Introducing **R** for statistical analysis

Jose M Sallan `jose.maria.sallan@upc.edu`

Quantitative Research Methods

Outline

1. Introducing **R** and RStudio
2. Data structures in **R**
3. Packages in **R**
4. Reading data files in **R**
5. Statistical data analysis with **R**
6. Conclusions

1. Introducing **R** and RStudio

2. Data structures in **R**

3. Packages in **R**

4. Reading data files in **R**

5. Statistical data analysis with **R**

6. Conclusions

The **R** environment

R is an integrated suite of software facilities for data manipulation, calculation and graphical display. Among other things it has:

- a large, coherent, integrated collection of intermediate tools for data analysis.
- graphical facilities for data analysis.
- a well developed, simple and effective programming language, which includes conditionals, loops, user defined recursive functions and input and output facilities.

R is open source software, that can be extended through **packages**.

RStudio

RStudio is an integrated developed environment (IDE) for **R**, including:

- a console.
- syntax-highlighting editor that supports direct code execution.
- tools for plotting, history, debugging and workspace management.

In this course we will use **RStudio desktop** (free)

Getting **R** and RStudio

R y RStudio are available for most operating systems.

- Getting **R**: `https://www.r-project.org/`.
- Getting RStudio: `https://www.rstudio.com/`.

1 Introducing **R** and RStudio

2 Data structures in **R**

3 Packages in **R**

4 Reading data files in **R**

5 Statistical data analysis with **R**

6 Conclusions

Numeric variables


```
> x <- 2
> X <- 3
> x

[1] 2

> X

[1] 3

> x <- x +1
> x

[1] 3
```

- We can store values in **variables** in a **R** session.
- It is preferable to use <- as assingment operator.
- Variables can be updated, and are case sensitive.

Strings

R can also store **strings**:

```
> a <- "Hello World"
> a
[1] "Hello World"
> b <- "203"
> b
[1] "203"
> c <- as.numeric(b)
> c
[1] 203
```

Variable **b** is a string and variable **c** is numeric.

Factors

We can represent categorical variables with factors. Factors variables especify a factor level for each element.

```
> estado <- c("tas", "qld", "sa", "sa", "sa", "vic", "nt",
+ "act", "qld", "nsw", "wa", "nsw", "nsw", "vic", "vic",
+ "vic", "nsw", "qld", "qld", "vic", "nt", "wa", "wa",
+ "qld", "sa", "tas", "nsw", "nsw", "wa", "act")
> estado <- factor(estado)
> levels(estado)
```

```
[1] "act" "nsw" "nt"  "qld" "sa"  "tas" "vic" "wa"
```

Vectors

vectors store a set of variables of the same type.

Numerical vector of length 5:

```
> d <- numeric(5)
> d
[1] 0 0 0 0 0
```

Defining vectors using `c()`:

```
> e <- c(4,-1,2,3)
> e
[1]  4 -1  2  3
```

A vector of strings:

```
> f <- c("ab", "l", "fz", "a")
> f
[1] "ab" "l"  "fz" "a"
```

A vector of logicals:

```
> g <- c(TRUE, FALSE, TRUE)
> g
[1]  TRUE FALSE  TRUE
```

Subsetting vectors

In **R** vector positions start from 1.
We can use `which` to obtain a subset of vector components satisfying a condition:

```
> s <- c(2, 3, 4, 6, 9, 1, 3)
> s[1]

[1] 2

> s[c(2,5)]

[1] 3 9

> which(s>5)

[1] 4 5

> s[which(s>5)]

[1] 6 9
```

Matrices

We can define matrices of two or more dimensions using a vector as input:

```
> A <- matrix(1:16, 4, 4)
> A

     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
```

```
> B <- matrix(1:16, 4, 4,
+             byrow = TRUE)
> B

     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
[4,]   13   14   15   16
```

Accesing matrices elements


```
> A
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
```

```
> A[2, 3]
[1] 10
> A[ ,3]
[1]  9 10 11 12
> A[2, ]
[1]  2  6 10 14
```

Lists

A **list** is an ordered collection of elements. List elements can be of different types and sizes:

```
> list <- list(albert = 54, bryan = A, carlos = c(1,2,3))
> list[[1]]
```

```
[1] 54
```

```
> list$carlos
```

```
[1] 1 2 3
```

Data frames

A **data frame** is a list of vectors of equal length.

- Data frame columns must have names, and rows can have. We can access them with `rownames` and `colnames`, respectively.
- We can access to row *i* with `df[i, ]`, and to column *j* wiht `df[, j]`.
- As with lists, we can access columns by names using the `$` operator.

Data for statistical analysis is stored in data frames: columns are **variables** and rows **observations**.

Acessing elements of a data frame

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

```
> head(mtcars) #first df rows

                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1

> tail(mtcars) #last df rows

                mpg cyl  disp  hp drat    wt qsec vs am gear carb
Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.7  0  1    5    2
Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.9  1  1    5    2
Ford Pantera L 15.8   8 351.0 264 4.22 3.170 14.5  0  1    5    4
Ferrari Dino   19.7   6 145.0 175 3.62 2.770 15.5  0  1    5    6
Maserati Bora  15.0   8 301.0 335 3.54 3.570 14.6  0  1    5    8
Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.6  1  1    4    2
```

Acessing elements of a data frame


```
> length(mtcars) # of variables
[1] 11
> nrow(mtcars)   # of observations
[1] 32
> mtcars[3, ]
            mpg cyl disp hp drat   wt  qsec vs am gear carb
Datsun 710 22.8   4  108 93 3.85 2.32 18.61  1  1    4    1
> mtcars$mpg[1:10]
 [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2
```

1 Introducing **R** and RStudio

2 Data structures in **R**

3 Packages in **R**

4 Reading data files in **R**

5 Statistical data analysis with **R**

6 Conclusions

Packages in **R**

R is delivered with a set of `base` functions and datasets. You can add new ones according to your needs installing **packages** from the CRAN repositories.

- Install packages like in `install.packages("psych")` or in the **Package** tab in RStudio.
- To get package functionalities in a **R** session type `library(psych)`.

R packages examples

Some examples of available packages:

`psych`
`corrplot`
`car`
`AER`
`lavaan`
`foreign`
`dplyr`
`ggplot2`

1 Introducing **R** and RStudio

2 Data structures in **R**

3 Packages in **R**

4 Reading data files in **R**

5 Statistical data analysis with **R**

6 Conclusions

R and data file formats

Usual file formats to read in **R**:

- Read and write text files `.txt` with `read.table` and `write.table`.
- Read and write `.csv` files with `read.csv` and `write.csv`.
- Read and write RDS files with `readRDS` and `saveRDS`.
- Read SPSS, SAS files with `foreign`.

The standard file formats for data in **R** are `.csv` and `.rds` (compressed).

Reading a data file in **R**

Steps to follow:

- Set working directory with `setwd` function or in **Files** tab of RStudio.
- Read file and assign name.
- Take into account if the firs row of the file contains variable names with parameter `header`.

1 Introducing **R** and RStudio

2 Data structures in **R**

3 Packages in **R**

4 Reading data files in **R**

5 Statistical data analysis with **R**

6 Conclusions

Statistical data analysis with **R**

R allows for a wide variety of analysis. Here we will focus on **statistical analysis in social sciences**.

Some of the topics to be covered:

- Examine **data structure**.
- Deal with **missing data**.
- Examine data properties with **graphs**.
- Perform **statistical analysis**.

I will illustrate this topics with a small example.

Examining data

Once you start analyzing data, the first step is to examine it. A protocol for examining data includes:

1. Checking the beginning and end of data frame with `head` and `tail`.
2. Check data frame structure with `str`.
3. Get basic statistics and information on missing data with `summary`.
4. Detect missing data with `is.na` and list complete cases with `complete.cases`.

Beginning and end of data frame

```
> head(airquality)

  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6

> tail(airquality)

    Ozone Solar.R Wind Temp Month Day
148    14      20 16.6   63     9  25
149    30     193  6.9   70     9  26
150    NA     145 13.2   77     9  27
151    14     191 14.3   75     9  28
152    18     131  8.0   76     9  29
153    20     223 11.5   68     9  30
```

Checking data structure

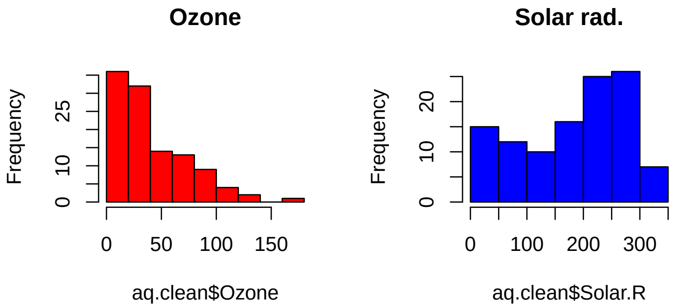

```
> str(airquality)

'data.frame':        153 obs. of  6 variables:
 $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
 $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
 $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
 $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
 $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
 $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...

> summary(airquality)

     Ozone           Solar.R           Wind             Temp
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00
 1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:72.00
 Median : 31.50   Median :205.0   Median : 9.700   Median :79.00
 Mean   : 42.13   Mean   :185.9   Mean   : 9.958   Mean   :77.88
 3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00
 Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00
 NA's   :37       NA's   :7
     Month            Day
 Min.   :5.000   Min.   : 1.0
 1st Qu.:6.000   1st Qu.: 8.0
 Median :7.000   Median :16.0
 Mean   :6.993   Mean   :15.8
 3rd Qu.:8.000   3rd Qu.:23.0
 Max.   :9.000   Max.   :31.0
```

Dealing with missing data

```
> aq.clean <- airquality[which(!is.na(airquality$Ozone)
+                                  & !is.na(airquality$Solar.R)), ]
> nrow(airquality)

[1] 153

> nrow(aq.clean)

[1] 111

> summary(aq.clean)

     Ozone          Solar.R           Wind            Temp      
 Min.   :  1.0   Min.   :  7.0   Min.   : 2.30   Min.   :57.00  
 1st Qu.: 18.0   1st Qu.:113.5   1st Qu.: 7.40   1st Qu.:71.00  
 Median : 31.0   Median :207.0   Median : 9.70   Median :79.00  
 Mean   : 42.1   Mean   :184.8   Mean   : 9.94   Mean   :77.79  
 3rd Qu.: 62.0   3rd Qu.:255.5   3rd Qu.:11.50   3rd Qu.:84.50  
 Max.   :168.0   Max.   :334.0   Max.   :20.70   Max.   :97.00  
     Month            Day       
 Min.   :5.000   Min.   : 1.00  
 1st Qu.:6.000   1st Qu.: 9.00  
 Median :7.000   Median :16.00  
 Mean   :7.216   Mean   :15.95  
 3rd Qu.:9.000   3rd Qu.:22.50  
 Max.   :9.000   Max.   :31.00  
```

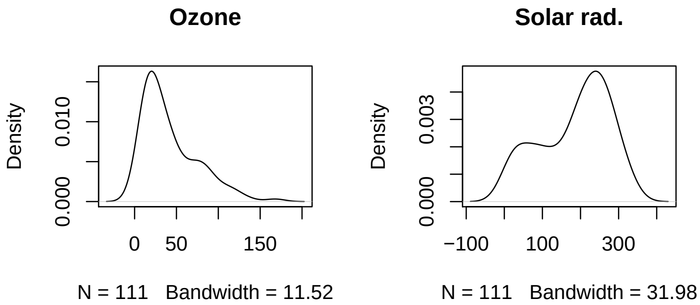
One variable plots: histograms

```
> par(mfrow=c(1,2))
> hist(aq.clean$Ozone, col="red", main="Ozone")
> hist(aq.clean$Solar.R, col="blue", main="Solar rad.")
```



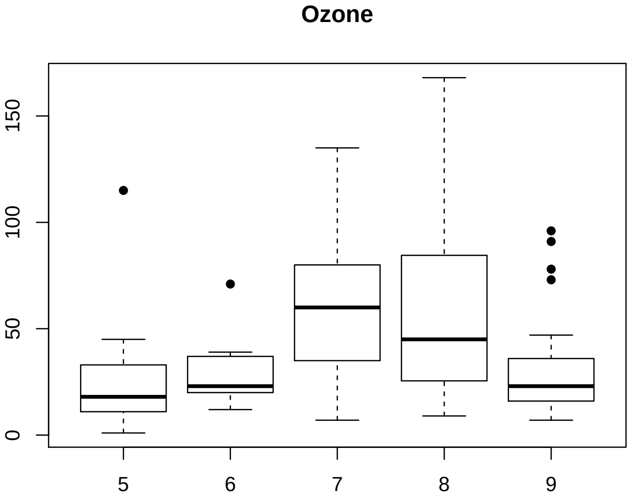
One variable plots: density plots

```
> d.ozone <- density(aq.clean$Ozone)
> d.solar <- density(aq.clean$Solar.R)
> par(mfrow=c(1,2))
> plot(d.ozone, main="Ozone")
> plot(d.solar, main="Solar rad.")
```



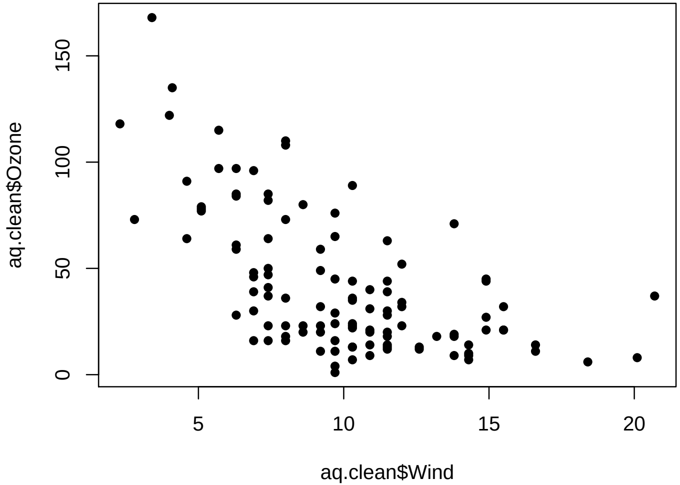
One variable plots: boxplots


```
> boxplot(Ozone ~ Month, data=aq.clean, pch=16, main="Ozone")
```



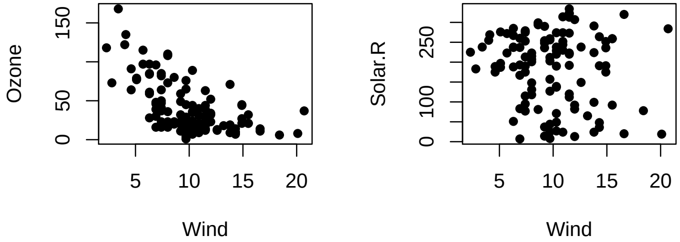
Two variable plots: scatterplots

```
> plot(aq.clean$Wind, aq.clean$Ozone, pch=16)
```



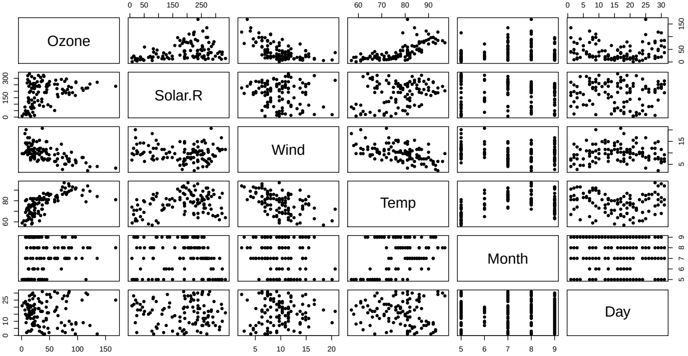
Two variable plots: scatterplots


```
> par(mfrow=c(1,2))
> plot(aq.clean$Wind, aq.clean$Ozone, pch=16, xlab="Wind", ylab="Ozone")
> plot(aq.clean$Wind, aq.clean$Solar.R, pch=16, xlab="Wind", ylab="Solar.R")
```



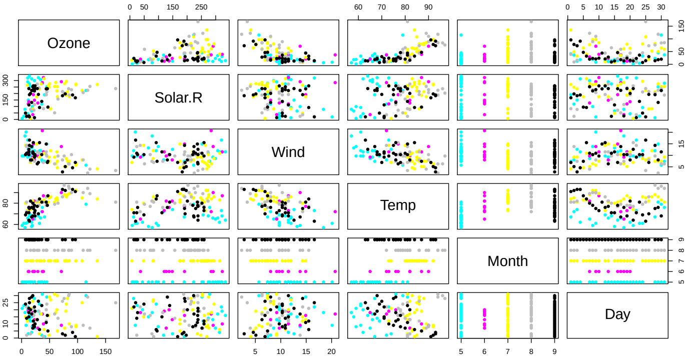
Two variable plots: scatterplots

```
> plot(aq.clean, pch=16)
```



Two variable plots: scatterplots

```
> plot(aq.clean, pch=16, col=aq.clean$Month)
```



t-test analysis

Comparing means of temperature Temp for months 5 and 9 in airquality

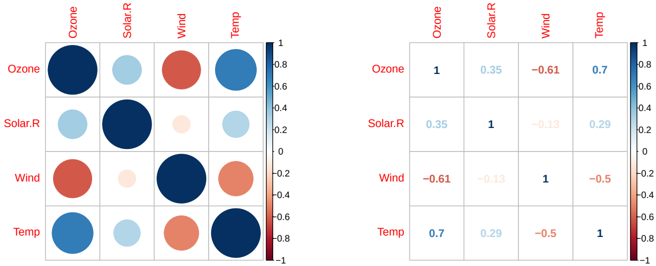
```
> aq.clean$Month <- factor(aq.clean$Month)
> airquality.59 <- aq.clean[which(aq.clean$Month==5 | aq.clean$Month==9), ]
> t.test(Temp ~ Month, data=airquality.59)

        Welch Two Sample t-test

data:  Temp by Month
t = -5.0182, df = 50.847, p-value = 6.752e-06
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -14.614451  -6.261986
sample estimates:
mean in group 5 mean in group 9
       66.45833        76.89655
```

Correlations with psych


```
> library(psych)
> corr.test(aq.clean[, 1:4])

Call:corr.test(x = aq.clean[, 1:4])
Correlation matrix
        Ozone Solar.R  Wind  Temp
Ozone    1.00    0.35 -0.61  0.70
Solar.R  0.35    1.00 -0.13  0.29
Wind    -0.61   -0.13  1.00 -0.50
Temp     0.70    0.29 -0.50  1.00
Sample Size
[1] 111
Probability values (Entries above the diagonal are adjusted for multiple tests.)
        Ozone Solar.R Wind Temp
Ozone       0    0.00 0.00    0
Solar.R     0    0.00 0.18    0
Wind        0    0.18 0.00    0
Temp        0    0.00 0.00    0

 To see confidence intervals of the correlations, print with the short=FALSE option
```

Correlations with `corrplot`

```
> library(corrplot)
> par(mfrow=c(1,2))
> cor.aq <- cor(aq.clean[, 1:4])
> corrplot(cor.aq, method = "circle")
> corrplot(cor.aq, method = "number")
```



1 Introducing **R** and RStudio

2 Data structures in **R**

3 Packages in **R**

4 Reading data files in **R**

5 Statistical data analysis with **R**

6 Conclusions

Conclusions

1. **R** is a strong platform for statistical analysis (not only for research purposes).
2. **R** learning curve can be smoothed with RStudio.
3. You can learn about your data exploiting the graphical possibilities of **R**.
4. You can adapt **R** to your needs installing packages.