Exploratory Data Analysis

Giovanni Saraceno

Contents

Differently from **inferential statistics** (it will be addressed in the following sections) that deals with uncertainties in estimates and inferences about one or more populations, **Exploratory Data Analysis** aims to reveal interesting patterns and help to prepare the data in the best way for the following analyses.
It can be roughly summarized in three big parts:

1. *Structure and summary of data*: To check the type of variables in the data set and compute location indexes (e.g., mean, median), variability indexes (e.g., variance) of the variables of interest
2. *Exploratory plots*: Histograms, box plots, bar plots, correlogram or scatter plots (e.g., skeweness, outliers)
3. *Preprocessing step*: Are there any NAs? Missing values? Integer variables to be converted in factors?

We will show the code using classical functions in R base and more recent `tidyverse` and `ggplot2`.

```
library(tidyverse)
```

```
## Warning: il pacchetto 'tidyverse' è stato creato con R versione 4.3.2
```

```
## Warning: il pacchetto 'ggplot2' è stato creato con R versione 4.3.3
```

```
## Warning: il pacchetto 'stringr' è stato creato con R versione 4.3.3
```

```
## Warning: il pacchetto 'lubridate' è stato creato con R versione 4.3.2
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(dplyr)
```

We consider the data set `peso.Rdata` defined in the Introduction notes.

```
dati <- read.table("../Introduction/peso.txt", sep=";")
```

```
str(dati)
```

```
## 'data.frame':    50 obs. of  4 variables:
##  $ X: chr  "C" "V" "S" "C" ...
```

```
##  $ Y: chr  "S" "L" "A" "L" ...
##  $ Z: int  3 3 2 3 1 0 0 3 0 0 ...
##  $ W: num  67.4 92 69.6 62.2 62.8 ...
```

```
attach(dati)
```

Generally, a data set contains information about a random sample. For each observation (rows), characteristics, also called variables (data set columns), are recorded. This means that the data represent realizations of one or more measurements performed on a sample of individuals.

Variables can be classified as:

Qualitative (categorical): These express characteristics that cannot be measured numerically but allow assigning a sample observation to a category or group. For example, the variables X and Y are categorical. If there is no intrinsic order, the variables are called nominal, whereas if the values can be ordered, they are ordinal. *Quantitative (numerical)*: The measurements are expressed as numbers. Numerical data can be continuous, like the variable W representing weight, or discrete, like the variable Z, representing the number of dependents.

Discrete Variables

Discrete variables can take on a finite number of values. Both qualitative and discrete numerical variables fall into this category. In these cases, it is useful to examine the frequency distribution of all its values. In R, the `table()` function is used:

```
table(X)
```

```
## X
##  C  N  S  V
## 10 12 16 12
```

```
table(Y)
```

```
## Y
##  A  L  O  S
## 13 12  9 16
```

```
table(Z)
```

```
## Z
##  0  1  2  3  4
## 14  7  8 11 10
```

The same result can be obtained with the `summary()` function:

```
summary(X)
```

```
##    Length     Class      Mode
##        50 character character
```

Alternatively, we can calculate relative frequencies by dividing by the total number of observations:

```
table(X) / length(X)
```

```
## X
##    C    N    S    V
## 0.20 0.24 0.32 0.24
```

```
table(Y) / length(Y)
```

```
## Y
##    A    L    O    S
```

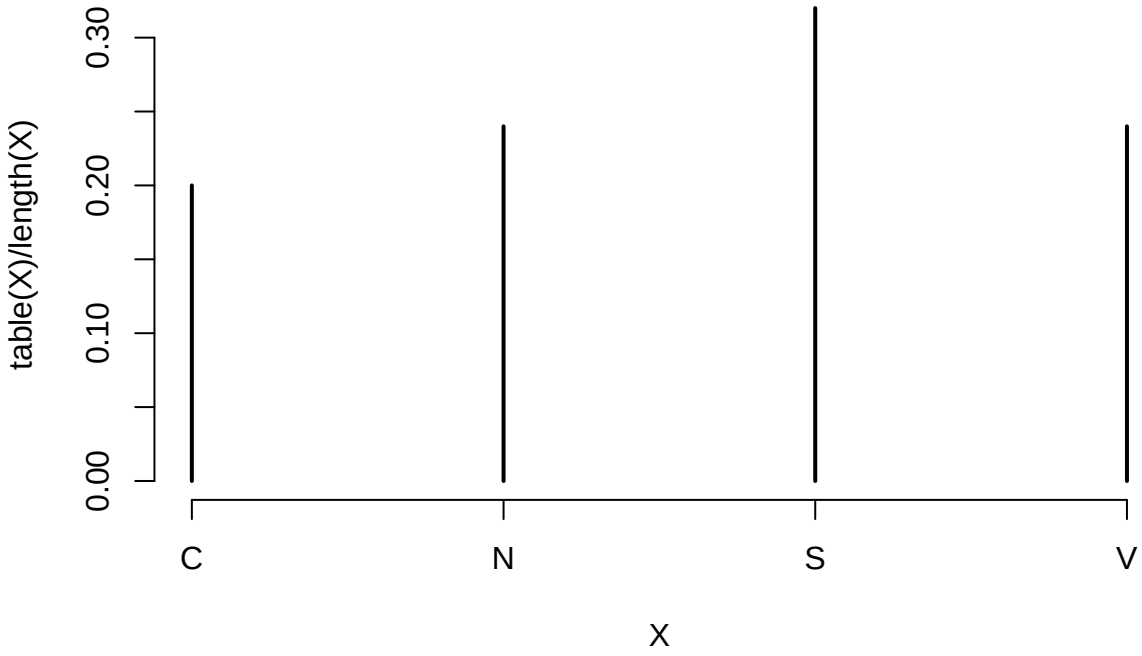
```
## 0.26 0.24 0.18 0.32
```

```
table(Z) / length(Z)
```

```
## Z
##    0    1    2    3    4
## 0.28 0.14 0.16 0.22 0.20
```

The frequency table can be represented using a bar plot:

```
plot(table(X) / length(X))
```

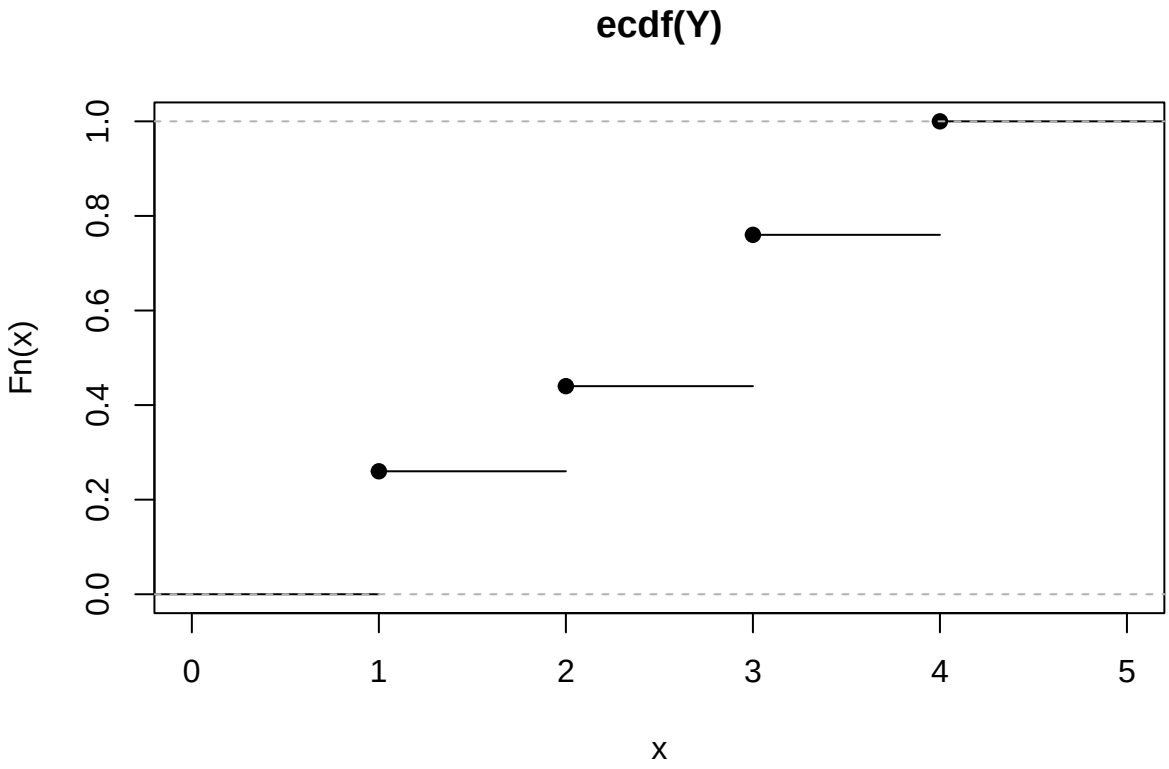


If there is an intrinsic order, cumulative frequencies can be calculated and visualized:

```
Y <- ordered(Y, levels = c("A", "O", "S", "L"))
cumsum(table(Y) / length(Y))
```

```
##    A    O    S    L
## 0.26 0.44 0.76 1.00
```

```
plot(ecdf(Y))
```

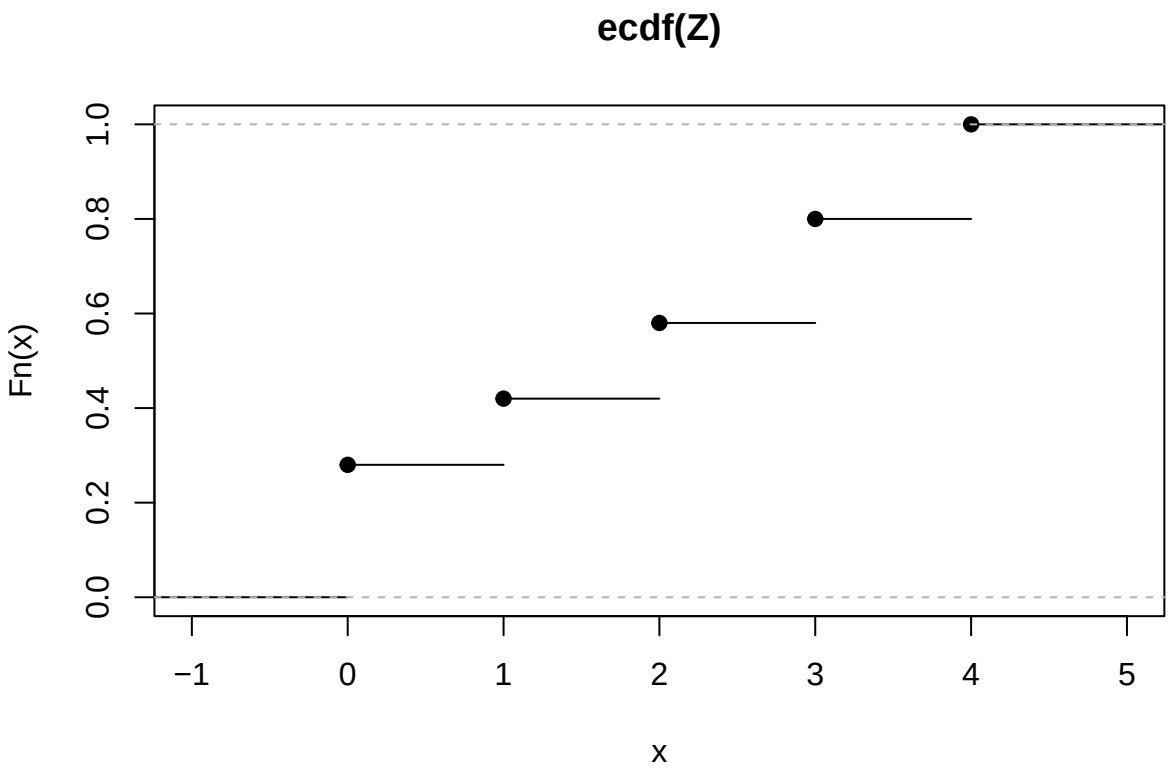


For numerical variables:

```
cumsum(table(Z) / length(Z))
```

```
##    0    1    2    3    4
## 0.28 0.42 0.58 0.80 1.00
```

```
plot(ecdf(Z))
```



If we try the same approach for the continuous variable W:

```
table(W)
```

```
## W
## 33.8775213811631  35.415381519041 35.6541383784125 46.8793378145431
##                1                1                1                1
## 52.7563552228405 53.5093799191515 54.6999662397492 55.6815339944367
##                1                1                1                1
## 56.8334866492771 58.2672500739823 58.3348777630801 58.3741853493409
##                1                1                1                1
##  60.620274255931 62.1671633352717 62.4897250497831 62.8292713451724
##                1                1                1                1
## 65.3465638930294 65.4002856629698 66.2775496879411 67.3636262541611
##                1                1                1                1
## 67.6755853135583 68.4544098140893  68.740736621617 68.7978424311212
##                1                1                1                1
## 69.2732924143287 69.5525561112997 70.5068119580714 70.5337080538567
##                1                1                1                1
## 70.6596373620813 70.9898462988698 71.9264050204625 72.3093811329276
##                1                1                1                1
## 72.5118550761639 72.5875217892273 73.7989398733261 74.0285707226046
##                1                1                1                1
## 74.3343427103841 75.1902425061576 75.5592639184782 75.7420006340731
##                1                1                1                1
## 75.9049996280411 77.9022338200939 78.0231473179288 79.6829638948667
##                1                1                1                1
```

```
## 81.4791220807459 81.7437461127363 82.6824333573602 91.9674895901944
##                1                1                1                1
## 94.2333713818525 95.9258755494911
##                1                1
```

It provides no useful information because all values are different, reflecting that W is described by a continuous random variable.

Continuous Variables

Let's now explore descriptive statistics for continuous numerical variables.

Consider the following example: in a factory assembling devices, three different production line organizations are tested over three days: the current organization, old, and two new ones, new1 and new2. The productivity, measured as the number of completed operations, is recorded for 288 workers and stored in the data set organization.

```
organization <- read.table("organizzazione.txt", sep=",")
```

```
str(organization)
```

```
## 'data.frame':    288 obs. of  3 variables:
##  $ old : int  676 676 680 681 682 681 686 685 689 686 ...
##  $ new1: int  732 732 734 733 736 738 738 736 737 736 ...
##  $ new2: int  731 731 730 732 733 733 734 730 732 733 ...
```

```
attach(organization)
```

Start by studying the variable old. Basic statistics include mean, standard deviation, variance, median, minimum, and maximum:

```
mean(old)
```

```
## [1] 705.3333
```

```
sd(old)
```

```
## [1] 10.2512
```

```
var(old)
```

```
## [1] 105.0871
```

```
median(old)
```

```
## [1] 706
```

```
range(old)
```

```
## [1] 676 729
```

Empirical quantiles can be obtained with the `quantile()` function:

```
quantile(old)
```

```
##   0%  25%  50%  75% 100%
##  676  699  706  713  729
```

```
quantile(old, seq(0, 1, 0.1))  # Deciles
```

```
##    0%   10%   20%   30%   40%   50%   60%   70%   80%   90%  100%
## 676.0 692.0 697.0 700.0 703.0 706.0 708.0 711.0 714.0 718.3 729.0
```

A summary of these key statistics can be obtained using the `summary()` function:

```
summary(old)
```

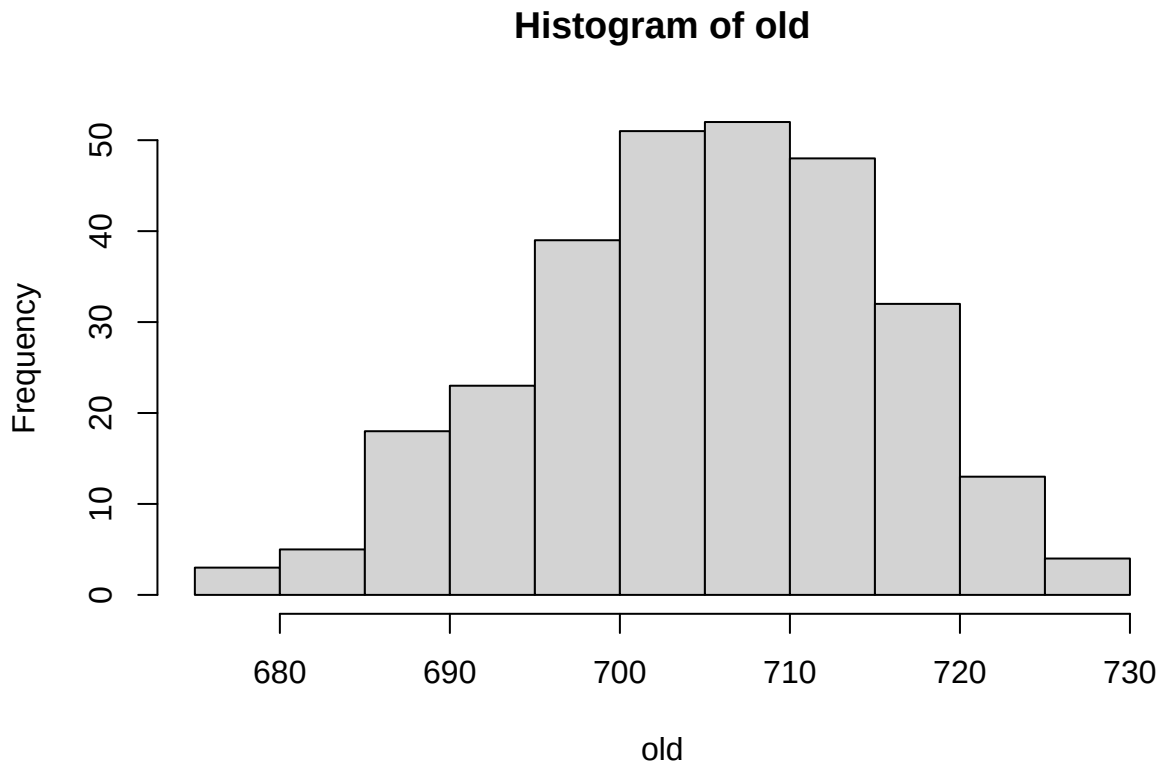
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   676.0   699.0   706.0   705.3   713.0   729.0
```

```
summary(organization)
```

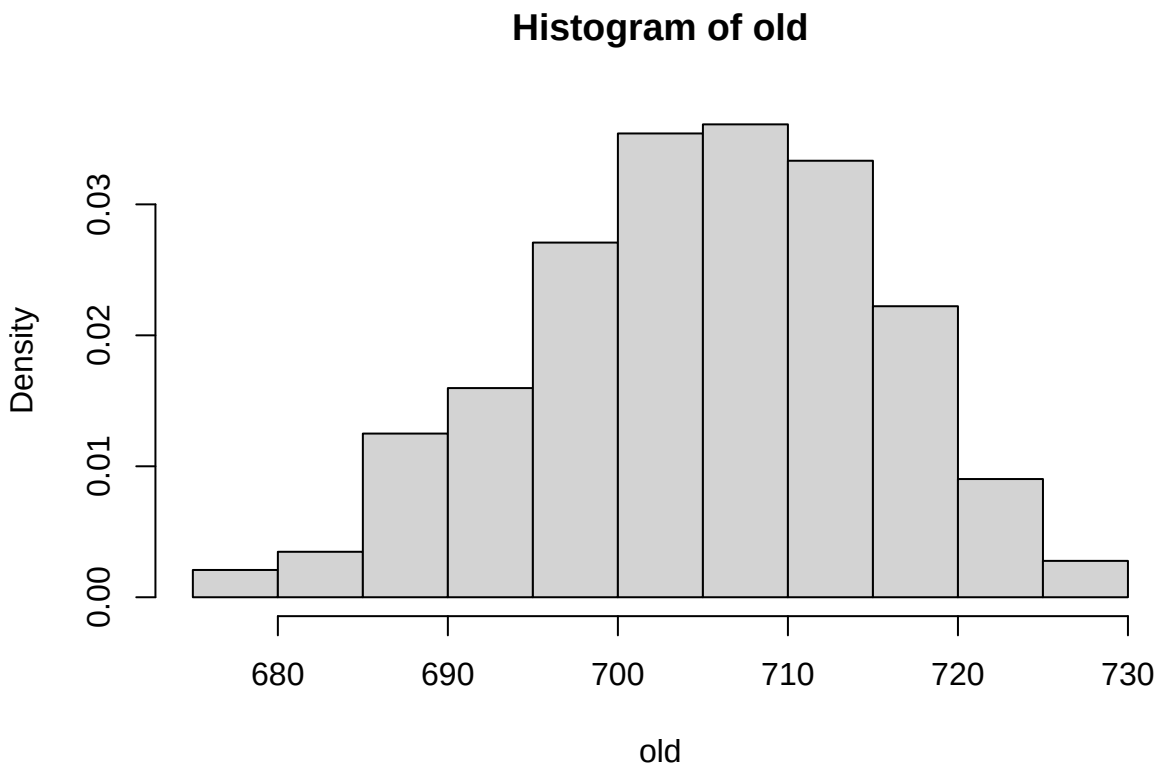
```
##       old             new1            new2
##  Min.   :676.0   Min.   :671.0   Min.   :679.0
##  1st Qu.:699.0   1st Qu.:688.0   1st Qu.:707.0
##  Median :706.0   Median :699.0   Median :719.0
##  Mean   :705.3   Mean   :700.8   Mean   :719.2
##  3rd Qu.:713.0   3rd Qu.:711.2   3rd Qu.:730.2
##  Max.   :729.0   Max.   :759.0   Max.   :758.0
```

Frequency distributions of continuous numerical variables can be visualized with histograms:

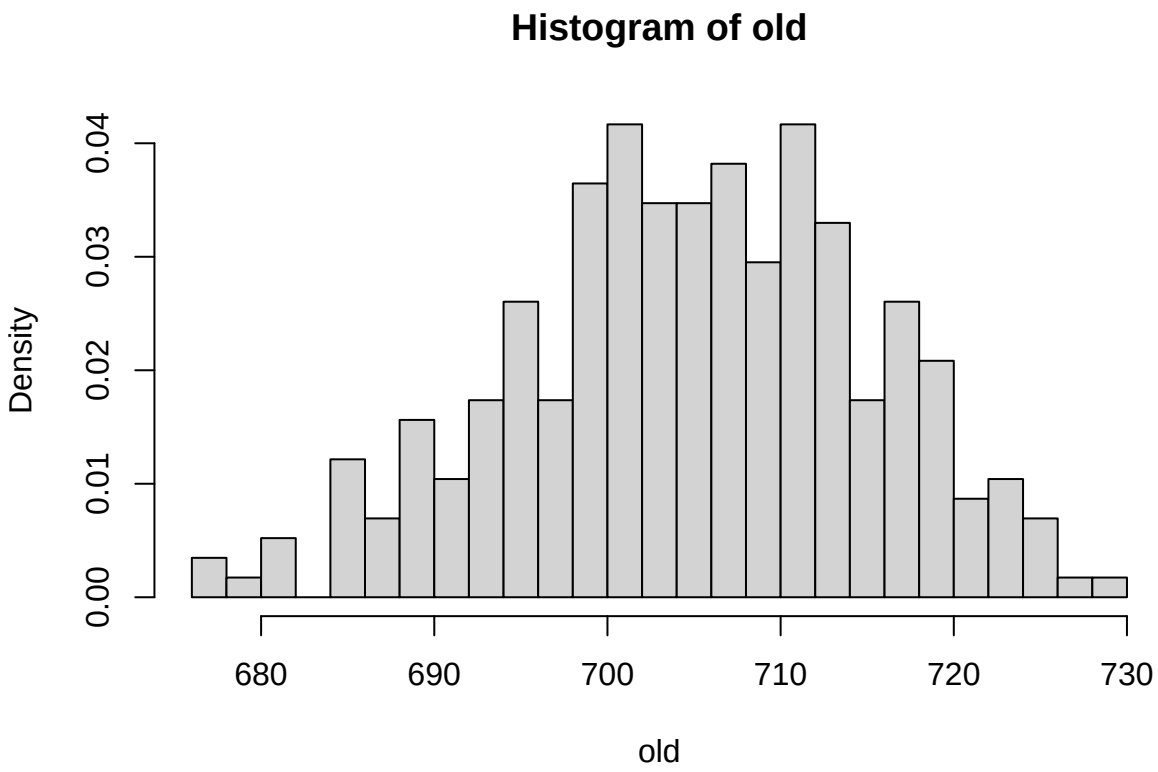
```
hist(old)
```




```
hist(old, freq = FALSE)  # Density
```

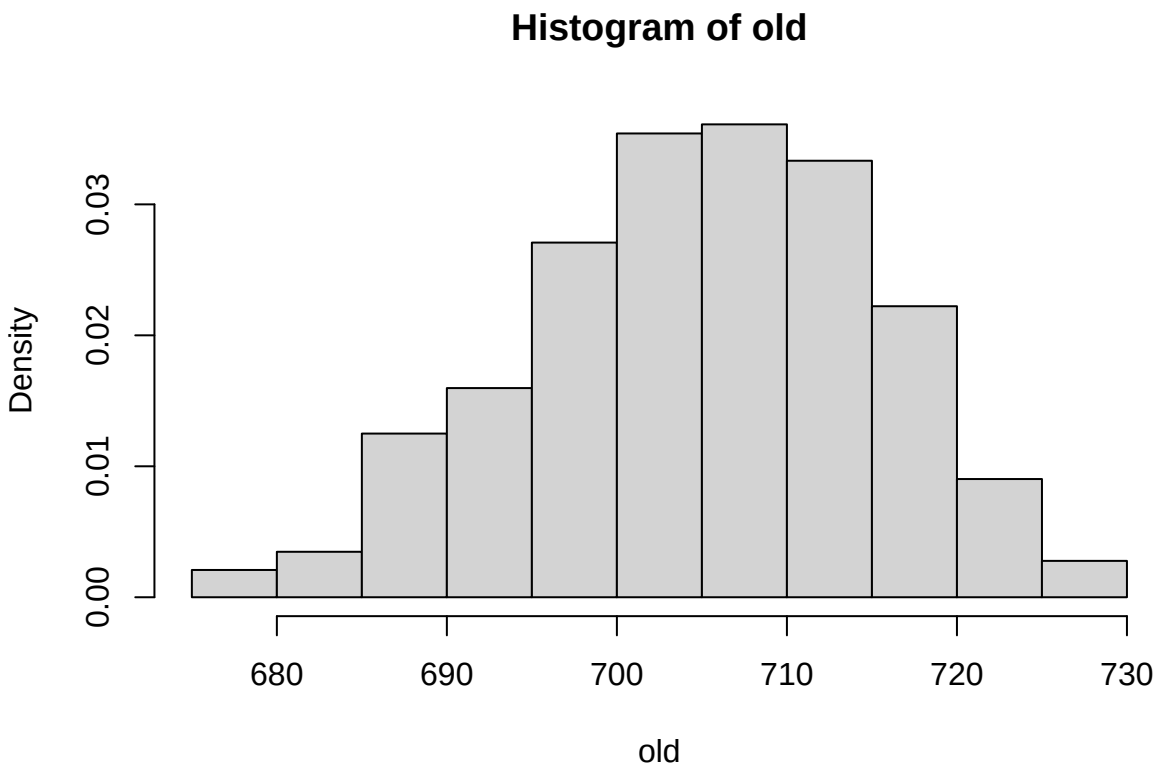



```r
hist(old, freq = FALSE, breaks = 20)  # Custom intervals
```



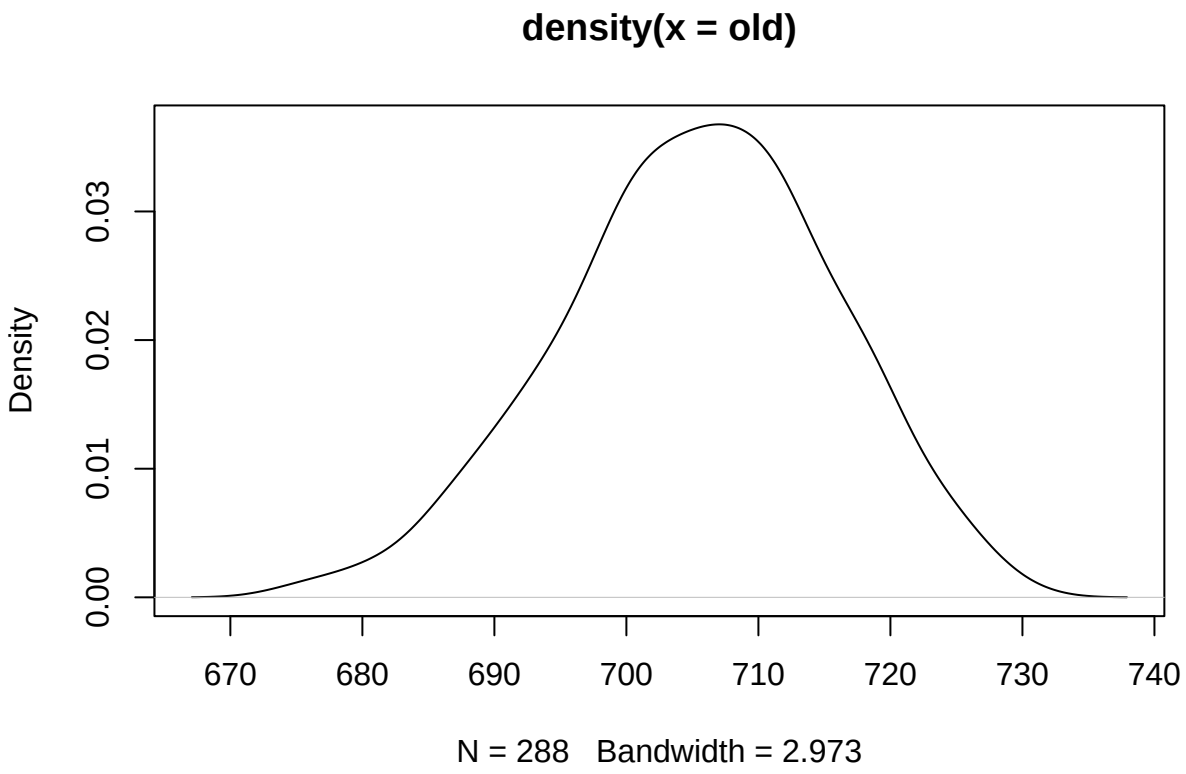
The Freedman-Diaconis method for choosing interval widths:

```
hist(old, freq = FALSE, breaks = "FD")
```



Kernel density estimation provides a continuous approximation of the empirical probability density:

```
plot(density(old))
```

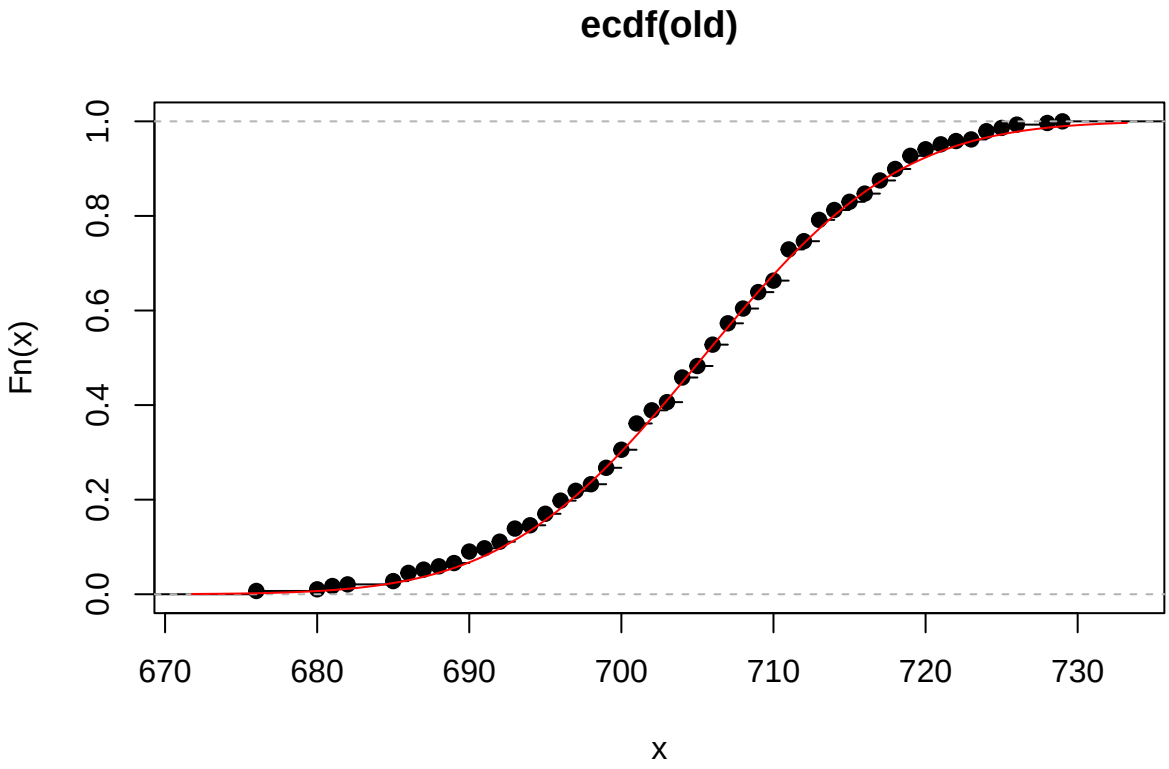


To compare with a known probability density (e.g., normal distribution):

```
hist(old, freq = FALSE, breaks = seq(665, 740, 5), ylim = c(0, 0.04))
curve(dnorm(x, mean = mean(old), sd = sd(old)), col = "red", add = TRUE)
```

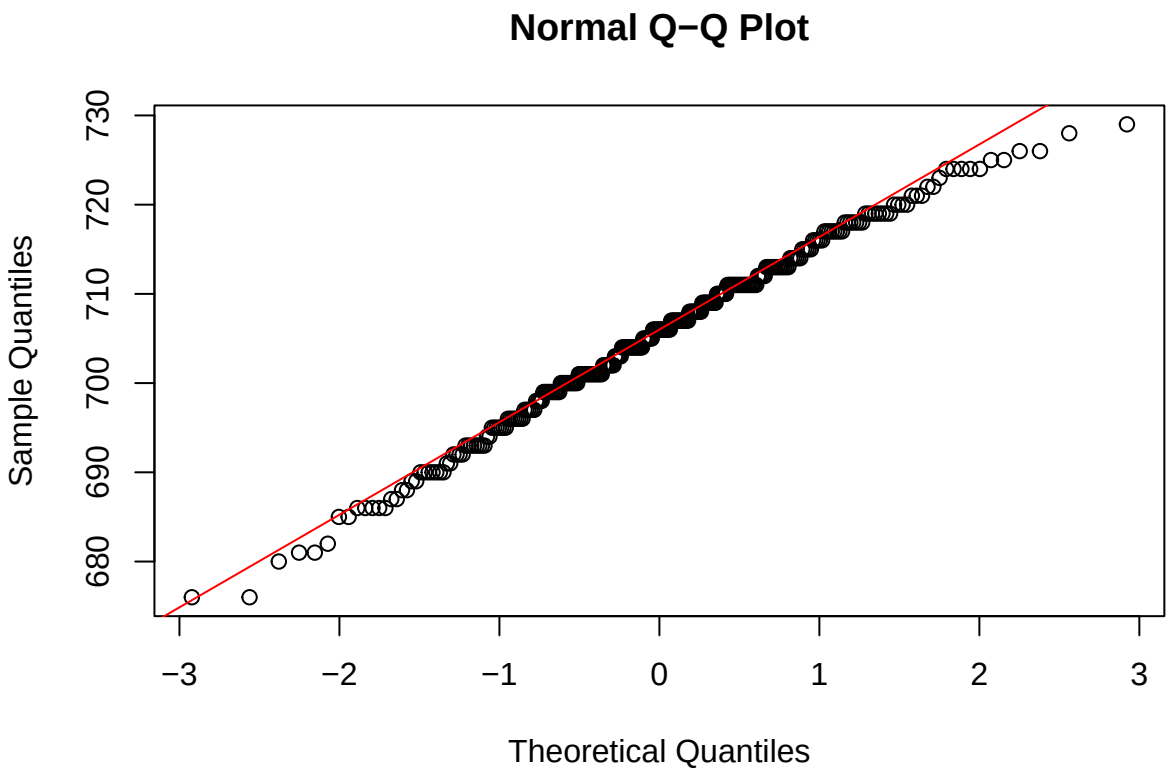
The empirical cumulative distribution function (CDF):

```
plot(ecdf(old))
curve(pnorm(x, mean = mean(old), sd = sd(old)), add = TRUE, col = "red")
```



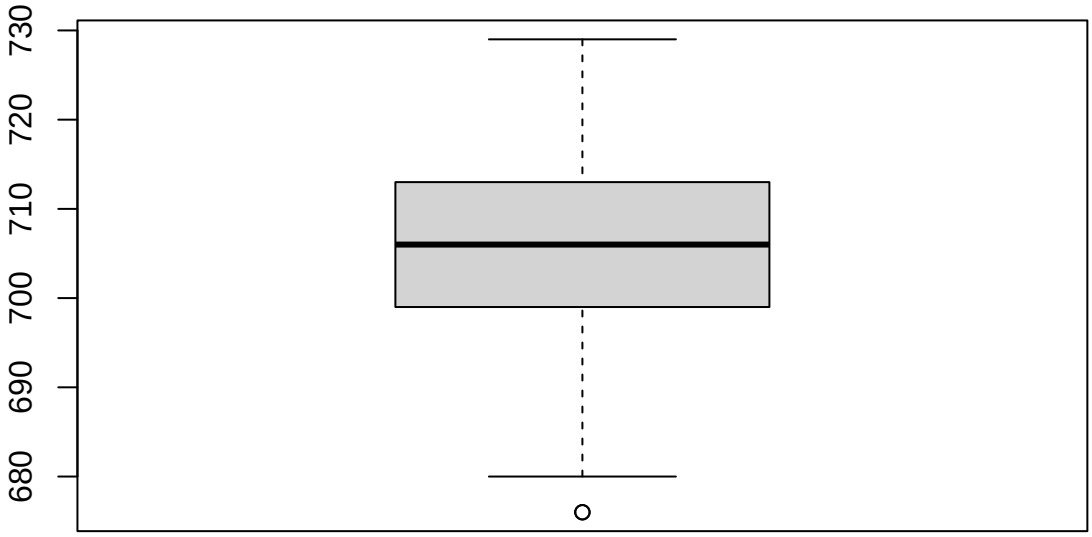
A QQ-plot compares the data against theoretical quantiles of a normal distribution:

```
qqnorm(old)
qqline(old, col = "red")
```



Finally, boxplots summarize distributions visually:

```
boxplot(old)
```



The three horizontal lines that make up the box in a **boxplot** represent the quartile values (from bottom to top: Q1, median, and Q3). The outer lines (the "whiskers") indicate the minimum and maximum values within a distance of $1.5 \times$ IQ from the nearest quartile, where IQ is the interquartile range. Any observation outside this range is considered "extreme," or an **outlier**, and is represented separately. This type of representation can be particularly useful when comparing two or more distributions.

The primary role of the statistics calculated so far is to characterize **location**, such as the mean and median, and **scale**, such as the standard deviation and interquartile range. To these measures, we can add **skewness** and **kurtosis**.

Skewness **Skewness** is a measure of symmetry, or more precisely, the lack of symmetry in a distribution. A distribution is symmetric if it is evenly distributed around its central point. For example, the standard Gaussian distribution is symmetric around the mean $\mu = 0$.

Given a sample $(Y_1, \ldots, Y_N)$, skewness can be calculated as:

$$g_1 = \frac{\sum_{i=1}^{N} (Y_i - \bar{Y})^3 / N}{s^3}$$

where $\bar{Y}$ is the sample mean, s is the standard deviation, and N is the number of observations. Note that here s is calculated by dividing by N rather than $N - 1$. This formula is known as the Fisher-Pearson coefficient of skewness. Some software also implements an adjusted version:

$$G_1 = \frac{\sqrt{N(N-1)}}{N-2} \cdot \frac{\sum_{i=1}^{N} (Y_i - \bar{Y})^3 / N}{s^3}$$

The skewness of a normal distribution is 0, as is the case for any symmetric distribution. Negative skewness

indicates that the observations are more concentrated on the left, while positive skewness indicates a greater concentration on the right, with the right tail being longer than the left.

Kurtosis **Kurtosis** compares the tails of a data distribution to those of a normal distribution, providing information on whether the distribution has heavier or lighter tails. The formula for kurtosis is:

$$k_1 = \frac{\sum_{i=1}^{N}(Y_i - \bar{Y})^4/N}{s^4}$$

where $\bar{Y}$ and s are the sample mean and standard deviation, and N is the number of observations. Again, s is calculated by dividing by N. The kurtosis of the standard normal distribution is 3. Therefore, a more commonly used definition is:

$$k_2 = \frac{\sum_{i=1}^{N}(Y_i - \bar{Y})^4/N}{s^4} - 3$$

This adjustment makes the kurtosis of the standard normal distribution equal to 0. Positive values (*heavy-tailed*) indicate that the distribution has larger tails than a normal distribution, while negative values (*light-tailed*) indicate smaller tails.

What is an outlier?

An *outlier* is a value or an observation that is distant from other observations, that is to say, a data point that differs significantly from other data points. Alternatively, outliers can be defined as observations not following the *assumed model*, that is values that deviate so much from other observations one might suppose a different underlying sampling mechanism.

How to detect outliers R?

- Descriptive statistics
- Histogram
- Boxplots
- Percentiles

Outlier detection should be an important step during pre-processing, that is the identification of observations which "behave" differently with respect the majority of the data.
Sometimes outliers may be due to some errors (measurement, reporting, ...) and then they can be discarded, substitued with `NA` (and then apply a method able to handle `NA`), or substituted with an *estimate* of their value.
However, there is the possibility that these "anomalous" observations underline a specific pattern in the data, and then should be included and considered during the statistical analysis.

What is a missing value?

A missing value is one whose value is unknown. Missing values are represented in R by the `NA` (not available) symbol. `NA` is a special value whose properties are different from other values.
Missing values can be informative, understanding if the missing is due to some reporting error or underline some specific pattern. One possible option during pre-processing would be removing the entire observations (rows) with at least one `NA`, or entire columns, if `NA`'s occur only for specific variables.
Remark: In both cases we lose information, removing all the related available data.
This may be a problem when our sample has few observations. An alternative option is to consider methods, from the literature, specifically designed for handling `NA`.

Exercises

Exercise 6

Write a function named `kurtosis` to calculate the kurtosis indices k_1 and k_2 given in the formulas:

$$k_1 = \frac{\sum_{i=1}^{N}(Y_i - \bar{Y})^4/N}{s^4}$$

$$k_2 = \frac{\sum_{i=1}^{N}(Y_i - \bar{Y})^4/N}{s^4} - 3$$

Write a function named `skewness` is a similar way.

Exercise 7

The data file `fondi.txt` contains the returns of 30 funds categorized into two types labeled `A` and `B`. Analyze the two variables separately. Specifically:

1. Create a table with the minimum, Q_1, mean, median, Q_3, maximum, standard deviation, and interquartile range for both variables, rounded to two decimal places.
2. Plot the density and empirical cumulative distribution functions for each variable separately.
3. Compute skewness and kurtosis indices.
4. Compare the two variables using boxplots and empirical cumulative distribution functions after standardizing the data.

Comment on the results.

Exercise 8

We have 50 subjects (25 with Alzheimer disease and 25 healthy individuals).
We analyze the response time (in milliseconds) of the Decoding Test VIPER-NAM: Images will appear on the screen for a short period of time and then disappear. Four letters will then appear, only one of which will correspond to the letter of the object. The user must choose the correct letter as quickly as possible. The response time is collected 10 times for each individual.
We also "collect" the Attention Control Scale (ATTC, i.e., self-report scale that is designed to measure attention focusing and attention shifting). The ATTC consists of 20 items (we will consider only one here) that are rated on a four-point Likert scale from 1 (almost never) to 4 (always). We also "collect" sex and age for each individual.

```r
set.seed(1234)

generateData <- function(time){

  db <- data.frame(Age = sample(c(15:60), 50, replace = TRUE),
                   Sex = sample(c(0, 1), 50, replace = TRUE),
                   Group = c(rep(0, 25),rep(1, 25)))

  db <- db %>% mutate(ATTC1 = ifelse(Group == 1,
                                     sample(c(3,4),1),
                                     sample(c(1:4),1)),
                      Response_Time = log(Age) * rgamma(50, shape = 300) +
                        log(time) * rgamma(50, shape = 300) +
                        Sex * rgamma(50, shape = 300) +
                        Group *rgamma(50, shape = 300) +
                        log(ATTC1) * rgamma(50, shape = 300))
```

```
  return(db)
}

db <- sapply(c(1:10), function(x) generateData(x), simplify = FALSE)
db <- bind_rows(db)
db$Time <- rep(1:10, each = 50)
db$ID <- rep(1:50, 10)
```

- Structure and summary of the data