Whats' beyond Concerto: An introduction to the R package *catR*

Session 2:

Overview of R software and *catR* package

The Psychometrics Centre, Cambridge, June 10th, 2014

Outline:

1. The R software

2. R GUI and RStudio

3. R language

4. R packages

1. The R software

R was created in 1997

Similar to the S language (used in e.g., S-PLUS)...

... but is open source and freely accessible

Network of developpers and researchers in R

R website: http://www.r-project.org

Specific meetings and conferences (useR!)

Regular updates and personnal improvements:

- first version R-0.49 released April 23, 1997
- last version R-3.1.0 released April 10, 2014

Increasing litterature about R and its applications

1. The R software

R can handle a variety of tasks:

- Descriptive statistics
- Basic and advanced figures
- Basic and advanced statistical methods (ANOVA, PCA, factor analysis, GLM, bootstrap, neural networks, time series, ...)
- Easy data generation processes and design of simulation studies
- Interaction with other statistical software (e.g. WinBUGS)
- Easy loading and management of data sets from Excel, SAS, SPSS, ...
- Personal implementation and optimization possible (through e.g. packages)

1. The R software

Not all these features will be looked at today!

Some useful references and reading:

- Spector, P. (2008). *Data manipulation with R.* New York: Springer.

- Zuur, A. F., Ieno, E. N., & Meesters, E. H. W. G. (2009). *A beginner's guide to R.* New York: Springer.

- ... and many other nice books from the UseR! series

Check also my slides from former workshop on R (Trier, Germany, 2013):

http://hdl.handle.net/2268/147096
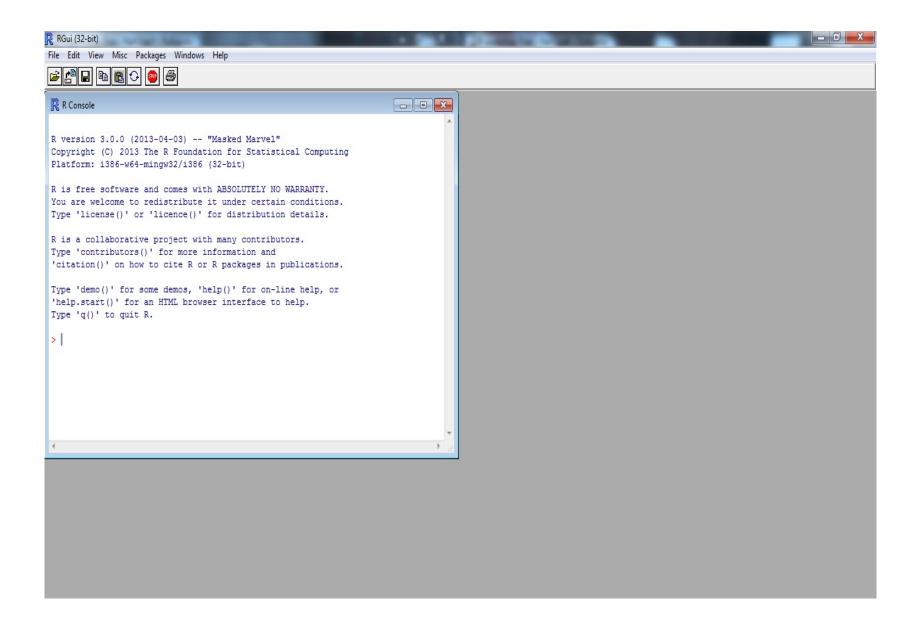
2. R GUI and RStudio

If you install R from the CRAN (Comprehensive R Archive Network) website:

http://www.r-project.org

then you will install the GUI (graphical user interface) of R

R GUI is the basic interface to program in R...

... so it is not necessarily the most fancy interface :-)

2. R GUI and RStudio

2. R GUI and RStudio

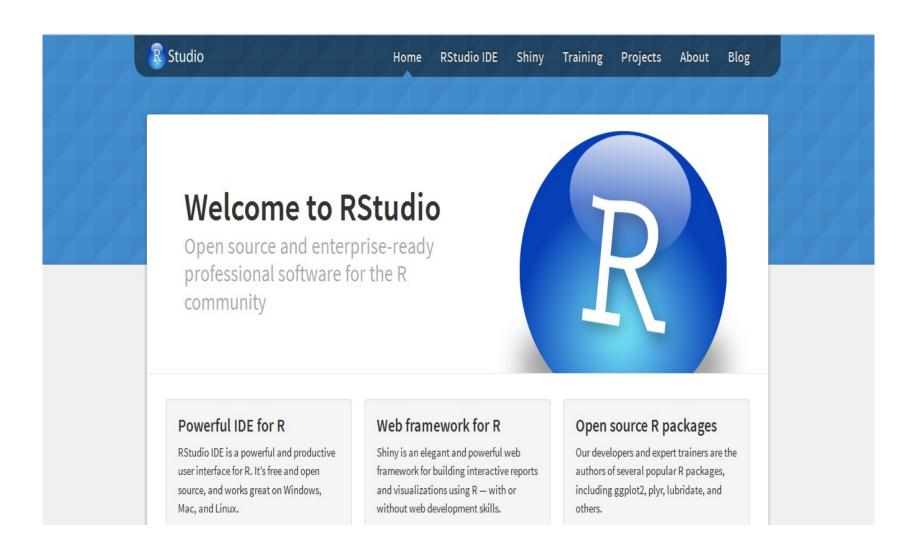Fortunately (for you) it exists a more user-friendly interface: the RStudio software

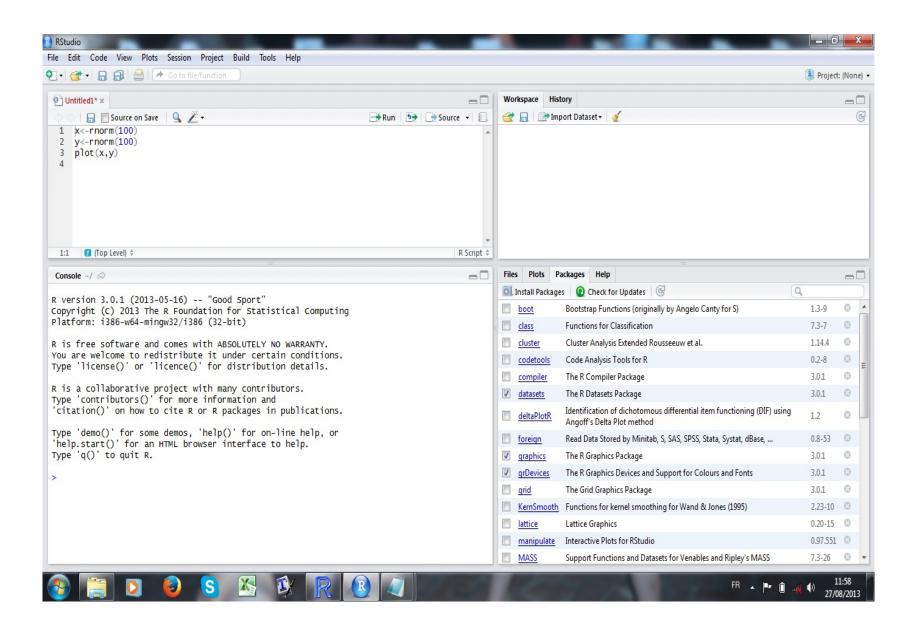RStudio can be downloaded and installed freely from

https://www.rstudio.com/

RStudio works similarly to R GUI but optimizes windows and display information in a much more enhanced way

RStudio definitely to be preferred for non-crazy R users...

... though I will illustrate *catR* with R GUI :-)

2. R GUI and RStudio

2. R GUI and RStudio

3. R language

R has a specific syntax and all instructions written in the console will be executed in a line-by-line approach

Errors will be mentioned and execution will stop

Basic example: let's write "3+2" in the console:

```
R> 3+2
```

and execute it by pressing ENTER

(Important: R> must NOT be written in the console!)

Let's see how R reacts...

3. R language

The following appears in the console:

```
> 3+2
[1] 5
>
```

Surprisingly, R computed 3+2 and returned value 5 :-)

Preceding symbol [1] helps in identifying the component number when a vector is returned

3. R language

Now, let's execute "3+a" in the console

```
R> 3+a
```

Output is

```
> 3+a
Error: object 'a' not found
>
```

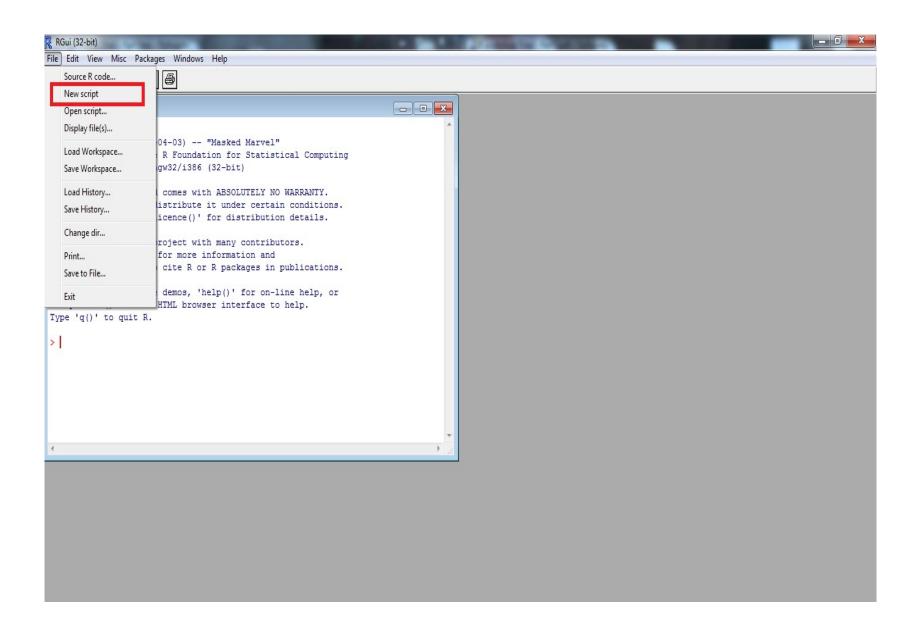Since object **a** was not yet created in R, it doesn't exist and R detects an error

Error message is returned, process stops and nothing more is done
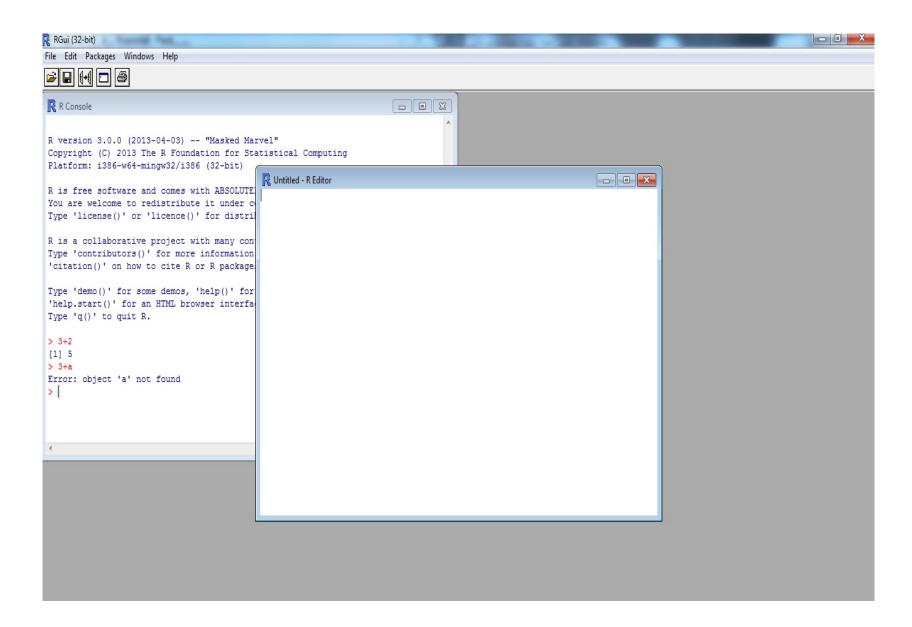
3. R language

Most often, analyses in R require several successive steps of calculation

One mistake stops the whole process and all steps must be written once again...

More efficient to write all R code in a script file first, then copy-paste (or execute it) in the console

Any text editing software (Word, Notepad etc.) is convenient, but R has an internal script editor (and RStudio opens automatically an R script window)

3. R language

3. R language

3. R language

Within the script window, any R code can be written

Nothing is being executed at once in the console

To execute some R code written in the script:

1. select the R code with the mouse
2. Click-right and select "Run line or selection"
   (this step can be replaced by CTRL+R)

3. R language

R language is based on several elements:

- predefined functions that are available internally
- additional functions available in packages (from CRAN or elsewhere)
- internal variables (to store data or output)
- vectors and matrices
- data frames and lists

3. R language

Functions are called by their name, and one or several arguments must be specified (in parentheses)

Example internal functions are:

- basic calculations: `exp()`, `log()`, `abs()`, `sqrt()`, `cos()`, `sin()`, ...
- graphics: `plot()`, `hist()`, `pie()`, `boxplot()`, ...
- basic statistics: `mean()`, `sd()`, `var()`, `quantile()`, `cor()`, ...
- distributions: `dnorm()`, `dchisq()`, `dpois()`, `dbinom()`, `dt()`, ...
- standard tests: `t.test()`, `chisq.test()`, `ks.test()`, `wilcox.test()`, ...
- advanced statistical models and methods: `glm()`, `prcomp()`, `factanal()`, `anova()`, `ksmooth()`, ...

3. R language

Some functions may not be available by default (see later)

Before going on, two important remarks:

- R makes distinction between lowercase and uppercase letters: `abs()` works, but not `Abs()`!

- Each function in R (either internally or from external package) has a help file with (sometimes) useful information
  - can be accessed by typing first

    `?`

    directly followed by the function name, without ()
  - for instance,
    ```
    R> ?anova
    ```
    opens the help file related to `anova()` function

3. R language

Internal variables may be created to store data and provisional or final results

Variable name is up to you (remember that lowercase and uppercase are not the same!)

Storing some value in a variable is done by means of the logical symbol

$$<-$$

Example: let's write

```
R> x <- 5 + 2
```

R will create an internal variable x with value 7 (to see it, write down x in the console)

3. R language

Matrices can also be created and handled in R

Matrices are arrays with rows and columns

Elements of a matrix identified by means of two indicators, one for row number, one for column number (in this order)

Consider a $I \times J$ matrix stored in the R variable `mat`

Element (i, j) of `mat` is obtained by

$$\texttt{mat[i,j]}$$

i-th row and j-th column of `mat` are obtained by

$$\texttt{mat[i,]} \text{ and } \texttt{mat[,j]}$$

3. R language

Matrices should contain only numeric values

However, data sets might hold mix of numeric and character / qualitative variables (such as category names)

Handled in R by means of data frames

Data frames are lists of R objects of different formats, most often vectors of numeric or character values of the same length

Display in a "data set" format with column names

Each column is an element of the list (i.e. a vector) and the column name is the variable name

Hint: most often, data sets loaded in R have the data frame format (see later)
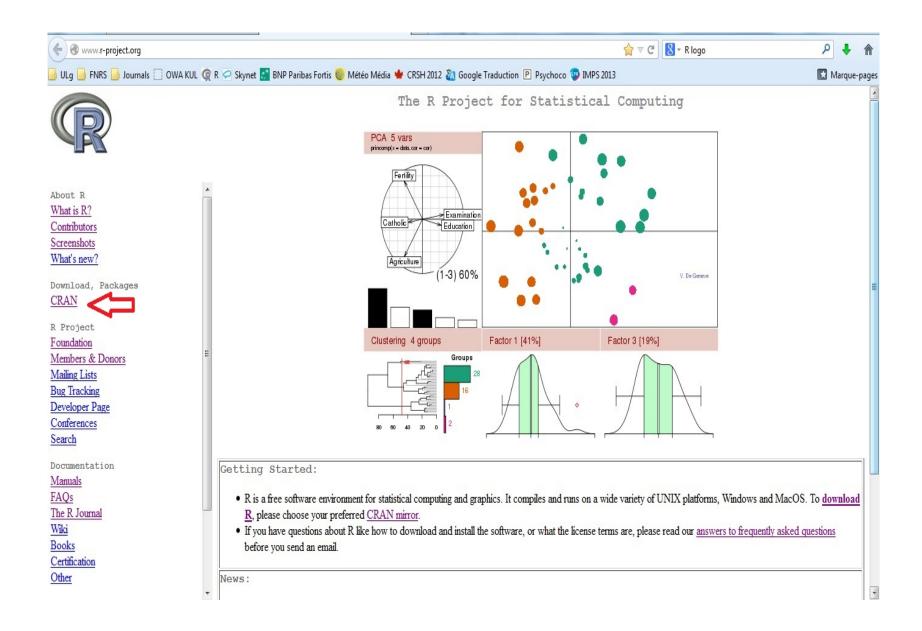
4. R packages

R holds a lot of predefined functions for basic statistics and graphics
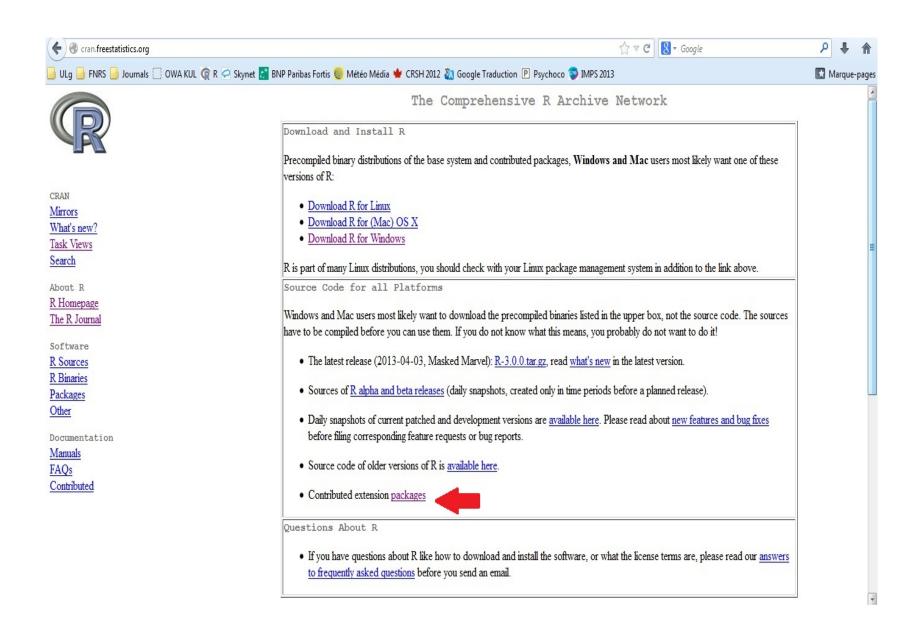
However, R can also be used to develop specific routines... such as CAT applications
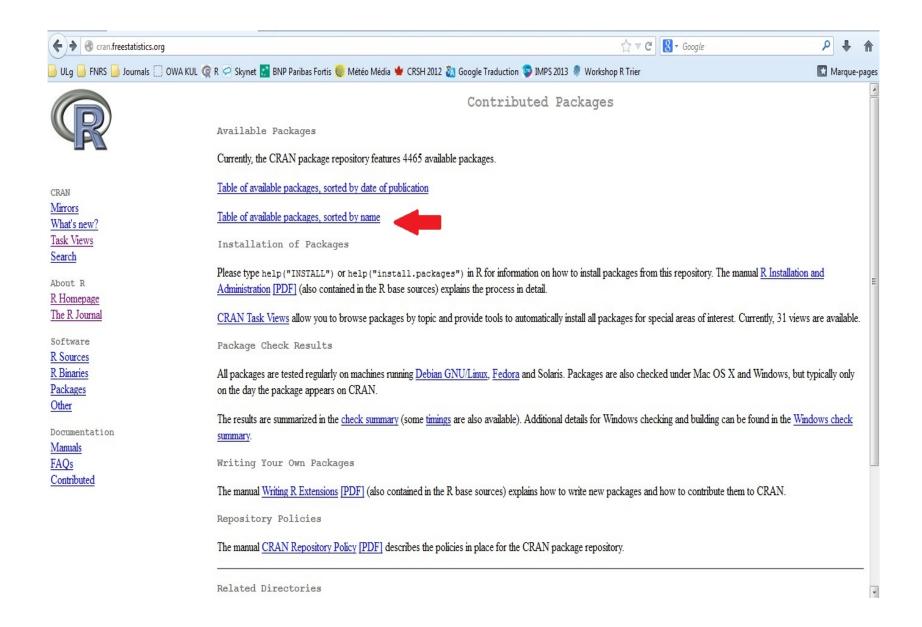
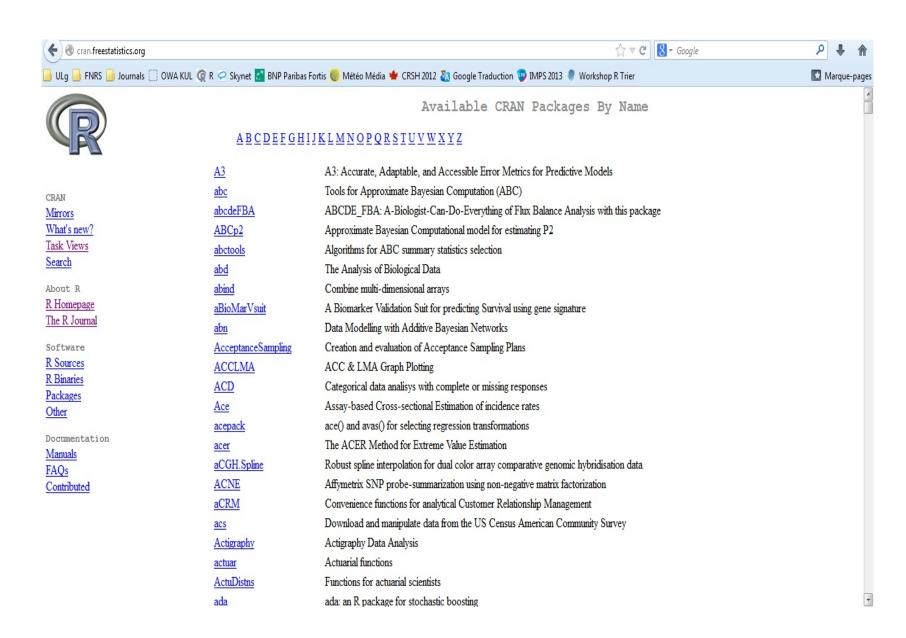Easiest way to get stable and portable R code: creation of an R package

Packages hold several functions, with a full description (help file)

Most of them available from CRAN:

- 4465 packages on April 27th, 2013
- 5540 packages on May 14th, 2014
- ... how many on June 10th, 2014?

4. R packages

4. R packages

4. R packages

4. R packages

4. R packages

Best way to install an R package: make use of "Install" menu in
  R GUI / RStudio

Packages must be installed once and loaded with each new session
  of R

Packages must be re-installed after each update of R itself

To load an R package (for instance *catR*):

```
R> require(catR)
```

This allows R to access to all functions within *catR* package