Statistics with R

Marco Bittelli and Martina Zappaterra
Department of Agriculture and Food Sciences, University of Bologna, Italy

Preface

The notes collected here have been written over a few years as a support material for classes in Statistics and Experimental Methodology that we teach at the University of Bologna. The notes and the R computer code is original material that we wrote, while the experimental data were collected during experiments that we performed in collaboration with colleagues and that have been published and cited in the notes. Some data have been given to us as teaching examples. We are grateful to Livia Antisari for sharing soil data from her experiments in the Emilia-Romagna Appennines, Aldo Gardini e Carlo Trivisano for code used to analyze the soil data, and Gabriele Antolini for Emilia-Romagna weather data. We also would like to thank Roberto Olmi for computer code and notes.

Contents

1
Introduction

1.1 Introduction

In the last four decades there have been an active development of analytical tools in statistics, based on the power of computer technology and computation. Today, the integration of classical theory and computational tools is an important component of current teaching curricula. It is very effective to teach statistics using programming languages that allows for direct application of concept to read world data, from the beginning of the learning process. There are a variety of good commercial programs for applications in statistics, including MatLab and others. However, the open source R programming language is the most popular one.

The goal of these notes is to help the student learn the most important tools in R that will allow for statistics and data analysis. The notes are divided in two main parts.

The first part of the notes will introduce the use of R and RStudio (Chapter 2) and fundamental concepts of data management programming that are necessary to understand the program and utilize it (Chapters 3-6). Concepts of *data types* and *operators* are provided, along with *conditional statements* and *loops*. In particular data requires *importing data*. Importing data may be more complicated than expected since the data may present *missing data*, *time series* may be organized with specific data formatting that should be properly read and so forth. After successfully importing the data, the second step is *visualization*. A good visualization will show the data in ways that the student may not expect or raise new questions about the data. These notes will explain the different visualization procedures, the type of graphs and other visualization options.

The second part is concerned with statistical analysis. This part presents an introduction to the most common one taught as preliminary classes in college statistics. Chapter 7 introduces basic notions of *probability*, including basic information on set theory, definition of probability, the law of large numbers and conditional probability. The chapter also introduces the Bayes theoream with examples of Bayesian statistics.

Chapter 8 describes the concept of continuous and discrete *random variables and distributions*. This chapter describes the bimodal, normal, t-student, geometric, chi-square, logistic and Poisson distributions.

Chapter 9 enters into statistical methods with principles of descriptive statistics. The chapter begins with descriptive statistics such as the construction of *classes*, *cumulative curves* and data visualization tools. Then it describes measures of *central tendency* and *variability* with examples in R on real data. Concepts of *quantiles* and *box plots* are provided.

Chapter 10 introduces inferential statistics with concepts of populations, samples and inferential methods. Chapter 11 introduces the concept of linear models, linear regression and correlation. Chapter 12 discusses the most common experimental designs, which are cornerstone for a good experiment and therefore a solid statistical analysis.

Chapter 13 describes the analysis of variance (ANOVA) with some simple applications and further examples on more complex data analysis.

Chapter 14 describes multivariate statistics including covariance matrices, correlation matrices, distances, principal component analysis, cluster analysis and other common techniques, while Chapters 14 non−linear optimization regression procedures and least squares concepts applied to non linear models.

2
RStudio

In this notes the integrated development environment (IDE) RStudio is used for programming. RStudio allows for using an intuitive and powerful graphical interface to write programs in the language R. It is available in two formats: RStudio Desktop which is a freeware desktop application and RStudio Server runs on a remote server and allows accessing RStudio using a web browser.

RStudio allows for installation of a large number of R packages used in these notes. There are many packages for statistical analysis for basic statistics to multivariate statistics. It has image analysis, geostatistics, GIS and many other packages.

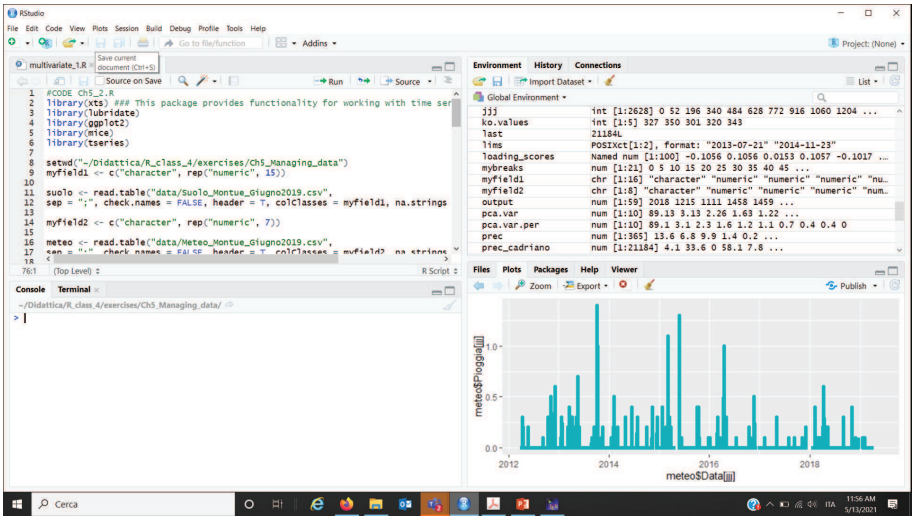


Fig. 2.1

3
Data types and operators

In statistical analysis we usually classify data into qualitative and quantitative data. Data are observations of a variable we are dealing with. Qualitative data are the ones that can be described through specific attributes, for instance the sex of a person (female or male). For a qualitative variable numerical measurement is not possible. Gender is qualitative variable, presence or absence of a pathogen in a sample. When a variable is qualitative and it allows for two possibility is called a dichotomous or binary variable.

Quantitative variables are the ones that can be represented by a number such as the air temperature (10, 15 Celsius). Quantitative data are classified in discrete variables such as daily average temperature, or the number of units in a population. Continuous variables present all values of a given range are possible. We may be limited to measure all the possible values by the accuracy and precision of the measurement device.

Ordinal variables are variables with a score, with a scale. A leaf damage from a pathogen can be classified with 1,2,3 and 4. Where 1 is a leaf with no damage, 2 presents little areas of chlorotic presence and so forth to level 4 where all the leave is covered by the pathogen damage or lesions.

It is important to identify which type of variables we are dealing with, since it will affect the type of statistics we will apply.

To analyze data, being qualitative, ordinal or quantitative with the use of computer, it is important to understand the concept of a variable from a computer stand point. A variable is reserved memory in the computer to store values. Therefore when we create a variable we are simply reserving a space in the computer memory. The variable can have different format, it can be a text such as the word *cat*, or it can be a value of temperature such as *24*. Depending on the type of variable, computer uses different ways of storing that memory space. In this chapter, data types and operators are described.

Another important concept is difference between an input variables also called predictor or independent variable and the output variables also called response or dependent variables. Ideally, a cause−effect system is identified, with a deterministic structure, however often we can only find associations. There are five data types in R:

- Vector
- Matrix
- Array
- List
- Data Frame

Let's now discuss these five tipes.

3.1 Vector

A vector is a sequence of data elements of the same type. There five classes of vectors

- Logical (True or False)
- Integer (1, 3, 2500)
- Numeric (1.34, -3200.12, 3.1415)
- Complex (4 + 3i, 1+2i)
- Character ("Hello")

When we talk about vectors we can have a single value or a sequence of values.

The following statement is such that we type x = 15, we then enter and R returns the value of the variable x that it is now equal to 15. The number [1] into the squared parenthesis means that even a scalar is defined as a vector of length equal to 1. Therefore the value 15 is the first element of the vector x.

```
> x=15
> x
[1] 15
>
```

A vector can be created, containing 4 elements. It is important here to specify that we do not refer as a vector as we do in linear algebra or physics, but in this contest a vector is simply a series of consecutive values. If the vector y is created, by typing $y <- c(2, 4, 6, 8)$, and then typing y at the prompt, R returns the input values.

```
> y <- c(2, 4, 6, 8)
> y
[1] 2 4 6 8
>
```

The instruction c before the parenthesis is used to concatenate the values. Note that in R an arrow is used $<-$, it means that the value are assigned to the variable y.

Let's now create a Logical vector. After opening R studio, a new script is created by opening a new file and naming it `Datatypes.R`. The following statements are then written:

```
#Vectors
#Logical
Vtr1=c(TRUE,FALSE)
```

When this script is executed (for instance by selecting only the line with the vector definition and hitting run), a new variable is created. This is visible in the windows on the upper part of the screen, called `Environment`.

The third vector type is Integer. It means that the variable is a number without decimal places.

The third type is Numeric. It means that the variable is a number that can present decimal places.

The fourth type of vector is Complex. It means that the variable is a complex number with a real and an imaginary part. Finally the fifth type is Character, that is a variable written as text, where we can store a single character or a sequence of words as well.

Now we can use R and create these vectors as example.

```
#Vectors

#Logical
vtr1 =c(TRUE, FALSE)
class(vtr1)

#Integer
vtr2= c(6L,34L,9999L)
class(vtr2)

#Numeric
vtr3= c(6,43.23,55555)
class(vtr3)
```

Now if we execute this statement, we will see that a new variable has been created, that is a logical data type and it stores two values, TRUE and FALSE. If we want to know to which class the variable belongs we can type the instruction `class` as shown above. Executing the program will return the class type. If we type the name of the variable in the R console, for instance `vtr2`, the program will return the values of the vector. When I write an `L` after the number, it is treated as an integer.

The data type of a variable can be changed with the instruction below, where the variable `Year` belonging to the data.frame `NevadaPrec` is converted from `int` to `num`:

```
NevadaPrec$Year <- as.numeric(NevadaPrec$Year)
```

To print a number in scientific notation, the following instruction is used.

```
formatC(1/3.630781e-09, format = "e", digits = 2)
```

3.1.1 Operators

Operators are construct that can manipulate the value of the operands. There are four types of operators:

- Arithmetic
- Relational
- Assignment
- Logical

3.1.2 Arithmetic

There are many types of arithmetic, such as

- Addition → a+b
- Subtraction → a-b
- Multiplication → a*b
- Division → a/b
- Modulus → a%%b
- Exponent → a $\hat{}$ b
- Floor Division → a% / %b

A simple example is:

```
>print(5+3)
[1] 8
```

We can change it to division, multiplication, etc.

There are also other operators such as the modulus. In this case it provides the reminder of a division. For example, the expression 5%%2 would return 1 because 5 divided by 2 has a quotient of 2 and a remainder of 1, while 12%%3 would evaluate to 0 because the division of 12 by 3 has a quotient of 4 and leaves a remainder of 0.

```
>print(5%%2)
[1] 1
```

The exponent operator returns the exponent of the base `a` raised to the power of `b`.

```
>print(2^2)
[1] 4
```

Finally the floor division, which is the following. If the reminder of a division is a decimal number, then the floor division rounds it up to the previous whole number.

```
>print(22%/%7)
[1] 3
```

The division 22/7 returns the value 3.142857, but since it is a floor division, it is rounded to the previous whole number that is 3.

Several mathematical operations are possible such as logarithms:

```
log(10)
[1] 2.302585
```

or sin or cos:

```
sin(1.571)
[1] 1
```

where 1.571 is the value of $\pi/2$, since angles are in radians.

Numbers can be formatted according to a specific language as shown below.

```
> formatC(0.6569866,digits=2,format="e")
[1] "6.57e-01"
```

3.1.3 Relational

Relational operators are used to compare the actual values of two variables and then the output is a boolean value (TRUE or FALSE). There are many types of relational operators, such as

- Equal to → a==b
- Not Equal To → a! =b
- Greater Than → a>b
- Less Than → a<b
- Greater Than Equal To → a>=b
- Less Than Equal To → a<=b

The following examples shows two applications:

```
> a=4
> b=5
> a>b
[1] FALSE
```

```
> a=4
> b=5
> a==b
[1] FALSE
```

Also with a not equal to:

```
> 1!=1
[1] FALSE
```

Relational operators are used in condition statements.

3.1.4 Assigment

There are two types of assignment operators: `Left` and `Right`. It means that they can go from left to right or from right to left.

For example for the `Left`: x = 3 is an equal symbol, but in R it is also possible to use the x < −3 symbol, that it means that the variable x has a value of 3. For the `Right` 20 = x is an equal symbol, but in R it is also possible to use the 20 − > x symbol, that it means the value of 20 is now assigned to the variable x.

Both of these directions are possible.

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

But also assign the number 20 to the variable x.

```
> 20 ->x
> x
[1] 20
```

Also a name can be given to a variable and then check the value of that variable:

```
apple<- 6
> apple >2
[1] TRUE
```

In R **vectors** are created by using the instruction below:

```
> lemon <-c(2,4,5,2)
> lemon
[1] 2 4 5 2
```

but also with strings:

```
> gender <-c("M","M","M","F","F","F")
> gender
[1] "M" "M" "M" "F" "F" "F"
```

Vectors can also be created as

```
> minnie <- 1:6
> minnie
[1] 1 2 3 4 5 6
```

It can also be done with the `seq()` instruction. Where the first number is the minimum, the maximum and the interval.

```
> minnie <- seq(1,10,2)
> minnie
[1] 1 3 5 7 9
```

It is possible to set the number of elements into an interval:

```
minnie2<- seq(1,20, length.out=5)
> minnie2
[1] 1.00 5.75 10.50 15.25 20.00
```

It is possible to identify the elements of a vector by using square brackets, where the number identify the element in the vector.

```
> minnie[1]
[1] 1
```

Or to extract the elements within an interval:

```
> minnie[1:3]
[1] 1 3 5
```

3.1.5 Functions

There are many functions in R that can be used. For instance, the minimum and max values:

```
> min(minnie)
[1] 1

> max(minnie)
[1] 9
```

It is possible to determine the `length` of a vector:

```
cat <- c(1:1000)
length(cat)
```

It is possible to determine the `mean` of a vector:

```
> mean(minnie)
[1] 5
```

Numbers can be sorted:

```
> rabbit <-c(123,4,67,990,3,6)
> sort(rabbit)
[1] 3 4 6 67 123 990
```

And then assign it to a new vector:

```
> sorted_rabbit <-sort(rabbit)
> sorted_rabbit
[1] 3 4 6 67 123 990
```

A list is created, where the first element is number, the second a string and the third a boolean:

```
jupiter <- list(2, "rabbit", T)
```

The elements can be called with the double squared bracket:

```
> jupiter[[2]]
[1] "rabbit"
```

3.1.6 Logical

There are three types of logical operators; `AND`, `NOT` and `OR`.

- AND → $a\&b$
- NOT → a|b
- OR → !a

`AND` combined each element of vectors and gives an output TRUE if both elements are TRUE, `NOT` combines each elements of a vector and gives an output TRUE if one element is TRUE. Finally `OR` takes each element of the vector and gives the opposite logical value. An example is:

```
> value1 =c(TRUE, FALSE, TRUE, FALSE)
> value2=c (FALSE, TRUE, TRUE, FALSE)
> print(value1 &value2)
[1] FALSE FALSE TRUE FALSE
```

So with the statement `AND`, it is checked when both values are TRUE. So each value has been checked with the corresponding value of the two vectors. So only if both values are TRUE, then only in that case it is printed that both values are TRUE. Now, in case of an `OR` operator:

```
> value1 =c(TRUE, FALSE, TRUE, FALSE)
> value2=c (FALSE, TRUE, TRUE, FALSE)
> print(value1 |value2)
[1] [1] TRUE TRUE TRUE FALSE
```

In that case, only when both statement are FALSE, then it is printed as FALSE, otherwise they are printed as TRUE, since the first vector or the second vector can have a TRUE in the first, second and third position.

3.2 Matrix

Matrix are objects that are arranged into a two-dimensional layout.

- data is the input vector data becomes the data entry of the matrix
- nrow is the number of rows to be created
- ncol is the number of columns to be created
- byrow is a logical statement. If it is TRUE, then the input elements are arranged by rows
- dimname is the name assigned to rows and columns

The statements is matrix(data,nrow,ncolumns,byrow,dimname).

```
> david <-matrix(data=1:20,5,4)
> david
[,1] [,2] [,3] [,4]
[1,] 1 6 11 16
[2,] 2 7 12 17
```

```
[3,] 3 8 13 18
[4,] 4 9 14 19
[5,] 5 10 15 20
```

To get the values, the rows and colums are called:

```
> luca[1,2]
[1] 6
```

Another example on how to create a matrix is now presented:

```
#Matrix

mtr=matrix(c(5:29),5,5)
```

The instruction c(5:29) means that a sequence of numbers starting from 5 to 29 will be stored, which are 25 numbers. So there will be 25 elements with increment of one. The number of rows are five and the columns are five. At this point we do not need to specify an order of arrangment and a name. The output will be:

```
     [,1] [,2] [,3] [,4] [,5]
     [1,] 5 10 15 20 25
     [2,] 6 11 16 21 26
     [3,] 7 12 17 22 27
     [4,] 8 13 18 23 28
     [5,] 9 14 19 24 29
```

Without specification, R organize a matrix by columns, if we change it

```
     mtr=matrix(c(5:29),5,5, TRUE)

     then the output will be:

     [,1] [,2] [,3] [,4] [,5]
     [1,] 5 6 7 8 9
     [2,] 10 11 12 13 14
     [3,] 15 16 17 18 19
     [4,] 20 21 22 23 24
     [5,] 25 26 27 28 29
```

```
> is.matrix(luca)
[1] TRUE
```

The transpose can be computed

```
julie <-t(luca)
julie
```

3.3 Array

An array is similar to a matrix. The syntax is

```
#Array

arr=array(c(0:15),dim=c(4,4,2,2))

, , 1, 1

[,1] [,2] [,3] [,4]
[1,] 0 4 8 12
[2,] 1 5 9 13
[3,] 2 6 10 14
[4,] 3 7 11 15

, , 2, 1

[,1] [,2] [,3] [,4]
[1,] 0 4 8 12
[2,] 1 5 9 13
[3,] 2 6 10 14
[4,] 3 7 11 15

, , 1, 2

[,1] [,2] [,3] [,4]
[1,] 0 4 8 12
[2,] 1 5 9 13
[3,] 2 6 10 14
[4,] 3 7 11 15

, , 2, 2

[,1] [,2] [,3] [,4]
[1,] 0 4 8 12
[2,] 1 5 9 13
[3,] 2 6 10 14
[4,] 3 7 11 15
```

This array will store sixteen elements. The instruction `dim=c(4,4,2,2))` is used to define the dimension of the array. The first two columns are specifying the standard size of each matrix that is going to be stored into the array, then it will be stored into a 2 cross 2 array (where the first indicates the column). Note that R prints the column first and then the row. So the first matrix (1,1) is first colum, first row; the second one is column 2, row 1 and so forth.

3.4 List

The list is similar to the vector, but the only difference is that in a list we can store different data types in a list. An example shows two vectors,

```
vtr7= c(5.6,9, 18)
vtr8= c("How are you", 'Fine thanks')
```

The first one is populated by numbers, while the second is populated by text. Now, a list is created where we pass `vtr7` and `vtr8`.

```
> list1= list(vtr7,vtr8)
> list1
[[1]]
[1] 5.6 9.0 18.0

[[2]]
[1] "How are you" "Fine thanks"
```

None of these variables have been converted, so the list preserve the original data type.

3.5 Data Frame

A data frame is a table that can store the data into an order manner. It can be data that have created in an excel spreadsheet from an experiment. An example on how to create a data frame is provided.

```
>vtr1=c(1:5)
>vtr2=c("Lucio", "Fabrizio", "Anna", "Rita","Sofia")
>vtr3=c(65,80,60,63,72)
```

A data frame is created by passing the name of the vectors:

```
>data.frame(vtr1,vtr2,vtr3)

vtr1 vtr2 vtr3
1 1 Lucio 65
2 2 Fabrizio 80
3 3 Anna 60
4 4 Rita 63
5 5 Sofia 72
```

4
Conditional Statements and Loops

There are three types of conditional statements. The first statement is the `If` statement.

4.1 If statement

```
       var1=20
       var2=30
       if ((var1+var2)>30) {
              print("Greater than 30")}
       [1] "Greater than 30"
```

So, when the condition is true then the statement is printed. If the value was 60

```
var1=20
var2=30
if ((Var1+var2)>60) {
  print("Greater than 30")}
```

the program would not print anything, since the condition is not met and the program does to execute the print statement.

4.2 Else If statement

If there are more than one conditions that must be checked, than we could use the `else if`

```
if ((expression) 1) {
 Statement
                   }
else if (expression 2)
{
 Statement 2
                   }
```

Here an example is shown. It is important to use the correct indentation.

```
var1=20
```

```
var2=30
if ((var1+var2)>60) {
  print("Greater than 60")
  } else if ((var1+var2)>40)
  {
  print("Greater than 40")
  }
```

4.3 Else statement

The Else statement is executed if no other condition is met.

```
var1=20
var2=30
if ((var1+var2)>100) {
  print("Greater than 100")
} else if ((var1+var2)>60)
{
    print("Greater than 60")}
else print ("Number is less than 60")
```

So in this case, instead of doing nothing if the condition is not met, than the `else` statement allows to get out of the loop. So all the conditions were checked, and if no conditions was met, then the else statement allows for provide information about the outcome of the conditional statement.

4.4 For loop

The `For−`loop is a control flow statement where an iteration of a given lenght is specified, allowing for the code to be executed repeatedly. This instructions is very important in programming since it allows to iteratively control and modify variables in arrays or matrices and to perform operations over large number of data in a very efficient way.

In general a for−loop is written in two parts: a header specifying the iteration, and a body where the operation are executed, once per iteration. Usually the header declares a counter or a loop variable that is used to determine how many times the operations written in the body are executed.

Below is an example, where to the variable `k1` is assigned ten numbers having a normal distribution. The instruction `rnorm()` generate random numbers having a normal distribution with mean equal to `mean` and standard deviation equal to `sd`. By typing `k1` the generated values are printed on the console, then the variable squared is initiazed to zero. The `For` loop is then written by defining the number of times the algorithm will iterate (1:10) and after the curly bracket the instructions are written. The function `squared[]` is defined by the multiplication of `k1` by itself. Note that the variable `i` is the index of the iteration. Finally the values are printed to the console. The index is also printed outside the loop.

```
k1 <- rnorm(10)
k1
squared <- 0
for(i in 1:10) {

       squared[i] <- k1[i]*k1[i]
       print(squared[i])
}
print(i)
```

4.4.1 Nested For−loop

A nested loop is a loop within a loop, an inner loop within the body of an outer one. The first pass of the outer loop execute the inner loop, then the outer loop is executed again and it passes again into the inner loop and so forth.

Below is an example where the example above was modified by including an inner loop that prints out the constant number 4 for two times. The counter `j` goes from 1:2, and the variable `const` is printed. This is not a very interesting operation but it is useful to understand how nested loops work.

```
k1 <- rnorm(10)
k1
squared <- 0
const <- 0
for(i in 1:10) {

       squared[i] <- k1[i]*k1[i]
       print(squared[i])

       for(j in 1:2) {

              const[j] <- 4
              print(const[j])


       }

}

print(i)
print(j)
```

The output of this loop is shown here. The outer loop compute the squared number of `k1`, it prints it to the console and then it execute the inner loop. The inner loop is executed two times, printing the value of 4. The iteration goes back to the outer loop and so forth.

```
[1] 3.148467
[1] 4
[1] 4
[1] 0.007544777
[1] 4
[1] 4
[1] 1.913666
```

4.5 While loop

The `While-`loop is a control flow statement where an iteration of a given statement is performed *while* a condition is met. In other words a test expression is evaluated and the body of the loop is entered if the result is of this test is TRUE.

```
while (test_expression)
{
       statement
}
```

An simple example is presented below

```
i <- 2
while (i < 12) {
       print(i)
       i = i+1
}
```

The output for this code is:

```
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
[1] 11
```

5
Managing Data

5.1 Qualitative and quantitative data

In statistics, data are usually classified as **qualitative** and **quantitative**. Qualitative data describes a specific attribute such as gender (male or female), or a general description of age such as young, adult, elderly. On the other hand quantitative data are described on a numerical scale. For instance, the air temperature in degree Celsius, or the age of a person. Let's assume that we have a group of six people, three men and three women, respectively of age 9, 10, 50, 80, 14 and 91. Two vectors are now built using a **qualitative** scale defined as 'young', 'adult' and 'elderly'.

```
> gender <-c("M","M","M","F","F","F")
> age <-c("Young","Young","Adult","Elderly","Young","Elderly")
> str(gender)
 chr [1:6] "M" "M" "M" "F" "F" "F"
> str(age)
 chr [1:6] "Young" "Young" "Adult" "Elderly" "Young" "Elderly"
```

The instruction `str()` display the internal **str**ucture of an R object. In R there is a convenient and powerful help, with the description of each command and function. It is obtained by typing a question mark, followed by the name of the command.

```
> ?str()
```

In this case, when the structure of the vector `gender` and `age` were inquired, by using the instruction `str()`, the program printed `chr [1:6]` followed by the names of the elements. The `chr` indicates that the vector is populated by *character* values. The character was described in the section above on data types.

In general computers and programming languages do not like to deal with characters and text data such as words, symbols, etc. therefore it is often advisable to convert a qualitative information into a numerical scale. The function `factor` is used to encode a vector as a factor (the terms 'category' and 'enumerated type' are also used for factors). `factor` returns an object of class 'factor', which has a set of integer codes the length of x.

An example is the list of cities in Italy, when we are asked by a software to provide our place of birth. The user will see a list of city to choose from (Agrigento, Ancona, Avellino,..., Bari, Bologna,...). The computer will not store into the memory a list of strings (character), but it will assign an enumerated type to each city. Depending on the programming language, different instructions are used. In R, the category is

assigned with the instruction `factor`. In this piece of code, a new variable is created (`age2`), on which the returned values from the function `factor` are stored. When `age2` is typed, the output is a list of the vector elements and the specification of the `Levels`. When the the internal **str**ucture of the variable is inquired with `str()`, the output returns a list of three enumerated codes (1,2,3) corresponding to the the three levels (young, adult and elderly).

```
> age2<-factor(age)
> age2
[1] Young Young Adult Elderly Young Elderly
Levels: Adult Elderly Young
> str(age2)
 Factor w/ 3 levels "Adult","Elderly",..: 3 3 1 2 3 2
```

Note that in this case, the attributes are ordered in alphabetical order. If not other instructions are provided **R** orders the names in alphabetical order. If, an incremental order of age is desired, it is possible to do so by using the `ordered` instruction, directly into the `factor` command:

```
> age2 <- factor(age,levels=c("Young","Adult","Elderly"),ordered=TRUE)
> age2
[1] Young Young Adult Elderly Young Elderly
Levels: Young < Adult < Elderly
> str(age2)
Ord.factor w/ 3 levels "Young"<"Adult"<..: 1 1 2 3 1 3
#Here it assures that age is factorized as a number
age3 <- as.numeric(age2)
```

Note that now the numerical values are assigned with 1 for 'young', 2 for 'adult' and 3 for 'elderly', which is more intuitive since it is incremental with age. Now, the same procedure is applied to the vector `gender`, where a new vector `gender2` is created using an ordered `factor` command, where the male has numeric value of 1 and the female of 2.

```
> gender2 <- factor(gender,levels=c("M","F"),ordered=TRUE)
> gender2
[1] M M M F F F
Levels: M < F
> str(gender2)
 Ord.factor w/ 2 levels "M"<"F": 1 1 1 2 2 2
```

The actual numeric value (**quantitative**) for the age of the group of people can now be provided, by defining a vector `numage`.

```
> numage=c(9,10,50,80,14,91)
> numage
[1] 9 10 50 80 14 91
> str(numage)
```

```
num [1:6] 9 10 50 80 14 91
```

The vector is now created containing the six ages as numerical value.

5.2 Import Data

There are different ways to import data in R. The most convenient one is reading the data directly from a file or multiple files. For simplicity through this book, a dataset of soil physical and chemical data will be used as a main example. However, other dataset are always presented depending on a specific topic. Clearly the reader is encouraged to apply the exercises to her or his own data. The data presented here were collected during an experimental campaign in the Northern Italian Apennines Mountain Range in the Emilia−Romagna region. In the table 5.1, a subset of a soil dataset is shown.

Table 5.1 Physical and chemical properties of soil profiles.

Soil	horizon	UD	LD	type	thickness	pH_{water}	pH_{KCl}	sand	silt	clay
C1	A1	0	6	epi	6	4.7	3.4	832	130	38
C1	A2	6	16	epi	10	4.9	3.8	745	195	60
C1	B	16	27	endo	11	4.9	3.9	712	219	69
C1	BC	27	30	endo	3	4.1	4	689	219	92
C2	A1	0	6	epi	6	4.3	3.1	756	197	47
...	...	...	...	...	...	...	...	...	...	...

5.3 Open a Data File

As discussed above, it is not convenient to enter the data manually. R provides a variety of option to read a data file.

5.3.1 Setting the working directory

First the working directory should be set by typing the pathway. Remember that you must use the forward slash / in R. This operation can also be performed in RStudio, by selecting the window Session and Set Working Directory.

```
setwd("~/Didattica/R_class_4/exercises/Ch5_Managing_data")
```

By typing this statement (get working directory), the current working directory is displayed.

```
> getwd()
"C:/Users/marco.bittelli.PERSONALE/Documents/Didattica/
R_class_4/exercises/Ch5_Managing_data"
```

The command dir() will list the folders and files contained in the directory. The home directory in Windows R is set using the environment variable R_USER Set this using

Windows (search from the Start Menu for "environment variable"). Whatever you set this to will become what R uses for .

```
> dir()
```

5.3.2 Read and open data files

Data can be stored in different formats. For instance they can be saved in the MS Excel program, or as ASCII file such as comma separated file (.csv) or text file (.txt). As an example the soil dataset is saved as *Horizons.xls*, *Horizons.csv* and *Horizons.txt*.

1. Open an excel file (.xlsx)
2. Open a comma separated file (.csv)
3. Open a text file (.txt or .dat)

5.3.3 Read an Excel file

It is possible to open excel files with the following instruction:

```
      library(readxl)
      setwd("~/Didattica/R_class_4/exercises/Ch5_Managing_data")
  Horizons <- read_excel("data/Horizons.xls")
```

5.3.4 Read a comma separated file

With the command below data are read, using the command `read.csv` that allows for reading the data contained in the data file. This function is used to read data that are organized in a matrix with data contained in each column, as shown in the section above.

```
Horizons <- read.csv("data/Horizons.csv", sep = ",",
check.names = FALSE, header = TRUE,na.strings = c("NA", "NAN"), dec=".")
```

This command contains a series of arguments that can be specified.

1. data directory
2. sep
3. check.names
4. header
5. na.strings
6. decimal points
7. row names

The first row of the data file may contain information about the variable and it is called *header*. If the file contains a header it should be specified by setting the argument `header = TRUE` to TRUE. The argument `sep = ","` indicates the characters that is used to separate the data. In our case it is a colon. If there are missing data then we should specify how they are called in the data file. In this case the symbol for missing data is NA. The instruction *c("NA", "NAN")* is used to concatenate the characters into one vector, allowing for using both NA and NAN as symbols for missing data.

5.3.5 Read a text file

The reading of a text file is very similar, with the instruction `read.table`.

```
income <- read.table("data/Horizons.txt", sep = ";",
check.names = FALSE, header = TRUE,
na.strings = c("NA", "NAN"), dec=".")
```

5.3.6 View and modify Data

After having imported the file into the dataframe `Horizons`, the data can be visualized with the instructions `View` or `edit`. Information about the dataframe is obtained by using the command `str`, which stands for structure. This command will provide information about the number of observations, the number of variables and the datatype of the variables, as shown below.

```
Classes ‘tbl_df’, ‘tbl’ and 'data.frame': 136 obs. of  35 variables:
$ ...1       : chr  "C1" "C1" "C1" "C1" ...
$ horizon    : chr  "A1" "A2" "B" "BC" ...
$ upper depth: num  0 6 16 27 0 6 15 21 0 4 ...
$ lower depth: num  6 16 27 30 6 15 21 30 4 10.5 ...
$ type       : chr  "epi" "epi" "endo" "endo" ...
```

To view the values of a specific variable, the dollar sign must be written after the name of the dataframe. For instance the variable sand (in fraction over thousands) is shown below.

```
Horizons$sand
[1] 832 745 712 689 756 772 743 738 719 756 610 540 670 670 643 754
547 756 742 868 784 821 505 758 715 740 678 657 838 771 716 716 713
```

it is possible to print all the value for this vector. The command `edit()` open an editor windows

```
edit(Horizons)
```

that allows for performing a few simple operations. In many cases it can be useful to have an index column (or an identification number ID) as a first left column with simple incremental numbers for each record (1,2,3..., 32561). A simple way is to type

```
Horizons$ID <- seq.int(nrow(Horizons))
> income$ID
[1] 1 2 3 4 5 6 7 8 ...
```

that includes a variable `ID` to the `data.frame`. Typing `Horizons$ID` at the console, returns incremental values from 1 to 32561.

A useful command is `head()`, that provides the header of the dataframe.

```
> head(Horizons)
```

It is possible to change the data type of a variable with the instruction

```
> Horizons$ID <- as.integer (Horizons$ID)
```

5.3.7 Attaching

R objects that reside in other R objects can require a lot of typing to access. For example in the example above to refer to a variable `sand` in the dataframe, one could type `Horizons$age`. In some cases is not desirable to type long names and using the dollar sign all the time. The `attach()` function in R can be used to make objects within dataframes accessible in R with fewer keystrokes. By typing

```
> attach(Horizons)
> names(Horizons)
```

The instruction `names()` lists all the variable names. It is then possible to `detach()` the data. The instruction `attach()` can lead to confusion since duplication of variables can occur, so it should be used with care.
A useful command is the `dim`ension command. It returns the dimension of a vector or dataframe. In this case the dimension of the dataframe income, is of 32561 rows and 15 columns.

```
> dim(Horizons)
[1] 136 35
```

Another command that provides information is the `lenght()`, that returns the number of observations in a vector or in variable

```
> length(Horizons$sand)
[1] 136
```

5.4 Managing Dates

Often in data analysis we are working with time series. To properly analyze time series an efficient managing of date formats is necessary. The first step is to convert a string or a number into a date format. Dates can be imported from character, numeric, POSIXlt, and POSIXct formats using the `as.Date` function from the `base` package. In the example below, two dates are read and assigned to the vector `dates`. As we have described above R does not recognize this string as a date, but simply as a character value `char`. Indeed by inquiring about the structure of the vector `dates`, it will return a vector of two elements of character type.

```
> str(dates)
chr [1:2] "05/27/84" "07/07/05"
```

The `as.Date` function converts this vector character into a date format:

```
dates <- c("05/27/84", "07/07/05")
formattedDates <- as.Date(dates, "%m/%d/%y")
```

Now the new vector `formattedDates` is a Date variable.

```
> str(formattedDates)
Date[1:2], format: "1984-05-27" "2005-07-07"
```

Another format may have letters as:

```
dates <- c("May 27 1984", "July 7 2005")
formattedDates <- as.Date(dates,
format = "%B %d %Y")
```

The ISO 8601 international standard format %Y-%m-%d is returned

```
> formattedDates
[1] "1984-05-27" "2005-07-07"
```

When importing data from Excel the format may be a numeric value. It is still possible to import it and convert it.

```
dates <- c(30829, 38540)
formattedDates <- as.Date(dates,origin = "1899-12-30")
```

Since Excel defines an origin for the time, which is 1899, it must be specified.

```
> formattedDates
[1] "1984-05-27" "2005-07-07"
```

5.4.1 Change Date Format

The format can be changed to a desired format as:

```
format(formattedDates,"%a %b %d")
[1] "Sun May 27" "Thu Jul 07"
```

which returs the day of the day of the week, the month and the day. Below is reported a table with the most common formats:

Table 5.2 Conversion specifications for dates

Conversion	Description	Example
%a	Abbreviated weekday	Sun, Thu
%A	Full weekday	Sunday, Thursday
%b or %h	Abbreviated month	May, Jul
%B	Full month	May, July
%d	Day of the month 01-31	27, 07
%j	Day of the year 001-366	148, 188
%m	Month 01-12	05, 07
%U	Week 01-53 with Sunday as first day of the week	22, 27
%w	Weekday0-6Sunday is 0	0, 4
%W	Week 00-53 with Monday as first day of the week	21, 27
%x	Date, locale-specific	
%y	Year without century 00-99	84, 05
%Y	Year with century on input:	
	00 to 68 prefixed by 20, 69 to 99 prefixed by 19	1984, 2005
%C	Century	19, 20
%D	Date formatted %m/%d/%y	05/27/84, 07/07/05
%u	Weekday 1-7 Monday is 1	7, 4

When importing data with only two digits for the years, **R** assumes that years 69 to 99 are 1969-1999, while years 00 to 68 are 2000–2068 (probably subject to change in future versions of R). Often, this is not what it is intended to have happen. One solution it provides is to assume all dates **R** is placing in the future are from the previous century. Here the dates from 1984, 2005 and 2020 are imported. However, it is not clear from what century. The instruction would return that 84 is 1984, 05 is 2005 and 20 is 2020.

```
dates <- c("05/27/84", "07/07/05", "08/17/30")
formattedDates <- as.Date(dates, "%m/%d/%y")
> formattedDates
[1]"1984-05-27" "2005-07-07" "2030-08-17"
```

With the instruction below it is possible to assign the date before today to the previous century:

```
correctCentury <- as.Date(ifelse(formattedDates > Sys.Date(),
format(formattedDates, "19%y-%m-%d"),
format(formattedDates)))
```

```
> correctCentury
[1] "1984-05-27" "2005-07-07" "1930-08-17"
```

with the `ifelse` it is selected if the dates are larger that the system date, then they should be formatted as from the 1900 (previous century), Indeed today is:

```
> Sys.Date()
[1] "2020-10-05"
```

which is October, 5, 2020. Therefore the year 2030 was transformed into 1930. Obviously this is a solution that would not work with dates obtained from simulation into the future such as climate scenarios.

5.4.2 Use dates and endpoints to select variables

When working with time series, various statistical analysis are performed over individual time intervals such as days, months or year. For instance when analysing weather data, cumulative annual precipitation is of interest. In this example precipitation data from 1961 to 2018, from five experimental station in the Emilia Romagna region are analyzed. The sign is used to comment section of the code. The code is presented below

```
#CODE Ch5_2.R
### provides functionality for working with time series
library(xts)
library(lubridate) ### manage dates
library(ggplot2) ### allows plotting
library(dplyr)
library(readxl) ### import excel file
#The excel file Prec_ER_daily.xlsx is opened, the structure of the
#newly created dataframe is analysed and the data are visualized.

setwd("~/Didattica/R_class_4/exercises/Ch5_Managing_data")
Prec_ER_daily <- read_excel("data/Prec_ER_daily.xlsx")
str(Prec_ER_daily)
View(Prec_ER_daily)

# Only data from the Piacenza station are analyzed

Prec_ER_daily_cadriano <- Prec_ER_daily[,c("date","cadriano")]
str(Prec_ER_daily_cadriano)

#save prec in a single vector (prec_cadriano) and read the first year

prec_cadriano <- Prec_ER_daily_cadriano$cadriano
firstYear <- year(Prec_ER_daily_cadriano$date[1])
firstYear

#plot(Prec_ER_daily_cadriano$date,Prec_ER_daily_cadriano$cadriano)
```

```
#CUMULATIVE BY DAY

date<-as.Date(Prec_ER_daily_cadriano$date)
start <- date[1]
end <- date[365]
start
end
theDate
as.Date(theDate)
theDate<-start
theDate
vec_date<-vector()
precip<-vector()
prec_daily<-0
cumprec<-0
cumprec_vec<-vector()
i<-1

while (theDate <= end){
       vec_date[i]<-theDate
       prec_daily=prec_cadriano[i]
       cumprec<-cumprec + prec_cadriano[i]
       cumprec_vec[i]<-cumprec
       output<-c(i,prec_daily,cumprec)
       print(output)
       theDate <-theDate + days(1)
       i<-i+1
}

#plotting
plot(as.Date(vec_date),cumprec_vec,type="l")

#CUMULATIVE BY YEAR

# it reads the array of endpoints and subset by endpoints
#indexArray <- endpoints(Prec_ER_daily_cadriano$date, on = "years")
#print(indexArray)
#Then is runs a cycle over the years, for each year it reads the
#first index (first) and last (last) index from the endpoints.
#The first index is increments by one since the endpoints is fixed
#on the last value of the previous year.
#By using the indexes it selected the data subset for that year and save
#it into the variable prec.
#From these data various statistics (such as the cumulative value)
#are performed on the array. The command c() is used
```

```
#to concatenate the values.

sumprec<-vector()
vec_year<-vector()
indexArray <- endpoints(Prec_ER_daily_cadriano$date, on = "years")
print(indexArray)

for (year in 1961:2018)
{
       i = year - firstYear + 1
       first = indexArray[i]+1
       last = indexArray[i+1]
       prec = prec_cadriano[first:last]
       vec_year[i]<-year
       sumprec[i]<-sum(prec)
       output<- c(year, sumprec)
       print(as.integer(output))
}
plot(vec_year,sumprec,type="h",xlab="Year",
ylab="Cumulative Precipitation [mm]")
```

The results of this program are plotted in Fig.5.1

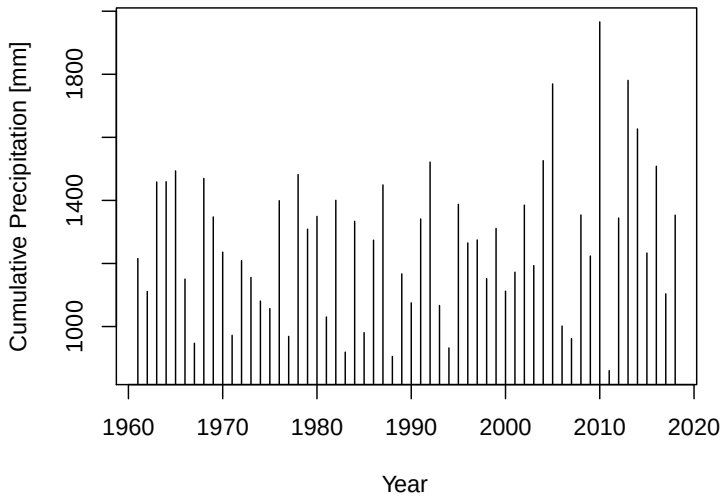


Fig. 5.1 Yearly cumulative precipitation at the Cadriano station in Emilia Romagna, Italy.

Here another example is presented where dates are in the data file. The file is organized as shown below. The first column contains the date (month, day and year), the second column the time of the day (the data are collected every ten minutes), and then the remaining columns contain soil water content data at 0.2, 0.4, 0.6, 0.8, 1, 1.2 and 1.4 meters below the ground surface. Data were collected from an experimental station as presented in

```
Date      Hour WC_0.2 WC_0.4 WC_0.6 WC_1.0 WC_1.2 WC_1.4
3/27/2012 15:20 0.311 0.324 0.374 0.257 0.364 0.296
3/27/2012 15:30 0.302 0.326 0.383 0.258 0.356 0.301
3/27/2012 15:40 0.316 0.323 0.401 0.254 0.356 0.301
3/27/2012 15:50 0.309 0.331 0.382 0.256 0.358 0.301
3/27/2012 16:00 0.299 0.327 0.367 0.262 0.363 0.303
3/27/2012 16:10 0.317 0.322 0.376 0.263 0.359 0.304
3/27/2012 16:20 0.286 0.324 0.458 0.262 0.358 0.301
3/27/2012 16:30 0.309 0.331 0.378 0.262 0.359 0.302
3/27/2012 16:40 0.324 0.331 0.383 0.267 0.359 0.303
3/27/2012 16:50 0.310 0.333 0.389 0.267 0.363 0.301
3/27/2012 17:00 0.312 0.326 0.493 0.265 0.356 0.299
3/27/2012 17:10 0.314 0.334 0.181 0.264 0.363 0.297
3/27/2012 17:20 0.309 0.332 0.396 0.261 0.371 0.299
```

Managing files with dates may be cumbersome since they can be presented in a variety of different ways. In the example below, a data file with weather data is opened with the instruction described above.

```
#CODE Ch5_3.R
library(xts)
library(lubridate)
library(ggplot2)
library(mice)
library(tseries)

setwd("~/Didattica/R_class_4/exercises/Ch5_Managing_data")
myfield1 <- c("character", rep("numeric", 15))

suolo <- read.table("data/Suolo_Montue_Giugno2019.csv",
sep = ";", check.names = FALSE, header = T,
colClasses = myfield1, na.strings = c("NA", "NAN"))

myfield2 <- c("character", rep("numeric", 7))

meteo <- read.table("data/Meteo_Montue_Giugno2019.csv",
sep = ";", check.names = FALSE, header = T,
colClasses = myfield2, na.strings = c("NA", "NAN"))
```

```
suolo$Data <- gsub("/","-",suolo$Data)
#replace the symbol "/" with the symbol "-" in the column Data
datastamp <- parse_date_time(suolo$Data, orders="mdy HM")

suolo$Data <- datastamp

## Here the axis are controlled

lims <- as.POSIXct(strptime(c("2013-07-21 00:00","2013-11-23 00:00"),
format = "%Y-%m-%d %H:%M"))

base_plot_soil<-ggplot(data = suolo,
aes(x = suolo$Data, y = suolo$WC_0.4))+
geom_line(color = "blue", size = 1) + ylim(0,0.6)

base_plot_soil +
scale_x_datetime(limits = lims,date_labels = "%d/%m/%y %H:%M")


#Use of endpoints to plot one data a day

iii <- endpoints(suolo$Data, on = "days")

lims <- as.POSIXct(strptime(c("2013-07-21 00:00","2014-11-23 00:00"),
format = "%Y-%m-%d %H:%M"))

base_plot2_soil<- ggplot(data = suolo[iii, ],
aes(x = suolo$Data[iii], y = suolo$WC_0.2[iii]))+
geom_line(color = "blue", size = 2)+ ylim(0,0.6)

base_plot2_soil
+ scale_x_datetime(limits = lims,date_labels = "%d/%m/%y %H:%M")

### Here the meteo data are plotted

meteo$Data <- gsub("/","-",meteo$Data)
datastamp <- parse_date_time(meteo$Data, orders="mdy HM")

meteo$Data <- datastamp

## Plot Average Temp
lims <- as.POSIXct(strptime(c("2013-07-21 00:00","2014-11-23 00:00"),
format = "%Y-%m-%d %H:%M"))
base_plot1_meteo<-ggplot(data = meteo, aes(x = meteo$Data, y = meteo$T_air))+
geom_line(color = "#00AFBB", size = 2)
```

```
base_plot1_meteo + scale_x_datetime(limits = lims,date_labels = "%d/%m/%y %H:%M")


## Plot Average Net radiation
base_plot2_meteo<-ggplot(data = meteo,
aes(x = meteo$Data, y = meteo$Radsol_netta))+
geom_line(color = "#00AFBB", size = 2)
base_plot2_meteo +
scale_x_datetime(limits = lims,date_labels = "%d/%m/%y %H:%M")

## Plot Precipitation
base_plot3_meteo<-ggplot(data = meteo, aes(x = meteo$Data, y = meteo$Pioggia))+
geom_line(color = "#00AFBB", size = 2)
base_plot3_meteo +
scale_x_datetime(limits = lims,date_labels = "%d/%m/%y %H:%M")

## Define Endpoints for Meteo
jjj <- endpoints(meteo$Data, on = "days")

ggplot(data = meteo[jjj, ], aes(x = meteo$Data[jjj],
y = meteo$Pioggia[jjj]))+ geom_line(color = "#00AFBB", size = 2)
```

Note that in the instruction `read.table`, one of the argument is `colClasses`. With this instruction a vector of classes is specified to be assumed for the columns. In this example it is specified that the first column is a character (Data) and then there are 7 columns of numeric data.

```
       myfield2 <- c("character", rep("numeric", 7))
```

The function `gsub` is used to substitute characters in a given vector. Here the backslash in the date is replaced with a minus sign.

```
       suolo$Data <- gsub("/","-",suolo$Data)
```

The next instruction is used to parse an input vector into POSIXct date-time object. The `POSIXlt` and `POSIXct` represents calendar dates and times.

```
       datastamp <- parse_date_time(suolo$Data, orders="mdy HM")
       suolo$Data <- datastamp
```

First, the function allows `parse_date_time()` for specification of the order in which the formats occur without the need to include separators and % prefix. Such a formatting argument is refered to as "order". Second, it allows the user to specify several format-orders to handle heterogeneous date-time character representations.

5.5 Subsetting

In data analysis is often necessary to select and exclude variables and observations. There are many different options to select a subset of data. The simplest one is to use brakets. By using the example above with the income data.
With the use of square brakets, it is possible to select an interval for a specific variable. In this example, observations 2 to 20 are selected for the variable `pH_water`

```
> Horizons$pH_water[2:20]
[1] "4.9000000000000004" "4.9000000000000004" "4.0999999999999996"
[4] "4.2999999999999998" "4.2999999999999998" "4.2999999999999998"
...
```

As it was presented in the example about the cumulative precipitation in Emilia Romagna. A subset of the dataframe is obtained by using the square brakets, where the first element are the rows and the second elements are the columns. In the example below all the rows are selected, while only the columns `date` and `cadriano` are selected:

```
Prec_ER_daily_cadriano <- Prec_ER_daily[,c("date","cadriano")]
```

Another powerful option is to use the library `dplyr` that allows to use the function subset in a very flexible fashion.

```
#CODE Ch5_4.R
#SUBSETTING
library(dplyr)

library(readxl)
setwd("~/Didattica/R_class_4/exercises/Ch5_Managing_data")
Horizons <- read_excel("data/Horizons.xls")
View(Horizons)
edit(Horizons)
str(Horizons)

Horizons$pH_water[2:20]

newdata <- subset(Horizons, pH_water <= 5 & sand < 500 & type == "epi")
edit(newdata)
```

The `subset` can be used to class matrix or dataframe. The instruction is `subset(x, subset, select, drop = FALSE, ...)`, where `x` is the object to be subsetted, `subset` is the logical expression indicating elements or rows to keep: missing values are taken as false. `select` expression, indicating columns to select from a data frame. further arguments to be passed to or from other methods.

To select one of more columns:

```
newdata <- subset(Horizons, select= c("pH_water","sand"))
```

5.6 Merging

In many cases, there are many dataset with different information on the same statistical properties. In the example below it can be useful to combine information about soil profiles (saved in the dataset `Profiles`), with information contained with information provided in the `Horizons`. The merging can be performed by exploiting the variable `Factor` and the function `merge`.

In this section, data collected from different soil profiles are presented. Here the data files are imported with the instruction `read.csv`.

```
setwd("~/Didattica/R_class_4/exercises/Ch5_Managing_data")
Profiles <- read.csv(file = "/data/Profiles.csv", header = TRUE, sep = ",")
Horizons <- read.csv(file = "/data/Horizons.csv")
```

The data recorded in the `Profiles` are information about six soil profiles (C1,C2,C3,...) corresponding to the ones in the file `Horizons`. The file `Profiles` contains information about geographic location (latitude, longitude and altitude) as well as vegetation coverage, Calcium and Carbon stock, which are not contained in the file `Horizons`

```
  PROF COOR_X COOR_Y ALT VEG LIT Ca_stock C_stock_30 C_stock_0
1 C1 636811 4895267 1710 PRATO CEV 2.4 177 44
2 C2 637080 4895192 1697 CON CEV 2.5 157 93
3 C3 637557 4896855 1700 FAG CEV 6.5 178 65
4 C4 635826 4895003 2027 PRATO MOD 4.6 99 52
5 C5 635892 4895230 1939 MIRT MOD 2.7 193 80
6 C6 635763 4895302 1901 PRATO MOD 3.2 69 58
```

The data recorded in the `Horizon` are information regarding the profiles and horizons

With the following instruction the dataframe `Horizons` and `Profiles` are merged into the dataframe `Horizons`.

```
#CODE Ch5_6.R
#MERGING
library(readxl)
setwd("~/Didattica/R_class_4/exercises/Ch5_Managing_data")
Profiles <- read.csv(file = "data/Profiles.csv", header = TRUE, sep = ",")
Horizons <- read.csv(file = "data/Horizons.csv")

edit(Profiles)
edit(Horizons)

Horizons_merged<-merge(x = Horizons,y = Profiles,by.x = "Soil",by.y = "Soil")

edit(Horizons_merged)
```

The arguments `x` and `y` specify the dataframe to merge, while `by.x` and `by.y` specify the mutual variables in the different dataframe. The operation adds columns to the dataframe coming from the other dataset.

6
Data Visualization

Before starting with any kind of statistical analysis, data should be looked at. Graphs and charts allows for exploration and learning about the structure and nature of the data. For instance, they allow for understanding the range of a variable, possible outliers, intervals. Moreover, effective data visualizations make it easier to communicate ideas and findings to other people. Therefore before introducing basic and more advanced statistical concepts, data visualization is discussed. The general package for visualization in R is `plot`. A more advanced package is `ggplot`. For many graphs `plot` already provides enough option to obtain a high quality graph that can be used for thesis, scientific papers, technical reports.

There are a variety of options to plot and represent data.

6.1 Create graphs using plot

The package `plot` is the generic package to create graph. The basic instruction is

```
plot(x,y,..)
```

where `x` and `y` are the two variables to plot and the dots indicate options. In the example below, a radargram collected with Ground Penetrating Radar (GPR) is plotted. For details about this geo−physical method see (**?**). As described above the instruction `attach` is used to attach the database to the **R** search path. This allows for searching the database by simply giving the variable names, without having to refer to the dataframe each time.

```
#CODE Ch6_1.R
#PLOTTING
setwd("~/Didattica/R_class_4/exercises/Ch6_Data_Visualization")


GPR <- read.table("data/gpr.dat",
sep = "\t", header = T, na.strings = c("NA", "NAN"))

edit(GPR)
attach(GPR)

plot(ns, Amplitude1,
xlim=c(0,60) , ylim=c(-40000,40000),
type="l",
```

```
pch=1,
cex=1,
col=1,
xlab="Time [ns]", ylab="Amplitude",
main="GPR radargram"
)
```

The default **R** plot `pch` symbol is 1, which is an empty circle. This value can be changed to `pch` = 19 (solid circle) or to `pch`= 21 (filled circle) if points are used. To change the color and the size of points, use the following arguments: col : color (hexadecimal color code or color name). In this graph the `col=1` is for black. The instruction `type` = ”l” refers to lines. Other types are ”p” for points, ”b” for both, ”h” for histograms.

To plot two data series in the same plot, the following instruction can be used. The instruction `lines` is used to add the new dataseries.

```
plot(GPR$ns, GPR$Amplitude1,xlim=c(0,60) , ylim=c(-40000,40000),col="black",
type="l",xlab="Time [ns]", ylab="Amplitude",
main="GPR radargrams")
lines(GPR$ns, GPR$Amplitude2, col="red",type="l")
```

The results are shown in Fig.6.1.

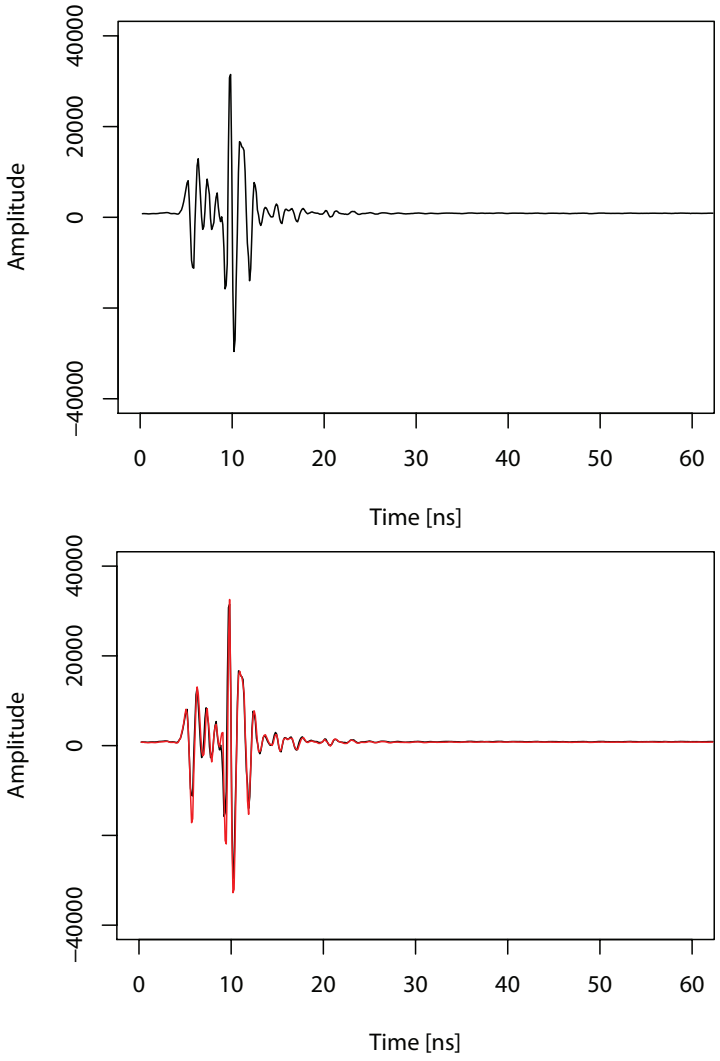


Fig. 6.1 Example of data visualization. Radargrams obtained from GPR. Top: one data series, bottom: two data series.

6.1.1 Plot modification

The plot can be modified by adding captions, determining the range of x and y, choosing different plot type. In the example above the range of x and y was chosen, as well as the color of the line, the captions and the graph title.

```
plot(GPR$ns, GPR$Amplitude1,xlim=c(0,60), ylim=c(-40000,40000),col="black",
       type="l",xlab="Time [ns]", ylab="Amplitude",main="GPR radargrams")
```

Figure 6.2 shows the numbers to be used to plot different kind of symbols for points. The instruction to select the symbols is `pch =` . For instance `pch = 0` is used to select an empty square.

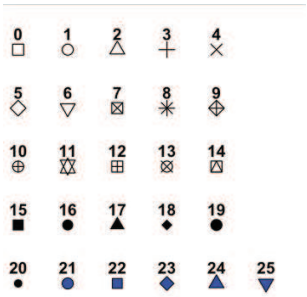


Fig. 6.2 List of plot symbols

6.2 Create graphs using ggplot

Generally, the instruction starts with `ggplot()`, supply a dataset and aesthetic mapping (with `aes()`). You then add on layers (like `geom_point()` or `geom_histogram()`), scales (like `scale_colour_brewer()`), faceting specifications (like `facet_wrap()`) and coordinate systems (like `coord_flip()`). In this first example, data from the data.frame `income` are plotted for a first observation of data.

```
ggplot(data = income,aes(x=ID[], y = age[]))+ geom_point()
+ xlim(0,100) + ylim(10,70)
```

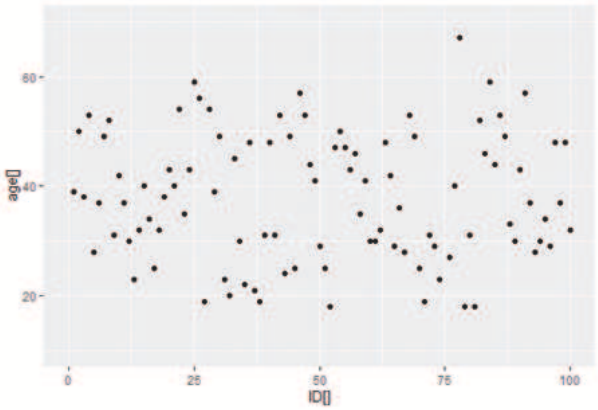


Fig. 6.3 Example of data visualization

The syntax of ggplot is straightforward. The `data=data.frame` is to provide the data.frame to ggplot. Then `aes` stands for the aesthetic mappings to use for plot.

Then we the addition sign (+), other options can be added to the plot. For instance, here the plots is created with points and the limits of the x and y axis are defined. Figure 6.4 shows the output of the instruction above, with the incremental `ID` number and the age. This figure is not very informative since it simply list the age of the group elements.

6.2.1 Plotting data: examples

In this section and example on how to plot data collected in the experimental station for soil and weather is presented. The **R** code is show below.

```
#CODE Ch5_3.R
library(xts)
library(lubridate)
library(ggplot2)
library(mice)
library(tseries)

setwd("~/Didattica/R_class_4/exercises/Ch5_Managing_data")
myfield1 <- c("character", rep("numeric", 15))

suolo <- read.table("data/Suolo_Montue_Giugno2019.csv",
sep = ";", check.names = FALSE, header = T,
colClasses = myfield1, na.strings = c("NA", "NAN"))

myfield2 <- c("character", rep("numeric", 7))

meteo <- read.table("data/Meteo_Montue_Giugno2019.csv",
sep = ";", check.names = FALSE, header = T,
colClasses = myfield2, na.strings = c("NA", "NAN"))


suolo$Data <- gsub("/","-",suolo$Data)
#replace the symbol "/" with the symbol "-" in the column Data
datastamp <- parse_date_time(suolo$Data, orders="mdy HM")

suolo$Data <- datastamp

## Here the axis are controlled

lims <- as.POSIXct(strptime(c("2013-07-21 00:00","2013-11-23 00:00"),
format = "%Y-%m-%d %H:%M"))

base_plot_soil<-ggplot(data = suolo, aes(x = suolo$Data, y = suolo$WC_0.4))+
geom_line(color = "blue", size = 1) + ylim(0,0.6)

base_plot_soil + scale_x_datetime(limits = lims,date_labels = "%d/%m/%y %H:%M")


#Use of endpoints to plot one data a day

iii <- endpoints(suolo$Data, on = "days")
```

```
lims <- as.POSIXct(strptime(c("2013-07-21 00:00","2014-11-23 00:00"),
format = "%Y-%m-%d %H:%M"))

base_plot2_soil<- ggplot(data = suolo[iii, ],
aes(x = suolo$Data[iii], y = suolo$WC_0.2[iii]))+
geom_line(color = "blue", size = 2)+ ylim(0,0.6)

base_plot2_soil + scale_x_datetime(limits = lims,date_labels = "%d/%m/%y %H:%M")

### Here the meteo data are plotted

meteo$Data <- gsub("/","-",meteo$Data)
datastamp <- parse_date_time(meteo$Data, orders="mdy HM")

meteo$Data <- datastamp

## Plot Average Temp
lims <- as.POSIXct(strptime(c("2013-07-21 00:00","2014-11-23 00:00"),
format = "%Y-%m-%d %H:%M"))
base_plot1_meteo<-ggplot(data = meteo, aes(x = meteo$Data, y = meteo$T_air))+
geom_line(color = "#00AFBB", size = 2)
base_plot1_meteo +
scale_x_datetime(limits = lims,date_labels = "%d/%m/%y %H:%M")


## Plot Average Net radiation
base_plot2_meteo<-ggplot(data = meteo,
aes(x = meteo$Data, y = meteo$Radsol_netta))+
geom_line(color = "#00AFBB", size = 2)
base_plot2_meteo +
scale_x_datetime(limits = lims,date_labels = "%d/%m/%y %H:%M")

## Plot Precipitation
base_plot3_meteo<-ggplot(data = meteo,
aes(x = meteo$Data, y = meteo$Pioggia))+
geom_line(color = "#00AFBB", size = 2)
base_plot3_meteo +
scale_x_datetime(limits = lims,date_labels = "%d/%m/%y %H:%M")

## Define Endpoints for Meteo
jjj <- endpoints(meteo$Data, on = "days")

ggplot(data = meteo[jjj, ],
aes(x = meteo$Data[jjj], y = meteo$Pioggia[jjj]))+
```

```
geom_line(color = "#00AFBB", size = 2)
```

R provides the possibility of including `endpoints`, with the following instruction:

```
iii <- endpoints(suolo$Data, on = "days")
```

```
## Here a vector giorni is created with the values of the dates at the endpoints
giorni <- date(suolo$Data[iii])
colril <- c(2:22)
```

It extract index values of a given `xts` object corresponding to the last observations given a period specified by `on`. When the vector endpoints is specified, it returns

```
> iii
[1] 0 52 196
340 484 628 772
916 1060 1204 1348
1492 1636 ... ...
```

where the numbers are the index of the data corresponding to the endpoints of the data for a given day.

```
ggplot(data = suolo[iii, ],
aes(x = suolo$Data[iii], y = suolo$WC_0.2[iii]))+
geom_line(color = "#00AFBB", size = 2)
```

Now that the endpoints have been created, it is possible to plot the variables from the `data.frame` suolo, only at the endpoints.

6.2.2 Managing dates

`ggplot2` supports three date and time classes: `POSIXct`, `Date` and `hms`. Depending on the class at hand, axis ticks and labels can be controlled by using

```
scale_*_date
scale_*_datetime
scale_*_time,
```

respectively. Depending on whether one wants to modify the x or the y axis.

```
scale_x_*
scale_y_*
```

are to be employed.

In the code below the instruction parse_date_time is used to read the character vector `Data` and transform it into a `POSIXct` object. This option is very useful since **R** does not recognize a string as a date, therefore it must be instructed. The option `orders` allows for specification of the order of days, month, years, hours, minutes, etc. The instruction below is used to save on the vector `Data` the new format for the dates. The complete program is shown below.

```
#replace the symbol "/" with the symbol "-" in the column Data
suolo$Data <- gsub("/","-",suolo$Data)
datastamp <- parse_date_time(suolo$Data, orders="mdy HM")
suolo$Data <- datastamp
```

```
library(xts)
library(lubridate)
library(ggplot2)
library(mice)
library(tseries)

setwd("~/Didattica/R_class_3/exercises/OpenDataFiles")

myfield1 <- c("character", rep("numeric", 15))

suolo <- read.table("~/Didattica/R_class_3/exercises/
OpenDataFiles/data/Suolo_Montue_Giugno2019.csv",
sep = ";", check.names = FALSE, header = T,
colClasses = myfield1, na.strings = c("NA", "NAN"))

myfield2 <- c("character", rep("numeric", 7))

meteo <- read.table("~/Didattica/R_class_3/exercises/
OpenDataFiles/data/Meteo_Montue_Giugno2019.csv",
sep = ";", check.names = FALSE, header = T,
colClasses = myfield2, na.strings = c("NA", "NAN"))


suolo$Data <- gsub("/","-",suolo$Data) #replace the symbol
"/" with the symbol "-" in the column Data
datastamp <- parse_date_time(suolo$Data, orders="mdy HM")

suolo$Data <- datastamp


## Here the axis are controlled

lims <- as.POSIXct(strptime(c("2013-07-21 00:00","2013-11-23 00:00"),
format = "%Y-%m-%d %H:%M"))

base_plot_soil<-ggplot(data = suolo,
aes(x = suolo$Data, y = suolo$WC_0.4))+
geom_line(color = "blue", size = 1) + ylim(0,0.6)

base_plot_soil + scale_x_datetime
```

```
(limits = lims,date_labels = "%d/%m/%y %H:%M")
###################################

#Use of endpoints to plot one data a day

iii <- endpoints(suolo$Data, on = "days")

lims <- as.POSIXct(strptime(c("2013-07-21 00:00","2014-11-23 00:00"),
format = "%Y-%m-%d %H:%M"))

base_plot2_soil<- ggplot(data = suolo[iii, ],
aes(x = suolo$Data[iii], y = suolo$WC_0.2[iii]))+
geom_line(color = "blue", size = 2)+ ylim(0,0.6)

base_plot2_soil +
scale_x_datetime(limits = lims,date_labels = "%d/%m/%y %H:%M")


### Here the meteo data are plotted

meteo$Data <- gsub("/","-",meteo$Data)
datastamp <- parse_date_time(meteo$Data, orders="mdy HM")

meteo$Data <- datastamp

## Plot Average Temp
lims <- as.POSIXct(strptime(c("2013-07-21 00:00","2014-11-23 00:00"),
format = "%Y-%m-%d %H:%M"))
base_plot1_meteo<-ggplot(data = meteo, aes(x = meteo$Data, y = meteo$T_air))+
geom_line(color = "#00AFBB", size = 2)
base_plot1_meteo + scale_x_datetime(limits = lims,date_labels = "%d/%m/%y %H:%M")


## Plot Average Net radiation
base_plot2_meteo<-ggplot(data = meteo,
aes(x = meteo$Data, y = meteo$Radsol_netta))+
geom_line(color = "#00AFBB", size = 2)
base_plot2_meteo + scale_x_datetime(limits = lims,date_labels = "%d/%m/%y %H:%M")

## Plot Precipitation
base_plot3_meteo<-ggplot(data = meteo,
aes(x = meteo$Data, y = meteo$Pioggia))+
geom_line(color = "#00AFBB", size = 2)
base_plot3_meteo + scale_x_datetime(limits = lims,date_labels = "%d/%m/%y %H:%M")
```

```
## Define Endpoints for Meteo
jjj <- endpoints(meteo$Data, on = "days")

ggplot(data = meteo[jjj, ], aes(x = meteo$Data[jjj], y = meteo$Pioggia[jjj]))+
  geom_line(color = "#00AFBB", size = 2)
```

An example of the output for air temperature is shown below.

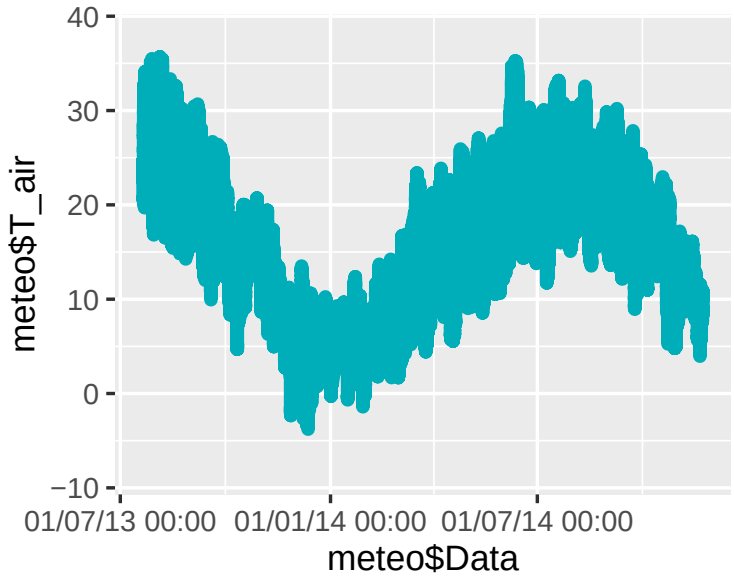


Fig. 6.4 Example of data visualization for air temperature

7
Probability

7.1 Definition

The definition of ”probability” is closely linked to the intuitive definition of a random (or random) event. An event is random when its occurrence cannot be predicted in a deterministic way, due to an imperfect knowledge of the conditions that lead or not to its occurrence. Let’s clarify with a classic example: the toss of a coin produces the ”heads” event or the ”tails” event in a way that cannot be predicted: the result of the toss of a coin (or die, or the drawing of a card from a deck) is a random event. In reality, the event itself would be perfectly determined from a physical point of view: if the coin were thrown under the exact same conditions, the result would always be the same; therefore the event is random due to our ”ignorance” about the initial conditions. More complex examples can be given. The exact duration of a train journey is a random event, as it depends on ”accidental” factors which are, as such, not predictable. The weight of an object is random as, due to accidental errors, the scale produces a different result in a series of repeated measurements.

The definition of probability is still a controversial topic, which divides two different schools of thought. For our purposes, it is sufficient to introduce the concept of probability in the simplest situations, and then generalize the definition.

It is intuitive to assign a value of 0 to an event that cannot occur. Just as, in everyday language, we are used to assigning a probability 1 to a certain event (for example, the probability that the height of a human being is less than 3 meters is 100%, i.e. 1). To define the probability in intermediate cases, it is necessary to introduce some auxiliary concepts:

7.1.1 Definition of probability in terms of relative frequency

The definition of probability in terms of

$$P = \frac{M}{N}$$

and it can be calculated as proportion of favoreable cases (M) with respect to the total number of cases (N), makes sense only when the events are reduced to a ”pattern of cases” and, moreover, are finite in number. This situation rarely occurs in practice, and it is therefore necessary to introduce other definitions of probability. The analysis of this equation suggests a possible definition, even when events do not form a complete class etc. Let’s think, for example, of throwing a dice: even if we do not resort to any symmetry property (the one that allows us to assign p = 1/6 to the output of

each single face), supposing we roll the die a large number of times and to record the frequency of each result on a sheet, we will have obtained at the end of a total number of throws N1 times one, N2 times two,... .., N6 times six. It seems natural to define the probability of the various results through their relative frequency N_1/N, N_2/N,... .., N_6/N. We observe, in fact, that the relative frequency is a number between 0 and 1 and that it is clearly indicative of the "probability" of obtaining a certain result (if, for example, in 100,000 throws the number 1 never comes out, we can begin to think that the die is loaded).

The above is in summary the content of a theorem proved by Bernoulli and known as the "law of large numbers" described below, for which the relative frequency of an event tends to its probability when the number of "proofs" tends to infinity as shown above.

7.1.2 Axiomatic Definition of probability

The above definition is not entirely satisfactory. The main problem is that the definition in terms of relative frequency requires to be able to repeat an experiment a large number of times (even if only "conceptually"), and this is not always possible or reasonable. From a mathematical point of view, probability can be defined as follows: given an event E, we assign to E a number P(E) which we call probability of E, which satisfies the following conditions:

- P(E) is greater than or equal to 0
- P(E) = 1 only if E occurs with certainty
- Given two incompatible events A and B, the probability of their sum is equal to the sum of their probabilities: P(A + B) = P(A) + P(B)

Incompatible events are events that cannot occur at the same time. For instance, when rolling a die and event A is to roll a 1, 2, or 3 and event B is to roll a 4, 5, or 6, then events A and B are incompatible. It is not possible to simultaneously obtain both a 2 and a 5 when rolling a dice. Only one of the two events can occur. These properties and the axioms will be defined in more details in the sections below.

7.2 Sample Space and Events

In random experiments the list of all possible outcomes is called the **sample space**, usually detoned by $\mathcal{S}$. Sample spaces can be finite or infinite, and the elements can be continuos or discrete. $\mathcal{F}$ is a set of subsets or better a *family* of sets or, with the event nomenclature, a family of events with given properties. A few examples are given here for the rolling of a die:

- What is the change of getting 1 when rolling a die. Assuming that the due is fair, the change is one in size, therefore 1/6.
- What is the chance of getting a 1 or 2. One and two are two of the equally possible one sixths, therefore the change of getting 1 or 2 is $2/6 = 1/3$.
- What is the change of getting 1, 2, 3, 4, 5 or 6. Obviusly it is 100 %.

- What is the chance of not rolling a 6? This change can be obtained by knowing that the change of getting a 6 is 1/6, therefore the 16.6 %, as a consequence the change of not getting a 6 is 100 % - 16.6 % which is 83.3 % (equivalent to 5/6).

Further examples could be presented, including also the throwing of multiple dice. From a mathematical standpoint to describe changes and probability, set theory is employed. We refer to classic books for set theory, where the concept of set and its properties are described in details. For instance, the throws of dice is an example of events:

- A: "The dice displays an even number", identified by the set $A = \{2, 4, 6\}$
- B: "The dice displays a number < 3. $B = \{1, 2\}$
- C: "Event = 6" . $C = 6$

The totality of events is named $\mathcal{S}$. The set $\{A, B, \{6\}, \mathcal{S}\}$ is a family of events, as $\{A, B\}$ is as well, $\{\Omega, \varnothing\}$, etc. Also the $\{\varnothing, \Omega\}$ sets are *all possible* subsets of $\mathcal{S}$ and they form a family. How many subsets of Ω are possible ? For instance, in one throw, how many are the possible events ?

A general classification of events is:

- *Full group of events*: a set of events forms a complete group when at least one of them occurs with certainty. Example: heads, tails in the flip of a coin.
- *Incompatible events*: two events are incompatible if they cannot occur simultaneously. Example: score 2 and score 3 when rolling a die.
- *Equally likely events*: two events are equally likely if there is no reason to assign them a different probability. This concept is somewhat unsatisfactory, because it suffers from "circularity" (that is, it requires having already defined the probability). However, it can be justified by resorting to considerations of symmetry (think of the faces of a die) or the principle of insufficient reason: if I know nothing about two events, I have no reason to consider one more probable than the other.
- *Elementary events and compound events*: an event is elementary when it cannot be decomposed into other events. For example, the event "the drawn card is a two of spades" is an elementary event. An event is said to be composed when it can be obtained as a "sum" of elementary events. For example: "the result in the roll of a die is an even number" corresponds to the sum of the events "a 2 comes up", "a 4 comes out" and "a 6 comes out".

Elements of a set can be *mutually exclusive* if two or more outcomes cannot occur simultaneously, and they are *exhaustive* if all possible outcomes are present in the list. It means that every time the experiments is performed one of the outcome of the sample space will occur. A collection of elements belonging to the sample space is called an *event* and **set theory** is employed to derive relationships between events and to derive probabilities.

In **set notation** the sample space $\mathcal{S}$ is called *universal set*, the set that includes all the possible outcomes, while an event A is called a *subset*. The main set operations are:

- **Union**. The union of two sets A and B is a set of all elements which belongs to A, to B or to both. It is written as $A \cup B = (x \in A \text{ or } x \in B \text{ or both})$.
- **Intersection**. The intersection of two sets A and B is a set which contains all the elements common to A and B. It is written as $A \cap B = (x \in A \text{ and } x \in B)$.
- **Complement**. The complement A^c of a set A is the set of elements which belongs to the universal set $\mathcal{S}$ but do not belong to A^c. It is written as $A^c = (x \notin A)$.

Set operations can be represented by the **Venn diagrams** as shown in Fig.7.1

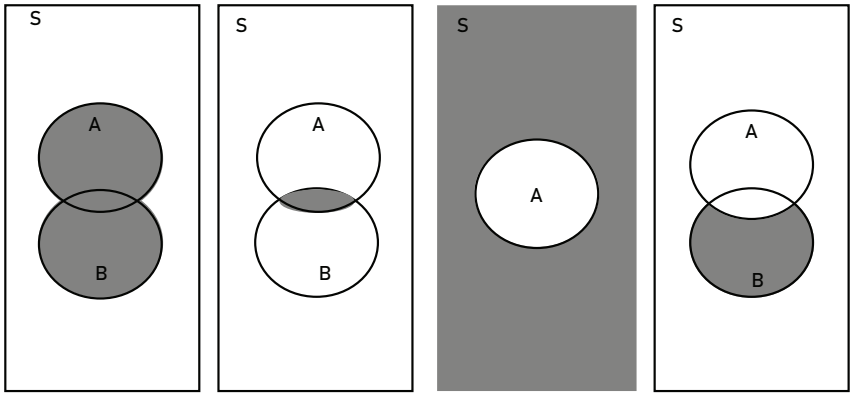


Fig. 7.1 S is the universal set. From left to right: the union $A \cup B$, the intersection $A \cap B$, the complement A^c of A and the complement of A^c intersection B, $A^c \cap B$

A set that contains no element is called **empty set** and it is denoted $\varnothing$.

As described above two events A and B are *mutually exclusive* if A and B have no elements in common, therefore their intersection is an empty set, $A \cap B = \varnothing$, in set theory these two sets are called *disjoint*.

The probability of any events must satisfy three axioms:

- **Axiom 1**: $0 \leq P(A) \leq 1$
- **Axiom 2**: $P(S) = 1$
- **Axiom 1**: $P(A \cup B) = P(A) + P(B)$ if (A) and (B) are mutually exclusive.

7.2.1 Dice with R

In this section examples of probabilities are presented, by simulating the throwing of dice with R.

The first example is to generate one number included between 1 and 6 with the instruction `sample`. This instruction will be used many times throughout this book. Sample generates random numbers and permutations. It takes a sample of the specified size from the elements of x using either with or without replacement. The function has the following arguments:

```
sample(x, size, replace = FALSE, prob = NULL)
```

where `x` is either a vector of one or more elements from which to choose, or a positive integer; `size` a non−negative integer giving the number of items to choose; `replace` defines if sampling should be with replacement? `prob` a vector of probability weights for obtaining the elements of the vector being sampled.

Sampling with replacement is used to find probability with *replacement*. In other words, the probability of some event is computed where for instance there is a number of balls, cards or other objects, and you replace the item each time you choose one, so the overall total number of elements in the set does not change. This method results in the possibility of choosing the same element multiple times. Indeed, by choosing replacement, it is possible to pick a number and then put the number back in the set, therefore the same number could be choose again. In this case the probability are independent.

Sampling *without replacement* is a way to figure out probability without replacing the element. In other words, the first chosen item is not replaced before choosing the second and so forth. This dramatically changes the odds of choosing sample items. Here the game of dice is implemented with some instructions. In the first example the instruction `sample` is used to roll once a dice with 6 faces.

```
sample(1:6,size= 1, replace=TRUE)
```

The outcome of this computation is the simulation of one roll, determining random numbers comprises between 1 and 6 as shown below.

```
> sample(1:6,size= 1, replace=TRUE)
[1] 4
> sample(1:6,size= 1, replace=TRUE)
[1] 3
> sample(1:6,size= 1, replace=TRUE)
[1] 4
```

The same instruction can be written as follows:

```
sample(1:6,1)
```

and the outcome will be the same. To roll the die ten times:

```
sample(1:6,size= 10, replace=TRUE)
```

and the outcome will be the ten numbers. Since `replace` is set to TRUE, there can be pairs, triplets and so on of the same number.

```
[1] 1 6 5 1 3 3 5 6 1 2
```

The function `samples` allows for generating more numbers. In this example, two numbers are generated (like rolling two dice), they are visualized and them and summed up. The same instruction can be placed into a function:

```
dice= function(n){
       sample(1:6,size=n, replace=TRUE)
}
outcome=dice(10)
outcome
```

and the same outcome is obtained.

```
tworoll= sample(1:6,size=2, replace=TRUE)
tworoll
sum(tworoll)
```

the outcome will be:

```
[1] 2 2
[1] 4
```

In the next example, a code is written to check the theoretical probabilities described above. The die is rolled 1000 times and the numbers of times the number 2 is obtained are counted.

```
s= sample(1:6,size=1000, replace=TRUE)
s==2
sum(s==2)
```

The outcome is

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE FALSE FALSE FALSE FALSE FALSE TRUE
FALSE FALSE FALSE.........
[1] 152
```

where the FALSE and TRUE are the results of the test `s==2` executed 1000 times and 152 is the number of times the outcome was 2. The proportion is 152/1000= 0.152, or 15.2 %. Since the probability for a single number should be 1/6 or 0.1666 or ≅ 16 %, the outcome is a little smaller. If the die is rolled 10000 times, the results obtained are 0.169, 0.160, 0.17, 0.168 and so forth. By increasing the number of rolls the number gets closer and closer to the theoretical probability value. With 100000 rolls the number is always 0.1669, 0.1668 and so forth, therefore it is correct to the second decimal.

In the example below the probability of obtaining the number 2 from a consecutive and incremental number of rolls is computed.

```
count=1
rolls<-vector()
x<-vector()
for(rolls in 1:1000){
       s=sample(1:6,size=rolls,replace=TRUE)
```

```
        #print(s)
        summation<-sum(s == 2)
        #print(summation)
        ratio<-summation/rolls
        print(ratio)
        x[rolls]<-ratio
}

plot(x, type = "s", col = "red", lwd = 1,
main = "",xlab = "Number of rolls",
ylab = "Probability [-]")
```

The output of this program is plotted in Fig. 7.2

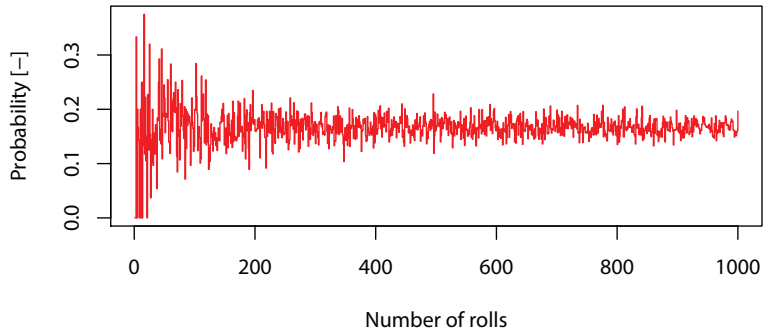


Fig. 7.2 Fraction of die rolls that are 2 at each roll in a simulation. The proportion tends to get closer to the probability $1/6 \cong 0.166$ as the number of rolls increases.

At increasing numbers of rolls the probability $(\overline{\mathsf{P_n}})$ tends to stabilize around its theoretical value (P). This tendency is described by the **Law of Large Numbers**.

7.2.2 Law of large numbers

Be F_n the relative frequency of success in n independent trials, and p the probability of success in each trial. Then F_n, for $n \to \infty$, *converge in probability* to p. Here with it "converges in probability" means that the probability of the event: $\{|F_n - p| < \epsilon\}$ approximate 1, with large n and small ϵ. The law of large numbers is a theorem about the probability of an event, valid in the framework of the adopted theoretical model. It can be stated as:

The proportion $(\overline{\mathsf{P_n}})$ *of occurrences with a particular outcome converges to the probability* (P) *of that outcome, as increasing numbers are collected*

It exists an empirical law that postulate: the frequency of success approximate the probability P and the approximation tends to improve at the increasing number of trials or experiments.

7.2.3 Addition rule

Two probabilities are called *mutually exclusive* or *disjoint* if they cannot happen simultaneously. For instance the probability of obtaining 1 and an even number. The probability can be summed. For instance the probability of getting 1 or 2:

$$P(1 \text{ or } 2) = P(1) + P(2) = 1/6 + 1/6 = 1/3 \tag{7.1}$$

Similarly, all the outcomes can be summed:

$$\begin{aligned} P(1 \text{ or } 2, \text{or } 3, \text{or } 4, \text{or } 5, \text{or } 6) = \\ P(1) + P(2) + P(3) + P(4) + P(5) + P(6) = \\ 1/6 + 1/6 + 1/6 + 1/6 + 1/6 + 1/6 = 1 \end{aligned} \tag{7.2}$$

This law is called addition rule, and it applies when the outcomes are mutually exclusive.

7.2.4 Probability of mutually inclusive outcomes

In many instances, probabilities are not disjointed such for a deck of 52 cards.

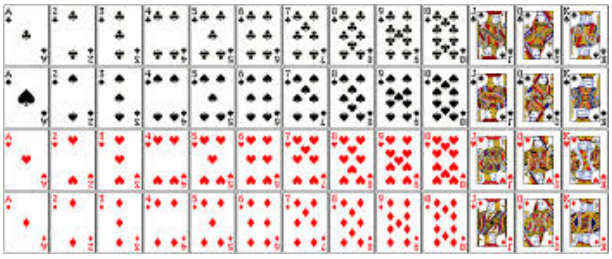


Fig. 7.3 Card deck with 52 cards.

The 52 cards are split into four suits: club (♣), diamond (♦), spade (♠) and heart (♥). Each suit has its 13 cards labeled: A (ace), 2, 3, ..., 10, J (jack), Q (queen) and K (king). Thus, each card is a unique combination of a suit and a label. The 12 cards represented by the jacks, queens, and kings are called face cards. The cards that are club and spade are typically colored black while the other two suits are typically colored red.

The question could be what is the probability that a randomly selected card is a heart ? and what is the probability that a randomly selected card is a face ? In this case the probabilities are not disjointed since it is possible to obtain a card that is a heart and it is facecard.

To describe the probability it is convenient to graphically represent it with set theory concept, the **Venn Diagram**. This diagram allows for representation of a number of elements in a set that are shared with another set. On the left the oval encompass the set of hearts (♥), which are thirteen cards, while the oval on the right represents the set for face card, which are twelve cards. When a card is both a heart and a face

card it falls into the intersection of the ovals since it belongs to both sets. In set theory this is called an intersection set. The intersection of two sets, A and B, denoted by $A \cap B$, is the set containing all elements of A that also belong to B (or equivalently, all elements of B that also belong to A.

If a card is a heart but not a face card than it falls on the left part of the left oval (10 cards), likewise if a card is face card but not a heart it falls in the right part of the right-oval (9 cards). The probability that a card is a heart but not a face card is $10/52 = 0.19$ and the probability that a card is a face card but not a heart is is $9/52 = 0.17$.

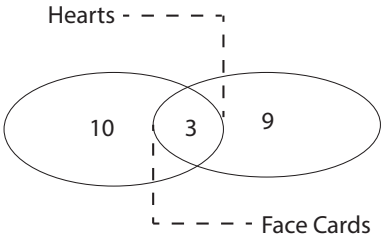


Fig. 7.4 Venn diagram for hearts and face cards.

The event that a randomly selected card is a heart is represented by A and the event that it is a face card represented by B.

How is the event $P(A \text{ or } B)$ computed ? These events are not disjointed, since the three cards J(♥), Q(♥) and K(♥) are present in both categories. In this case the addition rule for disjoint probabilities cannot be used. The Vienn Diagram is used. Firsr the probabilities of the two events are computed:

$$P(A) + P(B)) = P((P(A))) + P(\text{facecards}) = 13/52 + 12/52 = 0.25 + 0.23 = 0.48 \quad (7.3)$$

Using this calculation the three cards were counted twice, once for the hearts and one for the face cards. The correction of this error is performed by using:

$$\begin{aligned} P(A \text{ or } B) &= P(\heartsuit \text{ or facecard}) = \\ P(\heartsuit + \text{facecard}) &- P(\heartsuit \text{ and facecard}) = \\ = 13/52 + 12/52 &- 3/52 = 22/52 = 11/26 = 0.42 \end{aligned} \quad (7.4)$$

The general equation for two events A and B , disjoint or not, for the probability that at least one of them will occur is

$$P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B) \quad (7.5)$$

In statistics the term *or* means *and* − *or* unless it is explicitly stated otherwise. Thus, A or B occurs means A, B, or both A and B occur.

7.3 Conditional probability and Bayes Theorem

Often data present interesting relationships between two or more variables that are useful to investigate. For example a car insurance company will consider information about a person's driving history to assess the risk that they will be responsible for an accident. These types of relationships are the realm of conditional probabilities.

7.3.1 Contingency Table

Let's assume a dataset called `photo_leaf_classification` represent a sample of 1822 photos of leaves with a yellow spots from a fungal disease. Researchers have been working to develop an image analysis algorithm based on machine learning (ML) to improve the automatic classification of leaves having or not having yellow spots. The 1822 photos represents a test for their classification. Each photo gets two classifications: the first is called ML and gives a classification from a machine learning (ML) system of either pred yellow spots or not (TRUE or FALSE). The algorithm presents a system to classify a leaf without spots even if it has few spots. The independent test was performed by a group of researchers with visual inspection and these data are considered the reference source or truth.

The following *contingency table* summarizes the results of the study:

	Truth		
MachineLearning	TRUE	FALSE	Total
PredTRUE	197	22	219
PredFALSE	112	1,491	1,603
Total	309	1,513	1,822

The question is now: If a leaf has the presence of spots by human inspection (TRUE), what is the chance the ML classified correctly the photo as having spots ? The probability can be estimated using the data. Of the 309 leaves presenting spots, the ML algorithm correctly classified 197 of the photos having spots. Therefore the probability is

$$P = \frac{197}{309} = 0.638$$

What is the probability that a leaf that did not have the presence of spots was correctly classified as FALSE from the ML algorithm. In analogy with the previous computation:

$$P = \frac{112}{1603} = 0.069$$

7.3.2 Marginal and joint probabilities

The data presented in the contingency table lists row and column totals for each variable separately in the photo classify data set. These totals represent *marginal*

probabilities for the sample, which are the probabilities based on a single variable. For instance, the probability based solely on the ML variable is a marginal probability:

$$P = \frac{219}{1822} = 0.12$$

This probability is the probability of predicted presence of spots over the photographed leaves analyzed with machine learning.

If the estimation of the machine learning method and the direct observations are combined, the probability is called *joint probability*

$$P = \frac{197}{1822} = 0.11$$

In this case it is the probability that both methods will reveal a presence of spots on the 1822 leaves, in this case the probability is slightly smaller.

Summarizing, if a probability is based on a single variable, it is a *marginal probability*, on the other hand if the probability of outcomes for two or more variables or processes is called a *joint probability*.

	Truth		
MachineLearning	TRUE	FALSE	Total
PredTRUE	0.11	$1.21 \cdot 10^{-2}$	0.12
PredFALSE	$6.15 \cdot 10^{-2}$	0.82	0.88
Total	0.17	0.83	1

The probability can then be summarized into a table

Table 7.1 Joint probability distribution for the leaves classification data set.

Joint outcome	Probability
ML is TRUE and truth is TRUE	0.1081
ML is TRUE and truth is FALSE	0.0121
ML is FALSE and truth is TRUE	0.0615
ML is FALSE and truth is FALSE	0.8183

7.3.3 Conditional probability and Bayes Theorem

Given two events A and B, the occurrence of A may or may not affect the likelihood of B occurring. For example, if a box contains one white ball and ten black ones, drawing the white ball makes certain ($P = 1$) the next drawing of a black ball. Conversely, if the balls are returned to the box after the draw, the draw of the white ball has no influence on the subsequent draw.

We denote by $P(B|A)$ the probability of B "conditional" on the occurrence of event A. We now define the probability of the product between two events, meaning that event $A * B$ consists of "A occurs and B occurs" (for example, in the roll of a die,

both A = "2 comes out" and B = "comes 3", the event $A * B$ = "comes 2" on the first roll and "comes 3" on the second roll or with a second die). It is evident that $P(A * B) = P(A) * P(B)$ only if the two events A and B are independent. It can be shown that, in general:

$$P(A * B) = P(A) * P(B|A) \tag{7.6}$$

therefore

$$P(A * B) \leq P(A) * P(B) \tag{7.7}$$

with the equality that holds only if $P(B|A) = P(B)$, that is, if B is independent of A. It can also be shown (and it is intuitive) that if B is independent of A also A is independent of B, and therefore the following holds:

$$P(A * B) = P(B) * P(A|B) \tag{7.8}$$

it follows that:

$$P(A|B) = \frac{P(A * B)}{P(B)} \tag{7.9}$$

and substituting equation 7.6 into 7.9 Finally:

$$P(A \mid B) = \frac{P(A) * P(B|A)}{P(B)} \tag{7.10}$$

From equation 7.10 it is possible to rigorously define what is defined as "learning from experience" and which is the basis of all sciences. Let's see how. Suppose we are dealing with a complete set of incompatible events $H_1, H_2, \ldots, H_N$, which we will call "hypothesis" for reasons that will be clear shortly. For each of the hypotheses H_i we can apply the equations above:

$$P(H_i \mid E) = \frac{P(H_i) * P(E|H_i)}{P(E)} \tag{7.11}$$

where E is any event. Since E occurs in conjunction with only one of the H_i (by definition, since the H_i are incompatible), we can therefore write:

$$E = E * H_1 + E * H_2 + ... + E * H_N \tag{7.12}$$

from the addition formula we have

$$P(E) = P(E * H_1) + P(E * H_2) + ... + P(E * H_N) \tag{7.13}$$

and from 7.7:

$$P(E) = P(H_1) * P(E|H_1) + P(H_2) * P(E|H_2) + ... + P(H_N) * P(E|H_N) \tag{7.14}$$

therefore

$$\mathsf{P(E)} = \sum_{i=1}^{N} \mathsf{P(H_i)} * \mathsf{P(E|H_i)} \tag{7.15}$$

Therefore equation 7.10 becomes:

$$\mathsf{P(H_i \mid E)} = \frac{\mathsf{P(H_i)} * \mathsf{P(E|H_i)}}{\sum_{i=1}^{N} \mathsf{P(H_i)} * P(E|H_i)} \tag{7.16}$$

Equation 7.16 is known as *Bayes equation*, and it allows to compute how our confidence is updated in the hypothesis H every time a given event has occurred. Overall, the Bayesian inference scheme is formally quite simple: the uncertainty about a parameter θ after data y have been observed, is computed simply by specifying $P(y|\theta)$ (usually referred to as the likelihood $l(\theta, y)$ when viewed as a function of θ) and $P(\theta)$, and normalizing their product to make it a probability distribution.

It can also be described as: $\mathsf{P(H_i \mid E)}$ is the posterior probability, $\mathsf{P(H_i)}$ is the prior probability without the data, $\mathsf{P(E|H_i)}$ is the likelyhood, which is the probability that the data could be generated by the model with a given parameter value and $\sum_{i=1}^{N} \mathsf{P(H_i)} * P(E|H_i)$ is the evidence.

In the example above the machine learning predicted if a photo of a leaf displayed spots from a disease. The estimation is not perfect (it has error) but it can be a very useful and fast method to classify leaves based on automatized methods, rather than from visual inspection. It is of interest to better understand how to use this information to improve the ability to estimate a second variable, which for the example above is the truth measurement.

7.3.4 Examples

Spotted leaves. We go back to the example of the spotted leaves. The probability that a random photo from the data set present spots is about 0.17 (309/1822). The question is: if we know that machine learning predicted that the leaf had spots, can we get a better estimate of the probability that the leaf has indeed spots ? The answer is yes. As an example a subset is selected of 219 cases where the ML classified the leaf as TRUE (having spots):

$$\mathrm{P(truth\ is\ TRUE\ given\ ML\ has\ predicted\ TRUE)} = \frac{197}{219} = 0.9$$

In this case the probability increased from 0.638 to 0.9 since the denominator was smaller. It is called a *conditional probability* because the probability was computed under a condition: the ML classifier prediction determine that the photo had spots (TRUE). The conditional probability is made of two parts: the outcome of interest and the condition.

It is useful to think of the condition as information we know to be true, and this information usually can be described as a known outcome or event.

We generally separate the text inside our probability notation into the outcome of interest and the condition with a vertical bar:

$$\mathrm{P}(\text{truth is TRUE} \mid \text{ML has predicted TRUE}) = \frac{197}{219} = 0.9$$

It simply reads that the visual observation detects a spot since the machine learning detected a spot, therefore the probability is 0.9.

In many cases, marginal and joint probabilities are provided instead of count data. For example, disease rates are commonly listed in percentages rather than in a count format.

Shooter. Two shooters each fire one shot at a target. The probability of the first shooter hitting the target is 0.8, that of the second is 0.4. It is observed that the target has been hit; what is the probability that it was the first shooter to hit him?
Solution:
Before the "target hit" event, there are four possible hypotheses:

- H_1 = both shooters miss
- H_2 = both hit the mark
- H_3 = the first hits the mark and the second doesn't
- H_4 = the second hits the mark and the first doesn't

Given $B1$ = "first shooter to score", and $B2$ = "second shooter to score", the probabilities of the four hypotheses are as follows (note that $B1$ and $B2$ are independent). We denote, as before, with B^{I} 'the event opposite to B.

- $\mathsf{P}(\mathsf{H}_1) = \mathsf{P}(\mathsf{B}_1^{\mathsf{I}}) * \mathsf{P}(\mathsf{B}_2^{\mathsf{I}}) = 0.2 * 0.6 = 0.12$
- $\mathsf{P}(\mathsf{H}_2) = \mathsf{P}(\mathsf{B}_1) * \mathsf{P}(\mathsf{B}_2) = 0.8 * 0.4 = 0.32$
- $\mathsf{P}(\mathsf{H}_3) = \mathsf{P}(\mathsf{B}_1) * \mathsf{P}(\mathsf{B}_2^{\mathsf{I}}) = 0.8 * 0.6 = 0.48$
- $\mathsf{P}(\mathsf{H}_4) = \mathsf{P}(\mathsf{B}_1^{\mathsf{I}}) * \mathsf{P}(\mathsf{B}_2) = 0.2 * 0.4 = 0.08$

The probabilities of $E =$ "*hit target*" conditional on the 4 hypotheses are, of course:

- $\mathsf{P}(\mathsf{E}|\mathsf{H}_1) = 0$
- $\mathsf{P}(\mathsf{E}|\mathsf{H}_2) = 0$
- $\mathsf{P}(\mathsf{E}|\mathsf{H}_3) = 1$
- $\mathsf{P}(\mathsf{E}|\mathsf{H}_4) = 1$

Once E occurs, H_1 and H_2 become impossible and the probabilities of H_3 and H_4 - conditional on the occurrence of E - become:

$$\mathsf{P}(\mathsf{H}_3 \mid \mathsf{E}) = \frac{0.48 * 1}{0.48 * 1 + 0.08 * 1} = \frac{6}{7} \tag{7.17}$$

and

$$\mathsf{P}(H_4 \mid \mathsf{E}) = \frac{0.08 * 1}{0.48 * 1 + 0.08 * 1} = \frac{1}{7} \tag{7.18}$$

The core of Bayes' formula lies in 7.17 and 7.18. Before experiment (E) the probability of hypothesis H_3 was 48%, after the experiment its probability increased to 86%.

Although artificial, the example shows how experience changes our knowledge of a phenomenon.

Equation says that the "a posteriori probability" $\mathsf{P}(\mathsf{H}|\mathsf{E})$ of a hypothesis H given the occurrence of the event E can be calculated by multiplying the "a priori probability" $\mathsf{P}(\mathsf{H})$ of the same hypothesis by the "likelihood" of the experimental datum E, $\mathsf{P}(\mathsf{E}|\mathsf{H})$, divided by the probability of the event E.

The likelihood of event E *(or experiment) given a hypothesis* H *measures how much this event is "compatible" with the hypothesis itself.* In the simple example above, the likelihood was 0 or 1, but this is generally not the case. Let's see it with another, more realistic example.

Clinical test. A clinical test gives a 90% accurate result, i.e. 90% positive and 10% negative when performed on people with a certain disease M. Let's assume the test is 95% negative and 5% positive, when performed on healthy people. It can also be described having 10% of false negative and 5% of false positive.

The incidence of M disease in the population is known, for example this incidence is 2%. The test is performed on a patient, which is positive. How likely is the patient to really have M?

Solution:

The hypotheses H = "the patient is sick" and H' "the patient is healthy" have, before the test, the following probabilities: $\mathsf{P}(\mathsf{H}) = 0.02$ and $\mathsf{P}(\mathsf{H}') = 0.98$. After the T test (positive), we have:

$$\mathsf{P}(\mathsf{H}|\mathsf{T}) = \frac{\mathsf{P}(\mathsf{H}) * \mathsf{P}(\mathsf{T}|\mathsf{H})}{\mathsf{P}(\mathsf{T})} = \frac{0.02 * 0.9}{0.02 * 0.9 + 0.98 * 0.05} = 0.27$$

If we have no other indications that the patient is suffering from M, the probability that he is ill - once the test is positive - is 27 %. This surprising result is due to the low a priori probability of the disease M.

What if the test gives more "false positives", for example if the probability of the test being positive on healthy people is 10%? In this case, the numerator does not change, but the denominator becomes:

$$\mathsf{P}(\mathsf{H}|\mathsf{T}) = \frac{\mathsf{P}(\mathsf{H}) * \mathsf{P}(\mathsf{T}|\mathsf{H})}{\mathsf{P}(\mathsf{T})} = \frac{0.02 * 0.9}{0.02 * 0.9 + 0.98 * 0.1} = 0.15$$

from which: $\mathsf{P}(\mathsf{H}|\mathsf{T})$ decrease to 15.5%.

A simple code in R is shown below.

```
content# H = sick patient
# A priori Probability
p.H <- 0.02
p.notH <- 1 - p.H
p.T.givenH <- 0.9
```

```
p.T.given.notH <- 0.1

p.T <- p.H*p.T.givenH + p.notH*p.T.given.notH

(p.H.givenT <- p.H*p.T.givenH/p.T)...
```

Lighthouse

This example is taken and reworked from *Gull (1988) in "Bayesian inductive inference and maximum entropy", in Maximum entropy and Bayesian methods in science and engineering, Kluwer.*

The example is remarkable for at least three reasons. (1) It shows how the Bayesian approach is a " paradigm " that includes techniques normally used in a more or less uncritical way. (2) It shows how the mean is not **always** the best estimate of a random quantity, and how the solution exists even when the central limit theorem does not hold. (3) Finally, it shows how the information present in the data always wins in the end, if it is sufficient, and how the influence of the first choice of the *a priori* probability becomes negligible as the experimental data grows.

Description

A lighthouse is in a certain position on a straight stretch of coastline, at a position X_0 along the beach, measured from an arbitrarily chosen origin, and at a distance of Y_0 from the sea. The lighthouse is in constant rotation and emits short collimated flashes, at random time intervals (and therefore θ angles). Photo detectors on the beach record the flash, but not the θ angle from which the beam is coming.

The experimental data is the set of $\{x_k\}$ positions of the photo - detectors that have been activated by a flash.

Suppose, for simplicity (just so as not to have to infer two parameters, but only one) we know the distance Y_0. What is the X_0 position? How can we estimate it from the data?

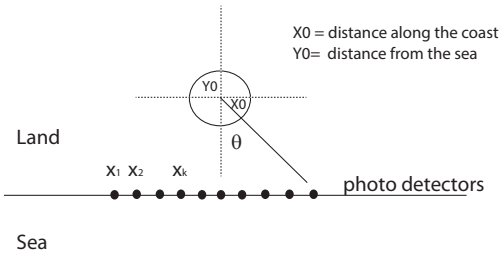


Fig. 7.5 Schematic of the problem

Solution

At each x_k correspond an azimuth value θ_k. For the light beam to be visible, the angle must be between $-\pi/2$ and $\pi/2$ (extremes not included), i.e .:

$$\theta_k \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$$

Obviously, if the angle is exactly $\pm\pi/2$ the light beam does not hit the coast. Realistically we can assign a uniform probability density to the azimuth θ_k, that is to the $k-th$ datum. If we denote by X the unknown position (while Y_0 is known):

$$p(\theta_k|X, Y_0) = \frac{1}{\pi}$$

The value $\frac{1}{\pi}$ comes from the integration of the uniform probability distribution:

$$\int_{-\infty}^{+\infty} p(\theta)d\theta = 1$$

In this case, the integral is definite:

$$p(\theta_k|X, Y_0) = \int_{-\pi/2}^{\pi/2} p(\theta)d\theta = \frac{1}{\pi/2 - (-\pi/2)} = \frac{1}{\pi}$$

Trigonometric considerations, when X_0 is known, allow us to say that:

$$Y_0 \tan(\theta_k) = x_k - X_0$$

By knowing X_0, Y_0 and x_k, the $\tan(\theta_k)$ is obtained and the angle is obtained with the arc tangent function.
The position of the photo−detector activated by the light beam is therefore:

$$x_k = X_0 + Y_0 \tan(\theta_k)$$

and the tangent of the angle is:

$$\tan(\theta_k) = \frac{x_k - X_0}{Y_0}$$

therefore the angle is

$$\theta_k = \operatorname{atan}\left(\frac{x_k - X_0}{Y_0}\right)$$

We use the variable transformation from θ to x to get the probability density of x_k. If $x = x(\theta)$, for dx and $d\theta$ infinitesimal:

$$p(x)dx = p(\theta)d\theta$$

therefore

$$p(x) = p(\theta)\left|\frac{d\theta}{dx}\right|$$

(the reason for the absolute value is that it must be a length ratio, i.e. always positive). The derivative $d\theta/dx$ is given by the derivative of the arc tangent with respect to x which is:

$$\frac{d\theta}{dx} = \frac{Y_0}{\left[Y_0^2 + (x_k - X_0)^2\right]}$$

while $p(\theta) = 1/\pi$ as shown above.
Finally:

$$p(x_k|X_0, Y_0) = \frac{Y_0}{\pi\left[Y_0^2 + (x_k - X_0)^2\right]}$$

In summary, if we know the (X_0, Y_0) position of the lighthouse, the probability of recording a flash at the x_k position has the Cauchy distribution. The Cauchy distribution is explained in details in the next chapter about probability distributions. The Cauchy distribution is often used in statistics as an example of a "pathological" distribution since both its expected value and its variance are undefined. The Cauchy distribution does not have finite moments of order greater than or equal to one.

NOTE: If we did not know the distance Y_0 of the lighthouse from the sea, we would have to estimate two parameters, a somewhat more complex problem (the solution of which risks overshadowing what we are interested in showing here).

To estimate (infer) the X parameter (the position of the light beam), we need to estimate the posterior probability of X, given Y_0 and the records $\{x_k\}$:

$$p(X|x_k, Y_0)$$

From Bayes' theorem:

$$p(X|\{x_k\}, Y_0) = \frac{p(\{x_k\}|X, Y_0)p(X|Y_0)}{p(\{x_k\}|Y_0)}$$

The term in the denominator does not depend on the parameter sought. The first term of the numerator is what is called the " likelihood " of the data, the second is the a priori probability of X. When we have no idea what a priori distribution a variable has, it is reasonable to take it uniform in a sensible range $[X_{min}, X_{max}]$ and zero outside:

$$p(X|Y_0) = p(X) = \begin{cases} \alpha, & X \in [X_{min}, X_{max}] \\ 0, & X \notin [X_{min}, X_{max}] \end{cases} \tag{7.19}$$

If the data x_k are independent, as it is reasonable to assume, the probability $p(\{x_k\}|X, Y_0)$ is the product of the probabilities of the single events x_k, therefore:

$$p(\{x_k\}|X, Y_0) = \prod_{k=1}^{N} p(x_k|X, Y_0)$$

Take the logarithm of the posterior probability $p(\{x_k\}|X, Y_0)$

$$L = \log(p(\{x_k\}|X, Y_0) = \beta - \sum_{k=1}^{N} \log(Y_0^2 + (x_k - X)^2)$$

where β includes everything that does not depend on the X parameter. The estimate of the position X_0 is obtained by looking for the maximum of the *a posteriori* distribution, that is, theoretically looking for the value of X which is a solution of:

$$\frac{dL}{dX} = 2 \sum_{k=1}^{N} \frac{x_k - X}{Y_0^2 + (x_k - X)^2} = 0 \tag{7.20}$$

Numerical procedure

The explicit solution of (7.20) is not analytically feasible. Instead of solving it numerically, it is more instructive to see how the posterior probability $\exp(L)$ behaves as the number of detections $\{x_k\}$ changes. This is what the following R code does, where we assume Y_0 = 1 km and the " true " value of X_0 2 km. The code generates N angles θ_k and from these it calculates x_k, since the true value of X_0 is known.

```
##Ch7_3.R
## Where is the light ?

# distance from sea
Y0 <- 1
# distance along the coast
X0 <- 2
# possible values of X (positions of photo-detectors)
dx <- 0.05
X <- seq(-5,5,dx)
Nx <- length(X)

#########################################################################
# numero di rilevazioni
# da variare per osservare l'effetto sulla distribuzione a posteriori
N <- 10
#########################################################################

# tetak <- runif(k,-pi/2,pi/2)
# instead of taking theta between -pi/2 and pi/2
# theta is selected such to determine "possible" x
# included between -x_max and x_max
x.max <- 50
# Here the angle is obtained using the arctan function, where x.max in an angle
tetak.max <- atan(x.max)
tetak <- runif(N,-tetak.max,tetak.max)
tetak
# compute the positions of the detectors activated by flash light
xk <- X0+Y0*tan(tetak)
# What is the distribution of the positions ?
hist(xk,main="")

L <- rep(0,Nx)
for (i in 1:Nx){
lk <- log(Y0^2+(xk-X[i])^2)
L[i] <- sum(lk)
}
```

```
hist(lk)

# posterior probability
post <- exp(-L)
plot(X,dx*post/sum(post),t="l",ylab="p(X|x,Y0)")
abline(v=2,col="blue",lty=2)
abline(v=mean(xk),col="red")
```

Results

This is what happens (typically, the calculation is stochastic ...) as the number of detections changes.

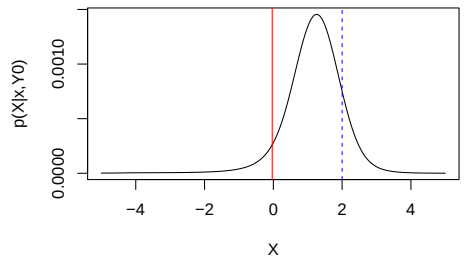


Fig. 7.6 N = 4

The true value (in blue) and the average value of x_k (in red) are superimposed on the probability density curve. Due to the choice of a uniform distribution for θ (which is reflected in a Cauchy distribution for likelihood), and a uniform a priori probability, with little data the maximum posterior probability rarely hits the real position X_0.

For N = 100 detections, the maximum a posteriori probability starts hitting the true value (figure 7.8) practically always, but the average value of x_k can be very far from it.

At first sight this thing is surprising, because we are used to attribute almost " magical " properties to the average value, by virtue of the central limit theorem which asserts that for a sample $\{x_1 ... x_n\}$, trait from a distribution with mean μ and variance σ^2 the distribution of the mean value $\bar{x}$ tends to a normal distribution with mean μ and variance σ^2/n, for $n \to \infty$.

The problem here is that the Cauchy distribution violates the validity conditions of this theorem, because it has large tails and such as to give a moment of order 2 of infinite value. From this example we can observe that although the central limit theorem is not valid, and therefore the average is not a sensible estimate of the position of the lighthouse, we can still calculate the a posteriori distribution and the maximum of this gives us the correct position of the lighthouse. We note, incidentally, that this procedure coincides with the search for the maximum likelihood! In addition to

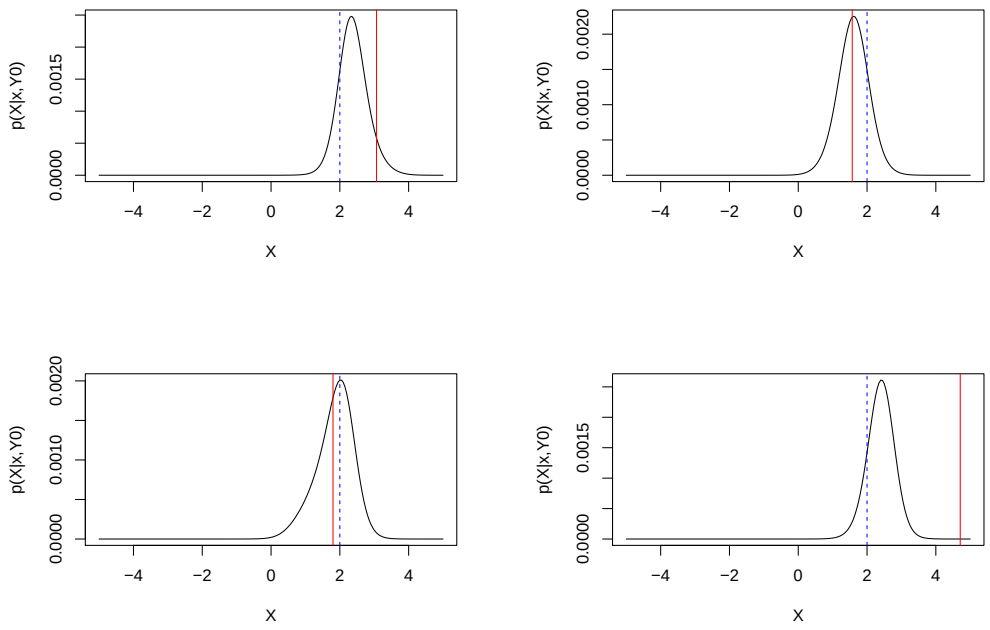


Fig. 7.7 N = 10

shedding light on the latter, whose motivation is often unclear in any other way, it is evident that nothing prevents us from using information on the position of the lighthouse, if we have it, and using it by imposing a different form of a priori probability, and making the calculation of the true position correct and faster.

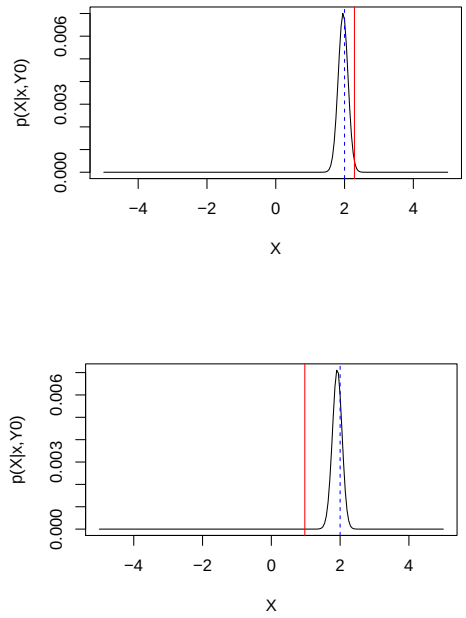


Fig. 7.8 N = 100

7.3.5 Procedure for Baeysian Analysis

1. Identify the data to answer the question. Which variables to use and which variables to study or predict.
2. Formulate a descriptive model (mathematical formulation and parameters) for the relevant data.
3. Specify a prior distribution on the parameters. The formulation should be realistic.
4. Use Bayesian inference to change the distribution and credibility of the probabilities after data have been collected and analyzed.
5. Check that the posterior predictions mimic the data with reasonable accuracy. If not, then consider a different descriptive model

7.4 Probability and determinism: the Buffon's needle

An interesting example of a problem that can be solved with either a deterministic approach or a probabilistic one is the Buffon's needle. Buffon's needle was the earliest problem in geometric probability to be solved.

We draw on a piece of paper a series of parallel lines with distance among each other of `2a`. A needle, of length `2L`, with `L<a` is dropped from above. The position of the needle is identified based on the distance `x` from the centre of the needle to the

closest line and with the angle θ generated from the intersection of the needle with the lines.

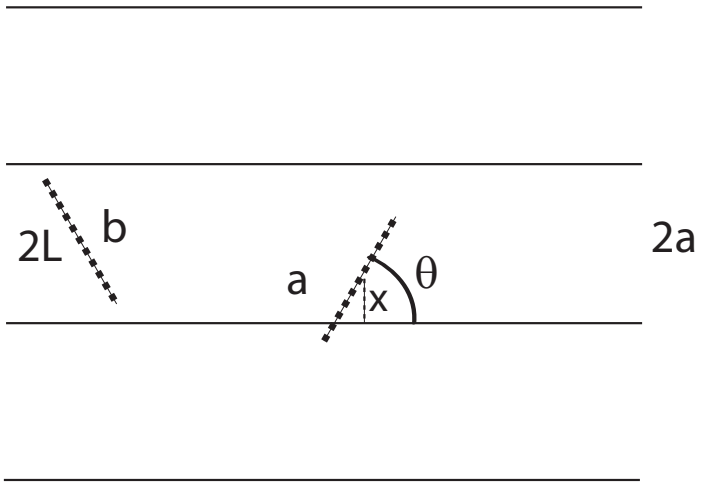


Fig. 7.9 Example of a needle (a) that lies across a line, and of needle (b) that does not.

The needle intersect the line if $x \leq L\texttt{sin}\theta$. The space Ω of the possible cases is the "rectangle" of sides (a,π), since we consider the distance of the closest line and the angle comprises between 0 and π.

To estimate the probability that the needle will intersect the line we must define a space of favoreable cases F. From the condition defined above this space is comprised between the line $x = 0$ and $x = L\texttt{sin}\theta$:

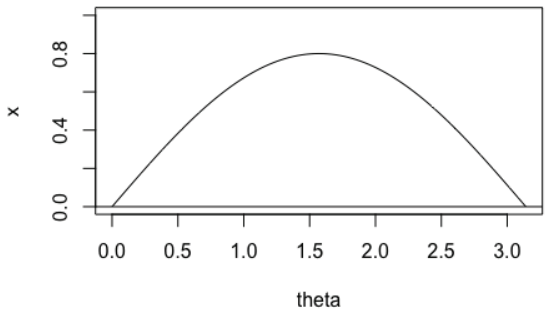


Fig. 7.10 "Rectangle" of sides (a,π)

We define the areas Ω and of F as $a(\Omega)$ and $a(F)$ respectively.

$$a(\Omega) = \pi a$$

while:

$$a(F) = \int_0^{\pi} L\sin\theta d\theta = 2L$$

The ratio $P = a(F)/a(\Omega)$ is the seeked *estimated probability* P. Substituting the values of $a(\Omega)$ and $a(F)$ it leads to:

$$P = \frac{2L}{\pi a}$$

Therefore, if we experimentally obtain p , from a sufficiently large number of throws, it is possible to estimate the value of π:

$$\pi = \frac{2L}{aP}$$

The R code below estimates the distribution of π in `Nruns` repetitions of the experiment that consist in throwing the needle `N` times.

```
##Ch7_4.R
## Buffon's needle problem

# a is the half-distance between lines
a <- 1
# L is the half-length of the needle
L <- 0.8

# repeat the experiment Nruns times
Nruns <- 200
result <- rep(0,Nruns)

for (k in 1:Nruns){
       #N is the number of throws of the needle
       N <- 100000
       #This instruction runif generates numbers with uniform distribution
       #with intervals from min (0) to max (2a)
       x <- runif(N,0,2*a)

       # The distance x is defined as that from the nearest line
       for (i in 1:N)
       if(x[i]>a) x[i] <- 2*a-x[i]

       #This instruction runif generates angles with uniform distribution
       #with intervals from min (0) to max (pi)
       theta <- runif(N,0,pi)

       L_sin_theta <- L*sin(theta)
```

```
        # compute frequency of line-crosses
        x<=L_sin_theta -> test
        p <- sum(test==TRUE)/N

        result[k] <-2*L/(a*p)
}

hist(result,main="Distribution of PI")
```

The results is depicted in Figure 7.4

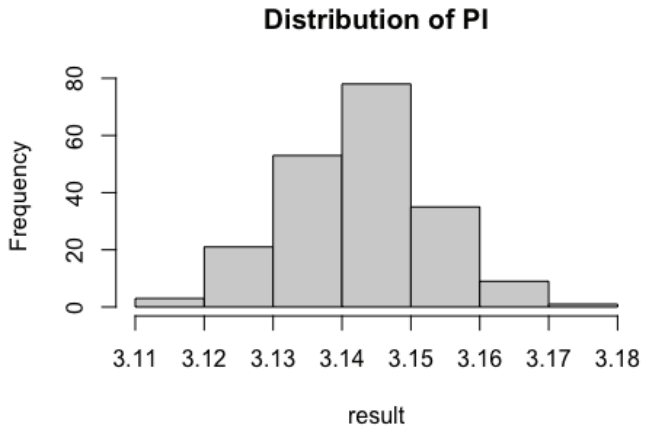


Fig. 7.11

7.5 Probability Distribution

A probability distribution is a table of all disjoint outcomes and their associated probabilities. The questions could be: what are the elements of the sample space $\mathcal{S}$? What is the probability that the sum of two faces is equal to 5 ? and so forth. The total number of possible combinations, therefore the sample space is $\mathcal{S} = 6 \times 6 = 36$.

A continuous probability distribution is a function that describes the assignment of probabilities to the occurences of values taken by a variable. It is usually called **probability density function (PDF)**. The area of the probability density function is

$$\int_{-\infty}^{+\infty} f(x)dx = 1 \tag{7.21}$$

Figure 7.5 shows the probability density function for a normal distribution with $\mu = 0$ and $\sigma = 1$. The area under the curve is equal to 1.

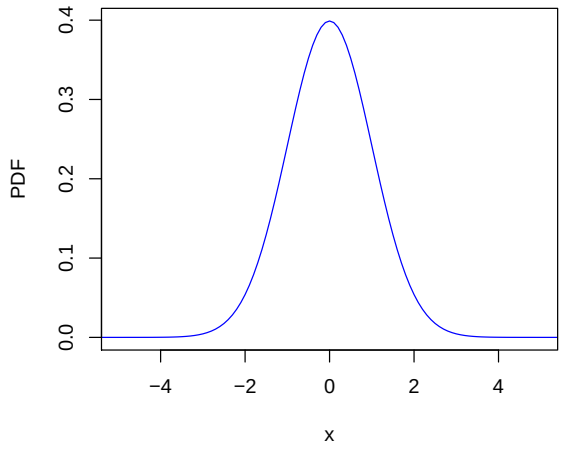


Fig. 7.12 Probability density function for a normal distribution having mean=0 and standard deviation = 1.

The balancing point of the curve is called **Expectation (E)**:

$$E[x] = \int_{-\infty}^{+\infty} xf(x)dx \tag{7.22}$$

Expectation works on a random number. It defines the **centroidal axis** of the PDF and is also known as the **mean** of x, $m(x) = E(x)$. A common measure of the dispersion of the distribution of x is the **variance** of x:

$$var[x] = E\left([x - m(x)]^2\right) = \int_{\text{all } x} [x - m(x)]^2 f(x)dx \tag{7.23}$$

The **covariance** is defined as:

$$\text{Cov}[x, y] = E\left([x - E(x)][y - E(y)]\right) \tag{7.24}$$

the values of $E(x)$ and $E(y)$ are compareable to the mean, the center of gravity of the distribution.

The PDF can be integrated to obtain a cumulative distribution curve (CDF) as depicted in Figure 7.5

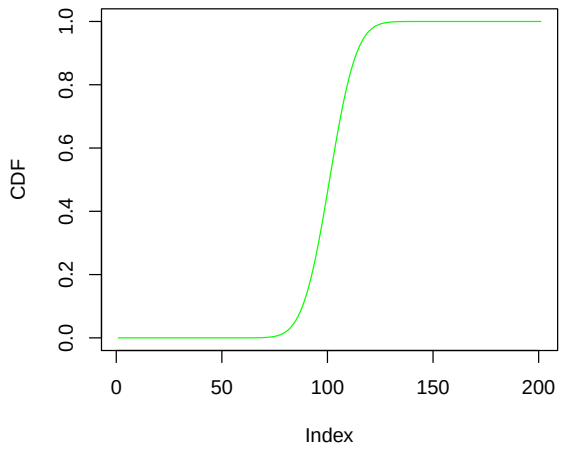


Fig. 7.13 Cumulative density function for a normal distribution having mean=0 and standard deviation = 1. Note that the cumulative curve reaches a value of 1, corresponding to the total area of the PDF.

Random does not mean uniform, normal or other distribution. Many types of distribution are possible for a random variable.

The code to plot the graphs for the PDF and CDF is shown below.

```
# Create a sequence of numbers between -10 and 10 incrementing by 0.1.
x <- seq(-10, 10, by = .1)

# Choose the mean as 0 and standard deviation as 1.
y <- dnorm(x, mean = 0, sd = 1)
y_cum<-pnorm(x, mean = 0, sd = 1)

plot(x,y,type="l",col="blue",xlim = c(-5,5),ylab=("PDF"))
plot(y_cum,type="l",col="green",ylab=("CDF"))
```

In the next chapter several distribution functions will be described.

Example. The probability of the occurrences of the sum of two dice is presented in Table 8.1. The total number of possible combinations is given by $6 \times 6 = 36$. Clearly the probability of obtaining 2 is only $1/36$ since it can be obtained only by $1 + 1$. On the other hand the combinations to obtain 3 are given by $2 + 1$ or $1 + 2$, therefore it is $2/36$ and so forth.

Table 7.2 Data of income depending on 5 measured features for 20 individuals

Dice Sum	Probability
2	1/36
3	2/36
4	3/36
5	4/36
6	5/36
7	6/36
8	5/36
9	4/36
10	3/36
11	2/36
12	1/36

In the code below, two vectors for the sum and the probabilities as reported in Table 8.1, are created and binded into a vector called `prob.dist`.

```
sum<-c(2,3,4,5,6,7,8,9,10,11,12)
prob<-c(1/36,2/36,3/36,4/36,5/36,6/36,5/36,4/36,3/36,2/36,1/36)
prob.dist<-c(sum,prob)
barplot(prob,names.arg=sum, main="",
xlab="Sum of two Dice",col="green")
```

Then a bar plot of the distribution is plotted in Fig. 7.5

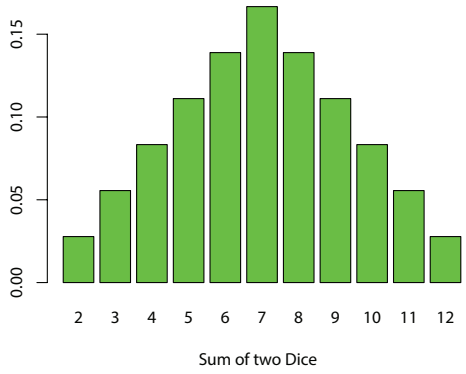


Fig. 7.14 Probability distribution for the sum of two dice.

7.6 Exercises

1. From a pack of well-shuffled 52 cards randomly pick a card. Compute the probability to pick a heart or an ace. Hint: the card ace of heart is common to both sets, therefore the sets are not mutually exclusive. content...

8
Distributions of Random Variables

8.1 Random variables

A *random variable* (rv) is a function that assumes its values with a given probability P. When one of the numbers that are observed (from instance from an experiment), those numbers are referred as *random variables*. In this notes the notation for a random variable is X. The set of all possible values of X is called *range* of X. Random variables can be *discrete* or *continuous*. A *discrete random variable* (*drv*) comes from a countable set of discrete values, while a *continuous random variable* (*crv*) can take any number in a continuous sample space.

8.1.1 Discrete random variables

In many application in data analysis, the outcome of the experiment is numerical. A **random variable** (r.v.) often denoted by capital letters like $X, Y and Z$ is a numerical value obtained from the experiment. Since the experiment can produce a variety of outcomes usually we refer to single elements as $x_1, x_2, x_3, ..., x_n$ which are outcomes of the general random variable X.

X is a discrete random variable if the range of X is a countable set:

$$S_X = [x_1, x_2, ..., x_n] \tag{8.1}$$

8.1.2 Continuos random variables

A continuos random variable is a random variable defined by a continuous set of numbers, referred as an *interval*. The interval contains all of the real numbers between two limits. For instance

$$[x_1, x_2] = (x | x_1 \leq x \leq x_2) \tag{8.2}$$

is a closed interval defined by all the real numbers between x_1 and x_2 including both x_1 and x_2.

There are many variables measured through experiments that lead to a *crv*, such as the arrival time of a particle, the voltage across a resistor, the photon reaching a solar radiation sensor.

8.2 Distributions

As introduced in the previos chapter, a probability distribution is a mathematical function that provides the probabilities of occurrence of different possible outcomes of

a random phenomena. The sample space, often denoted by Ω is the set of all possible outcomes of a random phenomenon being observed; it may be any set of real numbers, a set of vectors, a set of arbitrary non-numerical values. Many distributions have been derived describing different phenomena. For instance for some processes or phenomena, experiments provided information to apply a specific distribution. There are many distributions, the most common ones are: *Uniform, Bernoulli, Binomial, Geometric, Hypergeometric, Exponential, Poisson, Weibull, Normal, Log-Normal, Chi−Squared, Student's t, Gamma* and *Beta*. Clearly, some of them are strictly related both in terms of mathematical form and applications such as the Normal, Log-Normal, Chi−Squared and Student's t. When studying a process and analyzing data it is important to observe the data and identify the most suitable distribution to be applied to derive probabilities.

8.3 Uniform distribution

The uniform distribution is a distribution where equal probability is assigned to the random variable, within a given interval. It is applied when there are no reason to think that an occurence, an event would have higher probability to occur with respect to another event for the same process. It is often used in random sampling, when numbers should have the same probability, it is employed in finance and economics. In physics the emission of radioactive particles is often described with uniform distributions. Another example is the presence of a winning raffle ticket in a number N of tickets sold to people. At the beginning of the raffle the probability is the same of each ticket to be the winning one.

The uniform distribution can be continuous and discrete. The continuous uniform distribution the PDF is constant over the possible values of x.

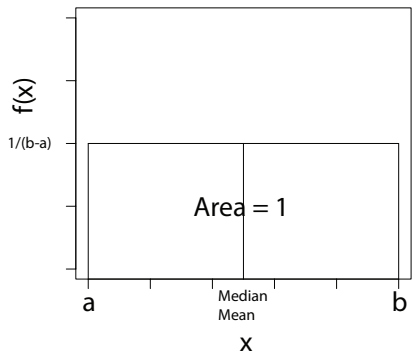


Fig. 8.1 Uniform distribution

The minimum value and maximum value that x can take are called a and b respectively (Fig.8.3). Any intervals in the interval $[a, b]$ or (a, b) are equally likely to occur.

Since the geometry is a rectangle, and the Area is equal to 1, it leads to:

$$\text{Area} = (b-a)f(x) = 1 \tag{8.3}$$

and

$$f(x) = \frac{1}{b-a} \tag{8.4}$$

Formally:

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } a \leq x \leq b \\ 0 & \text{elsewhere} \end{cases}$$

The median of the distribution is the value that splits the distribution in two equal parts and, since it is a symmetric distribution, the mean and median are equal and it is the midpoint between a and b. So the median is:

$$\text{median} = \frac{a+b}{2} \tag{8.5}$$

and so is the mean

$$\mu = \frac{a+b}{2} \tag{8.6}$$

The variance is

$$\sigma^2 = \frac{1}{12}(b-a)^2 \tag{8.7}$$

Usually for continuous distribution, finding the area under the curve requires integration. They can be analytical or numerical integrations, depending upon the mathematical form of the distribution, that may have an analytical integral or not.

For the uniform distribution the areas are simply rectangles. Suppose that the distribution has a value of $a = 20$ and $b= 25$, $f(x)$ is therefore $f(x) = \frac{1}{25-20} = 0.2$ for values of $20 \leq x \leq 25$. The area is therefore:

$$\text{Area} = \frac{1}{5}(25-20) = 1 \tag{8.8}$$

What is the probability that $P(x > 23)$. Probabilities are areas under the curve, therefore the area to the right of 23 must be found:

$$\text{P}(x > 23) = \text{base} \times \text{height} = (25-23)\frac{1}{5} = 0.4 \tag{8.9}$$

Using R it is easy to generate and analyse the uniform distribution. In the example below, where x values are generated within the interval 15 to 30. A random generation of uniform numbers is performed. Then the PDF and CDF of the distribution are computed and plotted.

```
x <- seq(15, 30, by = 0.01)
xx <- runif(x)
plot(xx)
dx <- dunif(x, min = 20, max = 25, log = FALSE)
px <- punif(x, min = 20, max = 25, lower.tail = TRUE, log.p = FALSE)
plot(x,dx,type="l",ylab=("f(x)"))
plot(x,px,type="l",ylab=("CDF"))
hist(xx,prob=T)
```

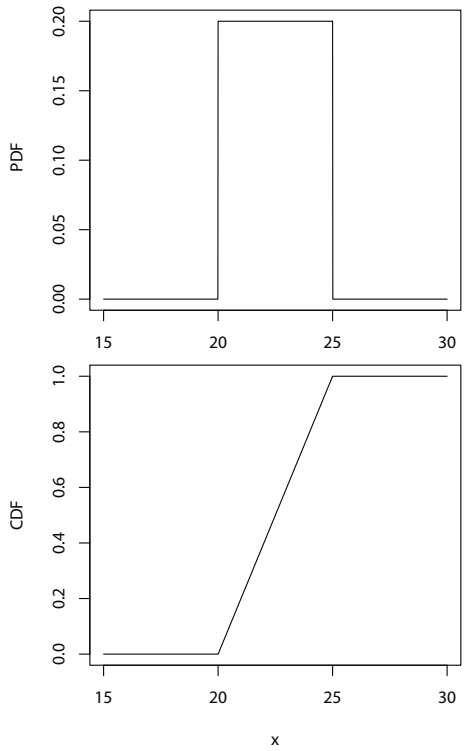


Fig. 8.2 PDF and CDF for the uniform distribution generated with R

8.4 Bernoulli distribution

Let us consider the flipping of a coin one time. What is the distribution of the number of heads in a single toss. This example has a Bernoulli distribution. We assume to have a single trail where each can have two possible mutually exclusive outcomes:

$$\begin{cases} a & \text{P(success)} = \text{p} \\ b & \text{P(failure)} = 1 - \text{p} \end{cases}$$

We define the random variable $X = 1$ if a success occurs and $X = 0$ if a failure occurs. Then X has a *Bernoulli* distribution:

$$P(X = x) = p^x(1-p)^{1-x} \tag{8.10}$$

for $x = 0$ or $x = 1$. It is also called probability mass function of the *Bernoulli* distribution. Now writing this distribution for the two individual values. For $x = 1$:

$$P(X = x) = p^1(1-p)^{1-1} = p \tag{8.11}$$

and for $x = 0$

$$P(X = x) = p^0(1-p)^{1-0} = 1-p \tag{8.12}$$

which is the same of what was written above, for success and failure, but with the formulation of eq. 8.10 it is written in one single equation. The mean of a Bernoulli random variable is

$$\mu = p \tag{8.13}$$

and the variance is

$$\sigma^2 = p(1-p) \tag{8.14}$$

In Italy there are 403,454 medical doctors over a population of 59,55 milion people. The ratio is 0.006, therefore there are about 6 medical doctors every 1000 Italians. There is therefore 1 medical doctor every 166.6. Let us approximate to 160. One Italian citizen is selected, what is the probability that he or she is a medical doctor ? and what is the distribution of the number of medical doctors ? We have one single trail. So the condition for the Bernoulli distributions are met so the number of medical doctor will have a Bernoulli distribution with parameter p and it is simply $p = \frac{1}{160}$. If the variable X is the number of medical doctor for a sample of size one, then the probability is:

$$P(X = x) = (\frac{1}{160})^x(1 - \frac{1}{160})^{1-x} \tag{8.15}$$

and this is for $x = 0$ or $x = 1$ since the person will a medical doctor or not. Substituting these values will lead to, for $x = 1$:

$$P(X = x) = (\frac{1}{160})^1(1 - \frac{1}{160})^{1-1} = \frac{1}{160} \tag{8.16}$$

and for $x = 0$

$$P(X = x) = (\frac{1}{160})^0(1 - \frac{1}{160})^{1-0} = \frac{159}{160} \tag{8.17}$$

So one may wonder why to make this calculation for something so simple, since we knew that there was one person every 160. First the *Bernoulli* distribution is a concise and mathematically precise description of the probability. Moreover, this distribution is the base for other common distributions that are built from the *Bernoulli* distribution and from the assumption of *independent Bernoulli trials*, such as the Binomial or the geometric distribution.

8.5 Binomial distribution

Let us consider n *independent Bernoulli trials*, where independent means that the outcome of one trial does not affect the outcome of the following trials. As an example we choose the outcome of getting heads from flipping a fair coin five times. Therefore the *r.v.* $X =$ number of heads from flipping a fair coin 5 times. The possible outcomes are many, we could have $THTHT, HHHTT, HHTTH$ and so forth. The probability of success is denoted p ant it is constant for each experiment. It is important to remember that the values n and p are known and constant and that each experiment has the identical probability of any other.

The question is: *what is the probability of success in n trials* ? For instance, to obtain 30 heads in 100 throws, $\mathsf{P}\{X = k\text{success}, n \text{ trials}\}$. If X is a discrete *r.v.* whose field is $\{0, 1, 2, \ldots, n\}$. We indicate the success with 1 and failure with 0. For instance $n = 10, k = 4$: $1, 1, 0, 0, 0, 1, 0, 0, 1, 0$. Defining with A the set of success and B the set of failure, the probability of the previous outcome is given by the law of combined probability for independent events:

$\mathsf{P}\{A \cap B\} = \mathsf{P}\{A\}\,\mathsf{P}\{A\}$. In general:
$\mathsf{P}\{\bigcap A_i\} = \prod_i \mathsf{P}\{A_i\}$ $\mathsf{P}\{1, 1, 0, 0, 0, 1, 0, 0, 1, 0\} =$

$\mathsf{P}\{1\} \times \mathsf{P}\{1\} \times \cdots \times \mathsf{P}\{1\}$

Therefore $\prod_i \mathsf{P}\{A_i\} = pp(1-p)(1-p)(1-p)p(1-p)(1-p)p(1-p) = p^4(1-p)^{10-4}$
The probability of the outcome of k success in n trials is:
$p^k(1-p)^{n-k}$.
How many of these sequences are possible, with n and k successes ? They are a number equal to the many different ways of partitioning the n in different positions in the sequence of k success and $n-k$ failures. In other words the number of n objects that are equal to eachother. The number of sequences is given by:

$$\frac{n!}{k!(n-k)!} \tag{8.18}$$

Now, the computation of the outcome of number of heads (H) from flipping a coin five times, is computed by using eqn.8.18, where from combinatory theory:

$$5^{C_0} = \frac{n!}{k!(n-k)!} = \frac{5!}{0!(5-0)!} = \frac{5\times4\times3\times2\times1}{(1)(5\times4\times3\times2\times1)} = \frac{120}{120} = 1$$

$$5^{C_1} = \frac{n!}{k!(n-k)!} = \frac{5!}{1!(5-1)!} = \frac{5\times4\times3\times2\times1}{(1)(4\times3\times2\times1)} = \frac{120}{24} = 5$$

$$5^{C_2} = \frac{n!}{k!(n-k)!} = \frac{5!}{2!(5-2)!} = \frac{5\times4\times3\times2\times1}{(2\times1)(3\times2\times1)} = \frac{120}{12} = 10$$

$$5^{C_3} = \frac{n!}{k!(n-k)!} = \frac{5!}{3!(5-3)!} = \frac{5\times4\times3\times2\times1}{(3\times2\times1)(2\times1)} = \frac{120}{12} = 10$$

$$5^{C_4} = \frac{n!}{k!(n-k)!} = \frac{5!}{4!(5-4)!} = \frac{5\times4\times3\times2\times1}{(4\times3\times2\times1)(1)} = \frac{120}{24} = 5$$

$$5^{C_5} = \frac{n!}{k!(n-k)!} = \frac{5!}{4!(5-5)!} = \frac{5 \times 4 \times 3 \times 2 \times 1}{(5 \times 4 \times 3 \times 2 \times 1)(1)} = \frac{120}{120} = 1$$

Therefore the probabilities are:

$$P(x = 0) = \frac{5^{C_0}}{32} = \frac{1}{32} = 0.031$$

$$P(x = 1) = \frac{5^{C_1}}{32} = \frac{5}{32} = 0.156$$

$$P(x = 2) = \frac{5^{C_2}}{32} = \frac{10}{32} = 0.312$$

$$P(x = 3) = \frac{5^{C_3}}{32} = \frac{10}{32} = 0.312$$

$$P(x = 4) = \frac{5^{C_4}}{32} = \frac{5}{32} = 0.156$$

$$P(x = 5) = \frac{5^{C_5}}{32} = \frac{1}{32} = 0.031$$

The binomial r.v. $\mathcal{B}(k; n, p)$ represents the probability of k success in n indipendent trials, each with a probability of success p. We can write:

$$\mathcal{B}(k; n, p) = X_1 + X_2 + \cdots + X_n$$

where $X_i = 1$, if the i-th trial is successful, otherwise = 0. Since $X_i = p$ and $X_i = p(1-p)$, it follows:

$$\frac{\mathcal{B} - np}{\sqrt{np(1-p)}} = \frac{\mathcal{B} - p}{\sqrt{\frac{p(1-p)}{n}}} \approx \mathcal{N}(0, 1)$$

In **R** the binomial distribution is implemented into the function `rbinom(n, ns, p)`. This function has three arguments: the number of random draws (n), the number of coins being flipped on each draw (ns), and the probability of a heads (p) as the outcome. With the function `rbinom(n, ns, p)`, simulates 5 *random* flips of a single coin using a bimodal distribution:

```
n<- 5
ns<- 1
p<- .5
y <- rbinom(n, ns, p)
y
```

The outcomes could be:

```
0 1 0 0 1
1 0 0 0 0
1 0 1 0 1
...
```

where the head is 1 and the tail is 0. To estimate the exact probability density at a given point, the function `dbinom(n, ns, p)` is used. The arguments here are the density being estimated (1 head), the number of coins (5), and the probability of producing a head (0.5).

```
n<- 1
ns<- 5
p<- .5
y <- dbinom(n, ns, p)
y
```

The results of this code is 0.15625, as expected. Replacing the value of `n` in the function will produce the expected values of probabilities described above. To compute all the values the number of successes are computed from 0 to 5:

```
n<- 0:5
ns<- 5
p<- .5
y <- dbinom(n, ns, p)
y
plot(type="h",n,y,xlab="Number of heads",ylab="P(X)")
```

The results are the probabilities as listed above.

```
[1] 0.03125 0.15625 0.31250 0.31250 0.15625 0.03125
```

Figure 8.5 shows the probabilities of heads for trials of 5 throws.

8.6 Normal Distribution

Among the many distributions, the most common one is the Normal Distribution. It is a symmetric, unimodal, bell curve. It is very common simply because many variables tend to follow a normal distribution. For instance the heights of human adults follow a normal distribution. The general symbolic form is:

$$N = (\mu, \sigma) \tag{8.19}$$

where μ is the population mean and σ is the population standard deviation.

The density curve is:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right], \quad -\infty < x < \infty \tag{8.20}$$

where μ and σ are the mean and the standard deviation of the random variable x with density $p\,(x)$ (Fig.8.6).

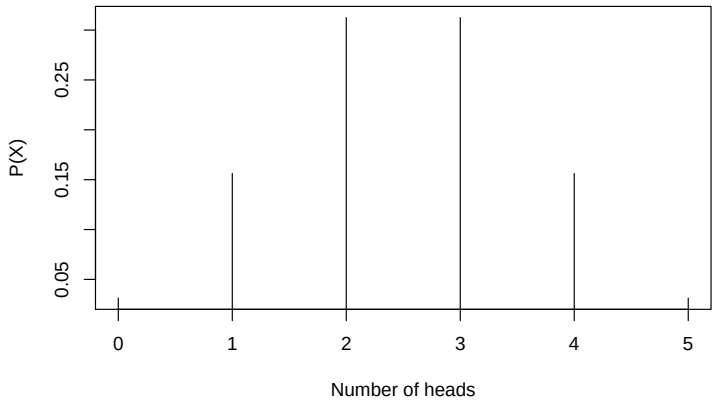


Fig. 8.3 Probabilities of heads for trials of 5 throws.

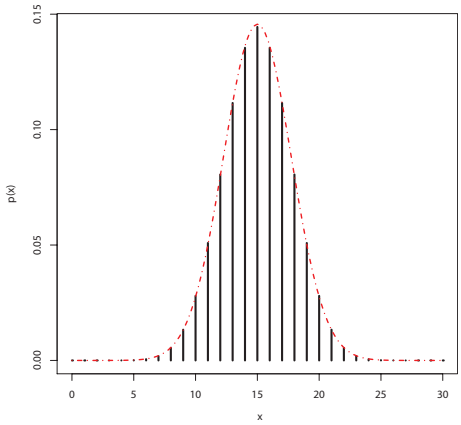


Fig. 8.4 Normal distribution

The bell curve becomes larger at increasing values of standard deviation as depicted in Fig. 8.6. Since it is a density distribution, the area under the curve is always

Usually, the cumulative form of the Gaussian function, is obtained by integration of eqn 8.20 with respect to x:

$$F(x) = \frac{1}{2}\left(1 + \text{erf}\left[\frac{(x-\mu)}{\sigma\sqrt{2}}\right]\right) \text{ for } (x > \mu) \tag{8.21}$$

$$F(x) = \frac{1}{2}\left(1 - \text{erf}\left[\frac{(x-\mu)}{\sigma\sqrt{2}}\right]\right) \text{ for } (x \leq \mu) \tag{8.22}$$

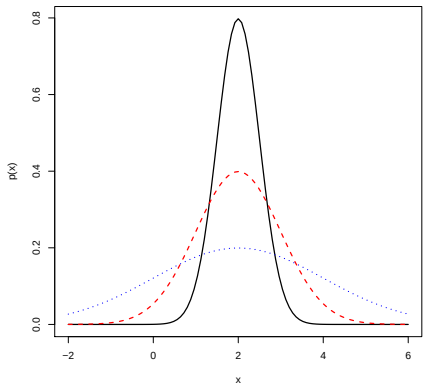


Fig. 8.5 Normal distributions with $\mu = 2$ and $\sigma = 0.5, 1, 2$ (continuos, interrupted lines and points.)

where erf[ ] is the error function. For the computation of the error function a numerical approximation from **?**) can be used:

$$\text{erf}(x) = 1 - (0.3480242T - 0.0958T^2 + 0.7478556T^3)\ \exp\ (-x^2) \tag{8.23}$$

where $T = 1/(1 + 0.47047x)$.
It is possible to generate random numbers having a normal distribution using the instruction `rnorm()`. A normalized distribution having mean equal to zero (`mean=0`) and standard deviation equal to one (`sd=1`) is obtained by specifying the values of mean and standard deviation, as shown below:

```
random_numb_normal <- rnorm(1000,mean=0,sd=1)
plot(random_numb_normal)
```

Figure 8.6 depicts the random numbers, from index 1 to 1000 is:
The numbers can be visualized by using frequency classes.

```
values <- rnorm(1000, mean= 0, sd = 1)
hist(values, col = 'green', freq=F)
```

The graph simple displays the generated numbers from 1 to 1000. Although the distribution is normal, it is not clear from a simple list of numbers. The distribution appears normal when the numbers are plotted by interval of frequency classes. The concept of frequency and classes is described in details in the next chapter. The plot (Fig.8.6) of the distribution looks now more familiar:

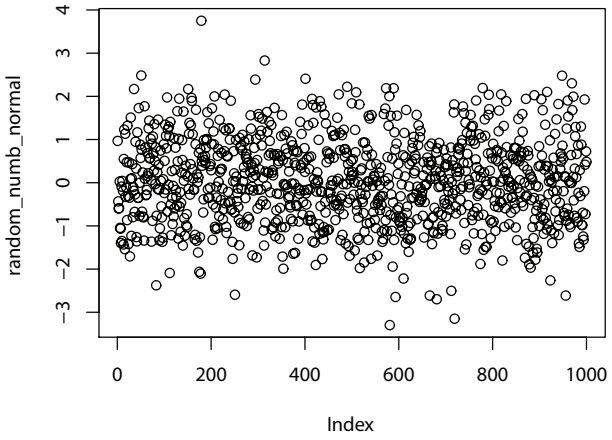


Fig. 8.6 Random numbers

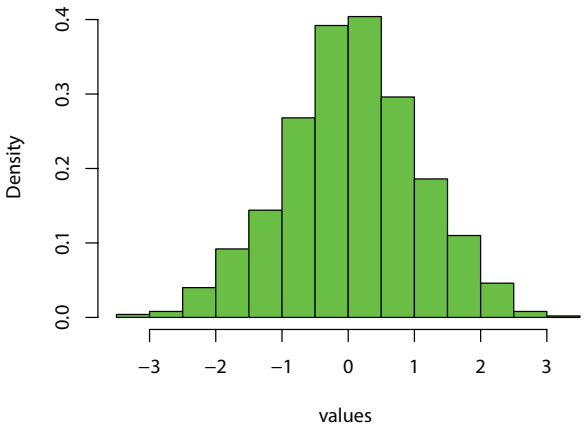


Fig. 8.7 Normal distribution of random numbers by classes

Note that the distribution is normalized with mean = 0 and standard deviation = 1. The code below (also shown in Chapter 7) allows for plotting the PDF and the CDF for the normal distribution.

```
# Create a sequence of numbers between -10 and 10 incrementing by 0.1.
x <- seq(-10, 10, by = .1)

# Choose the mean as 0 and standard deviation as 1.
y <- dnorm(x, mean = 0, sd = 1)
y_cum<-pnorm(x, mean = 0, sd = 1)

plot(x,y,type="l",col="blue",xlim = c(-5,5),ylab=("PDF"))
```

```
plot(y_cum,type="l",col="green",ylab=("CDF"))
```

Figure 8.6 show the PDF and CDF for the normal distribution.

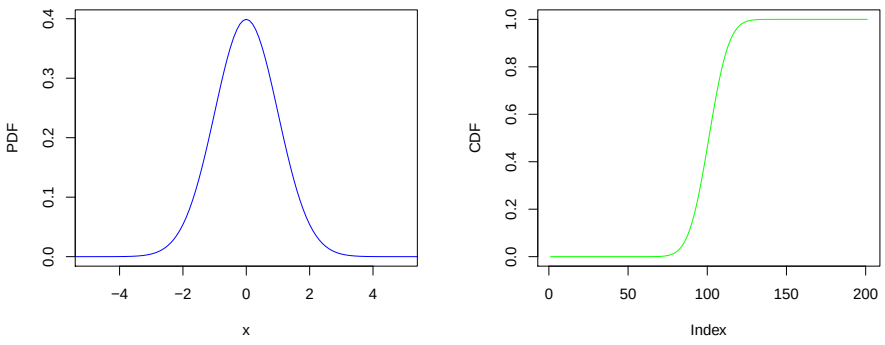


Fig. 8.8 PDF and CDF for a the normal distribution.

8.6.1 Transformations

If the frequency distribution for a soil property follows a normal probability density function, its position and dispersion are easily and conveniently described by the arithmetic mean and the variance. Probabilities of occurrence for various values of the property are easily found. The most useful of the transformations for soil physical properties is the log transform. The frequency distribution of the transformed data is symmetric with a mean of:

$$\langle \ln x \rangle = \frac{1}{n} \sum \ln x \tag{8.24}$$

and variance

$$\sigma^2 = \frac{1}{n-1} \sum (\ln x_i - \langle \ln x \rangle)^2 \tag{8.25}$$

8.6.2 Central limit theorem

The "central limit theorem" says that, under general conditions, the distribution of a *r.v.*, given by a summation of random variables, tends to be distributed as a normal distribution by increasing the number of the additive elements. Be n independent random variables $\{X_1, X_2, \ldots, X_n\}$, all having the same distribution, with finite mean μ and finite variance σ^2. The central limit theorem says that, for $n \to \infty$ (n large):

$$X_1 + X_2 + \cdots + X_n \longrightarrow \mathcal{N}(n\mu, n\sigma^2)$$

Note that this convergence is a *convergence in distribution*. Even if it is said that the sequence of r.v. converges in a distribution, are the partition functions that are converging, not the random variables.

In the standardized form:

$$\frac{X_1 + X_2 + \cdots + X_n - \mathrm{n}\mu}{\sigma\sqrt{\mathrm{n}}} \longrightarrow \mathcal{N}(0,1)$$

the central limit theory is one of the most important theory in statistics, since it states that regardless the shape of the distribution the sample means will normally distribute, allowing for using normal distributions for inferential statistics even for nor−normal distributions for the population.

8.7 t-student distribution

One of the most important test within the branch of inferential statistics is the Students *t−test*. The Student's t−test for two samples is used to test whether two groups (two populations) are different in terms of a quantitative variable, based on the comparison of two samples drawn from these two groups. A Student's t−test for two samples allows for determining whether the two populations from which two samples are drawn are different. Here the t−student distribution is described, while in the section on inferential statistics an example of application of a t−student test is presented.

The normalized Normal distribution is:

$$\mathcal{Z} = \frac{(\bar{X} - \mu)}{\sigma/\sqrt{n}} \tag{8.26}$$

Examples in R were presented where a sample of the random variable was drawn from the population. Commonly, the population standard deviation σ is not known, so it is not possible to use the value σ in the formula, but instead the sample standard deviation s is used:

$$\mathcal{Z} = \frac{(\bar{x} - \mu)}{s/\sqrt{n}} \tag{8.27}$$

the sample standard deviation is not a fixed number but it has a statistical distribution that varies from sample to sample. Therefore it is defined a t−student distribution:

$$t = \frac{(\bar{x} - \mu)}{s/\sqrt{n}} \tag{8.28}$$

This distribution has a similar shape of the normal distribution but with greater variance. It has $(n - 1)$ degree of freedom. As the degree of freedom increases the t-student will tend toward a normal distribution. The code below shows how the t-student distribution approaches the normal distribution at increasing values of df.

Density, distribution function, quantile function and random generation for the t distribution with df degrees of freedom is called with these instructions:

```
dt(x, df, ncp, log = FALSE)
pt(q, df, ncp, lower.tail = TRUE, log.p = FALSE)
qt(p, df, ncp, lower.tail = TRUE, log.p = FALSE)
rt(n, df, ncp)
```

```
x <- seq(-4, 4, length=100)
mean<-mean(x)
sd<-sd(x)
sd
#Apply a normal distribution
y_norm<-dnorm(x,mean = 0, sd =1 )
mean
# Applying the dt() function
y_dt <- dt(x, df = 2)
y_dt_5df <- dt(x, df = 5)
y_dt_10df <- dt(x, df = 10)
y_dt_30df <- dt(x, df = 30)
# Plotting
plot(x,y_dt, type = "l", ylab =("PDF"),las=1,ylim=c(0,0.5),col="yellow")
lines(x,y_dt_5df,col="red")
lines(x,y_dt_10df,col="blue")
lines(x,y_dt_30df,col="green")
lines(x,y_norm,col="black",lty=2, lwd=3)
```

The figure below shows the t−student distribution for different degrees of freedom (2,5,10 and 30). The normalized normal distribution is also depicted in the graph (dotted line). Note that above 30 degrees of freedom the t−student distribution approaches the normal distribution.

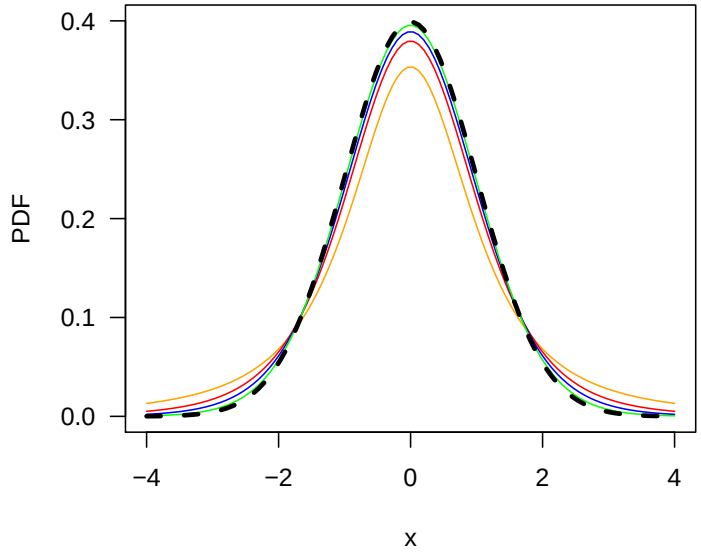


Fig. 8.9 t−student distribution for 2 (orange line),5 (red line), 10 (blue line) and 30 (green line) degrees of freedom. The normal distribution is plotted with black line.

8.8 Poisson distribution

The Poisson distribution is used when a **counting process** is analyzed.

A stocastic process $\{X(t), t \geqslant 0\}$ is considered a counting process if the $X(t)$ represents the total number of counted events. With the notion "up to time t" it means "before and *at* time t". If the event comes exactly at time t, it is counted. This a process with continuous time and with discrete states $s = N$.

The number of people who are entering into a store from the opening time until 10 AM is described as a counting process, but the number of people that are in the store at 10 AM is not a counting process. A counting process may be the number of births of a given animal until a given day, the number of songs written by a musician, the number of car accidents, the number of phone calls arriving at an help−desk.

A counting process $\{X(t), t \geqslant 0\}$ is defined by listing some properties:

1. $\{X(t)\} \geqslant 0$.
2. $\{X(t\})$ it takes only integer values.
3. if $s < t$ then $X(s) \leqslant X(t)$.
4. if $s < t$ then the difference $X(t) - X(s)$ counts the number of events in the time interval $(s, t]$.

Therefore the differences $X(t) - X(s)$ are only non−negative, integer values. In particular if $X(0) = 0$, $\{X(t)\}$ represents the number of counted events in the time interval $[0, t]$. These processes are called *purely discontinuous*. For the counted events, other terms can be used such as "arrivals" or "emissions" from a source. The positive random variables $X(t) - X(s)$ $\forall t, s$ are called *increments* of the process and they are particularly important when they are *indipendent* and *stationary*.

A counting process has independent increments if the number of arrivals, realizations of the r.v. $X(t) - X(s)$, that are happening in any time interval $(s, t]$ are independent from the number of arrivals, realizations of the r.v. $X(v) - X(u)$, occured in any other time interval *disjointed* $(u, v]$, for instance in $[0, s]$. This definition means that the random variables representing the number of arrivals in disjointed intervals are indipendent. It is like the probability of n arrivals in $(s, t]$ does not change even though it is conditioned to the information about the arrivals in different time intervals from $(s, t]$. This independence of increments is the analogue of the independence of trials for the Bernoulli process. Process at independent increments could be the number of customers entering a store or the songs written by a musician, but it is not reasonable to apply it to the number of births. In this case if the number of birth is large enough it is not plausible that the future birth will not depend upon the births occuredbefore.

A counting process has *stationary increments* or *homogeneous increments*, if the number of arrivals in any time interval $[t, t + \Delta t]$ depends only upon the length of the time interval Δt, but not on t, therefore on the position of the interval on the time axis. In other words, a r.v. "number of arrivals" in the interval Δt has the same distribution of the variable "number of arrivals" in $t + \Delta t$.

Hence, the stationarity property of the increments is written:

$$X(z + t + \Delta t) - X(z + t) \sim X(t + \Delta t) - X(t)$$

Or, given $\Delta t = t - s$, with $t > s$:

$$X(t + z) - X(s + z) \sim X(t) - X(s), \qquad \forall z > 0, \quad \forall t > s \tag{8.29}$$

This means that arrivals are " equally likely " for every t. This property is the continuous analogue of the constancy of the probability of success p in a Bernoulli process. In the shop example, that's not reasonable assume stationarity, as there are almost certainly intervals with greater influx of customers. Also in the other examples, the processes that describe them are not stationary. For births, stationarity would be plausible if the population of individuals remained constant; for the musician, his inspiration is likely to vary over time. Finally, note that the fact that the process has stationary increments does not mean that it is stationary, as exemplified by the Poisson process.

The counting process $\{X(t)\}$ is a Poisson process if:

a) $X(0) = 0$. In other words, the origin of the times arises at the moment that the count is started .

b) The process has independent increments.

c) The probability of n of arrivals in a range of finite length Δt is given by the random variable $X(\Delta t)$ with so-called Poisson distribution:

$$\mathsf{P}\{X(\Delta t) = n\} = \frac{(\lambda \Delta t)^n}{n!} e^{-\lambda \Delta t}, \qquad n = 0, 1, \ldots \tag{8.30}$$

Therefore, we have an event that occurs randomly, but the probability of occuring over a certain time interval (Δ t) is constant. For instance the event can occur 4 time per century or 500 times per second. Therefore $\lambda = \frac{4}{century}$ or $\lambda = \frac{500}{second}$.

Since P is constant it does not matter how many times the event has occured before, so the events are independent or uncorrelated, as explained above. Now we want to know the probability that the event will occurs m times into a time interval.

So for an event that happens with $\lambda = \frac{4}{century}$, we may want to know the probability that it will happen once, twice or not at all in the next decade.

The Poisson distribution defines the probability that the random variable X takes the values n. $\mathsf{P}\{X = n\}$ can take any non−negative whole number value. So the value n is the number of time that the variable will take the value (the event happens) in the time interval Δ t and $\mathsf{P}\{X(\Delta t) = n\}$ is the probability that the event will occur in that interval of time.

The mean of the distribution is $\mu = \lambda$ and the variance is also equal to lambda $\sigma^2 = \lambda$. **Overall, the distribution has only one parameter λ and describes the number of events that occur in a fixes time intervals.**

Since the time interval is a constant often the distribution is written as:

$$\mathsf{P}\{X = n\} = \frac{\lambda_1^n}{n!} e^{-\lambda_1}, \qquad n = 0, 1, \ldots \tag{8.31}$$

by defining $\lambda_1 = \lambda \Delta t$

The mean does not have to be an integer value.

The Poisson distribution has different shape depending on the values of λ.

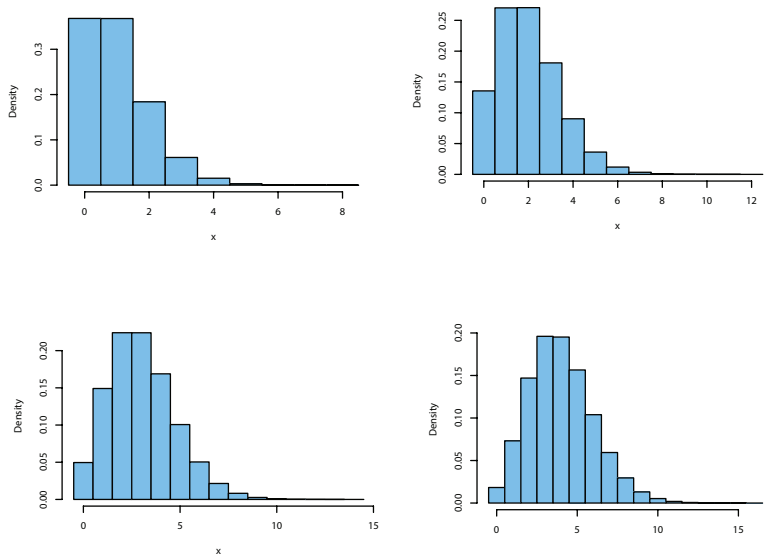


Fig. 8.10 Poisson distribution for λ=1, 2, 3 and 4 from the upper left toward the lower right.

8.8.1 Examples

One nanogram of Plutonium will have an average of 0.31 radioactive decays / 1 sec. We monitor the sample for 10 seconds, what is the probability to one decay occurs during this time ?

Here $\lambda_1 = 0.31 \times 10t$

$$\lambda_1 = \lambda \Delta t = 0.31 \times 10 = 3.1 \tag{8.32}$$

therefore X has a Poisson distribution with $\lambda = 4.6$ and with probability:

$$\mathsf{P}\{n, 3.1\} = \frac{3.1^n e^{-3.1}}{n!} \tag{8.33}$$

It is possible to make a table with $n = 1$ to 7.

Table 8.1 Poisson probability

n	$\mathsf{P}\{n, 3.1\}$ %
0	4.5
1	14
2	21.6
3	22.4
4	17.3
5	10.7
6	5.6
7	2.5

Note that the probability peaks at $n = 3$, which is very close to $\lambda_1 = 3.1$.

```
#POISSON
## lambda is known
set.seed(2)
deltat<- 3600 # 1 hr=3600 sec
lambda<- 0.0028
ev<- 0 # initial time t=0
t<- 0
while (t<deltat) {
       t<- t + rexp(1,lambda)
       ev<- c(ev,t) # adding the t values to zero; as well as ev<-c(0,t)
       print(ev) # print to see who events develop over time
} # end while

ev # arrival time are written
length(ev) # (count ev=0
plot(ev,0:(length(ev)-1),type="s",xlab="tempo (s)",ylab="n. arrival(t)")
# s is for "stair step" it makes steps
abline(v=deltat,lty=3)
# when the system changes state (new arrival) it takes a step on the graph;
# the time between one finish and the next is given by the exponential
```

```
##### The average number of arrivals (snow) in deltat is known
# practically like the first program, here snow lambda is calculated
set.seed(2)
deltat<- 3600
neventi<- 10
lambda<- neventi/deltat
lambda
1/lambda
ev<- 0
t<- 0
```

```
while(t<deltat){
       t<- t + rexp(1,lambda)
       ev <- c(ev,t)
}
ev
length(ev)

plot(ev,0:(length(ev)-1),type="s",xlab="tempo (s)",ylab="no. arrivi(t)")
abline(v=deltat,lty=3)
```

9
Descriptive statistics

9.1 Frequencies

An important representation of data is through the use of frequency histograms or relative frequency histograms. Both of these graphical techniques are applicable only to quantitative data. Before plotting the resultsm, data must be organized. There are different possibilities:

9.1.1 Absolute Frequency

The absolute frequency is the number of times a value appears. It is represented as

$$0 \leq n_i \leq n \qquad \sum_{i=1}^{k} n_i = n$$

where the subscript represents each of the values. So if a value or precipitation (50 mm) appears 4 times, its absolute frequency is 4.

9.1.2 Relative frequency

The relative frequency is the number of times a value appears, divided by the total number of data. It is obtained by dividing the absolute frequency of a certain value by the total number of data.

$f_i = n_i / n$

$$0 \leq f_i \leq 1 \qquad \sum_{i=1}^{k} f_i = 1$$

9.1.3 Percentual frequency

The percentual frequency is the number of times a value appears, divided by the total number of data and multiplied by 100.

$p_i = n_i / n \times 100$

$$0 \leq p_i \leq 100 \qquad \sum_{i=1}^{k} p_i = 100$$

9.1.4 Example

We have $n = 25$ statistical units (people), with the character $X =$ drink (what they drink with pizza). There are 4 modalities, modality = 4 (beer, water, Coke and wine)

The qualitative data are assigned a code, and identifier 1=beer, 2 = water, 3 = Coke and 4 = wine. The data are for 25 customers: 3, 4, 1, 1, 3, 4, 3, 3, 1, 3, 2, 1, 2, 1, 2, 3, 2, 3, 1, 1, 1, 1, 4, 3, 1.

Table 9.1 Distribution of the character X

Modality	Absolute f.	Relative f.	Percentual f.
beer	10	0.40	40
water	4	0.16	16
coke	8	0.32	32
wine	3	0.12	12
Sum	25	1	100

The data organized in the table can be used to construct a *frequency histogram* or a *relative frequency histogram*. In this case on the x-axis, the modality is represented since it is a qualitative value (beer, wine, etc.). In the next example, on the x-axis will be represented classes or intervals, in case a quantitative variable is used.

It is possible to plot these frequencies with, on the left, absolute frequencies and on the right, relative frequencies.

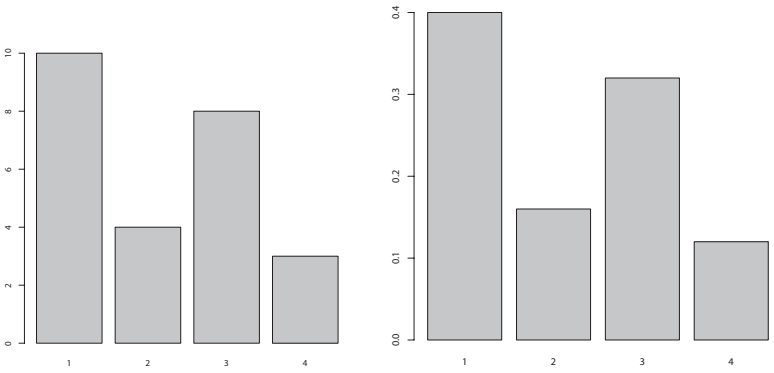


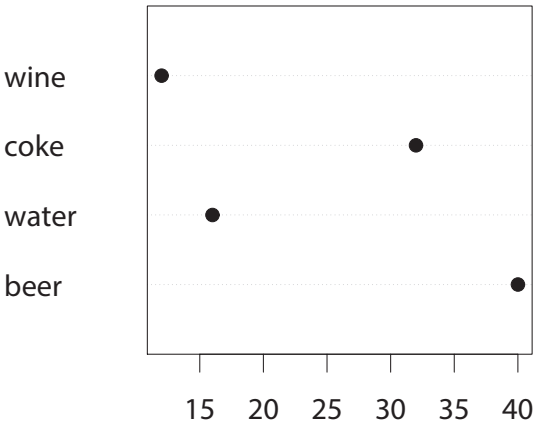
Fig. 9.1 Left: Absolute frequencies; Right: Relative frequencies

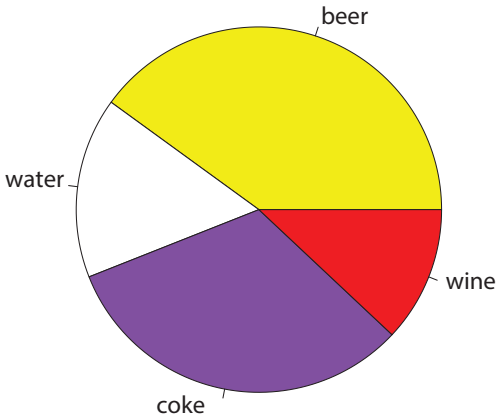
As discussed above the first thing to do is to plot the data. Some data visualizations are better than others. There are many different options. It is possible to use dot graphs, such the one below.

Another possibility are pie charts. Pie charts are a very bad way of displaying information. The eye is good at judging linear measures and bad at judging relative areas. A bar chart or dot chart is a preferable way of displaying this type of data.

The size of acute angles tends to be underestimated, and the size of obtuse angles overestimated. This is one reason pie charts are usually a bad idea. We also misjudge areas poorly. We have known for a long time that area-based comparisons of quantities are easily misinterpreted or exaggerated.

Cleveland (1985): "Data that can be shown by pie charts always can be shown by





a dot chart. This means that judgements of position along a common scale can be made instead of the less accurate angle judgements." This statement is based on the empirical investigations of Cleveland and McGill as well as investigations by perceptual psychologists.

The computation of these frequencies can be performed with R. First a vector `Vtr1` is created to create an index of the 25 measurements.

```
> Vtr1 <- c((1:25))
```

Then, a second vector `Vtr2` stores the recorded values:

```
> Vtr2<-c(3, 4, 1, 1, 3, 4, 3, 3, 1, 3, 2, 1, 2,
       1, 2, 3, 2, 3, 1, 1, 1, 1, 4, 3, 1)
```

Next, a `data.frame` named `drink` is created from the two vectors

```
> drink=data.frame(Vtr1,Vtr2)
```

If the `drink` data.frame is called it returns:

```
drink
   Vtr1 Vtr2
1  1    3
2  2    4
3  3    1
4  4    1
5  5    3
6  6    4
7  7    3
8  8    3
9  9    1
10 10   3
11 11   2
12 12   1
13 13   2
14 14   1
15 15   2
16 16   3
17 17   2
18 18   3
19 19   1
20 20   1
21 21   1
22 22   1
23 23   4
24 24   3
25 25   1
```

The drinks (1=beer, 2 = water, 3 = Coke, 4 = wine) are four types. So we can define it as `factors`, by calling it `drink_type`:

```
drink_type=factor(Vtr2)
drink_type
 [1] 3 4 1 1 3 4 3 3 1 3 2 1 2 1 2 3 2 3 1 1 1 1 4 3 1
Levels: 1 2 3 4
```

When the command `drink_type` is types on the console, it returns the list and the four levels. To get a frequency table of a categorical variable in R, the `count()` function in the `plyr` package is used. The package `plyr` is installed

```
install.packages('plyr')
```

The `count()` function can then be used:

```
> library(plyr)
```

```
> categ_count<- count(drink_type)
> categ_count
  x freq
1 1 10
2 2 4
3 3 8
4 4 3
```

The function returns the number of each categorical types, (10 people ordered beer, 4 ordered water, 8 ordered Coke and 3 ordered wine). This is defined above as **absolute frequency** and a variable `categ_count` was created to save this numbers.

```
> str(categ_count)
'data.frame': 4 obs. of 2 variables:
 $ x : Factor w/ 4 levels "1","2","3","4": 1 2 3 4
 $ freq: int 10 4 8 3
```

Obviously, the total number is 25, one unit for each customer. Now the relative frequency is each value of absolute frequency divided by the total number of units, as shown above.

```
> rel_freq <- categ_count$freq /25
> rel_freq
[1] 0.40 0.16 0.32 0.12
```

Note that, since `categ_count` is a data.frame and not a single variable, only the property $freq was invoked to be then divided by 25. This operation returned the **relative frequency**.

```
> perc_freq <- (categ_count$freq /25)*100
> perc_freq
[1] 40 16 32 12
```

Finally, the **percentual frequency** can be computed.

To plot the results of the frequency distribution, `ggplot` allows for providing the dataframe and the variable. It automatically recognize the four different classes and plot a bar graph.

```
ggplot(data.frame(drink_type), aes(x=Vtr2)) +
+ geom_bar()
```

The output of this instruction is the figure below:

The figure generated by `ggplot` can be saved as encapsulated post script (.eps), which is a good format to manupulating and printing high quality figures.

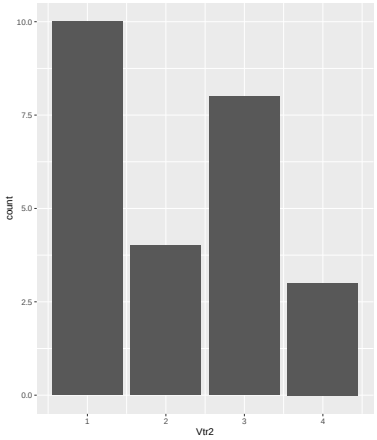


Fig. 9.2 Left: Absolute frequencies plotted with ggplot

9.2 Classes

Now, another example will be shown to introduce the concept of classes. The data shown in the table below describes cumulative annual precipitation (in inches) from 1873 to 1978. (Data are taken from B. Efron, R.J. Tibshirani, An Introduction to the Bootstrap, Chapman & Hall, New York,1993).

The data can be displayed with histograms:

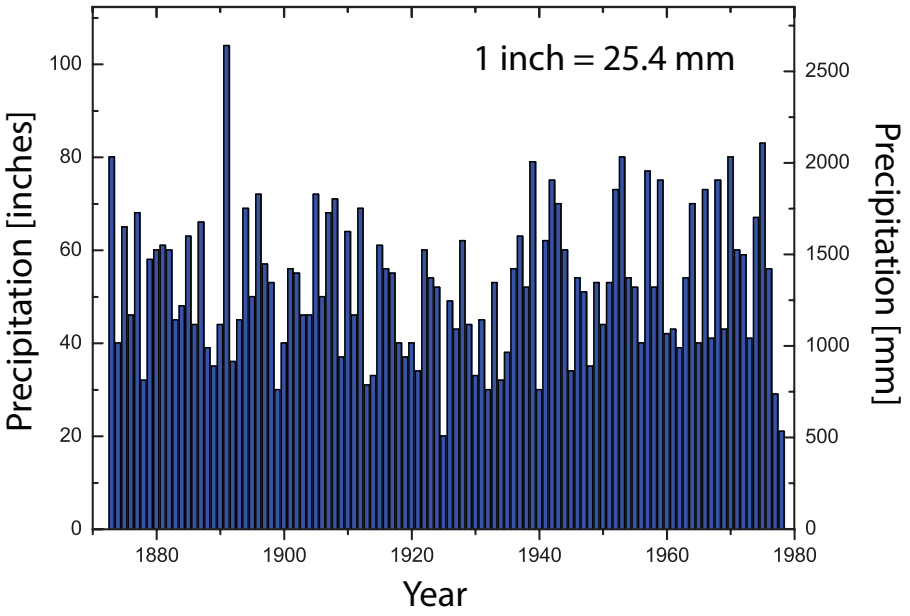


Fig. 9.3 Annual precipitation in Nevada City from 1873 to 1978

Table 9.2 Annual precipitation (inches) at Nevada City from 1873 to 1978

year	inch	year	inch	year	inch	year	inch
1873	80	1900	40	1927	43	1954	54
1874	40	1901	56	1928	62	1955	52
1875	65	1902	55	1929	44	1956	40
1876	46	1903	46	1930	33	1957	77
1877	68	1904	46	1931	45	1958	52
1878	32	1905	72	1932	30	1959	75
1879	58	1906	50	1933	53	1960	42
1880	60	1907	68	1934	32	1961	43
1881	61	1908	71	1935	38	1962	39
1882	60	1909	37	1936	56	1963	54
1883	45	1910	64	1937	63	1964	70
1884	48	1911	46	1938	52	1965	40
1885	63	1912	69	1939	79	1966	73
1886	44	1913	31	1940	30	1967	41
1887	66	1914	33	1941	62	1968	75
1888	39	1915	61	1942	75	1969	43
1889	35	1916	56	1943	70	1970	80
1890	44	1917	55	1944	60	1971	60
1891	104	1918	40	1945	34	1972	59
1892	36	1919	37	1946	54	1973	41
1893	45	1920	40	1947	51	1974	67
1894	69	1921	34	1948	35	1975	83
1895	50	1922	60	1949	53	1976	56
1896	72	1923	54	1950	44	1977	29
1897	57	1924	52	1951	53	1978	21
1898	53	1925	20	1952	73		
1899	30	1926	49	1953	80		

or with points:

A useful method to describe data is to create classes. Classes are simply intervals, classes of ranges) of the variable. For instance in the Table 9.3, classes of 200 mm were created.

Table 9.3 Distribution of frequencies of "Precipitation in Nevada City"

Precipitation (mm)	Absolute freq.	Relative freq.	Percentual freq.
$[200, 400)$	0	0	0
$[400, 600)$	2	0.019	1.9
$[600, 800)$	5	0.047	4.7
$[800, 1000)$	14	0.132	13.2
$[1000, 1200)$	23	0.217	21.7
$[1200, 1400)$	19	0.179	17.9
$[1400, 1600)$	16	0.151	15.1
$[1600, 1800)$	12	0.113	11.3
$[1800, 2000)$	9	0.085	8.5
$[2000, 2200)$	5	0.047	4.7
$[2200, 2400)$	0	0	0
$[2400, 2600)$	0	0	0
$[2600, 2800)$	1	0.009	0.9
Sum	106	1	100

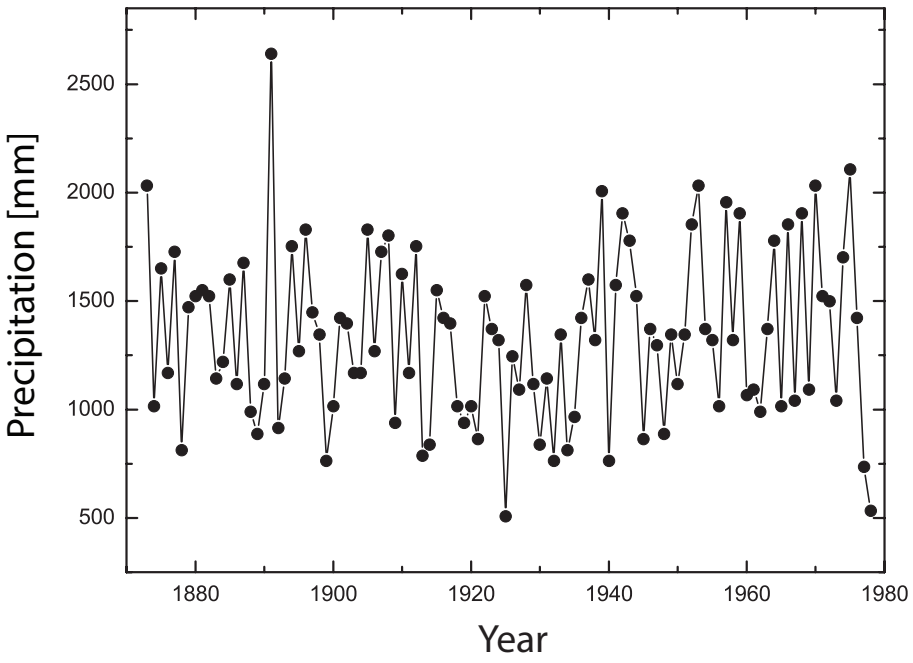


Fig. 9.4 Annual precipitation in Nevada City from 1873 to 1978

In this case, to perform this operation with R, the data file must be imported. This instruction has been shown in the Chapter *Data Management*. In this case the file is called `Nevada_prec.dat` and the separator is a `tab`.

```
#CODE Ch9_1.R
Nevada_prec <- read.table("C:/Users/marco.bittelli.PERSONALE/Documents
/Didattica/R_class_2/exercises/descriptive_stat/Nevada_prec.dat",
sep = "", check.names = FALSE, header = T, na.strings = c("NA", "NAN"))
```

The instruction `read.table` reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file. Now the data in the second column (annual cumulative precipitation) should be cumulated and classified. Before performing this operation, the data are transformed into [mm].

```
Nevada_prec$Prec_mm = Nevada_prec$Prec * 25.4
```

Here a new variable `Prec_mm` has been created. This can be inquired with the instruction `str()`.

```
str(Nevada_prec)
'data.frame': 106 obs. of 3 variables:
 $ Year : int 1873 1874 1875 1876 1877 1878 ...
 $ Prec : int 80 40 65 46 68 32 58 60 61 60 ...
 $ Prec_mm: num 2032 1016 1651 1168 1727 ...
```

Data can be plotted in any moment, with a simple instruction like

```
> plot(Nevada_prec$Prec_mm)
```

It is possible to collect data into **classes**, of type $x_1 \dashv x_{i+1}$ or $x_1 \vdash x_{i+1}$, in case it is preferred to include the lower limit instead of the upper limit by specifying `right= FALSE` as an additional argument of the function.

```
classes<-table(cut(Nevada_prec$Prec_mm, breaks=c(0,200,400,600,800,1000,
1200,1400,1600,1800,2000,2200,2400,2600,2800)),right=TRUE )
```

The output of this statement is

```
> classes
(0,200] (200,400] (400,600] (600,800] ...
0 0 2 5 ...
```

which are the number of events belonging to the different classes. Also notice that R printed the upper value as a close interval as specified. Note that classes are obtained if they start from zero.

There is another method to obtain classes, by using the function `hist()`:

```
histogram=hist(Nevada_prec$Prec_mm, c(0,200,400,600,800,1000,
1200,1400,1600,1800,2000,2200,2400,2600,2800),plot=TRUE)
```

The structure of the variable is then:

```
str(histogram)
List of 6
 $ breaks : num [1:15] 0 200 400 600 800 1000 1200 1400 1600 1800 ...
 $ counts : int [1:14] 0 0 2 5 14 23 19 16 12 9 ...
 $ density : num [1:14] 0.00 0.00 9.43e-05 2.36e-04 6.60e-04 ...
 $ mids : num [1:14] 100 300 500 700 900 1100 1300 1500 1700 1900 ...
 $ xname : chr "Nevada_prec$Prec_mm"
 $ equidist: logi TRUE
 - attr(*, "class")= chr "histogram"
```

where the `breaks` are specified, then the number of samples `count`ed within each class, the density, etc.

The classification by classes is useful to understand the frequency of given events. In the graph below, the occurrence of years with precipitation comprised between 400 and 600 mm.
Often it is useful to normalize the distribution to obtain a better representation of the occurrence of events.

9.2.1 Tabular and Graphical representation

The values for the cumulative frequencies are presented in the table below

A very useful representation of the cumulative frequencies is the use of cumulative graphs. How many units of the ensemble have a value smaller than x^* ? In the 20%

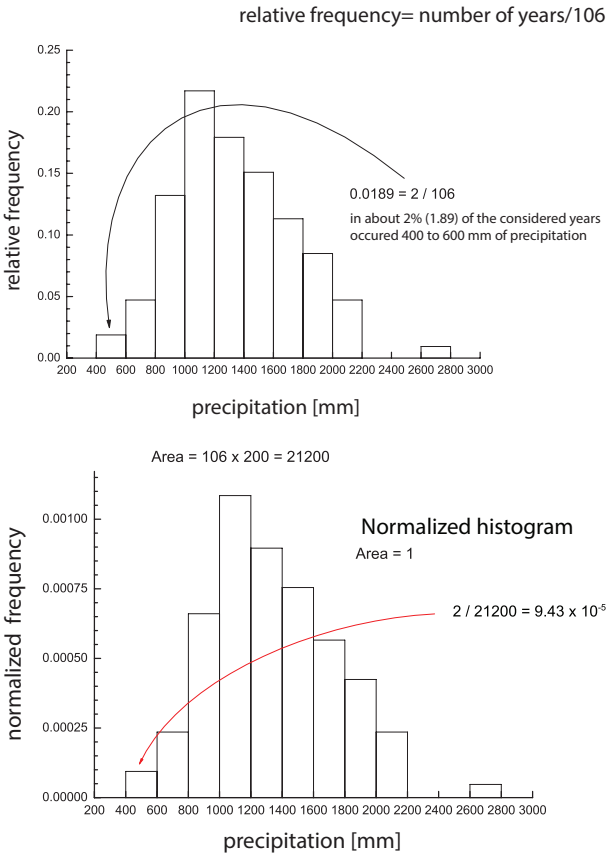


Table 9.4 Distribution of frequencies for "precipitation at Nevada City"

precipitazioni (mm)	n_i	N_i	F_i	P_i
$[200, 400)$	0	0	0	0
$[400, 600)$	2	2 + 0 = 2	0.02	1.89
$[600, 800)$	5	2 + 5 = 7	0.07	6.61
$[800, 1000)$	14	7 + 14 = 21	0.20	19.82
$[1000, 1200)$	23	21 + 23 = 44	0.42	41.51
$[1200, 1400)$	19	44 + 19 = 63	0.59	59.43
$[1400, 1600)$	16	63 + 16 = 79	0.75	74.53
$[1600, 1800)$	12	79 + 12 = 91	0.86	85.85
$[1800, 2000)$	9	91 + 9 = 100	0.94	94.34
$[2000, 2200)$	5	100 + 5 = 105	0.99	99.06
$[2200, 2400)$	0	105 + 0 = 105	0.99	99.06
$[2400, 2600)$	0	105 + 0 = 105	0.99	99.06
$[2600, 2800)$	1	105 + 1 = 106	1	100

(19.8%) of the years there are precipitation less than 1000 [mm].In the 6% of the years precipitation are *not* less than 2000 mm. For 94.34% of the years, precipitation are < di 2000 mm, therefore ≥ 2000 mm: 100 − 94.34 ≈ 6). In the figure below, cumulative frequencies are plotted:

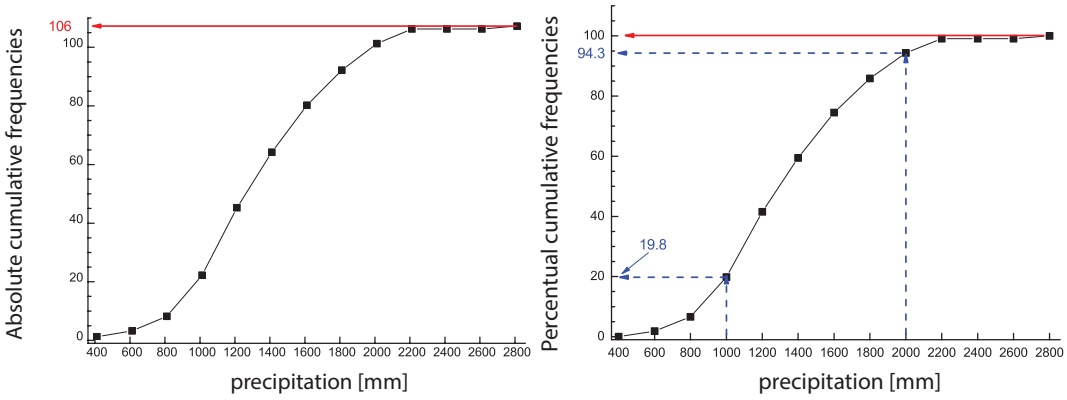


Fig. 9.5 Absolute (left) and relative (right) cumulative precipitation at the Nevada city experimental station (Nevada, USA).

9.3 Cumulative curves and frequencies

A cumulative curve is often important to understand the behaviour of a variable. In this example, daily precipitation data collected in the Emilia Romagna region (Italy) from 1961 to 2018, are used (**?**). The original data set is available at the Eraclito Project website.

As described above data can be represented using frequencies distribution and classes to obtain a first idea at the distribution. A very useful representation is a cumulative curve, frequency and distribution. For instance, cumulative precipitation curves as function of time, provide an idea of the seasons where precipitation occurs, as well as the total amount of precipitation over a given period. For instance, if the cumulative curve is steep in the period March−June and Sept−December, it means that a large amount of precipitation occurs in the spring and fall. This is typical of Mediterranean climates. On the other hand, in tropical areas large amount of precipitation occurs in the summer months due to the Monsoon, determining a steep cumulative curve during summer.

9.3.1 Absolute Cumulative Frequency

The **absolute cumulative frequency** of a mode is

$$N_i = n_1 + n_2 + \cdots + n_i$$

is given by the summation of the individual data. The code below shows how to compute and plot a cumulative curve for data from the experimental station of Cadriano (Bologna, Italy) for the year 1961. The code below reads the data from an excel file. It parse the date value as a date format and define variables to select the time period to use to compute the cumulative curve. A `while` loop is written to loop over the period and compute the cumulative curve.

```
#CODE Ch9_2.R
library(xts) ### provides functionality for working with time series
library(lubridate) ### manage dates
library(dplyr)
library(readxl) ### import excel file
#The excel file Prec_ER_daily.xlsx is opened, the structure of the
#newly created dataframe is analysed and the data are visualized.

setwd("~/Didattica/R_class_4/exercises/Ch9_statistical_inference")
Prec_ER_daily <- read_excel("Prec_ER_daily.xlsx")

# Only data from the Cadriano station are analyzed
Prec_ER_daily_cadriano <- Prec_ER_daily[,c("date","cadriano")]
prec_cadriano<-Prec_ER_daily_cadriano$cadriano
str(Prec_ER_daily_cadriano)

#Daily cumulative curve

date<-as.Date(Prec_ER_daily_cadriano$date)
start <- date[1]
end <- date[365]
start
end
theDate
as.Date(theDate)
theDate<-start
theDate
vec_date<-vector()
precip<-vector()
cumprec_vec<-vector()
prec_daily<-0
cumprec<-0

i<-1

while (theDate <= end){
       vec_date[i]<-theDate
       prec_daily=prec_cadriano[i]
       cumprec<-cumprec + prec_cadriano[i]
       cumprec_vec[i]<-cumprec
       output<-c(i,prec_daily,cumprec)
       print(output)
       theDate <-theDate + days(1)
       i<-i+1
}
```

```
#plotting
plot(as.Date(vec_date),cumprec_vec,type="l",xlab="Time [day]"
,ylab="Cumulative precipitation [mm]")
```

Figure 9.6 depicts cumulative precipitation for the Cadriano experimental station. As described above, in the year 1961, about 200 mm of precipitation occurred between January−March, about 400 mm in the period April−October and 600 mm in the period October−January. Clearly, there is variability among different years, and estimators must be used to derive more detailed information.

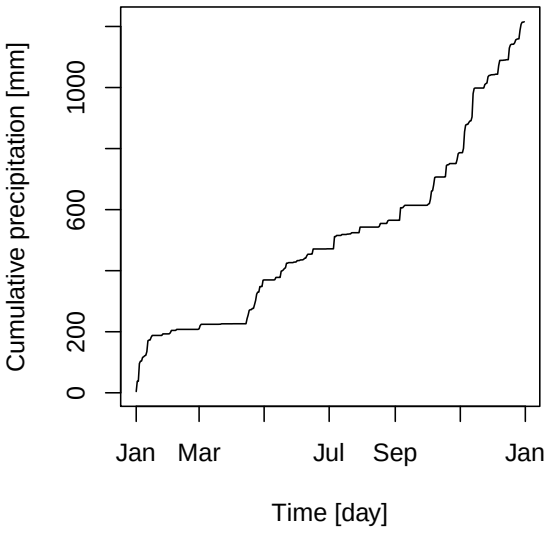


Fig. 9.6 Cumulative precipitation at the Cadriano (Bologna, Italy) experimental station for the year 1961.

9.3.2 Relative Cumulative Frequency

The data above are plotted as absolute values. Another useful way of plotting these data is to use a relative cumulative curve. The **relative cumulative frequency** of a mode is

$$F_i = f_1 + f_2 + \cdots + f_i$$

```
#CODE Ch9_9_2.R
library(xts) ### provides functionality for working with time series
library(lubridate) ### manage dates
library(dplyr)
library(readxl) ### import excel file
#The excel file Prec_ER_daily.xlsx is opened, the structure of the
#newly created dataframe is analysed and the data are visualized.

setwd("~/Didattica/R_class_4/exercises/Ch9_statistical_inference")
Prec_ER_daily <- read_excel("OpenDataFiles/data/Prec_ER_daily.xlsx")

# Only data from the Cadriano station are analyzed
Prec_ER_daily_cadriano <- Prec_ER_daily[,c("date","cadriano")]
prec_cadriano<-Prec_ER_daily_cadriano$cadriano
str(Prec_ER_daily_cadriano)

#Daily cumulative curve
max_prec_value<-sum(Prec_ER_daily_cadriano$cadriano[1:365])

date<-as.Date(Prec_ER_daily_cadriano$date)
start <- date[1]
end <- date[365]
start
end
theDate
as.Date(theDate)
theDate<-start
theDate
vec_date<-vector()
precip<-vector()
prec_daily<-0
cumprec<-0
cumprecrel<-0
cumprec_vec<-vector()
cumprec_vec_rel<-vector()
i<-1

while (theDate <= end){
       vec_date[i]<-theDate
       prec_daily=prec_cadriano[i]
       cumprec<-cumprec + prec_cadriano[i]
       cumprec_vec[i]<-cumprec
       cumprecrel<-cumprec/max_prec_value
       cumprec_vec_rel[i]<-cumprecrel
       output<-c(i,prec_daily,cumprecrel)
```

```
        print(output)
        theDate <-theDate + days(1)
        i<-i+1
}

#plotting
plot(as.Date(vec_date),cumprec_vec_rel,type="l",
xlab="Time [day]",ylab="Cumulative precipitation [mm]")
```

Figure 9.7 depicts relative cumulative precipitation for the Cadriano experimental station.

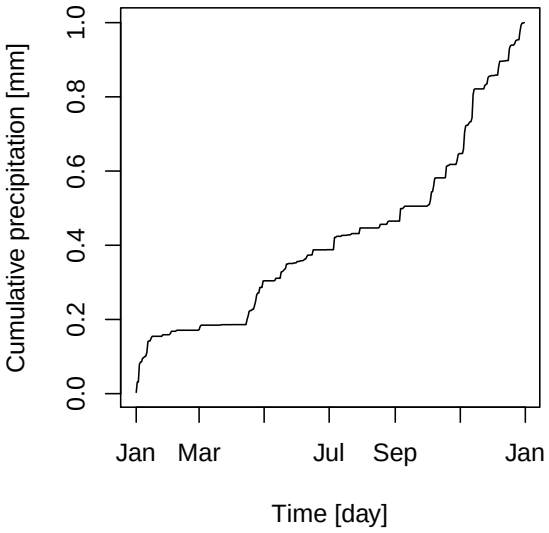


Fig. 9.7 Relative cumulative precipitation at the Cadriano (Bologna, Italy) experimental station for the year 1961.

9.3.3 Percentual Cumulative Frequency

The **percentual cumulative frequency** of a mode is

$$P_i = p_1 + p_2 + \cdots + p_i$$

where the data plotted above are expressed as percentage instead of fraction.

9.4 Measures of Central Tendency

The simplest statistical inference problem is **Point estimation**, where a single value (statistic) from the sample data to estimate a population parameter.

In general, define θ as a parameter of the population described by random variables X. X is unknown (the distribution is unknown), and θ is unknown. The quantity θ is called *parameter* (of interest) and it is a constant of the population. θ is a measure of the distribution of one or more character of the population, for instance the mean μ or the variance σ^2.

9.4.1 Means

The arithmetic mean is defined as the sum of the measurements divided by the total number of measurements. Usually, the *population mean* is denoted by the greek letter μ, while the *sample mean* is denoted by the symbol $\bar{x}$. The population mean is computed over the complete set of measurements, while the sample mean is computed over a subset sample of the measurements.

$$\bar{x} = \frac{\sum_i x_i}{n} \tag{9.1}$$

where $\bar{x}$ is the arithmetic mean, x_i is each individual measurement and n is the total number of measurements. In the example below, the vector `x` is defined, containing a number n of measurements and to the variable `am` (which stands for arithmetic mean) is assigned the output of the function `mean(x)`. The function `mean()` is an internal function of R, that computes the arithmetic mean. Note that the total number of elements (`n`) is read by the function as elements of the vectors. Finally, the result is printed on screen.

```
> x<- c(5.24, 5.55, 4.69, 4.39, 6.87, 5.15 ,
4.61, 5.20, 5.49, 4.81,2.74, 3.50, 5.19, 5.40,
3.81, 6.49, 6.34, 4.45, 5.10, 3.17)
> am<- mean(x)
> am
[1] 4.9095
>
```

Note that if the data have no values (NA), it must be specified in the mean as

```
> x<- c(5.24, 5.55, 4.69, 4.39, 6.87, 5.15 ,
4.61, 5.20, NA, 4.81,2.74, 3.50, 5.19, 5.40,
3.81, 6.49, 6.34, 4.45, NA, 3.17)
```

```
> am<- mean(x,na.rm=TRUE)
> am
[1] 4.9095
>
```

It means that the NA (not available data) are removed from the computation of the mean. This must be used also for other parameters like variance and standard deviation.

There are other means that can be used. Important ones are the **geometric**, the **harmonic** and the **logarithmic** means. The **geometric mean** is a mean or average, which indicates the central tendency or typical value of a set of numbers by using the product of their values:

$$\left(\prod_{i=1}^{n} x_n\right)^{1/n} = \sqrt[n]{x_1 x_2 x_n} \tag{9.2}$$

This mean is used when the elements of the sample ranges several orders of magnitude and the arithmetic mean is not a representative value of central tendency. If we have two properties, the first one takes bigger numbers (from 300 to 450), while the second takes small numbers (from 6 to 3). In the first case it is an increase in 50%, while in the second case it is a decrease in 50%.

With the use of an arithmetic mean, the first one has a much higher weight, with the mean going from 153 to 227, because the first number is increasing but the second number, although it is decreasing, has little effect.

```
> x<-c(300,6)
> y<-c(450,3)
> mean(x)
[1] 153
> mean(y)
[1] 226.5
```

With the geometric mean the second variable is weighted more heavily, with the average going from 42.4 to 36.7.

```
> library(EnvStats)
> x<-c(300,6)
> y<-c(450,3)
> geoMean(x)
[1] 42.42
> geoMean(y)
[1] 36.74
```

In general the geometric mean is a more accurate estimator when the progression is multiplicative instead of additive. A geometrical analogy could be drawn if we plot the numbers on a linear scale for values that are close, and on a exponential curve. In the second case the geometric mean will return a more accurate representation of the distribution of numbers.

Another useful mean is the **harmonic mean**:

$$H = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \frac{1}{x_n}} = \left(\frac{\sum_{i=1}^{n} x_i^{-1}}{n} \right)^{-1} \tag{9.3}$$

Since the harmonic mean is based on reciprocal of numbers, it is useful when dealing with ratios and relationship among ratios. Other measures of position are the **median** and the **mode**.

The **median** is the value separating the higher half from the lower half of a data sample. It is the value in the middle of the distribution. So for odd number of elements it is (1,3,4,**6**,8,9,11). For even number of elements, it is the mean between the two middle number(1,3,4,**6,8**,9,11,13), therefore $6 + 8/2 = 7$. Therefore the median value is the central value, which is the value having half the observations smaller and half larger than it.

The **mode** is the value of y at which the frequency distribution is maximum. For symmetrical distributions, the median, mean and mode coincide.

The means are also referred to each class.

9.4.2 Means for classes

The arithmetic mean is a quantitative property that can be subdivided in classes:

$$\bar{x} = (1/n) \sum_{i=1}^{N} c_i n_i \tag{9.4}$$

where c_i is the central value of the i-th class and n_i is the absolute frequency. If $c_i = \bar{x}_i$, there are no approximations.

An example is presented, with the following classes: $[0, 5), [5, 10), [10, 15)$, therefore with $N = 3$. The absolute frequencies are n_i: $3, 5, 2$. The central values are $(5 + 0)/2 = 2.5, (5 + 10)/2 = 7.5, (10 + 15)/2 = 12.5$. The mean is $\approx [(3 \times 2.5) + (5 \times 7.5) + (2 \times 12.5)]/(3 + 5 + 2) = 7$. Given n data in N classes, of numerosity $n_1, \ldots, n_N$; x_{ij}, with i-th data of the j-th class.

The mean of the j-th class is:

$$\bar{x}_j = (1/n_j) \sum_{i=1}^{n_j} x_{ij} \; (j = 1, \ldots, N) \tag{9.5}$$

The mean of the n data is:

$$\bar{x} = \frac{1}{n} \sum_{j=1}^{N} \sum_{i=1}^{n_j} x_{ij} = \frac{1}{n} \sum_{j=1}^{N} n_j \bar{x}_j = \sum_{j=1}^{N} f_j \bar{x}_j$$

The difference (residuals) from the mean is $x_i - \bar{x}$, the sum of the residuals $\sum_{i=1}^{n} (x_i - \bar{x}) = 0$ and $\sum_{i=1}^{n} (x_i - a)^2 =$ minimum for $a = \bar{x}$ namely, the mean is the value that is *closer* to all the observations.

9.4.3 Median

9.4.4 Mode

9.5 Measures of Variability

It is important to use indicators of variability in a distribution of data. The simplest but useful measurement is the **range**, which defines the interval between the minimum and maximum value of the distribution. Many different measures can be obtained by measuring the deviations $y - \bar{y}$. The first that would come to mind is the mean deviation. However, if deviations have opposite signs, the total mean deviation could be zero. Therefore, a possibility is to ignore the minus sign and compute the absolute values. However, a more easily interpreted function of the deviations is the **variance**, which is the sum of the squared deviations of the measurements from their mean.

$$s^2 = \frac{\sum_i (x_i - \bar{x})^2}{n - 1} \tag{9.6}$$

Also for the measures of variability, different symbols are used, where s^2 is the **sample variance** and σ^2 is the **population variance**. The use of $(n-1)$ is not arbitrary, since it makes an unbiased estimator of the population variance. If we were to draw a very large number of samples, each of size n, from the population of interest and we compute s^2 for each sample, the average sample variance would equal the population variance σ^2. Had we have divided by n in the definition of the sample variance (s^2), the average sample variance computed from a large number of samples would be less than the population variance, hence s^2 would tend to underestimate σ^2. Ideally, the population variance σ^2, would be computed as

$$\sigma^2 = \frac{\sum_i (x_i - \mu)^2}{n} \tag{9.7}$$

but often the entire population is unknown, so we replace the population mean with its best estimate that is the sample mean, as shown in eqn 9.6. The problem is that subtracting $\bar{y}$ from the sum makes the sum as small as it possibly could be. Since the value of $\bar{y}$ is centered around the distribution, while μ can be any value. Therefore, the population variance is likely to be larger than the sample variance because deviations of sample values from μ are likely to be larger than deviations from $\bar{y}$. For this reason, the sample variance is divided by $(n-1)$ to allow for larger values of sample variance. It can be proved mathematically that using $(n-1)$ allows for an unbiased estimation of population variance. However, if possible, it is generally more useful to know the mean and variance of the population rather than that of the sample.

The **sample coviariance** is a measure of the relationship between two paired datasets:

$$\mathrm{Cov}_{x,y} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) \tag{9.8}$$

9.6 Coefficient of variation

The coefficient if variation is a adimensional number:

$$CV = \frac{\sigma}{\bar{x}} \tag{9.9}$$

it can also be expressed in percentage:

$$CV = \frac{\sigma}{\bar{x}} \times 100 \tag{9.10}$$

it is used when all the values of the distribution are positive. It compares the variability among variables of different nature. For instance, the length and weight of a group of cats have been measured. The average length is 45 cm and the standard deviation is 5 cm $\Rightarrow CV_l \approx 11\%$. After weighting them, the mean is 6.5 Kg and the standard deviation is 1.5 Kg $\Rightarrow CV_p \approx 23\%$. Therefore, with respect to the mean, the weight of cats is more variable than their length.

9.7 Quantiles

Quantiles are cutting points that divide the range of a probability distribution into continuous intervals with equal probabilities, or dividing the observations in a sample.

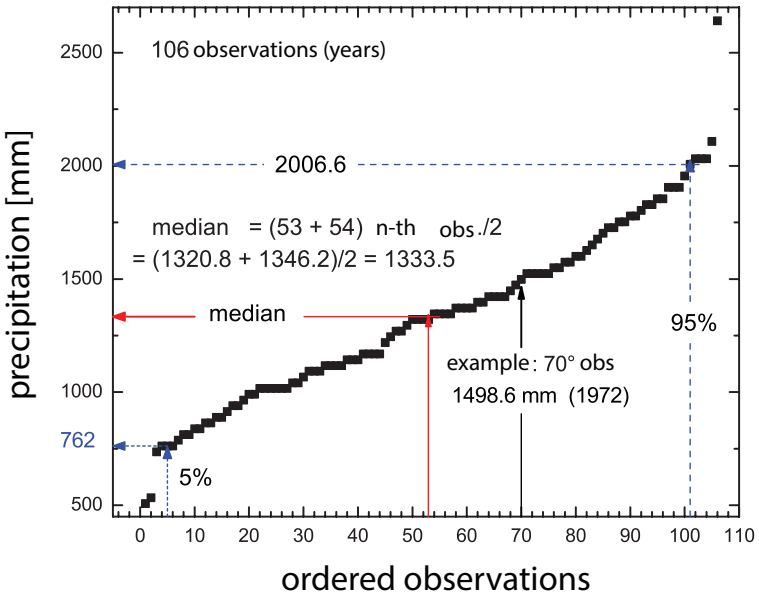
For instance quartiles are the three cut points that divide a dataset into four equal−sized groups (0−25, 25−50, 50−75 and 75−100%). A decile is a cut point that divide the distribution in ten intervals.

The precipitation data are ordered in incremental values as shown in the table below.

Table 9.5

	anno	inch	mm		ordinati (mm)
1	1873	80	2032	1	508
2	1874	40	1016	2	533.4
3	1875	65	1651	3	736.6
4	1876	46	1168.4	4	762
5	1877	68	1727.2	5	762
6	1878	32	812.8	6	762
7	1879	58	1473.2	7	787.4
8	1880	60	1524	8	812.8
9	1881	61	1549.4	9	812.8
10	1882	60	1524	10	838.2
...	...	...	...	...	...
97	1969	43	1092.2	97	1905
98	1970	80	2032	98	1905
99	1971	60	1524	99	1905
100	1972	59	1498.6	100	1955.8
101	1973	41	1041.4	101	2006.6
102	1974	67	1701.8	102	2032
103	1975	83	2108.2	103	2032
104	1976	56	1422.4	104	2032
105	1977	29	736.6	105	2108.2
106	1978	21	533.4	106	2641.6

The ordered data are plotted. The graph depicts the median value, computed as described above, the value of the 70^{th} observation, and the 5 and 95 % quantiles.
The instruction to order data is shown below using the function `sort()`.




```
NevadaPrec <- read.table("C:/Users/marco.bittelli.PERSONALE/
Documents/Didattica/R_class_3/exercises/descriptive_stat
/data/Nevada_prec.dat",sep = "", check.names = FALSE,
header = TRUE, na.strings = c("NA", "NAN"))

NevadaPrec$Prec_mm = NevadaPrec$Prec * 25.4

NevadaPrec$Year <- as.numeric(NevadaPrec$Year)

str(NevadaPrec)

#sorting data in increasing order
NevadaPrec$Prec_mm_sorted<- sort(NevadaPrec$Prec_mm,
decreasing = FALSE, na.last = NA)
NevadaPrec$Prec_mm_sorted
```

The quantiles are computed with the instruction below.

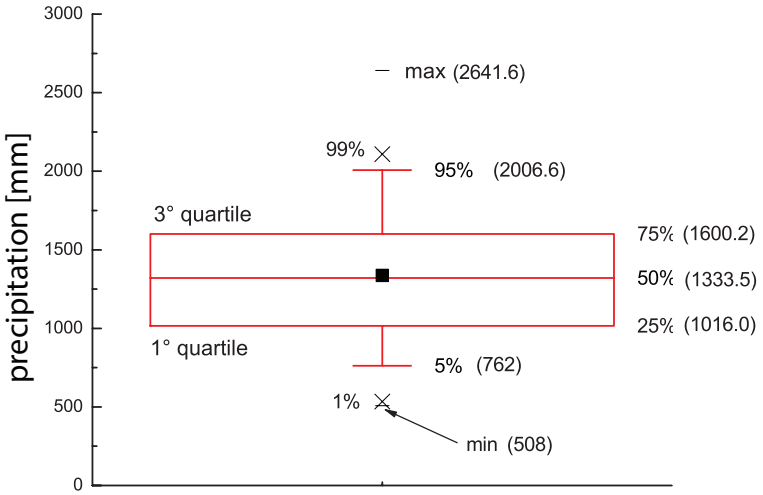
```
#compute quantiles
quantile(NevadaPrec$Prec_mm_sorted,probs = seq(0, 1, 0.05))
0% 5% 10% 15% 20% 25% 30% 35% ...
 508.00 768.35 850.90 933.45 1016.00 1022.35 1092.20 1136.65 ...
```

9.8 The box plot

The box plot is a useful graph to represent the data. It provides information about the simmetry of the distribution and incorporates numerical measures of central tendency. The graph belows show the an *histogram* by classes (left) and a *box plot* on the right. The box plot provided information about the variability of the distribution as well. The box plot uses the median and **hinges** of a distribution. Hinges are very similar to quartiles, but depending on the distribution they may differ very slightly from the first and third quartiles. The instruction below shows how to plot the box plot of the precipitation data.

```
#plot the box plot
boxplot(NevadaPrec$Prec_mm_sorted)
```

The **box plot** (also called **skeletal box plot**) is constructed by drawing a box between the lower and upper quartiles, with a solid line drawn accross the box to locate the median. Usually a straight line is drawn to connect the box to the largest value and another straight line is drawn to connect the box the smallest value.



9.9 Exercises

1. Download daily precipitation from ARPAE web site Dexter 3. Import the data, format it to be parsed into dates and run frequency analysis
2. Separate the daily precipitation data in classes. Compute mean and standard deviation. Plot the cumulative graph and discuss the results.
3. Plot the box plot for the data.

10
Inferential statistics

Statistical inference is the process of data analysis to infer properties of an underlying probability distribution. Inferential statistical analysis *infers* properties of a population, for example by testing hypotheses and deriving estimates.

Descriptive statistics is solely concerned with properties of the observed data, and it does not rest on the assumption that the data come from a larger population. On the other hand inferential statistics attempts to derive information about the entire population from knowledge acquired from samples. A lot can go wrong with statistical inference, and this is one reason that beginners are so anxious about it.

In his book Statistical Rethinking, R. McElreath (2015) presents a flow chart to be used as a general guideline when facing a statistical problem. Statistical tools are so many and so different that it is hard for the beginner to select the appropriate direction.

One of the most import problems is the tacit belief that the proper objective of statistical inference is to test null hypotheses. This concept is based on the well−known theory presented by Karl Popper (1902–1994) who argued that science advances by falsifying hypotheses. He argued that science works by developing hypotheses that are, in principle, falsifiable. Many philosophers of science argued that science is not described by the falsification standard, as Popper recognized and argued. In fact, deductive falsification is impossible in nearly every scientific context. In this section, R. McElreath presented two reasons for this impossibility.

- Hypotheses are not models. The relations among hypotheses and different kinds of models are complex. Many models correspond to the same hypothesis, and many hypotheses correspond to a single model. This makes strict falsification impossible.
- Measurement matters. Even when we think the data falsify a model, another observer will debate our methods and measures. They don't trust the data. Sometimes they are right.

For both of these reasons, deductive falsification never works. The scientific method cannot be reduced to a statistical procedure, and so our statistical methods should not pretend to do so. An extended discussion about this topic is presented by R. McElreath, as an introduction to Bayesian statistics. In general the discussion and the flow chart presented below is presented to underline the concept that statistical models are indeed just models. They can be very powerful tools able to unveil important information buy they can also lead to erroneous results. For these reason, in the process of scientific

inquire is always of utmost importance to have a clear understanding of the model we are using and its inherent assumptions.

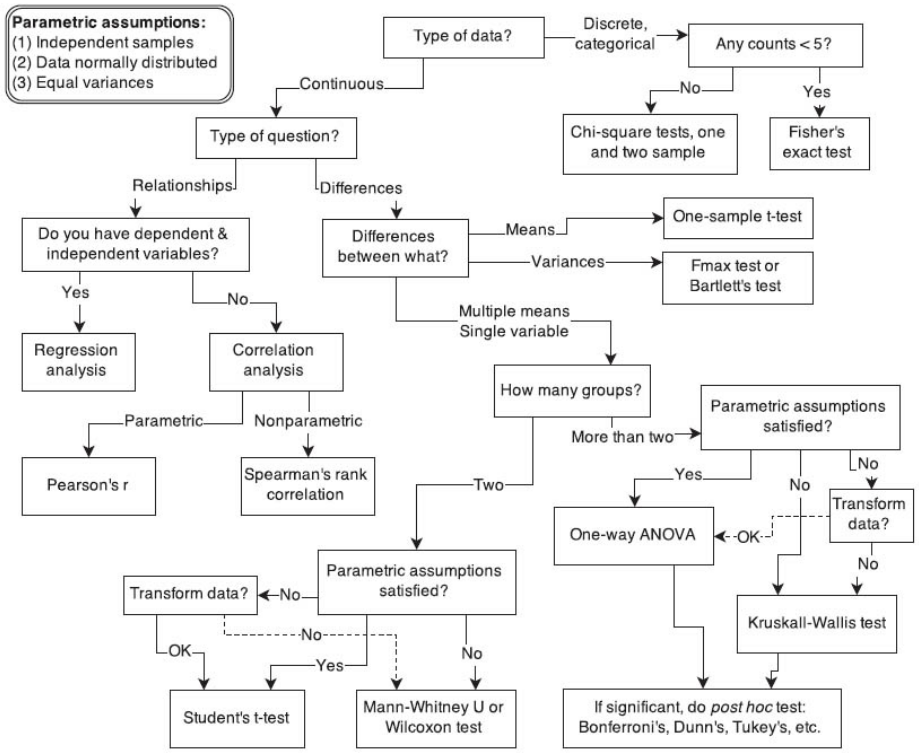


Fig. 10.1 Example decision tree, or flowchart, for selecting an appropriate statistical procedure. From Statistical Rethinking by R. McElreath (2015).

Following the scheme in Figure 10 the first step is to clearly define the nature of our data. In many cases the data are quantitative (expressed by numbers) such as air temperature or cumulative daily precipitation, but in other cases they can be qualitative therefore identified by name or rank and grouped in number of observations in each category therefore categorical. For instance a statistics can be performed on the number of objects produced by a company classified as optimal, acceptable or reject. Otherwise a transportation study could be classifying the type of transportations used by citizens such as train, car or bus. A commercial study could be classifying the quality of a wine based on a test which classify the wine into four categories. When the data are categorical the two common approach are the Chi−square test or the Fisher's exact test, depending on the number of counts.

For quantitative data the first step is to formulate the correct question. Are we looking at relationship, for instance how air temperature depends on elevation or at differences among different measurements. In the case of relationships it is important to identify the dependent and independent variable. If we are looking at the dependence of air temperature with elevation, clearly the independent variable is elevation and the dependent variable is air temperature. After having determined the variables, regression analysis can be performed with linear or non−linear models. If we cannot clearly identify dependent and independent variables, it is possible to follow a correlation analysis, that can be parametric or non−parametric.

If the experiment is focusing in differences, the type of analysis may follow a different direction. For instance, we are looking at the differences in protein content of different plants cultivated under different amount of nitrogen fertilization. We prepare an experiment with different plots and we apply different dose of nitrogen. In this case we are looking at differences. The differences can be first assessed by quantifying differences of means or differences of variances. The simplest test for differences among means is the *t−test*

10.1 Quantitative data

10.2 Population and sample

x is a sample of numerosity n: $x = (x_1, x_2, \ldots, x_n)$. *Sample* is defined as a subset of the investigated population, where the population can be finite or infinite. *Population* is the set of the statistical units of interest. It can also be defined as the set of all measurements of interest to the sample collector. It could be the number of freshmen students enrolling at the department of physics, the number of companies in a region, the total residents of a nation, the concentration of nitrogen in a soil over a given region, the results of an experiment and so forth. In some cases the population can be known, for instance the number of students enrolled in the department of physics at the University of Bologna. In this case, if we want to know the average age of the students, it is possible to compute it. The 7 mean is a number, just one number. We do not need to use inferential statistics to know this information, it is sufficient to use descriptive statistics. On the other hand, in many cases the population cannot be measured. The reasons can be many. The number of elements are too large, such as the size of the sand particles in the soils of a given region, the concentration of a pollutant in the atmosphere, the number of leaves in a forest. In some cases, the experiment can be performed only on a limited number of sample for financial, practical and time constrain. For instance during a political pool it is not possible to interview all the possible voters in an election, therefore only samples are used. In all these cases, the statistical analysis becomes inferential.

To infer means to deduce or conclude (something) from evidence and reasoning rather than from explicit statements. Statistical inference is the process of drawing conclusions about populations or scientific evidence from data. It is the process of using data analysis to infer properties of an underlying distribution of probability. Commonly, the inference is performed about a population while having data only on samples.

It is possible to look at the numbers again, $(x_1, x_2, \ldots, x_n)$ as realizations of the discrete random variables(*drv*) $(X_1, X_2, \ldots, X_n)$, in the following way.

A sample of numerosity equal to 10, is extracted from the finite population made by all the final graduation votes (in 110/110) of the graduated students in Statistical Science in the year 2007-2008. For instance the series below:

$$109, 110, 100, 107, 99, 109, 110, 98, 100, 105$$

Another sample, still of numerosity equal to 10, could be:

$$100, 108, 96, 110, 98, 104, 110, 99, 100, 102$$

and so forth. In the example the *drv* X_1 took the value of 109 in the first sample, 110 in the second and so forth. The (*drv*) X_2 took the value of 100 in the first sample, 108 in the second sample, etc.. The same is repeated for the other values that the *drv* can take (X_i), in the range 66 to 110.

Given the numerosity of the sample, n, the vector of the *drv*) is:

$$\mathbf{X} = (X_1, X_2, \ldots, X_n) \tag{10.1}$$

which is a multi−variate random variable. All the possible realizations of the set of *drv* $(X_1, X_2, \ldots, X_n)$ creates the space of the samples (S). S is the set of all the samples that can be extracted from the population. It is possible to think at S as a n-dimensional space where the samples are point in space.

Let us suppose that the *drv* $(X_1, X_2, \ldots, X_n)$ are independent random variables and identically distributed (IID)(it may not be always the case), then the sample is said to be *random*. The definition said that the *drv* $(X_1, X_2, \ldots, X_n)$ have the same distribution, but did not specified which distribution. Indeed, often the distribution is not known, but it is possible to find a solution with the procedures of *statistical inference*.

The hypothesis that the *drv* $(X_1, X_2, \ldots, X_n)$ are (IID) means that the observed data or the results of the measurements are not correlated. Obviously this is not always true and often it is not possible to determine this *a priori*. Therefore the assumption is that the correlation are *weak enough*, such that are not going to significantly affect the results of the statistical analysis. It is assumed, otherwise, that the sample is *representative*, meaning that it will 'represent' the entire population, or somehow a faithful images (hologram).

10.3 Deriving the mean and variance of different random variables

Let's suppose we have two independent random variables X and Y. As described above the expected values for these two random variables are:

$$E[X] = \mu_X \tag{10.2}$$

and

$$E[Y] = \mu_Y \tag{10.3}$$

The expected value of a random variable X, denoted $E(X)$ is a generalization of the weighted average, and is intuitively the arithmetic mean of a large number of independent realizations of X. The variance is:

$$\mathrm{Var}(X) = E[(X - \mu_X)^2] = \sigma_X^2 \tag{10.4}$$

and

$$\mathrm{Var}(Y) = E[(Y - \mu_Y)^2] = \sigma_Y^2 \tag{10.5}$$

We introduce a third random variable Z defined as $Z = X + Y$. The expected value is:

$$E[Z] = E[X + Y] \tag{10.6}$$

which means that the mean of X plus the mean of Y:

$$\mu(Z) = \mu_X + \mu_Y \tag{10.7}$$

and if we have another r.v. A such that:

$$E[A] = E[X - Y] \tag{10.8}$$

which means that the mean of X minus the mean of Y:

$$\mu(A) = \mu_X - \mu_Y \tag{10.9}$$

So what is the variance of random variable Z and A.

$$\mathrm{Var}(Z) = \mathrm{Var}(X) + \mathrm{Var}(Y) \tag{10.10}$$

which is

$$\sigma_Z^2 = \sigma_{X+Y}^2 = \sigma_X^2 + \sigma_Y^2 \tag{10.11}$$

The variance of the r.v. A is exactly the same thing:

$$\mathrm{Var}(A) = \mathrm{Var}(X) + \mathrm{Var}(Y) \tag{10.12}$$

$$\sigma_A^2 = \sigma_{X+Y}^2 = \sigma_X^2 + \sigma_Y^2 \tag{10.13}$$

why it is not minus as for the mean ?

$$\sigma_A^2 = \sigma_{X-Y}^2 = \sigma_{X+(-Y)}^2 = \sigma_X^2 + \sigma_{-Y}^2 \tag{10.14}$$

now we will see that the variance of negative $-Y$ is:

$$\sigma_{-Y}^2 = \mathrm{Var}(-Y) = E\left[(-Y - E(-Y))^2\right] = E\left[(-1)^2(Y + E(-Y))^2\right] \tag{10.15}$$

since $E(-Y) = -E(Y)$, therefore

$$\sigma_Y^2 = E\left[(Y - E(Y))^2\right] \tag{10.16}$$

Therefore the variance of the difference of two independent random variable is equal to the sum of the variances.

$$\sigma_A^2 = \sigma_{X-Y}^2 = \sigma_X^2 + \sigma_Y^2 \tag{10.17}$$

Indeed, it does not matter if we take the negative or the positive of the variable, since we are measuring the absolute distance, so it makes sense. So the quantity σ_Y^2 and σ_{-Y}^2 are the same thing. The important aspect of this derivation is that the mean of differences is the same as the differences of their means:

$$\mu_{X-Y} = \mu_X - \mu_Y \tag{10.18}$$

and the variance of differences is the same as the sum of their variances:

$$\sigma_{X-Y} = \sigma_X + \sigma_Y \tag{10.19}$$

Now, let us consider two random variable X and Y:

If we consider the difference between there two random variables:

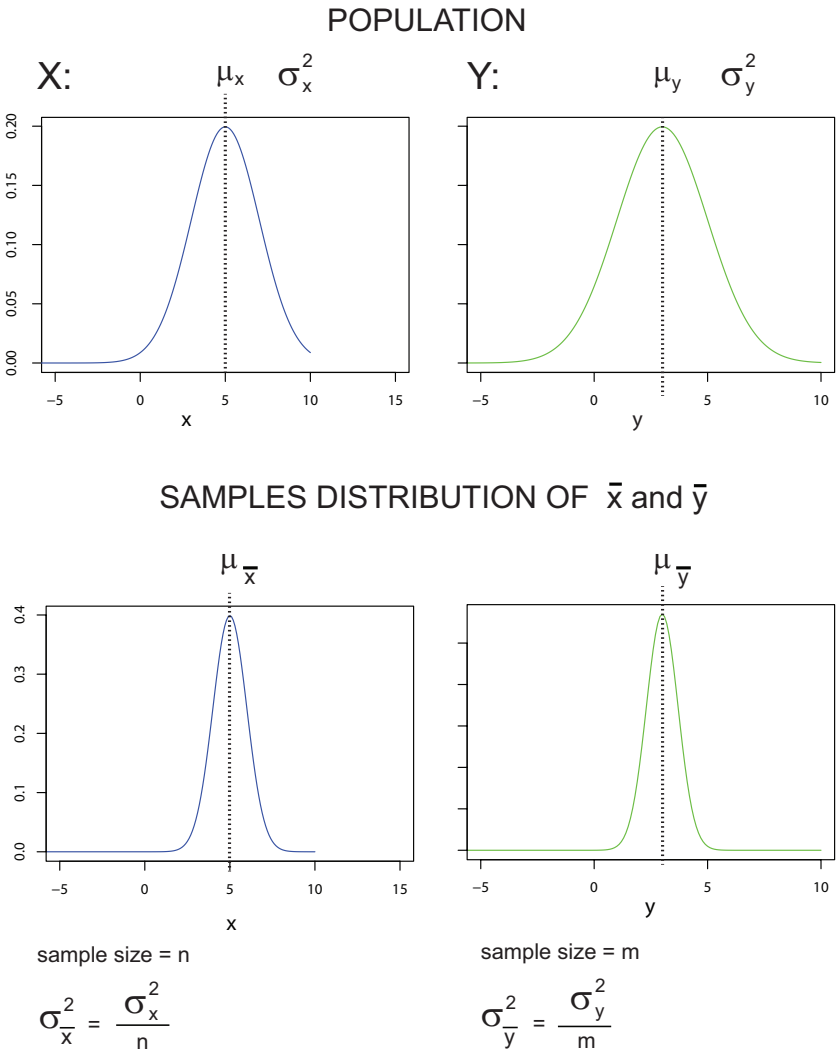


Fig. 10.2 Sampling process from a population

$$Z = \bar{X} - \bar{Y} \tag{10.20}$$

where $\bar{X}$ and $\bar{Y}$ are now the random variables of the means, we will have

$$\sigma^2_{\bar{X}-\bar{Y}} = \sigma^2_{\bar{X}} + \sigma^2_{\bar{Y}} = \frac{\sigma^2_X}{n} + \frac{\sigma^2_Y}{m} \tag{10.21}$$

a new distribution with larger variance will be obtained since the variances of the two distribution are obtained:
Therefore the standard deviation is:

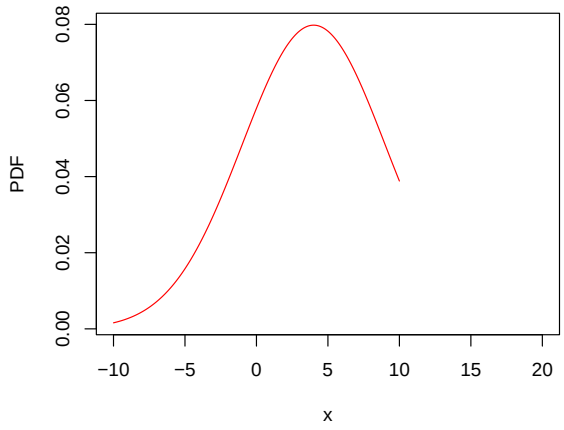


Fig. 10.3 Difference between sampling populations

$$\sigma_{\bar{X}-\bar{Y}} = \sqrt{\frac{\sigma_X^2}{n} + \frac{\sigma_Y^2}{m}} \tag{10.22}$$

it is possible to notice that this equation looks like a distance formula, which will help to visualize these concepts in a geometrical way. Moreover, these concepts will be the basis for starting to compare different populations means to identify stastically significant differences among populations.

10.3.1 Example

In this example, the precipitation at Nevada City are used. The population are the precipitation at Nevada City from 1873 to 1978. mean: μ = 1337.57 standard deviation:

σ = 391.83

It is better to write as σ = 390 e μ = 1300 e σ = 390 since in the error it is enough one decimal points. These parameters are referred to the population and they are called *parameters*. The computation of these parameters is shown below, where the data are imported, the values are converted into [mm], and then mean, variance and standard deviation are computed.

```
#CODE Ch10_1.R
setwd("~/Didattica/R_class_4/exercises/Ch9_statistical_inference")
Nevada_prec <- read.table("Nevada_prec.dat"
, sep = "", check.names = FALSE, header = T, na.strings = c("NA", "NAN"))
#Convert precipitation from inches to mm
Nevada_prec$Prec_mm = Nevada_prec$Prec * 25.4
Nevada_prec$Prec_mm < numeric()
mu<- mean(Nevada_prec$Prec_mm) #mean
```

```
V<- var(Nevada_prec$Prec_mm) #variance
S<- sd(Nevada_prec$Prec_mm) #standard deviation
```

The output is

```
> mu
[1] 1337.574
> V
[1] 153534.4
> S
[1] 391.8347
>
```

The parameters described above (mean, variance and standard deviation) are known fixed numbers representing a **population**, since the entire data set for that period is known. In many cases the population is unknown and, as described above, the purpose of **inferential statistics** is to derive the unknown parameters from a **sample**. For instance a sample is:

$$n = 20: \begin{array}{l} 1422.4, 1752.6, 762.0, 1854.2, 1143.0, 1371.6, 1041.4, 1320.8, 812.8, 812.8, \\ 914.4, 889.0, 1600.2, 1803.4, 1778.0, 1016.0, 2108.2, 1092.2, 1270.0, 762.0 \end{array}$$

In this case 1422.4 is the realization of the discrete random variable $X_1, \ldots,$ 762.0 is the realization of the discrete random variable X_{20}, where they are all IID. Now the mean of the population (μ) is estimated through the sample mean, note that as we described before, the symbol for the mean is now $\bar{y}$ where the mean for this sample is $\bar{y} = (1422.4 + 1752.6 + \cdots + 762.0)/20 = 1276.4$. The standard deviation (s) can also be estimated:

$$s = \left[\sum_{i=1}^{20} (x_i - \bar{x})^2 / (n-1) \right]^{1/2} = 418.9 \tag{10.23}$$

Now $\bar{y}$ and s are computed from data from the sample, therefore they are *estimated* values, not true values.

As discussed above, the simplest statistical inference problem is **point estimation**, where a single value (for instance the mean) from the sample data is used to *estimate* a population parameter. In general, define θ as a parameter of the population described by random variables X. X is unknown (the distribution is unknown), and θ is unknown. The quantity θ is called *parameter* and it is a constant of the population. θ is a measure of the distribution of one or more character of the population, for instance the mean μ or the variance σ^2.

The **estimation** of the parameter θ is function of the observations of the sample $\hat{\theta} = t(x_1, \ldots, x_n)$. *When another sample is used, another value is obtained* $\hat{\theta}$. Therefore $\hat{\theta}$ is a realization of the *drv* $\Theta = t(X_1, \ldots, X_n)$.

The discrete random variable Θ is called **estimator**. Since Θ is a random variable, it has a distribution. This is called the principle of *plug-in* (substitution). In other words:

the mean of the population is estimated from the sample mean, the population variance is obtained from the sample variance and the population standard deviation is obtained from the sample standard deviation. Note that not always the plug−in principle is a good choice.
An example is given where a sample with $n = 20$ is considered:

$$914.4, 1371.6, 762.0, 1117.6, 1092.2, 1905.0, 1600.2, 1320.8, 1016.0, 1244.6,$$
$$1701.8, 1270.0, 1524.0, 889.0, 1625.6, 1117.6, 1066.8, 1143.0, 1752.6, 1473.2$$

The mean is $\bar{y} = 1295.4$ and the standard deviation is $s = 315.7$. The values of $\bar{y}$ are realizations of the estimator *sample mean*:

$$\overline{X} = \sum_{i=1}^{n} X_i / n \tag{10.24}$$

Now $\Theta = \overline{X}$. In the following figures different distributions made by sample averages of 5, 20 and 50 (still with $n = 20$) are depicted. Because of the *central limit theorem*, the realizations of $\overline{X}$, by increasing the number, they tend to assume the well−known bell shape, typical of the normal distribution.

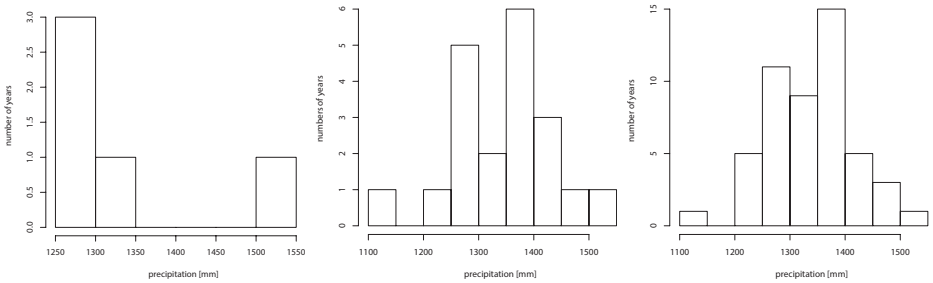


Fig. 10.4

Figure 10.3.1 shows the normal distribution, superimposed over the distribution of the sample means for the precipitation example. The histogram is normalized (total area = 1) for comparison against the normal distribution. The total number of sample means is 100, but the sample size is 25 on the left and 75 on the right.
The numerical results are Left: mean = 1347.4, standard deviation = 63.6 ; Right: mean = 1335.7, standard deviation = 26.9.

10.4 Confidence Intervals

In the section above the concept of **point estimator** was provided. However a point estimation is not a reliable assessment of the population without a measure of **uncertainty** about the estimation. How close is the estimated mean close to the population mean.

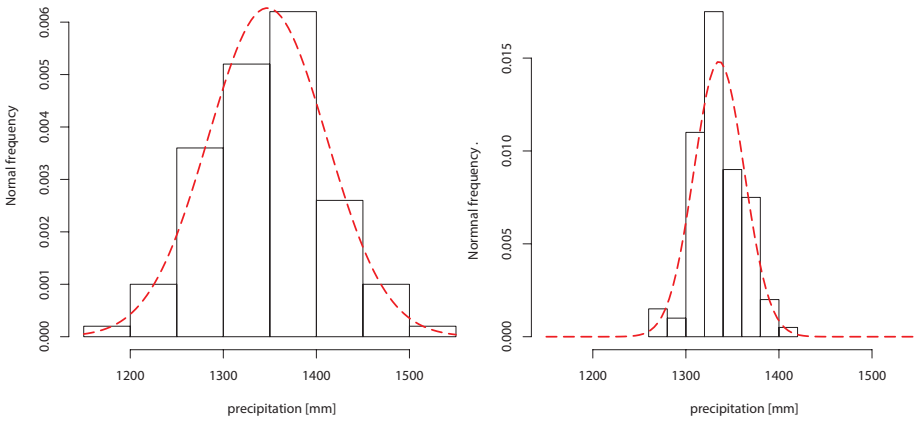


Fig. 10.5

There are several ways to quantify uncertainty. A common method is the concept of **confidence interval**. The confidence interval can be conceptualized as:

$$\bar{x} \pm \text{a Margin of Error} \tag{10.25}$$

where $\bar{x}$ is the sample mean, which is used as **estimator** for the population mean μ. We start from a Normal distribution with mean (μ) and variance (σ^2/n):

$$\mathcal{N}(\mu, \frac{\sigma^2}{n}) \tag{10.26}$$

To properly quantify the error a few assumptions are made: (a) the sample is randomly selected from the population, (b) the population is normally distributed, standardizing $\bar{X}$ by subtracting the mean and divide by the standard deviation:

$$\mathcal{Z} = \frac{(\bar{X} - \mu)}{\sigma/\sqrt{n}} \sim \mathcal{N}(0, 1) \tag{10.27}$$

where $\mathcal{Z}$ is a normalized normal distribution with mean $= 0$ and variance$= 1$. $\bar{X}$ is a random variable with a normal distribution with mean μ and variance σ. $\bar{X}$ represents the many possible means that are obtained by drawing different samples. The third assumption (c) is that the population standard deviation σ is known. In practice this is commonly rare since in many cases the population is unknown and therefore its standard deviation as well, therefore the population standard deviation will be estimated from samples as well. For now, the assumption is that the population σ is known.

The general equation for the $(1 - \alpha)$ confidence interval for μ is:

$$\bar{X} \pm \frac{z_\alpha}{2} \frac{\sigma}{\sqrt{n}} \tag{10.28}$$

where the term on the right end side is the margin of error. $(1 - \alpha)$ is the confidence level and it can be 95% or 99%. The 95% is the most common choice. The confidence

interval is computed by choosing the correct z_α value. Figure 10.4 shows the normalized normal distribution for a confidence interval of 95%. The graph shows the indicated values of $z_\alpha/2$ (± 1.96 in this case), the area $(1-\alpha)$ and the two tails which are $\alpha/2$. Each tail is obtained by splitting α evenly in the right and left tails. For any confidence level, the appropriate $z_\alpha/2$ value must be computed. In R it is possible to compute the value by typing the `qnorm` which returns the quantile given a fraction value.

```
qnorm(0.975)
[1] 1.959964
```

Since 1-0.025 = 0.975.

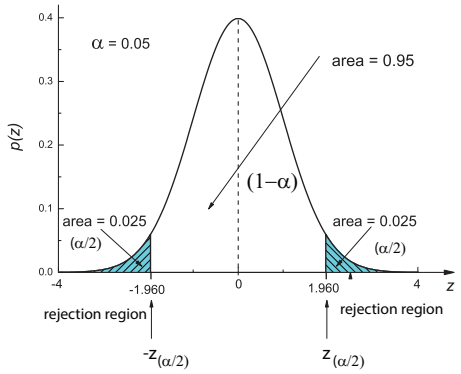


Fig. 10.6 Confidence intervals for α = 0.05.

The standard confidence interval α equal to 0.05 (95% confidence interval, $1-\alpha$ = 0.95) is:

$$\mathrm{P}\left(-1.960 \leq \frac{(\bar{X}-\mu)}{\sigma/\sqrt{n}} \leq 1.960\right) = 0.95 \tag{10.29}$$

multiplying by $(\sigma/\sqrt{n})$ and subtracting $\bar{X}$ from each term:

$$\mathrm{P}\left(\bar{X} - 1.960\,\frac{\sigma}{\sqrt{n}} \leq \mu \leq \bar{X} + 1.960\,\frac{\sigma}{\sqrt{n}}\right) = 0.95 \tag{10.30}$$

where $\bar{X}$ is a random variable and also $\bar{X}_- = \bar{X} - 1.960\,\sigma/\sqrt{n}$ and $\bar{X}_+ = \bar{X} + 1.960\,\sigma/\sqrt{n}$ are random variables.

This equation describes that 95% of the values determined by the realizations $\bar{X}_-$ and $\bar{X}_+$, cover the value of μ. If a single sample is observed with mean $\bar{x}$, which is a realization of $\bar{X}$ it can be said that (with 95% confidence), that:

$$\bar{x} - 1.960\,\frac{\sigma}{\sqrt{n}} \leq \mu \leq \bar{x} + 1.960\,\frac{\sigma}{\sqrt{n}} \tag{10.31}$$

the interval $[\bar{x} - 1.960\,\sigma/\sqrt{n}, \bar{x} - 1.960\,\sigma/\sqrt{n}]$ is the confidence interval at 95%.

Obviously, the **population mean** μ is a fixed number, although it may be unknown, *it is not a random variable*. Therefore it is wrong to say that the mean of the population

has a probability of 95% to be within the interval $[\bar{x} - 1.960\,\sigma/\sqrt{n}, \bar{x} - 1.960\,\sigma/\sqrt{n}]$. The variable that changes is $\bar{x}$, that varies from sample to sample. It is better to say that the *confidence interval* covers the true value of the mean. Indeed, it is better to use the word *coverage interval*, meaning that the interval described above has a probability 95% to *cover* the unknown value of μ. Therefore it exists a probability equal to α that the sample comes from a population where the `mean` is outside of the interval.

Example: Let us assume to have a sample $(9.0, 7.0, 14.0, 13.0, 5.0, 10.4, 6.6, 8.5, 7.5)$ collected from a population with normal distribution and known $\sigma = 1$. The sample mean is $\bar{x} = 81/9 = 9$. Substituting the values of sample mean $(\bar{x})$ and variance (σ) into eq. 10.31, into the confidence interval at 95%, leads to:

$$9 - 1.960\,\frac{1}{\sqrt{9}} \leq \mu \leq 9 + 1.960\,\frac{1}{\sqrt{9}} = [8.35, 9.65] \tag{10.32}$$

There is 95% confidence that this interval will cover the *unknown true mean* of the population. In this case, since the true mean (μ) was 9, the outcome was correct.

In the example below, a sequence of numbers for a sample is created. An estimator plug-in is computed.

```
#Code Ch10_2.R
#Confidence Intervals for a random distribution and
#plug-in estimator for a randomly selected sample
# Define sample elements
y <- c(52, 104, 146, 10, 50, 31, 40, 27, 46)

#Confidence intervals (for a normal distribution)

#Variance
sig2p <- function(x) {var(x)}
# Confidence interval 90%
conf.level<-0.90

# Standard Interval
int.stand <- function(x,conf.level) {
theta.hat<-mean(x) # theta.hat = estimated mean
SE <- sqrt(sig2p(x)/length(x)) # plug-in sigma/sqrt(n)
alpha<-1-conf.level
# Density, distribution function,
#quantile function and random generation for the normal
#distribution with mean equal to
#mean and standard deviation equal to sd.
#half on one side and one on the othertheta.hat
#+ c(-z.1malpha*SE,+z.1malpha*SE) #average +- standard error
z.1malpha <- qnorm(1-alpha/2)
       }
```

```
#Sample data
theta.hat<-mean(y)
theta.hat
SE <- sqrt(sig2p(y)/length(y))

int.stand(y,conf.level)

plot(function(x) dnorm(x,theta.hat,SE),0,110)
quant<-quantile(y, probs = c(0.025, 0.1587, 0.5, 0.8413, 0.975))
segments(quant[1],0,quant[1],2,lty=2,col="red",lwd=2)
segments(quant[4],0,quant[4],2,lty=2,col="red",lwd=2)
```

Here the previous example about precipitation in Nevada city is discussed. The standard deviation σ is unknown and the population from which the sample was collected is not a normal distribution. Nevertheless, it is possible to compute an "approximated level of confidence" based on the convergence toward a normal distribution, described by the central limit theorem. Moreover, it is possible to estimate σ by using the sample standard deviation s. Here the sample that was used before is considered again with $n = 20$: $1422.4, 1752.6, \ldots, 762.0$.

The sample mean is $\bar{x} = 1276.4$. So the population mean ($\mu = 1337.57$) was estimated using the sample mean $\bar{x}$, interpreted as a realization of a random variable $\bar{X}$. Now, it is of interest to determine the accuracy of the estimation. The standard deviation of the population ($\sigma = 391.83$), was estimated from the sample standard deviation $s = 418.9$, again a realization of the random variable sample standard deviation.

$$Var\left(\bar{X}\right) = \sigma^2/n \tag{10.33}$$

where σ^2 was replaced by s^2. The confidence interval at 95% is then:

$$\left[1276.4 - 1.960\,\frac{418.9}{\sqrt{20}}, 1276.4 + 1.960\,\frac{418.9}{\sqrt{20}}\right] = [1092.76, 1459.94] \tag{10.34}$$

Note that the size of the interval $2 \times z_{\alpha/2}\left(\sigma/\sqrt{n}\right)$ is independent from the mean, while it is function of the standard deviation σ (or from the estimator s), therefore from the intrinsic variability of the sample, and from the sample size (number n of elements). In the code below the R code is presented.

The function `sample()` is used to perform an operation of re-sampling. The idea is to simulate a collections of samples, without knowing the entire population. Obviously in this case, we know the total number of cumulative precipitation for each year, so the population is known. In this exercise we pretend to create a certain number of collected samples, generated randomly.

The `for` loop is generating a series of random numbers starting always from the same seed, therefore the series is always the same. It is useful to employ this method, such that differences in various `statistical experiments` are not due to the variation in the generated random numbers. An important feature of the function `sample`

Let us imagine to extract numbers from the bingo, where numbers are in the interval (1-90). If samples of 25 elements are collected each time, but then I put the collected numbers back into the box, those numbers could be picked up again. This method is called with re−introduction. If there are many numbers (maybe 10,000), the computed mean will not be very different if the reintroduction method is selected. However, if the number is small (like the example of the bingo), then the reintroduction method can generate biased means. The selection of this method is possible by using the instruction `replace=FALSE`, there is not reintroduction. The overall procedure is a permutation.

The instruction `set.seed(seed)` set the seed of R's random number generator, which is useful for creating simulations or random objects that can be reproduced. For example to create simulated values that are reproducible.

```
> set.seed(4)
> rnorm(4)
[1] 0.2167549 -0.5424926 0.8911446 0.5959806
> set.seed(4)
> rnorm(4)
[1] 0.2167549 -0.5424926 0.8911446 0.5959806
```

The results keep being the same every time, otherwise the rnorm function would return different values each time. The code below employs this function.

```
setwd("~/Didattica/R_class_4/exercises/Ch9_statistical_inference")
Nevada_prec <- read.table("Nevada_prec.dat"
, sep = "", check.names = FALSE, header = T, na.strings = c("NA", "NAN"))
#Convert precipitation from inches to mm
Nevada_prec$Prec_mm = Nevada_prec$Prec * 25.4
Nevada_prec$Prec_mm < numeric()
mu<- mean(Nevada_prec$Prec_mm) #mean
V<- var(Nevada_prec$Prec_mm) #variance
S<- sd(Nevada_prec$Prec_mm) #standard deviation

xx<-numeric()
mu<-numeric()
sigma<- numeric()

ltot<- 100

for(l in 1:ltot){
       #generate the seed to create a random series,
       #that is always the same (with the same seed)
       set.seed(l+400)
       #number of elements for each sample,
       #with permutation because replace = FALSE
       B<- 25
       xx<- sample(Nevada_prec$Prec_mm,B,replace=F)
```

```
        mu[l]<- mean(xx)
        sigma[l]<- sd(xx)
}
mu
sigma

hist(mu, xlab="mm di pioggia",ylab="freq. norm.",main=" ",
prob=T,plot=T,xlim=c(1150,1550))#,ylim=c(0,0.0065))
#,ylim=c(0,0.018))

m.mu<- mean(mu) #means of the sample means
sd.mu<- sd(mu)
m.mu
sd.mu
#density of the Gaussian function
curve(dnorm(x,m.mu,sd.mu),add=T,lty=5,lwd=2,col="red")
#density of the function
```

Figure 10.4 shows the distribution (histogram) of randomly selected samples from the precipitation values at Nevada city, superimposed over a normal distribution.

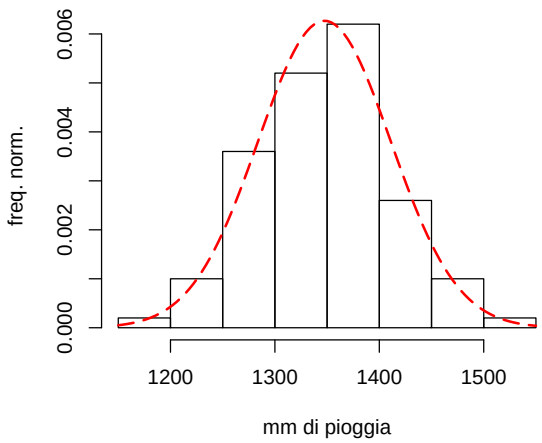


Fig. 10.7 Ramdomly generated samples superimposed over a normal distribution for the Nevada Precipitation values.

10.4.1 Confidence intervals for t-Student distribution

The estimation and test procedures about μ presented earlier in this chapter were based on the assumption that the population variance was known or that we had enough observations (samples) to allow the sample standard deviation s to be a reasonable estimate of the population standard deviation σ. In this section a test is presented to be used when σ is unknown, no matter the sample size. For example, to determine the average concentration of a drug into a patient blood stream one hour after the patient suffering from a rare disease was treated with that drug, may not be possible to obtain a random sample of 30 or more observations at a given time.

This test was derived by W.S. Gosset who faced the problem of estimating the mean quality of beer brews, but based on small samples. He thought that using a normal distribution with small σ would lead to falsely reject the null hypothesis at a slightly higher rate than that specified by α. He derived the distribution and percentage points of the test statistic for normal distribution for $n < 30$. He published the results under the pen name Student, because against the company policy to publish his results.

Therefore if σ is unknown, it is not possible to write:

$$\mathcal{Z} = (\bar{X} - \mu)/(\sigma/\sqrt{n}) \sim \mathcal{N}(0,1) \tag{10.35}$$

but it is possible to write:

$$\mathcal{Z} = \frac{\bar{X} - \mu}{S/\sqrt{n}} \sim t_{n-1} \tag{10.36}$$

It means that the statistics of the first member of the equation is distributed like a random variable for a t Student with $n-1$ degrees of freedom. In the equation above, the rv are $\bar{[X]}$ and S.

With the t Student (19 degrees of freedom), the quantile corresponds to $1-\alpha = 0.95$, which for the normal distribution is $z_{\alpha/2} = 1.960$, now it is $t_{\alpha/2}^{[n-1]} = 2.093$ $(n-1 = 19)$, a little larger than 1.960.

The confidence interval is then written as:

$$\left[\bar{x} - t_{\alpha/2}^{[n-1]} \frac{s}{\sqrt{n}}, \bar{x} + t_{\alpha/2}^{[n-1]} \frac{s}{\sqrt{n}}\right] \tag{10.37}$$

In this example is $[1080.3, 1472.4]$, larger than the one before.

If $n \gtrsim 20 - 30$, then $t_{\alpha/2}^{[n-1]} \approx z_{\alpha/2}$. It means that the t Student tends to a normal distribution, with increasing the number of elements in the sample.

It is possible to compute the confidence intervals for the t-student distribution by using

```
qt(0.975,10)
[1] 2.228139...
```

To obtain a value of 1.96 (corresponding to the normal distribution) we must have above 1000 degree of freedom.

```
> qt(0.975,1000)
[1] 1.962339
```

10.4.2 Terminology

The standard deviation of an estimator is often called *standard error*, and it is indicated with $se(\cdot)$. For example, for a *sample mean* of a *rv*, $\bar{X}$, the standard error of the mean $\text{se}(\bar{X})$ is

$$\text{se}(\bar{X}) = \sqrt{\text{Var}\left[\ \bar{X}\ \right]} = \sigma/\sqrt{n} \tag{10.38}$$

If σ is unknown, standard error is estimated with $s/\sqrt{n}$. It meansthat the standard error is a way to call the ratio:

$$\frac{\text{sample standard deviation}}{\text{sample size}} \tag{10.39}$$

In the example about precipitation $\bar{x} = 1276.4$, the standard error is written $se(\bar{x}) = 418.9/\sqrt{20} = 93.67$. A common way to refer to the results is to write Un modo frequente per riferire il risultato della stima intervallare è scrivere $\bar{x} \pm \text{se}(\bar{x})$. In the example:

$$\bar{x} = 1276 \pm 94 \tag{10.40}$$

The standard error is written with two significant digits $\text{se}(\bar{x})$. With the previous terminology the confidence interval is $[1182, 1370]$ with an approximation level of $\approx 68\%$.

10.5 Hyphotesis tests

The sample **y** of numerosity $n_y = 9$, is analyzed with respect to another sample **z** of numerosity $n_z = 7$. The two samples are:

$$\begin{array}{ll} \mathbf{ze} = (94, 197, 16, 38, 99, 141, 23) & n_z = 7 \\ \mathbf{y} = (52, 104, 146, 10, 50, 31, 40, 27, 46) & n_y = 9 \end{array} \tag{10.41}$$

The hypothesis test is formulated when the question is: *are the two sample different?* or better *Are the elements of the sample A larger than the elements of the sample B ?*

Within an experiment framework, performed with different methods, then the questions formulated above are equivalent to ask: *is method x better than method y ?*

The data described above are collected from an experiment by Efron et al., where 16 rats were subjected to a treatment with a new drug. Seven rats were the treatment (they received the drug) while nine rats were the control group (they did not receive the drug). To explore if the group **ze** are larger than the group **y** it is possible to compare the sample means.

$$\overline{ze} = 86.857 \qquad \text{and} \qquad \overline{y} = 56.222$$

from which $\overline{ze} - \overline{y} = 30.635$, where d^{obs}, is the observed difference. The d^{obs} is an estimate of what is usually called $\hat{\theta}$.

Now it was observed that the sample means were different. However is this difference a real difference between the samples, or the difference is due to a random or a systematic error ? In other words: does the drug work or not ? The problem is well known and it brings to the concept of hypothesis test. The **first step** is to formulate a null hypothesis H_0. In our case we assume that the sample ze was extracted from a population described by the random variable Z, and that the sample y was extracted from a population described by the random variable Y. The enunciation of the null hypothesis is:

$$H_0 : Z = Y$$

which means that the two samples are collected from two populations with features that have the same distribution. In other words there are no differences. H_0 states that the probabilistic behavior of the sample ze is the same of the sample y, no matter which sample is collected from the the two populations.

The **second step** is to build a statistical test. For instance the difference between the sample means:

$$D = \overline{Z} - \overline{Y}$$

Note that D is a random variable, which is the difference between the two random variables $\overline{Z}$ e $\overline{Y}$.

In our case, more realizations of D are observed if H_0 is not true, than if H_0 is true. In our case, the more realizations of D occurs, more we assume that H_0 is not true. Obviously to quantify the realizations of D, the probability density of D should be known or at least we can formulate a conjecture about it. If the distribution of D is known, then it is possible to compute the probability that the realizations of D are larger than the observed values of d^{obs}.

Then, what is the probability that a realization of D is larger than d^{obs}?

To define the *probability that the rejection of the null hypothesis is only given by change we use the* "p-value".

Another example is to test if the income of a family from Bologna is significantly different from 2000 euros a month. So the null hypothesis is

$$H_0 : \mu = 2000$$

while

$$H_1 : \mu \neq 2000$$

10.6 Example

Here we are presenting an example where the concept presented in the sections above is applied.

We are trying to test if two low sugar diet will help overweight people to loose weight. One group of 100 people are assigned to a low sugar diet, while another group of 100 people ($n = m$) are kept on the same diet with lower calories but with the same

amount of sugar, usually the latter is called the *control* group. After 3 months the first group lost (as average) 4.2 kg, while the second group lost 3.3 kg. At a first look it seems that the first diet was indeed effective in reducing weight. So the mean weight loss $\bar{x}$ and standard deviation $\bar{s}$ were:

$$\text{Low sugar} = \bar{x} = 4.2\,,\ \bar{s_x} = 2.11[\text{kg}]$$

and for the control group:

$$\text{Control} = \bar{y} = 3.3\,,\ \bar{s_y} = 1.83[\text{kg}]$$

at a first superficial look it looks like the low sugar group lost more weight that the control. If the difference is computed:

$$\bar{x} - \bar{y} = 4.2 - 3.3 = 0.9\,[\text{kg}]$$

Now the question is to get a 95 % confidence interval around this number. So, as explained above, we want to look at the distribution (assuming that it is normal) of the difference of the means.

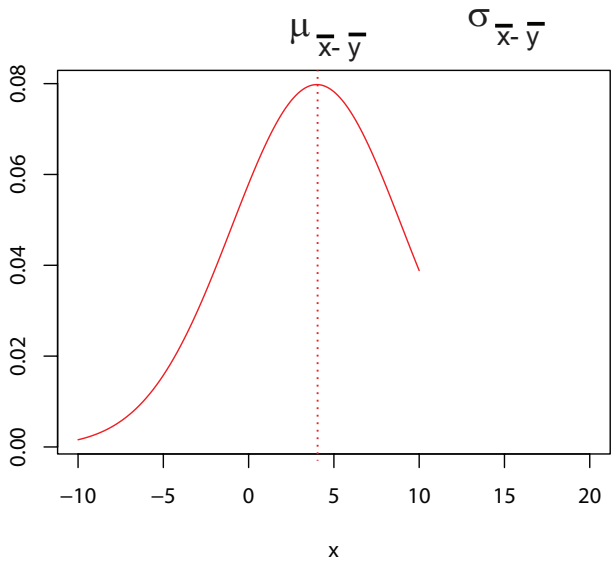


Fig. 10.8 Distribution of the differences of the means.

This is going to have a mean and a standard deviation as shown in Figure 10.6. We want to make some inferences about this distribution, based on our samples. We want to define an interval where we know that the true mean will be *covered by this interval*. How many standard deviations we need to go to cover this interval ? This is done with the so called Z table that provides the values for the just one tail. For instance the 0.975 gives a value of z = 1.96, as described above. Or, only 2.5 % of the samples are going to be more that 1.96 standard deviations away from the mean.

It can also be written as there are 95 % changes that the value $\mu_{\bar{X}-\bar{Y}} = 1.91$ will be covered within the distance:

$$\sigma_{\bar{X}-\bar{Y}} \times 1.96 \tag{10.42}$$

So the question is how to calculate $\sigma_{\bar{X}-\bar{Y}} \times 1.96$, therefore to compute the standard deviation of the distribution. So the standard deviation will be:

$$\sigma_{\bar{X}-\bar{Y}} = \sqrt{\frac{\sigma_X^2}{n} + \frac{\sigma_Y^2}{m}} \tag{10.43}$$

so by replacing the values the computation will be:

$$\sigma_{\bar{X}-\bar{Y}} = \sqrt{\frac{\sigma_{\bar{X}}^2}{100} + \frac{\sigma_{\bar{Y}}^2}{100}} \tag{10.44}$$

since σ_X and σ_Y are unknown, we can approximate it with the sample standard deviations:

$$s_{\bar{X}-\bar{Y}} = \sqrt{\frac{s_{\bar{X}}^2}{100} + \frac{s_{\bar{Y}}^2}{100}} \tag{10.45}$$

leading to:

$$s_{\bar{X}-\bar{Y}} = \sqrt{\frac{2.11^2}{100} + \frac{1.83^2}{100}} = 0.27 \tag{10.46}$$

now the interval can be computed:

$$0.27 \times 1.96 = 0.52 \tag{10.47}$$

so the confidence interval will be the difference of the means $\pm$ the value computed above:
now the interval can be computed:

$$0.9 \pm 0.52 = 0.52 \tag{10.48}$$

now the interval can be computed:

$$0.38 \leq \text{CI} \leq 1.42 \tag{10.49}$$

the expected value of the sample means and the expected values of the population is the same, therefore this interval gives us an interval that will cover the expected value of the population.