Programmation statistique avec R

Introduction et éléments de base

J. R. Lobry adapté de Deepayan Sarkar

Université Claude Bernard Lyon I – France

Biologie & Modélisation 2007-2008 (saison 2)

Table des matières

Premiers pas

Qu'est ce que R ?

- R est un environnement permettant de faire des analyses statistiques et de produire des graphiques. C'est également un langage de programmation complet.
- Nous allons utiliser ici R comme une boîte à outils pour faire des analyses statistiques standard.
- Cependant, il faut bien comprendre que R est un langage de programmation. C'est cet aspect qui fait que R est différent des autres logiciels statistiques.

Les informations sur R sont disponibles sur la homepage du projet R : `http://www.r-project.org`, c'est le premier résultat pour la recherche de la lettre "R" avec le moteur de recherche google.

Qu'est ce que R ? Une métaphore

```
library(fortunes)
fortune("busses")
```

```
When talking about user friendliness of computer software I like the
analogy of cars vs. busses: [...]
Using this analogy programs like SPSS are busses, easy to use for the
standard things, but very frustrating if you want to do something
that is not already preprogrammed.
R is a 4-wheel drive SUV (though environmentally friendly) with a
bike on the back, a kayak on top, good walking and running shoes in
the passenger seat, and mountain climbing and spelunking gear in the
back. R can take you anywhere you want to go if you take time to
learn how to use the equipment, but that is going to take longer than
learning where the bus stops are in SPSS.
   -- Greg Snow
      R-help (May 2006)
```

Notre objectif pour ce cours

. . . est simplement de nous familiariser avec  .

- Apprendre les bases du langage
- Apprendre à manipuler des données
- Apprendre à faire un graphique
- Apprendre à utiliser la documentation et le système d'aide

Lancer et quitter 

- *Unix/Linux* : entrer R dans un terminal
- *Mac OS X* : double-click sur R
- *Windaube* : double-click sur R.bin

Pour quitter  , entrer `q()` sur la ligne de commande.

Premier pas : Interaction avec R

On utilise généralement R interactivement, selon un cycle question-et-réponse :

- Vous entrez une commande et tapez la touche "Retour à la ligne".
- R exécute cette commande (avec affichage d'un résultat si besoin est)
- R attend une autre commande

Quelques exemples simples

Dans les exemples suivants, ce qui est entré par l'utilisateur figure en rouge, et la réponse de R est en bleu. Par exemple :

```
2 + 2
```

```
[1] 4
```

Ces exemples ont été exécutés avec la version de R donnée en pied de page.

Quelques exemples simples

```
 exp(-2)
```

```
[1] 0.1353353
```

```
 log(100, base = 10)
```

```
[1] 2
```

```
 runif(10)
```

```
 [1] 0.45704284 0.59913858 0.65647496 0.95074624 0.01629246 0.60504689
 [7] 0.91514271 0.62570242 0.38427088 0.12137021
```

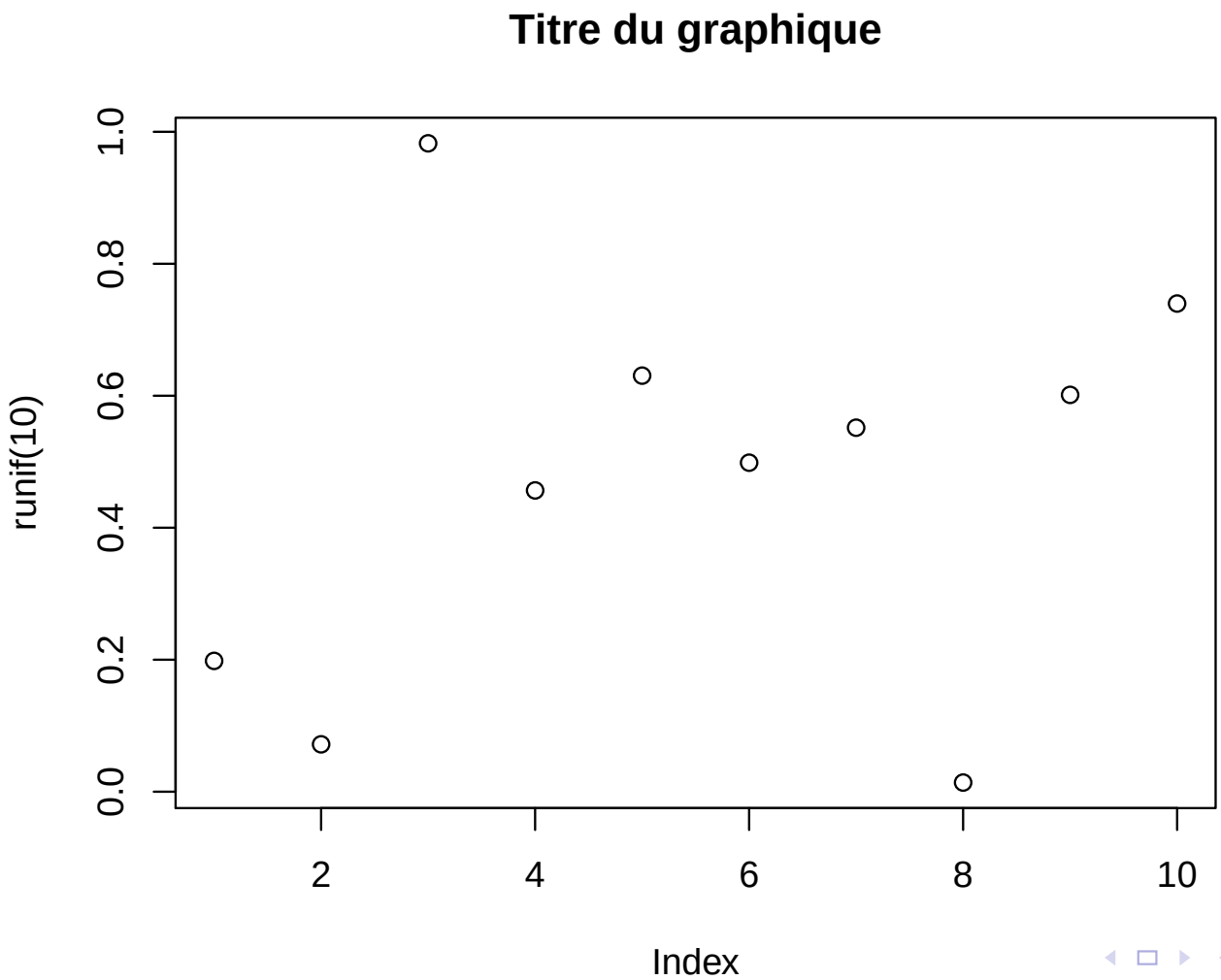
La dernière commande produit 10 nombres pseudo-aléatoires compris entre 0 et 1. Le résultat affiché est un vecteur de 10 nombres. Les nombres entre crochets au debut de chaque ligne donnent l'indice du premier nombre de la ligne.

Les fonctions

- `exp()`, `log()` et `runif()` sont des fonctions.
- Les appels aux fonctions sont indiqués par la présence de parenthèses.
- La plupart des choses utiles sous R sont faites par des fonctions.

Les fonctions graphiques

```
plot(runif(10), main = "Titre du graphique")
```



Pour en savoir plus

- Pour un public francophone, un très bon point de départ est le manuel d'Emmanuel Paradis, R pour les débutants, qui a la particularité d'exister également en version internationale (*R for Beginners*). Les deux sont disponibles (`http://www.r-project.org/`) dans la rubrique **Documentation**, sous-rubrique **Contributed**.
- Plusieurs milliers de pages d'enseignement en français de statistiques sous  sont disponibles ici : `http://pbil.univ-lyon1.fr/R/`. Les niveaux vont de l'initiation au niveau post-doctoral, à vous d'explorer.

Manipuler des données

Variables et Affectations

Comme la plupart des langages de programmation, R a des variables auxquelles on peut affecter une valeur. Pour cela on utilise l'opérateur '<-' ou '->'. L'opérateur classique '=' marche aussi.

```
x <- 2
y <- x + 3
s <- "ceci est une chaine de caracteres"
x
```

```
[1] 2
```

```
y
```

```
[1] 5
```

```
s
```

```
[1] "ceci est une chaine de caracteres"
```

```
x + x
```

```
[1] 4
```

```
x^y
```

```
[1] 32
```

Noms des variables

Les noms de variables sont très flexibles. N'importe quelle variable peut stocker n'importe quelle valeur (il n'y a pas besoin de déclarer les variables). Cependant, il faut savoir que :

- Les noms de variables ne peuvent pas commencer par un chiffre ou un caractère spécial
- Les noms sont sensibles à la casse des caractères (un caractère minuscule comme `x` est différent d'un caractère majuscule comme `X`)
- Quelques noms courants sont déjà utilisés par  (*e.g.* `c`, `q`, `t`, `C`, `D`, `F`, `I`, `T`) et doivent être évités

Noms prédéfinis

La liste des noms prédéfinis dans la bibliothèque de base peut être consultée ainsi :

```
noms <- ls("package:base")
length(noms)
```

```
[1] 1134
```

Il y a donc 1134 noms prédéfinis dans la la bibliothèque de base. Les noms prédéfinis de moins de 3 caractères sont les suivants :

```
 [1] "abs" "all" "any" "Arg" "by"  "c"   "cat" "col" "cos" "cut" "det"
[12] "dim" "dir" "exp" "F"   "for" "gc"  "get" "gl"  "I"   "if"  "Im"
[23] "log" "ls"  "Map" "max" "min" "Mod" "pi"  "q"   "qr"  "raw" "Re"
[34] "rep" "rev" "rle" "rm"  "row" "seq" "sin" "sub" "sum" "svd" "t"
[45] "T"   "tan" "try" "unz" "url" "xor"
```

Vecteurs

- Les types élémentaires dans R sont tous des vecteurs
- Même un simple nombre est un vecteur de longueur 1

La construction `c(...)` peut être utilisée pour générer un nouveau vecteur :

```
poids <- c(60, 72, 57, 90, 95, 72)
poids
```

```
[1] 60 72 57 90 95 72
```

Arithmétique vectorielle

Les opérations arithmétiques usuelles

- + pour faire des additions
- - pour faire des soustractions
- * pour faire des multiplications
- / pour faire des divisions
- ^ pour élever à la puissance

et les fonctions mathématiques travaillent élément par élément sur les vecteurs et produisent un autre vecteur :

```
taille <- c(1.75, 1.8, 1.65, 1.9, 1.74, 1.91)
taille^2
```

```
[1] 3.0625 3.2400 2.7225 3.6100 3.0276 3.6481
```

```
imc <- poids/taille^2
imc
```

```
[1] 19.59184 22.22222 20.93664 24.93075 31.37799 19.73630
```

```
log(imc)
```

```
[1] 2.975113 3.101093 3.041501 3.216102 3.446107 2.982460
```

Arithmétique vectorielle : recyclage

Quand deux vecteurs ne sont pas de même longueur, le plus court est recyclé. La commande suivante ajoute 0 à tous les éléments impairs et 2 à tous les éléments pairs de la variable `imc` :

```
 imc
[1] 19.59184 22.22222 20.93664 24.93075 31.37799 19.73630
 c(0, 2)
[1] 0 2
 imc + c(0, 2)
[1] 19.59184 24.22222 20.93664 26.93075 31.37799 21.73630
```

Fonctions vectorisées

Beaucoup de fonctions résument un vecteur de données en produisant un nombre à partir d'un vecteur. Par exemple :

```
sum(poids)
```

```
[1] 446
```

```
length(poids)
```

```
[1] 6
```

```
poids.moy <- sum(poids)/length(poids)
poids.moy
```

```
[1] 74.33333
```

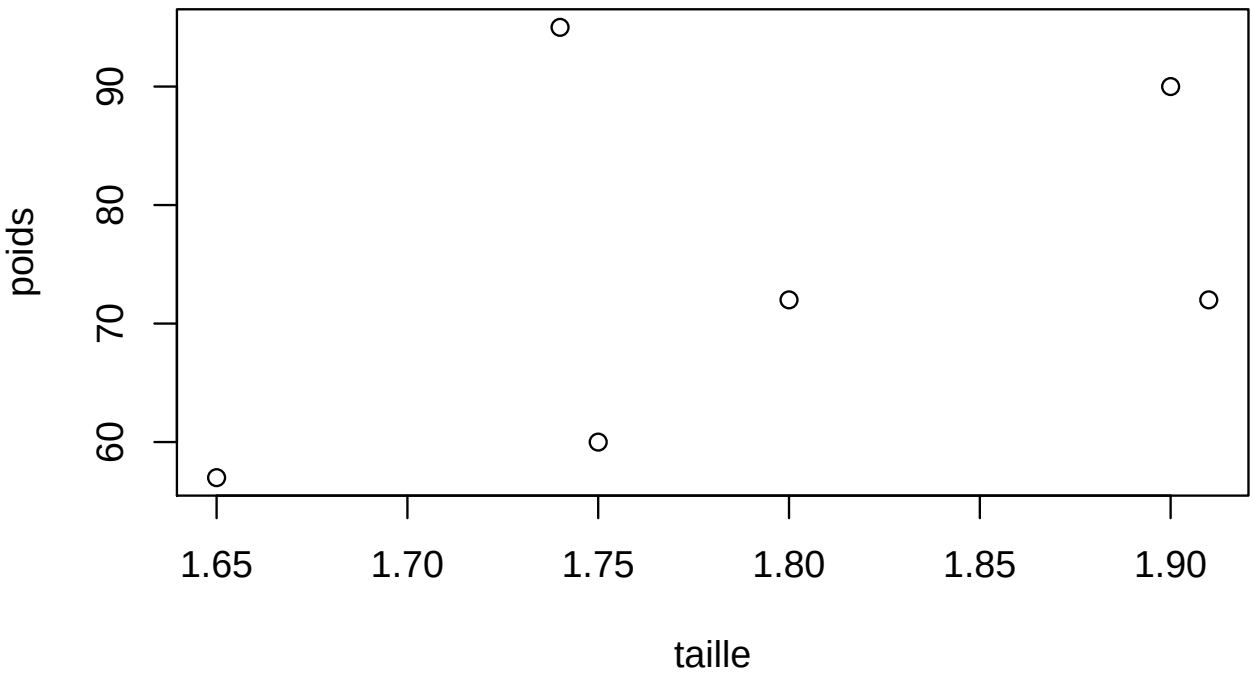
La dernière commande calcule la moyenne de `poids` qui vaut donc ici 74.3.

Graphiques

plot()

La manière la plus simple de produire des graphiques sous R est d'utiliser la fonction `plot()` :

```
plot(x = taille, y = poids)
```



Options et retouche

Les fonctions graphiques de R comportent de nombreuses options qui permettent de contrôler de façon très fine les graphiques. Par exemple, les paramètres de la fonction `plot` utilisée par défaut sont :

```
args(plot.default)
```

```
function (x, y = NULL, type = "p", xlim = NULL, ylim = NULL,
    log = "", main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
    ann = par("ann"), axes = TRUE, frame.plot = axes, panel.first = NULL,
    panel.last = NULL, asp = NA, ...)
NULL
```

L'argument `...` signifie qu'il y a encore d'autre paramètres graphiques possibles. Ils sont contrôlés par la fonction `par()`.

Options et retouche

```
names(par())
```

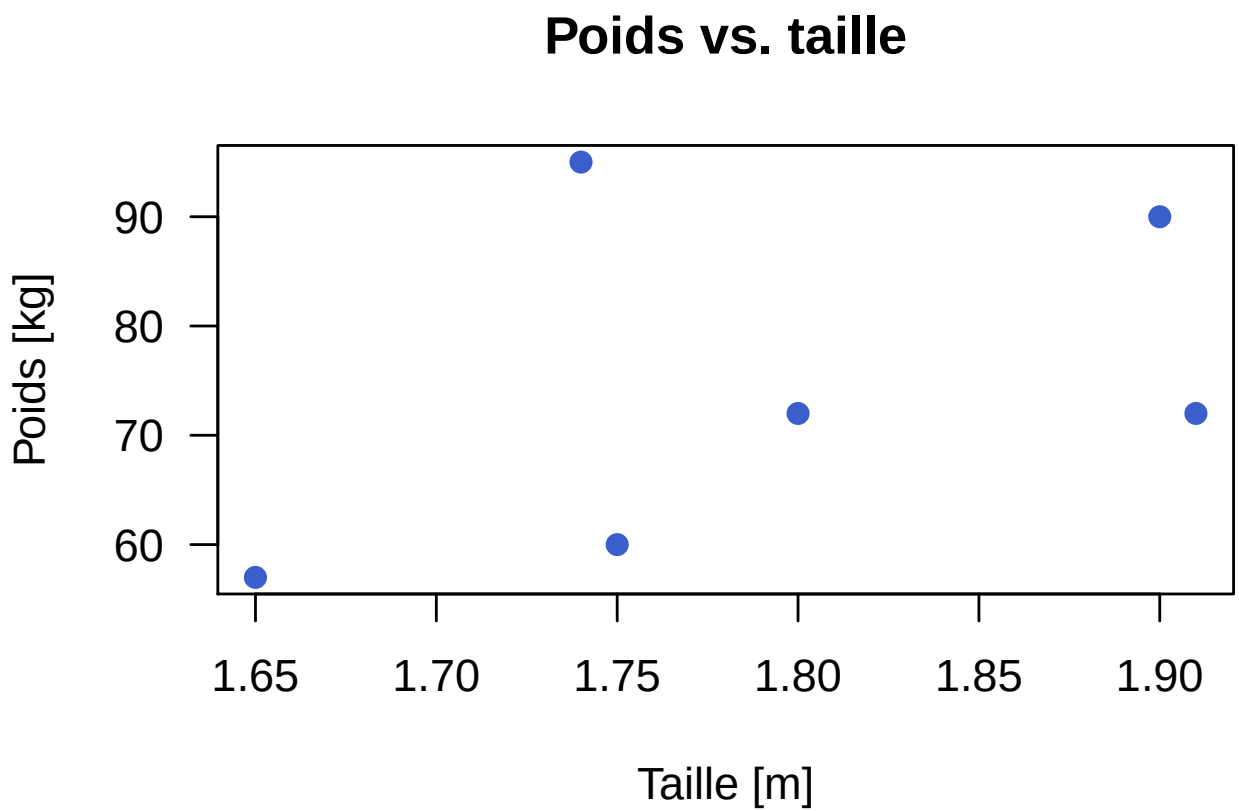
```
 [1] "xlog"      "ylog"      "adj"       "ann"       "ask"
 [6] "bg"        "bty"       "cex"       "cex.axis"  "cex.lab"
[11] "cex.main"  "cex.sub"   "cin"       "col"       "col.axis"
[16] "col.lab"   "col.main"  "col.sub"   "cra"       "crt"
[21] "csi"       "cxy"       "din"       "err"       "family"
[26] "fg"        "fig"       "fin"       "font"      "font.axis"
[31] "font.lab"  "font.main" "font.sub"  "lab"       "las"
[36] "lend"      "lheight"   "ljoin"     "lmitre"    "lty"
[41] "lwd"       "mai"       "mar"       "mex"       "mfcol"
[46] "mfg"       "mfrow"     "mgp"       "mkh"       "new"
[51] "oma"       "omd"       "omi"       "pch"       "pin"
[56] "plt"       "ps"        "pty"       "smo"       "srt"
[61] "tck"       "tcl"       "usr"       "xaxp"      "xaxs"
[66] "xaxt"      "xpd"       "yaxp"      "yaxs"      "yaxt"
```

Pour une exploration systématique des paramètres graphiques, voir la fiche http://pbil.univ-lyon1.fr/R/fichestd/tdr75.pdf.

Options et retouche

Un exemple de graphique utilisant quelques options :

```
plot(x = taille, y = poids, pch = 19, col = "royalblue3",
    las = 1, main = "Poids vs. taille", xlab = "Taille [m]",
    ylab = "Poids [kg]")
```

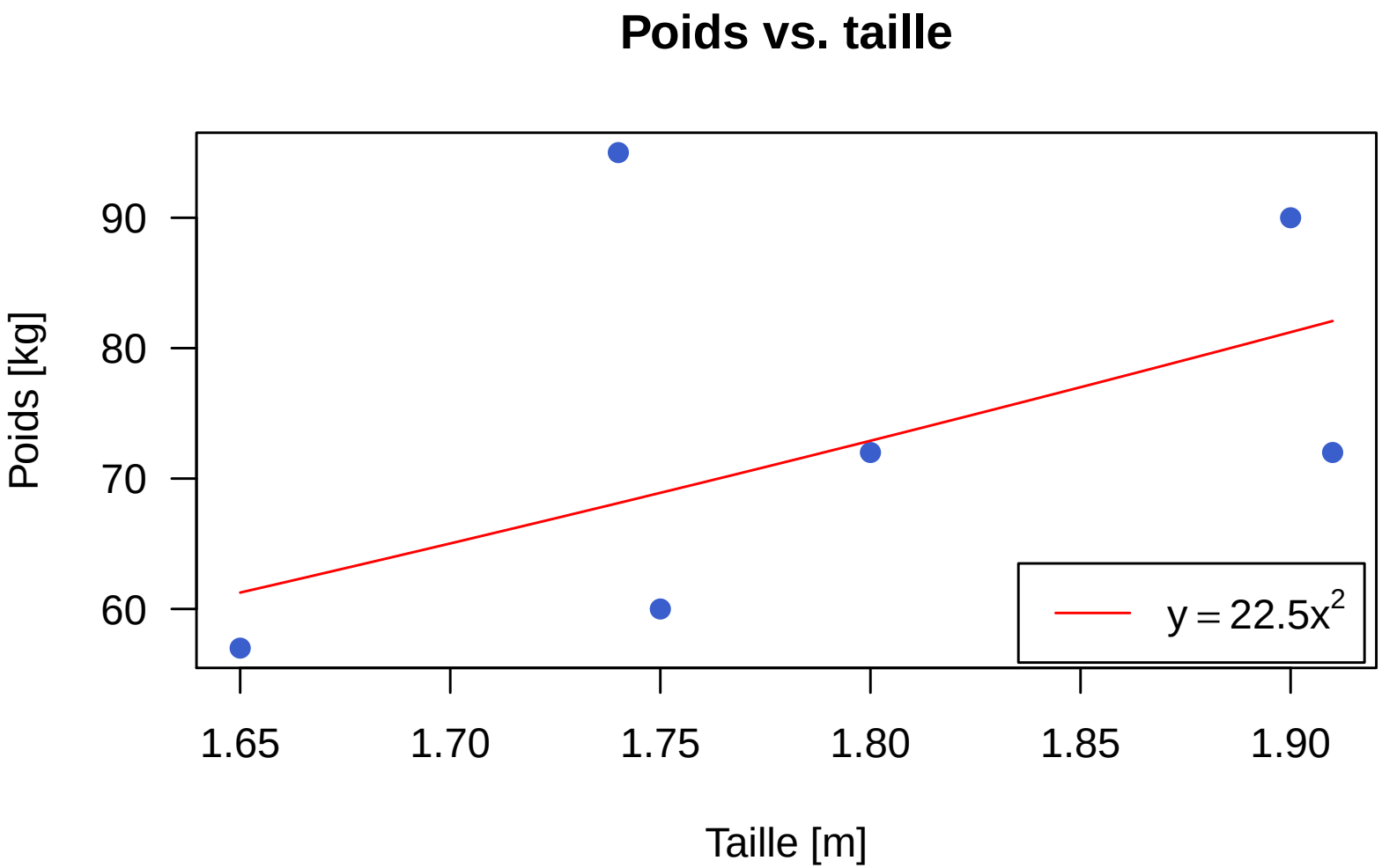


Options et retouche

Il existe de nombreuses fonctions permettant de retoucher un graphique, par exemple :

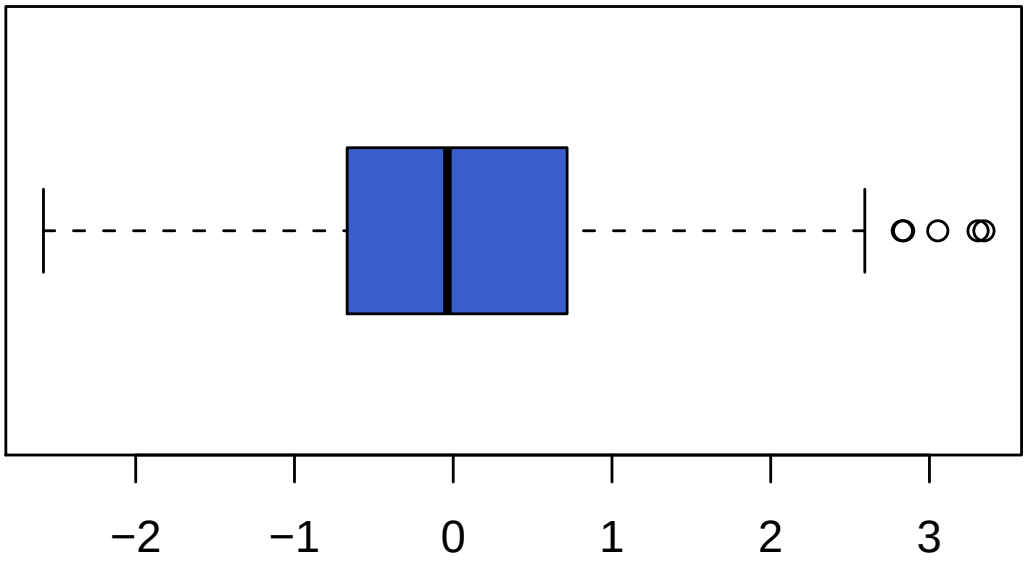
```
plot(x = taille, y = poids, pch = 19, col = "royalblue3",
    las = 1, main = "Poids vs. taille", xlab = "Taille [m]",
    ylab = "Poids [kg]")
x <- seq(from = min(taille), to = max(taille), length = 100)
lines(x = x, y = 22.5 * x^2, col = "red")
legend("bottomright", inset = 0.01, legend = expression(y ==
    22.5 * x^2), lty = 1, col = "red")
```

Options et retouche



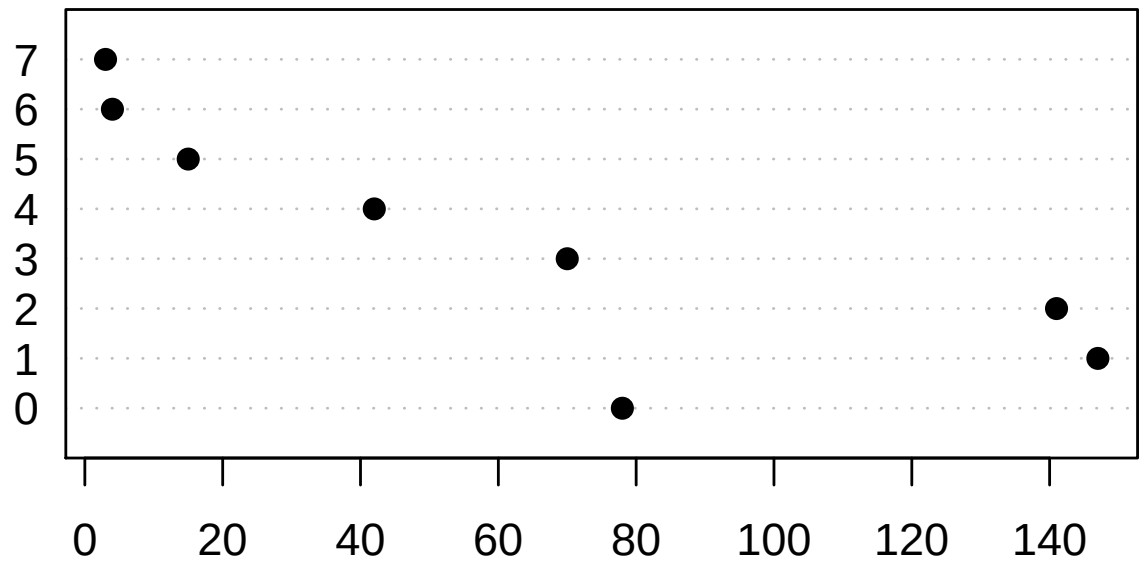
boxplot()

```
boxplot(rnorm(500), col = "royalblue3", horizontal = TRUE)
```



dotchart()

```
dotchart(table(rpois(500, lambda = 2)), pch = 19)
```



Obtenir de l'aide

help.start()

 a beaucoup trop d'outils pour que quiconque puisse les retenir tous, il est donc très important de savoir comment retrouver les informations pertinentes en utilisant le système d'aide.
`help.start()` ouvre une fenêtre avec une interface pour l'aide de type HTML. Un très bon point de départ. Il y a un lien vers un manuel très détaillé pour les débutants appelé "An Introduction to  ", ainsi que des listes par sujets.

RSiteSearch()

La fonction `RSiteSearch()` fait une recherche dans l'ensemble des documents (manuels, documentation, archives des listes de diffusion) du site de R.

help.search()

Quand vous voulez obtenir de l'aide sur un sujet donné, mais que vous ne savez pas quelle est la bonne page d'aide, la fonction `help.search()` est très utile. Essayez par exemple :

```
> help.search("logarithm")
```

help(sujet) ou ?sujet

`help(sujet)` que l'on peut aussi écrire `?sujet` affiche la page d'aide pour le sujet ou la fonction `sujet`. Toutes les fonctions de  ont une page d'aide. Quand on connaît le nom de la fonction où du sujet qui nous intéresse, c'est en général le meilleur moyen d'apprendre à l'utiliser.

example()

Les pages d'aide sont généralement très détaillées. Elles contiennent souvent, entre autres :

- Une section "See Also" qui donne les pages d'aide sur des sujets apparentés
- Une section "Description" de ce que fait la fonction
- Une section "Examples" avec du code illustrant ce que fait la fonction documentée. Ces exemples peuvent être exécutés directement en utilisant la fonction `example()`, essayez par exemple :

```
example(plot)
```

apropos()

Un autre outil utile est la fonction `apropos()` qui donne une liste de tous les sujets contenant (exactement) l'argument :

```
apropos("plot")[1:10]
```

```
 [1] ".__C__recordedplot" "assocplot"          "barplot"
 [4] "barplot.default"    "biplot"             "boxplot"
 [7] "boxplot.default"    "boxplot.matrix"     "boxplot.stats"
[10] "cdplot"
```

Nous n'avons donné que les 10 premiers éléments, la liste complète est trop longue.

args()

La fonction `args()` donne la liste des arguments d'une fonction :

```
args(plot.default)
```

```
function (x, y = NULL, type = "p", xlim = NULL, ylim = NULL,
    log = "", main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
    ann = par("ann"), axes = TRUE, frame.plot = axes, panel.first = NULL,
    panel.last = NULL, asp = NA, ...)
NULL
```

Les paquetages (packages) R

library()

R utilise un système de bibliothèques, les *packages*.

- Chaque bibliothèque est une collection regroupant des outils d'une même thématique.
- R est lui même une bibliothèque appelée `base`
- Certaines bibliothèques sont automatiquement disponibles lorsque R est lancé, d'autres doivent être chargées avec la fonction `library()`

installed.packages()

Certaines bibliothèques sont pré-installées avec R. La liste des bibliothèques installées est donnée par la fonction :

```
installed.packages()
```

Il y a beaucoup d'autre bibliothèques développées par des utilisateurs de R disponibles sur le site du CRAN (Comprehensive R Archive Network).

search()

Certaines bibliothèques sont automatiquement disponibles lorsque  est lancé. À n'importe quel moment, la liste des bibliothèques chargées est donnée par la fonction `search()` :

```
search()
```

```
 [1] ".GlobalEnv"        "package:ISwR"      "package:fortunes"
 [4] "package:stats"     "package:graphics"  "package:grDevices"
 [7] "package:utils"     "package:datasets"  "package:methods"
[10] "Autoloads"         "package:base"
```

install.packages()

D'autres bibliothèques peuvent être chargées par l'utilisateur. Nous allons charger la bibliothèque `ISwR` qui contient des données illustrant le texte. Ceci peut être fait avec :

```
library(ISwR)
```

De nouvelles bibliothèques peuvent être téléchargées et installées avec la fonction `install.packages()`. Par exemple, pour installer la bibliothèque `ISwR` (si elle n'est pas déjà installée), on peut utiliser :

```
install.packages("ISwR", lib = getwd())
library(help = ISwR, lib = getwd())
```

La dernière commande donne la liste de toutes les pages d'aide de la bibliothèque.