

Programmazione statistica in R: corso teorico-pratico

Parte 1

Damiano G. Preatoni

Unità di Analisi e Gestione delle Risorse Ambientali – *Guido Tosi Research Group*
Dipartimento di Scienze Teoriche e Applicate, Università degli Studi dell'Insubria
prea@uninsubria.it

7-9/7/2021



Contenuti

- 1 Presentazione dell'ambiente R
- 2 Il sistema R: come si lavora, cosa si fa
- 3 Principi di programmazione
- 4 Il linguaggio R
- 5 Usare R



Sommario

- 1 **Presentazione dell'ambiente R**
- 2 Il sistema R: come si lavora, cosa si fa
- 3 Principi di programmazione
- 4 Il linguaggio R
- 5 Usare R

Dalla teoria alla pratica...

L'analisi dei dati è un elemento imprescindibile.¹

Quali strumenti?

- riproducibilità
 - flessibilità
 - falsificabilità
-
- 1970-1980: pochi pacchetti generici (SAS, SPSS, minitab, S-PLUS)
 - ▶ uso specialistico per le procedure più comune, programmi *ad hoc* in generale
 - 1980-1990: pacchetti proprietari e specializzati (SYSTAT, stata, Origin, ecc.)
 - ▶ tendenza a legarsi ad uno specifico *software*
 - 1990-oggi: banalizzazione e *canned software*
 - ▶ lo *spreadsheet* è l'unico strumento di "calcolo"²

¹Science, 340(6134): 814-815 doi:10.1126/science.1231535

²http://www.scientific-computing.com/features/feature.php?feature_id=218



Perché R?

- È libero e gratuito³
- È uno standard in ambiente scientifico (e non solo!)
- Riesce a gestire lavori complessi in modo semplice (Big Data!)
- Moduli specializzati per *qualunque analisi*⁴
- Elaborati grafici di qualità tipografica (*press-ready*)

³Ihaka R., Gentleman R. 1996. R: a language for data analysis and graphics. *Journal of Computational and Graphical Statistics* 5: 299-314.

⁴marzo 2009: 1723 pacchetti differenti; ottobre 2013: 4953, luglio 2014: 5721, novembre 2015: 7542, ottobre 2016: 9369, aprile 2017: 10374, novembre 2020: 16568, luglio 2021: 17812



Perché R?

- È libero e gratuito³
- È uno standard in ambiente scientifico (e non solo!)
- Riesce a gestire lavori complessi in modo semplice (Big Data!)
- Moduli specializzati per *qualunque analisi*⁴
- Elaborati grafici di qualità tipografica (*press-ready*)
- Costringe a ragionare (*no wizards, no monkey-typing*)
- Dà dipendenza

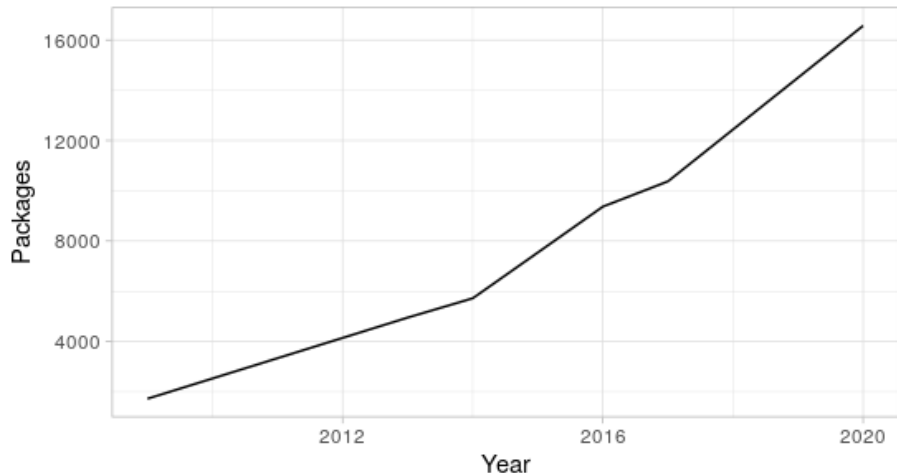


³Ihaka R., Gentleman R. 1996. R: a language for data analysis and graphics. *Journal of Computational and Graphical Statistics* 5: 299-314.

⁴marzo 2009: 1723 pacchetti differenti; ottobre 2013: 4953, luglio 2014: 5721, novembre 2015: 7542, ottobre 2016: 9369, aprile 2017: 10374, novembre 2020: 16568, luglio 2021: 17812



Perché R?

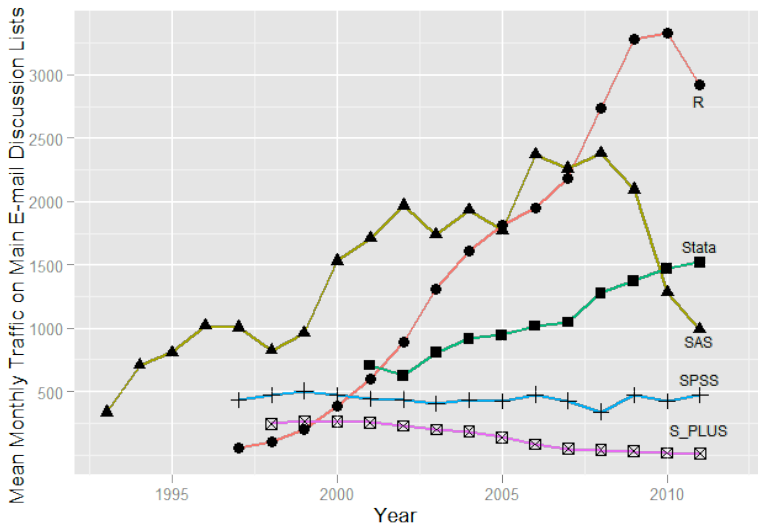


³Ihaka R., Gentleman R. 1996. R: a language for data analysis and graphics. *Journal of Computational and Graphical Statistics* 5: 299-314.

⁴marzo 2009: 1723 pacchetti differenti; ottobre 2013: 4953, luglio 2014: 5721, novembre 2015: 7542, ottobre 2016: 9369, aprile 2017: 10374, novembre 2020: 16568, luglio 2021: 17812



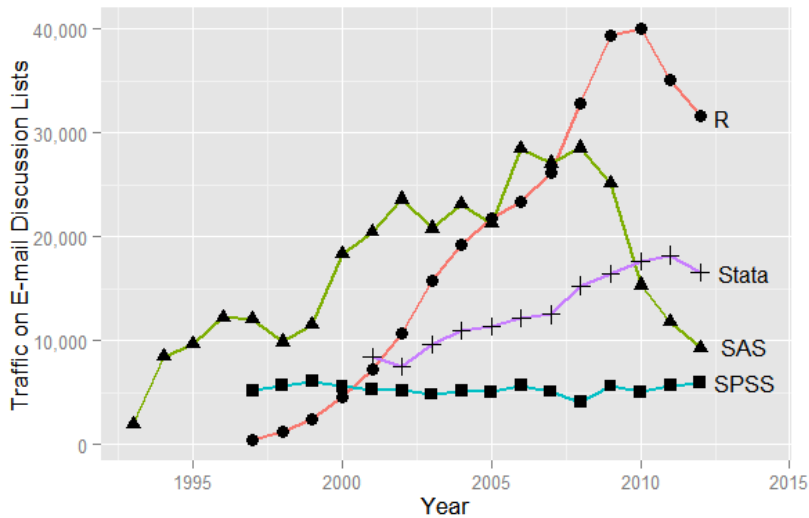
Perché R?



⁵<http://r4stats.com/articles/popularity/>



Perché R?

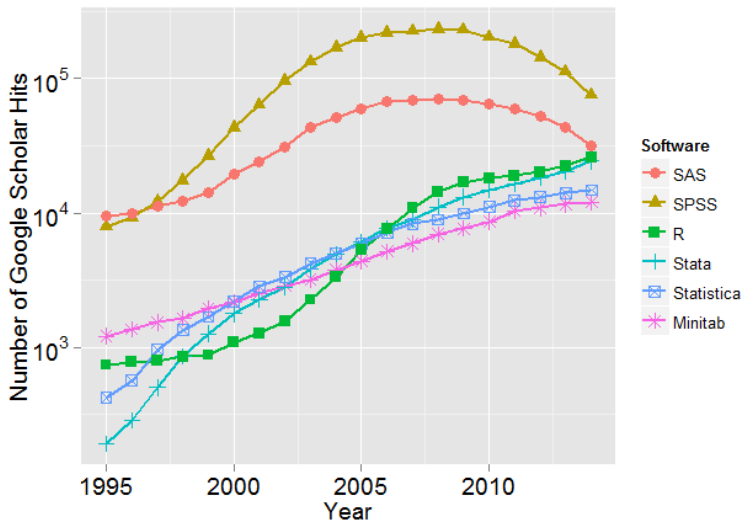


5

⁵<http://r4stats.com/articles/popularity/>



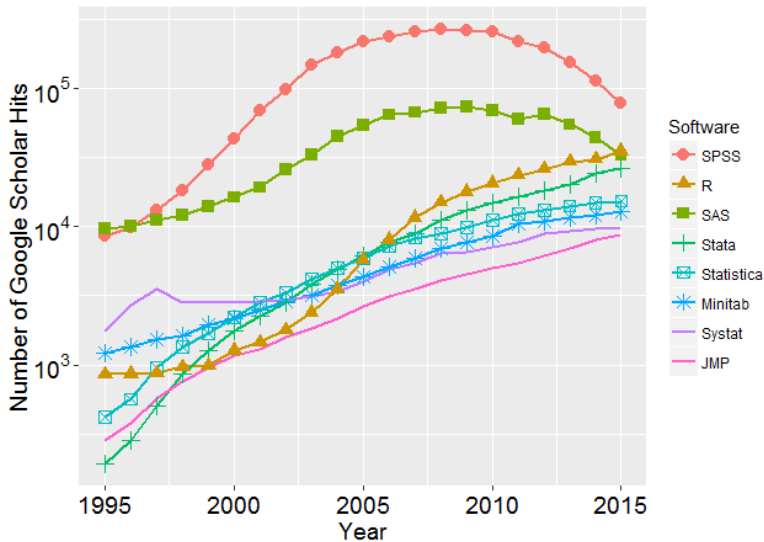
Perché R?



5

⁵<http://r4stats.com/articles/popularity/>

Perché R?

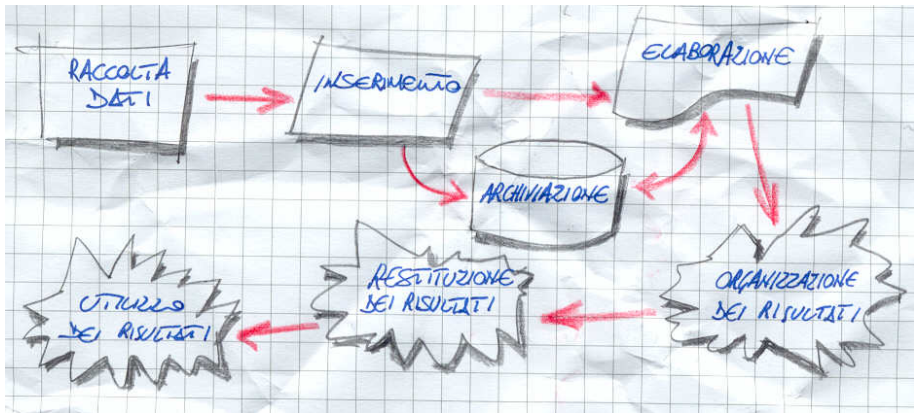


5

⁵<http://r4stats.com/articles/popularity/>



Il flusso di lavoro



Il flusso di lavoro: *step by step* o *one shot*?

R

- un'analisi comprende una serie di passaggi
- ogni passaggio produce dei risultati intermedi, che possono essere utilizzati nei passaggi successivi
- non vengono mostrati a video (“*stampati*”) risultati, se non a richiesta

Altri programmi

- un'analisi è eseguita in un unico passaggio
- il risultato non è “riutilizzabile”
- il risultato è un *output* lungo e prolisso



Altri software

Vantaggi

- velocità nell'eseguire analisi standard
- interattività
- visibilità del singolo dato



Vantaggi apparenti

- velocità nell'eseguire analisi standard
 - ▶ e con le analisi non standard? quali e quante analisi sono previste?
- interattività
 - ▶ interattività significa interagire con i *dati*, non con i comandi del programma
- visibilità del singolo dato
 - ▶ *big data*: a che serve “vedere” una matrice di 7000 righe e 200 colonne?



Altri software

Vantaggi

- velocità nell'eseguire analisi standard
- interattività
- visibilità del singolo dato

... ma a che prezzo?

- impossibile effettuare analisi “non standard”
- interattività significa interagire con i dati, non con i comandi
- ha senso poter esaminare una matrice di 7000 righe e 200 colonne?
- scarsa estensibilità
- scarsa qualità e inflessibilità dell'output
- impossibile (o quasi) da automatizzare



Perché R è meglio!

Vantaggi

- suddivisione del problema
- possibilità di salvare e riutilizzare gli intermedi
- *single point of truth*



Perché R è meglio!

Vantaggi

- suddivisione del problema
- possibilità di salvare e riutilizzare gli intermedi
- *single point of truth*

... e completa libertà di azione!

- R è un *linguaggio*, quindi è possibile programmare *qualsunque cosa*
- la CLI^a è più produttiva di una GUI^b, e garantisce ripetibilità
- ... se “complicato” significa “sapere cosa si sta facendo”, R è semplice, SPSS no!

^aCommand Line Interface

^bGraphical User Interface



Sommario

- 1 Presentazione dell'ambiente R
- 2 Il sistema R: come si lavora, cosa si fa**
- 3 Principi di programmazione
- 4 Il linguaggio R
- 5 Usare R

L'ambiente di lavoro

```
prea@mus:~$ R
R version 2.10.1 (2016-05-03) -- "Bipedally Educational"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: i686-pc-linux-gnu (32-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.
'Principi di...'

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.

prea@mus:~$ R
[Previously saved workspace restored]

L'ambiente di lavoro

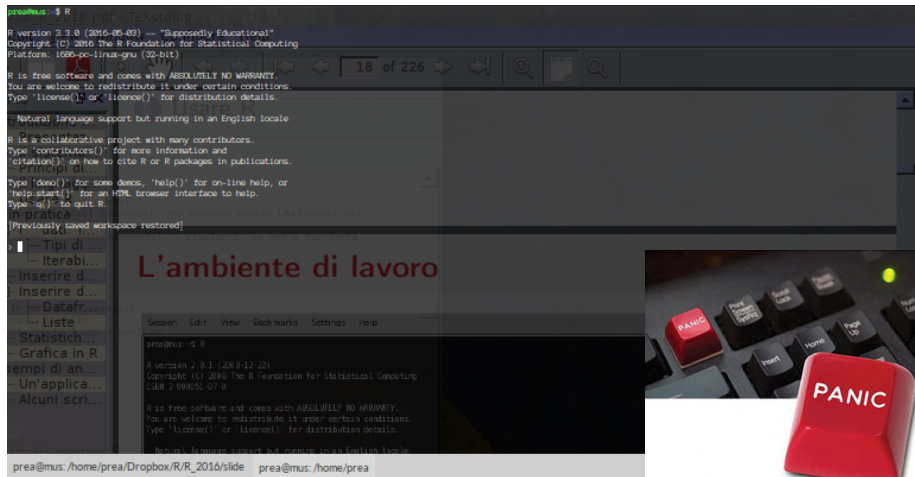
Session Edit View Bookmarks Settings Help
prea@mus:~$ R
R version 2.10.1 (2016-12-22)
Copyright (C) 2016 The R Foundation for Statistical Computing
ISBN 2-599-051-07-9

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
```



L'ambiente di lavoro



The image shows a terminal window with the R console output and a keyboard with a red 'PANIC' key. The terminal window displays the R version 2.10.0 (2016-05-03) and the R logo. The R logo is a stylized 'R' with a white background and a red outline. The terminal output includes the R version, copyright information, and a list of help topics. The keyboard has a red 'PANIC' key and a green light on the right side.

```
prea@mus:~$ R
R version 2.10.0 (2016-05-03) -- "Supposedly Educational"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: i686-pc-linux-gnu (32-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.
'Principi di...'
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.

prea@mus:~$ R
[Previously saved workspace restored]

prea@mus:~$ R
[Previously saved workspace restored]

prea@mus:~$ R
[Previously saved workspace restored]
```

Tipi di
Iterabi...
Inserire d...
Inserire d...
Datafr...
Liste
Statistich...
Grafica in R
empi di an...
Un'applica...
Alcuni scri...


Session Edit View Bookmarks Settings Help

```
prea@mus:~$ R
R version 2.10.1 (2016-12-22)
Copyright (C) 2016 The R Foundation for Statistical Computing
ISBN 2-599-051-07-9

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
```

prea@mus: /home/prea/Dropbox/R/R_2016/slide prea@mus: /home/prea



A close-up of a keyboard with a red 'PANIC' key and a green light. The key is a standard keyboard key with a red top and white text. The green light is a small LED light on the right side of the keyboard.

L'ambiente di lavoro

La *shell* R non è sicuramente la “postazione di lavoro” più comoda...

...ma l'apparenza inganna!

Dato che un comando viene digitato da tastiera, un insieme di comandi può essere contenuto in un file di testo: normalmente si utilizza un *text editor* come *front end*⁶

Editor

- Costruzione di uno *script*
- Sviluppo di funzioni
- Riutilizzo di codice precedentemente scritto

Console

- Verifica di un singolo comando
- Esame interattivo dei dati

⁶o meglio, di un *editor* integrato in un ambiente dedicato a R



Apt-get it!⁷

I software indicati sono solo alcune delle numerose possibilità!

*NIX

KDE, IDE RKward

emacs ESS (emacs speaks statistics)

OSX

R.app

Komodo Editor

Windows

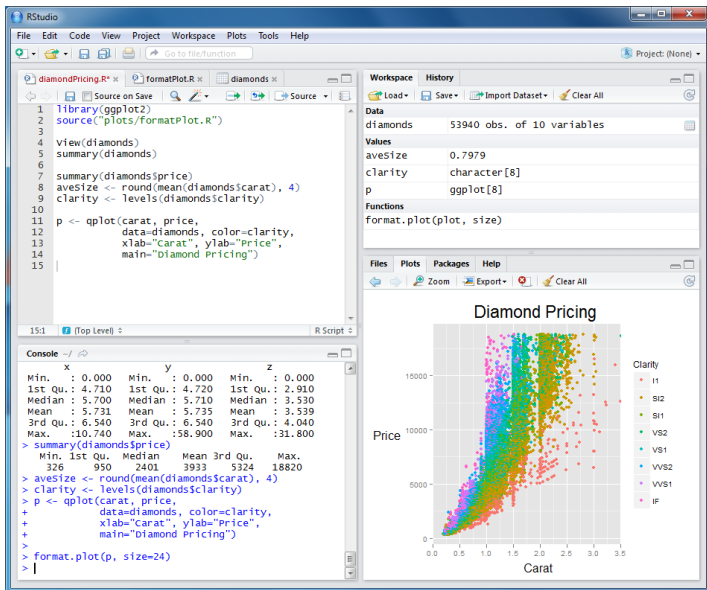
Editor: Tinn-R

IDE: RKward

Multipiattaforma!!

RStudio (<https://www.rstudio.com/products/rstudio/>)

Benvenuti a casa! ...Windows



Benvenuti a casa! ...Windows

The screenshot displays the RStudio environment. The main window is titled 'diamondPricing.R' and contains the following R code:

```
1 library(ggplot2)
2 source("formatPlot.R")
3
4 view
5 summ
6
7 summ
8 aveS
9 clar
10
11 p <-
12
13
14
15
```

A green callout box with the text 'Text Editor' and 'Creazione script, invio comandi alla console' is overlaid on the script editor.

The console window shows the following output:

```
Min. x : 0.000 Min. y : 0.000 Min. z : 0.000
1st Qu.: 4.710 1st Qu.: 4.720 1st Qu.: 2.910
Median: 5.700 Median: 5.710 Median: 3.530
Mean : 5.731 Mean : 5.735 Mean : 3.539
3rd Qu.: 6.540 3rd Qu.: 6.540 3rd Qu.: 4.040
Max. :10.740 Max. :58.900 Max. :31.800
> summary(diamonds$price)
Min. 1st Qu. Median Mean 3rd Qu. Max.
326 950 2401 3933 5324 18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+ data=diamonds, color=clarity,
+ xlab="Carat", ylab="Price",
+ main="Diamond Pricing")
> format.plot(p, size=24)
|
```

The 'Workspace' pane shows the loaded data:

Variable	Value
diamonds	53940 obs. of 10 variables
aveSize	0.7979
clarity	character [8]
p	ggplot [8]

The 'Plots' pane displays a scatter plot titled 'Diamond Pricing'. The x-axis is 'Carat' (ranging from 0.0 to 3.5) and the y-axis is 'Price' (ranging from 0 to 15000). The plot shows a positive correlation between carat weight and price, with points colored by clarity. A legend on the right lists the clarity levels: I1, SI2, SI1, VS2, VS1, VVS2, VVS1, and IF.



Benvenuti a casa! ...Windows

The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains R code for loading data, summarizing it, and creating a faceted plot.
- Workspace:** Shows the loaded 'diamonds' dataset with 53940 observations and 10 variables.
- Console:** Shows the execution of the R code, with a green callout box highlighting the direct interaction with the console.
- Plots Panel:** Displays a scatter plot titled 'Diamond Pricing' showing Price on the y-axis (0 to 15000) and Carat on the x-axis (0.0 to 3.5). The points are colored by Clarity, with a legend on the right showing categories: I1, SI2, SI1, VS2, VS1, VVS2, VVS1, and IF.

```
1 library(ggplot2)
2 source("plots/formatPlot.R")
3
4 view(diamonds)
5 summary(diamonds)
6
7 summary(diamonds$price)
8 aveSize <- round(mean(diamonds$carat), 4)
9 clarity <- levels(diamonds$clarity)
10
11 p <- qplot(carat, price,
12           data=diamonds, color=clarity,
13           xlab="Carat", ylab="Price",
14           main="Diamond Pricing")
15
```

Console:

```
Min. x: 0.000 Min. y: 0.000 Min. z: 0.000
1st Qu.:
Median:
Mean:
3rd Qu.:
Max. : I1
> summary(
  Min. 1s
  326
> aveSize
> clarity
> p <- qplot
+ data=diamonds, color=clarity,
+ xlab="Carat", ylab="Price",
+ main="Diamond Pricing")
> format.plot(p, size=24)
|
```

Workspace:

Values	
aveSize	0.7979
clarity	character [8]
p	ggplot [8]

Plots Panel:

Diamond Pricing

Price

Carat

Clarity

- I1
- SI2
- SI1
- VS2
- VS1
- VVS2
- VVS1
- IF



Benvenuti a casa! ...Windows

The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains R code for loading data, summarizing it, and creating a faceted plot.
- Console:** Shows the execution output, including summary statistics for 'diamonds\$price' and the execution of the plotting commands.
- Workspace:** Lists the loaded data object 'diamonds' and the created plot object 'p'.
- Plots Panel:** Displays a faceted scatter plot titled 'Diamond Pricing' with 'Price' on the y-axis and 'Carat' on the x-axis, faceted by 'Clarity'.

```
1 library(ggplot2)
2 source("plots/formatPlot.R")
3
4 view(diamonds)
5 summary(diamonds)
6
7 summary(diamonds$price)
8 aveSize <- round(mean(diamonds$carat), 4)
9 clarity <- levels(diamonds$clarity)
10
11 p <- qplot(carat, price,
12           data=diamonds, color=clarity,
13           xlab="Carat", ylab="Price",
14           main="Diamond Pricing")
15
```

```
Min. x: 0.000 Min. y: 0.000 Min. z: 0.000
1st Qu.: 4.710 1st Qu.: 4.720 1st Qu.: 2.910
Median: 5.700 Median: 5.710 Median: 3.530
Mean: 5.731 Mean: 5.735 Mean: 3.539
3rd Qu.: 6.540 3rd Qu.: 6.540 3rd Qu.: 4.040
Max.: 10.740 Max.: 58.900 Max.: 31.800
> summary(diamonds$price)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  326     950    2401    3933    5324   18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+           data=diamonds, color=clarity,
+           xlab="Carat", ylab="Price",
+           main="Diamond Pricing")
> format.plot(p, size=24)
|
```

Workspace	History
diamonds	53940 ob
Values	
aveSize	0.7979
clarity	characte
p	ggplot[8
Functions	
format.plot	plot, size

Diamond Pricing

Price

Carat

Clarity

- I1
- SI2
- SI1
- VS2
- VS1
- VVS2
- VVS1
- IF

Workspace content

Stato e valori delle variabili



Benvenuti a casa! ...Windows

The screenshot displays the RStudio interface. The script editor on the left contains the following R code:

```
1 library(ggplot2)
2 source("plots/formatPlot.R")
3
4 view(diamonds)
5 summary(diamonds)
6
7 summary(diamonds$price)
8 aveSize <- round(mean(diamonds$carat), 4)
9 clarity <- levels(diamonds$clarity)
10
11 p <- qplot(carat, price,
12            data=diamonds, color=clarity,
13            xlab="Carat", ylab="Price",
14            main="Diamond Pricing")
15
```

The console on the bottom left shows the output of the code, including summary statistics for the diamonds data and the execution of the plotting functions.

The Environment pane on the right shows the loaded data objects:

Data	Value
diamonds	53940 obs. of 10 variables
aveSize	0.7979
clarity	character [8]
p	ggplot [8]

The Plots pane on the bottom right displays a scatter plot titled "Diamond Pricing". The x-axis is labeled "Carat" and ranges from 0.0 to 3.5. The y-axis is labeled "Price" and ranges from 0 to 15000. The plot shows a positive correlation between carat weight and price, with points colored by clarity. A green callout box is overlaid on the plot with the text:

Plots
Grafici, con possibilità di salvataggio e storicizzazione



Benvenuti a casa! ...Apple OSX

The screenshot displays the RStudio interface on an Apple OSX system. The main editor window contains the following R code:

```
1 # Home Prices In Mid-West
2
3 homes <- read.csv("homePriceData.csv")
4 View(homes)
5 names(homes)
6 summary(homes$price)
7 summary(homes$age)
8
9 states <- levels(homes$state)
10 avePrice <- round(mean(homes$price),2)
11 aveAge <- round(mean(homes$age), 0)
12
13
14
```

The console window shows the execution of these commands and their output:

```
> homes <- read.csv("homePriceData.csv")
> View(homes)
> names(homes)
[1] "city"      "state"     "price"
[4] "age"       "condition" "remodeling"
[7] "neighborhood"
> summary(homes$price)
  Min.   75290
> summar
  Min.   0.00
> states
subset.matrix {base}
> avePri
substitute {base}
> aveAge
> old <- sub
```

The Workspace pane shows the 'homes' data frame with 580 observations and 7 variables. The Values pane displays summary statistics: aveAge (36), avePrice (416405.55), and states (character[11]).

The Help pane displays the documentation for the `subset` function:

Subsetting Vectors, Matrices and Data Frames

Description

Return subsets of vectors, matrices or data frames which meet conditions.

Usage

```
subset(x, ...)
```

Return subsets of vectors, matrices or data frames which meet conditions.

Arguments

- `x`: A vector, matrix or data frame.
- `subset`: A logical vector or expression.
- `select`: A character vector of column names.
- `drop`: A logical scalar.

`FALSE, ...`

`subset(x, subset, select, drop = FALSE, ...)`



Benvenuti a casa! ...Apple OSX

The screenshot shows the RStudio environment with the following components:

- Editor:** Contains R code for loading 'homePriceData.csv', viewing the data, and calculating summary statistics like 'avePrice' and 'aveAge'.
- Console:** Shows the execution of the code, with the output of the 'summary' function partially visible. A green callout box highlights the text: "Console 'comoda' Completamento del testo, help".
- Data Table:** Shows the structure of the 'homes' dataset: 580 observations of 7 variables. Summary statistics include 'aveAge' (36), 'avePrice' (416405.55), and 'states' (character[11]).
- R Documentation:** Opened to the 'subset' function page, showing the title 'Subsetting Vectors, Matrices and Data Frames' and a description: 'Return subsets of vectors, matrices or data frames which meet conditions.'



Benvenuti a casa! ...Apple OSX

The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains R code for loading and summarizing the 'homes' dataset.
- Console:** Shows the execution of the code, including the output of `summary(homes$price)` and `summary(homes$age)`.
- Workspace:** Shows the loaded 'homes' dataset with 580 observations and 7 variables.
- Help Browser:** Displays the documentation for the `subset` function, including its description and usage.

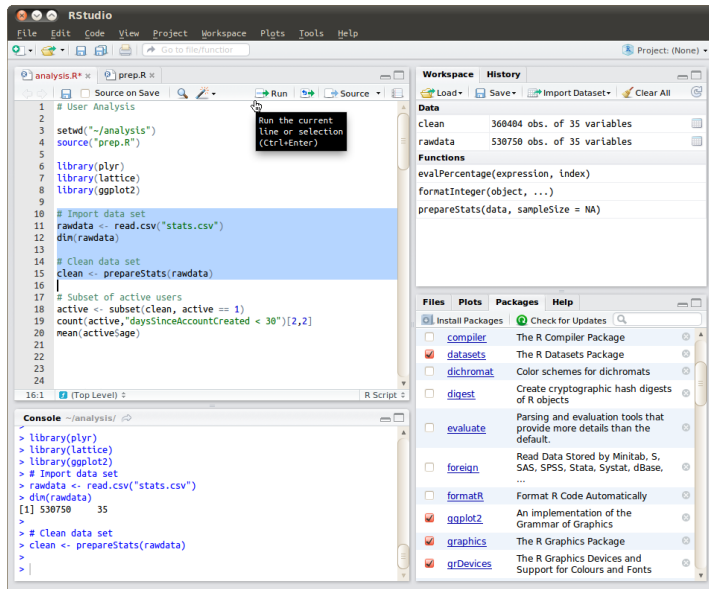
```
1 # Home Prices In Mid-West
2
3 homes <- read.csv("homePriceData.csv")
4 View(homes)
5 names(homes)
6 summary(homes$price)
7 summary(homes$age)
8
9 states <- levels(homes$state)
10 avePrice <- round(mean(homes$price),2)
11 aveAge <- round(mean(homes$age), 0)
12
13
14
```

```
> homes <- read.csv("homePriceData.csv")
> View(homes)
> names(homes)
[1] "city"      "state"     "price"
[4] "age"       "condition" "remodeling"
[7] "neighborhood"
> summary(homes$price)
  Min.   75290
 1st Qu. 100000
  Median 120000
 3rd Qu. 140000
  Max.   160000
> summary(homes$age)
  Min.   0.00
 1st Qu. 0.00
  Median 0.00
 3rd Qu. 0.00
  Max.   0.00
> states
[1] "CA" "TX" "FL" "NY" "IL" "WA" "AZ" "OR" "NV" "ID" "MT" "WY" "UT" "CO" "NM" "OK" "KS" "NE" "ND" "SD" "IA" "MO" "WI" "MI" "IN" "OH" "PA" "NY" "NJ" "DE" "MD" "VA" "NC" "SC" "GA" "AL" "LA" "MS" "AR" "OK" "TX" "LA" "MS" "AL" "GA" "FL" "CA"
> avePrice
[1] 120000
> aveAge
[1] 0
> old <- sub
```

Help browser
Pagine di manuale sempre a portata di mano



Benvenuti a casa! ...Ubuntu (GNOME)



The screenshot displays the RStudio environment with the following components:

- Source Editor:** Contains an R script with the following code:

```
1 # User Analysis
2
3 setwd("~/analysis")
4 source("prep.R")
5
6 library(plyr)
7 library(lattice)
8 library(ggplot2)
9
10 # Import data set
11 rawdata <- read.csv("stats.csv")
12 dlm(rawdata)
13
14 # Clean data set
15 clean <- prepareStats(rawdata)
16
17 # Subset of active users
18 active <- subset(clean, active == 1)
19 count(active, "daysSinceAccountCreated < 30")[2,2]
20 mean(active$age)
21
22
23
24
```
- Run Button:** A tooltip indicates: "Run the current line or selection (Ctrl+Enter)".
- Console:** Shows the execution output:

```
> library(plyr)
> library(lattice)
> library(ggplot2)
> # Import data set
> rawdata <- read.csv("stats.csv")
> dlm(rawdata)
[1] 530750 35
>
> # Clean data set
> clean <- prepareStats(rawdata)
>
> |
```
- Workspace:** Lists data objects: 'clean' (360404 obs. of 35 variables) and 'rawdata' (530750 obs. of 35 variables). It also lists functions: 'evalPercentage', 'formatInteger', and 'prepareStats'.
- Files:** A list of installed and available packages, including 'compiler', 'datasets', 'dichromat', 'digest', 'evaluate', 'foreign', 'formatR', 'ggplot2', 'graphics', and 'grDevices'.



Benvenuti a casa! ...Ubuntu (GNOME)

The screenshot shows the RStudio interface with a callout box highlighting advanced editor features. The callout box contains the following text:

- Editor avanzato
- Syntax highlighting,
- "point and shoot" (CTRL+Enter)

The RStudio interface includes the following elements:

- Source Editor:** Shows R code for data analysis, including library loading, data import, and cleaning.
- Console:** Displays the output of the R code, showing the number of observations and variables for the raw and clean data sets.
- Workspace:** Lists the data objects (clean, rawdata) and functions (evalPercentage, formatInteger, prepareStats) currently loaded in the environment.
- Files:** Lists installed and available R packages, such as compiler, datasets, dichromat, digest, evaluate, foreign, formatR, ggplot2, graphics, and grDevices.

```
1 # User Analysis
2
3 setwd("...")
4 source("...")
5
6 library(plyr)
7 library(lattice)
8 library(ggplot2)
9
10 # Import data set
11 rawdata <- read.csv("stats.csv")
12 dlm(rawdata)
13
14 # Clean data set
15 clean <- prepareStats(rawdata)
16
17 # Subsetting
18 active <- filter(clean, ... )
19 count(active, ... )
20 mean(active$page)
21
22
23
24
```

```
> library(plyr)
> library(lattice)
> library(ggplot2)
> # Import data set
> rawdata <- read.csv("stats.csv")
> dlm(rawdata)
[1] 530750 35
>
> # Clean data set
> clean <- prepareStats(rawdata)
>
```



Benvenuti a casa! ...Ubuntu (GNOME)

The screenshot displays the RStudio environment. The main editor window shows R code for data analysis. A tooltip over the 'Run' button indicates that clicking it will execute the current line or selection (Ctrl+Enter). The console window at the bottom shows the output of the code, including the loading of libraries (plyr, lattice, ggplot2), the import of a CSV file, and the calculation of the mean age of active users. The right-hand pane is divided into 'Workspace' and 'History' sections, and a 'Packages' section at the bottom. The 'Packages' section lists installed and available packages, with a green callout box highlighting the 'Packages' section and the text 'Stato e installazione di packages aggiuntivi'.

```
1 # User Analysis
2
3 setwd("~/analysis")
4 source("~/prep.R")
5
6 library(plyr)
7 library(lattice)
8 library(ggplot2)
9
10 # Import data set
11 rawdata <- read.csv("stats.csv")
12 dlm(rawdata)
13
14 # Clean data set
15 clean <- prepareStats(rawdata)
16
17 # Subset of active users
18 active <- subset(clean, active == 1)
19 count(active, "daysSinceAccountCreated < 30")[2,2]
20 mean(active$age)
21
22
23
24
```

Run the current line or selection (Ctrl+Enter)

Workspace History

Data

clean	368484 obs. of 35 variables
rawdata	538750 obs. of 35 variables

Functions

- evalPercentage(expression, index)
- formatInteger(object, ...)
- prepareStats(data, sampleSize = NA)

Files Plots Packages Help

Install Packages Check for Updates

- compiler The R Compiler Package
- datasets The R Datasets Package
- dichromat Color schemes for dichromats
- digest
- evaluate
- foreign
- formatR
- ggplot2 The implementation of the Grammar of Graphics
- graphics The R Graphics Package
- grDevices The R Graphics Devices and Support for Colours and Fonts

Console ~/analysis/

```
> library(plyr)
> library(lattice)
> library(ggplot2)
> # Import data set
> rawdata <- read.csv("stats.csv")
> dlm(rawdata)
[1] 538750 35
>
> # Clean data set
> clean <- prepareStats(rawdata)
>
>
```

16:1 (Top Level) R Script

Packages
Stato e installazione di packages aggiuntivi



Benvenuti a casa! ... RStudio server, Chrome browser

gis.dipbsf.uninsubria.it:8787

File Edit Code View Plots Session Build Debug Tools Help

demoniche_test.R* demoniche_examplscript.R*

```
1 library(demoniche)
2 library(rgdsl)
3
4 # notes: all in UTF32 32632
5
6 # Import land cover raster as "niche" data.
7 # Show raster has to be converted into a dataframe with the fields: patch_
8
9
10 # gray square id/distribution data (point)
11 initial_distribution <- readRDS(paste0("/home/gis/sciolatto_umbria/BASIS", layer="distribution_201210_clean"))
12
13
14 xs <- expand.grid(seq(1, 2, 0.5), seq(1, 2, 0.5)) # make coordinates
15 dist_latlong <- round(as.matrix(dist(xs)), 1)
16 net_id_index <- sort(unique(as.numeric(dist_latlong)))[[2]] # find coordinates
17
18 seeds_per_population_new <- rep(0, nrow(xs))
19 seeds_per_population <- sample(c(0, 0, 10000000), length(seeds_per_population_new), replace = TRUE)
20
21 dist_populations <- apply(xs, 1, function(eachPoint) splotains(as.matrix(xs), eachPoint, longLat=TRUE))
22
23 dispersal_constants <- c(0.7, 0.7, 0.1, 1000)
24
25 dispersal_probabilities <- dispersal_constants[1] * exp(-dist_populations * dispersal_constants[3]) / dispersal_constants[2]
```

Console

```
Warning in file(file, this[,append, "a", "w"]):
cannot open file '/home/prea/R/x86_64-pc-linux-gnu-library/3.1/manipulate/DESCRIPTION': Permission denied
Error in file(file, this[,append, "a", "w"]):
cannot open the connection
ERROR: installing package DESCRIPTION failed for package 'manipulate'
* removing '/home/prea/R/x86_64-pc-linux-gnu-library/3.1/manipulate'

29 Nov 2015 10:20:04 [rsession-prea] ERROR system error 131 (state not recoverable) [description=error installing package: * installing 'source'
package 'manipulate' ...
not cannot move '/home/prea/R/x86_64-pc-linux-gnu-library/3.1/manipulate' to '/home/prea/R/x86_64-pc-linux-gnu-library/3.1/LOCK-manipulate/mani
pulate': Permission denied
Warning in file.copy(f, toDir, TRUE) :
broken copying from '/home/prea/R/x86_64-pc-linux-gnu-library/3.1/manipulate/NAMESPACE' to '/home/prea/R/x86_64-pc-linux-gnu-library/3.1/manipulate/DESCRIPTION': Permission denied
Error in file(file, this[,append, "a", "w"]):
cannot open file '/home/prea/R/x86_64-pc-linux-gnu-library/3.1/manipulate/DESCRIPTION': Permission denied
Error in file(file, this[,append, "a", "w"]):
cannot open the connection
ERROR: installing package DESCRIPTION failed for package 'manipulate'
* removing '/home/prea/R/x86_64-pc-linux-gnu-library/3.1/manipulate'
] OCCURRED AT: core::error session::module_context::installPackage(const std::string&, const std::string&) /home/ubuntu/rstudio/src/cpp/session
n/SessionModuleContext.cpp:1079; LOGGED FROM: SEXPrec::session::modules::build::unnamed::r::installPackage(SEXPprec*, SEXPrec*) /home/ubuntu/rs
tudio/src/cpp/session/Modules/Build/SessionModule.cpp:1542
>
```

Benvenuti a casa! ... RStudio server, Chrome browser

Server edition!
Utilizzo condiviso,
tramite browser
web... anche da
tablet!

Environment History Files

Name	Size	Modified
.RData	87 B	Aug 5, 2014, 12:04 PM
.Rhistory	26.9 KB	Aug 5, 2014, 12:04 PM
.Rprofile	232 B	Apr 6, 2011, 1:02 PM
_legend.png	3.1 KB	Aug 5, 2014, 1:17 PM
audio		
Callosciurus.erythraeus.science.png	14 KB	Aug 5, 2014, 3:31 PM
Callosciurus_legend.png	2.6 KB	Aug 5, 2014, 3:31 PM
CC_nodispersal		
Desktop		
Documents		
Downloads		

Plots Packages Help Viewer

Name	Description	Version
abind	Combine Multidimensional Arrays	1.4-3
acepack	ace() and avas() for selecting regression transformations	1.3-3.3
animation	A Gallery of Animations in Statistics and Utilities to Create Animations	2.4
aqp	Algorithms for Quantitative Pedology	1.8-6
arm	Data Analysis Using Regression and Multilevel/Hierarchical Models	1.8-6
assertthat	Easy pre and post assertions.	0.1
betareg	Beta Regression	3.0-5
BH	Boost C++ Header Files	1.58.0-1
brew	Templating Framework for Report Generation	1.0-6
chron	Chronological Objects which can Handle Dates and Times	2.3-47

```
Warning in file(file, this[,append, "a", "w"]):  
cannot open file '/home/prea/R/x86_64-pc-linux-gnu-library/3.1/manipulate/DESCRIPTION': Permission denied  
Error in file(file, this[,append, "a", "w"]):  
cannot open the connection  
ERROR: installing package DESCRIPTION failed for package 'manipulate'  
* removing '/home/prea/R/x86_64-pc-linux-gnu-library/3.1/manipulate'  
  
29 Nov 2015 10:20:04 [session-prea] ERROR system error 131 (state not recoverable) [description=error installing package: * installing 'source'  
package 'manipulate' ...  
ret cannot move '/home/prea/R/x86_64-pc-linux-gnu-library/3.1/manipulate' to '/home/prea/R/x86_64-pc-linux-gnu-library/3.1/SHLOCK-manipulate/mani  
pulate': Permission denied  
Warning in file.copy(f, tofile, TRUE) :  
broken copying from '/home/prea/R/x86_64-pc-linux-gnu-library/3.1/manipulate/NAMESPACE': Permission denied  
Warning in file(file, this[,append, "a", "w"]):  
cannot open file '/home/prea/R/x86_64-pc-linux-gnu-library/3.1/manipulate/DESCRIPTION': Permission denied  
Error in file(file, this[,append, "a", "w"]):  
cannot open the connection  
ERROR: installing package DESCRIPTION failed for package 'manipulate'  
* removing '/home/prea/R/x86_64-pc-linux-gnu-library/3.1/manipulate'  
[1] OCCURRED AT: core::error session::module_context::installPackage(const std::string&, const std::string&) /home/ubuntu/rstudio/src/cpp/sessio  
n/SessionModuleContext.cpp:1079; LOGGED FROM: SEXPREC::session::modules::build::unnamed::r::r_installPackage(SEXPPREC*, SEXPREC*) /home/ubuntu/rs  
tudio/src/cpp/session/Modules/Build/SessionBuild.cpp:1942  
>
```

Benvenuti a casa! ... RStudio server, Chrome browser

```
1 library(demomiche)
2 library(rgdsl)
3
4 # notes: all in UTF32 32632
5
6 # Import land cover raster as "niche" data.
7 # Show raster has to be converted into a dataframe with the fields: patch_
8
9
10 # gray square | distribution data (point)
11 initial_distribution <- readRDS(paste0("~/home/gis/sciottolo_unbria/BASIS", layer="distribution_201210_clean"))
12
13
14 xs <- expand.grid(seq(1, 2, 0.5), seq(1, 2, 0.5)) # make coordinates
15 dist_latlong <- round(as.matrix(dist(xs)), 1)
16 net_id_index <- sort(unique(as.numeric(dist_latlong)))[[2]] # find coordinates
17
18 seeds_per_population_new <- rep(0, nrow(xs))
19 seeds_per_population <- sample(c(0, 8, 10000000), length(seeds_per_population_new), replace = TRUE)
20
21 dist_populations <- apply(xs, 1, function(eachPoint) splotains(as.matrix(xs), eachPoint, longLat=TRUE))
22
23 dispersal_constants <- c(0.7, 0.7, 0.1, 3000)
24
25 dispersal_probabilities <- dispersal_constants[1] * exp(-dist_populations * dispersal_constants[3]) / dispersal_constants[2]
```

Environment History Files

Name	Size	Modified
Home		
.RData	87 B	Aug 5, 2014, 12:04 PM
.Rhistory	26.9 KB	Aug 5, 2014, 12:04 PM
.Rprofile	232 B	Apr 6, 2011, 1:02 PM
_legend.png	3.1 KB	Aug 5, 2014, 1:17 PM
audio		
Callosciurus.erythraeus.science.png	14 KB	Aug 5, 2014, 3:31 PM
Callosciurus_legend.png	2.6 KB	Aug 5, 2014, 3:31 PM
CC_nodispersal		
Desktop		
Documents		
Downloads		

Plots Packages Help Viewer

Console

```
Warning in file(file, this[,append, "a", "w"]) :
cannot open file '/home/prea/R/x86_64-pc-linux-gnu-library/3.1/manipulate/DESCRIPTION': Permission denied
Error in file(file, this[,append, "a", "w"]) :
cannot open the connection
ERROR: installing package DESCRIPTION failed for package 'manipulate'
* removing '/home/prea/R/x86_64-pc-linux-gnu-library/3.1/manipulate'

29 Nov 2015 10:20:04 [session-prea] ERROR system error 131 (state not recoverable) [description=error installing package: * installing *so*
package 'manipulate' ...
we cannot move '/home/prea/R/x86_64-pc-linux-gnu-library/3.1/manipulate' to '/home/prea/R/x86_64-pc-linux-gnu-library/3.1/SHOULD-manipulate'
Permission denied
Warning in file.copy(f, to, recursive, TRUE) :
broken copying /NAMESPACE to /home/prea/R/x86_64-pc-linux-gnu-library/3.1/manipulate/NAMESPACE: Permission denied
Warning in file(file, this[,append, "a", "w"]) :
cannot open file '/home/prea/R/x86_64-pc-linux-gnu-library/3.1/manipulate/DESCRIPTION': Permission denied
Error in file(file, this[,append, "a", "w"]) :
cannot open the connection
ERROR: installing package DESCRIPTION failed for package 'manipulate'
* removing '/home/prea/R/x86_64-pc-linux-gnu-library/3.1/manipulate'
] OCCURRED AT: core::error session::module_context::installPackage(const std::string&, const std::string&) /home/ubuntu/rstudio/src/cpp/session/
n/SessionModuleContext.cpp:1079; LOGGED FROM: SEXPrec::session::modules::build::unnamed::installPackage(SEXPprec*, SEXPrec*) /home/ubuntu/rst
udio/src/cpp/session/Modules/Build/SessionModule.cpp:1542
>
```

**Per non perdersi...
"Cheat sheet" in PDF,
anche in italiano!**

Package	Description	Version
assertthat	Easy pre and post assertions.	0.1
betareg	Beta Regression	3.0-5
BH	Boost C++ Header Files	1.58.0-1
brew	Templating Framework for Report Generation	1.0-6
chron	Chronological Objects which can Handle Dates and Times	2.3-47

IDE e flusso di lavoro

The screenshot displays the RStudio interface with three main panels:

- Source Editor (Left):** Contains R code for loading data, summarizing it, and creating a faceted plot. The code is as follows:

```
1 library(ggplot2)
2 source("plots/formatPlot.R")
3
4 view(diamonds)
5 summary(diamonds)
6
7 summary(diamonds$price)
8 aveSize <- round(mean(diamonds$carat), 4)
9 clarity <- levels(diamonds$clarity)
10
11 p <- qplot(carat, price,
12           data=diamonds, color=clarity,
13           xlab="Carat", ylab="Price",
14           main="Diamond Pricing")
15
```
- Workspace (Top Right):** Shows the current environment with variables: `diamonds` (53940 objects), `aveSize` (0.7979), `clarity` (character vector), and `p` (ggplot object).
- Plots (Bottom Right):** Displays a scatter plot titled "Diamond Pricing". The x-axis is "Carat" (0.0 to 3.5) and the y-axis is "Price" (0 to 15000). Points are colored by "Clarity" (I1, SI2, SI1, VS2, VS1, VVS2, VVS1, IF).

Two callout boxes provide context:

- 1: Console** (bottom left): *Sperimentazione comandi, proof-of-concept*
- 2: Workspace content** (top right): *Debugging, ispezione variabili*



IDE e flusso di lavoro

The screenshot displays the RStudio IDE with three main panels:

- Source Editor (Left):** Contains R code for loading data, summarizing it, and creating a faceted plot. The code includes `library(ggplot2)`, `source("plots/formatPlot.R")`, `view(diamonds)`, `summary(diamonds)`, `summary(diamonds$price)`, `aveSize <- round(mean(diamonds$carat), 4)`, `clarity <- levels(diamonds$clarity)`, and `p <- qplot(carat, price, data=diamonds, color=clarity, xlab="Carat", ylab="Price", main="Diamond Pricing")`.
- Workspace (Top Right):** Shows the loaded data frame `diamonds` with 53940 observations. It also lists variables `aveSize` (0.7979), `clarity` (character), and `p` (ggplot object).
- Plots (Bottom Right):** Displays a faceted scatter plot titled "Diamond Pricing" showing Price vs. Carat, faceted by Clarity.

1: Console

Sperimentazione
comandi,
proof-of-concept

2: Workspace content

Debugging, ispezione
variabili

3: Help

Documentazione (inclusa
ricerca), copia-e-incolla
di esempi



IDE e flusso di lavoro

The screenshot displays the RStudio IDE with four callout boxes highlighting key components:

- 4: Text Editor** (green box): "Creazione script: 'bella copia'" (Creation of script: 'beautiful copy'). The script editor shows R code for loading the 'ggplot2' library and plotting 'Price' vs 'Carat' for diamonds with 'size' mapped to 'aveSize'.
- 2: Workspace content** (purple box): "Debugging, ispezione variabili" (Debugging, inspection of variables). The Workspace pane shows the 'diamonds' data frame with 53940 observations and the 'format.plot()' function.
- 1: Console** (purple box): "Sperimentazione comandi, proof-of-concept" (Experimentation with commands, proof-of-concept). The Console shows the execution of 'summary()' and 'format.plot()' commands.
- 3: Help** (purple box): "Documentazione (inclusa ricerca), copia-e-incolla di esempi" (Documentation (including search), copy-and-paste of examples). The Plots pane shows a scatter plot titled 'Diamond Pricing' with 'Price' on the y-axis and 'Carat' on the x-axis.



Sì, ma... cosa devo scrivere?

R è un *linguaggio*.

Si tratta di apprenderne *grammatica* e *vocabolario*.

Occorre anche qualche idea riguardo ad alcuni concetti di *programmazione*...



Sì, ma... cosa devo scrivere?

R è un *linguaggio*.

Si tratta di apprenderne *grammatica* e *vocabolario*.

Occorre anche qualche idea riguardo ad alcuni concetti di *programmazione*...

Più si utilizza R, più si arricchisce il proprio “vocabolario”.



Sommario

- 1 Presentazione dell'ambiente R
- 2 Il sistema R: come si lavora, cosa si fa
- 3 Principi di programmazione**
- 4 Il linguaggio R
- 5 Usare R

What is a programming language?

Un linguaggio di programmazione è un *linguaggio*, cioè una notazione di comodo per esprimere *idee*.

Un linguaggio di programmazione è un metodo pratico per definire e organizzare concetti riguardanti l'elaborazione di informazioni.

Requisiti di un linguaggio di programmazione

- rendere pratica l'elaborazione
- rendere efficiente l'uso degli elaboratori



Programmazione: perché studiarla?

- per migliorare la propria capacità di comprensione
- per migliorare le proprie capacità di *problem solving*
- per utilizzare al meglio uno strumento

Programmare significa *risolvere problemi*.

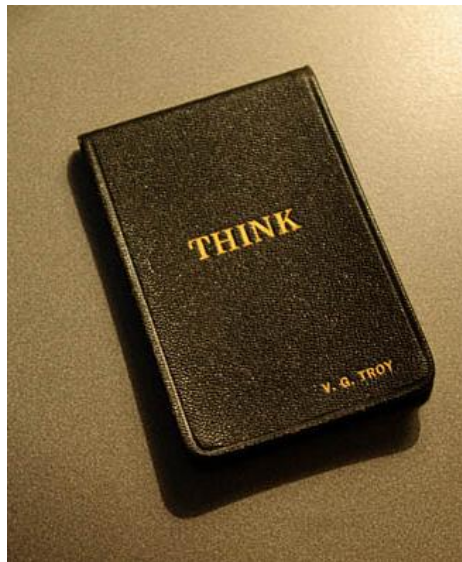


What is programming?

Programmare non è soltanto ordinare a un calcolatore di svolgere determinate operazioni. È un metodo per organizzare le idee riguardo a un dato fenomeno o processo.

Programmare è un metodo per pensare.

Qualunque linguaggio di programmazione offre un metodo⁸ per identificare concetti semplici e organizzarli in modo complesso, al fine di risolvere un problema.



⁸più o meno comodo...

Ingredienti

Nella programmazione si ha a che fare con due ingredienti di base:

- dati (informazione)
- procedure (operazioni da effettuare)

Dati e procedure vengono definiti e utilizzati mediante tre meccanismi:

primitive (o espressioni primitive), la entità concettuali più elementari su cui un linguaggio si basa;

combinazione di primitive, per generare espressioni via via più complesse;

astrazione tramite la quale poter generalizzare e manipolare concetti complessi



Un esempio di programma...

Ingredients

3 eggplants, peeled and cut lengthwise into 1/2 inch thick slices

salt

1/4 cup olive oil

1 tablespoon butter

1 pound [lean ground beef](#)

salt to taste

ground black pepper to taste

2 onions, chopped

1 clove garlic, minced

1/4 teaspoon ground cinnamon

1/4 teaspoon ground nutmeg

1/2 teaspoon fines herbs

2 tablespoons dried parsley

1 (8 ounce) can [tomato sauce](#)

1/2 cup red wine

1 egg, beaten

4 cups milk

1/2 cup butter

6 tablespoons [all-purpose flour](#)

salt to taste

ground white pepper, to taste

1 1/2 cups freshly grated Parmesan cheese

1/4 teaspoon ground nutmeg

Un programma è
una sorta di
“ricetta”...



Un esempio di programma...

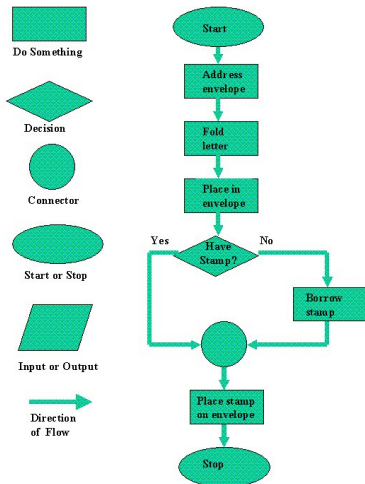
Directions

1. Lay the slices of eggplant on paper towels, sprinkle lightly with salt, and set aside for 30 minutes to draw out the moisture. Then in a skillet over high heat, heat the olive oil. Quickly fry the eggplant until browned. Set aside on paper towels to drain.
2. In a large skillet over medium heat, melt the butter and add the ground beef, salt and pepper to taste, onions, and garlic. After the beef is browned, sprinkle in the cinnamon, nutmeg, fines herbs and parsley. Pour in the tomato sauce and wine, and mix well. Simmer for 20 minutes. Allow to cool, and then stir in beaten egg.
3. To make the bechamel sauce, begin by scalding the milk in a saucepan. Melt the butter in a large skillet over medium heat. Whisk in flour until smooth. Lower heat; gradually pour in the hot milk, whisking constantly until it thickens. Season with salt, and white pepper.
4. Arrange a layer of eggplant in a greased 9x13 inch baking dish. Cover eggplant with all of the meat mixture, and then sprinkle 1/2 cup of **Parmesan cheese** over the meat. Cover with remaining eggplant, and sprinkle another 1/2 cup of cheese on top. Pour the bechamel sauce over the top, and sprinkle with the nutmeg. Sprinkle with the remaining cheese.
5. Bake for 1 hour at 350 degrees F (175 degrees C).

Un programma è
una sorta di
“ricetta”...



Un esempio di programma...



Un programma è una sorta di “ricetta”... .. per “cucinare” soluzioni a problemi definiti.

Cosa fa un programmatore?

- Definisce un problema



Cosa fa un programmatore?

- Definisce un problema
- Trova una soluzione
(... o più di una)



Cosa fa un programmatore?

- Definisce un problema
- Trova una soluzione
(... o più di una)
- Traduce la soluzione in istruzioni per il calcolatore
(cioè scrive un programma!)



Cosa fa un programmatore?

- Definisce un problema
- Trova una soluzione
(... o più di una)
- Traduce la soluzione in istruzioni per il calcolatore
(cioè scrive un programma!)
- Verifica che il programma funzioni correttamente
(il cosiddetto *debugging*)



Cosa fa un programmatore?

- Definisce un problema
- Trova una soluzione
(... o più di una)
- Traduce la soluzione in istruzioni per il calcolatore
(cioè scrive un programma!)
- Verifica che il programma funzioni correttamente
(il cosiddetto *debugging*)
- Se necessario individua eventuali errori e li corregge



Cosa fa un programmatore?

- Definisce un problema
- Trova una soluzione
(... o più di una)
- Traduce la soluzione in istruzioni per il calcolatore
(cioè scrive un programma!)
- Verifica che il programma funzioni correttamente
(il cosiddetto *debugging*)
- Se necessario individua eventuali errori e li corregge
- Documenta ciò che ha fatto
(per facilitare l'uso del programma e l'eventuale manutenzione)



Definire il problema

La definizione di un problema consiste nell'identificare *ciò che è noto*, cioè quali sono le informazioni in ingresso e *ciò che si vuole ottenere* (i dati in uscita).

È la parte più difficile del lavoro. . .



Definire il problema

Identificare ciò che è noto

Spesso anche un problema apparentemente banale, quando deve essere tradotto in un programma, presenta difficoltà.

- I termini utilizzati per descrivere il problema sono ambigui?
- Identificare i processi che determinano il problema.

Comunicare con chi dovrà usare (o chi ha richiesto⁹) il programma.



⁹... o con la vostra papera di gomma. Greg Duff, su IBM 3270, circa 1980.

Trovare una soluzione¹⁰

Schematizzare le possibili soluzioni può essere d'aiuto per definire la *strategia di programmazione*.

Esistono due tecniche di base:

Flowcharting schematizzare i vari passaggi creando un *diagramma di flusso*

Pseudocoding delineare i vari passaggi *per iscritto*

¹⁰... o anche due.

Trovare una soluzione

Flowcharting

Un diagramma di flusso è una rappresentazione grafica, passo per passo, delle “azioni” che un programma svolge.

Si utilizzano poligoni per rappresentare le singole azioni e frecce per indicarne lo svolgimento.

È una **mappa** del programma.

Esiste una norma ANSI che definisce i simboli utilizzabili.



Trovare una soluzione

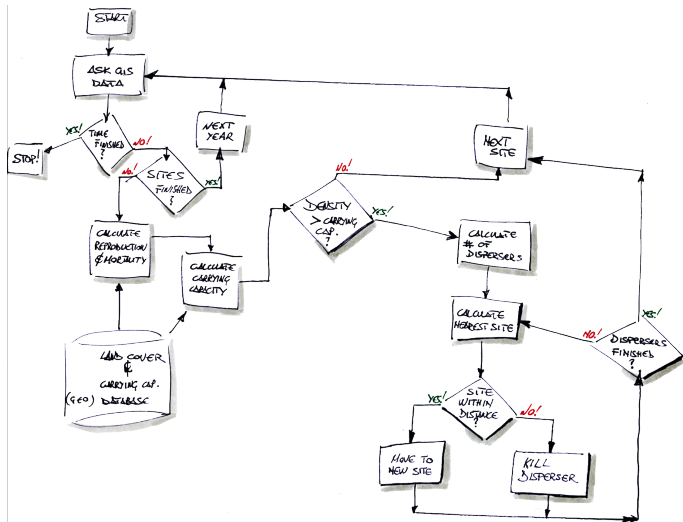
Pseudocoding

La pseudocodifica consiste nello scrivere uno *pseudo-programma*, in una sorta di linguaggio che non è un vero e proprio linguaggio (formale) di programmazione, ma è già molto simile ad esso.



Trovare una soluzione

Un esempio di *flowchart*



Trovare una soluzione

Un esempio di pseudocodice

Insertion Sort ($A[1..n]$):

for $j = 2$ to n $\{$
 // (I)
 $\text{key} = A[j]$
 $\text{// insert } A[j] \text{ into sorted sequence } A[1..j-1]$
 $i = j - 1$
 while ($i > 0$ && $A[i] > \text{key}$)
 $A[i+1] = A[i];$
 $i--;$
 $A[i+1] = \text{key}$
}

Diagram illustrating the insertion sort process. The array contains elements 3, 5, 10, 20, 4. The first four elements (3, 5, 10, 20) are grouped under a bracket labeled "Sorted". The element 4 is at index j . A vertical line is drawn between index $j-1$ and j . The value $\text{key} = 4$ is written below the array.



Trovare una soluzione

Lo scopo della rappresentazione *informale* (pseudocodifica, diagramma di flusso) del programma, in ultima analisi, è quello di ottenere un *modello concettuale* del programma stesso, propedeutico alla *rappresentazione formale*, cioè al programma vero e proprio.

Sia la pseudocodifica che il diagramma di flusso *non* dipendono da nessuna specifica formale di linguaggio.



Scrivere il programma

Il “codice sorgente”

Il modello concettuale (informale), viene tradotto secondo precise “regole di grammatica” in un dato *linguaggio formale* (linguaggio di programmazione). Come nei linguaggi umani (scritti e parlati), non c'è nessuna garanzia che una “frase” grammaticalmente corretta abbia un qualsiasi significato. . .



Verificare il programma

Nonostante in campo pratico si sostenga che un programma ben progettato può essere scritto correttamente sin dall'inizio...

Nonostante alcuni teorici insistano sul fatto che esistano metodi formali per provare che un programma è "corretto"...

1100 Started ^{in relay} Cosine Tape (Sine check) 10.00 test
1525 Started ^{Relays changed} Multi-Adder Test. Relay 3376
1545  Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
~~1630~~ 1630 Antangant started.
1700 closed down.



Verificare il programma

Debugging e verifiche

... Inevitabilmente un programma *può* contenere errori.

errori di “grammatica” refusi, sgrammaticature, segni di interpunzione. . .

errori di “sintassi” costrutti errati, uso scorretto o inesatto di tecniche o strutture dati. . .

errori “ideologici” cattiva progettazione. . .



Verificare il programma

Caccia all'errore

Esistono diverse tecniche per scoprire e correggere eventuali errori:

Rilettura esattamente come si rilegge un testo scritto, rileggere (o meglio *far rileggere a un'altra persona* il proprio codice)



Verificare il programma

Caccia all'errore

Esistono diverse tecniche per scoprire e correggere eventuali errori:

Rilettura esattamente come si rilegge un testo scritto, rileggere (o meglio *far rileggere a un'altra persona* il proprio codice)

Desk-checking rileggere il codice, seguendo il flusso delle operazioni del programma¹¹. In alcuni casi, è previsto che il codice venga riletto e verificato da altri programmatori¹²!

¹¹È sempre possibile rileggerlo e spiegarlo alla propria papera di gomma...

¹²Tipicamente, nel campo *Open Source*



Verificare il programma

Caccia all'errore

Esistono diverse tecniche per scoprire e correggere eventuali errori:

Rilettura esattamente come si rilegge un testo scritto, rileggere (o meglio *far rileggere a un'altra persona* il proprio codice)

Desk-checking rileggere il codice, seguendo il flusso delle operazioni del programma¹¹. In alcuni casi, è previsto che il codice venga riletto e verificato da altri programmatori¹²!

Traduzione un traduttore (o *compilatore*) è un programma che traduce il codice sorgente in forma utilizzabile direttamente dal calcolatore: eventuali errori vengono intercettati e notificati.

¹¹È sempre possibile rileggerlo e spiegarlo alla propria papera di gomma. . .

¹²Tipicamente, nel campo *Open Source*



Verificare il programma

Caccia all'errore

Esistono diverse tecniche per scoprire e correggere eventuali errori:

Rilettura esattamente come si rilegge un testo scritto, rileggere (o meglio *far rileggere a un'altra persona* il proprio codice)

Desk-checking rileggere il codice, seguendo il flusso delle operazioni del programma¹¹. In alcuni casi, è previsto che il codice venga riletto e verificato da altri programmatori¹²!

Traduzione un traduttore (o *compilatore*) è un programma che traduce il codice sorgente in forma utilizzabile direttamente dal calcolatore: eventuali errori vengono intercettati e notificati.

Debugging si tenta di “far girare” il programma, seguendone passo per passo il funzionamento, verificando la correttezza dello svolgimento delle operazioni.

¹¹È sempre possibile rileggerlo e spiegarlo alla propria papera di gomma. . .

¹²Tipicamente, nel campo *Open Source*



Documentare il programma

Alcuni programmatori pensano che un programma ben scritto si documenti da sè. . .

La documentazione è parte integrante di un programma, e non ha senso *scriverla dopo*.

- Origine e natura del problema
- Strategia adottata per risolverlo
- Rappresentazioni logiche (diagrammi di flusso, pseudocodice)
- Descrizione dei test di verifica e dei risultati attesi
- Modalità di utilizzo



Documentare il programma

Codice che “si documenta da sè”

La documentazione è essenziale

- per tenere traccia del lavoro svolto
- per lasciare una traccia a chi dovrà lavorare sul codice
- per poter riutilizzare in futuro codice già scritto
- per comunicare con altri

I believe that the time is ripe for significantly better documentation of programs, and that we can best achieve this by considering programs to be works of literature. Hence, my title: “Literate Programming”.

– Donald E. Knuth (1984), *Literate Programming*



Sommario

- 1 Presentazione dell'ambiente R
- 2 Il sistema R: come si lavora, cosa si fa
- 3 Principi di programmazione
- 4 Il linguaggio R**
- 5 Usare R

Alcune convenzioni

- R è *case sensitive*
- Comandi (in realtà *funzioni*) e *parole riservate*
- Variabili: possono avere qualunque nome
- Un comando viene terminato con Enter (o Invio), o con “;”
- È possibile creare comandi *compositi* racchiudendoli tra graffe ({ e })¹³
- È possibile inserire *commenti* dovunque: un commento inizia con “#”, i caratteri seguenti sino a fine riga vengono ignorati
- Se un comando non è grammaticalmente completo, R lo rende noto mediante il simbolo “+”

¹³in R, come in molti programmi e linguaggi di derivazione UNIX, sono comuni caratteri non presenti sulla tastiera italiana, quali { } ~ ecc.



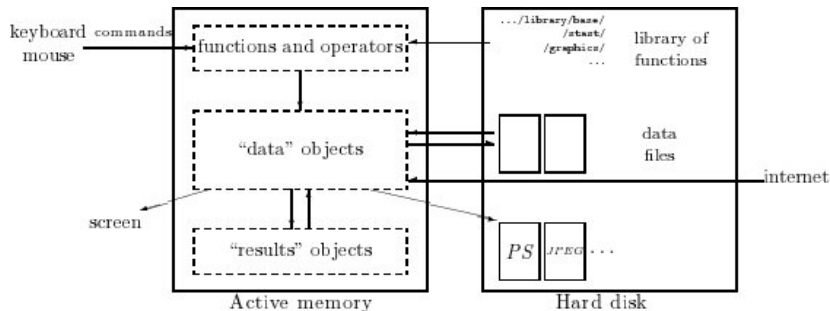
I “personaggi”

- R immagazzina **tutto** in memoria (RAM) variabili, dati, funzioni, risultati ecc.¹⁴
- in R qualunque “cosa” è un *oggetto*, cioè un insieme di dati e di funzioni per manipolarli



¹⁴Il che non esclude che si possa lavorare con la *memoria di massa* (disco fisso, dischi remoti, web, ecc.)

In R “tutto è un oggetto”



Un esempio

File: src/example_01.R

```
1 # let a new variable be, called 'n', with value equal to 10 + 2
  n <- 10 + 2
3 # o R, what is the value of n?
  n
5
# now n will be 3 + a random gaussian deviate (mean 1, sd 1)
7 n <- 3 + rnorm(1)
# o R, what is the value of n?
9 n
11 # and now?
   n <- 3 + rnorm(100)
```



Un esempio

File: src/example_01.R

```
1 # let a new variable be, called 'n', with value equal to 10 + 2
  n <- 10 + 2
3 # o R, what is the value of n?
  n
5
# now n will be 3 + a random gaussian deviate (mean 1, sd 1)
7 n <- 3 + rnorm(1)
# o R, what is the value of n?
9 n
11 # and now?
  n <- 3 + rnorm(100)
```

Ma che succede?

- `n` inizialmente è un numero intero
- successivamente (riga 7) `n` diventa un numero reale...
- infine, `n` diventa un *vettore* di 100 numeri reali!!!



Tutto in R è un oggetto...

- Un oggetto è una combinazione di dati e dei metodi per manipolarli
- Un oggetto “sa” gestirsi
- Un oggetto può “discendere” da altri oggetti, e ne eredita le “capacità” (*ereditarietà*)
- Diversi oggetti “sanno” compiere azioni simili (*polimorfismo*)



Tutto in R è un oggetto...

File: src/example_02.R

```
1 # load some canned data ...
  data(iris)
3 data(sunspots)
  data(precip)
5 # what sort of R objects we have?
  ls()
7 # and of what kind are they?
  class(iris)
9 class(sunspots)
  class(precip)
11 # a dataframe, a time series, a named numeric vector ...and yet
  summary(iris)
13 summary(sunspots)
  summary(precip)
```



Ma... che cos'è un oggetto?

Un oggetto (o *classe*) è un modello astratto di *dati* e *metodi* per operare su di essi: si parla in questo caso di *incapsulamento* di dati e funzioni collegate ai dati.

La programmazione a oggetti permette di sfruttare alcune tecniche:

incapsulamento “tutto è contenuto nell’oggetto”: una volta creata una classe di oggetti, li si può utilizzare senza preoccuparsi di come sia il loro “funzionamento interno”

polimorfismo diversi oggetti possono “fare cose simili in modo diverso”: è possibile creare metodi con lo stesso nome, che assolvono a funzioni simili

ereditarietà una classe di oggetti può *discendere* da una classe *progenitrice*: la classe di oggetti “figlia” avrà automaticamente tutte le caratteristiche ereditate dalla classe “madre”



Le conseguenze (vantaggiose!)

- Grazie al *polimorfismo*, è frequente che un dato comando R, una volta appreso, serva in molti altri casi (es.: `summarize`)
- È possibile *estendere* il linguaggio¹⁵, basandosi sulle classi già esistenti e modificandole grazie all'*ereditarietà*
- È possibile operare con strutture di dati parecchio complesse, senza che sia necessario conoscere il loro funzionamento interno, grazie all'*incapsulamento*

¹⁵... e siete caldamente incoraggiati a farlo, R è anche un linguaggio *funzionale*!



Comandi base

- Per assegnare un valore a una variabile si usa il simbolo `<-`
- Utilizzare come comando il nome di un oggetto, dà come risultato il *vedere* l'oggetto stesso
- Ricordarsi sempre che:
 - ▶ una funzione R ha obbligatoriamente le parentesi
 - ▶ R è *case-sensitive*
 - ▶ non si usa `\` (*backslash*) ma `/` (*slash*)



Comandi base

- Per assegnare un valore a una variabile si usa il simbolo `<-`
- Utilizzare come comando il nome di un oggetto, dà come risultato il *vedere* l'oggetto stesso
- Ricordarsi sempre che:
 - ▶ una funzione R ha obbligatoriamente le parentesi
 - ▶ R è *case-sensitive*
 - ▶ non si usa `\` (*backslash*) ma `/` (*slash*)

Cheat sheet!

È umanamente impossibile ricordarsi *tutti* i comandi. . .

R Cheat sheet



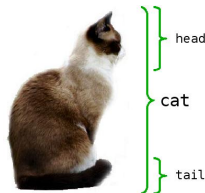
Manuale di sopravvivenza...

Molti comandi e costrutti in R imitano comandi e costrutti comuni in UNIX¹⁶

`ls()` list, elenco degli oggetti

`rm()` remove, elimina oggetti

`str()` structure, visualizza la struttura di un oggetto



`cat()` concatenate, concatena stringhe di testo, visualizza in forma grezza un oggetto

`head()` visualizza le prime parti di un oggetto

`tail()` visualizza le ultime parti di un oggetto

`help(),?` richiama il manuale

`help.search(),??` ricerca nel manuale

¹⁶Master Foo Discourses on Returning to Windows <http://catb.org/esr/writings/unix-koans/returning.html>

Un esempio: elenco oggetti

File: src/example_03.R

```
2 # what objects are present in the current R space?  
  ls  
4 # ops.. I meant 'call the function ls', but what I wrote was instead  
  # 'show me how the object called 'ls' is made inside '  
6  
8 # more politely  
  ls()
```

Un comando R è una *funzione*¹⁷, e deve *sempre* “avere le parentesi”!!

¹⁷ proprio nel senso di “ $f(x)$!”

R è vettoriale

Aniché operare con singoli valori (*scalari*), R assume di lavorare *sempre* su vettori.

Questo comporta:

- 1 l'enfasi sul fatto che una *variabile*¹⁸ è un vettore;
- 2 il fatto che un'operazione si effettui applicandola *implicitamente* a tutti gli elementi dell'intero vettore (R “ripete” da solo!)
- 3 un vettore in R non si limita solo a un vettore numerico. . .

File: src/example_04.R

```
1 # let 's have 100 random numbers, normally distributed
  my.variable <- rnorm(100)
3
4 # calculate a mean
5 mean(my.variable)
6
7 # what if I 'apply' a function?
  log.myvar <- log10(my.variable)
```

¹⁸Più appropriatamente, un campione di n realizzazioni di una variabile aleatoria

Mi servirebbe una funzione R per fare...

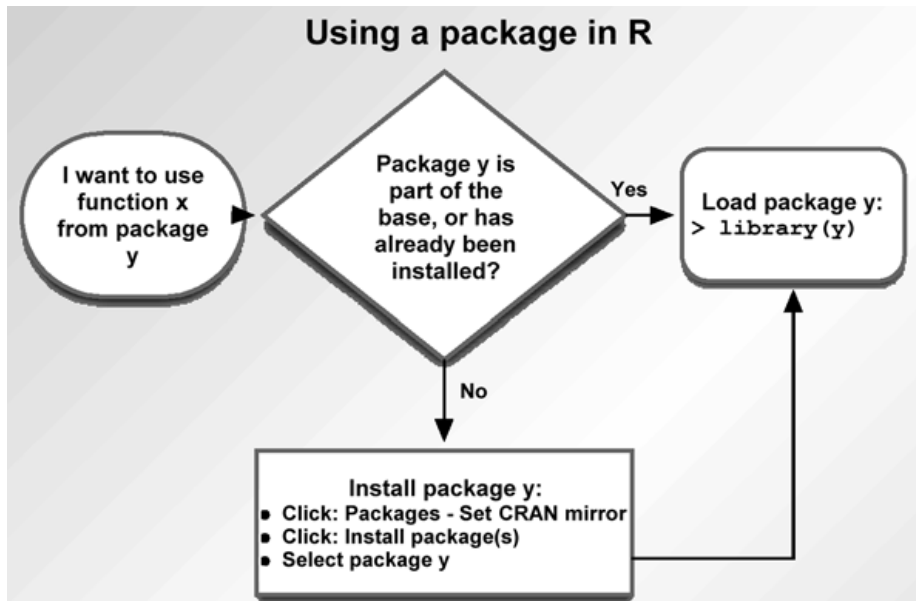
- 1 controllare gli *abstract* dei pacchetti esistenti
`http://cran.r-project.org/web/packages/`
- 2 installare il pacchetto desiderato con
`install.packages("a_package_name", dep=TRUE)`¹⁹
- 3 caricare il pacchetto con
`library(a_package_name)`²⁰

¹⁹selezionare il mirror CRAN più vicino

²⁰esiste una sottile differenza tra `library()` e `require()`...



Mi servirebbe una funzione R per fare...



Alcune estensioni utili

`foreign` legge/scrive i formati dati più comuni

`XLConnect` legge/scrive i fogli elettronici Excel (occorre avere installato Java!)

`readxl` legge i fogli elettronici Excel (“puro R”)

`lattice` *trellis graphics*

`ggplot2` *Wilkinson-Wickham “grammar of graphics”*

`circular` statistica circolare

`nlme` *mixed-model ANOVA*

`rggobi` analisi esplorativa di dati multidimensionali²¹

`ade4` analisi multivariata

`bioconductor` tutto per la bioinformatica
(<http://www.bioconductor.org/>)

... e molti molti altri ancora!!

²¹ purtroppo, dà problemi con RStudio...



L'unico comando da ricordare a memoria

?



L'unico comando da ricordare a memoria

?

This is all documented in TFM. Those who WTFM don't want to have to WTFM again on the mailing list. RTFM²².

– Barry Rowlingson, R-help mailing list, October 2003

²²Read That F... Manual

L'unico comando da ricordare a memoria

R: Cross Tabulation and Table Creation ▾

table {base}

R Documentation

Cross Tabulation and Table Creation

Description

`table` uses the cross-classifying factors to build a contingency table of the counts at each combination of factor levels.

Usage

```
table(..., exclude = if (useNA == "no") c(NA, NaN), useNA = c("no",  
  "ifany", "always"), dnn = list.names(...), deparse.level = 1)
```

```
as.table(x, ...)
```

```
is.table(x)
```

```
## S3 method for class 'table'
```

```
as.data.frame(x, row.names = NULL, ...)
```



Come leggere una *manpage*

Qualsiasi comando R possiede una pagina di manuale (o *manpage*)

Una qualsiasi *manpage* è organizzata in *sezioni*

intestazione nome del comando, breve descrizione

Description descrizione per esteso delle funzionalità

Usage “grammatica” del comando

Arguments descrizione di dettaglio di quanto specificato nella sezione
Usage

Details eventuali approfondimenti, riferimenti alla letteratura,
peculiarità

Value com'è organizzato il *risultato*

Source riferimenti all'implementazione utilizzata

References bibliografia!

See Also rimandi ad altri comandi o a concetti generali

Examples esempi pratici (provare con il copia-e-incolla!)



Come usare una *manpage*

Ma come arrivo a *quella* pagina?

- Se non si conosce il nome di un comando:
??<comando, o qualcosa di simile>
- Se si conosce il comando ma non ci si ricorda le “regole di grammatica”:
?<comando>

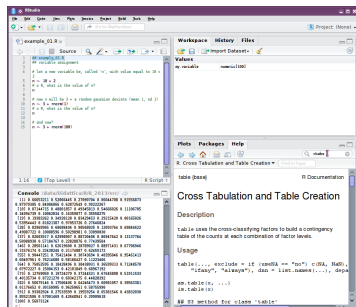


Come usare una *manpage*

Ma come arrivo a *quella* pagina?

- Se non si conosce il nome di un comando:
??<comando, o qualcosa di simile>
- Se si conosce il comando ma non ci si ricorda le “regole di grammatica”:
?<comando>

RStudio ha l'help integrato!

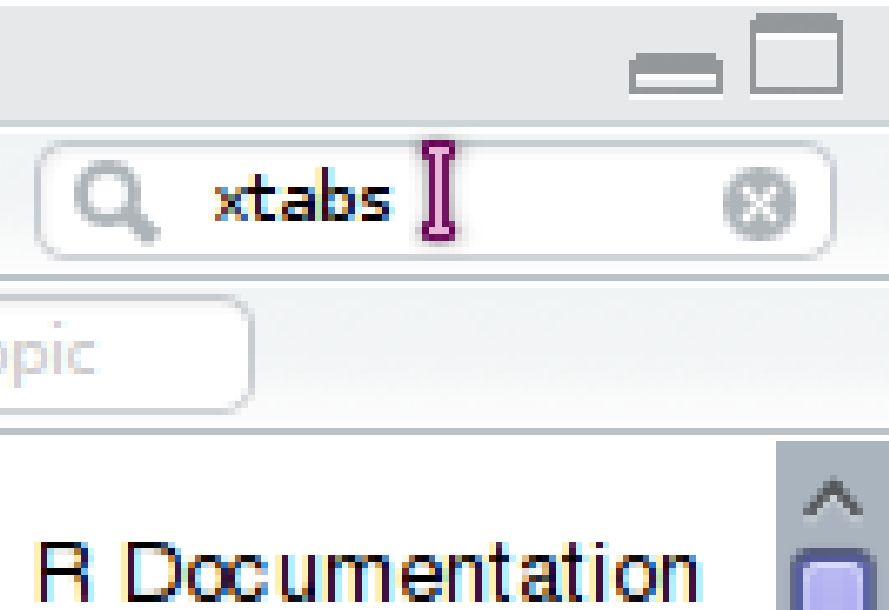


The screenshot shows the RStudio interface with the help page for the `table` function open in the Help pane. The console shows the execution of `table(mtcars)` and the resulting output. The help page includes a description of the function, usage instructions, and a code example.

```
## 83 method for class "table"
```



Come usare una *manpage*



Sommario

- 1 Presentazione dell'ambiente R
- 2 Il sistema R: come si lavora, cosa si fa
- 3 Principi di programmazione
- 4 Il linguaggio R
- 5 Usare R**

Analisi in R

The formulation of a problem is often more essential than its solution which may be merely a matter of mathematical or experimental skill.

– Albert Einstein



Analisi in R

The formulation of a problem is often more essential than its solution which may be merely a matter of mathematical or experimental skill.

– Albert Einstein

È più importante la *formulazione* di un problema che la sua soluzione.



Analisi in R

The formulation of a problem is often more essential than its solution which may be merely a matter of mathematical or experimental skill.

– Albert Einstein

È possibile scrivere un programma una volta che i termini del problema sono *ben definiti*



Come definire un problema?

- 1 Comprendere la natura e l'origine dei dati.
- 2 Definire degli obiettivi.
- 3 *Keep it simple and stupid*
- 4 Tradurre il problema in termini statistici^a
- 5 Non basta identificare un metodo per processare i dati... i risultati potrebbero comunque essere privi di senso!

^a“Se riesci a fare un grafico, riesci a fare l'analisi”



Per iniziare...

- 1 importazione dei dati
- 2 “organizzazione” dei dati (*condizionamento*)
- 3 statistiche descrittive (*summary*, *stem*)
- 4 analisi grafica: *plot*, *xyplot*²², *bwplot*²², *qplot*²³, *ggplot*²³
- 5 Ricerca (ed eliminazione) di eventuali *outlier*
- 6 Se necessario, *imputazione* di dati mancanti
- 7 Test per la normalità e relative conseguenti trasformazioni

²²package *lattice*

²³package *ggplot2*



Per iniziare...

- 1 importazione dei dati
- 2 “organizzazione” dei dati (*condizionamento*)
- 3 statistiche descrittive (*summary*, *stem*)
- 4 analisi grafica: *plot*, *xyplot*²², *bwplot*²², *qplot*²³, *ggplot*²³
- 5 Ricerca (ed eliminazione) di eventuali *outlier*
- 6 Se necessario, *imputazione* di dati mancanti
- 7 Test per la normalità e relative conseguenti trasformazioni
- 8 Analisi statistica vera e propria
- 9 Restituzione dei risultati (grafici e/o testuali)

²²package lattice

²³package ggplot2

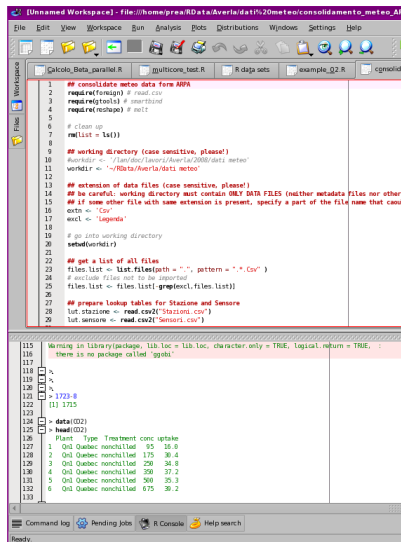
Ma... in pratica?

Le varie fasi del processo di elaborazione dati si concretizzano in differenti sezioni (consecutive, come in una ricetta di cucina...) di uno *script* in linguaggio R



Struttura base di uno script R

- 1 importazione dati
- 2 condizionamento dati
- 3 EDA (*Exploratory data analysis*²⁴)
- 4 analisi propriamente detta
- 5 grafici



```
1 ## consolidate meteo data form ARPA
2 require(foreign) # read.csv
3 require(gtools) # stringind
4 require(reshape) # melt
5
6 # clean up
7 rm(list = ls())
8
9 ## working directory (case sensitive, please!)
10 #workdir <- "~/Desktop/averla/2008/dati meteo"
11 workdir <- "~/RData/averla/dati meteo"
12
13 ## extension of data files (case sensitive, please!)
14 ## be careful: working directory must contain ONLY DATA FILES (neither metadata files nor other
15 ## if some other file with same extension is present, specify a part of the file name that causes
16 extn <- ".csv"
17 excl <- "Legenda"
18
19 # go into working directory
20 setwd(workdir)
21
22 ## get a list of all files
23 files.list <- list.files(path = ".", pattern = "*.csv")
24 # exclude files not to be imported
25 files.list <- files.list[-grep(excl, files.list)]
26
27 ## prepare Lookup tables for Stazione and Sensore
28 lut_stazione <- read.csv2("stazioni.csv")
29 lut_sensore <- read.csv2("sensori.csv")
```

```
115 Warning in library(package, lib.loc = lib.loc, character.only = TRUE, logical.return = TRUE,
116 : there is no package called 'ggobi'
117
118 [ ]
119 [ ]
120 [ ]
121 [ ] * 1723-8
122 [ ] 1715
123
124 [ ] = data(D2)
125 [ ] = head(D2)
126
127 Plant Type Treatment conc uptake
128 1 Qnl Quebec nonchilled 95 16.0
129 2 Qnl Quebec nonchilled 175 39.4
130 3 Qnl Quebec nonchilled 250 34.8
131 4 Qnl Quebec nonchilled 350 37.2
132 5 Qnl Quebec nonchilled 500 35.3
133 6 Qnl Quebec nonchilled 675 36.2
```

²⁴ *sensu* Tukey, 1977

And now... for something completely different!

An Introduction to R <http://cran.r-project.org/doc/manuals/R-intro.html>

R for beginners http://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf

icebreakR <http://cran.r-project.org/doc/contrib/Robinson-icebreaker.pdf>

Introduction to the R Project for Statistical Computing for use at ITC

<http://cran.r-project.org/doc/contrib/Rossiter-RIntro-ITC.pdf>

Esercitazioni di statistica biomedica—alcune note su R <http://cran.r-project.org/doc/contrib/DellOmodarme-esercitazioni-R.pdf>

<http://cran.r-project.org/doc/contrib/DellOmodarme-esercitazioni-R.pdf>

R reference cards <http://cran.r-project.org/doc/contrib/Short-refcard.pdf>

Rootless root—The Unix Koans of Master Foo

<http://catb.org/esr/writings/unix-koans/>

R Search Engine <http://rseek.org> (ma esiste anche il comando `RSiteSearch...`)

Practical Regression and Anova using R

<http://cran.r-project.org/doc/contrib/Faraway-PRA.pdf>





Programmazione statistica in R: corso teorico-pratico

Parte 2

Damiano G. Preatoni

Unità di Analisi e Gestione delle Risorse Ambientali – *Guido Tosi Research Group*
Dipartimento di Scienze Teoriche e Applicate, Università degli Studi dell'Insubria
prea@uninsubria.it

7-9/7/2021



Contenuti

- 6 I “dati” in R
- 7 Inserire dati in R: manualmente
- 8 Inserire dati in R: le strutture dati standard
- 9 Statistiche di base e analisi esplorativa
- 10 Grafica in R



Sommario

6 I “dati” in R

- Tipi di dato
- Iterabilità dei dati

7 Inserire dati in R: manualmente

8 Inserire dati in R: le strutture dati standard

- Dataframe
- Liste

9 Statistiche di base e analisi esplorativa

10 Grafica in R

Ma dove stanno i dati?

In R esistono diversi tipi di oggetti “contenitore”²⁵

Ciascun tipo di “contenitore” (in R: oggetto dati) può contenere dati di diverso “tipo” (in R: modo)

²⁵e numerosi altri vengono messi a disposizione da pacchetti specifici



Ma dove stanno i dati?

In R esistono diversi tipi di oggetti “contenitore”²⁵

Ciascun tipo di “contenitore” (in R: oggetto dati) può contenere dati di diverso “tipo” (in R: modo)

Modi

- logical
- integer
- numeric
- character
- factor
- ...

Data objects

- vector
- matrix
- list
- dataframe
- ...

²⁵e numerosi altri vengono messi a disposizione da pacchetti specifici



R “itera” da solo

In R si dà per scontato che un oggetto dati sia “iterabile”: qualsiasi operazione viene “applicata” a tutti gli elementi di quell’oggetto.



R “itera” da solo

In R si dà per scontato che un oggetto dati sia “iterabile”: qualsiasi operazione viene “applicata” a tutti gli elementi di quell’oggetto.

Applicazione iterativa

File: src/example_01.R

```
1 n <- 3 + rnorm(100)
```

R aggiunge 3 a tutti gli elementi del vettore n



R “itera” da solo

In R si dà per scontato che un oggetto dati sia “iterabile”: qualsiasi operazione viene “applicata” a tutti gli elementi di quell’oggetto.
Se un’operazione coinvolge elementi di lunghezza diversa, l’elemento più corto viene *riciclato*



R “itera” da solo

In R si dà per scontato che un oggetto dati sia “iterabile”: qualsiasi operazione viene “applicata” a tutti gli elementi di quell’oggetto. Se un’operazione coinvolge elementi di lunghezza diversa, l’elemento più corto viene *riciclato*

R “ricicla”

File: src/example_05.R

```
1 a <- c(1,1,1,1,1,1,1,1)
2 b <- c(4,0)
3
a + b
```

R “ripete” il vettore *b* in modo da poter iterare su tutti gli elementi del vettore *a*



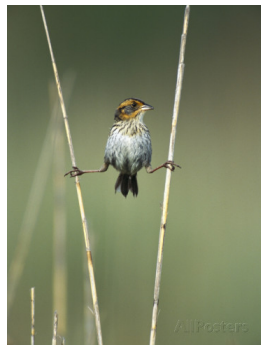
Sommario

- 6 I “dati” in R
 - Tipi di dato
 - Iterabilità dei dati
- 7 Inserire dati in R: manualmente**
- 8 Inserire dati in R: le strutture dati standard
 - Dataframe
 - Liste
- 9 Statistiche di base e analisi esplorativa
- 10 Grafica in R

Creare dati in R

Un po' di biometrie... (*Ammodramus caudacutus*, passero codacuta, N America ²⁶)

Wing cord	Tarsus	Head	Weight
59	22.3	31.2	9.5
55	19.7	30.4	13.8
53.5	20.8	30.6	14.8
55	20.3	30.3	15.2
52.5	20.8	30.3	15.5
57.5	21.5	30.8	15.6
53	20.6	32.5	15.6
55	21.5	NA	15.7

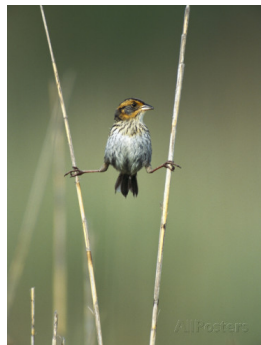


²⁶Chris Elphick, University of Connecticut, unpublished data, da Zuur et al., 2009, dataset originale: 1100 osservazioni.

Creare dati in R

Un po' di biometrie... (*Ammodramus caudacutus*, passero codacuta, N America ²⁶)

Wing cord	Tarsus	Head	Weight
59	22.3	31.2	9.5
55	19.7	30.4	13.8
53.5	20.8	30.6	14.8
55	20.3	30.3	15.2
52.5	20.8	30.3	15.5
57.5	21.5	30.8	15.6
53	20.6	32.5	15.6
55	21.5	NA	15.7



Come inserire questi dati?

²⁶Chris Elphick, University of Connecticut, unpublished data, da Zuur et al., 2009, dataset originale: 1100 osservazioni.



Creare dati in R

Esistono alcuni comandi di base per creare dati.



Creare dati in R

Esistono alcuni comandi di base per creare dati.

- `c()`
- `rep()`
- `matrix()`
- `list()`
- `data.frame()`



Creare dati in R ...“a mano” I

File: src/ZUUR_CodeforChapter2.r

```
1 Wingcrd <- c(59, 55, 53.5, 55, 52.5, 57.5, 53, 55)
  Wingcrd[1]
3
5 S.win <- sum(Wingcrd)
  S.win
7
9 Tarsus <- c(22.3, 19.7, 20.8, 20.3, 20.8, 21.5, 20.6, 21.5)
  Head <- c(31.2, 30.4, 30.6, 30.3, 30.3, 30.8, 32.5, NA)
  Wt <- c(9.5, 13.8, 14.8, 15.2, 15.5, 15.6, 15.6, 15.7)
11
13 sum(Head, na.rm = TRUE)
15
17 BirdData <- c(Wingcrd, Tarsus, Head, Wt)
  Id <- c(1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2,
           2, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4)
19
21 rep(c(1, 2, 3, 4), each = 8)
  rep(1 : 4, each = 8)
  a <- seq(from = 1, to = 4, by = 1)
  a
23
25 a <- seq(from = 1, to = 4, by = 1)
  rep(a, each = 8)
```



Creare dati in R ... "a mano" II

```
27
29 VarNames <- c("Wingcrd", "Tarsus", "Head", "Wt")
31 Id <- rep(VarNames, each = 8)
33 Z <- cbind(Wingcrd, Tarsus, Head, Wt)
35 n <- dim(Z)
37 n <- dim(Z)[1]
39 W <- vector(length = 8)
41 W[1] <- 59
43 W[2] <- 55
45 W[3] <- 53.5
47 W[4] <- 55
49 W[5] <- 52.5
51 W[6] <- 57.5
53 W[7] <- 53
55 W[8] <- 55
57
59 Dmat <- matrix(nrow = 8, ncol = 4)
61 Dmat
63
65 Dmat[, 1] <- c(59, 55, 53.5, 55, 52.5, 57.5, 53, 55)
67 Dmat[, 2] <- c(22.3, 19.7, 20.8, 20.3, 20.8, 21.5,
```



Creare dati in R ...“a mano” III

```
55      20.6, 21.5)
Dmat[, 3] <- c(31.2, 30.4, 30.6, 30.3, 30.3, 30.8,
57      32.5, NA)
Dmat[, 4] <- c(9.5, 13.8, 14.8, 15.2, 15.5, 15.6,
59      15.6, 15.7)

61 Dmat

63 colnames(Dmat) <- c("Wingcrd", "Tarsus", "Head", "Wt")
Dmat

65 Dmat2 <- as.matrix(cbind(Wingcrd, Tarsus, Head, Wt))

67 Dfrm <- data.frame(WC = Wingcrd, TS = Tarsus,
59      HD = Head, W = Wt)
Dfrm

71

73 M <- lm(WC ~ Wt, data = Dfrm)

75

77 AllData <- list(BirdData = BirdData, Id = Id, Z = Z,
      VarNames = VarNames)

79 AllData <- list(BirdData, Id, Z, VarNames)
```



Creare “a mano”? ... ma ne vale la pena?

File: src/ZUUR_CodeforChapter2_dataframe.R

```
1 DFrame <- data.frame(  
  Wingcrd <- c(59, 55, 53.5, 55, 52.5, 57.5, 53, 55),  
  Tarsus <- c(22.3, 19.7, 20.8, 20.3, 20.8, 21.5, 20.6, 21.5),  
  Head <- c(31.2, 30.4, 30.6, 30.3, 30.3, 30.8, 32.5, NA),  
  Wt <- c(9.5, 13.8, 14.8, 15.2, 15.5, 15.6, 15.6, 15.7)  
)
```



Creare dati in R

Esistono alcuni comandi di base per creare dati.

- `c()`
- `rep()`
- `matrix()`
- `list()`
- `data.frame()`



Creare dati in R

Esistono alcuni comandi di base per creare dati.

- `c()`
- `rep()`
- `matrix()`
- `list()`
- `data.frame()`

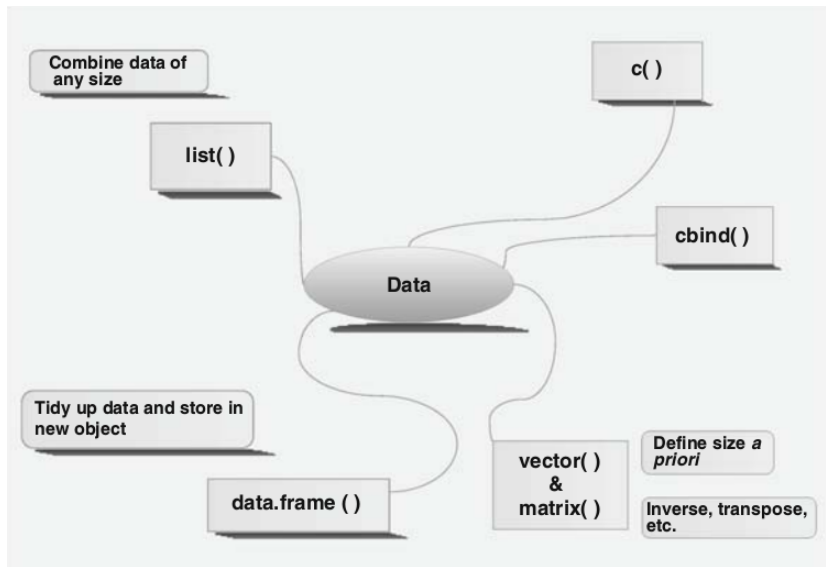
... ma i dati di solito sono già presenti in file esterni a R

Di norma, si tratta quasi sempre di individuare il sistema più comodo per *importare* file di dati esistenti in R

Tutti (o quasi) i comandi per importare creano dei *dataframe*



Creare dati in R



Comandi per caricare file di dati

`data(x)` carica un set di dati “di serie” in R
`read.table(...)` carica dati da un file di testo
`read.csv(...)` carica dati da un file di testo in formato `.csv`
`read.*(...)` forma generale...



Comandi per caricare file di dati

`data(x)` carica un set di dati “di serie” in R
`read.table(...)` carica dati da un file di testo
`read.csv(...)` carica dati da un file di testo in formato `.csv`
`read.*(...)` forma generale...

I comandi in R non sono difficili da imparare!

Quale potrebbe essere il comando per leggere un file Excel (`.xls`)?



Sommario

- 6 I “dati” in R
 - Tipi di dato
 - Iterabilità dei dati
- 7 Inserire dati in R: manualmente
- 8 Inserire dati in R: le strutture dati standard**
 - Dataframe
 - Liste
- 9 Statistiche di base e analisi esplorativa
- 10 Grafica in R

Dataframe

Il contenitore di dati più comune è il *dataframe*.

- analogo a un foglio elettronico (“righe e colonne”)
- colonne “tipizzate” (una data colonna è in un dato “modo”)
- un dataframe funziona come una matrice (indicizzazione per righe e colonne)
- le colonne hanno un nome



Dataframe

Il contenitore di dati più comune è il *dataframe*.

- analogo a un foglio elettronico (“righe e colonne”)
- colonne “tipizzate” (una data colonna è in un dato “modo”)
- un dataframe funziona come una matrice (indicizzazione per righe e colonne)
- le colonne hanno un nome

Un esempio

File: src/example_06.R

```
1 data(iris)
# see the 'head' ;)
3 head(iris)
# see the structure
5 str(iris)
# see column names
7 names(iris)
```


Dataframe

Il contenitore di dati più comune è il *dataframe*.

- analogo a un foglio elettronico (“righe e colonne”)
- colonne “tipizzate” (una data colonna è in un dato “modo”)
- un dataframe funziona come una matrice (indicizzazione per righe e colonne)
- le colonne hanno un nome

Lavorare con i dataframe

File: src/example_06.R

```
1 # refer to a column  
iris$Species  
3 iris[, 'Species']  
iris[,5]
```



Ma... com'è fatto “dentro” un oggetto R?

In R è possibile creare dei “contenitori per dati” anche estremamente articolati.

È sempre possibile esaminare la *struttura* di un oggetto R mediante il comando `str()`.



Ma... com'è fatto “dentro” un oggetto R?

In R è possibile creare dei “contenitori per dati” anche estremamente articolati.

È sempre possibile esaminare la *struttura* di un oggetto R mediante il comando `str()`.

“Contenitori” e strutture dati

Poter disporre di una struttura definita (e apparentemente **rigida**) in cui immagazzinare i dati è un vantaggio:

- si “demanda alla macchina” il compito di tenere in ordine i dati, evitando incoerenze!
- si può comunque accedere a qualunque elemento di una struttura dati!



Dataframe: *slicing*

Un dataframe è una matrice “specializzata”.

È possibile quindi utilizzare la notazione comunemente utilizzata per riferirsi a degli elementi della “matrice”

`dataframe[<riga>, <colonna>]`



Dataframe: *slicing*

Un dataframe è una matrice “specializzata”.

È possibile quindi utilizzare la notazione comunemente utilizzata per riferirsi a degli elementi della “matrice”

`dataframe[<riga>, <colonna>]`

Notazione “tra quadre”

Entro parentesi quadre è possibile specificare *qualsiasi espressione* che sia valida per righe e/o colonne:

- indici (numerici o “per nome”)
- espressioni logiche o algebriche (“filtraggio”)



Dataframe: *slicing*

Un dataframe è una matrice “specializzata”.

È possibile quindi utilizzare la notazione comunemente utilizzata per riferirsi a degli elementi della “matrice”

dataframe[<riga>, <colonna>]

Notazione “tra quadre”

File: src/example_06.R

```
1 # make 'slices'  
  iris[1:3, 'Sepal.Length']  
3 # select by row: only a species  
  iris[iris$Species=='setosa',]  
5 # select by row: two species  
  iris[iris$Species==c('setosa', 'versicolor'),]  
7 # select by row: sepal length <= 5  
  iris[iris$Sepal.Length <= 5,]  
9 # select by row: sepal length under the mean  
  iris[iris$Sepal.Length <= mean(iris$Sepal.Length),]
```



Dataframe: *slicing*

Un dataframe è una matrice “specializzata”.

È possibile quindi utilizzare la notazione comunemente utilizzata per riferirsi a degli elementi della “matrice”

```
dataframe[<riga>, <colonna>]
```

Si può lavorare “dentro” un dataframe

Un dataframe può essere collegato al *percorso di ricerca* dell’ambiente R mediante il comando `attach()`: non è più necessario specificare il nome del dataframe.

Il comando “opposto” è `detach`.

Non è comodo^a utilizzare `attach` quando si lavora con più dataframe contemporaneamente.

^a...anzi. è *vivamente sconsigliato!*



Liste

Un altro utile “contenitore” è la *lista* (`list`)

- collezione *eterogenea* di oggetti: può contenere *di tutto*!
- gli elementi di lista hanno un *nome*, oltre che una posizione
- funziona come un vettore, con qualche comodità in più
- è “naturalmente iterabile”



Liste: un esempio I

File: src/example_06b.R

```
1 ## Lists examples
2 n <- c(2, 3, 5) # a vector (of integers)
3 s <- c("aa", "bb", "cc", "dd", "ee") # a vector (of characters)
4 b <- c(TRUE, FALSE, TRUE, FALSE, FALSE) # a vector (of logicals)
5 x <- list(n, s, b, 3) # a list: x contains copies of n, s, b
6
7 # list slicing: square brackets
8 x[2]
9 x[c(2, 4)]
10 # direct access: _double_ square brackets
11 x[[2]]
12 # direct access is needed to _modify_ list elements
13 x[[2]][1] <- "ta"
14 x[[2]]
15 # the list contains a copy of s, which is unaffected
16 s
17 # named lists
18 v <- list(bob=c(2, 3, 5), john=c("aa", "bb"))
19 v
20 # named lists slicing
21 v["bob"]
22 # named lists direct access
23 v[["bob"]]
```



Dataframe: *do it yourself!* I

File: src/example_07.R

```
## example_07.R
2 ## data frames, part II
3 data(mtcars)
4 # is there any documentation on this built-in data set?
5 ?mtcars
6 # show the dataframe
7 mtcars
8 # size up the dataframe
9 nrow(mtcars)
10 ncol(mtcars)
11 str(mtcars)
12 # print cell value from the first row, second column
13 mtcars[1, 2]
14 # the same, using row and column names
15 mtcars["Mazda RX4", "cyl"]
16 # print a single column: type of transmission
17 mtcars[,9]
18 mtcars[[9]]
19 mtcars[["am"]]
20 mtcars$am
21 mtcars[, "am"]
22 # column slicing: get column 1
23 mtcars[,1]
24 mtcars["mpg"]
```



Dataframe: *do it yourself!* II

```
26 # get miles per gallon and horsepower
   mtcars[c("mpg", "hp")]
28 # row slicing: print a single row
   mtcars[24,]
30 # rows 3 and 24
   mtcars[c(3, 24),]
32 # get a row 'by name'
   mtcars["Camaro Z28",]
34 mtcars[c("Datsun 710", "Camaro Z28"),]
   # place in a new dataframe all the cars with automatic transmission
36 automatic <- mtcars$am == 0
   mtcars[automatic,]
38 # the same, in one line
   mtcars[mtcars$am == 0,]
40 # mileage for cars with automatic transmission
   mtcars[mtcars$am == 0,]$mpg
```



Sommario

- 6 I “dati” in R
 - Tipi di dato
 - Iterabilità dei dati
- 7 Inserire dati in R: manualmente
- 8 Inserire dati in R: le strutture dati standard
 - Dataframe
 - Liste
- 9 Statistiche di base e analisi esplorativa**
- 10 Grafica in R

Per iniziare...

- 1 importazione dei dati
- 2 “organizzazione” dei dati (*condizionamento*)
- 3 statistiche descrittive (*summary*, *stem*)
- 4 analisi grafica: `plot`, `xyplot`²², `bwplot`²², `qplot`²³, `ggplot`²³
- 5 Ricerca (ed eliminazione) di eventuali *outlier*
- 6 Se necessario, *imputazione* di dati mancanti
- 7 Test per la normalità e relative conseguenti trasformazioni
- 8 Analisi statistica vera e propria
- 9 Restituzione dei risultati (grafici e/o testuali)

²²package `lattice`

²³package `ggplot2`



Statistiche di base e analisi esplorativa

Una volta “acquisiti” i dati grezzi, è *indispensabile* “dare un'occhiata:

- è avvenuto qualche pasticcio nell'importazione?
- i dati sono del tipo atteso (stringhe che sembrano numeri. . .)
- esistono valori “estremi” (*outliers*)?
- i dati richiedono qualche trasformazione?
- i dati devono essere “normali”?
- . . .



Comandi per “esplorare”

`summary(x)` fa il “sommario” di qualunque tipo di dato: statistiche descrittive per un intero dataframe

`stem(x)` *stem-and-leaf plot*

`mean(x)` media

`median(x)` mediana

`sd(x)` deviazione standard

`unique(x)` (per variabili testuali) elenco dei valori, senza duplicati

`quantile(x, probs=)` calcolo dei quantili (default: quartili)

...



Comandi per “esplorare”

`summary(x)` fa il “sommario” di qualunque tipo di dato: statistiche descrittive per un intero dataframe

`stem(x)` *stem-and-leaf plot*

`mean(x)` media

`median(x)` mediana

`sd(x)` deviazione standard

`unique(x)` (per variabili testuali) elenco dei valori, senza duplicati

`quantile(x, probs=)` calcolo dei quantili (default: quartili)

...

Curiosamente, *non esistono* in R i comandi per calcolare l'errore standard e la varianza... come mai?



Comandi per “esplorare”

`summary(x)` fa il “sommario” di qualunque tipo di dato: statistiche descrittive per un intero dataframe

`stem(x)` *stem-and-leaf plot*

`mean(x)` media

`median(x)` mediana

`sd(x)` deviazione standard

`unique(x)` (per variabili testuali) elenco dei valori, senza duplicati

`quantile(x, probs=)` calcolo dei quantili (default: quartili)

R è un linguaggio *funzionale* . . .

Curios
la vari

```
se <-function(x) sd(x)/sqrt(length(x))
```

```
se <-function(x) sqrt(var(x)/length(x))
```

standard e



Analisi esplorativa: *do it yourself!*

Dati epidemiologici da Vicente et al. (2006).

Presenza di *Elaphostrongylus cervi* in esemplari di cervo e cinghiale (portatore sano).

- i dati sono in un file Excel? Come leggerlo?
- che struttura hanno i dati? Quante righe? Quante colonne?
- Cosa contengono le varie colonne?
- Il campione è omogeneo per *sex*?
- Il campione è omogeneo per *anno*?
- Il campione è omogeneo per *allevamento*?
- I valori di lunghezza testa-corpo sono diversi tra maschi e femmine?



Analisi esplorativa: *do it yourself!* I

File: src/example_07b.R

```
## example_07b.R
## load and explore deer data (Vicente et al. 2006) from Zuur et al. 2009.
2
4 rm(list=ls()) # this clears the R workspace
6 ## Where are the data?
## What if data are moved or renamed?
8
# store in a variable the path where the Excel file 'Deer.xls' is
10 wd <- '/home/rea/Dropbox/R/R_2015/slide/src/data/'
# Also store in a variable the file name
12 datafile <- 'Deer.csv'
14 # change R 'working directory' into the one where the spreadsheet is
setwd(wd)
16
# load the Excel file
18 Deer <- read.csv(datafile)
20 # explore: how big is the dataset?
nrow(Deer)
22 ncol(Deer)
24 # how are data organized?
str(Deer)
```



Analisi esplorativa: *do it yourself!* II

```
26
28 # how many observations for each sex?
   table(Deer$Sex)
30
32 # better recode Sex...
   Deer$SEX <- factor(Deer$Sex, levels=c(1,2), labels=c('M', 'F'))
34
36 # recode as well clas1_4
   Deer$CLASS <- factor(Deer$clas1_4, levels=c(1,2,3,4), labels=c('1', '2', '3', '4')
   )
38
40 # go on... how many observation for each sex?
   table(Deer$SEX)
42 # observations by year?
   table(Deer$Year)
44 # observations by farm
   xtabs( ~ Year, data=Deer)
46 # observations by sex and year
   table(Deer$Sex, Deer$Year)
48 # by sex and year
   xtabs( ~ SEX + Year, data=Deer)
50
52 # average LCT?
   mean(Deer$LCT)
   # NA?
   mean(Deer$LCT, na.rm=TRUE)
   # mean by sex
```



Analisi esplorativa: *do it yourself!* III

```
54 aggregate(LCT ~ SEX, data=Deer, FUN=mean, na.rm=TRUE)
# mean by sex and age class
56 aggregate(LCT ~ SEX + clas1_4, data=Deer, FUN=mean, na.rm=TRUE)
```



Bibliografia

- <https://cran.r-project.org/other-docs.html>
- Venables WN, Smith DM, R Core Team, 2015. An Introduction to R. <https://cran.r-project.org/doc/manuals/R-intro.pdf>
- Paradis E, 2005. R for beginners. https://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf
- Rossiter DG, 2014. Introduction to the R Project for Statistical Computing for use at ITC http://www.css.cornell.edu/faculty/dgr2/pubs/list.html#pubs_m
- Verzano J., 2002. SimpleR. Using R for Introductory statistics. <https://cran.r-project.org/doc/contrib/Verzani-SimpleR.pdf>
-
- Zuur AF, Ieno EN, Meesters EHWG, 2009. A Beginner's Guide to R. UseR! Series, Springer.
- Logan M, 2010. Biostatistical design and analysis using R : a practical guide. Wiley.

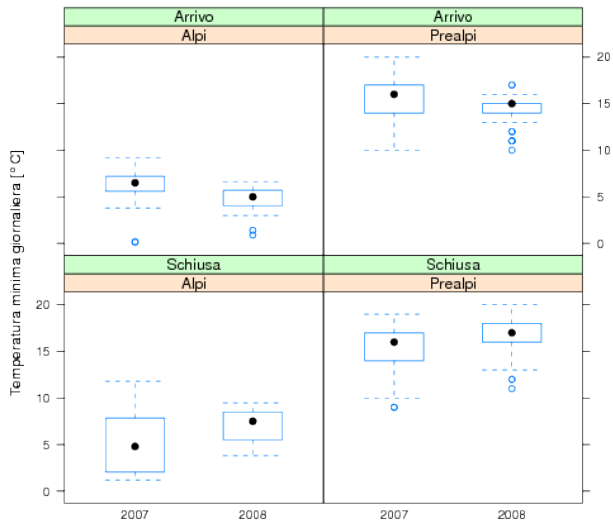




Sommario

- 6 I “dati” in R
 - Tipi di dato
 - Iterabilità dei dati
- 7 Inserire dati in R: manualmente
- 8 Inserire dati in R: le strutture dati standard
 - Dataframe
 - Liste
- 9 Statistiche di base e analisi esplorativa
- 10 Grafica in R**

Cosa c'è dietro a un grafico?



Cosa c'è dietro a un grafico?

Creare un grafico in R significa programmare le “regole” per costruirlo. Questo (ovviamente) non significa indicare “quali sono le serie di dati”, ma *come deve essere “calcolato”* il grafico.

Se un grafico non riesce come vi aspettavate. . . probabilmente ha ragione R!



Cosa c'è dietro a un grafico?

Creare un grafico in R significa programmare le “regole” per costruirlo. Questo (ovviamente) non significa indicare “quali sono le serie di dati”, ma *come deve essere “calcolato”* il grafico.

Se un grafico non riesce come vi aspettavate. . .
probabilmente ha ragione R!

La “grafica” non si inventa!

Cleveland W, 1985. Elements of Graphic Data. Brooks Cole.

Wilkinson L, 1999. The Grammar of Graphics. Springer.



Tre modi di fare grafica

`base` plot e famiglia... approccio per “serie di dati”

`lattice` grafici come *formule*

`ggplot2` *Grammar of Graphics*



Un esempio con plot I

File: src/example_07c.R

```
1 rm(list=ls())
3 data(iris)
5 attach(iris)
7 # specify X and Y
  plot(Petal.Length, Petal.Width)
9
11 # same, but using the 'formula interface'
  # note: Y ~ X
  plot(Petal.Width ~ Petal.Length)
13
15 # color by Species
  plot(Petal.Length, Petal.Width, pch=(21:23)[as.numeric(Species)], cex=1.2,
      xlab="Petal length (cm)", ylab="Petal width (cm)",
      main="Anderson Iris data",
      col=c("slateblue", "firebrick", "darkolivegreen")[as.numeric(Species)])
19
21 # add trendlines
  abline(v=mean(Petal.Length), lty=2, col="red")
23 abline(h=mean(Petal.Width), lty=2, col="red")
  abline(v=median(Petal.Length), lty=2, col="blue")
25 abline(h=median(Petal.Width), lty=2, col="blue")
```



Un esempio con plot II

```
27 # add a grid
   grid()
29
31 # add centroids
   points(mean(Petal.Length), mean(Petal.Width),
          cex=2, pch=23, col="black", bg="red")
33   points(median(Petal.Length), median(Petal.Width),
          cex=2, pch=23, col="black", bg="blue")
35
37 # add a subtitle
   title(sub="Centroids: mean (green) and median (gray)")
39
41 # free-format annotations
   text(1, 2.4, "Three species of Iris", pos=4, col="navyblue")
43
45 # a legend
   legend(1, 2.4, levels(Species), pch=21:23, bty="n",
         col=c("slateblue", "firebrick", "darkolivegreen"))
47
49 # regression lines
   abline(lm(Petal.Width ~ Petal.Length), lty="longdash", col="red")
   library(MASS) # for lqs function
   abline(lqs(Petal.Width ~ Petal.Length), lty=2, col="blue")
```





Programmazione statistica in R: corso teorico-pratico

Parte 3

Damiano G. Preatoni

Unità di Analisi e Gestione delle Risorse Ambientali – *Guido Tosi Research Group*
Dipartimento di Scienze Teoriche e Applicate, Università degli Studi dell'Insubria
prea@uninsubria.it

7-9/7/2021



Sommario

- 11 Un'applicazione pratica: frammentazione dell'habitat e body condition index nei Chiroteri
- 12 Alcuni script commentati

Un caso reale

In molte specie di Mammiferi, il *Body Condition Index* è indice indiretto della *fitness* degli individui.

Esistono differenze nella *fitness* di femmine riproduttive di *Rhinolophus ferrumequinum* in tre differenti siti?



Un caso reale

In molte specie di Mammiferi, il *Body Condition Index* è indice indiretto della *fitness* degli individui.

Esistono differenze nella *fitness* di femmine riproduttive di *Rhinolophus ferrumequinum* in tre differenti siti?

Dati disponibili

- biometrie
- numero di individui presenti per ogni colonia
- quota
- metriche di paesaggio



Un caso reale

In molte specie di Mammiferi, il *Body Condition Index* è indice indiretto della *fitness* degli individui.

Esistono differenze nella *fitness* di femmine riproduttive di *Rhinolophus ferrumequinum* in tre differenti siti?

Problemi

- i dati sono relativi a 4 siti
- gli individui sono 40
- quota e metriche di paesaggio sono relative ai 4 siti



Un caso reale

In molte specie di Mammiferi, il *Body Condition Index* è indice indiretto della *fitness* degli individui.

Esistono differenze nella *fitness* di femmine riproduttive di *Rhinolophus ferrumequinum* in tre differenti siti?

Soluzioni

- accorpare (media, sd) le biometrie per sito
- unire i dati biometrici accorpati ai dati di paesaggio



Un caso reale I

File: src/example_08_bats.R

```
## 1. prepare working environment
# clean up the workspace
rm(list=ls())

# we need some libraries:
library(foreign) # read.csv, read.csv2
require(Hmisc) # xYplot
require(lattice) # lattice plots

# specify a 'working directory'
setwd('/data/Didattica/R/R_2013/src/data')

## 2. get data
# site data (direct input)
site.data <- data.frame(localita=c("Cormons", "Pandona", "S. Cesario", "Toggiano"),
                        quota=c(58,140,56,600),
                        consistenza=c(190,360,30,26))

# read in bci data
bci.data <- read.csv2('BCI.csv', as.is=TRUE)
# read in landscape data
land.data <- read.csv('landscape.csv')
```



Un caso reale II

```
26 ## 3. data conditioning
28 # condition some variable in bci.data
29 bci.data$localita <- factor(bci.data$localita,
30                            levels=c('JAP', 'COR', 'SCP', 'TOG'),
31                            labels=c('Pandona', 'Cormons', 'S. Cesario', 'Toggiano'
32                                    ))
33 bci.data$secco <- factor(bci.data$secco)
34 bci.data$eta <- factor(bci.data$eta)
35 # we need only female adults
36 bci.data <- bci.data[bci.data$secco == 'f' & bci.data$eta == 'ad',]
37 # aggregate BCI by site
38 bci.avg <- aggregate(bci.data$BCI, by=list(bci.data$localita), FUN="mean")
39 names(bci.avg) <- c('localita', 'BCI.avg')
40 bci.sd <- aggregate(bci.data$BCI, by=list(bci.data$localita), FUN="sd")
41 names(bci.sd) <- c('localita', 'BCI.sd')
42 bci.avg <- merge(bci.avg, bci.sd, by="localita")
43 rm(bci.sd)
44 bci.avg <- merge(bci.avg, site.data, by="localita")
45 # now, dataframe bci.avg contains all average stats
46 # condition data in landscape metrics data
47 land.data$localita <- factor(land.data$LID, levels=c('D:\\w_rhifer\\uso_4', 'D
48             :\\w_rhifer\\uso_6', 'D:\\w_rhifer\\uso_9', 'D:\\w_rhifer\\uso_11'),
49             labels=c('Pandona', 'Cormons', 'S. Cesario', 'Toggiano'))
50 # merge bci + landscape data
51 land <- merge(bci.avg, land.data, by="localita")
52 land$LID <- NULL # delete LID column
```



Un caso reale III

4. Exploratory Data Analysis

BCI vs. elevation

```
xYplot(Cbind(BCI.avg,BCI.sd) ~ quota, groups=localita, bci.avg, type=c("p"),  
        method="bars", ylim=c(.25,.4), label.curves=TRUE, auto.key=TRUE, ylab="BCI",  
        xlab="Quota (m s.l.m.)")
```

```
cor.test(bci.avg$BCI.avg, bci.avg$quota)
```

BCI vs. colony size

```
xYplot(Cbind(BCI.avg,BCI.sd) ~ consistenza, groups=localita, bci.avg, type=c("p")  
        , method="bars", ylim=c(.25,.4), label.curves=TRUE, auto.key=TRUE, ylab="BCI"  
        , xlab="Consistenza stimata della colonia (individui)")
```

```
cor.test(bci.avg$BCI.avg, bci.avg$consistenza)
```

Automate correlation analysis between BCI and Landscape level metrics

We need to repeat the same analysis for 70 variables!!

get landscape variables names

```
land.vars <- names(land.data)[3:length(names(land.data))-1]
```

define correlation tests 'limits'

```
corr.sig <- 0.06 # passes test if less than or equal to
```

```
corr.val <- 0.3 # passes test if greater then or equal to
```



Un caso reale IV

```
78 land.vars.good <- vector() # to store interesting variable names
# open a 'sink file' to write tests results, but close it immediately
# to avoid writing garbage into
82 sink(file='landscape_correlations.txt', split=TRUE)
sink()
# begin plotting & making correlations
86 for(v in land.vars) {
  filename <- paste("plots/land_BCI",v,"png",sep=".")
  cat("plotting",filename,"...")
  x <- land[[v]]
  c <- cor.test(land$BCI.avg,x)
  # a little data stashing to have a vector of 'interesting' variables
  if(!is.na(c$estimate)) {
    s <- paste("r=",formatC(c[["estimate"]],3)," p=",formatC(c[["p.value"]],3),
              sep="")
    sink(file='landscape_correlations.txt', split=TRUE, append=TRUE)
    cat("correlation of", v, "with BCI", s, "\n")
    sink()
    if (abs(c[["estimate"]]) >= corr.val & c[["p.value"]] <= corr.sig) {
      land.vars.good <- c(land.vars.good, v)
      cat("SIGNIFICANT!")
    }
  } else {
    s <- "regression not possible"
  }
  plt <- xyplot(BCI.avg ~ x, land, type=c('p','r'), xlab=v, ylab="BCI", sub=s)
```

Un caso reale V

```
06 #direct.label(plt)
    png(filename=filename)
    print(plt)
08 dev.off()
    cat("\tDone.\n")
10 }
    sink()
12 rm(v,x,c,s,plt)
14 ## End of File
```



Qualche commento...

Nello *script* sono stati utilizzati diversi comandi per

- importare dati
- modificare dataframe
- aggregare dati
- automatizzare analisi ripetitive



Import/export

Esistono numerosi *package* per leggere e scrivere praticamente qualsiasi tipo di dato

foreign legge e scrive csv, dbf e formati di altri software per l'elaborazione dati

gdata fornisce una funzione `read.xls`, ma richiede il linguaggio perl²⁷

²⁷ che su UNIX è di serie. . .



Import/export

Esistono numerosi *package* per leggere e scrivere praticamente qualsiasi tipo di dato

foreign legge e scrive csv, dbf e formati di altri software per l'elaborazione dati

gdata fornisce una funzione `read.xls`, ma richiede il linguaggio `perl`²⁷

Il problema dei fogli elettronici

Il foglio elettronico è *comodo*, ma il formato fisico dei file cambia continuamente. In più, è preferibile utilizzare un *package* che funzioni su qualsiasi piattaforma.

Attualmente, conviene utilizzare o `XLConnect` (che utilizza Java), o `readxl`, che è “puro R”.

²⁷ che su UNIX è di serie. . .



Manipolazione dataframe

Esistono innumerevoli comandi per “lavorare” con i dataframe

cbind unisce dataframe “accostando” le colonne

rbind unisce dataframe “accostando” le righe

merge unisce due dataframe in base a una colonna chiave presente in entrambi



Manipolazione dataframe

Esistono innumerevoli comandi per “lavorare” con i dataframe

cbind unisce dataframe “accostando” le colonne

rbind unisce dataframe “accostando” le righe

merge unisce due dataframe in base a una colonna chiave presente in entrambi

R ricicla...

cbind richiede che i due dataframe abbiano lo stesso numero di righe.

rbind richiede che i dataframe abbiano la stessa struttura.

merge effettua invece una vera e propria operazione di *giunzione*.



Manipolazione dataframe

Esistono innumerevoli comandi per “lavorare” con i dataframe

cbind unisce dataframe “accostando” le colonne

rbind unisce dataframe “accostando” le righe

merge unisce due dataframe in base a una colonna chiave presente in entrambi

Il *package* `plyr` è in grado di “piegare” un dataframe in ogni modo possibile, in base a regole di accorpamento, ma utilizza una sintassi complessa²⁸.

²⁸ che alla fine conviene imparare

Aggregare e tabulare dati

È possibile *applicare* a una variabile di un dataframe una *funzione di aggregazione*²⁹ e ottenere un nuovo dataframe “aggregato” in base ai valori di una seconda variabile.

Per casi più complessi (es. aggregazione su più variabili) è più conveniente adottare un'altra strategia, di tipo *map-reduce*³⁰.

Si tratta di:

- 1 dividere il dataframe in più dataframe indipendenti (es. `split()`)
- 2 processare ciascun dataframe separatamente (es. `lapply()`)
- 3 “rimettere insieme i pezzi” (es. `plyr::ldply()` o brutalmente con `do.call(rbind, ...)`)

²⁹ad esempio `mean`, `sum`, `sd`...

³⁰Una strategia come *map-reduce* rende il programma di fatto *già parallelizzabile!*



Il tipo di dati factor

Serve per rappresentare variabili *nominali*

Viene rappresentato *internamente* con numeri interi (“levels”)

Viene rappresentato *esternamente* con stringhe (“labels”)



Aggregare e tabulare dati: xtabs

È possibile *crosstabulare*³¹ una variabile di un dataframe in base ad altre variabili (che vengono automaticamente convertite in factor).
xtabs utilizza un'*interfaccia a formula*, tecnica frequentissima in R.

³¹ cioè creare delle tabelle di contingenza



Aggregare e tabulare dati: xtabs

È possibile *crosstabulare*³¹ una variabile di un dataframe in base ad altre variabili (che vengono automaticamente convertite in factor).
xtabs utilizza un' *interfaccia a formula*, tecnica frequentissima in R.

Oggetti “formula”

```
<left side term> ~ <right side term>
```

³¹ cioè creare delle tabelle di contingenza



Aggregare e tabulare dati: xtabs

È possibile *crosstabulare*³¹ una variabile di un dataframe in base ad altre variabili (che vengono automaticamente convertite in factor).
xtabs utilizza un' *interfaccia a formula*, tecnica frequentissima in R.

Oggetti “formula”: esempi

`data$y ~ a + b * data$x`: un'equazione per una regressione

`d$response ~ d$dose * d$age + Error(d$subject)`: un modello con errore casuale per un'analisi della varianza

...

³¹ cioè creare delle tabelle di contingenza



Aggregare e tabulare dati: xtabs

È possibile *crosstabulare*³¹ una variabile di un dataframe in base ad altre variabili (che vengono automaticamente convertite in factor).
xtabs utilizza un' *interfaccia a formula*, tecnica frequentissima in R.

Oggetti “formula”: esempi

```
<variabile da tabulare> ~<dimensioni della tabella>: xtabs
```

Se non viene specificato nulla nel termine di sinistra, xtabs *conta le righe*.
Se viene specificata una variabile numerica, xtabs *somma*.

³¹ cioè creare delle tabelle di contingenza



Aggregare e tabulare dati: xtabs

È possibile *crosstabulare*³¹ una variabile di un dataframe in base ad altre variabili (che vengono automaticamente convertite in factor).
xtabs utilizza un' *interfaccia a formula*, tecnica frequentissima in R.

Oggetti "formula": xtabs

<variabile da tabulare> ~ <dimensioni della tabella>

File: src/xtabs.R

```
xtabs( ~ Specie , data )
  Specie
lepcom lepvar
  745    432
```

³¹ cioè creare delle tabelle di contingenza



Automazione: flusso del programma

Programmare significa innanzitutto demandare alla macchina i compiti ripetitivi.

R offre un linguaggio di programmazione sufficientemente ricco: ?Control

if(cond) expr test logici

if(cond) cons.expr else alt.expr test logici a due alternative

for(var in seq) expr iterazione per un numero finito di ripetizioni

while(cond) expr iterazione indefinita in base a una condizione

repeat expr iterazione indefinita

ifelse test logico “in line”

switch test logico a più “uscite”



Automazione: flusso del programma

Programmare significa innanzitutto demandare alla macchina i compiti ripetitivi.

R offre un linguaggio di programmazione sufficientemente ricco: ?Control

if(cond) expr test logici

if(cond) cons.expr else alt.expr test logici a due alternative

for(var in seq) expr iterazione per un numero finito di ripetizioni

while(cond) expr iterazione indefinita in base a una condizione

repeat expr iterazione indefinita

ifelse test logico “in line”

switch test logico a più “uscite”

Dato che R “itera naturalmente”, è meglio *evitare* di utilizzare cicli `for`.

In più, un `for` è difficilmente parallelizzabile, mentre `lapply` (e famiglia) lo è (*package parallel*)



Automazione: funzioni

R è un *linguaggio funzionale*, cioè si basa sulla definizione di *funzioni*

- funzioni *sensu* $f(x)$
- non necessariamente operanti solo su dati numerici
- *nuovi comandi* a tutti gli effetti
- operano indipendentemente dallo stato del programma

L'approccio funzionale consente di *estendere* le capacità di R, ad esempio allestendo librerie di funzioni personalizzate³²

³²... che possono anche essere utilizzate per creare nuovi *package*



Sommario

- 11 Un'applicazione pratica: frammentazione dell'habitat e body condition index nei Chiroteri
- 12 Alcuni script commentati

Proviamo da soli...

Creare una directory locale³³ nella quale copiare di volta in volta i *dataset* per ciascun esercizio.

Un *dataset* comprende uno *script* R e i relativi file di dati.

Ciascuno *script* è commentato, in modo da dare una spiegazione delle tecniche e dei comandi utilizzati.

In ogni *script* sono citati i riferimenti alle pagine di manuale relative ai diversi comandi utilizzati.

In caso di dubbio su qualunque comando *usare l'help*

R help system

?<nome comando> mostra la pagina di manuale

??<keyword> ricerca

RSiteSearch<keyword> cerca sul sito R

<http://rseek.org> “specializzazione” di Google per ricerche su R

³³ad esempio C:\R



Attenzione ai metacommenti!

@TODO

In alcuni punti degli *script* alcuni comandi dovranno necessariamente essere adattati, ad esempio nel caso dati e *script* siano in una cartella differente. Tali comandi sono preceduti da un commento che inizia con il metacommento @TODO.

@Q&A

In alcuni punti degli *script* l'uso di alcuni comandi particolari o di alcune tecniche *dovrebbe* far venire alcuni dubbi o quantomeno suscitare un minimo id curiosità. Tali comandi sono preceduti da un commento che inizia con il metacommento @Q&A.

Riferimenti

I commenti che iniziano con il testo Riferimenti rimandano alle pagine di manuale.

Lepri e gradienti altitudinali

Script file: src\example_09_hares2.R

Dataset: src\data\hares.csv

Descrizione: osservazioni di lepre alpina e lepre comune (2000–2008) in 5 Comprensori Alpini di Caccia della provincia di Sondrio.

Obiettivo

In Valtellina, *Lepus timidus* e *L. europaeus* occupano orizzonti altitudinali diversi?



Lepri e gradienti altitudinali I

File: src/example_09_hares2.R

```
# Gli oggetti di una sessione di lavoro R rimangono in memoria (i.e. nel
  workspace).
2 # Prima di iniziare una nuova sessione di analisi, e' opportuno "svuotare"
  # lo spazio di lavoro
4 # Riferimenti: ?rm
  rm(list=ls())
6
8 # Risulta comodo immagazzinare dati, script e risultati nella stessa directory.
  # Conviene utilizzare una directory separata per ciascuna analisi.
  # Spesso - pero' - succede di dover spostare i vari file, e al contempo non
10 # desiderabile dover rivedere l'intero script a causa del cambiamento del
  # nome di un file o dello spostamento di dati e script in una differente
12 # directory.
  # Una tecnica elementare consiste nel dichiarare in anticipo gli 'elementi
14 # variabili', in pratica definendoli come costanti. In caso di variazioni
  # sara' sufficiente sistemare i nuovi valori in un punto solo dello script.
16 # Definiamo come costante la directory nella quale stanno i dati
  ##@TODO modificare rispetto alla propria directory di lavoro
18 workdir <- "/data/Didattica/R/R_2013/src/"
  # definisce la 'cartella di lavoro' di R
20 setwd(workdir)
  ##Q&A: e se non uso setwd()? Dove 'lavora' (ad esempio salva i risultati) R?
22
24 ##@TODO modificare togliendo in carattere di commento a uno (e uno solo!)
```



Lepri e gradienti altitudinali II

```
##@TODO dei due blocchi di codice seguenti
## legge i dati direttamente da un foglio Excel (occorre avere installato
## il package XLConnect
#require(XLConnect)
#data <- readWorksheetFromFile(paste("data","hares.xls", sep="/"), sheet="totale
")
## legge i dati da un file di testo CSV (Comma Separated Values, occorre avere
## il package foreign)
library(foreign)
data <- read.csv(paste("data","hares.csv", sep="/"))
##Q&A: che differenza c'e' tra i comandi require() e library()?
##Q&A: che cosa fa il comando paste()? Provare a copiare e incollare nella linea
di comando solo la funzione paste("data","SOVRAPP_LC-VAR.def.csv", sep="/")
##Q&A: che razza di oggetto e' data, dal momento che non stato specificato
nulla di particolare?
#Riferimenti: ?data.frame
#-----
# rapido esame della struttura dei dati
head(data)
str(data)
##Q&A: quante osservazioni ho per ciascuna specie?
```

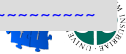

Lepri e gradienti altitudinali III

```
##Q&A: quante osservazioni ho per anno?  
52 ##Q&A: e combinando specie e anno?  
54 #Riferimenti: ?xtabs, ?formula  
56 #-----  
# Condizionamento dei dati  
58 ##Q&A: le variabili sono del 'tipo' corretto?  
#Specie  
50 #Comprensorio  
#Quota  
52 #Anno  
#Riferimenti: ?class, ?is, ?str  
54 # Sarebbe meglio se Specie e Comprensorio fossero variabili di tipo 'factor'  
anzich di tipo carattere  
56 ##Q&A: E' possibile fare di meglio, rispetto a usare delle sigle incomprensibili  
?  
58 # Una variabile di tipo factor possiede dei livelli, codificati con numeri  
interi  
# e a ciascun livello pu essere associata un'etichetta di comodo, che viene  
70 # utilizzata ad esempio nei titoli delle tabelle e nei grafici  
#Riferimenti: ?factor  
72 data$Specie <- factor(data$Specie, levels=c('lepcom', 'lepvar'), labels=c('L.  
europaeus', 'L. timidus'))  
74 data$Comprensorio <- factor(data$Comprensorio)
```



Lepri e gradienti altitudinali IV

```
76 # La variabile Comprensorio ha un numero di livelli non noto. Sarebbe meglio
76 # esaminare i valori esistenti e quindi ricodificarli con qualcosa di piu'
76 # comprensibile
78 # I comandi utilizzati sono presentati esclusivamente a titolo di esempio.
78 # In questo caso, e' comune in uno script R prima del blocco di codice commentato
78 # che
80 # funge da esempio, ma non e' essenziale ai fini della procedura di elaborazione,
80 # aggiungere un commento che tradizionalmente indica che il codice seguente e'
82 # appunto un esempio e non importante o necessario eseguirlo.
84 ## Not Run:
84 # summary(data$Comprensorio)
86 #> AV CH MO SO TI
86 #>157 94 218 314 394
88
88 data$Comprensorio <- factor(data$Comprensorio, levels=c("CH", "MO", "SO", "TI", "
88 AV"), labels=c("Val Chiavenna", "Morbegno", "Sondrio", "Tirano", "Alta Valle
88 "))
90
90 ## Not Run:
92 # summary(data$Comprensorio)
94 ##Q&A Il 'cambio di etichette' dovrebbe dare un risultato migliore ad esempio
94 ##Q&A nel caso delle tabulazioni...
96
96 ## Not Run:
98 # xtabs(~ Specie + Comprensorio, data)
100 #
```



Lepri e gradienti altitudinali V

```
02 # Analisi esplorativa: le quote medie almeno 'sembrano' diverse o no?
03 # si potrebbe pensare che sia necessario 'aggregare' le osservazioni in modo
04 # da avere un 'totale' per ciascuna quota. A pensarci bene, e' fondamentale
05 # tempo perso: e' molto piu' pratico lavorare sui dati grezzi!

06 # Ecco un esempio di inutile aggregazione
07 ## Not Run:
08 ## (data.freq <- data.frame(xtabs(~Quota+Specie, data)))

10 # il sistema piu' rapido per visualizzare 'le medie' di piu' 'gruppi' e' il box-
11 # and-whisker plot
12 boxplot(Quota ~ Specie, data)

14 # Il package lattice e' in grado di produrre grafici piu' complessi e in modo piu
15 # comodo.
16 # Lattice usa un'interfaccia a formula!
17 # In lattice, il comando equivalente a boxplot e' bwplot
18 bwplot(Quota ~ Specie, data)

19 ##Q&A, @TODO perche' il comando bwplot non funziona? E' presente ed e' stato
20 # caricato il package lattice?

21 # ggplot2 e' ancora meglio, da un punto di vista estetico
22 # GGplot2 si basa sulla metodologia creata da Leland Wilkinson nota come "
23 # grammar of Graphics"
24 ggplot(data, aes(x=Specie, y=Quota)) + geom_boxplot()

25 ##Q&A, @TODO E' stata caricata la libreria ggplot2?
```



Lepri e gradienti altitudinali VI

```
26 # Un altro modo di visualizzare i dati, utilizzando la grafica a pannelli di
    lattice
    histogram(~Quota | Specie, data=data, type="count", layout=c(1,2))
28 histogram(~Quota | Comprensorio + Specie, data=data, type="count")
    ##@TODO provare ad invertire l'ordine dei fattori nel comando precedente
30 histogram(~Quota | Anno + Specie, data=data, type="count")

32 # in ggplot2
    ggplot(data, aes(x=Quota)) + geom_histogram() + facet_grid(Specie ~ .)
34 ggplot(data, aes(x=Quota)) + geom_histogram() + facet_grid(Specie ~ Comprensorio)

36 # Un altro modo di visualizzare i dati, utilizzando uno specifico grafico
    # ('violin plot')
38 require(vioplot)
    vioplot(data[data$Specie == "L. europaeus", "Quota"], ##@Q&A: che significano quei
        nomi di variabile racchiusi tra quadre?
        data[data$Specie == "L. timidus", "Quota"],
        names= c('L. europaeus', 'L. timidus'), # specifica i nomi degli
            assi
        col='lightgray', # (quasi) nessuna rivista
            pubblica grafici a colori!
        ylim=c(0,3000) # forziamo la scala dell'
            asse Y su un intervallo 'comodo'
44 )

46 -----
48 # Per avere un'evidenza statistica delle differenze, occorre eseguire un test
    # sulla differenza tra le medie.
```

Lepri e gradienti altitudinali VII

```
50 # Ad esempio, il t di Student, che pero' richiede che le variabili oggetto del
51 # test abbiano distribuzione normale.
52 # Prima ancora - quindi - occorrer effettuare un test per la normalita' (ad
53 # esempio il test di Shapiro-Wilks).
54 #Riferimenti: ?shapiro.test
shapiro.test(data[data$Specie == "L. europaeus", "Quota"])
56 shapiro.test(data[data$Specie == "L. timidus", "Quota"])
##Q&A: sono distribuite normalmente? Indizio: cosa e' possibile capire dall'
58 ##Q&A: esempio nella pagina di manuale di shapiro.test?

50 # Un'alternativa (che non e' un test statistico) in forma grafica: il quantile-
51 # quantile plot mette in grafico i valori di una data variabile rispetto ai
52 # valori che quella variabile avrebbe se fosse normale.
53 # Riferimenti: ?qqnorm, ?qqplot, ?qline
54 qqnorm(data[data$Specie == "L. europaeus", "Quota"])
qqline(data[data$Specie == "L. europaeus", "Quota"], col='red')
56 qqnorm(data[data$Specie == "L. timidus", "Quota"])
qqline(data[data$Specie == "L. timidus", "Quota"], col='red')
58
##Q&A: c'e' per caso qualche ripetizione?

70 # Creare del codice contenente ripetizioni e' fondamentalemente una tecnica
suicida,
72 # in particolare nel caso in cui si debbano fare in seguito delle modifiche.
# La soluzione intelligente e' quella di creare una _funzione_, per poi
utilizzarla
74 # tutte le volte che si vuole.

76 myqqplot <- function(x) {
```



Lepri e gradienti altitudinali VIII

```
78 qqnorm(x)
   qqline(x, col='red')
}
30 myqqplot(data[data$Specie == "L. europaeus", "Quota"])
32 myqqplot(data[data$Specie == "L. timidus", "Quota"])
34 # c'e' ancora qualche ripetizione...
   lapply(split(data$Quota, data$Specie), myqqplot)
36
38 ##Q&A: E' possibile verificare la differenza tra quote con un t-test?
   # Riferimenti: ?t.test
90
## Not Run:
##t.test(Quota ~ Specie, data)
```



Chirotteri e *optimal foraging*

Script file: src\example_10_distance.R

Dataset: src\data\dist_tab.dbf

Descrizione: distanze percorse tra sito di ricovero diurno (*roost*) e aree di foraggiamento da femmine di *Myotis emarginatus* in differente stato riproduttivo.



Obiettivo

Le femmine che si sono riprodotte dovrebbero avere dei costi energetici maggiori, e quindi sfrutterebbero aree di foraggiamento più vicine al *roost*.



Chirotteri e optimal foraging I

File: src/example_10_distance.R

```
1 # 'pulizia' del workspace
  rm(list = ls())
3
4 # I dati sono in formato dBASE IV (.dbf). Occorre il package foreign per poterli
5 # caricare
  require(foreign)
7
8 #require(graphics)
9 #require(lattice)
10 #require(nlme)
11
12 # definizione della directory di lavoro
13 workdir <- "/data/Didattica/R/R_2013/src/"
14
15 # nome del file di dati
16 datafile <- "data/dist_tab.dbf"
17
18 # attiva la directory di lavoro, carica i dati
19 setwd(workdir)
20 dist_tab <- read.dbf(datafile)
21
22 ##@TODO: il condizionamento dei dati tutto da rifare!
23 # Data dictionary per il file dist_tab.dbf
24 # SRIP: stato riproduttivo: 0 -> non riproduttiva; 1 -> riproduttiva
25 # Riferimenti: ?factor
```



Chiotteri e optimal foraging II

```
## Not Run:
## dist_tab$SRIP <- factor(dist_tab$SRIP, levels=c(0,1), labels=c('non
  riproduttiva','riproduttiva'))
# pi opportuno dare alle variabili dei nomi pi comprensibili:
# Riproduzione, Mese, Individuo, Distanza
# Riferimenti: ?names
## Not Run:
## names(dist_tab) <- c('Riproduzione', 'Mese', 'Individuo', 'Distanza')

##@TODO: esplorazione dei dati: quali sono i fattori che possono influenzare la
##@TODO: distanza tra roost e aree di foraggiamento?

##Q&A: L'analisi della varianza richiede che la variabile responso sia
##Q&A: distribuita normalmente...
## Not Run:
## shapiro.test(dist_tab$Distanza)

#####
## Analisi della varianza
# modello saturo
aov.0 <- aov(Distanza ~ Riproduzione * Mese, data=dist_tab)
summary(aov.0)
##Q&A: possibile escludere il fattore 'Mese', per aumentare la sensibilit
##Q&A: del test su 'Riproduzione'?
## Not Run:
## aov.1 <- aov(Distanza ~ Riproduzione, data=dist_tab)
## summary(aov.1)
##Q&A: Riproduzione ha un effetto significativo?
##Q&A: Se s , occorrerebbe effettuare un test post-hoc...
```



Chirotteri e optimal foraging III

```
# Riferimenti: TukeyHSD
55 ## Not Run:
## TukeyHSD(aov.1)

57 #####
59 ## Ma quali sono le distanze medie?
# Occorre calcolare la media per i due gruppi (Riproduttive e non)
51 # Due possibili soluzioni:
# 1) metodo a forza bruta: estrarre i dati dal dataframe e calcolare media e
    deviazione standard
53 dist_tab.R <- dist_tab[dist_tab$Riproduzione == 'riproduttiva',]
mean(dist_tab$Distanza)
55 sd(dist_tab$Distanza)

57 dist_tab.R <- dist_tab[dist_tab$Riproduzione == 'non riproduttiva',]
mean(dist_tab$Distanza)
59 sd(dist_tab$Distanza)

71 # 2) approccio un po' pi' intelligente, facciamo gestire a R le 'ripetizioni'
# Riferimenti: ?by
73 by(dist_tab$Distanza, dist_tab$Riproduzione, summary)

75 ## End of File ##
```



Bilancio energetico in *Emys orbicularis* e *Trachemys scripta*

Script file: example_11_terrapin.R

Dataset: terrapin_vs_slider.txt

Descrizione: esperimenti di termoregolazione controllata in *Emys orbicularis* e *Trachemys scripta*.

Obiettivo

Esistono delle differenze nell'efficienza della termoregolazione che possono dare un vantaggio alla specie alloctona di tartaruga *Trachemys scripta* nella competizione con la testuggine palustre europea *Emys orbicularis*?



Bilancio energetico in testuggini autoctone e alloctone I

File: src/example_11_terrapi.R

```
1 # 'pulizia' workspace
  rm(list=ls())
3 ##@TODO modificare rispetto alla propria directory di lavoro
  workdir <- "/data/Didattica/R/R_2013/src/"
5 setwd(workdir)

7 # caricamento dati
  data <- read.table('data/terrapi_vs_slider.txt', header=TRUE)
9

11 #####
12 ## Nota
13 ## I comandi utilizzati per le varie fasi dell'analisi sono il pi delle volte
14 ## commentati (Not Run:). Confrontare ciascun comando con le rispettive pagine
15 ## di manuale, e, una volta compreso il funzionamento del comando, togliere i
16 ## caratteri di commento ed eseguire il comando.
17 ## anche possibile trovare delle soluzioni alternative...

19 ##Q&A: Le misure di temperatura sono state effettuate ogni 15'.
20 ##Q&A: Che cosa c' che non va nella variabile data$tempo e com' possibile
  sistemarlo?
21 # Soluzione: tempo stato codificato come numero sequenziale, che si riferisce
22 # a intervalli successivi di 15'. sufficiente moltiplicare per 15!
```

Bilancio energetico in testuggini autoctone e alloctone II

```
25 timespan <- 15 # minutes
26 data$tempo <- data$tempo * timespan
27
28
29 ##Q&A: Come varia la temperatura in funzione del tempo?
30 ##Q&A: possibile visualizzare le curve di temperatura separatamente per le due
31 specie?
32 # Soluzione: quando si tratta di 'ripetere lo stesso grafico' in funzione di una
33 # variabile fattoriale (in questo caso data$specie), il caso di usare lattice.
34 require(lattice)
35 xyplot(temperatura~tempo, group=specie, type=c('r','p'), data, auto.key=TRUE)
36 bwplot(temperatura~tempo | specie, data, horizontal=FALSE)
37
38 # Soluzione (2): utilizzare un conditioning plot, facendo passare una polinomiale
39 # per i punti sperimentali
40 # Riferimenti: ?coplot, ?panel.smooth
41 ## Not Run:
42 ## coplot(temperatura ~ tempo | specie, data, panel=panel.smooth)
43 ## coplot(temperatura ~ tempo | specie * sesso, data, panel=panel.smooth)
44
45 # Soluzione (3): panel plot in ggplot2, facendo passare una polinomiale
46 # per i punti sperimentali
47 # Riferimenti: ?ggplot, ?stat_smooth
48 require(ggplot2)
49 require(scales)
50 plt <- ggplot(data, aes(x=tempo, y=temperatura, color=specie)) + geom_point()
```



Bilancio energetico in testuggini autoctone e alloctone III

```
plt + facet_grid(specie ~ .)
plt + facet_grid(. ~ specie)
plt + facet_grid(specie ~ sesso)
plt + facet_grid(specie ~ sesso) + stat_smooth(method='lm')
plt + facet_grid(specie ~ sesso) + stat_smooth(method='loess')

##Q&A: Qual il modello pi adatto per rappresentare la variazione di
      temperatura?
##Q&A: (In altri termini: sufficiente una retta?)
# Riferimenti: ?lm (regressione lineare), ?nls (regressione non lineare)

##Q&A: Le analisi vanno ripetute due volta, una per ciascuna specie.
##Q&A: possibile identificare una soluzione al problema che utilizzi delle
##Q&A: funzioni?

#####
## Soluzione 1 - regressione lineare
# non viene utilizzata una funzione, ma un approccio iterativo mediante una
# iterazione finita (comando 'for').
# Riferimenti: ?Control

# Il comando for deve avere una variabile iterabile_ (un vettore, una lista, ecc
# in base alla quale ripetere di volta in volta.
# Creiamo un elenco delle specie.
# Riferimenti: ?unique, ?as.character
```



Bilancio energetico in testuggini autoctone e alloctone IV

```
75 lista.specie <- as.character(unique(data$specie))
77 # Se eseguiamo le analisi in modo iterativo, non è possibile accedere
78 # ai risultati se non quando il ciclo for è terminato.
79 # Occorre quindi creare un 'contenitore' che riceva i dati mentre il ciclo è in
80 # funzione, in modo da poterli consultare in seguito.
81 # Creiamo una lista, vuota (per ora!)
82 # Riferimenti: ?list
83 results <- list()
85 # ciclo for sugli elementi di lista.specie per il calcolo della regressione
86 # lineare
87 for (s in lista.specie) {
88     d <- subset(data, subset==specie==s) # ripetere per ciascuna specie
89     # 's' in un dataframe # estrae i dati relativi alla specie
90     results[[s]] <- lm(temperatura ~ tempo, d) # esegue la regressione lineare, e
91     # immagazzina il risultato
92 }
93 # un pò meglio, usando lapply, senza 'for'
94 splist <- split(data, data$specie)
95 results2 <- lapply(splist, function(x) lm(x$temperatura ~ x$tempo))
96 ##Q&A: Che cosa contengono adesso gli oggetti results e results2?
97 # Riferimenti: ?str, ?names
98 # Nota: per accedere a un elemento di una lista, si utilizza la notazione con
99 # le doppie parentesi quadre (vedi riga 89 sopra)
```



Bilancio energetico in testuggini autoctone e alloctone V

```
09 ##Q&A: Come si fa a vedere i risultati delle regressioni?
01 # Riferimenti: ?summary, ?plot

03 ## Not Run:
04 ## visualizza i risultati delle regressioni
05 # summary(results[['trachemys']])
06 # summary(results[['emys']])

07
08 ## Not Run:
09 ## visualizza i risultati delle regressioni
10 # plot(results[['trachemys']])
11 # plot(results[['emys']])

12
13 #####
14 ## Soluzione 2 – regressione non lineare
15 # Modello: esponenziale negativa
16 # In questo caso, all'interno del ciclo sono stati inseriti comandi per la
17 # visualizzazione immediata dei risultati.
18 results.nls <- list()
19 for (s in lista.specie) {
20   d <- subset(data, subset==specie==s)
21   # regressione non lineare (notare i 'guess values' per i parametri del modello)
22   results.nls[[s]] <- nls(temperatura ~ Tmin + Tdiff * (1 - exp(-(k/100)*tempo)),
23     d, start=list(Tmin=19, Tdiff=5, k=1))

```



Bilancio energetico in testuggini autoctone e alloctone VI

```
23 cat("
    n", "Non linear regression for id = ", s, "\n\n")
24 print(summary(results.nls[[s]]))
25 cat("\n\n")
26 }
27
28 ## stima dei parametri: viene eseguita una regressione non lineare per ciascun
    animale
29 ## i coefficienti e i valori dei parametri vengono immagazzinati in un dataframe
    ## creato in precedenza
30
31 # struttura per immagazzinare i risultati
32 animali <- unique(data[,c('id', 'specie', 'sesso')])
33 animali$AIC <- NA # Akaike Information Criterion
34 animali$Tmin <- NA # parametro: T iniziale
35 animali$Tmax <- NA # parametro: delta T
36 animali$Sk <- NA # parametro: tasso di riscaldamento
37
38 # calcolo iterativo delle regressioni
39 for (b in animali$id) {
40   d <- subset(data, subset=id==b)
41   r <- nls(temperatura ~ Tmin + Tdiff * (1 - exp(-(k/100)*tempo)), d, start=list(
42     Tmin=19, Tdiff=5, k=1))
43   cat("
    n", "Non linear regression for id = ", b, "\n\n")
```



Bilancio energetico in testuggini autoctone e alloctone VII

```
print(summary(r))
45 cat("\n\n")
   animali[animali$id == b,]$AIC <- AIC(r)
47   animali[animali$id == b,]$Tmin <- coef(r)[[ 'Tmin' ]]
   animali[animali$id == b,]$Tmax <- coef(r)[[ 'Tdiff' ]] + coef(r)[[ 'Tmin' ]]
49   animali[animali$id == b,]$k <- coef(r)[[ 'k' ]]
}

51 # analisi della varianza: k      differente tra specie?
53 cat("
      n", "ANOVA for k (negative exponential model)\n\n")
aov.k <- aov(k ~ specie * sesso, animali, family='gaussian')
55 summary(aov.k)
TukeyHSD(aov.k)
57 bwplot(k ~ specie, animali)

59 # analisi della varianza: Tmax   differente tra specie?
61 cat("
      n", "ANOVA for k (negative exponential model)\n\n")
aov.Tmax <- aov(Tmax ~ specie * sesso, animali)
63 summary(aov.Tmax)
TukeyHSD(aov.Tmax)
65 bwplot(Tmax ~ sesso | specie, animali)
## End of File ##
```



Predatori di semi, *pulsed resource systems* e cambiamenti climatici

Script file: `example_12_cones.R`

Dataset: `Pigne.xlsx`, `Pigne2.xls`

Descrizione: produttività di diverse specie di conifere in diverse aree di studio (Valle d'Aosta e Valtellina).

Obiettivo

Possono verificarsi degli anni di “sovrapproduzione” (*mast* o *pasciona*) di semi: è un meccanismo evolutivo (*predator satiation hypothesis*), ma è possibile che gli anni di “pasciona” dipendano anche da fattori climatici: questo potrebbe essere vero se in aree diverse e specie vegetali diverse gli anni in cui avviene sovrapproduzione sono sincroni.



Predatori di semi, *pulsed resource systems* e cambiamenti climatici I

File: src/example_12_cones.R

```
1 #####  
2 ### Testing conifer cones value acrosss areas and years ###  
3 #####  
4 #Q: cones production of different conifers species has been registered in several  
5 sites and in consecutive years. Cones production change across years... why  
6 ?  
7 #1. Seed predator or 2.Climate change? If climate change influence cones  
8 production is there any correlation between species , years and different  
9 areas?  
10 # 'clean' workspace  
11 rm(list=ls())  
12 #####WORKING ENVIRONMENT SETTINGS#####  
13 ##@TODO change according to your workdir  
14 workdir <- "/data/Didattica/R/R_2014/src/data"  
15 setwd(workdir)  
16 #####ATTENZIONE: i pacchetti che permettono di leggere e gestire i file xls/xlsx si  
17 possono appoggiare ad altri programmi che devono essere presenti sulla  
18 vostra macchina. la funziona read.xls del pacchetto gdata utilizza il  
19 programma perl. Se utilizzate SO Windows dovete specificare al comando dove  
20 risiede Perl. Se il vostro SO ? UNIX non ? necessario  
21 #perl="C:/Perl64/bin/perl.exe"##it is necessary to read xls files , at least on  
22 Windows...
```

Predatori di semi, *pulsed resource systems* e cambiamenti climatici II

```
L7 #####PACKAGES#####
L9 library(gdata)##read xls
L9 library(ggplot2)## graphics package
L9 library(reshape2)## modify dataset
L9 #library(XLConnect)## read and write xls and xlsx (it needs java)
L23 #####
L25 #####LOADING DATA#####
L27 #####
L29 #####CONE PER PLOT#####
L29 xlsxfile <- 'Pigne.xlsx'
L29 #vegx <- read.xls(xlsxfile , sheet=1, stringsAsFactors=FALSE, perl=perl)## su
L29 windows
L31 vegx <- read.xls(paste(workdir , 'data ' ,xlsxfile , sep='/') , sheet=1,
L31 stringsAsFactors=FALSE)
L33 ### have a look of data
L35 str(vegx)
L35 head(vegx)
L37 ##### FIXING DATA FOR ANALYSIS #####
L39 vegx[vegx$tree.species%in% 'P. abies' ,]$tree.species <- 'P. abies'
L39 vegx[vegx$Area%in% 'CEDRASCO' ,]$Area <- 'CED'
```



Predatori di semi, *pulsed resource systems* e cambiamenti climatici III

```
vegx$cones <- as.integer(vegx$cones) #number of cones is a character variable, but
  we need it as numeric
41 vegx <- vegx[!is.na(vegx$cones),] #dropping cones NaN values,
  ##ATTENTION: be careful at the differences between 0 (zero) and NA!
43
45 #####
  ####folding data to create proper dataframe#####
  #####
47
  ## we need to have mean, standard deviation and CV of number of cones
49 vegmean <- aggregate(cones~Area+year+tree.species, vegx, FUN=mean)
  names(vegmean) <- c("Area", "year", "tree.species", "mean")
51 vegsd <- aggregate(cones~Area+year+tree.species, vegx, FUN=sd)
  names(vegsd) <- c("Area", "year", "tree.species", "sd")
53 vegtot <- cbind(vegmean, vegsd$sd)
  names(vegtot) <- c("Area", "year", "tree.species", "mean", 'sd')
55 vegtot$cv <- vegtot[, "mean"] / vegtot[, "sd"]
57
  #####
  ##can we optimize the code above?#####
59 ##### TRY #####
  #vegtot <- setNames(aggregate(cones~Area+year+tree.species, vegx, FUN= function(x)
    c(mean(x), sd(x))), c("Area", "year", "tree.species", "mean", 'sd'))
51 #names(vegtot) <- c("Area", "year", "tree.species", "mean", 'sd')
53
  ## as usual while your are running your experiments you get new data and most of
  the time format is different...
```



Predatori di semi, *pulsed resource systems* e cambiamenti climatici IV

```
#####  
55 ##### load other data #####  
#####  
57 xlsvda <- 'Pigne2.xls'  
###read in from the same file two different sheets ###  
59 #rhemes <- read.xls(paste(workdir, 'data', xlsvda, sep='/'), sheet='Rhemes',  
  stringsAsFactors=FALSE, perl=perl) ##windows  
rhemes <- read.xls(paste(workdir, 'data', xlsvda, sep='/'), sheet='Rhemes',  
  stringsAsFactors=FALSE)  
71 #cogne <- read.xls(paste(workdir, 'data', xlsvda, sep='/'), sheet='Cogne',  
  stringsAsFactors=FALSE, perl=perl) ##windows  
cogne <- read.xls(paste(workdir, 'data', xlsvda, sep='/'), sheet='Cogne',  
  stringsAsFactors=FALSE)  
73  
##have a look of the new dataset ...  
75 str(rhemes)  
str(cogne)  
77  
##obviously format is different ...  
79 ##FIX the new dataset according to previous dataframe  
vdarh <- melt(rhemes, id=c("Area", "Area.code", "ID", 'Trap', 'Tree.species'),  
  variable.name = "year")## metl dataset according to previous dataframe  
81 vdarh$year <- substring(as.character(vdarh$year), 2,5)##something strange with  
year!  
83  
vdaco <- melt(cogne, id=c("Area", "Area.code", "ID", 'Trap', 'Tree.species'),  
  variable.name = "year")## melt dataset according to previous dataframe
```



Predatori di semi, *pulsed resource systems* e cambiamenti climatici V

```
vdaco$year <- substring(as.character(vdaco$year),2,5)##something strange with
year!
35
vda <- rbind(vdarh,vdaco)##join of the two dataframe
37 names(vda) <- c('Area', "Area.code","ID", 'Trap', 'tree.species', 'year', 'cones')
vdamean <- aggregate(cones~Area+year+tree.species ,vda ,FUN=mean)
39 names(vdamean) <- c("Area", "year", "tree.species", "mean")
vdasd <- aggregate(cones~Area+year+tree.species ,vda ,FUN=sd)
41 names(vdasd) <- c("Area", "year", "tree.species", "sd")
vdatot <- cbind(vdamean, vdasd$sd)
43 names(vdatot) <- c("Area", "year", "tree.species", "mean", 'sd')
vdatot$cv <- vdatot[, "mean"] / vdatot[, "sd"]
45
###what if factor levels are different?###
47 vdatot[vdatot$tree.species%in%'abete',]$tree.species <- 'P. abies'
vdatot[vdatot$tree.species%in%'Abete',]$tree.species <- 'P. abies'
49 vdatot[vdatot$tree.species%in%'larice',]$tree.species <- 'Larix'
vdatot[vdatot$tree.species%in%'Larice',]$tree.species <- 'Larix'
51 vdatot$year <- as.factor(vdatot$year)
53
###Finally the complete dataframe!###
vegetation <- rbind(vegtot, vdatot)
55
## Q: Is any correlation between cones production and area+species####
57 ##prepare data for correlation analysis
vegetation$tree.species <- as.factor(vegetation$tree.species)
59
```



Predatori di semi, *pulsed resource systems* e cambiamenti climatici VI

```
##dcast stretch dataframe
12 zz <- dcast(vegetation, year ~ Area+tree.species, value.var='mean', fun.aggregate=
    mean)
##have a look of the matrix
13 str(zz)
14 head(zz)
##correlation test
15 tt <- cor(zz[,2:19], use="pairwise.complete.obs")
16
17 #####
18 ###write matrix to file#####
19 #####
20 ##package XLConnect it is useful not just to read but also to write matrix
##where does it write it?
21 writeWorksheetToFile('cor_cones_speciesXarea.xls', tt, 'spec_vs_area',
    styleAction = XLC$STYLE_ACTION.XLCONNECT)
22
23 writeWorksheetToFile('matrice_pigne.xls', zz, 'spec_x_area', styleAction = XLC$
    STYLE_ACTION.XLCONNECT)
24
25 #####
26 ###some useful graphics###
27 #####
28 vegetation$year <- as.factor(vegetation$year)
29
30 ##base plot
```



Predatori di semi, *pulsed resource systems* e cambiamenti climatici VII

```
33 plot.0 <- ggplot(vegetation , aes(x=year , y=mean , fill=tree.species)) + facet_wrap(~
tree.species , scale='free_y') + geom_bar(stat = "identity")
34 plot.0
35
36 ##Fix axes text and rearrange scale
37 plot.1 <- plot.0 + theme(axis.text.x = element_text(angle = 90 , hjust = 0.5))
38 plot.1
39
40 ##Fix axes labes
41 plot.2 <- plot.1 + ylab('Number of Cones') + xlab('Year')
42 plot.2
43
44 ##Remove legend
45 plot.3 <- plot.2 + theme(legend.position="none")
46 plot.3
47
48 ##changing plot placement
49 plot.4 <- plot.3 + facet_wrap(~tree.species , scale='free_y' , ncol = 1)
50
51 ##underline mast year
52 plot.5 <- plot.4 + geom_vline(xintercept=as.numeric(vegetation$year[c(1,3,6,8,45)
53 ])) , linetype = "longdash")
54 plot.5
55
56 ##add title and print out graphic choosing dimension and resolution
57 plot <- plot.5 + ggtitle('MAST')
png(filename='PLOT.png' , width=15 , height=20 , units='cm' , res=300)
```



Predatori di semi, *pulsed resource systems* e cambiamenti climatici VIII

```
plot  
dev.off()
```

