POLITECNICO DI MILANO

Introduction to R

3rd lecture

Alessandro FERMI – Giovanna VENUTI

Outline of the lecture

In this lecture we will introduce

- How to access builtin datasets
- Built-in probability distributions
- Descriptive statistical tools in R
- Basic built-in tools for hypothesis testing

Remind: lists and data frames

In the first two lectures we have defined important data structures in R, i.e.

- arrays, that are defined through the “array()” function by the syntax

  Z <- array(data_vector, dim_vector)

- matrices, defined by the “matrix()” function: the general syntax is

  M <- matrix(data_vector, n_rows, n_cols)

- lists, very general recursive structures that are built by means of the “list()” function, whose general syntax is

  Lst <- list(name_1=object_1,…, name_k=object_k)

- data frames, that are particular lists with some restrictions on the components and that may defined by

  Data <- data.frame(name_1=object_1,…,name_k=object_k)

Remind: lists and data frames

To access the components of a data frame, or more generally a list, recall that its components are always numbered and can be accessed through the “[[.]]” operator, e.g.

```
>   Lst <- list(name="Fred", wife="Mary", no.children=3,
child.ages=c(4,7,9), name.children=c(“Tizio”,”Caio”,”Sempronio”))
>   Lst[[1]]
[1] "Fred"
```

or the components can be named in which case the following notation may be used

```
>   name$component_name
```

Example

```
>   Lst$name.children
[1] «Tizio» «Caio» «Sempronio»
```

Examples of data frames

R is supplied with more than 100 datasets that are stored in various packages. To give a look at the datasets currently loaded, the command

```
> data()
```

can be used. This gives an overview of the data sets available in the package “datasets”.
To list the data sets in all available packages, use

```
> data(package = .packages(all.available = TRUE))
```

All datasets contained in the packages currently loaded are directly available by name, e.g.

```
> data()
> cars
```

Built-in datasets

Remark: some packages may still use the obsolete convention in which "data()" was also used to load datasets into R, for example

```
> data()
> data(cars)
```

In order to see which pakages are loaded, use the command

```
>  search()
[1] ".GlobalEnv"        "package:stats"     "package:graphics"
[4] "package:grDevices" "package:utils"     "package:datasets"
[7] "package:methods"   "Autoloads"         "package:base"
```

Built-in datasets

If you need some data sets stored in some other packages that is not yet in the search path, you may either load its content with the function

```
>   library(name_package)
```

Example. The package “boot” can be loaded using

```
>   library(boot)
>   search()
[1] ".GlobalEnv"        "package:boot"      "package:stats"
[4] "package:graphics"  "package:grDevices" "package:utils"
[7] "package:datasets"  "package:methods"   "Autoloads"
[10] "package:base"
```

Otherwise, you can directly load the chosen dataset with the command

```
>   data(acme, package='boot')
>   ls()
```

Data frames from external files

The simplest way to construct a data frame from scratch is to use the “read.table()” function to read an entire data frame from an external file.

R input facilities are simple and rather inflexible. The files to be read must have already a specific form obtained by using other editors.
For instance, assume to have a file of GNSS Zenith Wet Delays

NAME	FLG	YYYY MM DD HH MM SS	MOD_U	CORR_U	SIGMA_U	TOTAL_U
COMO	P	2008 10 26 00 00 00	2.2286	0.16834	0.00140	2.39698
COMO	P	2008 10 26 01 00 00	2.2286	0.16491	0.00102	2.39355
COMO	P	2008 10 26 02 00 00	2.2286	0.15606	0.00084	2.38470
COMO	P	2008 10 26 03 00 00	2.2286	0.15900	0.00069	2.38764
COMO	P	2008 10 26 04 00 00	2.2286	0.14919	0.00094	2.37783
COMO	P	2008 10 26 05 00 00	2.2286	0.14156	0.00087	2.37020

Data frames from external files

Use the read.table() function to import these data into a data frame object

```
>   filepath <- ‘C:/Users/Alessandro/Documents/R_Work/COMOALL.TRP’
>   zwd <- read.table(filepath)
>   zwd
```

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12
1	NAME	FLG	YYYY	MM	DD	HH	MM	SS	MOD_U	CORR_U	SIGMA_U	TOTAL_U
2	COMO	P	2008	10	26	00	00	00	2.2286	0.16834	0.00140	2.39698
3	COMO	P	2008	10	26	01	00	00	2.2286	0.16491	0.00102	2.39355
4	COMO	P	2008	10	26	02	00	00	2.2286	0.15606	0.00084	2.38470
5	COMO	P	2008	10	26	03	00	00	2.2286	0.15900	0.00069	2.38764
6	COMO	P	2008	10	26	04	00	00	2.2286	0.14919	0.00094	2.37783
7	COMO	P	2008	10	26	05	00	00	2.2286	0.14156	0.00087	2.37020

To omit the first line use the command

```
>   zwd <- read.table(filepath, header=TRUE)
>   class(zwd)
[1] "data.frame"
```

Load data frames in a package

Now to check which datasets are stored in the “datasets” package write

```
>  data(package=”datasets”)
>  morley
>  filepath <- system.file("data", "morley.tab" , package="datasets"
>  filepath
[1] "C:/PROGRA~1/R/R-2~1.0/library/datasets/data/morley.tab"
>  morley <- read.table(filepath)
>  morley
>  class(morley)
[1] "data.frame"
```

Save data frames in a file

The counterpart of the function read.table() that may be used to save the changes on your data frame in an external file is the function write.table().
For instance the command

```
>   write.table(zwd,
'C:/Users/Alessandro/Documents/R_Work/COMOALL.TRP')
```

store the (changed) data frame in the file COMOALL.TRP.

R Editor

When invoked on a data frame or matrix, the function edit() brings up a separate spreadsheet-like environment for editing.

This is useful for making small changes once a data set has been read. The command

```
> xnew <- edit(xold)
```

will allow you to edit your data set xold, and on completion the changed object is assigned to xnew.

If you want to overwrite the original dataset xold, the simplest way is to use fix(xold), which is equivalent to xold <- edit(xold).

Use > xnew <- edit(data.frame()) to enter new data via the spreadsheet interface.

Data frame from an external file

EXAMPLE

- create a data frame containing the data in the external file “COMOALL.TRP” in your working direction (to get the path of your working directory use the command getwd())

- check that it is a data frame

- explore it and compute some basic descriptive statistics of the component CORR_U and TOTAL_U, using the functions mean(), var() length(), etc.

- define the 1D statistical variable with the data set zwd$CORR_U (I defined the classes of the s.v. using the command

```
> rel <- factor(cut(zwd$CORR_U, breaks=0.0 + 0.04*(0:20)))
> rel
> zwd_afreq <- table(rel)
> zwd_freq  <-  prop.table(zwd_afreq)
```

Data frames – attach() function

When working with data frames or lists, the notation $ notation is not always convenient.

For many purposes, it could be useful to make the components of the data frame or list temporarily visible in the current workspace.

This is achieved by means of the “attach()” function, e.g.

```
>   attach(morley)
```

This makes the data frame visible in the search path at position 2 (or above) and the components can be used as variables in their own right.

Data frames – attach() function

More precisely, write in the console the following commands

```
>   data()
>   morley
>   search()
>   ls(1)
```

The data frame ‘morley' is loaded, but it is not present in the search path.
Attaching it means making it visible in the search path

```
>   attach(morley, package=‘datasets’)
>   search()
>   ls(2)
>   ls(1)
```

Data frames – attach() function

At this point, an assignment like the following is possible

```
>    aux <- Speed / 1000
```

This does not change permanently the “morley” data frame.
If permanent changes have to be stored, then one has to use the $ notation specifying the name of the data frame.

Example

```
>   aux <- Speed / 1000
>   morley$Speed <- aux
```

However the new value is not visible until the data frame is detached with the “detach()” function

```
>  detach(morley)
>  ls(2)
```

Statistical tables in R

Having available a dataset stored in a data frame, it can be very useful to examine its distribution; as an other example, first of all, let us find the 'faithful' data frame

```
>   data(package=”datasets”)
>   faithful
>   class(faithful)
[1] "data.frame"
>   attach(faithful)
>   ls(2)
[1] "eruptions" "waiting«
>   summary(eruptions)
 Min.    1st Qu.  Median    Mean  3rd Qu.    Max.
  1.600   2.163    4.000    3.488   4.454    5.100
```

Statistical tables in R

The function summary() gives some basic statistical infos about the dataset at hand.

An alternative to the summary() function is the function fivenum() that gives a slightly different output. More precisely it returns Tukey's five number summary (minimum, lower-hinge, median, upper-hinge, maximum) for the input data.

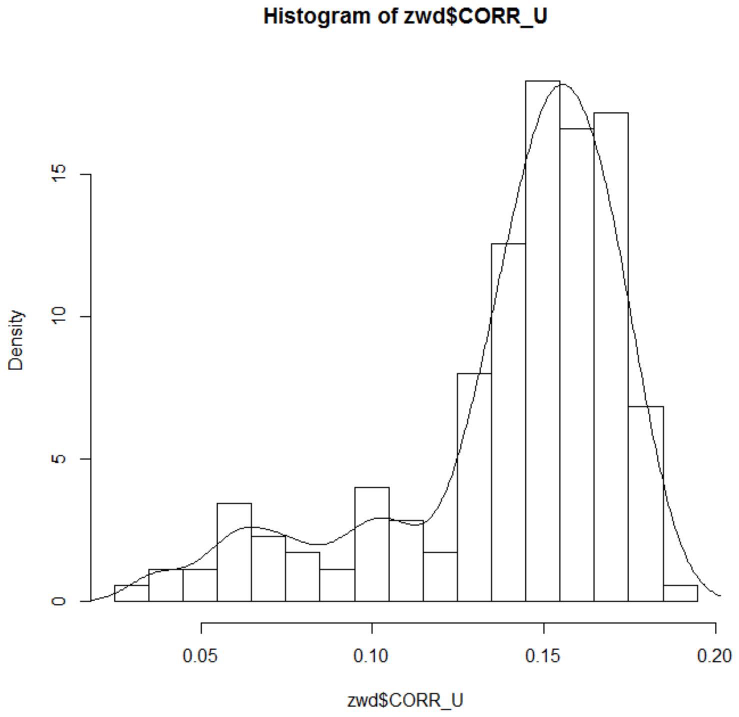
```
>  summary(eruptions)
 Min.    1st Qu.  Median    Mean  3rd Qu.    Max.
 1.600   2.163   4.000     3.488   4.454     5.100
> info <- fivenum(eruptions)
> info
[1] 1.6000 2.1585 4.0000 4.4585 5.1000
```

Statistical tables in R

Very similar to MATLAB, R has a hist() function to plot histograms.
Continuing the example with the data frame "faithful", use the commands

```
>   hist(eruptions)
> hist(zwd$CORR_U, seq(min(zwd$CORR_U)-0.01,
max(zwd$CORR_U)+0.01, 0.01))
> lines(density(zwd$CORR_U))
```

The function density() computes the kernel density estimation.
There are different kernel models that may be used.
See help(density) for more infos.



Statistical tables in R

We can also plot the empirical cumulative distribution function by using the function ecdf.

```
>   plot(ecdf(eruptions), do.points=FALSE, verticals=TRUE)
```

If you want to consider (and plot) subsets of the given data set, you may use the index vector system (seen in the first lecture). For example

```
>   long3 <- eruptions[eruptions > 3]
>   F_long3 <- edcf(long3)
>   F_long3
Empirical CDF
Call: ecdf(long3)
 x[1:79] =  3.067,  3.317,  3.333,  ...,  5.067,    5.1
```

Statistical tables in R

```
>   class(F_long3)
[1] "ecdf"     "stepfun"  "function"
>   plot(F_long3, do.points=FALSE, verticals=TRUE)
```

We can superimpose a theoretical distribution function model. In this case we may try a normal distribution

```
>   x <- seq(3, 5.4, 0.01)
>   lines(x, pnorm(x, mean=mean(long3), sd=sqrt(var(long3))), lty=3)
```

The argument ‘lty = 3’ is the line type chosen, in this case a dotted line

Statistical tables in R

Example. Use the functions seen so far to analyse the data frame «zwd».

Probability distributions in R

In R all most important probability distributions are implemented. A list of them can be found in the manuals.

Functions are provided to evaluate density and distribution functions and to compute any quantile $P(X < l) > q$.

Prefix the name of the probability distribution by

- 'd' for the density,
- 'p' for the CDF,
- 'q' for the quantile function
- 'r' for simulation (random deviates).

The first argument is x for dxxx, q for pxxx, p for qxxx and n for rxxx

Probability distributions in R

Example

- extract a sample from Student’s t-distribution with degree of freedom equal to 10 (for instance)
- use the function qqnorm() to compare this sample with the normal distribution
- extract a sample from the Fischer distribution (degree of freedom df1 = 5, df2 = 7) and compare this sample with the normal distribution

Remark. To generate a random sample from a uniform distribution You may also use the ‘runif()’ function.
Moreover to generate a random vector of integers the function ‘sample()’ is available.

Hypothesis testing in R

In R all «classical» tests for hypothesis testing are implemented!
Let us continue the example with the data frame ‘faithful’.

Example We can carry out a Kolmogorov-Smirnov test on the shape of the distribution density

```
>   ks.test(long3, "pnorm", mean = mean(long3), sd = sqrt(var(long3)))

One-sample Kolmogorov-Smirnov test

data:  long3
D = 0.0661, p-value = 0.4284
alternative hypothesis: two-sided
```

Hypothesis testing in R

Now if we want to carry out, for instance, a t-test to test whether our sample mean is equal to some theoretical value, we may consider the function

```
t.test(x, y = NULL, alternative = c("two.sided", "less", "greater"), mu = 0,
paired = FALSE, var.equal = FALSE, conf.level = 0.95, ...)
```

Example.

```
> t.test(eruptions, mu=3.6, var.equal=FALSE)
                    One Sample t-test

data:  eruptions
t = -1.6215, df = 271, p-value = 0.1061
alternative hypothesis: true mean is not equal to 3.6
95 percent confidence interval:
 3.351534 3.624032
sample estimates:
mean of x
 3.487783
```

Hypothesis testing in R

We can use the F test to test for equality in the variances of two samples, provided that the two samples are from normal distributions.
The general syntax is

```
var.test(x, y, ratio = 1, alternative = c("two.sided", "less", "greater"), conf.level = 0.95, ...)
```

Example. Import the data of the two external file «COMOALL_PPP.TRP». Consider the CORR_U component of the data frames from «COMOALL.TRP» and «COMOALL_PPP.TRP».
Perform a F-test between these two samples!

Hypothesis testing in R

```
>  zwd_dani <-
read.table('C:/Users/Alessandro/Documents/R_Work/DAN1SIGMAALL.TR
P', header=TRUE)
> var.test(zwd$CORR_U, zwd_dani$CORR_U)

        F test to compare two variances

data:  zwd$CORR_U and zwd_dani$CORR_U
F = 1.0146, num df = 174, denom df = 170, p-value = 0.9247
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.7515277 1.3691130
sample estimates:
ratio of variances
          1.014629
```

POLITECNICO DI MILANO

POLITECNICO DI MILANO 150°

GEOLaB Geomatics and Earth Observation lab

THANK YOU FOR YOUR ATTENTION!