site de référence : CRAN

3

<http://www.cran.r-project.org/>



# Démarrage / arrêt du logiciel

4

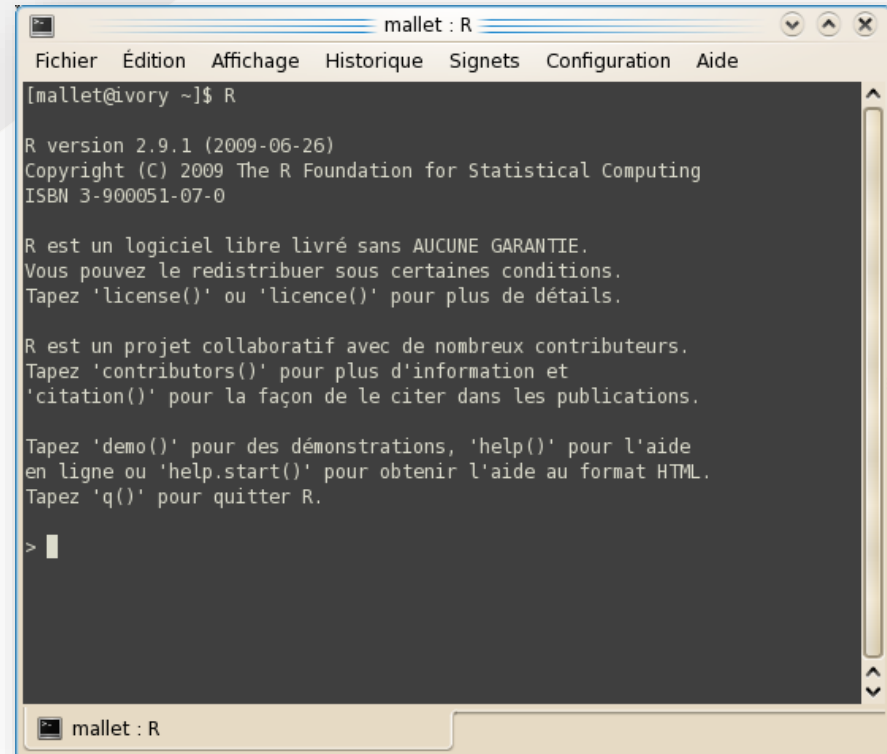
- Dans une console, taper « R »
- Noter le caractère de début d'une ligne de commande « > »

Exemple :

```
> 1+1/4 #[taper sur entrer]
[1] 1.25
> (1+1)/4
[1] 0.5
> 21.3*2
[1] 42.6
```

- Quitter la session en cours

```
> q()
Save workspace image? [y/n/c]
# [taper « n », pour « no »]
```



```
mallet : R
Fichier  Édition  Affichage  Historique  Signets  Configuration  Aide

[mallet@ivory ~]$ R

R version 2.9.1 (2009-06-26)
Copyright (C) 2009 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails.

R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.

Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.

> |
```

# Qu'est ce qu'une fonction ?

5

- Une fonction est un ensemble de commandes pré-programmées
- Une fonction se caractérise par :
  - > Son nom
  - > Ses arguments (informations préalables, nécessaires l'exécution de la fonction)

```
> cos(); mean(); var(); q() # exemple de noms de fonctions  
> sqrt(16) # 16 est ici un argument de la fonction sqrt()  
[1] 4
```

# Gérer son répertoire de travail

6

- ◉ Dans quel répertoire je travaille ?
  - > `getwd()`
- ◉ Changer de répertoire
  - > `setwd()`

```
> getwd()
[1] "/Users/gaellelelandais/Enseignements/Seance2"

> setwd("../Seance3")

> getwd()
[1] "/Users/gaellelelandais/Enseignements/Seance3"
```

# Obtenir de l'aide

7

- Appeler l'aide

- > ?NomFonction ; help(NomFonction); help.start();

> ?mean ; help(mean) ; help.search(« mean »)

- Sections de l'aide associée à une fonction

- > *Description* → A quoi sert la fonction ?
  - > *Usage* → Comment utiliser la fonction ?
  - > *Arguments* → Quels paramètres utilise la fonction en entrée ?
  - > *Details* → Description technique de la fonction
  - > *Value* → Quels paramètres sont retournés par la fonction en sortie ?
  - > *See also* → Existe-t-il des fonctions similaires ?
  - > *Example* → Cas concrets d'utilisation de la fonction

# Premières manipulations de variables

8

## ⦿ Affectation

```
> a = 1          # affectation dans les récentes versions de R  
> a <- 1         # affectation dans les anciennes versions de R
```

## ⦿ Utilisation / lecture

```
> a              # appel de la variable  
[1] 1            # le contenu de la variable est affiché  
> a + 1          # la variable peut être modifiée "à la volée"  
[1] 2
```

## ⦿ Remarque

- > A l'affichage un indice est noté entre les symboles « [ ] »



# Utilisation des vecteurs

9

- ⊙ Fonctions de création d'un vecteur
  - > `c()`, `x:y`, `seq()`, `rep()`, `append()`, etc.
- ⊙ Fonctions de manipulation d'un vecteur
  - > `data.class()`
  - > `length()`
  - > `sort()`, etc.

```
> c(1,2,3)
[1] 1 2 3
> 1:3
[1] 1 2 3
> seq(1,3)
[1] 1 2 3
> rep(1,3)
[1] 1 1 1
```

```
> data.class(c(1,2,3))
[1] "numeric"
> data.class(c("A","B","C"))
[1] "character"
> length(c(1,2,3))
[1] 3
> sort(c(4,5,2))
[1] 2 4 5
```

# Fonction « sample() »

10

## ◉ Exemple 1

- > Tirer aléatoirement 4 nombres dans un ensemble de valeurs comprises entre 1 et 40

```
> sample(1:40, 4)
[1] 26  6 25 34
> sample(1:40, 4, replace=TRUE) # tirage avec remise
[1]  7 33 27 27
```

## ◉ Exemple 2

- > Simuler 10 lancés d'une pièce de monnaie (les résultats possibles sont « pile » et « face »)

```
> sample(c("pile", "face"), 10, replace=TRUE, prob=c(0.4, 0.6))
[1] "pile" "pile" "face" "pile" "pile" "face" "face" "face"
"pile" "face"
```

# Fonction « `rnorm()` »

11

## ◉ Exemple 3

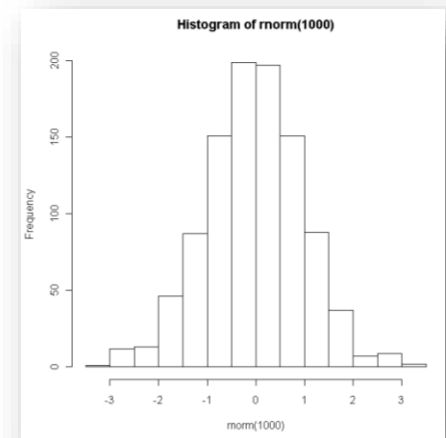
- > Tirer aléatoirement un ensemble de 10 nombres compatibles avec une distribution normale

```
> rnorm(10)
[1] 1.1451044 -1.1740811 2.1600010 0.8289392 -1.2881410
1.1022482 1.0495700 -0.4675296 0.3934182 1.0663837
```

- > Remarque : lorsque le nombre de valeurs tirées est grand, la « courbe en cloche » est retrouvée

```
> hist(rnorm(100))
```

—————→  
Par défaut, moyenne = 0  
et variance = 1



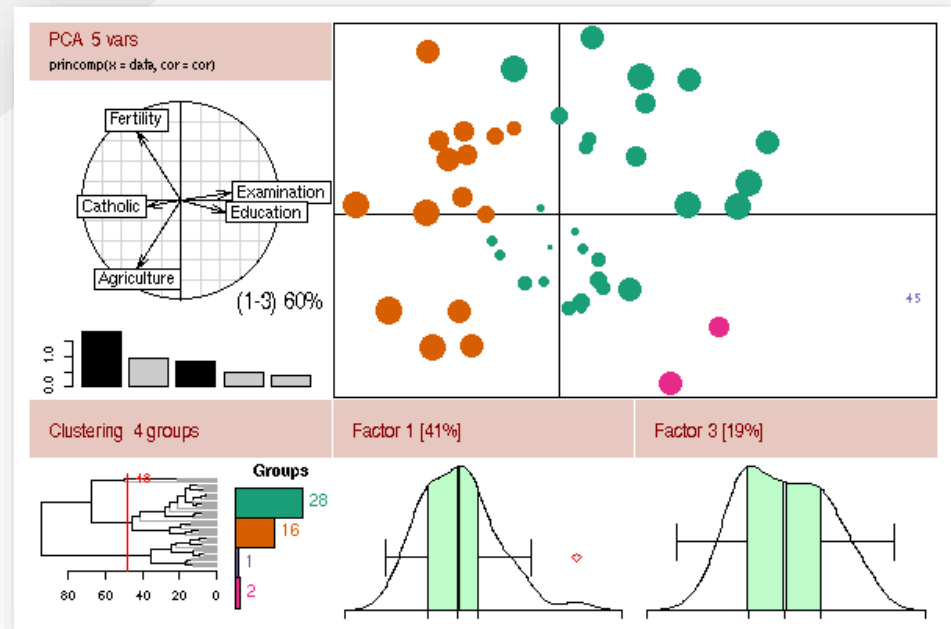
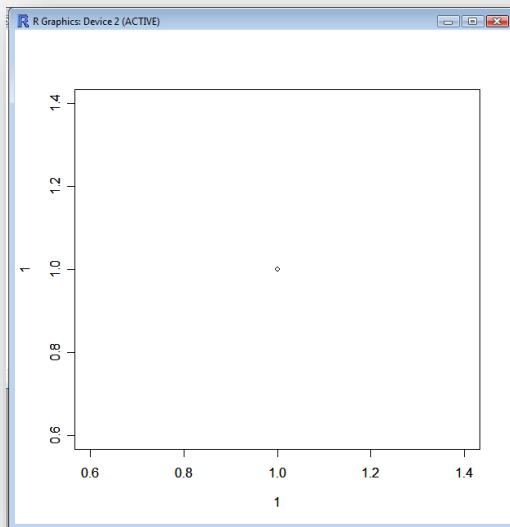
## 12

- Introduction à R -

# Représentations graphiques

13

- Possibilité de réaliser de très nombreuses représentations graphiques
  - Des plus simples aux plus élaborées



# Fonction « plot() »

14

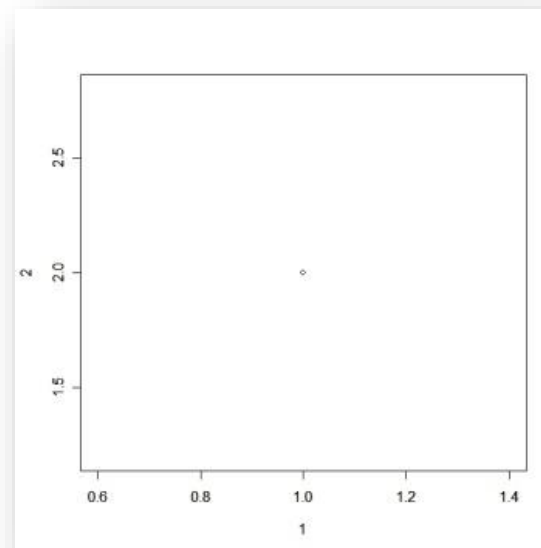
- Objectif

- > Fonction générique permettant de réaliser des représentations graphiques avec R

- Exemple

- > Représenter un point aux coordonnées  $x = 1$  et  $y = 2$

```
> plot(1,2)
```

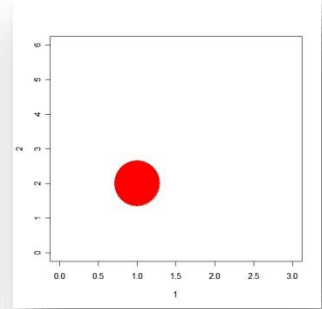
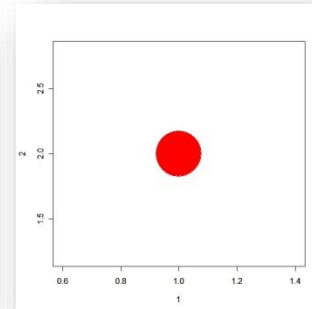
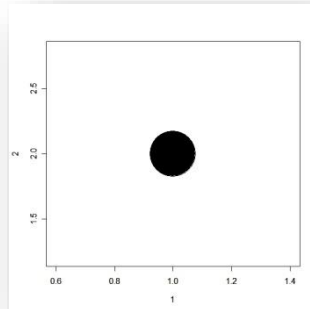
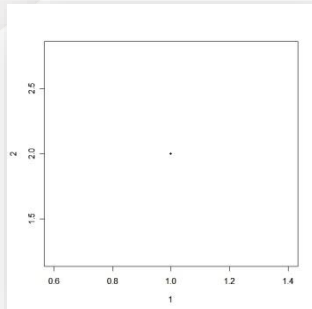
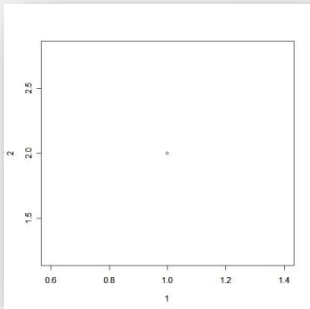


# Paramètres de la fonctions « plot() »

15

- Existence de très nombreux arguments qui permettent de
  - Modifier la forme, la couleur et la taille des points : *pch*, *col*, *cex*
  - Modifier les axes : *xlim*, *ylim*, *axis*, etc.
  - Ajouter des légendes : *xlab*, *ylab*, *title*, etc.
  - Et bien d'autres...

```
> plot(1,2) ; plot(1,2, pch = 20) ; plot(1,2, pch = 20, cex =  
20) ; plot(1,2, pch = 20, cex = 20, col = "red") ; plot(1,2,  
pch = 20, cex = 20, col = "red", xlim = c(0,3), ylim = c(0, 6))
```



# Superposition d'éléments sur le graphique « plot »

16

- ⊙ Ajout de courbes
  - > *lines()*, *points()*, *abline()*, etc.
- ⊙ Ajout d'une légende
  - > *legend()*

```
> plot(1,1)
> legend(1,1.2,c("c'est un point"), fill=T)
> abline(h=0.8)
> plot(1,1, main = « Mon graphique »)
> plot(1,1, xlab = « axe X », ylab = « axe Y »)
```



# Fonction « hist() »

17

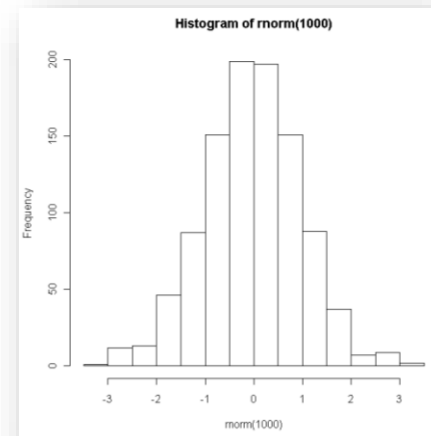
## Objectif

- > Représenter la distribution d'une variable aléatoire dans un échantillon
- > Abscisse : intervalles de valeurs pris par la VA
- > Ordonnée : effectifs observés dans chaque intervalle

## Exemple

- > Représenter l'histogramme de 100 valeurs choisies aléatoirement selon une loi normale

```
> hist(rnorm(100))
```

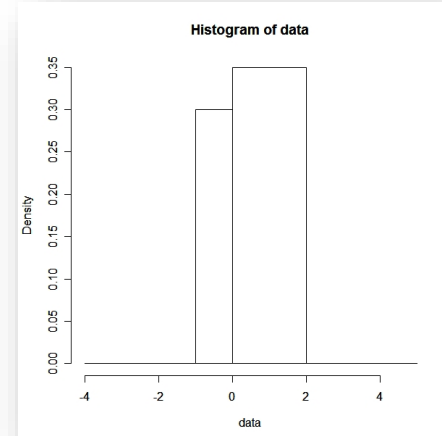
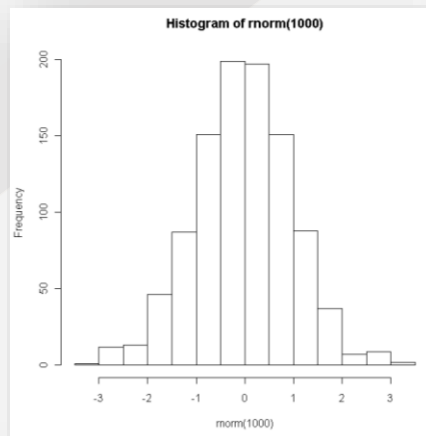


# Paramètres de la fonction « hist() »

18

- Existence de très nombreux paramètres qui permettent de
  - Modifier la largeur des « barres » de l'histogramme: *breaks*
  - Changer la couleur: *col* ; Etc.

```
> myData  
[1] 1.1138524 0.6422674 0.9551179 -0.2718710 -0.6115663 -  
0.6569689 0.5271689 1.6047569 0.2240304 0.7485861  
> hist(myData)  
> hist(myData, breaks = c(-4, -2, -1, 0, 2, 5))
```



# La fonction « `boxplot()` »

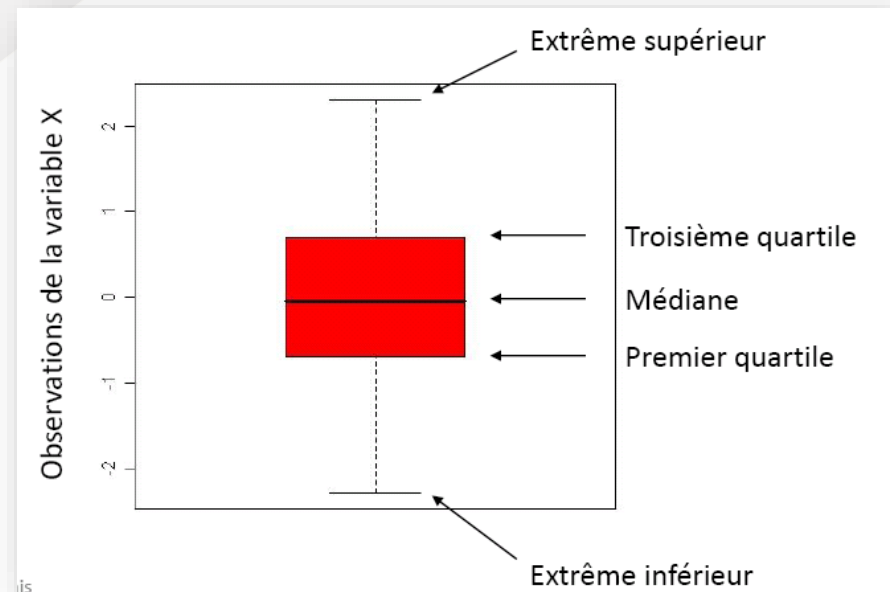
19

## ○ Définition

- Représentation graphique d'un ensemble de nombres. L'information est résumée en 5 valeurs

## ○ Exemple

```
> boxplot(myData, ylab =  
"observations de la variable x")
```



# Sauvegarde des graphiques

20

- ⊙ Possibilité de créer des fichiers images ou PDF
  - > `jpeg()`, `png()`, `bmp()`,
  - > `pdf()`
- ⊙ Ouverture et fermeture d'une fenêtre graphique
  - > `x11()`
  - > `dev.off()`
- ⊙ Organisation de la fenêtre graphique
  - > `par()`

```
> pdf("MonGraphique.pdf")  
> boxplot(myData)  
> dev.off()
```

# ⊙ Séance d'exercices