

Chap I: Introduction to R: First steps

A general introduction

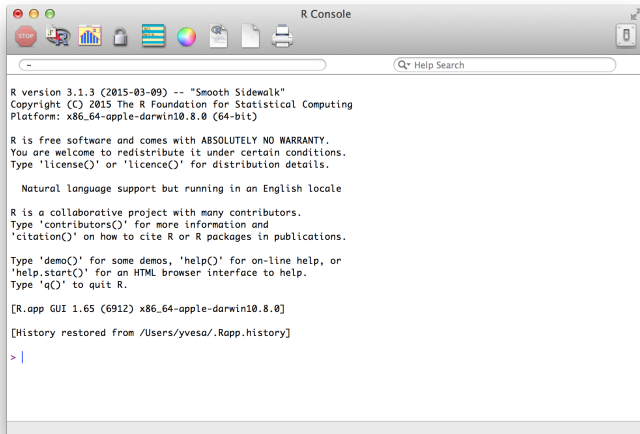
- ▶ R is a software for statistical computing and data analysis. It is also a programming language.
- ▶ R is freely distributed software (www.r-project.org) with contributions from developers from around the world. It is one of the main software in statistics.

A general introduction

- ▶ There are a couple of other competing commercial developments:
 - ▶ RStudio (partly free).
 - ▶ R Revolution (not free).
- ▶ Many of the big players in “data analytics” are integrating R in their software.
 - ▶ *Oracle R Enterprise*.
- ▶ Bottom line: there is a lot to gain by learning R.

A general introduction

R interface is very frugal.



```
R Console

R version 3.1.3 (2015-03-09) -- "Smooth Sidewalk"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin10.8.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

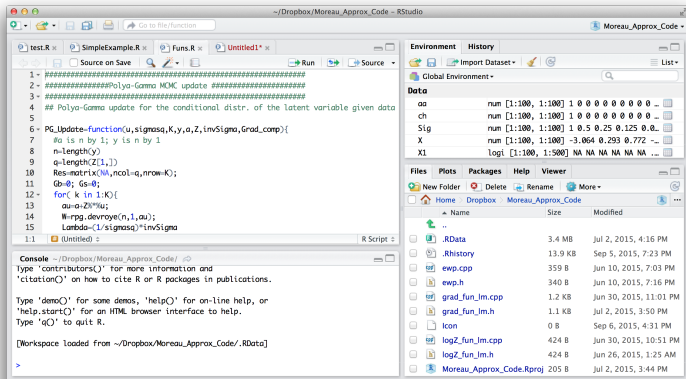
[R.app GUI 1.65 (6912) x86_64-apple-darwin10.8.0]

[History restored from /Users/yvesa/.Rapp.history]

> |
```

A general introduction

RStudio is a current development effort to bring up more functionality into the interface.



A general introduction

- ▶ As programming language, R is an interpreted language, as opposed to a compiled language. This is very useful for interactive data analysis.
- ▶ The software R itself contains a lot of pre-programmed mini-software (packages) that the user can call and use for various analysis.

Getting started

R offers two ways of working :

1. "A conventional approach": you open a file and write a program describing what you intend to do and run that program.
2. "An interactive approach": you interact with R and work one step at the time. We type in expressions and R evaluates them and returns a value if needed.

Getting started

- ▶ To start or quit R or Rstudio you can use the menu.

- ▶ To get help

```
> help.start().    #To access $R$ manual in HTML format
```

```
#Or
```

```
> ?sin #for specific help
```

(for example on the function 'sin').

- ▶ Note: the sign # marks the begin of a comment. The remaining of the line is not executed.

Getting started

1. An important concept in computer programming is the concept of variable. A variable in computer science is a name given to some storage location. In more practical terms, it is a binding between a symbol and a value. There are similarity with variables in mathematics.

```
x=20
```

```
y<-x+1
```

```
x+2->z
```

```
assign('u',x+3)
```

Getting started

Some guidelines and advices when creating variables:

1. A variable cannot start with a digit or underscore symbol _.
2. Variable naming is case sensitive.
3. Use meaningful names and avoid names that already exist in R (use "exists" to check) such as 'c', 'pi', 'ls', etc...

```
x<-pi
```

```
X=2
```

```
u=c(x,X)
```

```
exists('c')
```

Getting started

- ▶ A variable can have a type (or mode). This has implication on how it is stored and what kind of operations it supports.
- ▶ R supports many different modes: integer, double, logical, character and complex.
- ▶ However R is very loosely typed (a blessing and a curse!)

```
y=integer(1); y=2.3335;
```

```
x=2; #x's precise type is not very important
```

```
z='stat 406'
```

```
substr(z,start=1, stop=4)
```

```
substr(y,start=1, stop=4)
```

Getting started

It is important to understand R's organization.

As you create new variables in R, there are kept in the computer memory. It is useful sometimes to know what variables are currently in memory and be able to save or delete them.

```
ls()
```

```
ls.str() # both command lists existing variables
```

```
x=rnorm(1);u<-x+1;
```

```
save(list=c('x','u') , file='x1.RData') # save x and u  
                                         in file 'x1.RData'
```

```
rm(list=c('x','u')) # use rm(list=ls()) to delete all
```

```
load(file = 'x1.RData')
```

Getting started

- ▶ At the beginning of each R section, a directory is attached to the section called the 'working directory'.

```
> getwd(); # To see the current working directory  
> setwd('~ /Yves/Work/Teaching/Stat_406/  
Computing_in_Class');  
# To set the working directory.
```

Most of these can be easily done through the menu in R and Rstudio.

Getting started

- ▶ Whenever you try to read or save a file without the full path, the working directory (wd) will be used.
- ▶ At the end of an R session, you can choose to save all the objects in memory.
- ▶ A file “.RData” is then created for this purpose. You can choose to rename the file.
- ▶ You can later load the file using `load('filename')`.

Getting started

- ▶ Another important concept to know is the search directories. That is the sequence of Environments in which R searches for whatever variable or function you request.
- ▶ You can see that hierarchy with 'search()'. This hierarchy changes as you add or remove packages to your R session.
- ▶ The same variable name can exist in different environment.

Getting started

Example:

```
pi
```

```
find('pi')
```

```
pi=2
```

```
find('pi')
```

```
base::pi
```

```
# returns the variable pi from the package package:base
```


Getting started

- ▶ Beside variables, functions are the other most important concept in computer programming.
- ▶ A function is a piece of code that takes some input called arguments, performs a specific task and possibly returns a value.
- ▶ In order to correctly use a function we must properly set up its arguments.

Getting started

- ▶ In R we specify arguments either by name or by position.
- ▶ Suppose we want to generate 100 random variables $\mathbf{N}(0, 4)$. We need to use the function `rnorm`.

```
u<-rnorm(100,0,2)  #correct but not recommended
x=rnorm(n=100,mean=0,sd=2); #recommended use
y=rnorm(mean=0,sd=2,n=100); #correct but not recom.
```

- ▶ You can call the help information of a given function by using the function `'?'`.

```
?rnorm
```

- ▶ In Rstudio, you can use the `tab` key to get the code completion menu.

Getting started

- ▶ Many functions have default values for their arguments. By reading the help information on the function you can learn about which argument is required, which has default values.

- ▶ Example: to generate one r.v. $\mathbf{N}(0, 1)$, one could do:

```
u<-rnorm(1)  #correct but not recommended
```

- ▶ The following code also correctly generate 100 random variables $\mathbf{N}(0, 4)$. Can you explain why?

```
y=rnorm(s=2,n=100)  #correct but not recommended
```

Activity:

Generate $n = 1000$ i.i.d. random variable from $\mathbf{N}(0, 1)$, and plot their histogram. Histograms are done with the function "hist". Check the help to learn how to use that function.

To summarize:

- ▶ We will interchangeably use R and Rstudio in this class, but Rstudio is more user-friendly.
- ▶ All variables have a type (or mode). R is only loosely typed and type conversion is often automatic.
- ▶ R has many functions. We specify functions' arguments either by name (recommended) or by position. Read the help to learn how to properly set up the arguments of a function.
- ▶ You can save your objects using the function "save". This creates a binary file in your working dir. You can also load these binary files using the function "load".
- ▶ At the beginning of each R session, always set your working directory.