

Manipulation et visualisation de données dans R

Plan du cours

Chapitre 2 : Manipulation et visualisation de données dans R

2.1. Quelques fonctions utiles pour manipuler ses données

2.2. Quelques fonctions utiles pour visualiser ses données

2.3. Applications

 2.3.1. *À partir de données simulées*

 2.3.2. *À partir de données réelles*

 2.3.2.1 Issues de tableaux de données (data frame)

 2.3.2.2 Issues d'images (matrix)

Plan du cours

Chapitre 2 : Manipulation et visualisation de données dans R

2.1. Quelques fonctions utiles pour manipuler ses données

2.2. Quelques fonctions utiles pour visualiser ses données

2.3. Applications

2.3.1. *À partir de données simulées*

2.3.2. *À partir de données réelles*

2.3.2.1 Issues de tableaux de données (data frame)

2.3.2.2 Issues d'images (matrix)

Importer des données

Pour importer des données stockée dans un fichier texte (.txt) ou excel (.csv), on utilise les fonctions "read.table()" ou "read.csv()"

Un exemple avec la fonction "read.table()" :

- Syntaxe : `read.table(file, header, sep, dec, ...)`
 - *file* : nom du fichier avec l'extension (e.g., .txt)
 - *header* : argument de type booléen TRUE ou FALSE (par défaut) permettant d'indiquer si oui (TRUE) ou non (FALSE) la première ligne du fichier de données contient le nom des variables
 - *sep* : type de séparateur utilisé lors de l'enregistrement de vos données (e.g., virgule (","), point-virgule (";"), tabulation ("\t"), espace(" "))
 - *dec* : type de caractère utilisé (".") par défaut ou "," pour indiquer les décimales

Exemple à partir de données de tableau

Ouvrez Excel© et remplissez le tableau de données suivant :

Individus (id)	Poids (g)	Taille (mm)	Sexe (F/M)
----------------	-----------	-------------	------------

1	31	185	F
2	25	169	F
3	29	177	F
4	30	180	F
5	31	185	F
6	28	180	F
7	31	181	F
8	29	179	F
9	29	176	F
10	33	194	F
11	28	181	M
12	29	184	M
13	30	191	M
14	29	178	M
15	27	175	M
16	26	172	M
17	27	178	M
18	28	178	M
19	25	171	M
20	28	173	M

Souris des Cactus (*Peromyscus eremicus*)



Choix du format d'enregistrement

Enregistrez dans votre répertoire de travail le tableau de données au format ".txt" (e.g., "souris.txt") avec un séparateur de type tabulation :

The screenshot shows a Microsoft Excel spreadsheet titled "Classeur1 - Microsoft Excel". The spreadsheet contains a single sheet with data from row 1 to 22. Row 1 is a header with columns A, B, and C labeled "Individus (id)", "Poids (g)", and "Sexe (F/M)" respectively. Rows 2 through 22 contain data points. The "Sexe (F/M)" column uses the French convention where 'F' stands for female and 'M' for male.

A "Save As" dialog box is open in the foreground, centered over the spreadsheet. The dialog box has the title "Enregistrer sous" and shows the save location as "Bureau". The "Nom de fichier:" field contains "Classeur1". The "Type:" dropdown menu is open, displaying various file formats. The option "Texte (séparateur : tabulation)" is highlighted with a blue selection bar.

The Excel ribbon at the top includes tabs for Fichier, Accueil, Insertion, Mise en page, Formules, Données, Révision, Affichage, and Compléments. The Accueil tab is selected. The ribbon also features standard icons for Cut, Copy, Paste, Undo, Redo, and Save. The status bar at the bottom right shows "100%" and a small blue logo.

Importation dans R

Importez le fichier "souris.txt", contenant le tableau de données, à l'aide de la fonction "read.table()" :

```
> souris <- read.table("souris.txt", header=TRUE, sep="\t")
> str(souris)
'data.frame': 20 obs. of 4 variables:
 $ Individus..id.: int 1 2 3 4 5 ...
 $ Poids..g.      : int 31 25 29 30 31 ...
 $ Taille..mm.    : int 185 169 177 180 185 ...
 $ Sexe..F.M.    : Factor w/ 2 levels "F","M": 1 1 1 1 1 ...
```

Attention!!! R n'aime pas certains symboles du types parenthèses etc. Il pourrait donc être utile de renommer les variables

R a bien codé la variable "Sexe" en variable qualitative (factor) à deux modalités

Renommer des variables

Pour accéder aux noms des variables, utilisez la fonction "names()" et renommez ce vecteur de noms avec de nouveaux noms (e.g., "ind", "poids", "taille", "sex") :

```
> names(souris)
[1] "Individus..id." "Poids..g." "Taille..mm." "Sexe..F.M."
> names(souris) <- c("ind", "poids", "taille", "sex")
> str(souris)
'data.frame': 20 obs. of 4 variables:
 $ ind   : int  1 2 3 4 5 6 7 8 9 10 ...
 $ poids : int  31 25 29 30 31 28 31 29 29 33 ...
 $ taille: int  185 169 177 180 185 180 181 179 176 194 ...
 $ sex   : Factor w/ 2 levels "F","M": 1 1 1 1 1 1 1 1 1 1 ...
```

Recoder le type d'une variable

Pour basculer le type d'une variable (e.g., `integer`) vers un autre type (e.g., `numeric`), on utilise les fonctions "`as.logical()`", "`as.character()`", "`as.integer()`", "`as.numeric()`", "`as.double()`", "`as.factor()`", "`as.vector()`", "`as.matrix()`", "`as.array()`", "`as.data.frame()`", etc.

Utilisez ces fonctions pour coder la variable "ind" en type "character" et les variables "poids" et "taille" en type "numeric"

```
> souris$ind <- as.character(souris$ind)
> souris$poids <- as.numeric(souris$poids)
> souris$taille <- as.numeric(souris$taille)
> str(souris)
'data.frame': 20 obs. of 4 variables:
 $ ind   : chr  "1" "2" "3" "4" ...
 $ poids : num  31 25 29 30 31 28 31 29 29 33 ...
 $ taille: num  185 169 177 180 185 180 181 179 176 194 ...
 $ sex    : Factor w/ 2 levels "F","M": 1 1 1 1 1 1 1 1 1 1
```

Sélectionner un sous-ensemble de données

Pour sélectionner un sous-ensemble de données qui répondent à certaines conditions au sein d'un tableau de données, on peut utiliser la fonction "which()" :

➤ Syntaxe : `which(x, arr.ind)`

- *x* : un objet de type "logical" (TRUE/FALSE) dans un format "vector", "matrix" ou "array"
- *arr.ind* : argument de type booléen TRUE ou FALSE (par défaut) permettant d'afficher (TRUE) ou pas (FALSE) pour les formats "matrix" et "array" les coordonnées exactes de localisation des données pour lesquelles la condition se réalise (TRUE)

Le résultat renvoyé correspond à l'indice *i* de position de chacune des données pour lesquelles la condition se réalise (TRUE)

Où sont les femelles ?

Utiliser la fonction "which()" pour sélectionner les individus femelles au sein du jeu de données "souris" :

```
> souris$sex
[1] F F F F F F F F M M M M M M M M M M M M
Levels: F M
> souris$sex=="F"
[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE
FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> which(souris$sex=="F")
[1] 1 2 3 4 5 6 7 8 9 10
> index <- which(souris$sex=="F")
> souris[index, c("ind", "poids", "sex")]
   ind  poids sex
1    1     31   F
2    2     25   F
3    3     29   F
4    4     30   F
5    5     31   F
...
...
```

Construire une table de contingence

Pour calculer rapidement les effectifs d'individus par modalité d'une ou plusieurs variables, on utilise la fonction "table()" afin de construire une table de contingence :

- Syntaxe : `table(list, exclude, useNA, ...)`
 - *list* : un objet de type "factor" ou plusieurs objets de type "factor" stockés dans un objet de type "list"
 - *exclude* : vecteur contenant les modalités à exclure du calcul de la table de contingence, si besoin
 - *useNA* : argument à trois options permettant de gérer la présence de valeurs non-attribuées ou manquantes (NA) dans les données

Le résultat renvoyé est une table de contingence avec les effectifs par modalité croisée lorsque plus d'une variable qualitative est utilisée

Combiens de mâles et de femelles ?

À partir du jeu de données "souris", utilisez la fonction "table()" pour calculer les effectifs chez les deux sexes :

```
> table(souris$sex)
F   M
10  10
```

Ajouter une nouvelle variable qualitative (cf. "factor") de couleur du pelage "pel" à deux modalités : brun "B" ou gris "G" avec les trois premiers individus au pelage brun, les dix suivants au pelage gris et les sept derniers au pelage brun puis calculer la table de contingence sur les variables "sex" et "pel"

```
> souris[, "pel"] <- c(rep("B", 3), rep("G", 10), rep("B", 7))
> table(souris$sex, souris$pel)
B  G
F 3 7
M 7 3
```

Obtenir des informations par modalités

Pour calculer rapidement des informations statistiques (moyenne, variance, écart-type, etc.) par modalité d'une ou plusieurs variables, on utilise la fonction "tapply()" :

- Syntaxe : `tapply(x, index, fun, ...)`
 - *x* : un objet de type "numeric", bien souvent un vecteur de données quantitatives représentant une variable quantitative
 - *index* : un objet de type "factor" de la même taille que l'objet *x* et qui renseigne pour chaque observation du vecteur *x*, les modalités de la variable qualitative étudiée
 - *fun* : la fonction de calcul utilisée (e.g., moyenne, variance, écart-type, etc.) pour résumer l'information à l'échelle de la modalité

Poids moyens des mâles et des femelles ?

À partir du jeu de données "souris", utilisez la fonction "tapply()" pour calculer le poids moyen chez les deux sexes :

```
> tapply(souris$poids, souris$sex, mean)
  F      M
29.6 27.7
```

Même question mais cette fois-ci par modalité croisée des variables "sex" et "pel" :

```
> tapply(souris$poids, list(souris$sex, souris$pel), mean)
      B      G
F 28.33333 30.14286
M 27.14286 29.00000
```

Poids moyens des mâles et des femelles ?

NB : pour calculer les poids moyens par sexe, on peut également utiliser la fonction "lapply()" en travaillant sur une liste de vecteurs :

```
> F <- souris[souris$sex=="F", "poids"]  
> M <- souris[souris$sex=="M", "poids"]  
> lapply(list(F, M), mean)  
[[1]]  
[1] 29.6  
[[2]]  
[1] 27.7
```

Découper des chaînes de caractères

Pour découper des phrases ou du texte en fonction d'une séquence de caractères, on utilise la fonction "strsplit()" :

- Syntaxe : `strsplit(x, split, ...)`
 - *x* : un objet de type "character", bien souvent un vecteur de texte
 - *split* : une séquence de lettres, de mots ou de caractères recherchée dans le texte contenu dans l'objet *x* et permettant de redécouper ce texte en plus petites sections détectées avant et après la séquence de caractères

Le résultat renvoyé est une liste d'objets pour laquelle chaque élément est un vecteur de caractères contenant les sections de texte

Exemple à partir d'une liste d'espèces

Créez un vecteur de type "character" contenant la liste d'espèces suivante :

```
> liste_esp <- c("Peromyscus californicus", "Peromyscus  
eremicus", "Peromyscus interparietalis", "Peromyscus dickeyi")
```

À l'aide de la fonction "strsplit()", découpez ce vecteur de texte de manière à séparer le nom de genre "*Peromyscus*" du nom d'espèce :

```
> strsplit(liste_esp, " ")  
[[1]]  
[1] "Peromyscus" "californicus"  
[[2]]  
[1] "Peromyscus" "eremicus"  
[[3]]  
[1] "Peromyscus" "interparietalis"  
[[4]]  
[1] "Peromyscus" "dickeyi"
```

Sélectionner par séquence de caractères

Pour sélectionner une partie de vos données à partir d'une séquence de caractères reconnues au sein d'une chaîne de caractères, on utilise la fonction "grep()" :

- Syntaxe : `grep (pattern, x, ...)`
 - *pattern* : une séquence de lettres, de mots ou de caractères recherchée dans le texte qui est contenu dans l'objet *x*
 - *x* : un objet de type "character", bien souvent un vecteur contenant des éléments de texte et dans lequel chaque élément sera scanné pour détecter la séquence de caractères recherchée

Le résultat renvoyé correspond à l'indice *i* de position des éléments de l'objet *x* pour lesquelles la séquence de caractères recherchée a été détectée

Exemple de sélection

Sélectionnez les espèces dont le nom d'espèce se termine par "us" :

```
> sel <- lapply(strsplit(liste_esp, " "), function (x) x[2])
> sel
[[1]]
[1] "californicus"
[[2]]
[1] "eremicus"
[[3]]
[1] "interparietalis"
[[4]]
[1] "dickeyi"
> index <- grep("us", sel)
> liste_esp[index]
[1] "Peromyscus californicus" "Peromyscus eremicus"
```

Plan du cours

Chapitre 2 : Manipulation et visualisation de données dans R

2.1. Quelques fonctions utiles pour manipuler ses données

2.2. Quelques fonctions utiles pour visualiser ses données

2.3. Applications

2.3.1. *À partir de données simulées*

2.3.2. *À partir de données réelles*

2.3.2.1 Issues de tableaux de données (data frame)

2.3.2.2 Issues d'images (matrix)

Les principales fonctions

Pour tracer un graphique de données :

- `hist(x, ...)`
- `plot(x, y, ...)`
- `contour(x, y, z, ...)`
- `image(x, y, z, ...)`

Pour ajouter des éléments à un graphique existant :

- `abline(a, b, h, v, ...)`
- `points(x, y, ...)`
- `lines(x, y, ...)`
- `segments(x0, y0, x1, y1, ...)`
- `polygon(x, y, ...)`

Pour contrôler les paramètres graphiques :

- `par(font.axis, font.lab, font.main, font.sub, mfrow, ...)`

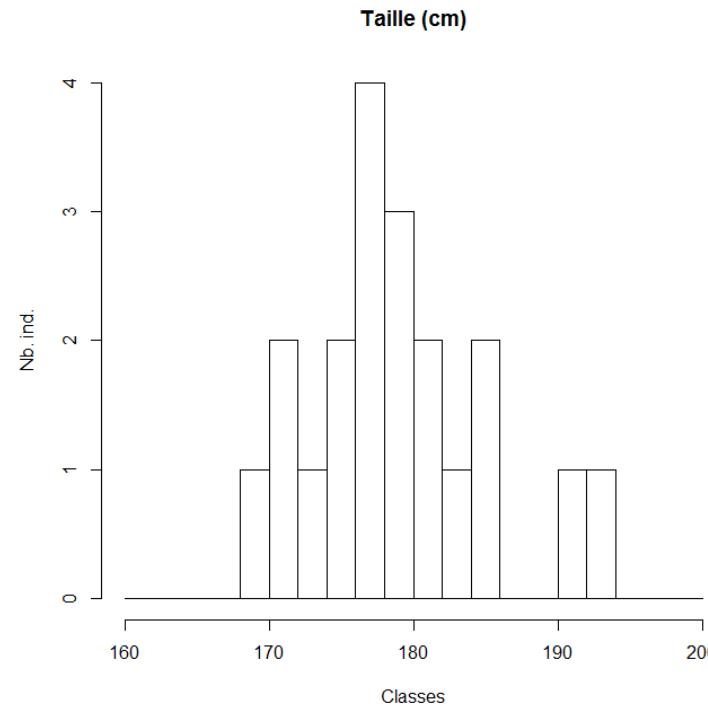
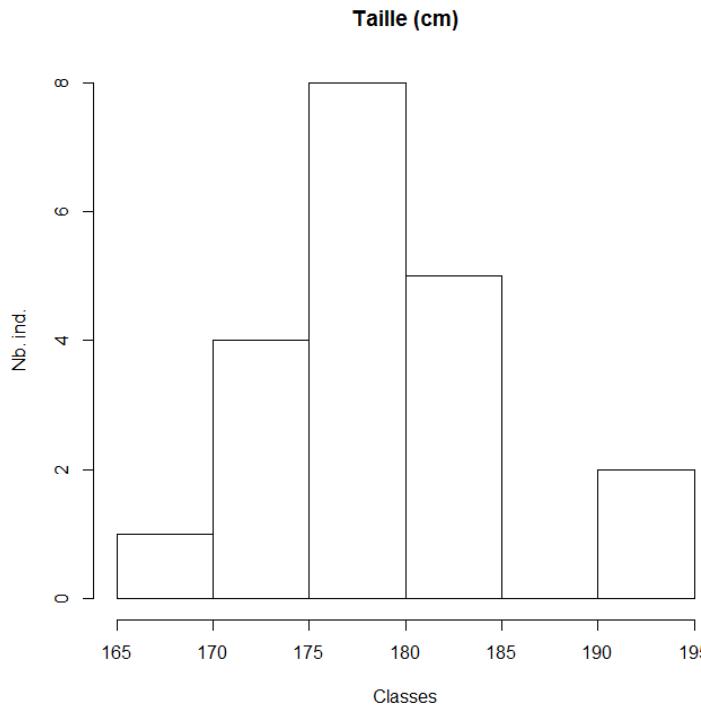
Pour contrôler la fenêtre d'affichage des graphiques :

- `windows(width, height, ...)`

Histogrammes

À l'aide de la fonction "hist()", tracez l'histogramme des tailles issues du jeu de données "souris" en utilisant des classes de taille de 2 cm :

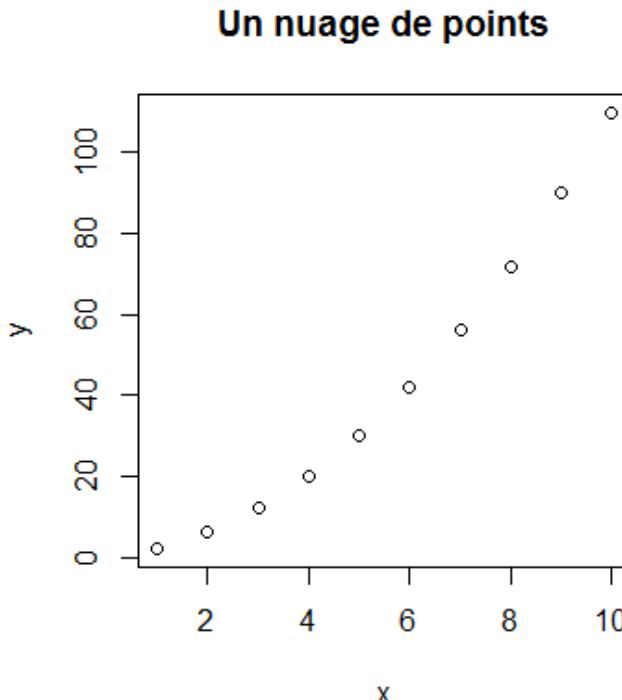
```
> T <- souris$taille  
> hist(T, main="Taille (cm)", xlab="Classes", ylab="Nb. ind.")  
> hist(T, breaks=seq(160, 200, 2), main="Taille (cm)",  
xlab="Classes", ylab="Nb. ind.")
```



Nuages de points

La fonction "plot()" permet de tracer un nuage de points entre deux variables qualitatives et de visualiser d'éventuelles relations :

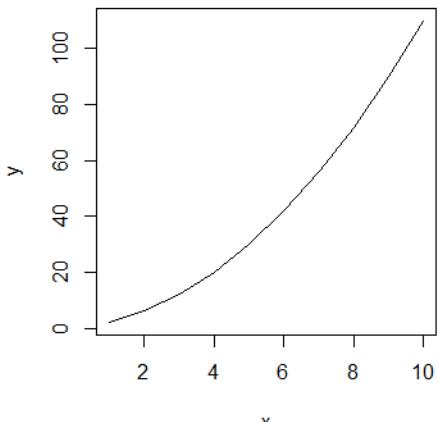
```
> x <- 1:10  
> y <- x+x^2  
> plot(x, y, main="Un nuage de points", type="p")
```



Plusieurs types de représentation

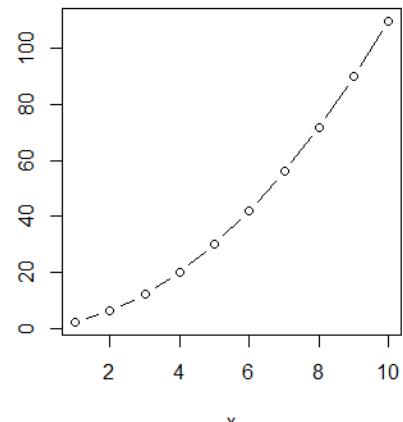
➤ type="l"

Un nuage de points



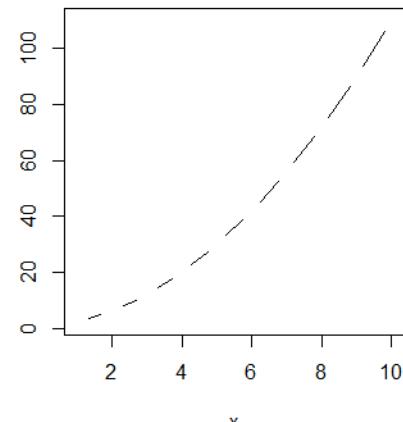
➤ type="b"

Un nuage de points



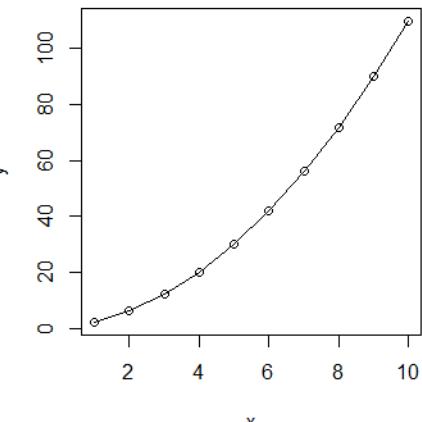
➤ type="c"

Un nuage de points



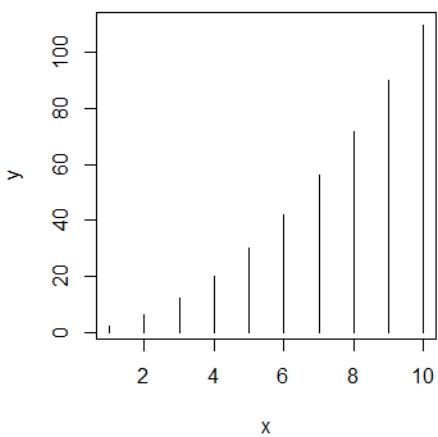
➤ type="o"

Un nuage de points



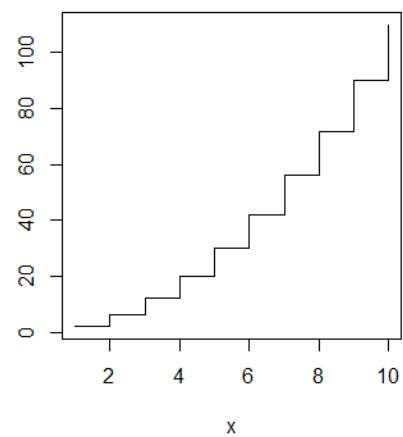
➤ type="h"

Un nuage de points



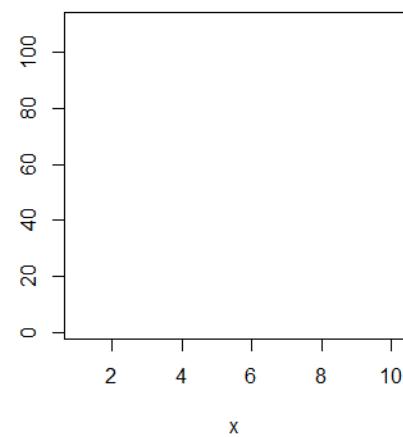
➤ type="s"

Un nuage de points



➤ type="n"

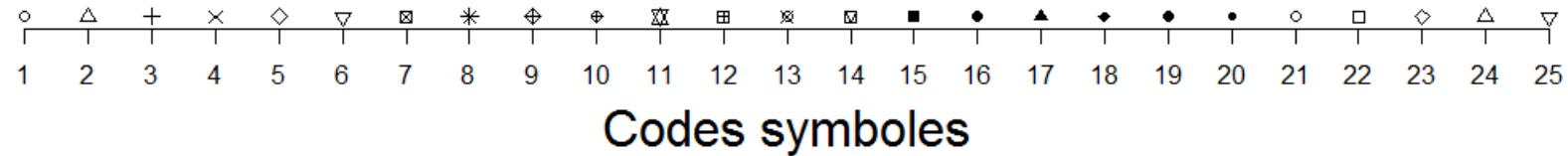
Un nuage de points



Les symboles

L'argument "pch" de la fonction "plot()" permet de changer la forme des symboles du nuage de points :

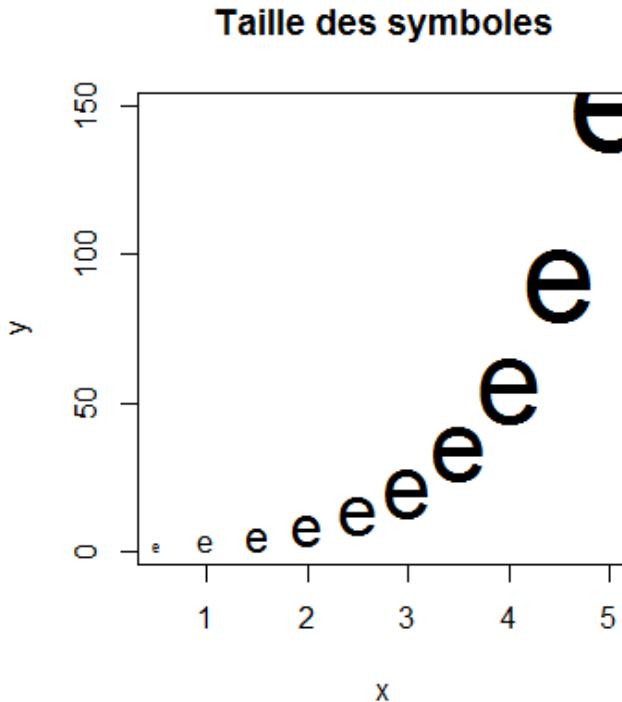
```
> x <- 1:25  
> y <- rep(1, 25)  
> plot(x, y, pch=x, axes=FALSE, xlab="Codes symboles", ylab="",  
cex.lab=2)  
> axis(side=1, at=seq(1, 25, 1))
```



La taille des symboles

L'argument "cex" de la fonction "plot()" permet de changer la taille des symboles du nuage de points :

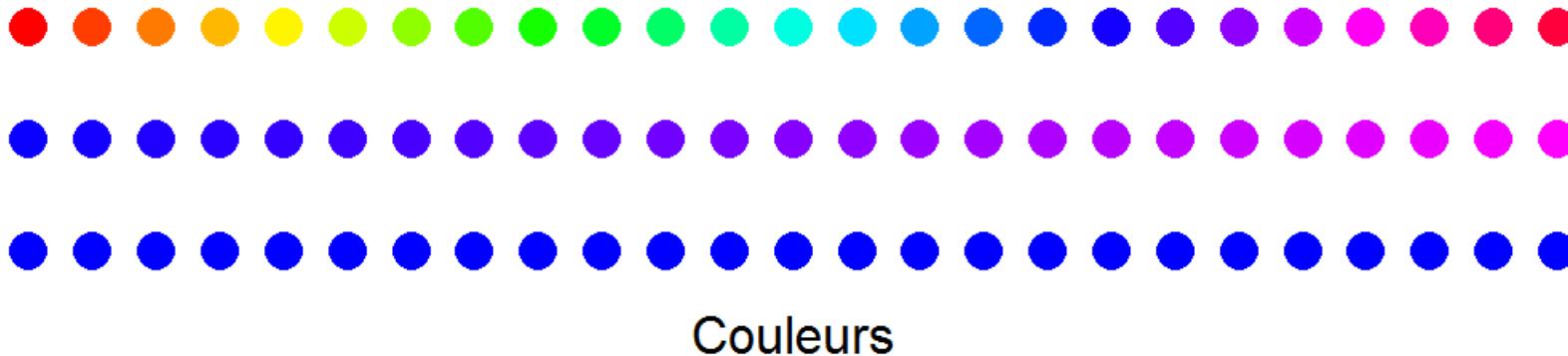
```
> x <- seq(0.5, 5, 0.5)
> y <- exp(x)
> plot(x, y, cex=x, pch="e", main="Taille des symboles")
```



La couleur des symboles

L'argument "col" de la fonction "plot()" permet de changer la couleur des symboles soit en spécifiant la couleur par son nom (e.g., "blue") soit en utilisant les fonctions "rainbow()" ou "rgb()":

```
> x <- 1:25
> y1 <- rep(1, 25)
> y2 <- rep(2, 25)
> y3 <- rep(3, 25)
> plot(x, y1, col="blue", axes=FALSE, xlab="Couleurs", ylab="",
cex.lab=2, cex=5, pch=20, ylim=c(0, 4))
> points(x, y2, col=rgb(x/max(x), rep(0, 25), rep(1, 25)),
cex=5, pch=20)
> points(x, y3, col=rainbow(length(x)), cex=5, pch=20)
```



Une palette de couleurs

La fonction "colors()" permet d'afficher la liste des noms de couleurs actuellement disponible dans R :

```
> colors()
[1] "white"          "aliceblue"       "antiquewhite"
[4] "antiquewhite1"  "antiquewhite2"  "antiquewhite3"
[7] ...
```

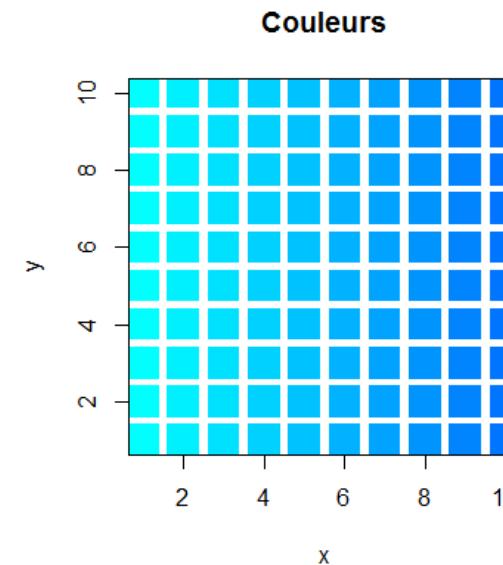
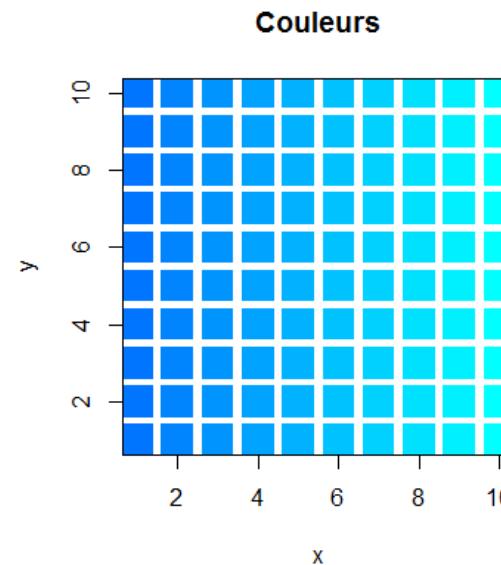
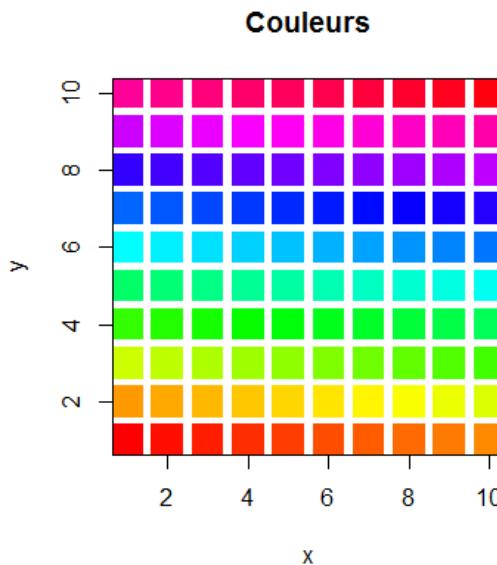
La fonction "palette()" permet de connaître ou de modifier la palette de couleurs actuellement utilisée dans l'argument "col" de la fonction "plot()" quand cet argument prends des valeurs numériques :

```
> palette()
[1] "black" "red"  "green3" "blue" "cyan" "magenta" "yellow"
[8] "gray"
> palette(rgb(0, seq(0, 1, length.out=10), 1))
> palette()
[1] "blue" "#001cff" "#0039ff" "#0055ff" "#0071ff" "#008e
[7] "#00aaaf" "#00c6ff" "#00e3ff" "cyan"
```

Jouer avec la palette de couleur

Exemple d'utilisation de la fonction "palette()" en combinaison avec l'argument "col" de la fonction "plot()":

```
> x <- rep(1:10, 10)
> y <- rep(1:10, each=10)
> palette(rainbow(length(x)))
> plot(x, y, pch=15, cex=3, col=1:100, main="Couleurs")
> plot(x, y, pch=15, cex=3, col=51:60, main="Couleurs")
> plot(x, y, pch=15, cex=3, col=rev(51:60), main="Couleurs")
```

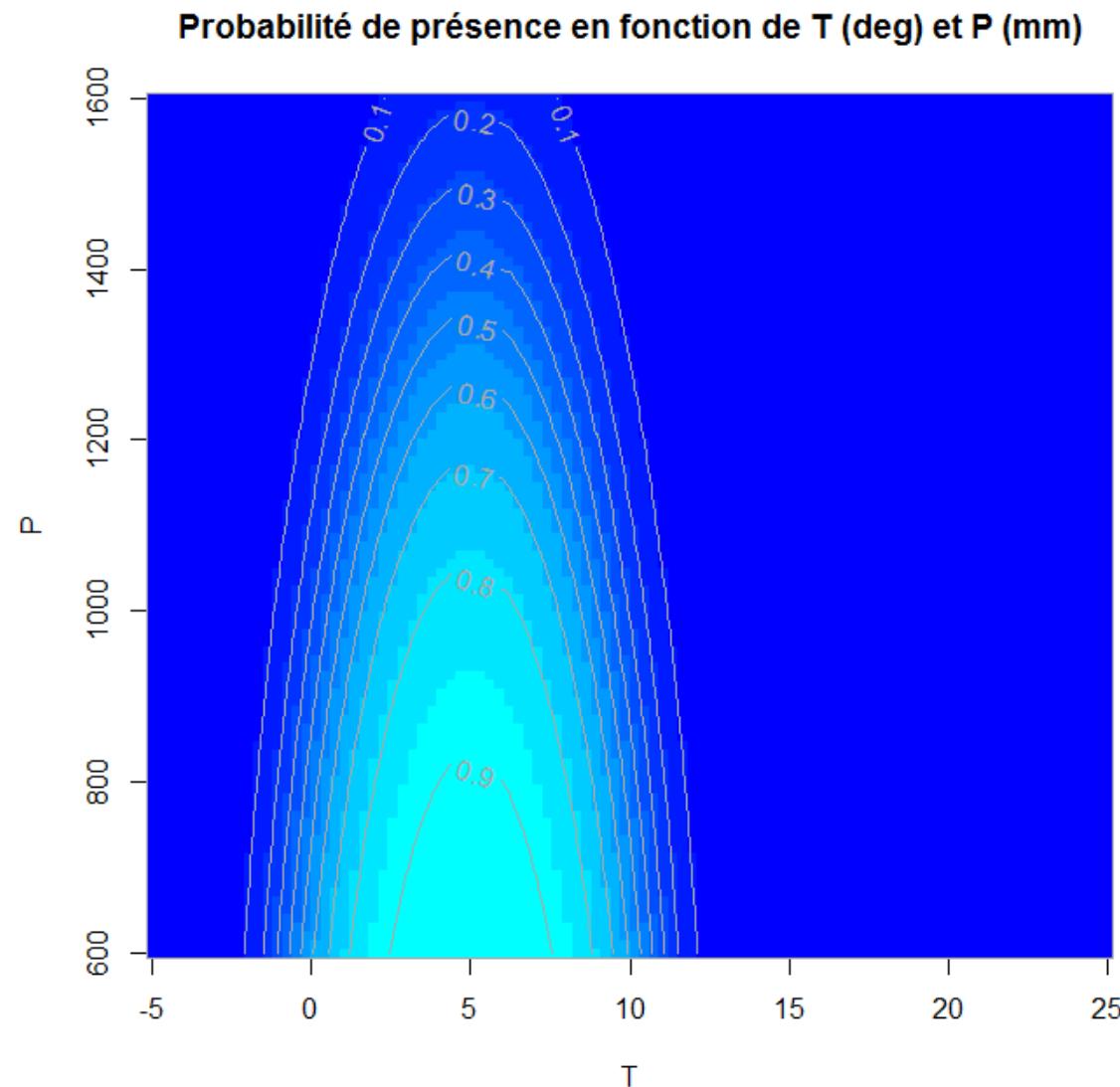


Images et isoclines

Les fonctions "image()" et "contour()" sont les fonctions à utiliser pour visualiser une troisième dimension dans un plan :

```
> x <- rep(seq(-5, 25, length.out=100), 100)
> y <- rep(seq(600, 1600, length.out=100), each=100)
> z <- 1/(1+exp(-(1+x-0.1*x^2+10^-4*y-2*10^-6*y^2)))
> z <- matrix(z, ncol=100, nrow=100)
> T <- seq(-5, 25, length.out=100)
> P <- seq(600, 1600, length.out=100)
> image(T, P, z, col=rgb(0, seq(0, 1, 0.1), 1), axes=FALSE)
> contour(T, P, z, add=TRUE, col="darkgrey", labcex=1)
> axis(side=1)
> axis(side=2)
> box(col="darkgrey")
> title("Probabilité de présence en fonction de T (deg) et P (mm)")
```

Images et isoclines

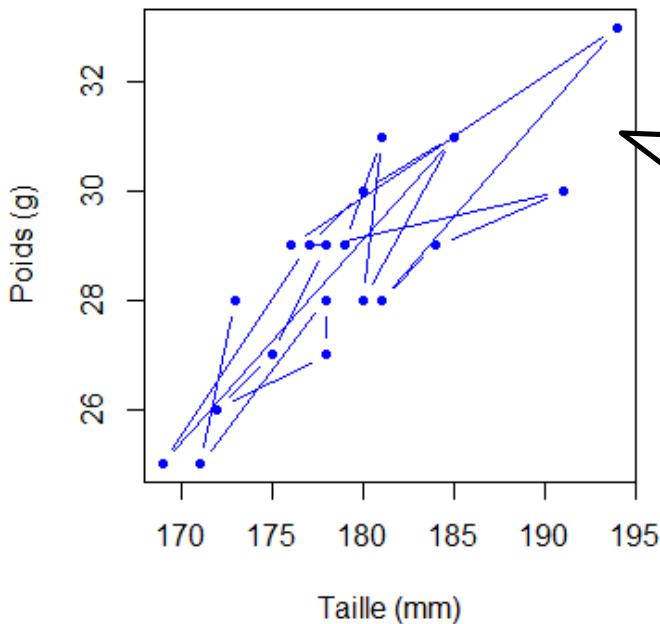


À vous de jouer

À partir du jeu de données "souris" et de la fonction "plot()", tracez le nuage de points du poids en fonction de la taille :

```
> P <- souris$poids  
> plot(T, P, xlab="Taille (mm)", ylab="Poids (g)", main="Souris  
des cactus", col="blue", pch=20, type="b")
```

Souris des cactus

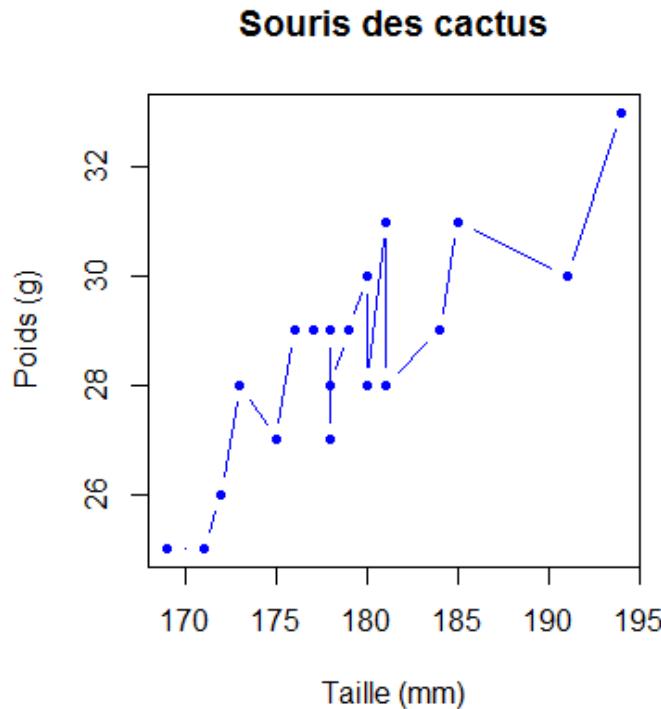


Attention!!! Sur ce type de représentation, il est nécessaire d'ordonner les valeurs le long de l'axe des abscisses

À vous de jouer

Utilisez la fonction "order()" pour ordonner les données le long de l'axe des abscisses (cf. taille) :

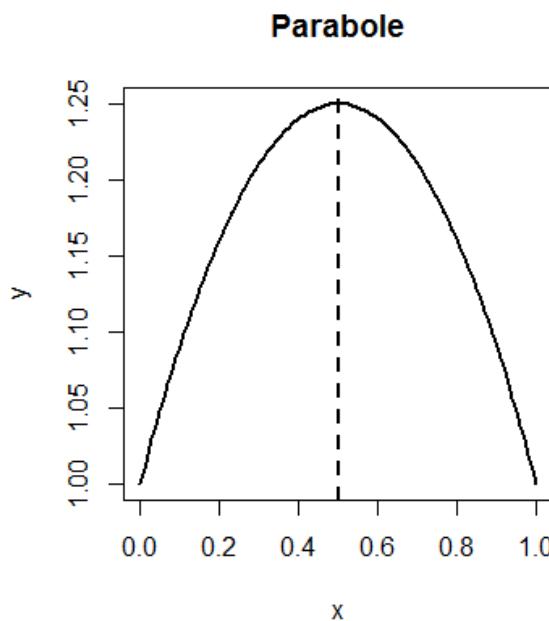
```
> plot(T[order(T)], P[order(T)], xlab="Taille (mm)",  
ylab="Poids (g)", main="Souris des cactus", col="blue", pch=20,  
type="b")
```



Ajouter des éléments à un graphique

Exemples d'éléments simples que l'on peut ajouter à un graphique existant :

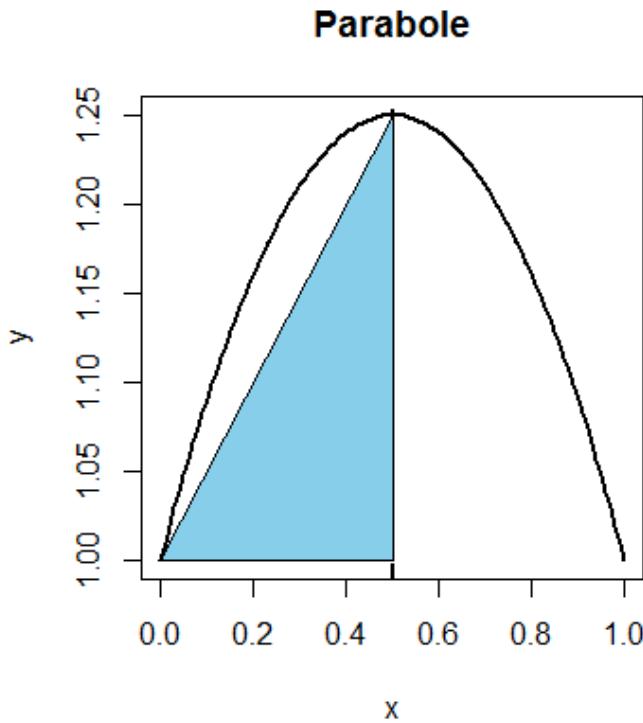
```
> x <- seq(0, 1, length.out=100)
> y <- 1+x-x^2
> plot(x, y, type="n", main="Parabole")
> lines(x, y, lwd=2)
> abline(v=0.5, lty=2, lwd=2)
```



Tracer l'aire sous une courbe

La fonction "polygon()" est très utile pour représenter l'aire sous la courbe par la méthode des trapèzes :

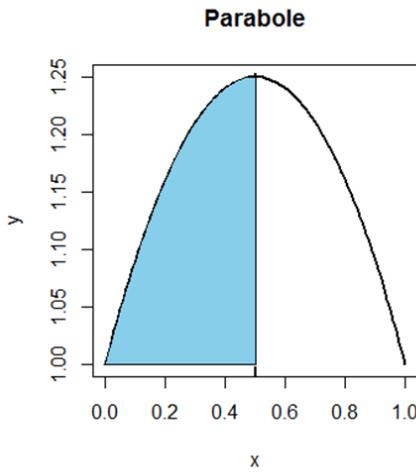
```
> cord.x <- c(0, 0.5, 0.5)
> cord.y <- c(1, 1.25, 1)
> polygon(x, y, col="skyblue")
```



Tracer l'aire sous une courbe

Pour affiner la représentation de l'aire et coller à la courbe, il suffit d'augmenter le nombre de trapèzes :

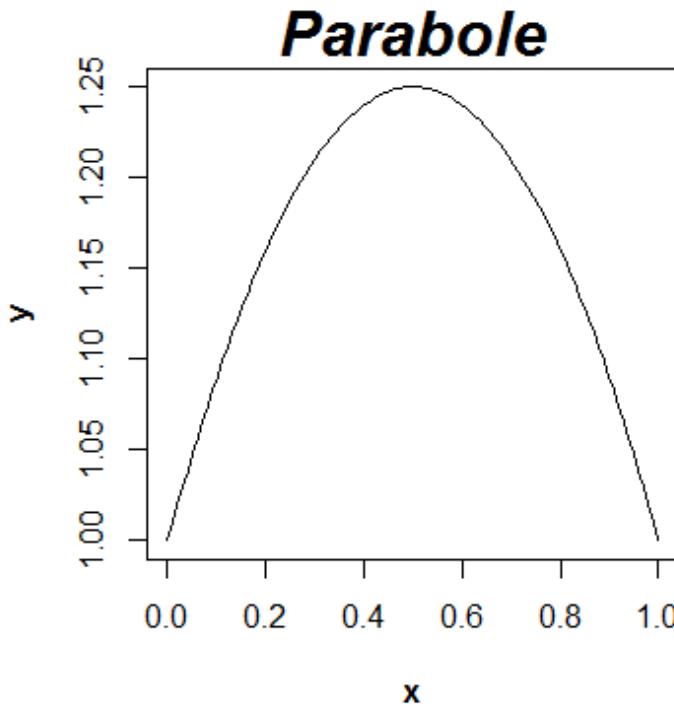
```
> f <- function(x)
  {
    y <- 1+x-x^2
    return(y)
  }
> cord.x <- c(0, seq(0, 0.5, 0.01), 0.5)
> cord.y <- c(1, f(seq(0, 0.5, 0.01))), 1)
> polygon(cord.x, cord.y, col="skyblue")
```



Customiser vos graphiques

La fonction "par()" permet, avec plus de 70 arguments aux options multiples, de personnaliser vos graphiques à l'infini pour un meilleur rendu esthétique :

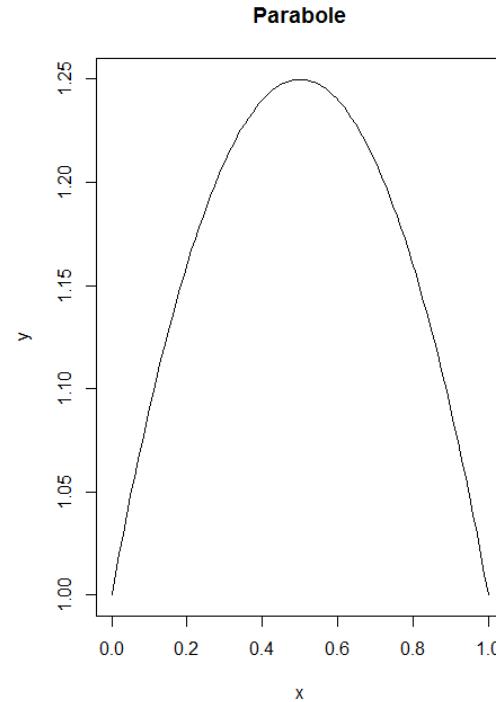
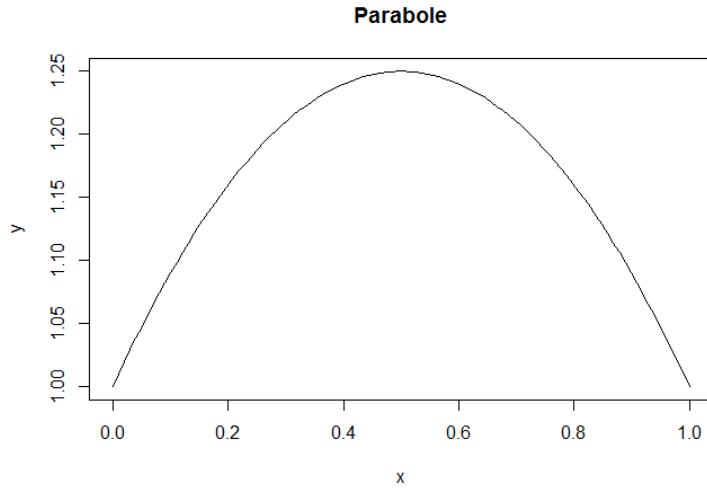
```
> par(font.axis=1, font.lab=2, font.main=4, cex.axis=1,  
cex.lab=1, cex.main=2, mar=c(5,5,2,1), las=0)  
> plot(x, y, type="l", main="Parabole")
```



Contrôler la fenêtre graphique

La fonction "windows()" permet de régler la taille (largeur et hauteur) de la fenêtre d'affichage graphique :

```
> windows(7, 5)
> plot(x, y, type="l", main="Parabole")
> windows(5, 7)
> plot(x, y, type="l", main="Parabole")
```



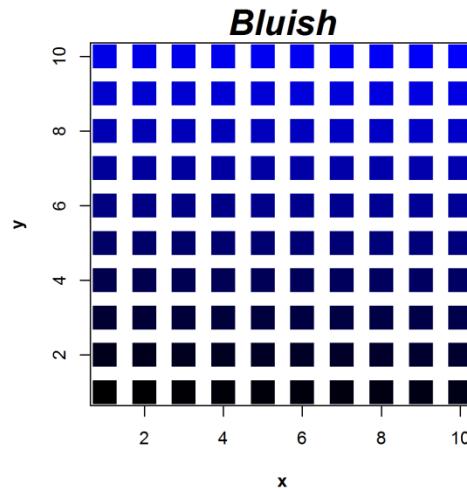
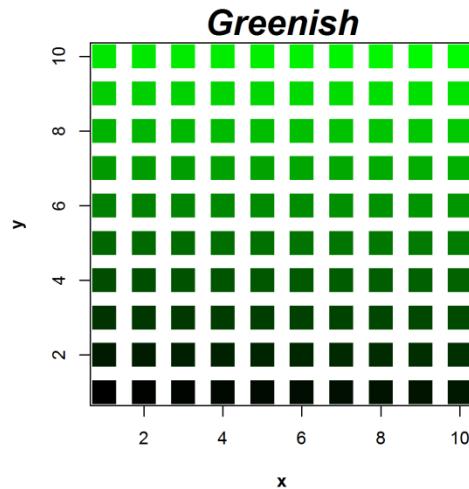
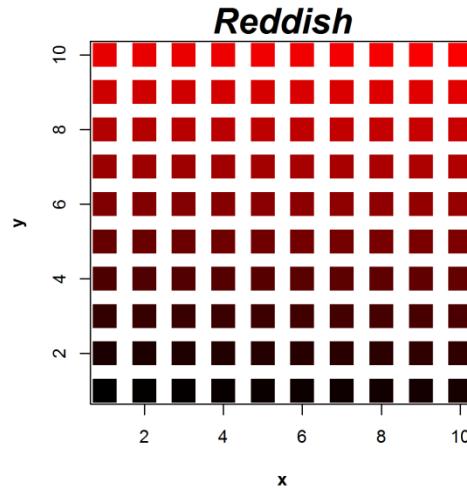
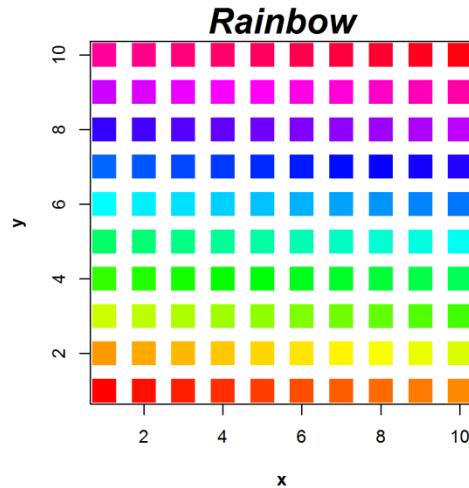
NB: imprimer un graphique

Les fonctions "pdf()", "bmp()", "jpeg()", "tiff()", et "png()" permettent de sauvegarder un graphique dans le répertoire de travail :

```
> tiff("Figure.tif", width=8, height=8, bg="white", units="in",
res=300)
> par(mfrow=c(2, 2), font.axis=1, font.lab=2, font.main=4,
cex.axis=1, cex.lab=1, cex.main=2, mar=c(5,5,2,1), las=0)
> x <- rep(1:10, 10)
> y <- rep(1:10, each=10)
> plot(x, y, pch=15, cex=3, col=palette(rainbow(length(x))),
main="Rainbow")
> plot(x, y, pch=15, cex=3, col=rgb(seq(0, 1, 0.01), 0, 0),
main="Reddish")
> plot(x, y, pch=15, cex=3, col=rgb(0, seq(0, 1, 0.01), 0),
main="Greenish")
> plot(x, y, pch=15, cex=3, col=rgb(0, 0, seq(0, 1, 0.01)),
main="Bluish")
> dev.off()
```

NB: imprimer un graphique

Il ne reste plus qu'à ouvrir le fichier nommé "Figure.tif" et situé dans le répertoire de travail :



Plan du cours

Chapitre 2 : Manipulation et visualisation de données dans R

2.1. Quelques fonctions utiles pour manipuler ses données

2.2. Quelques fonctions utiles pour visualiser ses données

2.3. Applications

2.3.1. *À partir de données simulées*

2.3.2. *À partir de données réelles*

2.3.2.1 Issues de tableaux de données (data frame)

2.3.2.2 Issues d'images (matrix)

Exercice 1

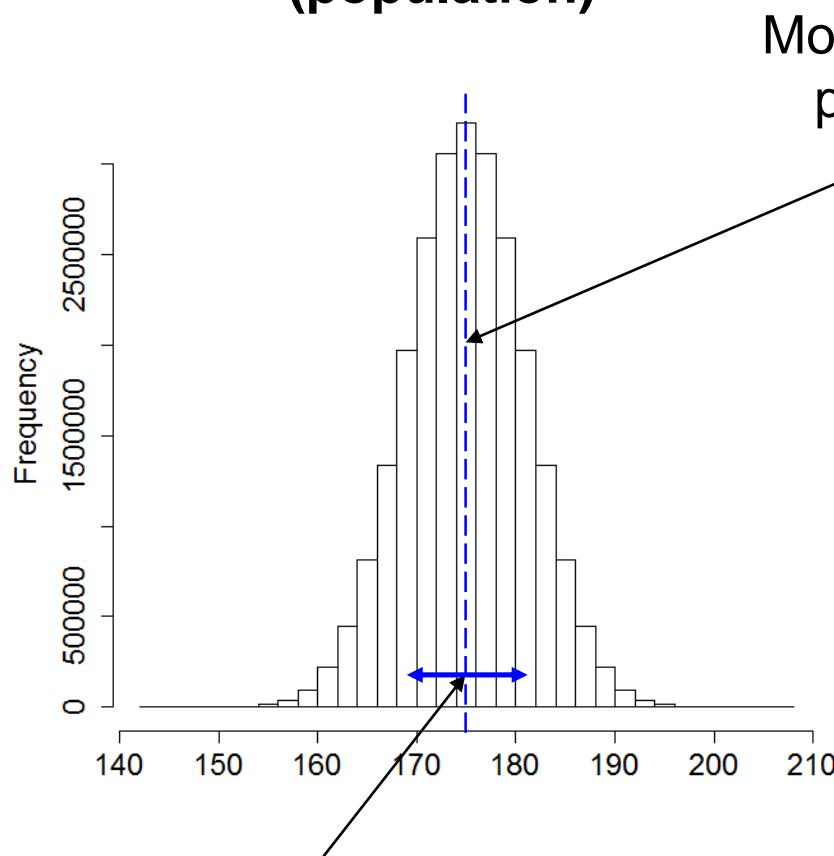
1. À l'aide de la fonction "rnorm()" commencez par créer la population "THpop2012" de 24,4 millions d'homme adulte observée en France en 2012 et dont la taille (cm) suit une loi Normale de moyenne 175 cm et d'écart type 6 cm
2. Tracez l'histogramme des taille (cm) en mode densité par tranche de 1 cm et à l'aide de la fonction "dnorm()", superposez-y la courbe de densité de la loi Normale de moyenne 175 cm et d'écart type 6 cm
3. À l'aide de la fonction "sample()" tirez aléatoirement et sans remise 10000 individus issus de la population "THpop2012" et stockez le résultat de ce tirage dans un vecteur nommé "ech1000taille10"
4. Transformez ce vecteur en une matrice de taille 1000*10 en utilisant un remplissage par ligne

Exercice 1

5. À l'aide de la fonction "apply()", calculez les 1000 moyennes des 1000 lignes et stockez le résultat dans un objet nommé "moy10"
6. Tracez l'histogramme de la distribution des 1000 moyennes et y ajouter une ligne verticale représentant la moyenne de la population d'origine. Que pensez vous de la moyenne d'un l'échantillon de taille 10 en tant qu'estimateur de la moyenne de la population ?
7. À l'aide de la fonction "sd()", calculez l'écart type des 1000 moyennes stockées dans l'objet "moy10"
8. Sachant que l'écart type de la population d'origine est de 6 cm et que la taille d'un échantillon est de 10, calculez l'erreur standard

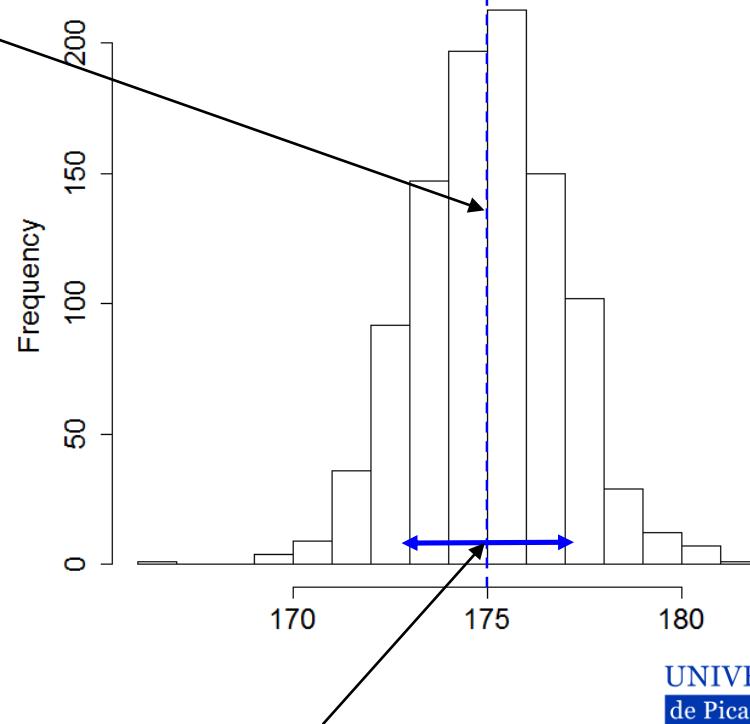
Exercice 1

Taille (cm) de 24,4 millions d'hommes adulte en France (population)



Ecart type des données
autour de la moyenne

Moyennes des tailles (cm) de 1000 échantillons de 10 individus chacun



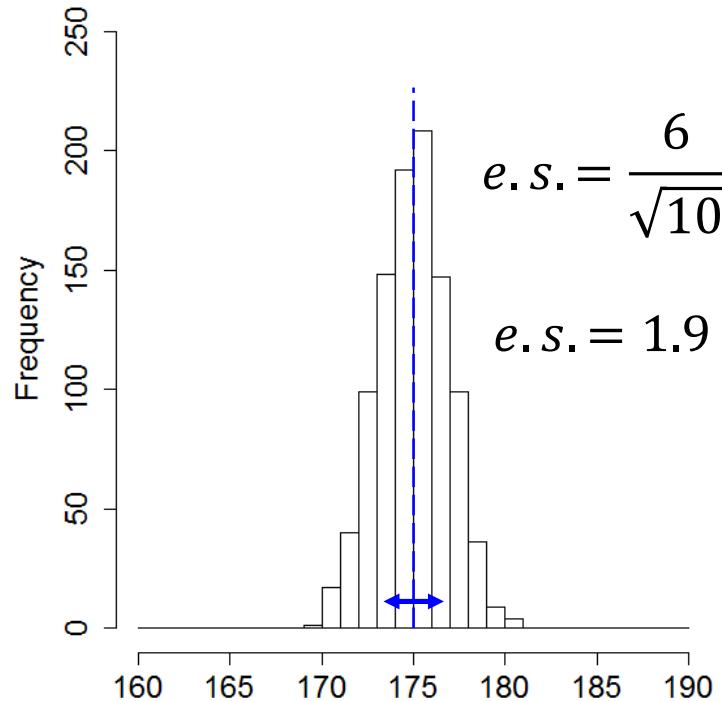
Ecart type des moyennes
= erreur standard

Exercice 2

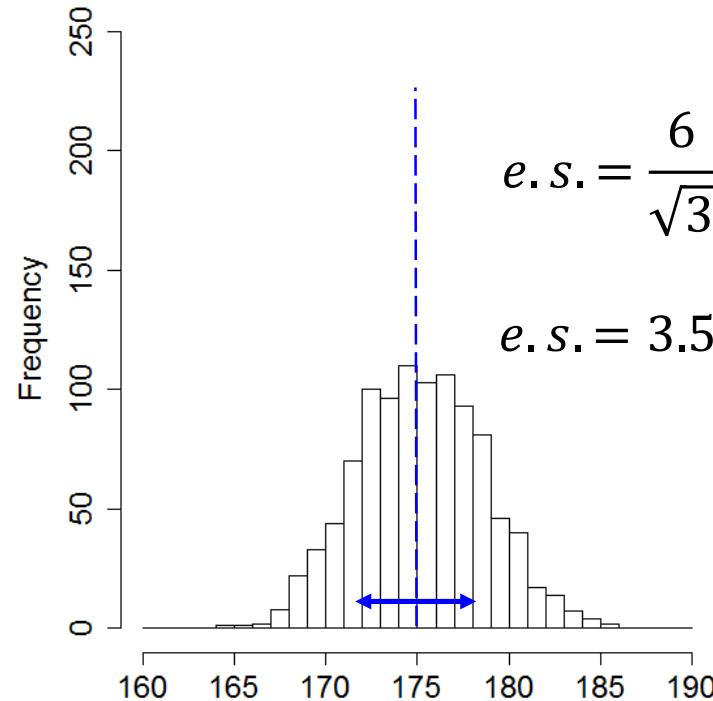
1. Répétez cette exercice pour 1000 échantillons chacun composé de 3 individus
2. Comparez la distribution des moyennes des 1000 échantillons de taille 3 avec celle des 1000 échantillons de taille 10, que constatez-vous ?

Exercice 2

Moyennes de 1000
échantillons de taille 10



Moyennes de 1000
échantillons de taille 3



L'erreur standard de la moyenne mesure la précision de la moyenne et cette précision augmente avec la taille de l'échantillon

Plan du cours

Chapitre 2 : Manipulation et visualisation de données dans R

2.1. Quelques fonctions utiles pour manipuler ses données

2.2. Quelques fonctions utiles pour visualiser ses données

2.3. Applications

2.3.1. *À partir de données simulées*

2.3.2. *À partir de données réelles*

2.3.2.1 Issues de tableaux de données (data frame)

2.3.2.2 Issues d'images (matrix)

Exercice 3

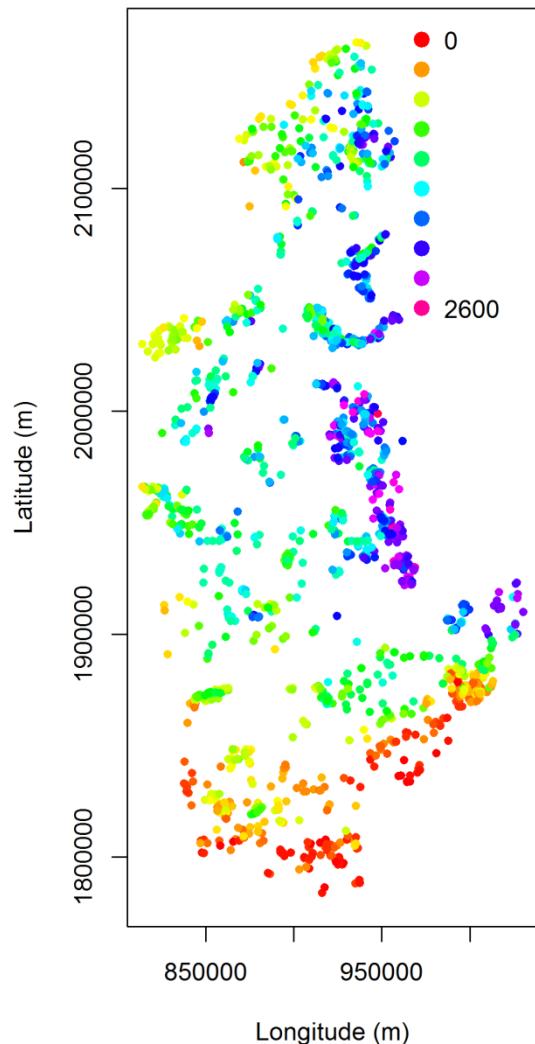
1. Importez le fichier "eco.txt", sauvegardé en mode tabulation, dans R
2. Vérifiez le type de chaque variable et modifiez-le si nécessaire
3. Calculez le nombre de relevé par "periode" (1 : 1905-1985, 2 : 1986-2005) et par "massif" ("alpes", "corse", "jura", "massifcentral", "pyrenees", "vosges")
4. Représentez la carte de distribution des relevés à l'aide des variables "xLamb2" et "yLamb2" représentants les coordonnées géographiques (m) dans le système de projection Lambert Zone II (NTF Paris)
5. Faites variez la taille du symbole de relevé en fonction de la variable "alti" d'altitude (m) et utilisez le code couleur suivant pour distinguer les 6 massifs montagneux sur cette carte : orange ("alpes"), rouge ("corse"), bleu ("jura"), vert ("massifcentral"), marron ("pyrenees") et rose ("vosges")
6. Ajoutez une légende à votre carte à l'aide de la fonction "legend()"

Exercice 3

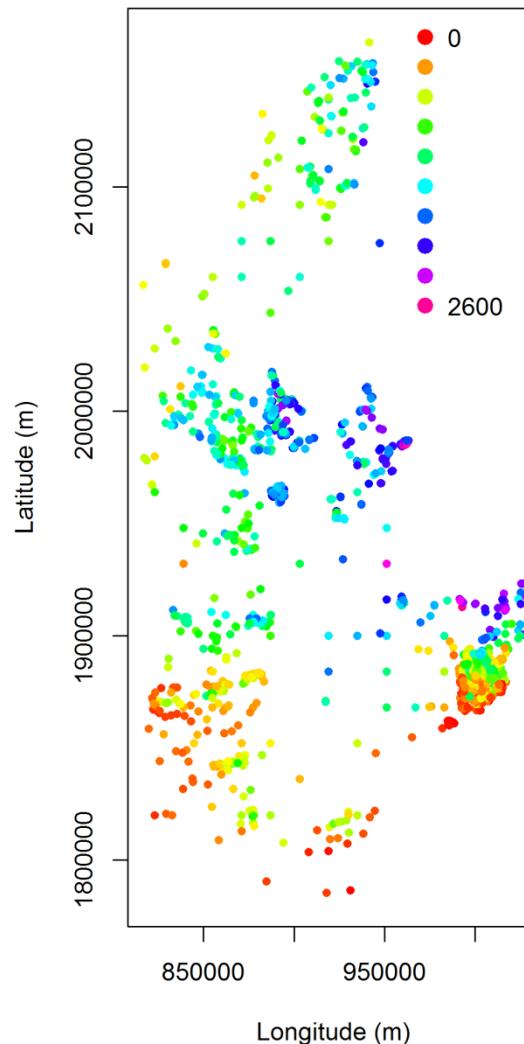
7. À l'aide d'une représentation graphique adapté, comparez la distribution des relevés en altitude entre les six massifs étudiés
8. Sur une même fenêtre graphique, représentez deux nuages de points, l'un pour les relevés localisés dans le massif "alpes" à la période 1 et l'autre pour les relevés localisés dans le massif "alpes" à la période 2
9. Pour chacun de ces deux nuages de points, utilisez le même code couleur pour représenter en couleurs "chaudes" les relevés situés à basses altitudes et en couleurs "froides" les relevés situés à hautes altitudes
10. Sauvegarder ce graphique au format .tiff

Exercice 3

Altitude (m) 1905-1985



Altitude (m) 1986-2005



Plan du cours

Chapitre 2 : Manipulation et visualisation de données dans R

2.1. Quelques fonctions utiles pour manipuler ses données

2.2. Quelques fonctions utiles pour visualiser ses données

2.3. Applications

 2.3.1. *À partir de données simulées*

 2.3.2. *À partir de données réelles*

 2.3.2.1 Issues de tableaux de données (data frame)

 2.3.2.2 Issues d'images (matrix)

Exercice 4

1. Allez sur le site de l'IGN® et téléchargez la BD ALTI® au pas de 250 m sur toute la France métropolitaine (format ASC) (zip, 13 Mo)

<http://professionnels.ign.fr/bdalti#tab-3>

The screenshot shows the professional space of the IGN website. At the top, there's a navigation bar with links to 'Tous les sites de l'IGN', 'Le portail de l'IGN', 'Fil d'actualité', and 'Résultats de l'enquête nationale'. On the right side of the header, there are links for 'MON ESPACE >> S'inscrire >>' and 'MON PANIER >> 0 article(s)'. Below the header, there's a main menu with categories: CATALOGUE, SERVICES PUBLICS, ENSEIGNEMENT ET RECHERCHE, ENTREPRISES, DOSSIERS & ACTUALITÉS, and SUPPORT & CONTACTS. A search bar labeled 'RECHERCHE' is located below the menu. The main content area has a title 'L'ESPACE PROFESSIONNEL'. On the left, there's a sidebar titled 'CATALOGUE' with links to various data types like 'Données par type >>', 'Données par échelle >>', 'Données raster >>', 'Données vecteur >>', 'BATI-3D®', 'BD ADRESSE®', 'BD ALTI®', 'BD CARTHAGE®', 'BD Forêt', 'BD TOPO®', and 'Carto3D®'. There's also a link to 'Découvrez la BD ALTI®'. In the center, there's a large image of a 3D terrain model of a mountainous area. To the right of the image, there's a section titled 'BD ALTI®' with a description: 'Référentiel du relief sur la France, la BD ALTI® est une gamme complète de modèles numériques de terrain (MNT) qui décrivent la forme du terrain à différentes échelles (du 1 : 25 000 au 1 : 1 000 000)'. Below this, there's a logo for 'IO OI' and a link to 'Accéder au téléchargement de la BD ALTI® aux pas de 250m, 500m et 1000m'. On the right side of the content area, there are two boxes: 'Caractéristiques' with links to 'Gratuit pour... >>' and 'Couverture >>', and 'Besoin d'un conseil ?' with a note aboutFAQs and tutors. At the bottom right, there's a logo for 'UNIVERSITÉ de Picardie Jules Verne'.

Exercice 4

2. Dézippez le dossier téléchargé sur le site de l'IGN®
3. Ouvrez le fichier "MNT250_L93_FRANCE.asc" à l'aide de votre éditeur de texte (cf. Bloc-notes), supprimez les 6 premières lignes, vérifiez le type de séparateur et enregistrez au format ".txt"
4. Importez le fichier " MNT250_L93_FRANCE.txt" dans R et lui donner un nom du type "mnt"
5. Vérifiez la classe de l'objet "mnt" et transformez le en "matrix"
6. Que constatez vous en affichant l'objet "mnt" avec la fonction "image()" ?
7. Remplacez les valeurs d'altitude abérantes par "NA" et affichez de nouveau l'objet "mnt"
8. Faites pivoter l'objet "mnt" de manière à bien orienter la carte