Introduction to R

PIG session 3

Thibault LAURENT
thibault.laurent@univ-tlse1.fr

Toulouse School of Economics

17th May 2011

General informations

Basic manipulation

Objects and Classes

Probability distributions

Computing

Graphics

Packages

What is R?

- Software dedicated to statistical and scientific computing using its own language and can be coupled with C, Fortran, Python code, etc.
- Free, included in GNU project.
- Multiplatform (Linux, Mac OS, Windows).
- Wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, etc.) and graphical techniques.
- Basic packages can be extendable by other packages.

R for economist?

- R is generally associated as a tool for statistician.
- However, it can be easily used by economist for testing/validating economic techniques on (real or simulated) data.
- Cloudly Chen (2009), "From Economics to R: Using R in Economics and Econometrics", suggests to combine both statistic and economic point of views with R, `http://www.cloudlychen.net/pdfs/economics_and_R.pdf`

R for SAS, Stata, MATLAB, etc. users?

Two good reasons for choosing R:

- R is free.
- It should do (quasi) the same things that the other scientific softwares (and more).

Some constraints:

- To be as faster as the others, it requires good knowledge of R
- Same things for SGBD and big data

Weblink (1)

`http://www.r-project.org/`

- textbooks written or approved by the R team (link *Manuals*).
- a Wiki, a FAQ and a Journal (The R journal).
- a mail-list (notes on the news, etc), a bibliography of books on R, information about the foundation, past and future conferences UseR!, links to R projects, examples of charts, etc ...

Weblink (2)

The Comprehensive R Archive Network (`http://cran.r-project.org/`) and its mirror sites (e.g. `http://cran.cict.fr/`) for downloading:

- software, version R-2.13.0 appeared in April 2011 and updated about 4 times a year.
- packages (over 3 000 at this date...) that are listed in alphabetical or thematic order like Econometrics, Optimization, Time Series, etc. in the *Task Views* link.
- other books written in several languages (*Contributed* link)

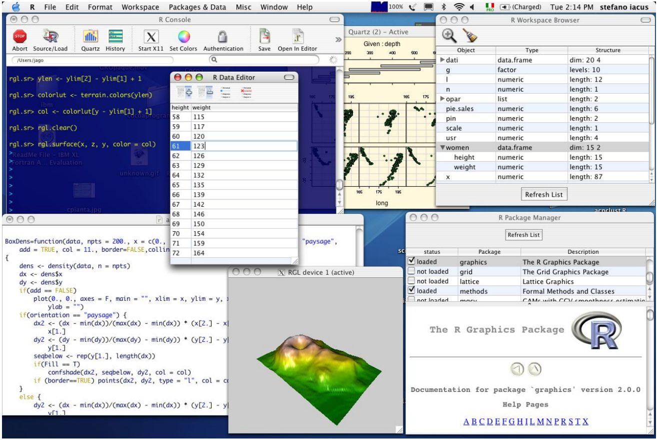
Bibliography

- W.N. Venables et al., *An introduction to R*, `http://www.r-project.org/`.
- K. Kleinman and N.J. Horton, *SAS and R, Data Management, Statistical Analysis and Graphics*, Chapman and Hall.
- All books in collection *Use R!* and *Pratique R*, Springer-Verlag.

Installing R

- An easy step-by-step guide to Windows installation may be found here: `https://wiki.duke.edu/display/DUKER/Install+R+Under+Windows`.
- on a MAC: go to `http://cran.r-project.org/` and select MacOS X.
- on Linux, depending on your distribution, see `http://www.stat.umn.edu/HELP/r.html#down-tux`.

The R console on a mac



```
rgl.sr> ylen <- ylim[2] - ylim[1] + 1

rgl.sr> colorlut <- terrain.colors(ylen)

rgl.sr> col <- colorlut[y - ylim[1] + 1]

rgl.sr> rgl.clear()

rgl.sr> rgl.surface(x, z, y, color = col)
>
>
>
>
>
>
>
>
```

height	weight
58	115
59	117
60	120
61	123
62	126
63	129
64	132
65	135
66	139
67	142
68	146
69	150
70	154
71	159
72	164

Object	Type	Structure
▶ dati	data.frame	dim: 20 4
g	factor	levels: 10
l	numeric	length: 12
n	numeric	length: 1
▶ opar	list	length: 2
pie.sales	numeric	length: 6
pin	numeric	length: 2
scale	numeric	length: 1
usr	numeric	length: 4
▼ women	data.frame	dim: 15 2
height	numeric	length: 15
weight	numeric	length: 15
x	numeric	length: 87

```
BoxDens=function(data, npts = 200., x = c(0.,                    "paysage",
    add = TRUE, col = 11., border=FALSE,collin
{
    dens <- density(data, n = npts)
    dx <- dens$x
    dy <- dens$y
    if(add == FALSE)
        plot(0., 0., axes = F, main = "", xlim = x, ylim = y, 
            ylab = "")
    if(orientation == "paysage") {
        dx2 <- (dx - min(dx))/(max(dx) - min(dx)) * (x[2.] - x
            x[1.]
        dy2 <- (dy - min(dy))/(max(dy) - min(dy)) * (y[2.] - y
            y[1.]
        seqbelow <- rep(y[1.], length(dx))
        if(Fill == T)
            confshade(dx2, seqbelow, dy2, col = col)
        if (border==TRUE) points(dx2, dy2, type = "l", col = c
    }
    else {
        dy2 <- (dx - min(dx))/(max(dx) - min(dx)) * (y[2.] - y
            y[1.]
```

status	Package	Description
☑ loaded	graphics	The R Graphics Package
□ not loaded	grid	The Grid Graphics Package
□ not loaded	lattice	Lattice Graphics
☑ loaded	methods	Formal Methods and Classes
□ not loaded	mgcv	GAMs with GCV smoothness estimatio

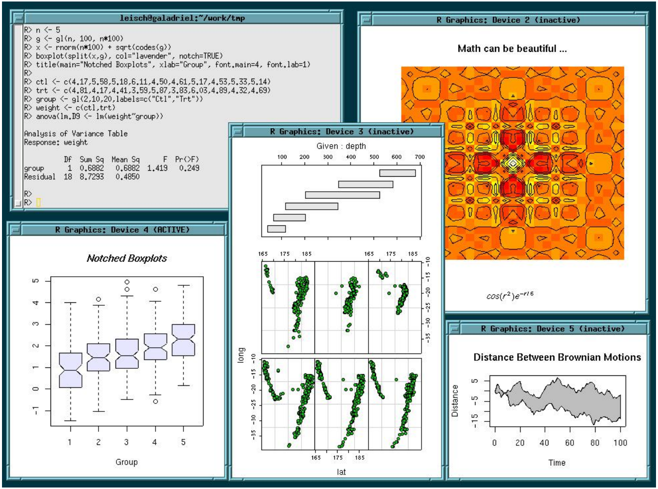
The R Graphics Package

Documentation for package `graphics' version 2.0.0

Help Pages

A B C D E F G H I L M N P R S T X

The R console on Linux

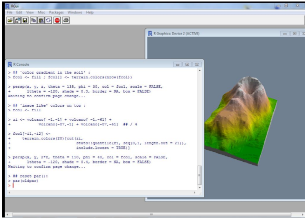


leisch@galadriel:~/work/tmp

```
R> n <- 5
R> g <- gl(n, 100, n*100)
R> x <- rnorm(n*100) + sqrt(codes(g))
R> boxplot(split(x,g), col="lavender", notch=TRUE)
R> title(main="Notched Boxplots", xlab="Group", font.main=4, font.lab=1)
R>
R> ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
R> trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
R> group <- gl(2,10,20,labels=c("Ctl","Trt"))
R> weight <- c(ctl,trt)
R> anova(lm.D9 <- lm(weight~group))

Analysis of Variance Table
Response: weight

          Df  Sum Sq  Mean Sq      F  Pr(>F)
group      1  0.6882   0.6882  1.419   0.249
Residual  18  8.7293   0.4850

R>
R> 
```

The R console on Windows

The Windows R GUI File menu has a number of useful commands, which have command line equivalents. *Save history* allows saving the list of commands entered as a journal. You can *Change directory* to where your project or class files sit, and *Display file* to see the contents of a text file. The *Packages menu* is used for installing and updating contributed packages.


RGui
File Edit View Misc Packages Windows Help
R Graphics: Device 2 (ACTIVE)
R Console

```
> ## `color gradient in the soil' :
> fcol <- fill ; fcol[] <- terrain.colors(nrow(fcol))

> persp(x, y, z, theta = 135, phi = 30, col = fcol, scale = FALSE,
+       ltheta = -120, shade = 0.3, border = NA, box = FALSE)
Waiting to confirm page change...

> ## `image like' colors on top :
> fcol <- fill

> zi <- volcano[ -1,-1] + volcano[ -1,-61] +
+            volcano[-87,-1] + volcano[-87,-61]  ## / 4

> fcol[-1l,-12] <-
+     terrain.colors(20)[cut(zi,
+                            stats::quantile(zi, seq(0,1, length.out = 21)),
+                            include.lowest = TRUE)]

> persp(x, y, 2*z, theta = 110, phi = 40, col = fcol, scale = FALSE,
+       ltheta = -120, shade = 0.4, border = NA, box = FALSE)
Waiting to confirm page change...

> ## reset par():
> par(oldpar)
>
```

```
> n <- 4
> (15 + 12.5 + 14 + 9.5)/n
> savehistory("program.Rhistory")
> getwd()
> file.show(file.choose())
```

R Editors

- The R editor for Windows or Mac-OS (‘File’ $<$ ‘New script’).
- *Emacs* or *gedit* for Linux.
- *Tinn-R* for Windows: many fonctionalities. See `http://www.sciviews.org/Tinn-R/`.

Getting help

- Use of the function `help()` (abbreviation `?`) when the name of the function is known

```
> help(solve)
```

- Sections *See also* and *Examples* very useful

```
> example(solve)
```

- function `help.search` when searching a key word in functions included in any package (base or dowloaded)

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

R commands

`A` and `a` are different symbols and would refer to different variables:

```
> A = 10
> a <- 6
```

```
# This is a comment
> a.1=A/a; a_1=a/A
```

If a command is not complete at the end of a line, R will give a different prompt ($+$ instead of $>$). The collection of objects currently stored is called the workspace:

```
> objects()
```

To remove objects:

```
> rm(a, A)
```

General informations

Basic manipulation

Objects and Classes

Probability distributions

Computing

Graphics

Packages

Vectors and assignement

To set up a vector named `x`, use the function `c()`:

```
> x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
```

This comand will inverse each element of `x` without changing the values of `x`:

```
> 1/x

[1] 0.09615385 0.17857143 0.32258065 0.15625000 0.04608295
```

New assignment for a vector of size 11:

```
> y <- c(x, cos(pi/4), exp(5), sqrt(5), log(10),
+     sin(pi), 2^3)
```

Vector arithmetic

```
> x
[1] 10.4  5.6  3.1  6.4 21.7
> y
 [1] 1.040000e+01 5.600000e+00 3.100000e+00 6.400000e+00
 [5] 2.170000e+01 7.071068e-01 1.484132e+02 2.236068e+00
 [9] 2.302585e+00 1.224647e-16 8.000000e+00
```

This assignement:

```
> v <- 2 * x + y + 1
```

is equivalent:

```
> 2 * c(x, x, 10.4) + y + rep(1, 11)
```

where `rep` is a function which repeats 11 times the value 1.

Vector arithmetic (2)

- To compute $x^T x$, we recommand:

  ```
  > crossprod(x)
  ```

  ```
         [,1]
  [1,] 660.98
  ```

  instead of the matrix product:

  ```
  > t(x) %*% x
  ```

  ```
         [,1]
  [1,] 660.98
  ```

- To compute xx^T, we recommand:

  ```
  > x %o% x
  ```

  instead of:

  ```
  > x %*% t(x)
  ```

Common functions for vector

```
> range(y)
```

```
[1] 1.224647e-16 1.484132e+02
```

equivalent to :

```
> c(min(y), max(y))
```

```
[1] 1.224647e-16 1.484132e+02
```

```
> var(y)
```

```
[1] 1879.661
```

equivalent to:

```
> sum((y - mean(y))^2)/(length(y) - 1)
```

```
[1] 1879.661
```

```
> sort(y)
```

```
 [1] 1.224647e-16 7.071068e-01 2.236068e+00 2.302585e+00
 [5] 3.100000e+00 5.600000e+00 6.400000e+00 8.000000e+00
 [9] 1.040000e+01 2.170000e+01 1.484132e+02
```

Generating regular sequences

To create the vector `c(1,2,...,15)`:

```
> 1:15
```

The function `seq` is a more general facility for generating sequences:

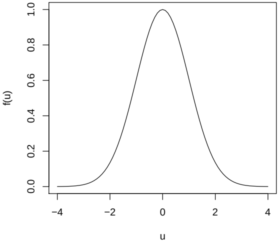
```
> u <- seq(-4, 4, by = 0.1)
```

For replicating an object:

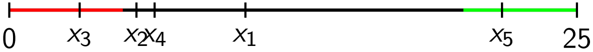
```
> rep(x, times = 2)

 [1] 10.4  5.6  3.1  6.4 21.7 10.4  5.6  3.1  6.4 21.7

> rep(x, each = 2)

 [1] 10.4 10.4  5.6  5.6  3.1  3.1  6.4  6.4 21.7 21.7
```

Generating regular sequences (2)

```
> f <- function(x) {
+     exp(-x^2/2)
+ }
> plot(u, f(u), type = "l")
```



Logical vector



```
> temp.1 <- x <= 5
> temp.2 <- x > 20
> temp.1

[1] FALSE FALSE  TRUE FALSE FALSE

> temp.2

[1] FALSE FALSE FALSE FALSE  TRUE

> temp.1 & temp.2

[1] FALSE FALSE FALSE FALSE FALSE

> temp.1 | temp.2

[1] FALSE FALSE  TRUE FALSE  TRUE
```

Logical vector (2)

- Equality/Difference

```
> temp.3 <- x == max(x)
> temp.4 <- x != min(x)
```

- Negation:

```
> temp.3

[1] FALSE FALSE FALSE FALSE  TRUE

> !temp.3

[1]  TRUE  TRUE  TRUE  TRUE FALSE
```

- `which` returns the indexes whose values equal `TRUE`:

```
> which(temp.1)
```

- Logical vectors may be used in ordinary arithmetic:

```
> sum(temp.1)
```

Missing values

```
> z <- c(1:3, NA, 5, 6)
> ind <- is.na(z)
```

Be care:

```
> sum(z)

[1] NA

> sum(z, na.rm = TRUE)

[1] 17

> sum(na.omit(z))

[1] 17
```

Character vectors

The double quote characters:

```
> c("P1", "P2", "P3")
[1] "P1" "P2" "P3"
```

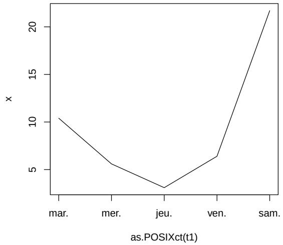
The use of function `paste` :

```
> paste("PiG session", 1:4, c("Basics", "Tokens",
+     "R", "Servor"))
[1] "PiG session 1 Basics" "PiG session 2 Tokens"
[3] "PiG session 3 R"      "PiG session 4 Servor"
> t1 <- paste("2011-05-", 17:21, sep = "")
> t2 <- paste("2011-", 5:9, "-01", sep = "")
> t1
[1] "2011-05-17" "2011-05-18" "2011-05-19" "2011-05-20"
[5] "2011-05-21"
> t2
[1] "2011-5-01" "2011-6-01" "2011-7-01" "2011-8-01"
[5] "2011-9-01"
```

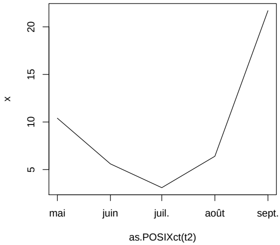
A POSIXct vector

as.POSIXct transforms a vector of character into a date

```
> plot(as.POSIXct(t1), x,
+      type = "l")
```



```
> plot(as.POSIXct(t2), x,
+      type = "l")
```



Index vectors

4 different issues to select obs.:

1. a logical

```
> x[x > mean(x)]
[1] 10.4 21.7
```

2. a vector of indexes

```
> x[c(1, 3, 5)]
[1] 10.4  3.1 21.7
```

3. a vector of indexes to exclude from the selection:

```
> x[-c(2, 4)]
[1] 10.4  3.1 21.7
```

4. a vector of character:

```
> names(x) <- c("TSE", "PSE", "LSE", "MIT", "UCL")
> x[c("UCL", "PSE")]
 UCL  PSE
21.7  5.6
```

General informations

Basic manipulation

Objects and Classes

Probability distributions

Computing

Graphics

Packages

Mode of a vector

- The basic modes are: *numeric*, *complex*, *logical* and *character*.
- Vectors must have their values all of the same mode.
- Conversion between the modes with functions `as.something()`:

```
> z <- 0:9
> digits <- as.character(z)
> d <- as.integer(digits)
```

Changing the length of an object

- Start with an empty object:

```
> e <- numeric()
```

- Add the third value of e:

```
> e[5] <- 1
> e

[1] NA NA NA NA  1

> e[1:4] <- 0
```

- Truncate the length:

```
> e <- e[3:5]
> e

[1] 0 0 1
```

Example of a bootstrap algorithm

```
> res <- numeric()
> B <- 10
> for (i in 1:B) {
+     res <- c(res, mean(sample(x, replace = TRUE)))
+ }
> res

 [1]  9.44  9.28  8.32 11.70  8.78 10.24  6.02 12.04  8.94
[10]  8.64
```

What is the class of an object?

Consider a series z_t, $t = 1, ..., T$ and the model:

$$y_t = c + \phi_1 y_{t-1} + z_t \text{ with } z_t \sim WN(0, \sigma_z^2)$$

An object of class `Arima`:

1. contains the estimation and s.d. of the parameters, the value of the log likelihood, the AIC, the residuals, etc.
2. may be used as argument of functions dedicated to the analysis of residuals, etc.

Example with class `Arima`

Function `str` returns a summary of the elements included in an object:

```
> example(arima)
> str(fit1)
```

The symbol $ (or @ depending on the structure of the object) selects one particular element of the object:

```
> fit1$coef
> fit1$loglik
> fit1$aic == -2 * fit1$loglik + 2 * (length(fit1$coef) +
+     1)
```

Example of use of function dedicated to `Arima`:

```
> tsdiag(fit1)
```

Usual object: `factor` (1)

A factor is a vector object used to specify a discrete classification (grouping) of the components of other vectors of the same length.

```
> y <- c("healthy", "healthy", "failing", "failing",
+     "healthy", "failing", "healthy", "failing",
+     "failing", "failing")
> yf <- factor(y)
> levels(yf)
```

```
[1] "failing" "healthy"
```

Suppose `x` is the Totat Debt divided by the Total asset:

```
> x1 <- c(0.85, 0.71, 1.5, 1.15, 1, 0.99, 0.5, 1.45,
+     1.9, 2)
```

The function `tapply`:

```
> tapply(x1, yf, mean)
```

```
 failing  healthy
1.498333 0.765000
```

Usual object: `factor` (2)

Suppose `x2` is the status of firms (`company` or `proprietorship`):

```
> x2 <- c("company", "propri", "propri", "propri",
+     "company", "company", "company", "company",
+     "propri", "company")
> x2f <- factor(x2)
```

The contingency table:

```
> table(y, x2f)
```

```
         x2f
y          company propri
  failing       3      3
  healthy       3      1
```

Export the results in a $\LaTeX$ format with library(xtable):

```
> require(xtable)
> tab1 <- addmargins(table(y, x2f))
> myLaTex.tab <- xtable(tab1, digits = 3, align = "l|cc|r",
+     caption = "Contingency Table")
```

Usual object: `factor` (3)

```
> print(myLaTex.tab, hline.after = c(0, 2), file = "V.tex",
+     size = "tiny")
```

A file `V.tex` is thus created and can be included in the document `.tex` with the command $\backslash input\{V.tex\}$:

	company	propri	Sum
failing	3.000	3.000	6.000
healthy	3.000	1.000	4.000
Sum	6.000	4.000	10.000

Table: Contingency Table

Usual object: `array` object

- (5 observations × 2 numeric variables) observed at 3 dates:

  ```
  > z <- 1:30
  > dim(z) <- c(5, 2, 3)
  ```

  equivalent to:

  ```
  > z <- 1:30
  > z <- array(z, dim = c(5, 2, 3))
  ```

- Array indexes. x_1 and x_2 at date t_1:

  ```
  > z[, , 1]
  ```

  x_2 at date t_2:

  ```
  > z[, 2, 2]
  ```

  The third first values of x_1 at each date:

  ```
  > z[1:3, 1, ]
  ```

Usual object: `matrix` object

- A `matrix` object

  ```
  > A <- matrix(1:6, 3, 2)
  ```

- is also a special `array`,

  ```
  > A <- array(1:6, dim = c(3, 2))
  ```

- also a concatenation row by row of two matrices with same number of columns,

  ```
  > A <- rbind(matrix(c(1, 2, 4, 5), 2, 2), c(3, 6))
  ```

- also a concatenation column by column of two matrices with same number of rows:

  ```
  > A <- cbind(1:3, 4:6)
  ```

Matrix facilities

- About the dimension:

  ```
  > dim(A)
  ```

  is equivalent to:

  ```
  > c(nrow(A), ncol(A))
  ```

- Matrix transpose is given by:

  ```
  > t(A)
  ```

- matrix of element by element product (two matrices of the same dimensions):

  ```
  > A * A
  ```

- matrix product:

  ```
  > A %*% A
  ```

- For square matrices: `diag` returns the diagonal of a matrix or constructs a diagonal matrix with a vector, `tr` returns the trace and `det` the determinant.

The apply function

The following command applies to each row of *A* the function sum:

```
> apply(A, 1, sum)
```

equivalent to :

```
> rowSums(A)
```

and better than:

```
> n.A <- nrow(A)
> res <- numeric(n.A)
> for (i in 1:n.A) {
+     res[i] <- sum(A[i, ])
+ }
> res
```

The command apply(A,2,mean) equivalent to colMeans(A) will compute the average mean of each column of A.

Linear equation and inversion

- Define y such as: $y = X\beta + \epsilon$ with $\epsilon \sim N(0,1)$, $X = [1\ x]$ and $\beta = (1\ 3)^T$

```
> X <- cbind(1, x)
> Beta <- c(1, 3)
> epsilon <- rnorm(nrow(X), 0, 1)
> y <- X %*% Beta + epsilon
```

- To resolve $\hat{\beta} = (X^T X)^{-1} X^T y$:

```
> solve((t(X) %*% X), t(X) %*% y)

       [,1]
  0.8408668
x 2.9953648
```

equivalent but more stable than:

```
> solve((t(X) %*% X)) %*% (t(X) %*% y)
```

Eigen values and eigen vectors

Consider the symmetric matrix $(x^T x)$:

- function `eigen` calculates the eigenvalues and eigenvectors of a symmetric matrix

```
> xtx <- x %o% x
> res.eigen <- eigen(xtx)
> str(res.eigen)
```

```
List of 2
 $ values : num [1:5] 6.61e+02 1.14e-13 6.28e-15 6.08e-15 -5.50e-14
 $ vectors: num [1:5, 1:5] -0.405 -0.218 -0.121 -0.249 -0.844 ...
```

- The eigen values:

```
> res.eigen$values
```

- The eigen vectors:

```
> res.eigen$vectors
```

SVD and QR decomposition

- Singular Value Decomposition: $X = UDV^T$, with U and V orthonormal matrices and D, diagonal matrix:

```
> res.svd <- svd(X)
> str(res.svd)
> apply(res.svd$u, 2, function(x) sum(x^2))
> apply(res.svd$v, 1, function(x) sum(x^2))
> res.svd$u %*% diag(res.svd$d) %*% res.svd$v
```

- QR decomposition

```
> res.qr <- qr(X)
> Q <- qr.Q(res.qr)
> R <- qr.R(res.qr)
> Q %*% R
```

Usual object: `list` object

Ordered collection of objects known as its *components*. A list could contain a numerical, a character, a matrix, a function, etc.

```
> tse <- list(labos = c("gremaq", "lerna", "arqade",
+     "gie"), nb = 200, nb.admin = c(13, 4, 0, 20))
```

- Number of components:

  ```
  > length(tse)
  ```

- Indexes: these two commands are equivalent

  ```
  > tse[[1]]
  > tse[["nb"]]
  ```

  But different to:

  ```
  > tse[1]
  ```

  which is a `list` object again

Usual object: `list` object (2)

A list object is `useful` to return the results of a function:

```
> stat.des <- function(x) {
+     list(length = length(x), mean = mean(x), median = median(x),
+         sd = sd(x), quantile = quantile(x), kurtosis = length(x) *
+             sum(x^4)/(sum(x^2)^2), skewness = sqrt(length(x)) *
+             sum(x^3)/(sum(x^2)^(3/2)))
+ }
> stat.des(x)
```

Usual object: `data.frame` object

It is a special `list` object. All components are vectors (of `numeric`, `character` or `logical`) with the same size.

- The function `data.frame` creates such an object:

```
> test.data <- data.frame(yf, x1, x2f)
```

- similar to `list` and `matrix` object for accessing to components:

```
> test.data$x1
> test.data[c(1, 3), ]
> test.data[["x1"]]
```

- editing a `data.frame`

```
> edit(test.data)
```

Some facilities with a `data.frame` object

- The function `str`, `summary` and `plot` applied to a `data.frame` give a good idea of the data:

```
> str(test.data)
> summary(test.data)
> plot(test.data)
```

- To change the names of the individuals:

```
> rownames(test.data) <- letters[1:10]
```

- or the names of the variables:

```
> colnames(test.data) <- c("y", "x1", "x2")
```

- Printing the beginning or the end of the `data.frame`:

```
> head(test.data, 2)
> tail(test.data, 2)
```

Working with a data.frame object

Function data is used to load a data.frame from the sources (basic or additionnal packages):

```
> data(iris)
> help(iris)
> str(iris)
```

The function attach applied to a data.frame permits to work directly on the variables.

```
> attach(iris)
> hist(Sepal.Length)
```

instead of:

```
> hist(iris$Sepal.Length)
```

But if it exists already a variable with same name, it may be "masked". Don't forget to use function detach before modifying the data.frame:

```
> detach(iris)
```

Modifying a `data.frame` object

We want to replace the observations "a", "c" and "e" of the variable x_1 by $\bar{x_1}$.

```
> x1.bar <- mean(test.data$x1)
```

These commands are equivalent:

```
> test.data[c("a", "c", "e"), 2] <- x1.bar
> test.data[c(1, 3, 5), 2] <- x1.bar
> test.data[c(1, 3, 5), "x1"] <- x1.bar
> test.data$x1[c(1, 3, 5)] <- x1.bar
> test.data$x1 <- replace(test.data$x1, c(1, 3,
+     5), x1.bar)
```

Adding a variable to a `data.frame` object

We want to create a binary variable equal to 1 if $x_1 > \bar{x_1}$ and 0 otherwise.

- The following commands:

```
> n <- nrow(test.data)
> x1.bin <- numeric(n)
> x1.bin[test.data$x1 > mean(test.data$x1)] <- 1
> test.data <- data.frame(test.data, x1.bin)
```

- are equivalent to:

```
> test.data$x1.bin <- ifelse(test.data$x1 > mean(test.data$x1),
+     "high", "low")
```

- function `merge` may be used to merge two `data.frame`. They must share a common key (identifier).

Aggregating a data.frame object

- function aggregate creates a data.frame:

```
> aggregate(cbind(x1, x1.bin) ~ y, data = test.data,
+     mean)
```

- function by prints results of aggregating:

```
> by(test.data[, c("x1", "x1.bin")], test.data$x2,
+     mean)
```

Importing/Exporting data

- If files *spain.txt*, *desbois.sav* and *donnees_insee_2006.csv* are included in the directory *C:/Pig session/*, we may change the current working directory of the R process:

  ```
  > setwd("C:/Pig session/")
  ```

- function `read.table` imports a text file:

  ```
  > spain <- read.table("spain.txt", header = TRUE)
  ```

  equivalent to:

  ```
  > spain <- read.table("C:/Pig session/spain.txt",
  +     header = TRUE)
  ```

- function `read.csv2` for csv file (obtained with Office Excel):

  ```
  > insee <- read.csv2("donnees_insee_2006.csv", header = TRUE)
  ```

Importing/Exporting data (2)

- others formats (SAS, Stata, SPSS, Access, etc) can be imported using functions included in the package **foreign**:

  ```
  > require(foreign)
  > farms <- read.spss("desbois.sav", to.data.frame = TRUE)
  ```

- functions `write.table`, `write.csv2`, etc. may be used to export a `data.frame` in a *.txt*, *.csv2* files.

General informations

Basic manipulation

Objects and Classes

Probability distributions

Computing

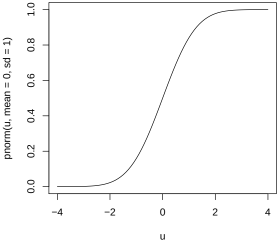
Graphics

Packages

Cumulative distribution function: $P(X \leq x)$

Example with the normal distribution:

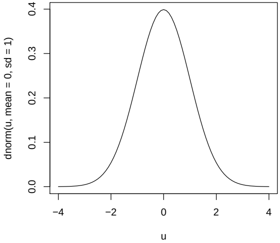
```
> u <- seq(-4, 4, 0.1)
> plot(u, pnorm(u, mean = 0,
+     sd = 1), type = "l")
```



Probability density function

Example with the normal distribution:

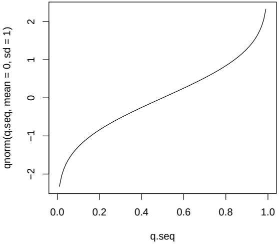
```
> plot(u, dnorm(u, mean = 0,
+      sd = 1), type = "l")
```



Quantile function $P(X \leq x) > q$

Example with the normal distribution:

```
> q <- seq(0, 1, 0.01)
> plot(q, qnorm(q, mean = 0,
+      sd = 1), type = "l")
```



Simulating from a distribution

Example with the normal distribution and representation of a “steam and leaf” plot of the simulated data:

```
> x.sim <- rnorm(100, 0, 1)
> stem(x.sim)

  The decimal point is at the |

  -2 | 51
  -1 | 9776544433322210000
  -0 | 99999887765555554444333221100
   0 | 0000233333444555556666777889999
   1 | 000112334455568
   2 | 1135
```

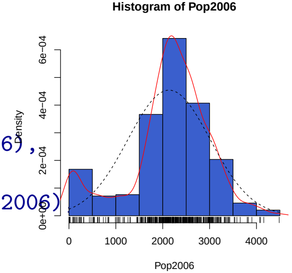
Available distributions

Distribution	R name	additional arguments
beta	beta	shape1, shape2, ncp
binomial	binom	size, prob
Cauchy	cauchy	location, scale
chi-squared	chisq	df, ncp
exponential	exp	rate
F	f	df1, df2, ncp
gamma	gamma	shape, scale
geometric	geom	prob
hypergeometric	hyper	m, n, k
log-normal	lnorm	meanlog, sdlog
logistic	logis	location, scale
negative binomial	nbinom	size, prob
normal	norm	mean, sd
Poisson	pois	lambda
signed rank	signrank	n
Student's t	t	df, ncp
uniform	unif	min, max
Weibull	weibull	shape, scale
Wilcoxon	wilcox	m, n

Examining the distribution of a set of data

Histogram, non parametric density estimation and normal distribution:

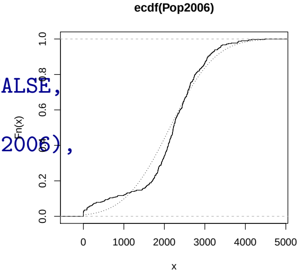
```
> attach(insee)
> hist(Pop2006, prob = TRUE,
+     col = "royalblue3")
> lines(density(Pop2006),
+     col = "red")
> u <- seq(min(Pop2006), max(Pop2006),
+     length.out = 100)
> lines(u, dnorm(u, mean = mean(Pop2006),
+     sd = sd(Pop2006)), lty = 2)
> rug(Pop2006)
> detach(insee)
```



Examining the distribution of a set of data (2)

Empirical cumulative distribution function compared to theoretical normal distribution:

```
> attach(insee)
> plot(ecdf(Pop2006), do.points = FALSE,
+     verticals = TRUE)
> lines(u, pnorm(u, mean = mean(Pop2006),
+     sd = sqrt(var(Pop2006))),
+     lty = 3)
> detach(insee)
```



Test of agreement with normality

- R provides the Shapiro-Wilk test:

```
> shapiro.test(insee$Pop2006)
> shapiro.test(x.sim)
```

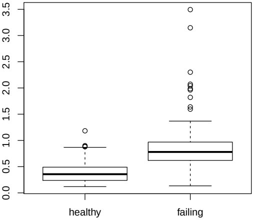
- and the Kolmogorov-Smirnov test:

```
> ks.test(insee$Pop2006, "pnorm", mean = mean(insee$Pop2006),
+     sd = sqrt(var(insee$Pop2006)))
> ks.test(x.sim, "pnorm", mean = 0, sd = 1)
```

One- and two-sample tests

The box plot and t-test:

```
> attach(farms)
> boxplot(r1 ~ DIFF)
> t.test(r1[DIFF == "healthy"],
+     r1[DIFF != "healthy"])
> detach(farms)
```



General informations

Basic manipulation

Objects and Classes

Probability distributions

Computing

Graphics

Packages

Loop (`for` and `while`)

- use `for` when the number of loops is fixed (= the size of the vector after `in`)
- use `while` when you have a stopping criteria
- use of braces `{}` around the statement is recommanded
- possibility to interupt a loop with `break` or `next`

```
> for (i in 1:10) {
+     print(i)
+ }
> som = 0
> for (j in -5:5) {
+     som = som + j
+     print(som)
+ }
> for (i in c(2, 4, 5, 8)) print(i)
> i = 0
> while (i < 10) {
+     print(i)
+     i = i + 1
+ }
```

Condition `if`, `else`, `ifelse`

- Classical call:

  ```
  if()
  {...}
  else
  {...}
  ```

- special call: `ifelse(test, 1st awnser, 2nd awnser).`

```
> y = z = 0
> for (i in 1:10) {
+     x = runif(1)
+     if (x > 0.5) {
+         y = y + 1
+     }
+     else {
+         z = z + 1
+     }
+ }
> y
> z
> x = rnorm(10)
> y = ifelse(x > 0, 1, -1)
```

Function

- f1 is the name of the created function and returns a numeric
- b=a gives a default value to b equal to a in the function f2
- function rate returns a list object, very useful when several informations to return

```
> f1 = function(x) {
+     return(x + 2)
+ }
> f1
> f1(3)
> f2 = function(a, b = a) {
+     a + b
+ }
> f2(a = 2, b = 3)
> f2(5)
> rate = function(p.begin, p.end, time) {
+     rate = (p.end/p.begin)^(1/time)
+     return(list(r = rate, t = time))
+ }
> result = rate(100, 500, 10)
> result$r
```

General informations

Basic manipulation

Objects and Classes

Probability distributions

Computing

Graphics

Packages

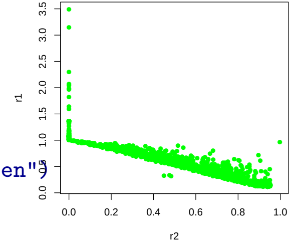
The `plot` function (1)

This is a generic function: the type of plot produced is dependent on the type or class of the first argument. If `x` and `y` are numeric variables, `plot(x,y)` is equivalent to `plot(y∼x)`.

```
> attach(farms)
> plot(r1 ~ r2, pch = 16,
+     col = "green")
```

gives the same plot than:

```
> plot(r2, r1, pch = 16, col = "green")
> detach(farms)
```



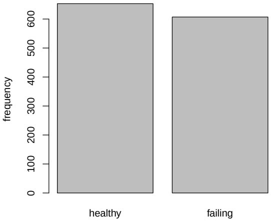
The `plot` function (2)

If `x` is a factor, `plot(x)` is equivalent to `barplot(table(x))`.

```
> attach(farms)
> plot(DIFF, col = "grey",
+      ylab = "frequency")
```

gives the same plot than:

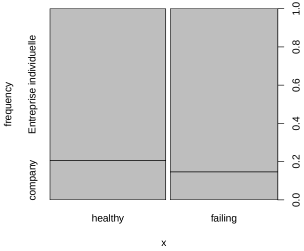
```
> barplot(table(DIFF), col = "grey",
+      ylab = "frequency")
> detach(farms)
```



The `plot` function (3)

If `x` and `y` are factor, `plot(x,y)`:

```
> attach(farms)
> plot(DIFF, STATUS, col = "grey",
+     ylab = "frequency")
> detach(farms)
```



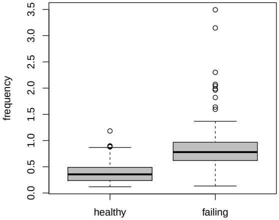
The `plot` function (4)

If `x` is a factor and `y` a numeric, `plot(y x)`:

```
> attach(farms)
> plot(r1 ~ DIFF, col = "grey",
+      ylab = "frequency")
> detach(farms)
```

gives the same plot than:

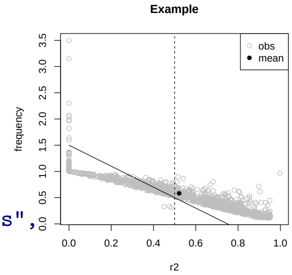
```
> boxplot(r1 ~ DIFF, col = "grey",
+      ylab = "frequency")
> detach(farms)
```



The points and lines function

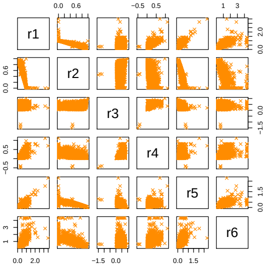
These function may be used on an existing plot, such as function abline, title, legend, etc.

```
> attach(farms)
> plot(r1 ~ r2, col = "grey",
+     ylab = "frequency")
> points(mean(r2), mean(r1),
+     pch = 16)
> u <- seq(0, 1, 0.01)
> lines(u, 1.5 - 2 * u)
> abline(v = 0.5, lty = 2)
> title("Example")
> legend("topright", legend = c("obs",
+     "mean"), pch = c(1,
+     16), col = c("grey",
+     "black"))
> detach(farms)
```

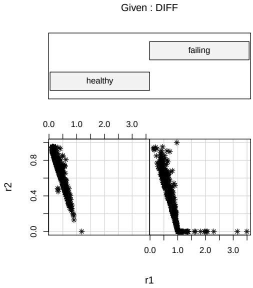


Displaying multivariate data

```
> pairs(farms[, 9:14], pch = 4,
+       col = "darkorange")
```



```
> attach(farms)
> coplot(r2 ~ r1 | DIFF, pch = 8)
> detach(farms)
```



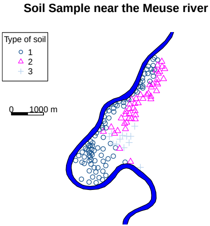
Graphical options

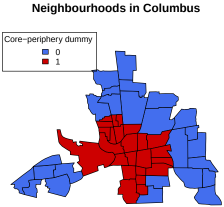
- See help(plot.default) for details on the options
- For the x-axis, legend, title, etc. use the function text, mtext, axis, title
- see example(plotmath) for mathematical annotation
- functions locator and identify are used for interactivity. See example(identify).

```
> example(identify)
> plot(1:10)
> identifyPch(1:10)
```

- Other graphical functions: dotchart, image, contour, persp. See the help or example of these functions.

Map with R


Soil Sample near the Meuse river
Type of soil
○ 1
△ 2
+ 3
0
1000 m


Neighbourhoods in Columbus
Core−periphery dummy
0
1

General informations

Basic manipulation

Objects and Classes

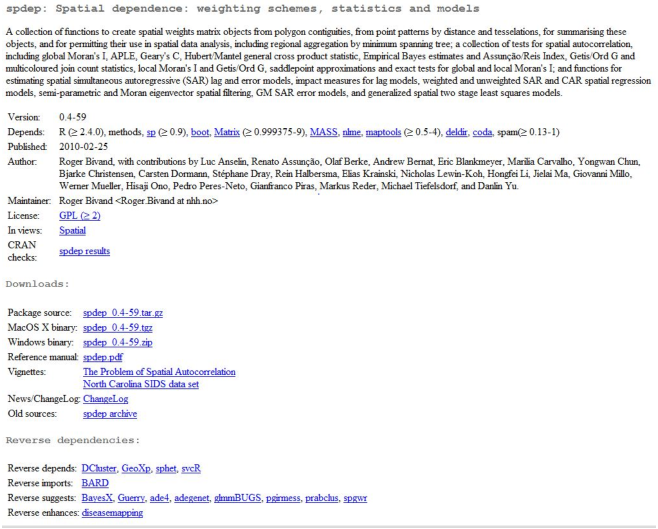
Probability distributions

Computing

Graphics

Packages

What is an R package ?



```
spdep: Spatial dependence: weighting schemes, statistics and models
```

A collection of functions to create spatial weights matrix objects from polygon contiguities, from point patterns by distance and tesselations, for summarising these objects, and for permitting their use in spatial data analysis, including regional aggregation by minimum spanning tree; a collection of tests for spatial autocorrelation, including global Moran's I, APLE, Geary's C, Hubert/Mantel general cross product statistic, Empirical Bayes estimates and Assunção/Reis Index, Getis/Ord G and multicoloured join count statistics, local Moran's I and Getis/Ord G, saddlepoint approximations and exact tests for global and local Moran's I; and functions for estimating spatial simultaneous autoregressive (SAR) lag and error models, impact measures for lag models, weighted and unweighted SAR and CAR spatial regression models, semi-parametric and Moran eigenvector spatial filtering, GM SAR error models, and generalized spatial two stage least squares models.

Version:	0.4-59
Depends:	R (≥ 2.4.0), methods, sp (≥ 0.9), boot, Matrix (≥ 0.999375-9), MASS, nlme, maptools (≥ 0.5-4), deldir, coda, spam(≥ 0.13-1)
Published:	2010-02-25
Author:	Roger Bivand, with contributions by Luc Anselin, Renato Assunção, Olaf Berke, Andrew Bernat, Eric Blankmeyer, Marilia Carvalho, Yongwan Chun, Bjarke Christensen, Carsten Dormann, Stéphane Dray, Rein Halbersma, Elias Krainski, Nicholas Lewin-Koh, Hongfei Li, Jielai Ma, Giovanni Millo, Werner Mueller, Hisaji Ono, Pedro Peres-Neto, Gianfranco Piras, Markus Reder, Michael Tiefelsdorf, and Danlin Yu.
Maintainer:	Roger Bivand <Roger.Bivand at nhh.no>
License:	GPL (≥ 2)
In views:	Spatial
CRAN checks:	spdep results

```
Downloads:
```

Package source:	spdep_0.4-59.tar.gz
MacOS X binary:	spdep_0.4-59.tgz
Windows binary:	spdep_0.4-59.zip
Reference manual:	spdep.pdf
Vignettes:	The Problem of Spatial Autocorrelation North Carolina SIDS data set
News/ChangeLog:	ChangeLog
Old sources:	spdep archive

```
Reverse dependencies:
```

Reverse depends:	DCluster, GeoXp, sphet, svcR
Reverse imports:	BARD
Reverse suggests:	BayesX, Guerry, ade4, adegenet, glmmBUGS, pgirmess, prabclus, spgwr
Reverse enhances:	diseasemapping

Install packages needed for exercises

Some contributed packages required over and above the base and recommended packages installed with R should now be installed from a CRAN mirror. If you chose the CICT mirror earlier, that choice will still apply. You may use the Packages menu if you like, but with well over 3000 contributed packages on CRAN, the command line has its attractions. Consider copying and pasting from the displayed script (see the right-click menu if displaying within R).

```
> install.packages(c("caschrono",
+     "GeoXp"))
```

The End

Thanks for attention.
thibault.laurent@univ-tlse1.fr
Tél: 88-99