

INTRODUCTION TO R

E. Fersini

INTRODUCTION

- Why R?



R is a **programming** and statistical **language**.



R is useful for **machine learning**, data analysis and **visualization**.



R is **simple** and easy to learn, read and write.



R is **open source**.



R has an IDE, **RStudio**.

INSTALLING R

- If you want to install R on your system:
 - Step 1 : Go to the link- <https://cran.r-project.org/>
 - Step 2 : Download and install R on your system.

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

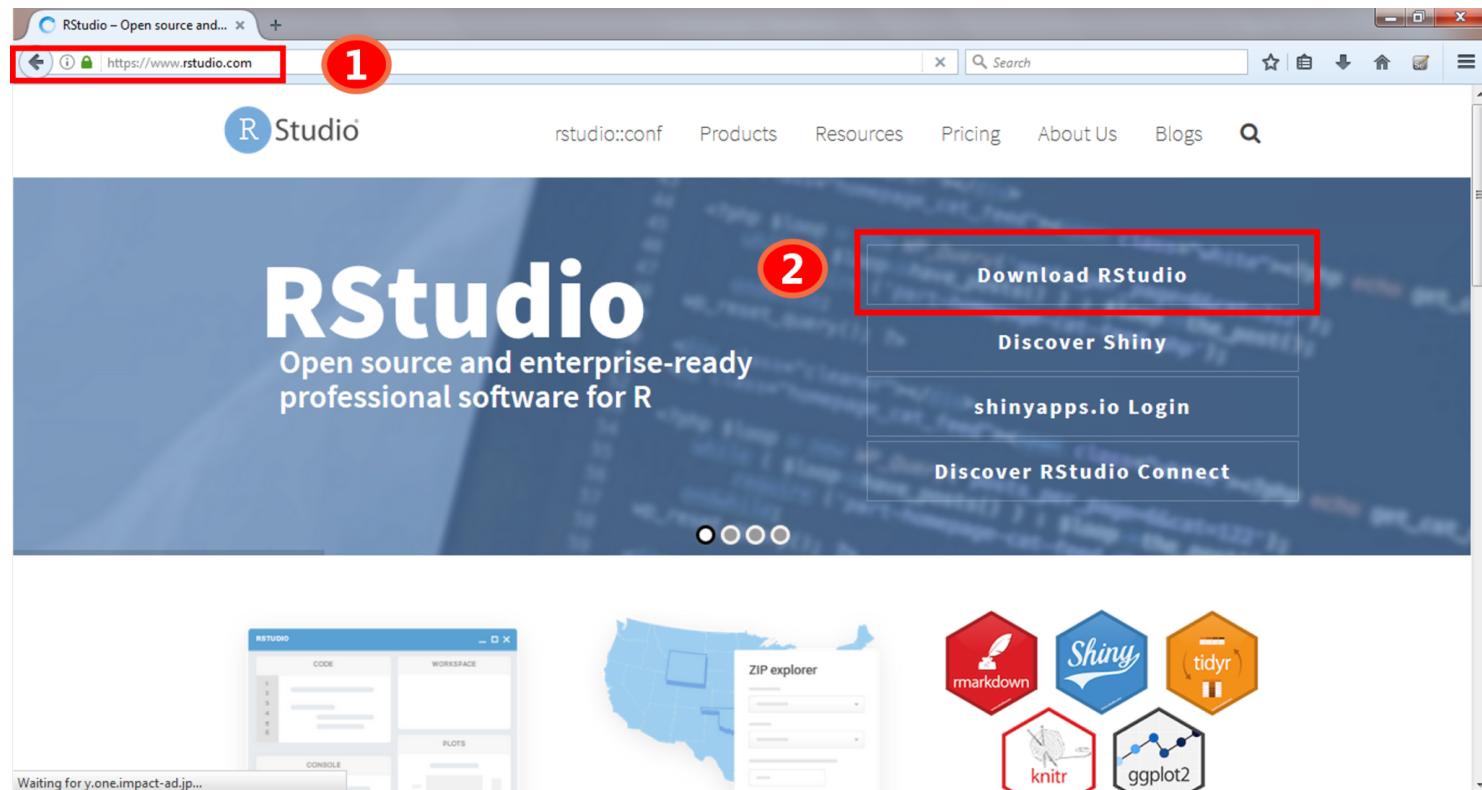
Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (Monday 2017-03-06, Another Canoe) [R-3.3.3.tar.gz](#), read [what's new](#) in the latest version.
- Sources of [R alpha and beta releases](#) (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are [available here](#). Please read about [new features and bug fixes](#) before filing corresponding feature requests or bug reports.
- Source code of older versions of R is [available here](#).
- Contributed extension [packages](#)

INSTALLING RStudio

- If you want to install RStudio on your system:
 - Step 1 : Go to the link <https://www.rstudio.com/>
 - Step 2 : Download and install RStudio on your system.



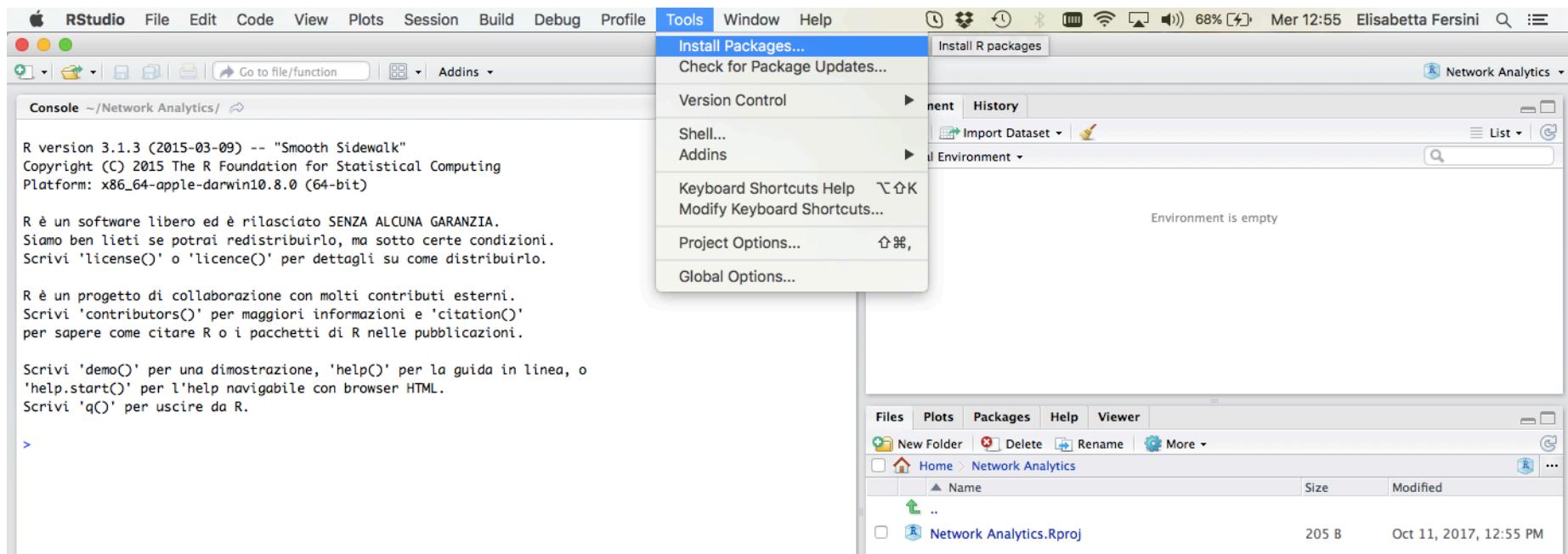
RStudio

The screenshot shows the RStudio interface with several panels:

- Environment Panel:** Shows the Global Environment tab with a list of objects.
- Console Panel:** Displays R startup messages and the current working directory (~ /Documents/R seminar series/2.1). It also shows the R version and license information.
- R Script Panel:** An empty script editor window titled "R Script".
- Help Panel:** A large orange box containing the text "Data Objects", "File ,Plots, Package installation and Help panels", and "CONSOLE (Script output panel)".

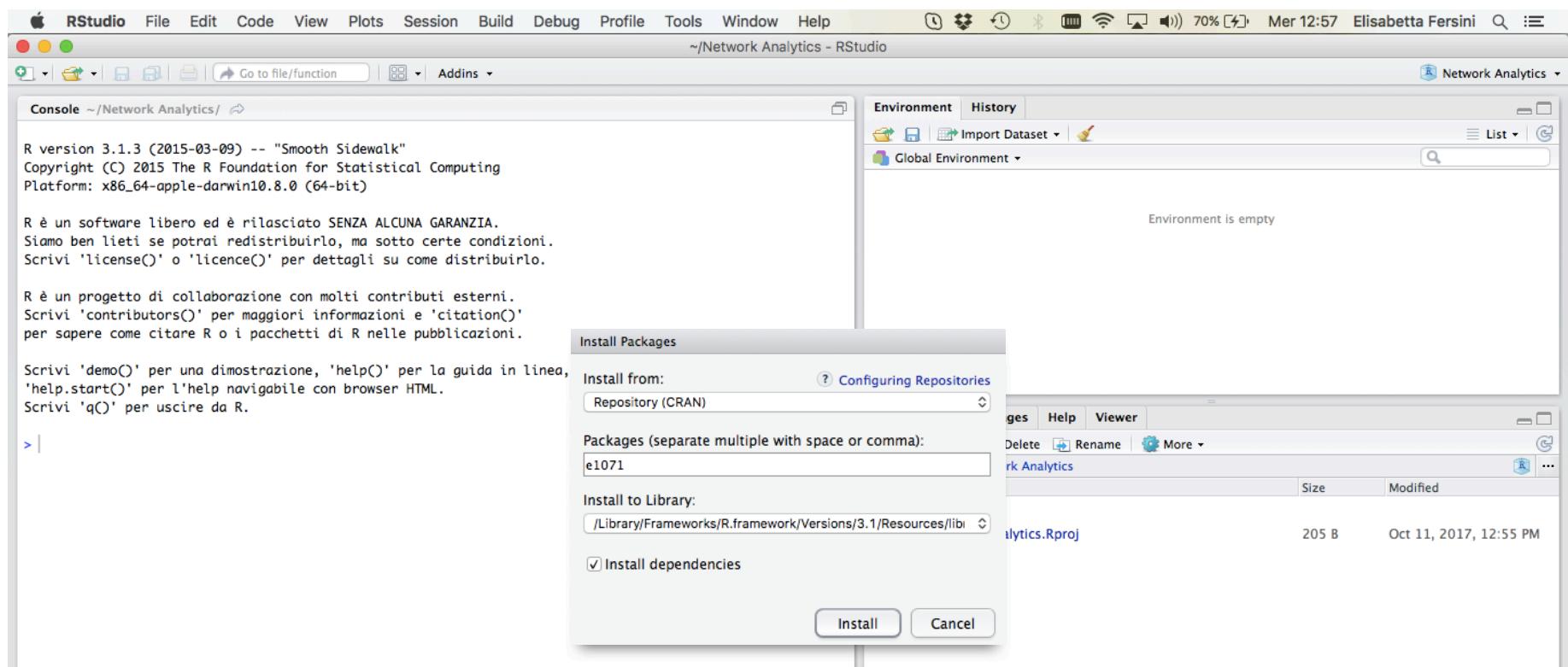
INSTALLING PACKAGES

- Any package that you want to use must be firstly installed:



INSTALLING PACKAGES

- Any package that you want to use must be firstly installed:



LOADING PACKAGES

- Once you have installed a package, you need to load it to use it:

```
library (e1071)
```

- If you would like to view the documentation of the package, you can use the help()function:

```
help(package ="e1071")
```

DATA OPERATORS IN R

- There are mainly 5 different types of operators:
 1. Arithmetic Operators: Perform arithmetic operations such as addition, subtraction, multiplication, division etc.
 2. Assignment Operators: Assignment operators are used to assign values.
 - Assignment Operators = <- -

```
x=5  
x  
[1] 5
```

```
x<-12  
x  
[1] 12
```

```
26 -x  
x  
[1] 26
```

DATA OPERATORS IN R

3. **Relational** Operator: It defines a relation between two entities

```
x=5  
x<9  
[1] TRUE
```

4. **Logical** Operator: Used often to verify conditions

```
2&3  
[1] TRUE  
  
2&!3  
[1] FALSE
```

DATA OPERATORS IN R

5. **Special** Operators: These operators are used for specific purpose, not for logical computation. Examples:
 - Creating the series of numbers in sequence for a vector

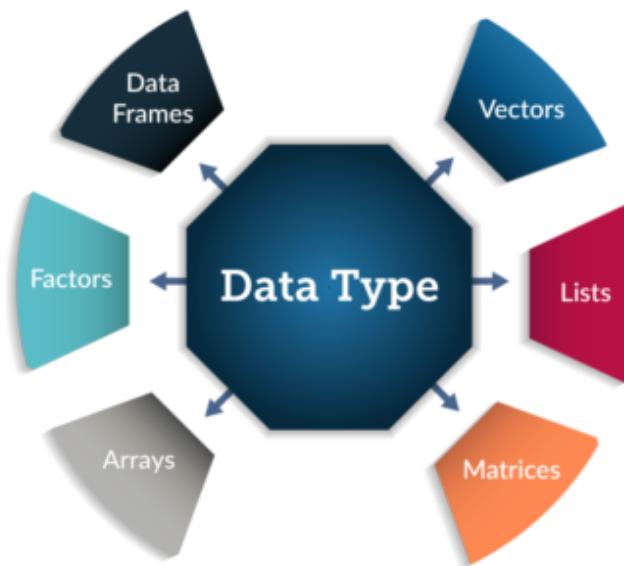
```
x=4:8  
x  
[1] 4 5 6 7 8
```

- `%in%` is used to identify if an element belongs to a vector

```
x=4:8  
y=6  
y%in%x  
[1] TRUE
```

DATA TYPES

- In R, we do not need to declare a variable as some data type. The variables are assigned with R-Objects and the data type of the R-object becomes the data type of the variable.
- There are mainly six data types present in R:



DATA TYPES

- **VECTOR:** It is a sequence of data elements of the **same basic type**

```
v=c(1,3,5,7)
```

- There are 5 classes of vectors:



Remark: in many versions numerical vectors are treated as double precision real numbers. To explicitly create integers, add a L at the end

```
v1=c(1L, 3L, 5L, 7L)
```

DATA TYPES

- **LIST:** Lists are the R objects which contain **elements of different types** like – numbers, strings, vectors and another list inside it.

```
>n = c(2, 3, 5)    ←-----  
Vector of Double  
>s = c("aa", "bb", "cc", "dd", "ee") ←-----  
Vector of Characters  
>x = list(n, s, TRUE)  
>x  
[[1]]  
[1] 2 3 5  
[[2]]  
[1] "aa" "bb" "cc" "dd" "ee"  
[[3]]  
[1] TRUE
```

←-----
List consisting of:
[[1]] vector of double
[[2]] vector of characters
[[3]] boolean

DATA TYPES

- **MATRIX:** Matrices are the R objects in which the elements are arranged in a two-dimensional rectangular layout. A Matrix is created using the matrix() function.
 - Example: `matrix(data, nrow, ncol, byrow, dimnames)` where,

```
Mat <- matrix(c(1:16), nrow=4, ncol= 4)
Mat
 [,1] [,2] [,3] [,4]
[1,] 1    5    9   13
[2,] 2    6    10  14
[3,] 3    7    11  15
[4,] 4    8    12  16
```

DATA TYPES

- **ARRAY:** Arrays are the R data objects which can store data in **more than two dimensions**. It takes vectors as input and uses the *dim* parameter to create an array.

```
>vector1= c(5,9,3)
```

```
>vector2 = c(10,11,12,13,14,15)
```

```
result = array(c(vector1,vector2),dim= c(3,3,2))
```

```
..1  
[,1] [,2] [,3]
```

```
[1,] 5 10 13
```

```
[2,] 9 11 14
```

```
[3,] 3 12 15
```

```
..2  
[,1] [,2] [,3]
```

```
[1,] 5 10 13
```

```
[2,] 9 11 14
```

```
[3,] 3 12 15
```

DATA TYPES

- **FACTORS:** Factors are the data objects which are **used to categorize the data** and **store it as levels**. They can store both strings and integers. They are useful in data analysis for statistical modeling.

```
c("East","West","East","North","North","East","West","West","East")
>factor_data <- factor(data)
>factor_data
[1] East West East North North East West West East
Levels: East North West
```

DATA TYPES

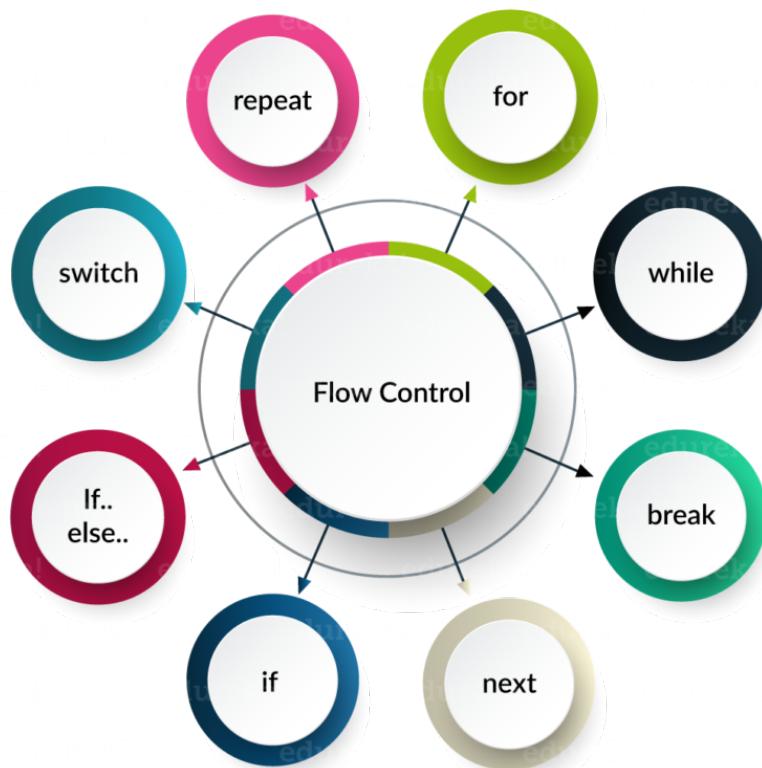
- **DATA FRAME**: A data frame is a **table** in which each column contains values of one variable and each row contains one set of values from each column.

```
>std_id = c (1:5)
>std_name = c("Rick","Dan","Michelle","Ryan","Gary")
>marks = c(623.3,515.2,611.0,729.0,843.25)
>std.data <- data.frame(std_id, std_name, marks)
>std.data
```

std_id	std	name	marks
1	1	Rick	623.30
2	2	Dan	515.20
3	3	Michelle	611.00
4	4	Ryan	729.00
5	5	Gary	843.25

FLOW CONTROL

- **Flow control statements** play an important role as they allow you to control the flow of execution of a script inside a function.



FLOW CONTROL

- **IF, ELSE and ELSEIF:** These control statement **evaluate** a single/multiple **condition(s)**.

```
x=5  
if(x>0){  
    print ("Positive number")  
}
```

```
x = -5  
if(x>0){print ("Positive number")}  
else {print ("Negative number")}
```

```
x = -5  
if(x>0){print ("Positive number")}  
else if (x<0) {print ("Negative number")}  
else print("Zero")
```

FLOW CONTROL

- **SWITCH:** These control statements are basically used to **compare** a certain expression to a known value.

```
vtr <- c(150,200,250,300,350,400)
option <-"mean"
switch(option,
      "mean" = print(mean(vtr)),
      "mode" = print(mode((vtr))),
      "median" = print(median((vtr)))
)
```

FLOW CONTROL

- **REPEAT**: The repeat **loop** helps to execute the same set of code again and again until a stop **condition** is met at the **end**.
- **WHILE**: same **loop**, but the **condition** is checked at the **beginning**.

```
x=2
while(x<1000){
    x=x^2
    print(x)
}
```

FLOW CONTROL

- **FOR:** For loops are used when you need to **execute** a block of code several number of **times**.

```
vtr= c(7,19,25,65, 45)
for( i in vtr){
  print(i)
}
```

FLOW CONTROL

- **Break:** Break statements help to **terminate** program, loops and switch.

```
x = 1:5
for (val in x){
    if (val == 3){
        break
    }
    print(val)
}
```

FLOW CONTROL

- **NEXT:** A next statement is used when you want to skip the current iteration of the loop without terminating it.

```
for(i in 1:15){  
  if((i%2)==0){  
    next  
  }  
  print(i)  
}
```

SOME USEFUL FUNCTIONS

- **NUMERIC** functions

Function	Description
abs(x)	absolute value
sqrt(x)	square root
ceiling(x)	ceiling(3.475) is 4
floor(x)	floor(3.475) is 3
trunc(x)	trunc(5.99) is 5
round(x, digits=n)	round(3.475, digits=2) is 3.48
signif(x, digits=n)	signif(3.475, digits=2) is 3.5
cos(x), sin(x), tan(x)	also acos(x), cosh(x), acosh(x), etc.
log(x)	natural logarithm
log10(x)	common logarithm
exp(x)	e^x

SOME USEFUL FUNCTIONS

- Functions on **STRINGS**

Function	Description
substr(x, start=n1, stop=n2)	Extract or replace substrings in a character vector. <code>x <- "abcdef"</code> <code>substr(x, 2, 4) is "bcd"</code> <code>substr(x, 2, 4) <- "22222" is "a222ef"</code>
grep(pattern, x , ignore.case=FALSE, fixed=FALSE)	Search for <i>pattern</i> in <i>x</i> . If <i>fixed</i> =FALSE then <i>pattern</i> is a regular expression . If <i>fixed</i> =TRUE then <i>pattern</i> is a text string. Returns matching indices. <code>grep("A", c("b","A","c"), fixed=TRUE) returns 2</code>
sub(pattern, replacement, x, ignore.case =FALSE, fixed=FALSE)	Find <i>pattern</i> in <i>x</i> and replace with <i>replacement</i> text. If <i>fixed</i> =FALSE then <i>pattern</i> is a regular expression. If <i>fixed</i> = T then <i>pattern</i> is a text string. <code>sub("\\s", ".", "Hello There") returns "Hello.There"</code>
strsplit(x, split)	Split the elements of character vector <i>x</i> at <i>split</i> . <code>strsplit("abc", "")</code> returns 3 element vector "a","b","c"
paste(..., sep="")	Concatenate strings after using <i>sep</i> string to seperate them. <code>paste("x",1:3,sep="")</code> returns <code>c("x1","x2" "x3")</code> <code>paste("x",1:3,sep="M")</code> returns <code>c("xM1","xM2" "xM3")</code> <code>paste("Today is", date())</code>
toupper(x)	Uppercase
tolower(x)	Lowercase

SOME USEFUL FUNCTIONS

- STATISTICAL functions

Function	Description
dnorm(x)	normal density function (by default m=0 sd=1) # plot standard normal curve x <- pretty(c(-3,3), 30) y <- dnorm(x) plot(x, y, type='l', xlab="Normal Deviate", ylab="Density", yaxs="i")
pnorm(q)	cumulative normal probability for q (area under the normal curve to the right of q) pnorm(1.96) is 0.975
qnorm(p)	normal quantile. value at the p percentile of normal distribution qnorm(.9) is 1.28 # 90th percentile
rnorm(n, m=0,sd=1)	n random normal deviates with mean m and standard deviation sd. #50 random normal variates with mean=50, sd=10 x <- rnorm(50, m=50, sd=10)
dbinom(x, size, prob) pbinom(q, size, prob) qbinom(p, size, prob) rbinom(n, size, prob)	binomial distribution where size is the sample size and prob is the probability of a heads (pi) # prob of 0 to 5 heads of fair coin out of 10 flips dbinom(0:5, 10, .5) # prob of 5 or less heads of fair coin out of 10 flips pbinom(5, 10, .5)
dpois(x, lamda) ppois(q, lamda) qpois(p, lamda) rpois(n, lamda)	poisson distribution with m=std=lamda #probability of 0,1, or 2 events with lamda=4 dpois(0:2, 4) # probability of at least 3 events with lamda=4 1- ppois(2,4)
dunif(x, min=0, max=1) punif(q, min=0, max=1) qunif(p, min=0, max=1) runif(n, min=0, max=1)	uniform distribution, follows the same pattern as the normal distribution above. #10 uniform random variates x <- runif(10)

LET'S START

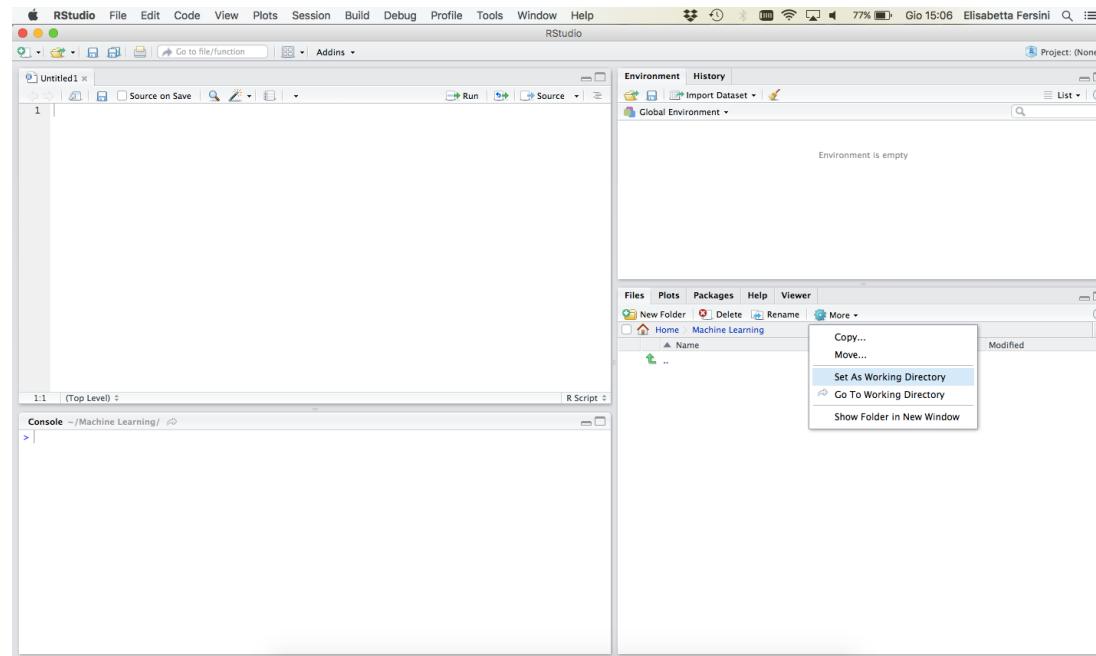


WORKING DIRECTORY

- Set your own **working directory**

```
setwd("~/Machine Learning")
```

- Alternatively



READING AND WRITING DATA

- Before starting to explore data, and learning from data, you must load the data into the R session.
- To **view** the **built-in datasets** of R:

```
data()
```

- R will return a list of datasets in a dataset package, and the list comprises the name and description of each dataset.

READING AND WRITING DATA

- To **load** a specific **dataset** into an R session:

```
data(iris)
```

- The dataset *iris* is now loaded into the data frame format, which is a common data structure in R to store a data table.
- To **view** the **data type** of *iris*, simply use the `class()` function:

```
class(iris)  
[1] "data.frame"
```

READING AND WRITING DATA

- To **store** an **object** in a file use the save() function.

```
save(iris, file="myData.RData")
```

- To **read** a **saved object** into an R session, use the load() function.
-

```
load("myData.RData")
```

READING AND WRITING DATA

- In addition to using built-in datasets, R also provides the read.table() function to **import** data from **text** into a data frame.

```
>test.data = read.table(text = "1 2 3 4",
col.names=c("a","b"),row.names = c("first","second"))
```

- To **export data** to a text file, you can use the write.table() function.

```
write.table(test.data, file = "test.txt" , sep = " ")
```

READING AND WRITING DATA

- You can also **export** data to a **comma-separated** file by using the write.csv() function:

```
write.csv(test.data, file = "test.csv")
```

- With the read.csv() function, the **csv** file can be **imported as a data frame**.

```
csv.data = read.csv("test.csv", header = TRUE, row.names=1)
```

MANIPULATE DATA

- To **select values**, you may use a bracket notation that designates the indices of the dataset.
 - The first index is for the rows and the second for the columns:

```
iris[1,"Sepal.Length"]
```

- You can also **select multiple columns** using `c()`:

```
Sepal.iris = iris[, c("Sepal.Length", "Sepal.Width")]
```

MANIPULATE DATA

- To **subset data** with the rows of given indices, you can specify the indices at the first index with the bracket notation.

```
Sepal.iris = iris[, c("Sepal.Length", "Sepal.Width")]
```

- To **subset data** with the rows of given indices, you can specify the indices at the first index with the bracket notation.

```
Five.Sepal.iris = iris[1:5, c("Sepal.Length", "Sepal.Width")]
```

MANIPULATE DATA

- It is also possible to **set conditions to filter** the data.

```
setosa.data = iris[iris$Species=="setosa",1:5]
```

- The which() function returns the indexes of satisfied data.
 - The following example returns indices of the iris data containing species equal to setosa:

```
which(iris$Species=="setosa")
```

MANIPULATE DATA

- Besides using the bracket notation, R provides a subset() function that enables users to **subset the data frame by observations with a logical statement.**

```
setosa.data = subset(iris, Species == "setosa")
```

- Most of the time, you may want to apply a **union or intersect** a condition (AND, OR) **while subsetting** data.

```
example.data= subset(iris, Petal.Length <=1.4 & Petal.Width >= 0.2,  
select=Species )
```

APPLYING BASIC STATISTICS

- For numeric values, you can perform **descriptive statistics**

```
mean(iris$Sepal.Length)
```

```
sd(iris$Sepal.Length)
```

```
var(iris$Sepal.Length)
```

```
min(iris$Sepal.Length)
```

```
max(iris$Sepal.Length)
```

```
median(iris$Sepal.Length)
```

```
range(iris$Sepal.Length)
```

```
quantile(iris$Sepal.Length)
```

APPLYING BASIC STATISTICS

- In order to obtain summary statistics on every numeric attribute of the data frame, one may use sapply.

```
sapply(iris[1:4], mean, na.rm=TRUE)
```

VISUALIZING DATA

- Calculate the frequency of species within the iris using the table command:

```
table.iris = table(iris$Species)  
table.iris
```

- As the frequency in the table shows, each species represents 1/3 of the iris data. We can **draw** a simple **pie chart** to represent the distribution of species within the **iris**:

```
pie(table.iris)
```

VISUALIZING DATA

- The hist() function creates a **frequency plot** of sorts along the x-axis.

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

- The boxplot() function allow you to convey a lot of information in one simple plot.

```
boxplot(Petal.Width ~ Species, data = iris)
```

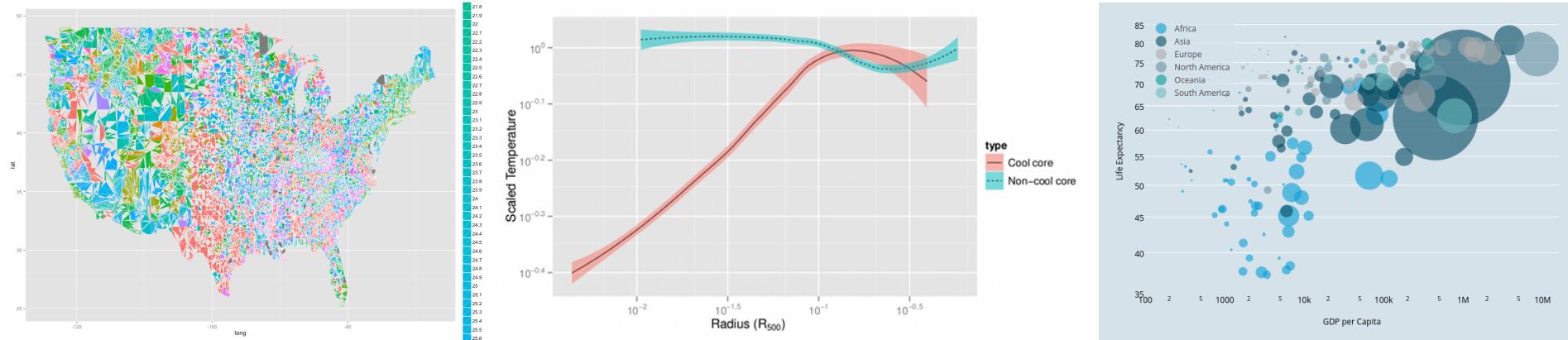
VISUALIZING DATA

- A **scatter plot** is used when there are two variables to plot against one another.

```
plot(x=iris$Petal.Length, y=iris$Petal.Width, col=iris$Species)
```

VISUALIZING DATA

- The **ggplot2** package is another plotting system for R. It allows users to add, remove, or alter components in a plot with a higher abstraction.
- For those of you interested in the topic of **ggplot2**, you can refer to this site: <http://ggplot2.org/>
- Some good examples of “fancy” plots can be found at <http://www.r-graph-gallery.com/portfolio/ggplot2-package/>



GETTING A DATASET FOR ML

- To obtain data from external data repositories:
 1. Access the UCI machine learning repository:
 - <http://archive.ics.uci.edu/ml/>

The screenshot shows the homepage of the UC Irvine Machine Learning Repository. The header features the UCI logo and the text "Machine Learning Repository" and "Center for Machine Learning and Intelligent Systems". A search bar and a red button labeled "View ALL Data Sets" are visible. The main content includes sections for "Latest News", "Newest Data Sets", and "Most Popular Data Sets".

Welcome to the UC Irvine Machine Learning Repository!

We currently maintain 308 data sets as a service to the machine learning community. You may [view all data sets](#) through our searchable interface. Our [old web site](#) is still available, for those who prefer the old format. For a general overview of the Repository, please visit our [About](#) page. For information about citing data sets in publications, please read our [citation policy](#). If you wish to donate a data set, please consult our [donation policy](#). For any other questions, feel free to [contact the Repository librarians](#). We have also set up a [mirror site](#) for the Repository.

Supported By: In Collaboration With:

Latest News:	Newest Data Sets:	Most Popular Data Sets (hits since 2007):
<p>2013-04-04: Welcome to the new Repository admins Kevin Bache and Moshe Lichman!</p> <p>2010-03-01: Note from donor regarding Netflix data</p> <p>2009-10-16: Two new data sets have been added.</p> <p>2009-09-14: Several data sets have been added</p> <p>2008-07-23: Repository mirror has been set up</p> <p>2008-03-24: New data sets have been added!</p> <p>2007-06-25: Two new data sets have been added: UJI Pen Characters, MAGIC Gamma Telescope</p>	<p>2015-01-03: NetlyOffice</p> <p>2014-11-18: eEMG for Basic Hand movements</p> <p>2014-11-05: Sentence Classification</p> <p>2014-10-23: Dow Jones Index</p> <p>2014-10-18: Geographical Origin of Music</p>	<p>668754: Iris</p> <p>469530: Adult</p> <p>395279: Wine</p> <p>328162: Car Evaluation</p> <p>318483: Breast Cancer Wisconsin (Diagnostic)</p>

GETTING A DATASET FOR ML

2. Click on **View ALL Data Sets**. Here you will find a list of datasets containing field names, such as Name, Data Types, Default Task, Attribute Types, # Instances, # Attributes, and Year:
3. Search for **iris** dataset

The screenshot shows the homepage of the UC Irvine Machine Learning Repository. At the top, there is a navigation bar with links for About, Citation Policy, Donate a Data Set, and Contact. Below the navigation bar is a search bar with a "Search" button and a link to "View ALL Data Sets" which is highlighted with a red box. The main content area features a banner for "Machine Learning Repository" with a logo of an antelope. Below the banner, a welcome message says "Welcome to the UC Irvine Machine Learning Repository!". It also includes a note about maintaining 308 data sets and links to About, Citation Policy, and Donation Policy. The page is supported by Rexa.info. On the left, there is a "Latest News" section with a list of recent updates. In the center, there is a "Newest Data Sets" section with a list of recent datasets added, each with a thumbnail, name, and date. On the right, there is a "Most Popular Data Sets (hits since 2007)" section with a list of popular datasets, each with a thumbnail, name, and hit count.

Latest News:		Newest Data Sets:		Most Popular Data Sets (hits since 2007):	
<p>2013-04-04: Welcome to the new Repository admins Kevin Bache and Moshe Lichman!</p> <p>2010-03-01: Note from donor regarding Netflix data</p> <p>2009-10-16: Two new data sets have been added.</p> <p>2009-09-14: Several data sets have been added.</p> <p>2008-07-23: Repository mirror has been set up.</p> <p>2008-03-24: New data sets have been added!</p> <p>2007-06-25: Two new data sets have been added: UJI Pen Characters, MAGIC Gamma Telescope</p>		<p>2015-01-03: NoiseOffice</p> <p>2014-11-18: sEMG for Basic Hand movements</p> <p>2014-11-05: Sentence Classification</p> <p>2014-10-23: Dow Jones Index</p> <p>2014-10-18: Geographical Origin of Music</p>		<p>668754: Iris</p> <p>469530: Adult</p> <p>395279: Wine</p> <p>328162: Car Evaluation</p> <p>318483: Breast Cancer Wisconsin (Diagnostic)</p>	

GETTING A DATASET FOR ML

- Click on [iris](#). This will display the data folder and the dataset description:

UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

Iris Data Set
[Download](#) [Data Folder](#) [Data Set Description](#)

Abstract: Famous database, from Fisher, 1936



Data Set Characteristics:	Multivariate	Number of Instances:	150	Area:	Life
Attribute Characteristics:	Real	Number of Attributes:	4	Date Donated:	1988-07-01
Associated Tasks:	Classification	Missing Values?	No	Number of Web Hits:	659046

Source:
Creator:
R.A. Fisher

GETTING A DATASET FOR ML

- Click on **Data Folder**, which will display a directory containing the iris dataset:

Index of /ml/machine-learning-databases/iris			
Name	Last modified	Size	Description
Parent Directory		-	
Index	03-Dec-1996 04:01	105	
bezdekIris.data	14-Dec-1999 12:12	4.4K	
iris.data	08-Mar-1993 16:27	4.4K	
iris.names	11-Jul-2000 21:30	2.9K	

Apache/2.2.15 (CentOS) Server at archive.ics.uci.edu Port 80

- You can use the `read.csv()` function to read the dataset:

```
iris.data = read.csv(url("http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"), header = FALSE, col.names = c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width", "Species"))
```

OTHER DATASET FOR ML

- Kdnuggets offers a resourceful **list of datasets** for machine learning (and data mining).
 - <http://www.kdnuggets.com/datasets/index.html>
- Kaggle offers a resourceful list of **datasets** and **competitions** for machine learning (and data mining).
 - www.kaggle.com

A CLOSER LOOK AT THE DATASET

- Before starting to create a machine learning model, we should have a closer look at the dataset:
 - *Dimensions.*
 - *Feature types.*
 - *Look at the row data*
 - *Visualising the dataset*

SIZE OF THE DATASET

- We can have an idea of the size of the dataset by using the dim() function:
 - how many instances (rows).
 - how many attribute/features/coordinate (columns) the dataset.

```
dim(iris)
```

FEATURE TYPE OF THE DATASET

- We also need to know the type of the attribute of a given dataset
 - It could be useful to use a specific model or to make some operations on the instances/features
 - Recall that the main feature can be:
 - *Double*
 - *Integer*
 - *String*
 - *Factor*

```
apply(iris, class)
```

Remark: class returns a character vector of all classes an object inherits from

LOOK AT THE ROW DATA

- To **look** at the **row data**, you can use the head() function:

```
head(iris)
```

- To **know** the **nominal classes** in the dataset, you can use the levels() function:
 - In the iris dataset, the classification problem is multinomial (3 levels)

```
levels(iris$Species)
```

Remark: the symbol **\$** allows us to have a direct access to a given attribute

VISUALIZING THE DATASET

- We can understand better our dataset, by using two types of visualization:
 - **Univariate**: to understand a single attribute
 - **Multivariate**: to understand the relationships between attributes

VISUALIZING THE DATASET

- In many many machine learning problems it is useful to distinguish input attributes (input space) from target variable (output space)

```
x = iris[,1:4]  
y = iris[,5]
```

- We can visualize the input attribute by using univariate plots, e.g. (single chart) box plot

```
par(mfrow=c(1,4))  
for(i in 1:4) {  
  boxplot(x[,i], main=names(iris)[i]) }
```

Remarks:

- The **par()** function combine multiple plots into one overall graph.
- The option **mfrow=c(nrows, ncols)** create a matrix of nrows x ncols plots that are filled in by row.
- mfcol=c(nrows, ncols)** fills in the matrix by columns.

VISUALIZING THE DATASET

- We can have a more clear idea by plotting the distribution of each feature of the input space

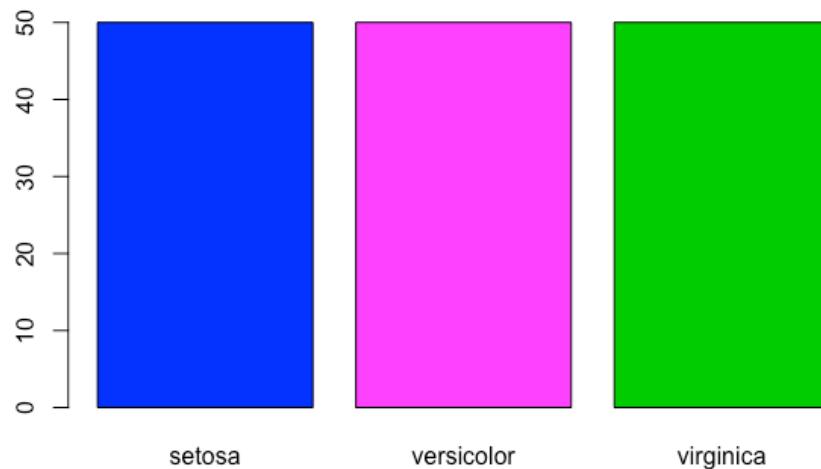
```
library(caret)
featurePlot(x, y, plot="density", scales=list(x=list(relation="free"),
y=list(relation="free")), auto.key=list(columns=3))
```

Remarks: Since the distributions are different in terms of scale, the options **scales=list(x=list(relation="free"), y=list(relation="free"))** automatically makes the chart scale free

VISUALIZING THE DATASET

- We can also **visualize** our **target variable distribution**, by using the plot() function:

```
plot(y,col=c(4,6,3))
```



VISUALIZING THE DATASET

- We can also try to understand if some features are “related” to some other features by using a **multivariate plot**
- Initialize the **caret** package and plot the scatterplot of all couples of input attributes

```
featurePlot(x=x, y=y, plot="pairs", auto.key=list(columns=3))
```