Ingeniería y Ciencia de Datos 1

Máster Universitario de Ingeniería Informática

Domingo López Rodríguez
Ángel Mora Bonilla

Published by Domingo López Rodríguez

dominlopez.quarto.pub/icd1

2024-10-01

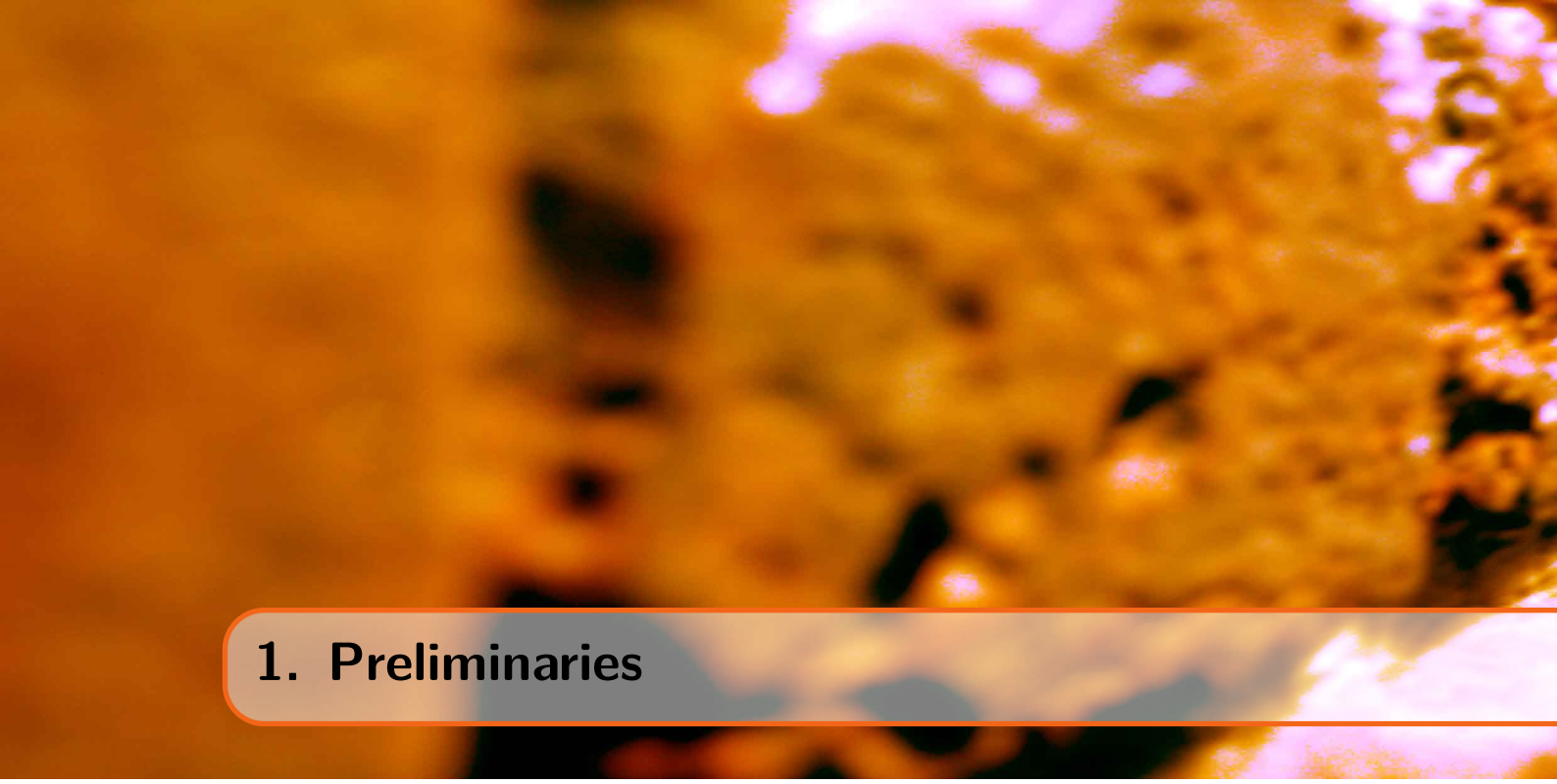
Contents

Preface

This book brings together some of the most common concepts about Data Science for our course *Ingeniería y Ciencia de Datos 1*, providing solved examples and the essential theory to understand the solutions.

We do not aim to replace full textbooks with formal proofs, but rather to complement them.

Each chapter addresses the most common questions and procedures in each topic.



1. Preliminaries

In this section, we recall some fundamental concepts in statistics, which are essential for understanding data and its characteristics. This is not intended to be an exhaustive dictionary of terms and theoretical results, just a reminder of some ideas and concepts that a student should possess by the time he/she starts the Master.

1.1 Descriptive Statistics

Descriptive statistics provide simple summaries of the data and form the foundation for further analysis. They help to describe, show, or summarize data in a meaningful way. Below, we explore key measures:

1.1.1 Central Tendency Measures

These metrics indicate the central point around which the data is distributed.

Remember 1.1 — Mean. The **mean** is the arithmetic average, calculated as the sum of all observations divided by the number of observations. It is calculated as:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

```
# Example in R
data <- c(5, 8, 3, 10, 5, 6)
mean(data)
## [1] 6.166667
```

Remember 1.2 — Median. The **median** is the middle value of a data set when it is ordered from least to greatest. If the number of observations is even, it is the average of the two middle numbers.

```
median(data)
## [1] 5.5
```

Remember 1.3 — Mode. The **mode** is the value that appears most frequently in the data set. A data set may have more than one mode.

```
# Using the mode function to find the most frequent value
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
getmode(data)
## [1] 5
```

Remember 1.4 — Quartile. Quartiles divide the data into four equal parts, each representing 25% of the data points. There are three quartiles:

- **Q1 (First Quartile)**: The value below which 25% of the data falls.
- **Q2 (Second Quartile or Median)**: The value below which 50% of the data falls.
- **Q3 (Third Quartile)**: The value below which 75% of the data falls.

Quartiles provide a robust way of understanding the spread of data without being affected by extreme values, making them especially useful when comparing distributions.

$$Q_1 = 25\text{th percentile}, \quad Q_2 = 50\text{th percentile (Median)}, \quad Q_3 = 75\text{th percentile}$$

```
quantile(data)
##   0%  25%  50%  75% 100%
##  3.0  5.0  5.5  7.5 10.0
```

1.1.2 Statistical Dispersion Measures

Dispersion metrics quantify the spread or variability of the data, indicating how spread out the values are.

Remember 1.5 — Variance. Variance is the average of the squared differences from the mean. It provides insight into the degree of spread in the data:

$$\text{var}(X) = \sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

```
var(data)
## [1] 6.166667
```

Remember 1.6 — Standard deviation. The **standard deviation** measures the amount of variation or dispersion in a data set. It is the square root of the variance:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

```
sd(data)
## [1] 2.483277
```

Remember 1.7 — Interquartile range. The **Interquartile range (IQR)** is a measure of statistical dispersion, representing the range within which the central 50% of values fall. It is the difference between the third quartile (Q3) and the first quartile (Q1):

$$\text{IQR} = Q_3 - Q_1$$

```
IQR(data)
## [1] 2.5
```

Remember 1.8 — Range. The **range** is the difference between the maximum and minimum values in the dataset:

$$\text{Range} = \max(x) - \min(x)$$

```
range(data)
## [1]  3 10
```

1.1.3 Skewness

Remember 1.9 — Skewness. Skewness quantifies the asymmetry of the probability distribution of a real-valued random variable about its mean. Positive skew indicates that the right tail is longer, while negative skew indicates that the left tail is longer.

$$\text{Skewness} = \frac{n}{(n-1)(n-2)} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s} \right)^3$$

```
library(e1071) # We need another library
## Warning: package 'e1071' was built under R version 4.3.3
skewness(data)
## [1] 0.299904
```

Remember 1.10 — Kurtosis. Kurtosis measures the "tailedness" of the probability distribution. High kurtosis indicates heavy tails, while low kurtosis indicates light tails compared to a normal distribution.

$$\text{Kurtosis} = \frac{n(n+1)}{(n-1)(n-2)(n-3)} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s} \right)^4 - \frac{3(n-1)^2}{(n-2)(n-3)}$$

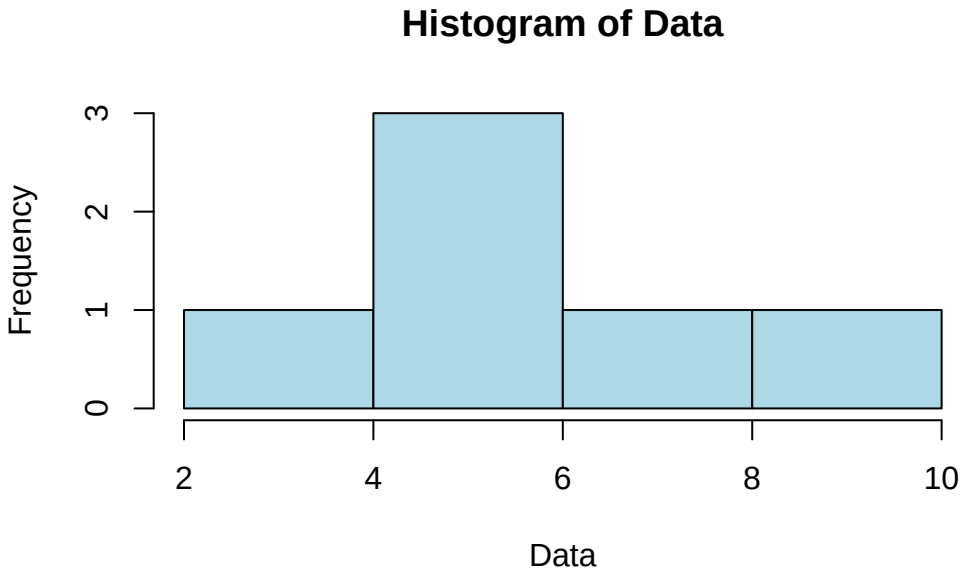
```
kurtosis(data)
## [1] -1.547176
```

1.1.4 Visualizing Descriptive Statistics

Visualizations are an essential part of descriptive statistics. Below, we create visualizations for the data, including a histogram, boxplot, and density plot. These help to visually assess the central tendency, spread, skewness, and potential outliers in the dataset.

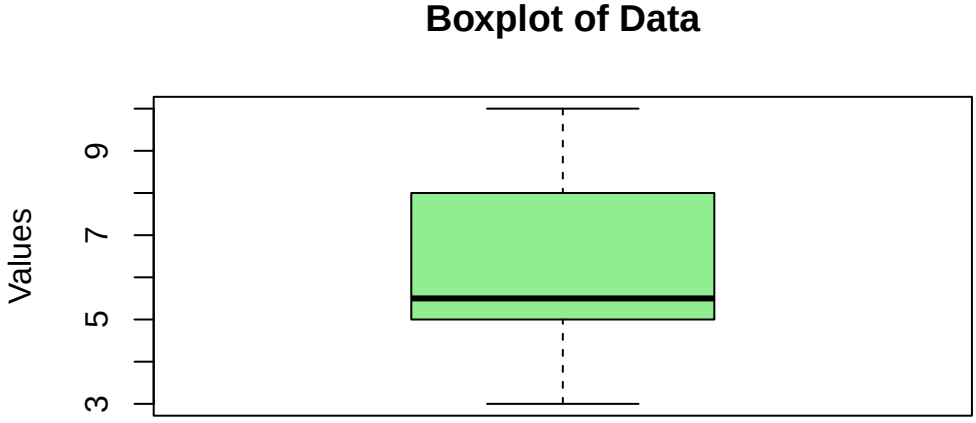
Remember 1.11 — Histogram. A **histogram** shows the distribution of a dataset by plotting the frequency of data points in discrete intervals (bins).

```
hist(data,
     main = "Histogram of Data",
     xlab = "Data",
     ylab = "Frequency",
     col = "lightblue",
     border = "black")
```



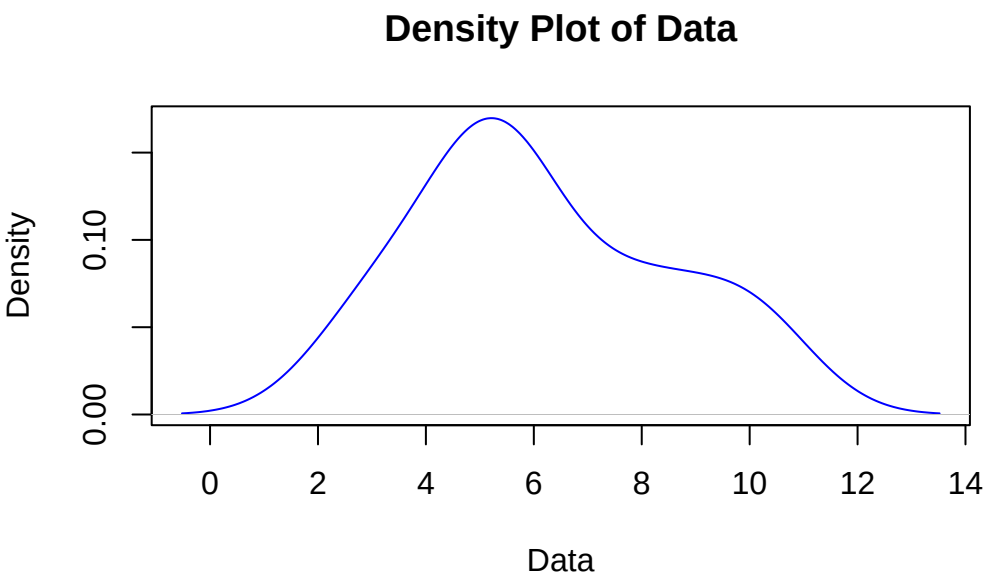
Remember 1.12 — Boxplot. A **boxplot** provides a graphical representation of the minimum, first quartile, median, third quartile, and maximum. It also highlights outliers.

```
boxplot(data,
        main = "Boxplot of Data",
        ylab = "Values",
        col = "lightgreen")
```



Remember 1.13 — Density plot. A **density plot** provides a smooth estimate of the distribution of the data, which can be useful to understand its overall shape.

```
plot(density(data),
     main = "Density Plot of Data",
     xlab = "Data",
     col = "blue")
```



By using these visualizations, we can better understand the data distribution and assess properties such as skewness, kurtosis, and the presence of outliers.

1.2 Random Variables and Probability

In this section, we explore key concepts related to random variables and their probability distributions, as well as important theorems in probability theory.

Remember 1.14 — Random variable. A **random variable** is a numerical quantity whose value is determined by chance.

1.2.1 Probability Distributions

Here, we recall some of the most used probability distributions.

Remember 1.15 — Distribution. A **probability distribution** describes how the values of a random variable are distributed, describing the likelihood of different values occurring for a random variable. Below are some commonly used probability distributions.

Remember 1.16 — Normal distribution. The **normal distribution**, also known as the Gaussian distribution, is characterized by two parameters: the mean (μ) and the standard deviation (σ). It is denoted as $X \sim N(\mu, \sigma^2)$.

The probability density function (PDF) of a normal distribution is given by:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Parameters:

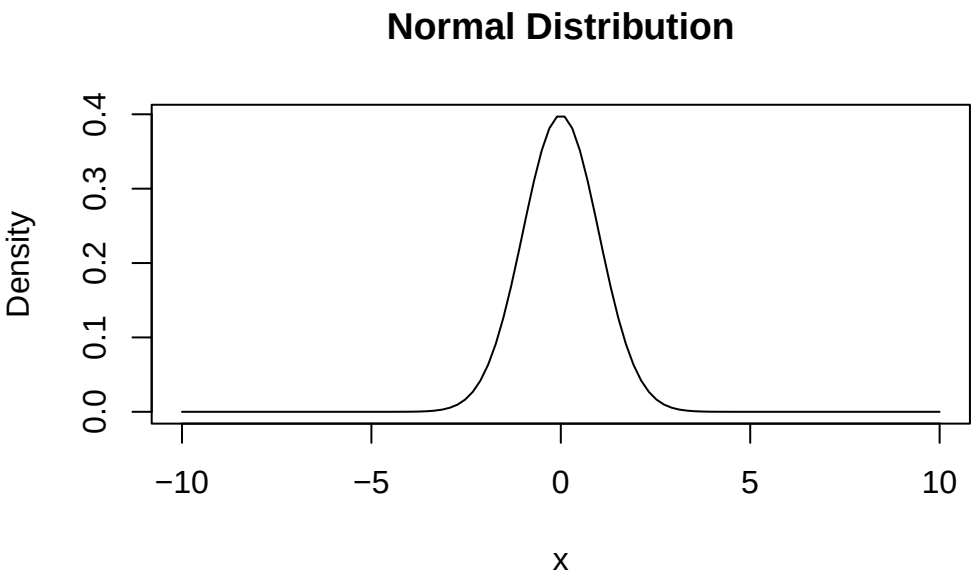
- μ: Mean of the distribution

- σ: Standard deviation of the distribution

Characteristics:

- Bell-shaped curve
- Symmetric around the mean
- Mean, median, and mode are all equal to μ
- Standard deviation parameter σ determines the spread of the distribution.

```r
# Example of a normal distribution in R
x <- seq(-10, 10, length = 100)
mu <- 0  # mean
sigma <- 1  # standard deviation
# dnorm gives the density function
y <- dnorm(x, mean = mu, sd = sigma)

plot(x, y,
     type = "l",
     main = "Normal Distribution",
     xlab = "x",
     ylab = "Density")
```



Remember 1.17 — Binomial distribution. The **binomial distribution** describes the number of successes in a fixed number of independent Bernoulli trials, each with the same probability of success p. It is denoted as $X \sim B(n,p)$.

The probability mass function (PMF) is given by:

$$P(X=k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad k = 0,1,2,\ldots,n$$

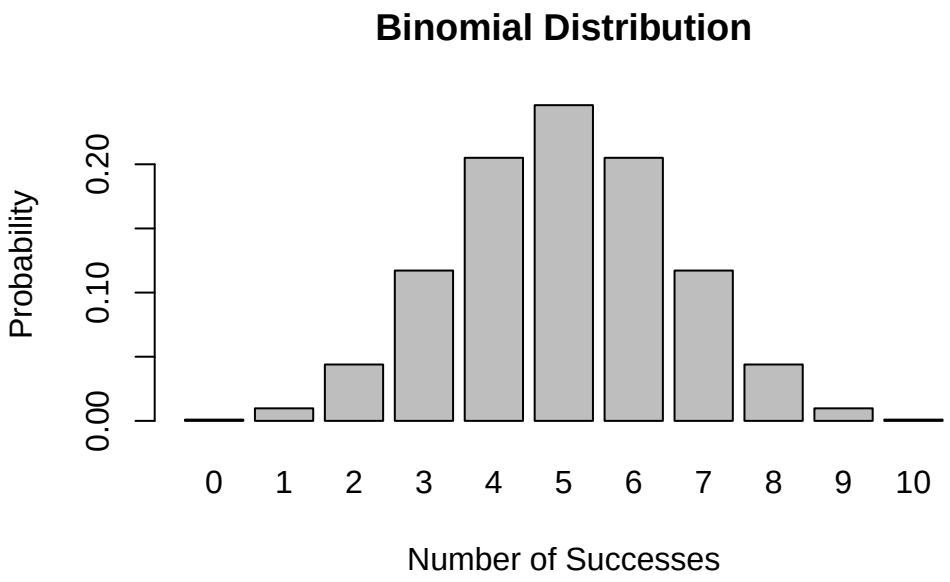
Parameters:

- n: Number of trials
- p: Probability of success in a single trial

Characteristics:

- Models the number of successes in a fixed number of independent Bernoulli trials
- Mean is np
- Variance is $np(1-p)$

```
n <- 10  # number of trials
p <- 0.5  # probability of success
x <- 0:n
y <- dbinom(x, size = n, prob = p)

barplot(y,
        names.arg = x,
        main = "Binomial Distribution",
        xlab = "Number of Successes",
        ylab = "Probability")
```



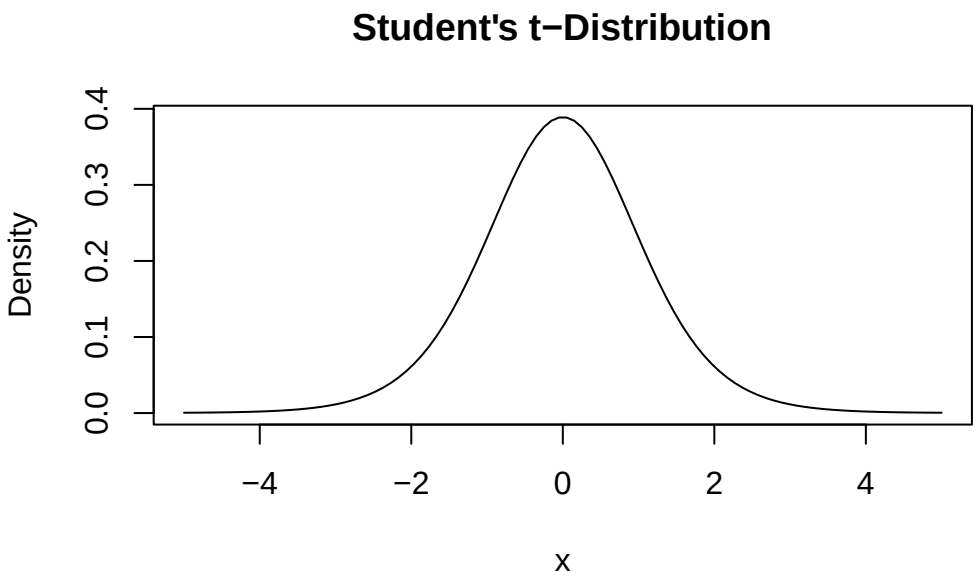
Remember 1.18 — Student's t distribution. **Student's t-distribution** is used when estimating the mean of a normally distributed population in situations where the sample size is small and population variance is unknown. It has a parameter, degrees of freedom (ν), and is denoted as $X \sim t_\nu$.

Remember 1.19 — Fisher-Snedecor's F-distribution. **Fisher-Snedecor's F-distribution** is used to compare two variances by analyzing the ratio of two independent chi-squared variables. It has two parameters, the degrees of freedom of the numerator (d_1) and the degrees of freedom of the denominator (d_2), an is denoted as $X \sim F(d_1, d_2)$.

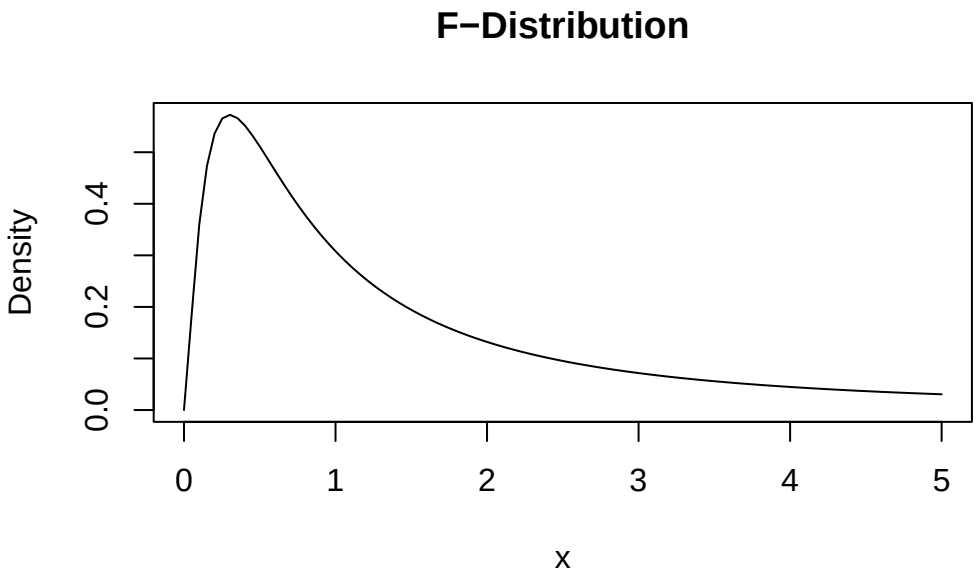
Remember 1.20 — Chi-square distribution. **Chi-Square distribution** is used in tests of independence and goodness of fit. It is denoted as $X \sim \chi^2_n$.

```
# Plotting t-distribution, F-distribution, and Chi-square distribution in R
# Student's t
x_t <- seq(-5, 5, length = 100)
y_t <- dt(x_t, df = 10)
# Ficher-Snedecor's F
```

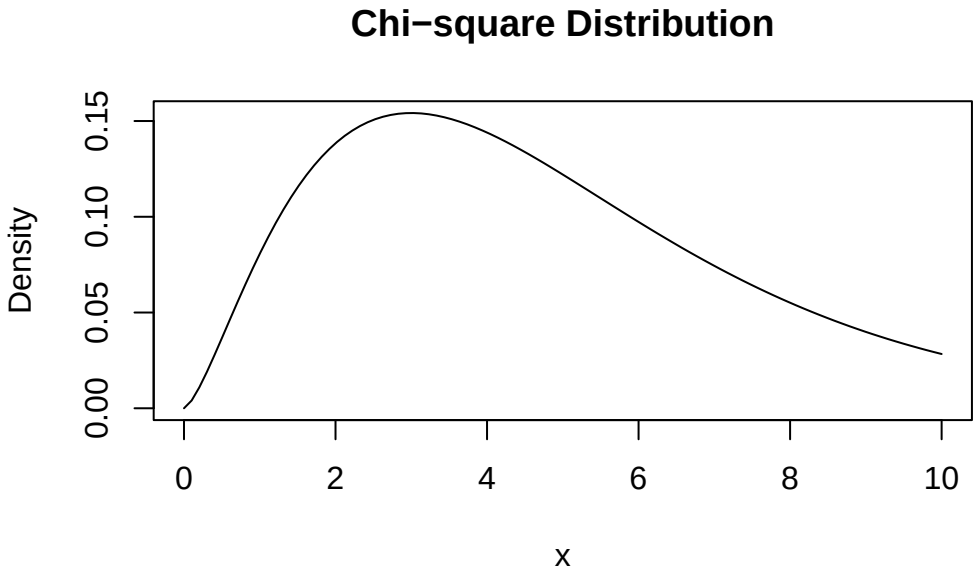
```r
x_f <- seq(0, 5, length = 100)
y_f <- df(x_f, df1 = 5, df2 = 2)
# Chi squared
x_chi <- seq(0, 10, length = 100)
y_chi <- dchisq(x_chi, df = 5)

# Plot t-distribution
plot(x_t, y_t,
     type = "l",
     main = "Student's t-Distribution",
     xlab = "x",
     ylab = "Density")
```




```r
# Plot F-distribution
plot(x_f, y_f,
     type = "l",
     main = "F-Distribution",
     xlab = "x",
     ylab = "Density")
```




```r
# Plot Chi-square distribution
plot(x_chi, y_chi,
     type = "l",
     main = "Chi-square Distribution",
     xlab = "x",
     ylab = "Density")
```



1.2.2 Probability Theory

The **Total Probability Theorem** is a fundamental rule that relates the probability of an event to the probabilities of conditional events that form a partition of the sample space.

Theorem 1.1 — Total probability. If $S_1, S_2, \ldots, S_n$ is a partition of the sample space, then the total probability of event A is

$$P(A) = \sum_{i=1}^{n} P(A|S_i)P(S_i)$$

■ **Example 1.1** If S_1 and S_2 are two mutually exclusive events that form a partition of the sample space, then:

$$P(A) = P(A|S_1)P(S_1) + P(A|S_2)P(S_2)$$

■

Bayes' theorem provides a way to update the probability of a hypothesis based on new evidence.

Theorem 1.2 — Bayes. Given two events A and B:

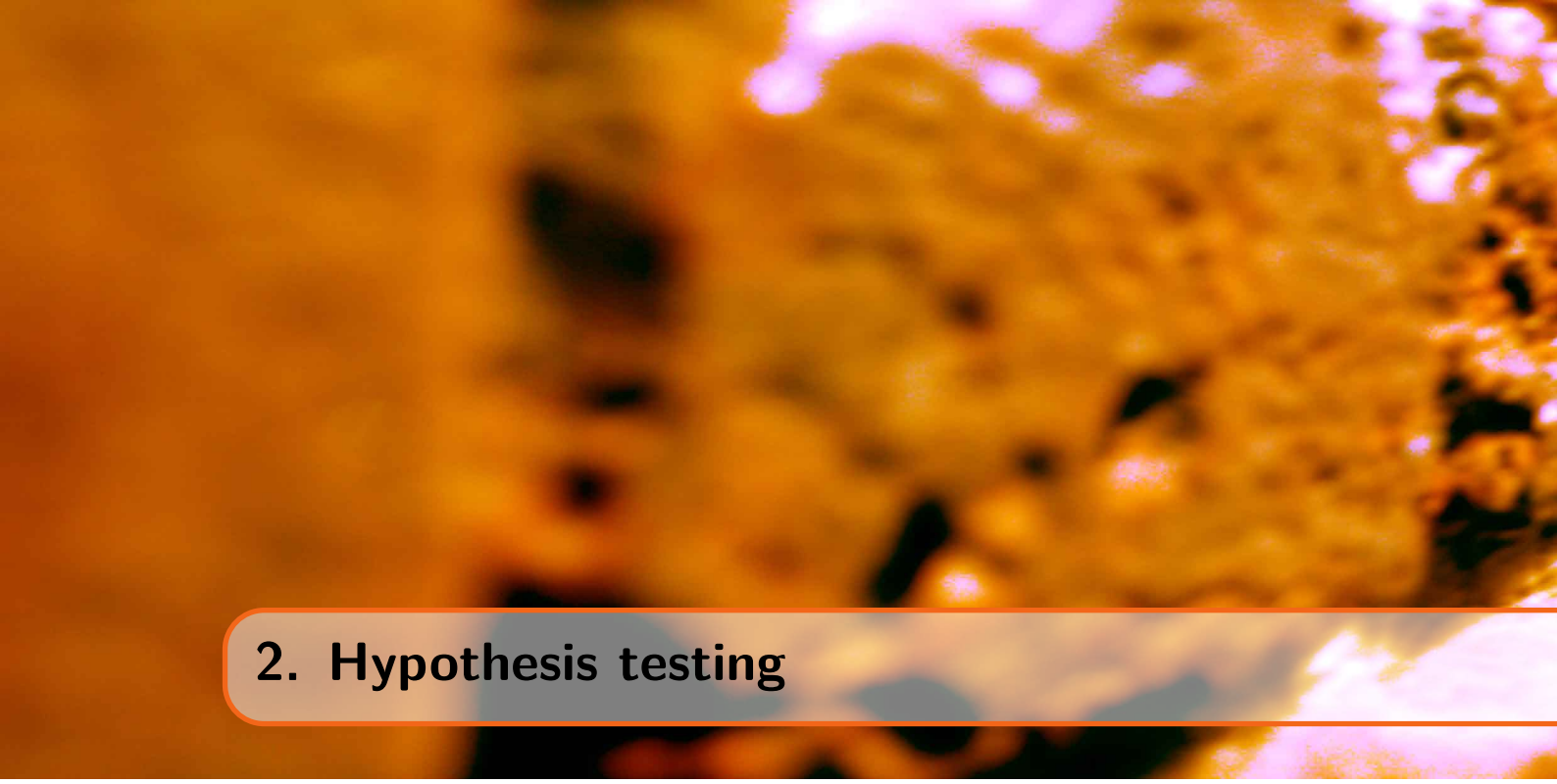
$$P(B|A) = \frac{P(B)P(A|B)}{P(A)}$$

where:

- $P(B|A)$ is the posterior probability: the probability of event B occurring given that A has occurred.
- $P(A|B)$ is the likelihood: the probability of event A occurring given that B has occurred.
- $P(B)$ is the prior probability: the initial estimate of the probability of B.
- $P(A)$ is the marginal likelihood: the total probability of A.

```
prior_B <- 0.2  # P(B)
likelihood_A_given_B <- 0.8  # P(A|B)
prob_A <- 0.5  # P(A)

posterior_B_given_A <- (prior_B * likelihood_A_given_B) / prob_A
posterior_B_given_A  # P(B|A)
## [1] 0.32
```

Statistics



2. Hypothesis testing

- It is virtually impossible to work on all the units of the group (**population**) you are interested in.
- It is usual to work on a subset of units (**sample**) taken at random and make inferences from this subset.
- By chance, your random sample may not be very representative of the population. Thus, even two samples taken from two similar populations may differ greatly.
- If there is really a difference between the samples, you need to know what differences you can expect by chance, and how to deal with the variation within samples.

Statistical tests are methods to use samples to make inferences about the populations.

Many statistical tests evaluate a strict null hypothesis H_0 against an alternative hypothesis H_A.

Procedure:

- Fit statistical models to data representing the hypothesis that we want to test.
- Use the probability to see when the results are likely to have happened by chance.

Then with these two tools, we test whether our statistical models (and therefore our hypothesis) fit the data.

We compute this idea with:

$$\text{test statistic} = \frac{\text{variance explained by the model}}{\text{variance not explained by the model}} = \frac{effect}{error}$$

There are lots of different tests, but all of them represent the same thing: **the amount of variance explained by the model**.

- The bigger the test statistic, the more unlikely it is for it to occur by chance.

- If the probability of obtaining the value of our test statistic is less than .05 then we generally accept the experiment hypothesis as true - **there is a significant effect of..** . But take care: the effect would not be important.

2.1 Hypothesis

- Hypothesis - prediction from your theory.

Hypothesis:

- H_A: Alternative hypothesis (experiment hypothesis). Ej: mean dispersal distance differs between male and female butterflies.
- H_0: Null hypothesis (opposite of H_A). Ej: mean dispersal distance does NOT differ between male and female butterflies

Statistical tests calculate a test statistic from the data to find out how likely the obtained result would be under the null hypothesis.

We can not prove H_A, but we can reject H_0.

2.1.1 Type I and Type II errors

- We use test statistics to tell us about the true state of the world.
- We are trying to check when there is an effect in the population.
- Two possibilities: an effect in the population and no effect.
- No way to know which of these possibilities is true.
- We can compute a test statistic and their associated probability to know which of the two is more likely.

Fisher said: *we must be very sure - 95% of confidence.* But it could occur an error.

Two mistakes:

- Type I: we believe that there is a genuin effect, and in fact, there is not.
- Type II: we believe that there is not a genuin effect, and in fact, there is.

To do so, a probability distribution of the test statistic is theoretically derived assuming the null hypothesis. The probability of the test statistic from the data given that the null hypothesis is true is then found using this theoretical distribution. This probability is termed the **P-value**.

> If the test statistic calculated from the data happens to be a value that is very rare under the null hypothesis, usually occurring at a probability of less than 5%, **(P-value $<$ 0.05), the null hypothesis is discarded** and the alternative hypothesis accepted.

Test result	H_0 true	H_A true	Decision
P-value >= 0.05	Correct	Type II error (false negative)	H_0 accepted; H_A discarded
P-value < 0.05	Type I error (false positive)	Correct	H_0 discarded; H_A accepted

2.2 Parametric tests

Many of the statistical procedures are parametric tests based on the normal distribution.

Parametric test:

- requires data from one large catalogue of distributions that statisticians have described
- data to be parametric - certain assumptions must be true
- if you use a parametric test when your data is not parametric, the results are likely to be innacurate

Assumptions of parametric tests:

- Normally distributed data
- Homogeneity of variance: samples comes from populations with the same variance
- Independence: data from different persons are independient

2.2.1 T-tests

T-tests are used to determine whether the means of two groups (1, and 2) are equal to each other. Assumption: both groups are sampled from normal distributions with equal variances.

H_0 the two means are equal

H_a the two means are not equal

In R, t-tests are calculated using the command **t.test()**.

Classical t-test

The classical t-test considers that the variance of the two groups are equivalent. It is defined:

$$t = \frac{m_1 - m_2}{\sqrt{\frac{S^2}{n_1} + \frac{S^2}{n_2}}}$$

where m_i is the mean of the group i, n_i is the size of the group i and S^2 is an estimator of the pooled variance of the two groups.

$$S = \frac{\sum(x - m_1)^2 + \sum(x - m_2)^2}{n_1 + n_2 - 2}$$

2.2.2 Welch t-statistic

It considers that the variance of the two groups are different and/or unequal sample sizes

In this case S_i is the standard deviation of the two group i.

2.2.3 T-tests in R

```
# independent 2-group t-test
t.test(y ~ x) # where y is numeric and x is a binary factor
# independent 2-group t-test
t.test(y1, y2) # where y1 and y2 are numeric
```

```
# paired t-test
t.test(y1, y2, paired = TRUE) # where y1 & y2 are numeric
# one sample t-test
t.test(y, mu = 3)
# mu is a number indicating the true value of the mean
```

Example

Data simulated from a normal distribution

```
# rnorm is a function() that draws random numbers from a normal distribution.
x <- rnorm(10)
y <- rnorm(10)
t.test(x, y)
##
##  Welch Two Sample t-test
##
## data:  x and y
## t = 0.45978, df = 17.472, p-value = 0.6513
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.9156287  1.4272506
## sample estimates:
##  mean of x  mean of y
## -0.1723934 -0.4282043
```

Example

Given the following list of heights of persons, let us check whether the average human height is significantly different from 1.77m.

```
height <- c(1.43, 1.75, 1.85, 1.74, 1.65,
            1.83, 1.91, 1.52, 1.92, 1.83)
t.test(height, mu = 1.77)
##
##  One Sample t-test
##
## data:  height
## t = -0.52046, df = 9, p-value = 0.6153
## alternative hypothesis: true mean is not equal to 1.77
## 95 percent confidence interval:
##  1.625646 1.860354
## sample estimates:
## mean of x
##     1.743
```

Therefore, we cannot reject the hypothesis that average human height is significantly different from 1.77m.

Example

Dispersal distance in male and female butterflies.
- H_0: male butterflies dispersal is similar from female butterflies. - H_A: male butterflies dispersal is different from female butterflies.

```
distance <- c(3, 5, 5, 4, 5, 3, 1, 2, 2, 3)
sex <- c("male", "male", "male", "male", "male",
         "female", "female", "female", "female",
         "female")
```

Assume equal variances and perform a two-tailed test.

```
t.test(distance ~ sex, var.equal = TRUE)
##
##  Two Sample t-test
##
## data:  distance by sex
## t = -4.0166, df = 8, p-value = 0.003859
## alternative hypothesis: true difference in means between group female and group male i
## 95 percent confidence interval:
##  -3.4630505 -0.9369495
## sample estimates:
## mean in group female   mean in group male
##                  2.2                  4.4
```

Thus, male butterflies dispersal is significantly different from female butterflies.

We can also specify a one-sided alternative hypothesis by adding the argument alternative= "less" or alternative = "greater" depending on which tail is to be tested

```
t.test(distance ~ sex,
       var.equal = TRUE,
       alternative = "greater")
##
##  Two Sample t-test
##
## data:  distance by sex
## t = -4.0166, df = 8, p-value = 0.9981
## alternative hypothesis: true difference in means between group female and group male i
## 95 percent confidence interval:
##  -3.218516       Inf
## sample estimates:
## mean in group female   mean in group male
##                  2.2                  4.4
```

The results of these tests state that female dispersal distance is not significantly greater than male dispersal distance.

2.2.4 Paired sample t-test

The sleep of students is affected by an exam. You ask 6 students how long they sleep the night before an exam and the night after an exam.

- H_0: There is a significant difference in means.
- H_A: There is not a significant difference in means.

```
sleep.before <- c(4, 2, 7, 4, 3, 2)
sleep.after <- c(5, 1, 3, 6, 2, 1)
# Note the _paired = TRUE_ argument!!
t.test(sleep.before, sleep.after, paired = TRUE)
##
##  Paired t-test
##
## data:  sleep.before and sleep.after
## t = 0.79057, df = 5, p-value = 0.465
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -1.501038  2.834372
## sample estimates:
## mean difference
##       0.6666667
```

Thus the data does not support an effect of exams on students sleeping time. **There is not a statistically significant difference in the means.**

2.2.5 Independent samples

```
t.test(y1, y2, paired = FALSE)
#By default, the variances are unequal
t.test(y1, y2, paired = FALSE, var.equal = TRUE)
```

2.3 Correlation analysis

To assess the relationship between two variables.

The goal of correlation analysis is to determine how related two variables are. This differs from regression analysis, which seeks to determine a line of best fit from the relationship and assumes that predictor variables is directly affecting (causing) the outcomes of the response variable.

Remember: **Correlation is not causation!**

2.3.1 Pearson Correlation

This test seeks to determine the level of relatedness between two variables using a score that runs from -1 (perfect negative correlation) to 1 (perfect positive correlation). A value of zero indicates no correlation.

It is advisable to test the assumption of normality before running this test.

```
cor.test(iris$Sepal.Length,
         iris$Petal.Length)
##
##  Pearson's product-moment correlation
##
```

```
## data:  iris$Sepal.Length and iris$Petal.Length
## t = 21.646, df = 148, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8270363 0.9055080
## sample estimates:
##       cor
## 0.8717538
```

This data shows a highly-significant (P-value < 2.2e-16) and strongly positive (0.87) correlation between these two variables. Note that in this case, the P-value is used to reject the null hypothesis that the true correlation is equal to zero.

2.3.2 Spearman Correlation

Spearman's ρ (rho) determines the level of correlation of two variables ranging from -1 to 1. The difference between the two measures is that Spearman uses the *rank-order* of the data rather than the raw values.

```
cor.test(iris$Sepal.Length,
         iris$Petal.Length,
         method = "spearman")
## Warning in cor.test.default(iris$Sepal.Length, iris$Petal.Length, method =
## "spearman"): Cannot compute exact p-value with ties
##
##  Spearman's rank correlation rho
##
## data:  iris$Sepal.Length and iris$Petal.Length
## S = 66429, p-value < 2.2e-16
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.8818981
```

produced similar, but not identical, results compared with Pearson's R

2.4 Cross-tabulation and the χ^2 test

Basic contingency tables would have two categorical variables. In many cases we may wish to test whether the two grouping variables are independent or there is any dependence bewteen the two categorical variables.

Two random variables x and y are called independent if the probability distribution of one variable is not affected by the presence of another.

One of the most common ways to analyze contingency tables is with the χ^2-test (Chi-square test). The χ^2 tests work by first calculating the difference between expected and observed values:

$$\sum \frac{(Observed - Expected)^2}{Expected}$$

In this case, expected values are those that would be obtained in the ideal case where the two variables are independent. If variables are A and B, the independence of A and B occurs if $P(A = a \cap B = b) = P(A = a) \cdot P(B = b)$ for all possible values a and b of the variables.

We propose the following hypothesis:

- H_0: The two variables are independent.
- H_1: The two variables are related.

2.4.1 Example

See Effectiveness of a Drug Treatment

2.4.2 Example

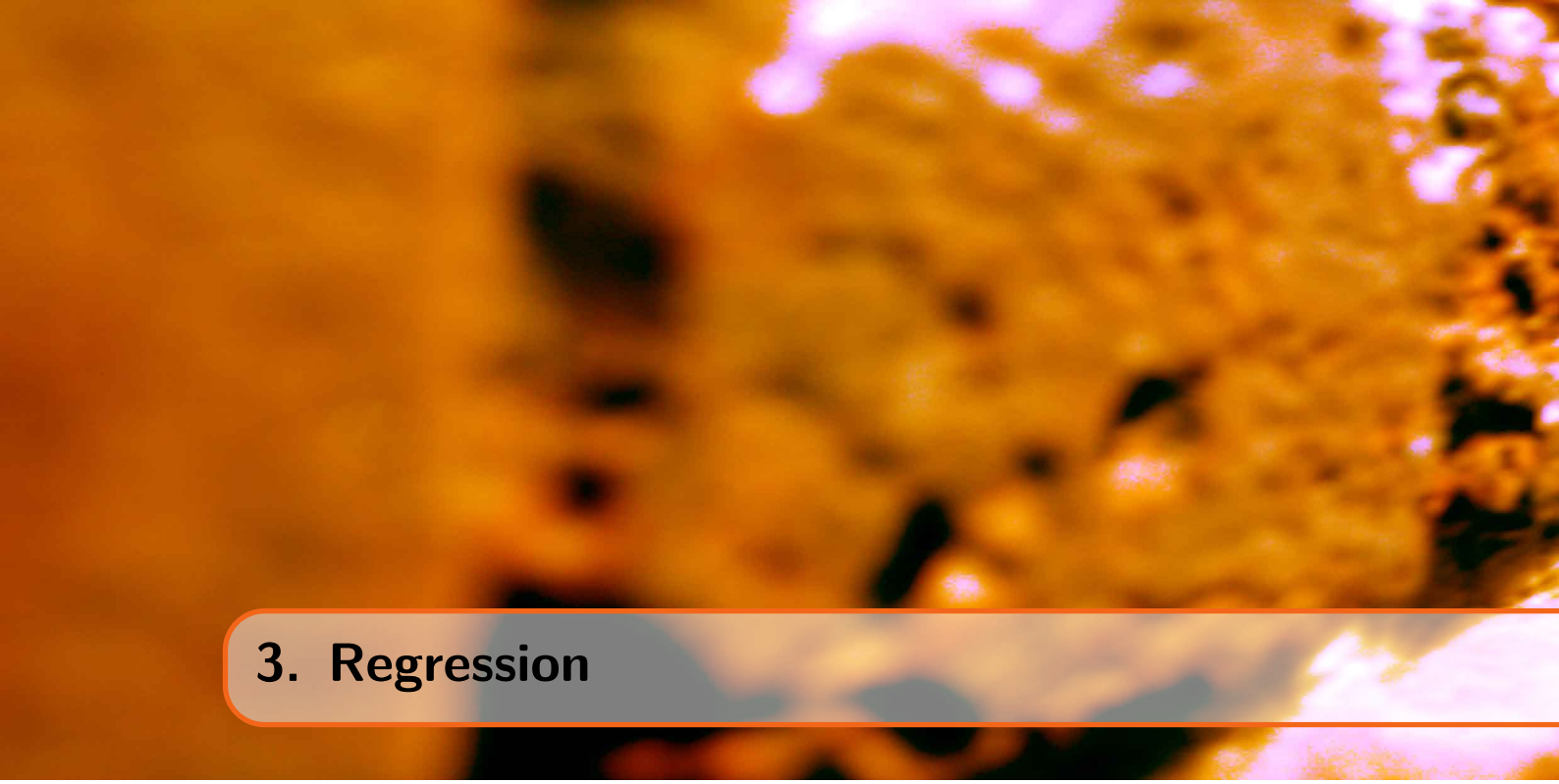
```
flyeyes <- read.csv(
  here::here("data",
             "3.9.csv"),
  header = T, stringsAsFactors = TRUE)
# Classical method with base R
tab <- table(flyeyes$Eyecolor, flyeyes$Group)
tab
##
##          A  B
##   Red   34 41
##   White 16  9
chisq.test(tab)
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  tab
## X-squared = 1.92, df = 1, p-value = 0.1659
```

```
# An alternative
library(lsr)
associationTest(~ Eyecolor + Group, data = flyeyes)
##
##      Chi-square test of categorical association
##
## Variables:   Eyecolor, Group
##
## Hypotheses:
##    null:        variables are independent of one another
##    alternative: some contingency exists between variables
##
## Observed contingency table:
##         Group
## Eyecolor  A  B
##    Red   34 41
##    White 16  9
```

```
##
## Expected contingency table under the null hypothesis:
##         Group
## Eyecolor    A    B
##    Red   37.5 37.5
##    White 12.5 12.5
##
## Test results:
##    X-squared statistic:  1.92
##    degrees of freedom:  1
##    p-value:  0.166
##
## Other information:
##    estimated effect size (Cramer's v):  0.139
##    Yates' continuity correction has been applied
```

The p-value indicates that these variables are independent.

The greater the difference between current and expected values, the greater the Chi-Sq value.



3. Regression

3.1 Modelling techniques

There are two types of basic numerical techniques to model the relationship between variables in a dataset: **interpolation** and **approximation** (adjustment).

In both cases, the goal is to get a function that best fits to a set of data (a cloud of data) from observations, data collected from sensors, datasets, etc.

Purpose:

- Prediction of future observations.
- Find relationships, functions between dataset variables.
- Description of the data structure.

3.1.1 Linear interpolation

The normal input is a data table $\{(x_i, y_i)\}$ and the goal is to find an interpolating function $\phi(x) = a_0 + a_1 f_1(x)$ (the easiest approach considers polynomial functions $f_1(x) = x$).

Then, the model will be:

$$\phi(x) = a_0 + a_1 x$$

where:

- a_0 is named the **intercept**.
- a_1 is named the **slope**.

This tries to find a straight line that passes through all the data points. This will only happen if all $\{(x_i, y_i)\}$ are collinear, which is not usually the case.

3.1.2 Polynomial interpolation

Assuming we have $n+1$ points $\{(x_0,y_0),(x_1,y_1),(x_2,y_2),\dots,(x_n,y_n)\}$, polynomial interpolation consists of finding a polynomial function $\phi(x)$ passing through the given points.

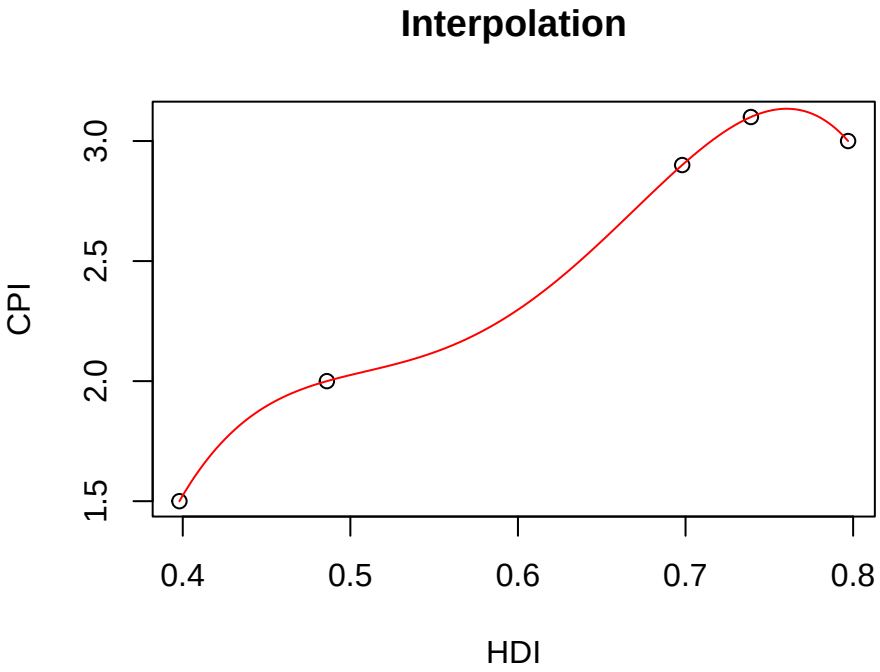
$$\phi(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n$$

Note that we are using that a basis of the polynomials of grade less than or equal to n is $B = \{1, x, x^2, \dots, x^n\}$.

If we impose $y_i = \phi(x_i)$, we obtain the following equations:

$$\left.\begin{array}{ll} a_0 + a_1 x_0 + a_2 x_0^2 + \dots & = y_0 \\ a_0 + a_1 x_1 + a_2 x_1^2 + \dots & = y_1 \\ \dots & \dots \\ a_0 + a_1 x_n + a_2 x_n^2 + \dots & = y_n \end{array}\right\}$$

This is a linear system that has a unique solution $\vec{a} = (a_0, \dots, a_n)$ if and only if all x_i are different.



3.1.3 Linear Regression: Minimizing the residual sum of squares

With a large number of observations, interpolation methods are not adequate, hence adjusting a *linear regression model* $y = \phi(x) = a_0 + a_1 x$ is used. Such a model reflects the effect of changing a variable x (independent variable) in variable y (dependent variable). It considers that there exist a linear relationship between the variables.

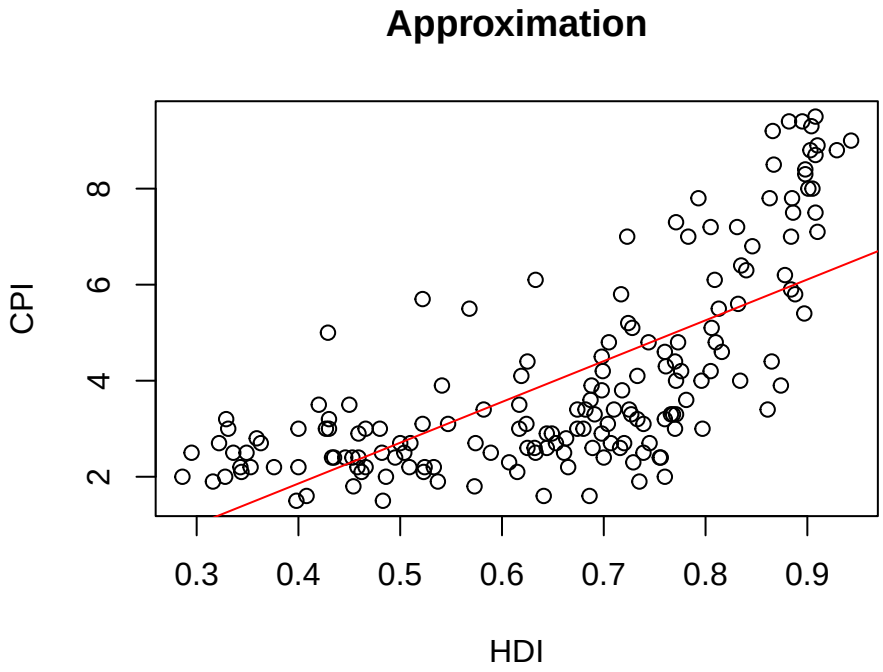
A common way to estimate the parameters of a statistical model is to *adjust* a function which minimize the errors. The most used method is that of minimizing the *residual sum of squares*, RSS, which is defined by:

$$RSS(\vec{a}) := \sum_{i=1}^{n} (y_i - (a_0 + a_1 x_i))^2$$

where $\vec{a} = (a_0, a_1)$.

To minimize the RSS, we just differentiate the above expression with respect to a_0 and a_1, equal to 0, and solve.

For those a_0, a_1, we have the linear model that we want to explain the changes in variable y due to changes in variable x.



Notes:

- Although regression is probably the most simple approach for supervised learning, linear regression is still a useful and widely used statistical learning method.
- Other more complex statistical learning approaches are generalizations of linear regression.

3.2 Linear regression with one predictor

The easier model is linear regression with one single predictor:

$$\phi(x) = a_0 + a_1 x$$

Finding this model is equivalent to finding a straight line that best fits the point cloud: $y_i \approx \phi(x_i) = a_0 + a_1 x_i$, minimizing the RSS.

3.2.1 Linear regression using R

We use the `lm()` function.

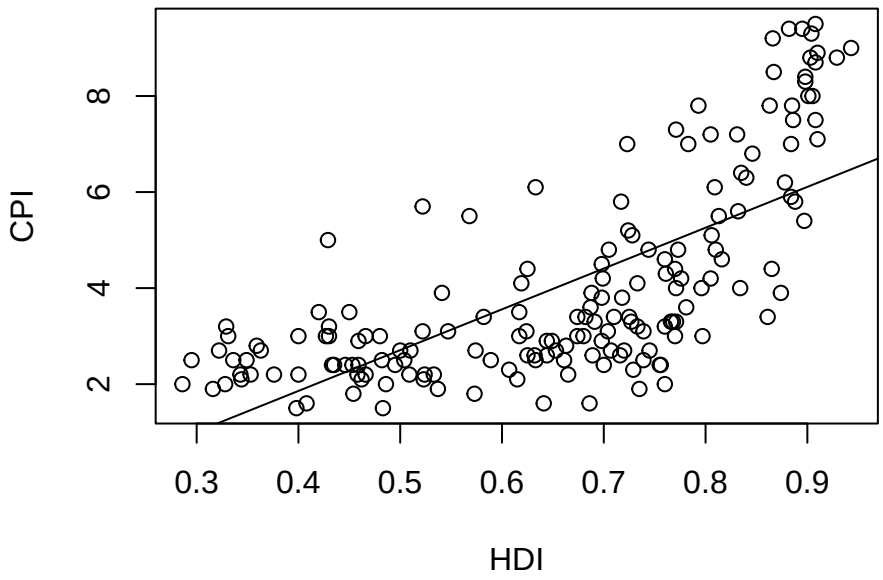
```
regression.model <- lm(formula = y ~ x,
                       data = dataset )
```

- `y` is the dependent variable, the output.
- `x` is the independent variable, the predictor

- `dataset` a dataset with the attributes `x` and `y`.

If we want to apply `lm()` to the `EconomistData` dataset, we use the following:

```
# execute only the fist time
# install.packages("readr")
require(readr)
library(readr)
CPIdata <- read_csv(
  here::here("data",
             "EconomistData.csv"))
attach(CPIdata)

# lm is a model formula
# ~ (read "is modeled as")
f1 <- lm(CPI ~ HDI)
f1
##
## Call:
## lm(formula = CPI ~ HDI)
##
## Coefficients:
## (Intercept)          HDI
##      -1.540        8.497

# We can plot the model
plot(HDI, CPI)
abline(f1)
```



The linear model is

$$\phi(x) = -1.540 + 8.497x$$

Which is the meaning of the coefficient 8.497 (the slope)? If x is increased in 1, then Y is increased in 1

Which is the meaning of the coefficient -1.540 (the intercept)? Value expected of y, if x has a value of 0.

3.2.2 Goodness of fit

Residuals

If the points are not colinear, there will be a difference between y_i and the predicted value $\overline{y} = \phi(x_i)$. This difference is called the **residual** $\epsilon_i = y_i - \overline{y}$.

Thus, $RSS(\vec{a}) = \|\epsilon\|_2^2 = \sum_{i=1}^{N} {\epsilon_i}^2$

We assume these residuals:

- They must have a zero average.
- If this is not the case, the bias must be measured.
- More points may need to be included in the data cloud.
- Errors must be uniformly distributed.
- We must wait for the residues to be uniformly distributed without patterns that detect anomalies.

- A first way would be to face the waste with the adjustment and the points should be around $y = 0$.

RSE

We define the **Residual Standard Error** as follows:

$$RSE = \sqrt{\frac{1}{n-2} RSS}$$

RSE is an estimate of the standard deviation of ϵ_i.

RSE is considered a measure of the lack of fit of the model to the data.

- If the predictions using the model are very good, we can conclude that the model fits the data very well.
- On the other hand, if the model doesn't fit the data well then the RSE may be quite large.

R-squared

R-squared (called the coefficient of determination) or fraction of variance explained is

$$R^2 = 1 - \frac{RSS}{TSS}$$

where

$$TSS = \sum_{i=1}^{n} (y_i - \overline{y})$$

- RSE is an *absolute measure* of lack of fit of the model.
- R^2 takes the form of a *proportion measure* (the proportion of variance explained), that is, the proportion of the variance in the outcome variable that can be accounted for by the predictor.

R^2 can be estimated with the correlation r between the input (X variable) and the output (Y variable) as follows: $R^2 = r^2$.

If R^2 is close to 1 indicates that a large proportion of the variability in the response has been explained by the regression. A number near 0 indicates that the linear model is wrong, or the inherent error is high, or both.

Note: The Pearson correlation is equivalent to running a linear regression model that uses only one predictor variable - the squared correlation r^2 is identical to the R^2 value for a linear regression with only a single predictor.

R-squared and Adjusted R-squared

R language returns Multiple R-squared and Adjusted R-squared.

If you add more predictors into the model, the R^2 value will increase (or at least it will be the same). In a regression model with K predictors, fit to a data set containing N observations, the adjusted R^2 is:

$$\text{adj. } R^2 = 1 - \left(\frac{\text{SS}_{res}}{\text{SS}_{tot}} \times \frac{N-1}{N-K-1} \right)$$

The adjusted R^2 value will only increase when the new variables improve the model performance.

3.2.3 Model quality with R

The main estimators can be found by executing the `summary()` function:

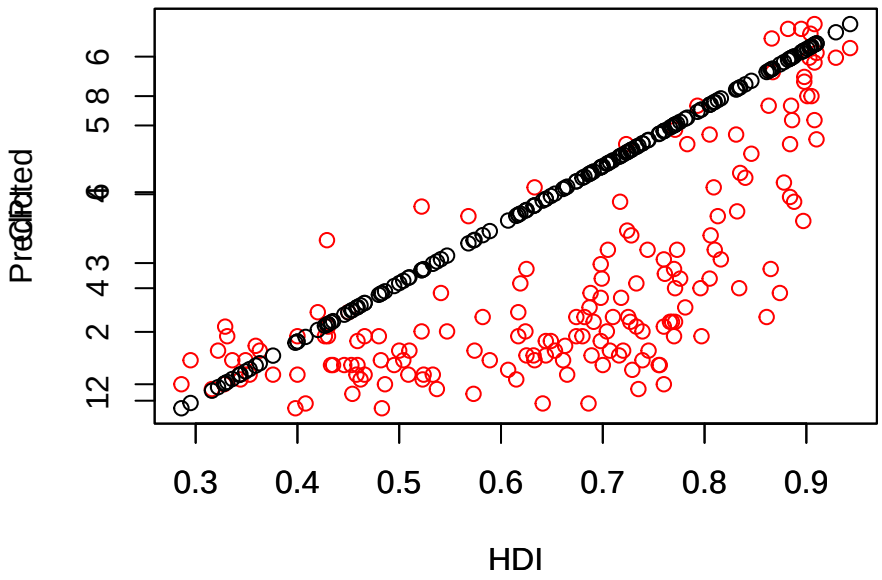
```
summary(f1)
##
## Call:
## lm(formula = CPI ~ HDI)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9180 -1.1872 -0.2029  1.0744  3.4453
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.5400     0.4453  -3.458 0.000686 ***
## HDI           8.4975     0.6539  12.994  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.506 on 171 degrees of freedom
## Multiple R-squared:  0.4968, Adjusted R-squared:  0.4939
## F-statistic: 168.9 on 1 and 171 DF,  p-value: < 2.2e-16
```

From those results, we can explain the model quality objectively.

We can also use plots to graph certain relationships between the residuals and the fitted values ($\phi(x_i)$) and check whether the residuals have 0 average, for instance.

We can check if the fitted curve (with the predicted values) fits the data cloud:

```
plot(HDI, CPI, col = "red")
par(new = TRUE)
plot(HDI,
     predict(f1,
             data.frame(x = HDI)),
     ylab = "Predicted")
```

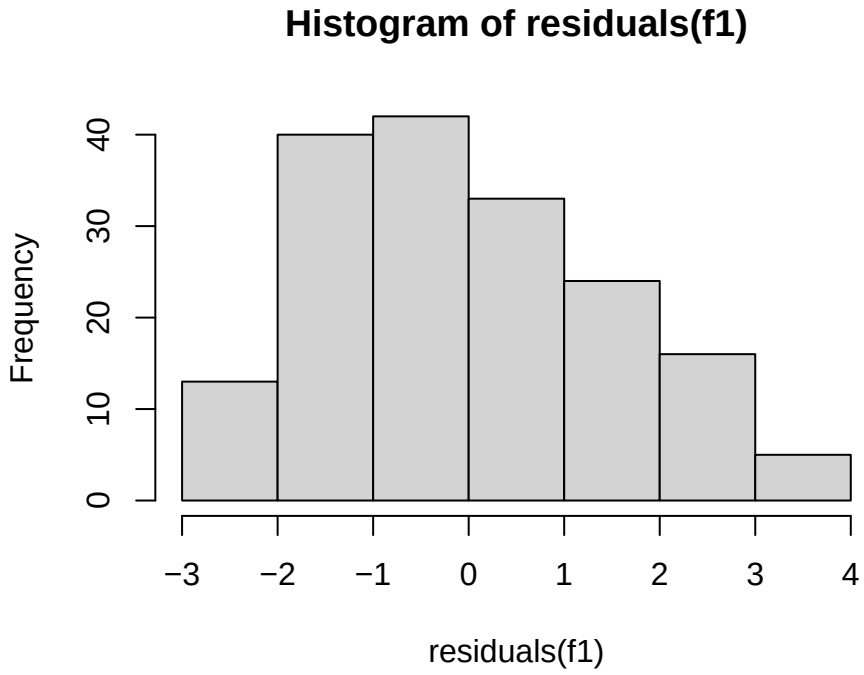


We get the fitted values with the `fitted()` function and the residuals with `residuals()` (easy to remember, aren't they?)

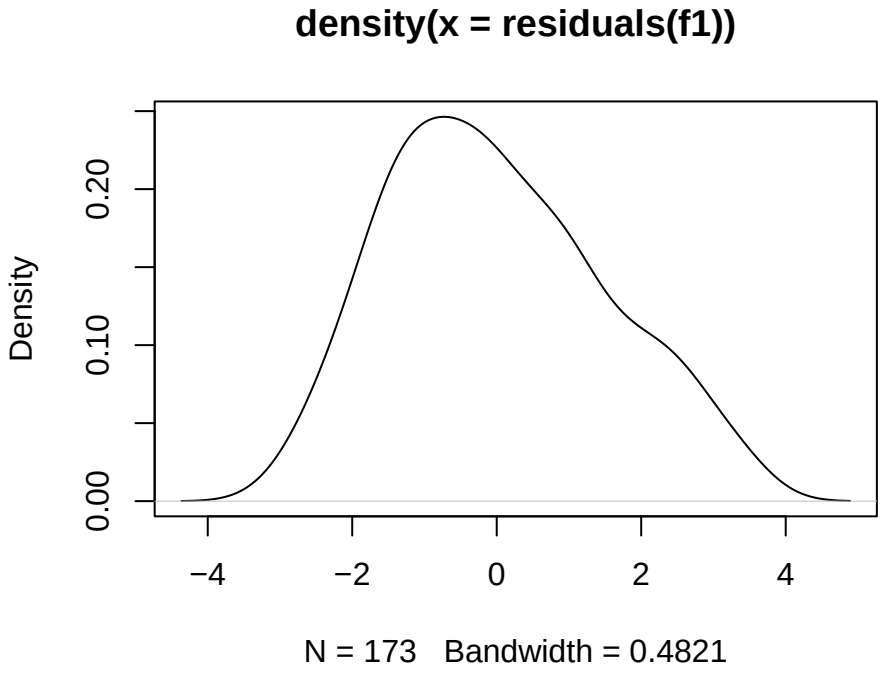
```
# Fitted values
head(fitted(f1))
##        1        2        3        4        5        6
## 1.841949 4.739580 4.391184 2.589725 5.232432 4.544139
# Residuals
head(residuals(f1))
##          1          2          3          4          5          6
## -0.3419488 -1.6395800 -1.4911844 -0.5897246 -2.2324322 -1.9441386
```

We can check the distribution of the residuals by using histograms or density plots:

```
hist(residuals(f1))
```




```
plot(density(residuals(f1)))
```



Exercise 3.1

- Import *ToyotaCorolla* dataset from the course website.
- Build linear models *m1, m2* using **Multiple Linear Regression**.
- Analyze the goodness of fit and model quality, explaining the results.

3.2.4 Are the coefficients meaningful?

We have obtained some coefficients:

```
coef(f1)
## (Intercept)         HDI
##   -1.540037    8.497452
```

but we do not know if these coefficients are indeed significant in the model or not.

To test whether they are meaningful in the model, it is used a statistical test.

This will explained later in the multiple linear regression.

3.2.5 How to generalize this linear regression

The linear model $\phi(x) = a_0 + a_1 x$ could be generalized in several ways.

We can also apply linear regression to more than 1 variable $x = \{x_1, x_2\}$.

For example, consider modeling temperature as a function of location $\{x_1, x_2\}$.

$$\phi(x) = a_0 + a_1 x_1 + a_2 x_2$$

or even,

$$\phi_2(x) = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1^2$$

Note: it is possible to model non-linear relationships. In general, a simple approach (we could use non-polynomial functions) is considered using polynomial basis functions, where the model has the form

$$\phi(x) = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n$$

3.3 Multiple Linear Regression

Given a set of variables $X_1, X_2, ..., X_m$, multiple linear regression computes the next model:

$$\phi(\vec{X}) = a_0 + a_1 X_1 + a_2 X_2 + ..., a_m X_m + \epsilon = Y$$

In this model a coefficient a_j is interpreted as the average effect on Y of a one unit increase in X_j, holding all other predictors fixed.

The ideal scenario is when the predictors are uncorrelated and in this case each coefficient can be estimated and tested separately.

Correlations amongst predictors cause problems because the variance of all coefficients tends to increase, sometimes dramatically.

Normally the independence is a utopia. Normally, predictors usually change together.

NOTE EVERYTHING that we have already explained for the case with one predictor is also valid for the remaining of this explanation: RSE, residuals, plots, etc.

3.3.1 Adding predictors with R

The formula in the `lm()` can be extended with more predictor variables in the right-hand side:

```
modelo1 <- lm(HDI.Rank ~ CPI + HDI)
```

3.3.2 Evaluating the Regression Coefficients

$$\phi(x) = a_0 + a_1 X_1 + a_2 X_2 + \ldots, a_m X_m$$

F-statistic

We establish the null hypothesis:

$$H_O : a_0 = a_1 = a_2 = \cdots = a_m = 0$$

Null hypothesis: there is no relationship between the predictors and the outcome

versus the alternative

$$H_a : \text{ at least one } a_i \text{ is non-zero}$$

Alternative hypothesis: the data are distributed as the regression model predicts

The hypothesis test is performed by means of **F-statistic**:

$$F = \frac{(TSS - RSS)/p}{RSS/(n-m-1)}$$

where

$$TSS = \sum (y_i - \overline{y})^2$$

When there is no relationship between the real valor and predictors, the F-statistic must take a value close to 1.

- If H_a is true the F-value must be greater than 1.
- If F-value is closer to 1, the answer to reject the hypothesis depends on values of n and m.
- Using the F-value, the p-value is computed.

p-value

Using p-value we can determine whether or not to reject H_0.

Evaluating the coefficients in R

```
summary(modelo1)
##
## Call:
## lm(formula = HDI.Rank ~ CPI + HDI)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -19.9326  -6.5788  -0.5189   5.9769  21.6061
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  293.9914     2.5014 117.529  < 2e-16 ***
## CPI           -2.9350     0.4153  -7.068 3.89e-11 ***
## HDI         -283.8765     5.0064 -56.703  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.178 on 170 degrees of freedom
## Multiple R-squared:  0.9782, Adjusted R-squared:  0.9779
## F-statistic:  3806 on 2 and 170 DF,  p-value: < 2.2e-16
```

- *p-value is essentially 0*: extremely strong evidence that at least one of the media is associated with output variable.
- `R` returns the p-value for each coefficient: These provide information about whether each individual predictor is related to the response, after adjusting for the other predictors.
- If, for a given coefficient, $Pr(>|t|) \approx 0$, the coefficient is significant - see *** in each row of the table.
- F-value is 3806 (must be greater than 1).

- p-value is close to 0.
- degrees of freedom: The number of independent pieces of information used in the estimation of a parameter.
- Multiple R-squared is used for evaluating how well your model fits the data. **97% of the variability in HDI.Rank is explained by CPI+HDI**.
- (Multiple R squared) measures the amount of variation in the response variable that can be explained by the predictor variables.
- Adjusted Rsquared adds penalties for the number of predictors in the model. Therefore it shows a balance between the most parsimonious model, and the best fitting model (ratio between the number of observations and the predictors). *If you have a large difference between your multiple and your adjusted Rsquared that indicates you may have overfit your model.*

3.3.3 Improving the models

Trying a more complicated model

- Polynomial regression.
- Polynomial orthogonal regression.
- Non linear regression.
- etc.

Classical polynomial approach

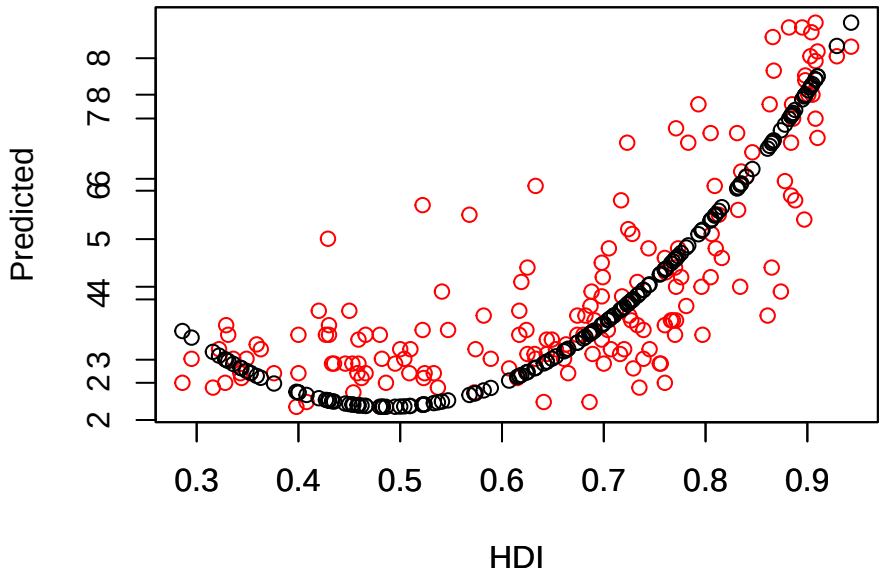
Trying a polynomial of degree n for $\phi(x)$.

We add to the model the term `I(X_i^n)`.

```
poli.2c <- lm(CPI ~ HDI + I(HDI^2))
poli.3c <- lm(CPI ~ HDI + I(HDI^2) + I(HDI^3))
poli.4c <- lm(CPI ~ HDI + I(HDI^2) + I(HDI^3)+ I(HDI^4))
poli.2c
##
## Call:
## lm(formula = CPI ~ HDI + I(HDI^2))
##
## Coefficients:
## (Intercept)          HDI     I(HDI^2)
##       9.552      -30.051       30.785
poli.3c
##
## Call:
## lm(formula = CPI ~ HDI + I(HDI^2) + I(HDI^3))
##
## Coefficients:
## (Intercept)          HDI     I(HDI^2)     I(HDI^3)
##      -7.116       59.605     -119.817       80.060
poli.4c
##
## Call:
## lm(formula = CPI ~ HDI + I(HDI^2) + I(HDI^3) + I(HDI^4))
##
## Coefficients:
## (Intercept)          HDI     I(HDI^2)     I(HDI^3)     I(HDI^4)
##    -0.08463      7.38593     18.62201    -75.89626     63.32330
```

We can check the goodness of fit for any of these models:

```
plot(HDI, CPI, col = "red", ylab = "")
par(new = TRUE)
plot(HDI,
     predict(poli.2c,
             data.frame(x = HDI)),
     ylab = "Predicted")
```



Orthogonal polynomial approach

We will use `poly(X_i, n)` to consider a new orthogonal polynomial of degree `n`.

```
poli.2o <- lm(CPI ~ poly(HDI, 2))
poli.3o <- lm(CPI ~ poly(HDI, 3))
poli.4o <- lm(CPI ~ poly(HDI, 4))
poli.2o
##
## Call:
## lm(formula = CPI ~ poly(HDI, 2))
##
## Coefficients:
##   (Intercept)  poly(HDI, 2)1  poly(HDI, 2)2
##         4.052         19.568         11.859
poli.3o
##
## Call:
## lm(formula = CPI ~ poly(HDI, 3))
##
## Coefficients:
##   (Intercept)  poly(HDI, 3)1  poly(HDI, 3)2  poly(HDI, 3)3
##         4.052         19.568         11.859          5.159
poli.4o
##
## Call:
## lm(formula = CPI ~ poly(HDI, 4))
##
## Coefficients:
##   (Intercept)  poly(HDI, 4)1  poly(HDI, 4)2  poly(HDI, 4)3  poly(HDI, 4)4
##        4.0520        19.5681        11.8590         5.1587         0.6239
```

Updating a pre-existing model

To update a model and add new predictors, we use the `update()` function.

```
p1 <- lm(CPI ~ HDI)
p2 <- update(p1, . ~ . + I(HDI^2))
p3 <- update(p2, . ~ . + I(HDI^3))
p4 <- update(p3, . ~ . + I(HDI^4))
p5 <- update(p4, . ~ . + I(HDI^5))
p6 <- update(p5, . ~ . + I(HDI^6))
p7 <- update(p6, . ~ . + I(HDI^7))
```

Exercise

- With the *ToyotaCorolla* dataset, build two polynomial and orthogonal models, and call them *m3* and *m4*.
- Repeat the analysis made for *m1* and *m2*. Also, analyze the significancy of the regression coefficients.

3.3.4 Comparing models with ANOVA

ANOVA (ANalysis Of VAriance) performs the Chi-square test to compare the models obtaining if the reduction in the residual sum of squares are statistically significant or not.

```
f1
##
## Call:
## lm(formula = CPI ~ HDI)
##
## Coefficients:
## (Intercept)          HDI
##      -1.540        8.497
poli.2c
##
## Call:
## lm(formula = CPI ~ HDI + I(HDI^2))
##
## Coefficients:
## (Intercept)          HDI     I(HDI^2)
##       9.552      -30.051       30.785
anova(f1, poli.2c) # We can add more models
## Analysis of Variance Table
##
## Model 1: CPI ~ HDI
## Model 2: CPI ~ HDI + I(HDI^2)
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    171 387.78
## 2    170 247.14  1    140.63 96.737 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The parameter $Pr(>F)$ is the probability that rejecting null hypothesis (the most complex model does not fit better than the simplest model) could be an error.

- Note: the second model must be an extended model of the first model.

Exercise

Compare all the models built for *ToyotaCorolla* using ANOVA and explain the results.

3.3.5 Interactions

Given a set of variables A, B, X, Y, Z, U we can test the following models, in addition to the combinations of these with the linear and polynomial models previously seen.

- $+$: combinations $(A + B)$
- $:$: interactions $(A : B)$. **Meaning of interactions**
- $*$ combinations and interactions $(A * B = A + B + A : B)$
- . (combinations of all variables)

Examples:

```
# f1(X)=aY + bZ
f1 <- lm( X ~ Y + Z )

# f2(X)=a(Y+Z) + bU
f2 <- lm( X ~ I(Y + Z) + U )

#f3(X)=aY + bY^2+cZ
f3 <- lm( X ~ I(Y) + I(Y^2) + I(Z) )
f3 <- lm( X ~   Y + I(Y^2) +     Z )

# f4(X): interactions
f4 <- lm( X ~ Y:Z )

# f5(X): interactions with the rest of variables
f5 <- lm( X ~ . )
```

3.4 Regression in R - Analysing *Boston* dataset

This dataset stores the *Housing Values in Suburbs of Boston.*

```
# package with a lot of datasets
library (MASS) # install if error
data(Boston) #Housing Values in Suburbs of Boston
names(Boston)
# ?Boston # See the descriptions of columns
```

3.4.1 Fit a simple linear regression

$$medv = \phi(lstat) = a_0 + a_1\, lstat$$

- **lstat**: lower status of the population (percent).
- **medv**: median value of owner-occupied homes in $1000s.

```
lm.fit <- lm(medv ~ lstat,
             data = Boston)
lm.fit
```

The model found is:

$$medv = \phi(lstat) = 34.55 - 0.95\,lstat$$

3.4.2 Analyzing the model

For more detailed information:

```
summary(lm.fit)
```

For more information we can use *names* with the linear model:

```
names(lm.fit)
# we ask for these information
coefficients(lm.fit)
```

Confidence interval for the coefficient estimates:

```
confint(lm.fit)
```

Visualizing the dataset+model:

```
attach(Boston)
# differents plots
plot(lstat, medv, col = "red ")
plot(lstat, medv, pch = 20)
plot(lstat, medv, pch = "+")
abline(lm.fit)
abline(lm.fit, lwd = 3)
abline(lm.fit, lwd = 3, col = "red")
```

3.4.3 Visualizing the linear model

```
oldpar <- par(mfrow = c(2, 2))
# divides the plotting region into a 2 × 2 grid of panels
plot(lm.fit)
par(oldpar)
```

3.4.4 Regression Diagnostics

- **residuals**
- **rstudent**
- **cooks.distance**

Search for information of these estimators, use with the dataset and explain.

Some plots regarding residuals, etc...

```
plot(predict(lm.fit),
     residuals(lm.fit))
```

```
plot(predict(lm.fit),
     rstudent(lm.fit))
```

- `hatvalues()`: On the basis of the residual plots, there is some evidence of non-linearity. Leverage statistics can be computed for any number of predictors using this function.

```
plot(hatvalues(lm.fit ))
which.max(hatvalues(lm.fit))
```

- `which.max()`: function identifies the index of the largest element of a vector. In this case, it tells us which observation has the largest leverage statistic.

3.4.5 Multiple regression

Fitting a model trying the regression with two input variables.

- **lstat**: lower status of the population (percent).
- **medv**: median value of owner-occupied homes in $1000s.
- **age**: age proportion of owner-occupied units built prior to 1940.

$$medv = \phi(lstat, age) = a_0 + a_1\, lstat + +a_2\, age$$

```
lm.fit2 <- lm(medv~lstat+age, data = Boston)
lm.fit2
summary(lm.fit2)
```

Data set contains 13 variables, how to select the best variables?:

- **Facing the output variable against the rest of variables**
- Analyzing the summary of the model.

```
lm.fit3 <- lm(medv ~ .,
              data = Boston)
summary(lm.fit3)
```

Interaction terms

```
lm.fit4 <- lm(medv ~ lstat * age,
              data = Boston)
summary(lm.fit4)
```

Non-linear transformations of the predictors

```
lm.fit5 <- lm(medv ~ lstat + I(lstat^2),
              data = Boston)
summary(lm.fit5)
```

The p-value associated with the quadratic term suggests that **lm.fit5** is an improved model.

Using ANOVA

Fist, we compare the model: linear (lm.fit) and quadratic (lm.fit5). If we represent these models, we could extract some important ideas:

```
#par(mfrow=c(2,2))
plot(lm.fit)
#par(mfrow=c(2,2))
plot(lm.fit5)
```

We can see a non-linearity in the relationship between *medv* and *lstat*.

Now, we will use `anova()` to perform a hypothesis test comparing the models.

```
anova(lm.fit, lm.fit5)
```

Interpretation: The null hypothesis says that the two models fit the data equally well, and the alternative hypothesis is that the extended model is superior.

See the F-statistic and the p-value associated.

This provides very clear evidence that the model containing the predictors *lstat* and $lstat^2$ is better than the model with just the predictor *lstat*.

3.5 Logistic Regression

3.5.1 Introduction

Logistic regression is an excellent tool for predicting the likelihood of something happening or not. It is used for classification tasks in quantitatives and qualitatives domains.

Firstly, we remind that linear regression corresponds to the family

$$p(y \mid x; \theta) = N(y; \theta^\top x; I)$$

To be used in the classification scenario it is necessary to define a different family of probability distributions. If we have two classes, class 0 and class 1, then we need only specify the probability of one of these classes.

The normal distribution over real-valued numbers that we used for linear regression is parametrized in terms of a mean. Any value we supply for this mean is valid. A distribution over a binary variable is slightly more complicated, because its mean must always be between 0 and 1.

For this task, logistic regression builds a linear model based on a transformed target variable.

As a summary, in Logistic Regression the output variable Y is binary, the input variables can be quantitative or qualitative and the distribution used is a binomial.

3.5.2 Binary response

As we have said, in linear regression we have:

$$y = f(X) = f(x_1, x_2, \ldots, x_k) = a_o + a_1 x_1 + \ldots + a_k x_k + \epsilon$$

The error ϵ follows a normal distribution with mean zero.

In logistic regression the response variable can be either true (success) or false (failure).

For a binary response, the conditional mean in the regression model becomes $P(y=1 \mid X)$.

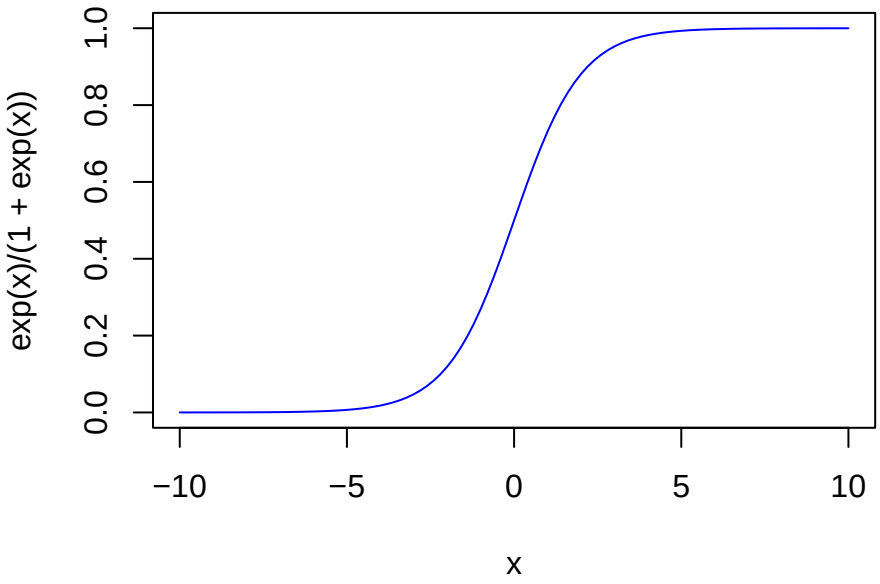
An acceptable function must always have values between 0 and 1. We want a response model where p is between 0 and 1.

$$p(X) = P(y=1 \mid X) = f(a_o + a_1 x_1 + ... + a_k x_k)$$

The logistic regression model links the predictor variables to probabilities through the equation

$$p = P(y=1 \mid X) = f(a_o + a_1 x_1 + ... + a_k x_k) = \frac{e^{a_o + a_1 x_1 + ... + a_k x_k}}{1 + e^{a_o + a_1 x_1 + ... + a_k x_k}}$$

This function leads to S-shaped curves between 0 and 1 such as those shown in the next graph.



Using algebra we have:

$$\log \frac{p}{1-p} = a_o + a_1 x_1 + ... + a_k x_k$$

The quantity $\frac{p}{(1-p)}$ relates the probability of success, p, to the probability of failure $(1-p)$.

- We refer to $\frac{p}{(1-p)}$ as the *odds of success*.
- The quantity $\log \frac{p}{1-p}$ is referred to as the *logit* of p and expresses the log odds of success.

Logistic regression, with its logit *link* $\log \frac{p}{1-p}$ modeled as a linear function of the predictor variables, is the most popular model for a binary outcome variable.

3.5.3 Logistic regression in R

- The R command `glm()` is used to fit logistic regression models.
- `glm()` is a very general routine (used for other kinds of regressions).

It can be used to fit generalized linear models (hence its name):

- listing the linear predictors,
- defining the error distribution,
- specifying a link function.

For logistic regression, the family of the error distribution is the **binomial**. The default, `family = binomial` [or `family = binomial(link = "logit")`] specifies the logistic regression model.

```
f_1 <- glm(formula, family = binomial, data)
```

Remember that

$$\text{if } f_1(X) = a_0 + a_1 x_1 + a_2 x_2 \ldots \text{ then } p = \frac{e^{f_1(X)}}{1 + e^{f_1(X)}}$$

3.5.4 Example

(From Logistic regression with glm)

We'll use the diamonds dataset:

```
# First time install
# install.package(yarr)
library(yarrr)
head(diamonds)
##   weight clarity color value
## 1   9.35    0.88     4 182.5
## 2  11.10    1.05     5 191.2
## 3   8.65    0.85     6 175.7
## 4  10.43    1.15     5 195.2
## 5  10.62    0.92     5 181.6
## 6  12.35    0.44     4 182.9
```

Steps:

- Create a binary variable called `value.g190` indicating whether the value of a diamond is greater than 190 or not.

```
diamonds$value.g190 <- diamonds$value > 190
```

- Use logistic regression with this new variable as output.

```
g.mod1 <- glm(formula = value.g190 ~ weight  + clarity + color,
              data = diamonds,
              family = binomial)
g.mod1
##
## Call:  glm(formula = value.g190 ~ weight + clarity + color, family = binomial,
##     data = diamonds)
```

```
##
## Coefficients:
## (Intercept)       weight      clarity        color
##    -18.8009       1.1251       9.2910      -0.3836
##
## Degrees of Freedom: 149 Total (i.e. Null);  146 Residual
## Null Deviance:       207.5
## Residual Deviance: 106.7     AIC: 114.7
summary(g.mod1)
##
## Call:
## glm(formula = value.g190 ~ weight + clarity + color, family = binomial,
##     data = diamonds)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -18.8009     3.4634  -5.428 5.69e-08 ***
## weight        1.1251     0.1968   5.716 1.09e-08 ***
## clarity       9.2910     1.9629   4.733 2.21e-06 ***
## color        -0.3836     0.2481  -1.547    0.122
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 207.52  on 149  degrees of freedom
## Residual deviance: 106.67  on 146  degrees of freedom
## AIC: 114.67
##
## Number of Fisher Scoring iterations: 6
```

$$V(x) = -18.8009 + 1.1251 \text{ weight } + 9.2910 \text{ clarity} - 0.3836 \text{ color}$$

- Are all the variables significant?

```
diamonds$fitted.value <- g.mod1$fitted.values
head(diamonds[c("weight", "clarity", "color", "value", "fitted.value")])
##   weight clarity color value fitted.value
## 1   9.35    0.88     4 182.5   0.16251772
## 2  11.10    1.05     5 191.2   0.82129665
## 3   8.65    0.85     6 175.7   0.03008442
## 4  10.43    1.15     5 195.2   0.84559083
## 5  10.62    0.92     5 181.6   0.44454833
## 6  12.35    0.44     4 182.9   0.08688269
```

To use this regression with a new diamond:

```
# We build a new data frame
# It may have more than 1 row
d.new <- data.frame(weight = 12,
```

```
                    clarity = 1.3,
                    color = 5)
# Evaluating in the model
# We obtain a prediction for each
# row of the "newdata" data frame
logit <- predict(g.mod1,
                 newdata = d.new)
```

This value is the *logit*-transformed probabilities. To compute the actual probability:

$$p = \frac{e^{V(x)}}{1 + e^{V(x)}} = \frac{1}{1 + e^{-V(x)}}$$

```
p <- 1 / (1 + exp(-logit))
p
##         1
## 0.9923129
```

This means that the new diamond will be valued over 190 with a probability of 99.23%.

4. Bayes

4.1 Bayesian probability theorems

If we recall the **conditional probability**:

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$$

we will be able to write:

$$\mathbb{P}(A \cap B) = \mathbb{P}(A|B) \cdot \mathbb{P}(B)$$

and, since $A \cap B = B \cap A$:

$$\mathbb{P}(A \cap B) = \mathbb{P}(B|A) \cdot \mathbb{P}(A)$$

This fact can be used to link $\mathbb{P}(A|B)$ and $\mathbb{P}(B|A)$:

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A) \cdot \mathbb{P}(A)}{\mathbb{P}(B)}$$

This result is known as **Bayes' theorem**.

4.2 Bayesian Inference

When modelling a phenomenon, one uses parameters that describe the behaviour of the system.

For example, in the toss of a coin a given number n of times, knowing the number of times it comes up *face* depends directly on a parameter which is the probability p that the coin comes up heads in an isolated toss. In fact, the probability distribution of the number of heads in those n flips is a binomial $B(n,p)$. Knowing p, we can estimate the average number of heads in n flips.

Continuing with the example, after carrying out n tosses, obtaining k heads, we ask ourselves if the coin is balanced, that is, if $p = 0.5$. In this case, we could approximate $\hat{p} = \frac{k}{n}$, and we would use a hypothesis test on the proportion of *successes* (heads) in n trials:

$$\begin{aligned} H_0 &: p = 0.5 \\ H_1 &: p \neq 0.5 \end{aligned}$$

This is the so-called **frequentist** view of statistics.

Bayesian inference has a slightly different purpose, as the parameter is considered as another variable:

- It is assumed that we have *a priori* knowledge about the distribution of the parameters of the phenomenon.
- We have data that provide a *credibility* or *likelihood* of the parameter.
- An *a posteriori* estimate of the probability distribution of the parameters is obtained, taking into account the data obtained.

All this is related by means of Bayes' Theorem above. Let θ be the parameter that models the phenomenon (for example, the probability p of getting heads when tossing a coin), and let D be the set of data obtained in an experiment of the phenomenon under study. Then:

$$\mathbb{P}(\theta|D) = \frac{\mathbb{P}(D|\theta) \cdot \mathbb{P}(\theta)}{\mathbb{P}(D)}$$

In short, it is usually written (since the denominator does not depend on θ):

$$\mathbb{P}(\theta|D) \propto \mathbb{P}(D|\theta) \cdot \mathbb{P}(\theta)$$

$\mathbb{P}(\theta|D)$ is the posterior distribution of the parameter (once we have taken the experimental data into account), $\mathbb{P}(D|theta)$ is the likelihood or credibility (what is the probability of having obtained the data we have obtained, knowing that the parameter has the value θ), and $\mathbb{P}(\theta)$ summarises the a priori knowledge about the parameter.

Bayesian inference can therefore be summarised as:

$$\text{Posterior} = \text{Likelihood} \cdot \text{Prior}$$

Given that the *Prior* can be prior knowledge or belief about the parameter distribution, then *Posterior* is the update of that knowledge or belief, taking into account the available evidence (which is represented by the term *Likelihood*).

4.3 Bayesian inference in R.

We show an example to demonstrate the connections between the three distributions described above. We are going to study the flips of a coin and what is the posterior probability of getting heads.

```
trials <- c('head', 'head', 'head', 'tail',
            'tail', 'head', 'head', 'tail',
            'tail', 'head')
N <- length(trials)
nHeads <- sum(trials == 'head')
nTails <- sum(trials == 'tail')
```

We do not know the value of θ, the probability of drawing heads, so we will consider 100 candidate values between 0 and 1:

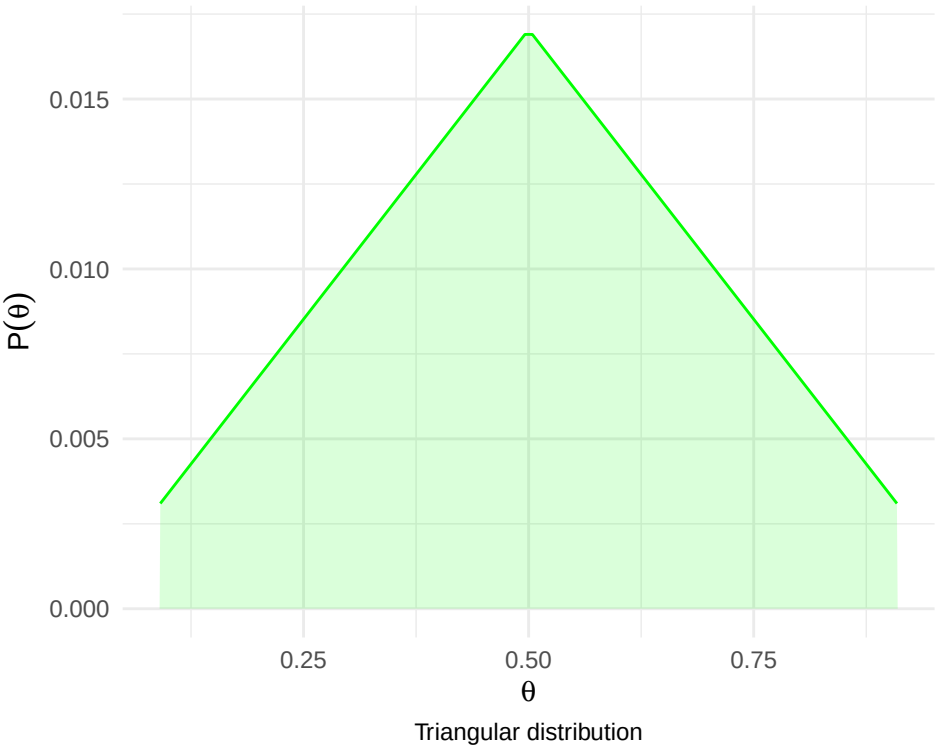
```
theta <- seq(from = 1 / (N + 1),
             to = N / (N + 1),
             length = 100)
```

4.3.1 Choice of *Prior*

Sometimes the choice of *prior* depends on some belief or prior knowledge about the parameter. It can be completely devoid of information (e.g., assume that any value of the parameter is equally likely), or provide quite a lot of information (consider that the value of the parameter, for example, is around 0.5).

Let's start by considering a *prior* with information: Let's assume that θ follows a triangular distribution, which takes the maximum (the mode) at $\theta = 0.5$:

```
# Triangular
pTheta <- pmin(theta, 1 - theta)
# We normalise so that the sum is 1
pTheta <- pTheta / sum(pTheta)
# We could represent it graphically in a simple way with:
# plot(theta, pTheta)
# lines(theta, pTheta)
```



4.3.2 Likelihood function

In the *frequentist* view of statistics, our maximum likelihood estimate of the parameter θ is the proportion of witnessed events over the total number of samples:

$$\hat{\theta} = \frac{\text{nHeads}}{N}$$

```
theta_hat <- nHeads / N
theta_hat
## [1] 0.6
```

In *Bayesian* inference, the likelihood function is used to redistribute the credibility of the a priori distribution according to the new evidence (the data).

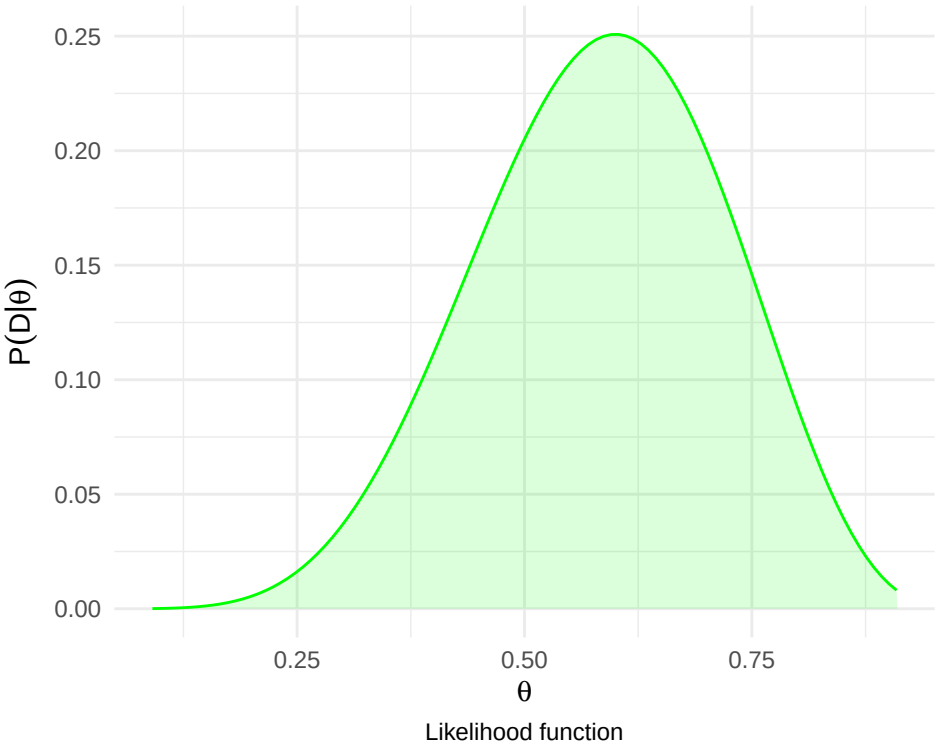
In this example, the likelihood function is given by a binomial distribution, which models how many successes occur among n attempts:

$$\mathbb{P}(\text{Roll } k \text{ heads out of } N \text{ trials}|\theta) = \binom{N}{k} \theta^k (1-\theta)^{N-k}$$

- N: number of times the event could occur, i.e. the number of attempts.
- k: number of times the event occurs.
- θ: the parameter of our model, which indicates the probability that in an isolated roll it comes up heads (Bernoulli experiment).

```
pDataGivenTheta <- choose(N, nHeads) * theta^nHeads * (1 - theta)^nTails
```

The likelihood has this approximate form:



4.3.3 Computation of the posterior distribution

The posterior distribution is computed from the prior using Bayes.

Recall that the denominator term in Bayes' theorem is $\mathbb{P}(D)$, the marginal probability of the data. By the total probability theorem, we can compute it this way:

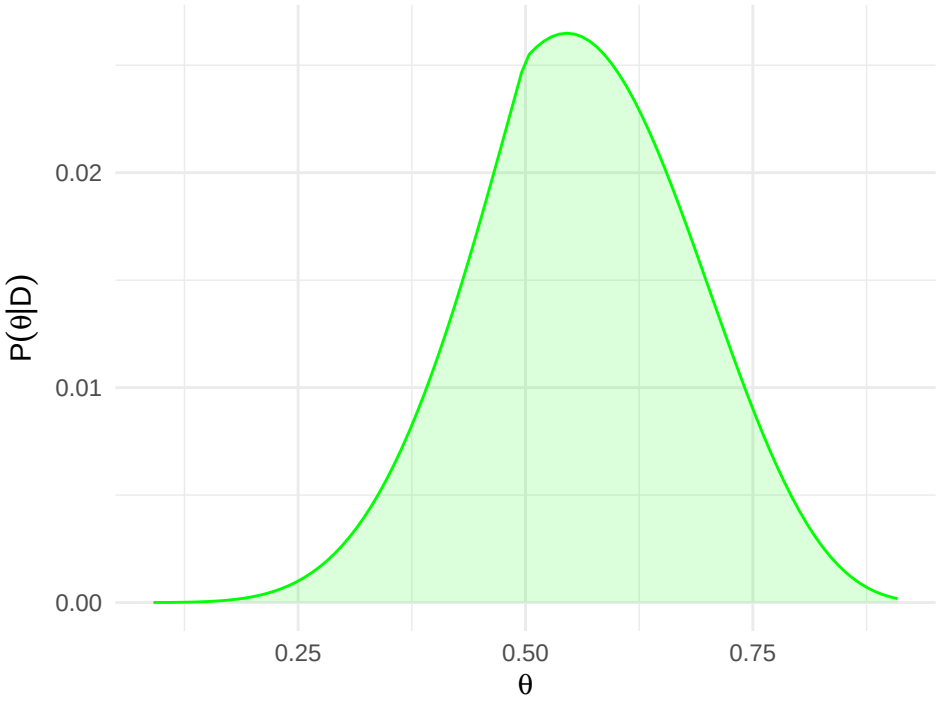
$$\mathbb{P}(D) = \int_0^1 \mathbb{P}(D|\theta) \cdot \mathbb{P}(\theta) \mathrm{d}\theta$$

Having discretised the possible values of θ, this integral is left as a sum:

$$\mathbb{P}(D) \approx \sum_i \mathbb{P}(D|\theta_i) \cdot \mathbb{P}(\theta_i)$$

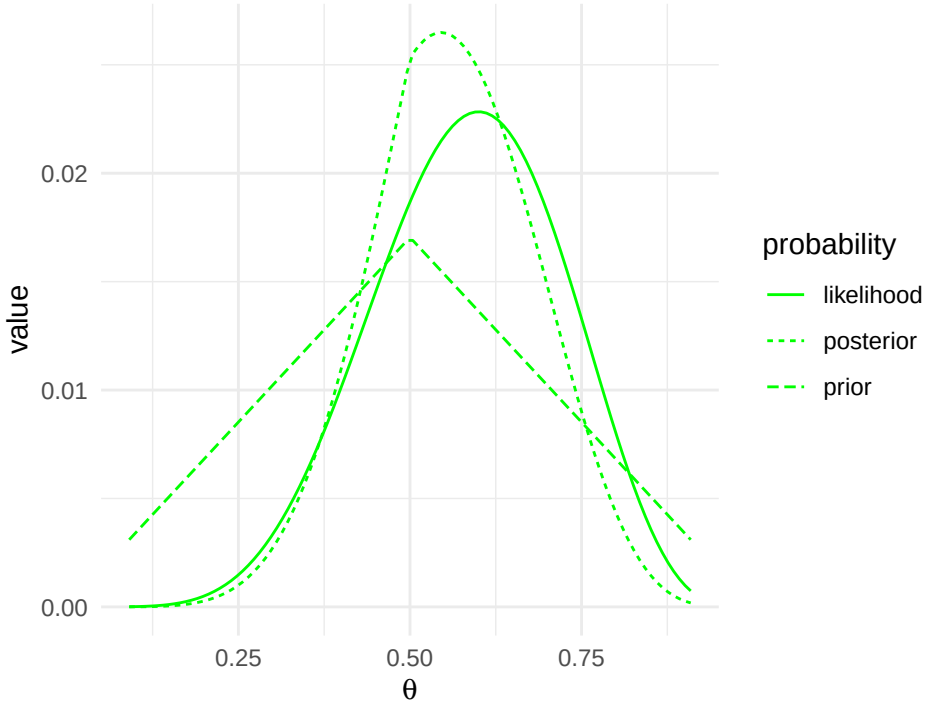
```
# Marginal probability of data
pData <- sum(pDataGivenTheta * pTheta)
# Bayes Theorem
pThetaGivenData <- pDataGivenTheta * pTheta / pData
```

Graphically:



A posteriori distribution of θ, using as prior distribution the triangular distribution

If we scale the likelihood function, we can see how the probability of the *prior* has been spread to resemble the new evidence (the *likelihood* or *likelihood*) and construct the *a posteriori* probability distribution of the θ parameter:



Prior, likelihood and posterior comparison

By setting it as a table, we can inspect those rows close to the one we are interested in, which is the one where the posterior is maximum:

```
df1 <- data.frame(theta = theta,
                  prior = pTheta,
                  likelihood = pDataGivenTheta,
                  posterior = pThetaGivenData)

# id stores the row with the maximum "posterior".
id <- which.max(df1$posterior)
kable(df1[(id - 3):(id + 3), ], row.names = FALSE)
```

theta	prior	likelihood	posterior
0.5206612	0.0163380	0.2208619	0.0261323
0.5289256	0.0160563	0.2264357	0.0263299
0.5371901	0.0157746	0.2315250	0.0264494
0.5454545	0.0154930	0.2360913	0.0264894
0.5537190	0.0152113	0.2400985	0.0264492
0.5619835	0.0149296	0.2435132	0.0263286
0.5702479	0.0146479	0.2463052	0.0261280

The **maximum a posteriori estimator** is a point estimator of θ. We could use the mode or mean, for example, of this distribution just calculated, to give an estimate of the parameter value:

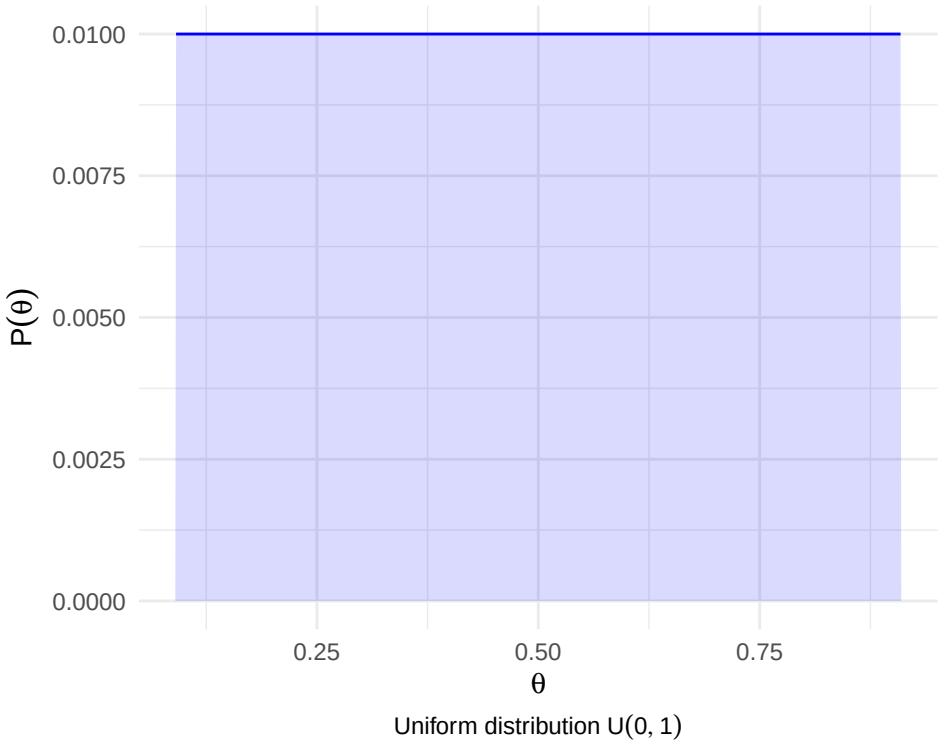
```
# Using mode:
theta_estimated_mode <- theta[which.max(pThetaGivenData)]
theta_estimated_mode
```

```
## [1] 0.5454545

# Using the mean or expectation:
theta_estimated_mean <- sum(theta * pThetaGivenData)
theta_estimated_mean
## [1] 0.5619734
```

4.3.4 Influence of the Prior.

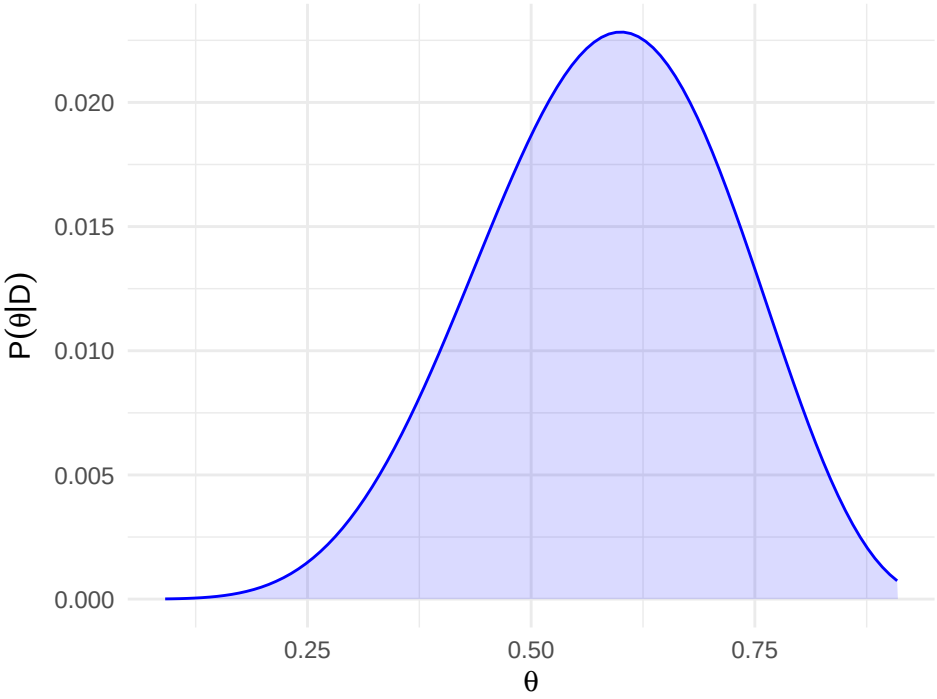
Let's start by using a *prior* that does not provide much information. We are talking about the fact that we do not know what the a priori value of θ might be and what we do is to assign the same probability to all possible values, using the uniform distribution:

```
# Uniform: No information
pTheta <- dunif(theta)
# We normalise so that the sum is 1
pTheta <- pTheta / sum(pTheta)
```



We repeat the above steps, resulting in the following a posteriori distribution:

```
# Marginal probability of data
pData <- sum(pDataGivenTheta * pTheta)
# Bayes Theorem
pThetaGivenData <- pDataGivenTheta * pTheta / pData
```



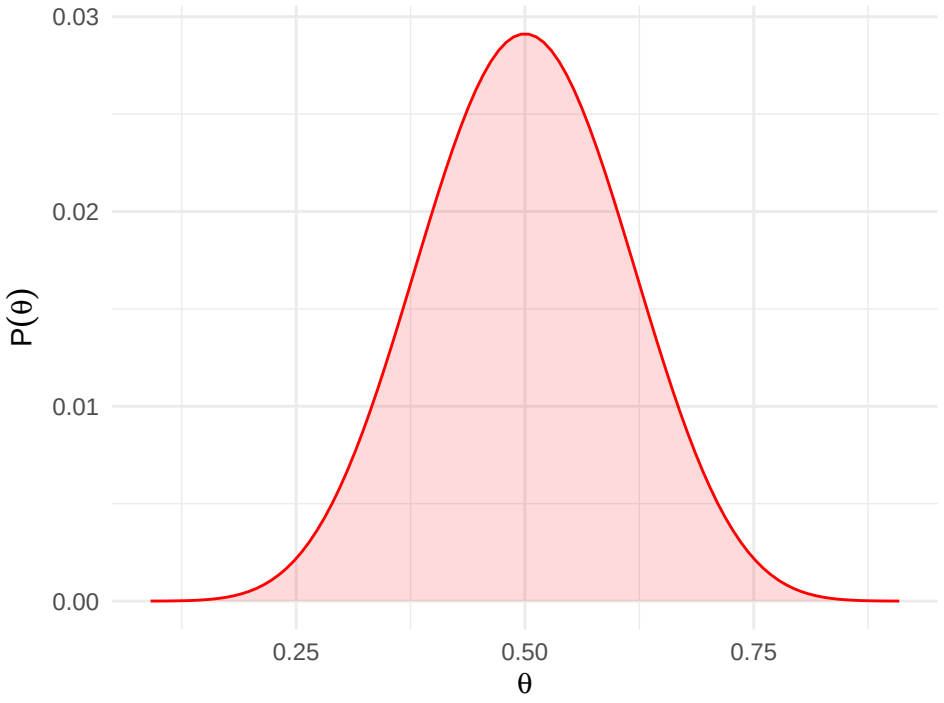
Posterior distribution of θ, using as a prior U(0, 1)

```
# Using mode:
theta_estimated_mode <- theta[which.max(pThetaGivenData)]
theta_estimated_mode
## [1] 0.6033058

# Using the mean or expectation:
theta_estimated_mean <- sum(theta * pThetaGivenData)
theta_estimated_mean
## [1] 0.5828379
```

Notice that, when the *prior* is not informative, the *posterior* resembles the frequentist approximation $\hat{\theta} = 0.6$.

Let us now use another a priori distribution, the beta of parameters 10, 10, which turns out to be very similar to a normal one, with mean 0.5:

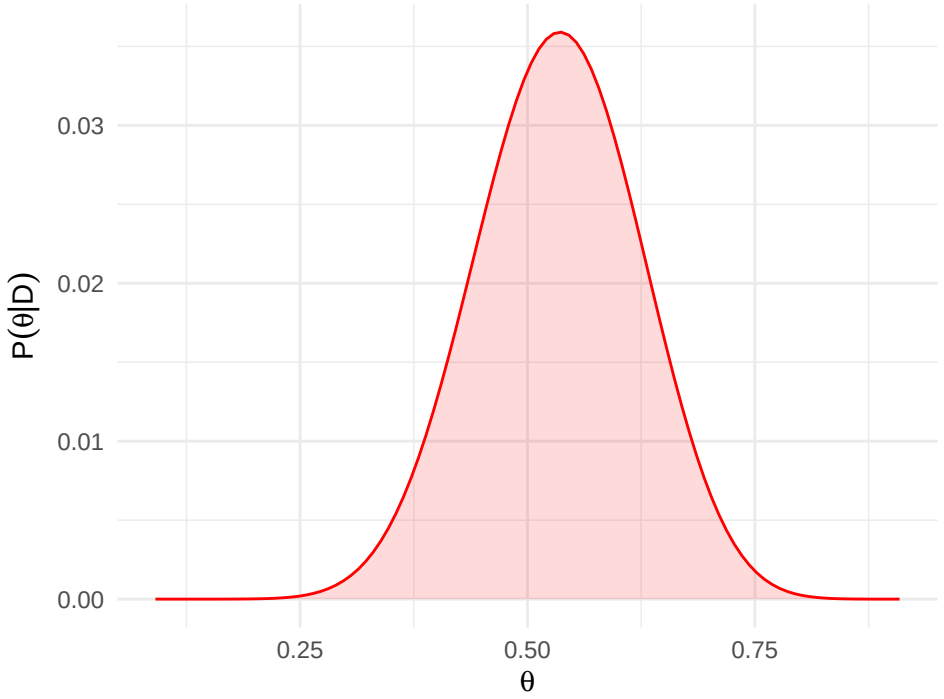
```
# Beta distribution with mean 0.5
pTheta <- dbeta(theta, 10, 10)
# We normalise so that the sum is 1
pTheta <- pTheta / sum(pTheta)
```



Distribution β(10, 10)

In this case

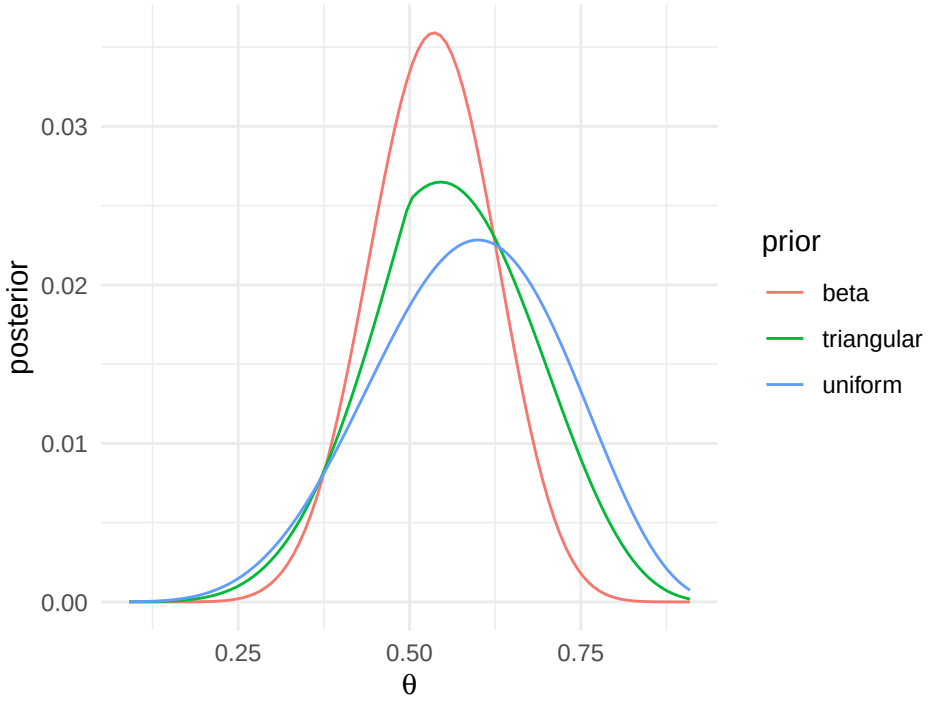
```
# Marginal probability of data
pData <- sum(pDataGivenTheta * pTheta)
# Bayes theorem
pThetaGivenData <- pDataGivenTheta * pTheta / pData
```



A posteriori distribution of θ, using as prior β(10, 10)

```
# Using mode:
theta_estimated_mode <- theta[which.max(pThetaGivenData)]
theta_estimated_mode
## [1] 0.5371901

# Using the mean:
theta_estimated_mean <- sum(theta * pThetaGivenData)
theta_estimated_mean
## [1] 0.5333333
```

We can see the a posteriori distributions according to the different *prior* in the following figure:



Comparison of posterior distributions for θ, according to the used prior

4.4 Interval estimation

A $100(1-\alpha)$ credibility interval is any interval (a,b) such that $\mathbb{P}(a<\theta<b|D)=1-\alpha$.

The shortest credibility interval is called the **interval of maximum a posteriori density**.

From R, we need the `HDInterval` package (installed with `install.packages("HDInterval")`), and it is used as follows:

```
library(HDInterval)
density <- data.frame(x = theta,
                      y = pThetaGivenData)
class(density) <- "density"

# By default, it considers 95% intervals
interval <- hdi(density, credMass = 0.95)
```

```
interval
##     lower     upper
## 0.3553719 0.7024793
## attr(,"credMass")
## [1] 0.95
## attr(,"height")
## [1] 0.005423003
```

This means that the interval of maximum a posteriori density (at 95%) is $(0.3553719, 0.7024793)$:

$$\mathbb{P}(0.3553719 < \theta < 0.7024793|D) = 0.95$$

We can check this in `R`, since

$$\mathbb{P}(a < \theta < b|D) = \int_a^b f(\theta|D)\mathrm{d}\theta$$

where $f(\theta|D)$ is the *a posteriori* density function. In the discrete case:

$$\mathbb{P}(a < \theta < b|D) \approx \sum_{a<\theta_i<b} \mathbb{P}(\theta_i|D)$$

So:

```
lower_interval <- interval["lower"]
upper_interval <- interval["upper"]
# Probability of theta in that interval
# It should be approximately 0.95.
p <- sum(pThetaGivenData[(theta > lower_interval) & (theta < upper_interval)])
p
## [1] 0.9426559
```

4.5 Bayesian hypothesis testing

Suppose we have two hypotheses H_0 (null) and H_1 (alternative) that we want to compare. Before conducting any experiment, we may have *beliefs* about which of these hypotheses is true. After conducting an experiment, Bayesian inference talks about the probability that the null hypothesis is true ($\mathbb{P}(H_0)$), and even the *a posteriori* probability of the hypotheses, which can be calculated using Bayes' Theorem:

$$\mathbb{P}(H_0|D) = \frac{\mathbb{P}(D|H_0)\mathbb{P}(H_0)}{\mathbb{P}(D)}$$

and the analogous for the alternative hypothesis H_1.

Operating with the posterior distributions, we arrive at:

$$\underbrace{\frac{\mathbb{P}(H_1|D)}{\mathbb{P}(H_0|D)}}_{\text{posterior odds}} = \underbrace{\frac{\mathbb{P}(D|H_1)}{\mathbb{P}(D|H_0)}}_{\text{Bayes factor}} \times \underbrace{\frac{\mathbb{P}(H_1)}{\mathbb{P}(H_0)}}_{\text{prior odds}}$$

The *prior odds ratio* tries to quantify what is believed a priori about the two hypotheses. If it is a high value, it indicates that there is a strong belief that the hypothesis is true. Analogously, the *posterior odds ratio* is interpreted on the a posteriori evidence.

The factor relating the two is the Bayes factor (BF), which plays a similar role to the p-value in frequentist hypothesis testing: it quantifies the strength of the evidence provided by the data.

Thus, in a Bayesian analysis, the Bayes factor is often referred to when testing. The reason why the BF is mentioned instead of the *posterior odds ratio* is that the latter is dependent on the *prior* chosen by each researcher or data analyst. To eliminate this bias, the BF is used.

Interpretation of the Bayes Factor

An intuitive explanation of the Bayes factor is as follows: if you run an experiment and, for example, $BF = 4$, this means that the data support that the alternative hypothesis is 4 times more likely than the null hypothesis.

In the following table, we show some interpretation of the BF about the evidence it implies in favour of the alternative hypothesis.

Bayes Factor	Interpretation
$0 < BF < 3$	Negligible evidence
$3 < BF < 20$	Positive (weak to moderate) evidence
$20 < BF < 150$	Strong Evidence
$BF > 150$	Very Strong Evidence

From `R`, the `BayesFactor` library can be used to calculate the Bayes factor from the data. In this case, the tests it performs for proportions are of the type:

$$H_0 : \theta = \theta_0$$
$$H_1 : \theta \neq \theta_0$$

To test whether $\theta = 0.5$, just do:

```
library(BayesFactor)
## Warning: package 'BayesFactor' was built under R version 4.3.1
## Loading required package: coda
## Warning: package 'coda' was built under R version 4.3.1
## Loading required package: Matrix
## Warning: package 'Matrix' was built under R version 4.3.1
## ************
## Welcome to BayesFactor 0.9.12-4.5. If you have questions, please contact Richard Morey
##
## Type BFManual() to open the manual.
## ************

# First argument: number of successes in the N attempts
# Second argument: the total number of attempts
# p is the value of theta in the null hypothesis
```

```
proportionBF(nHeads, N, p = 0.5)
## Bayes factor analysis
## --------------
## [1] Alt., p0=0.5, r=0.5 : 0.6908793 ±0%
##
## Against denominator:
##   Null, p = 0.5
## ---
## Bayes factor type: BFproportion, logistic
```

The higher the BF (here it is less than 1), the greater the evidence against the null hypothesis. In this case, there is little or no evidence that the alternative hypothesis ($\theta \neq 0.5$) is true.

4.6 References

Bayesian basics

Bayesian statistics - brms package

Film rating prediction - Frequentist vs. Bayesian model

Bayesian First Aid

Learning Bayes

Bayesian Thinking

Little Book of R for Bayesian Statistics

Bayesian networks

Bayesian Networks Tutorial

5. Time series

5.1 Introduction

5.1.1 Motivation

Applications - COVID - covid19forecast

Applications - from the book "Introductory Time Series with R, Paul S.P. Cowpertwait · Andrew V. Metcalfe, Springer"

1. Kyoto protocol - The arguments for reducing greenhouse gas emissions rely on a combination of science, economics, and **time series analysis**.

2. Change in the world's climate - `https://climatedata.imf.org/pages/climatechange-data`, `https://svs.gsfc.nasa.gov/12828/`.

Reports from [crudata.uea.ac.uk](https://crudata.uea.ac.uk/~timo/diag/tempdiag.htm?_ga=2.140802551.495728210.1698272062-1795576896.1698272062){.uri}

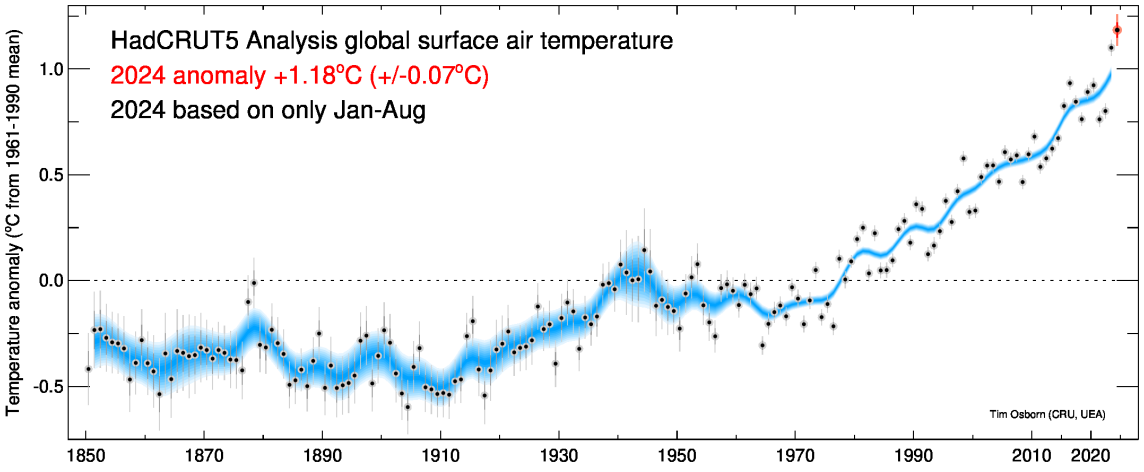


Figure 5.1.: Time series analysis

3. Singapore Airlines placed an initial order for twenty Boeing 787-9s and signed an order of intent to buy twenty-nine new Airbus planes, twenty A350s, and nine A380s (superjumbos). The airline's decision to expand its fleet relied on a combination of **time series analysis** of airline passenger trends and corporate plans for maintaining or increasing its market share.

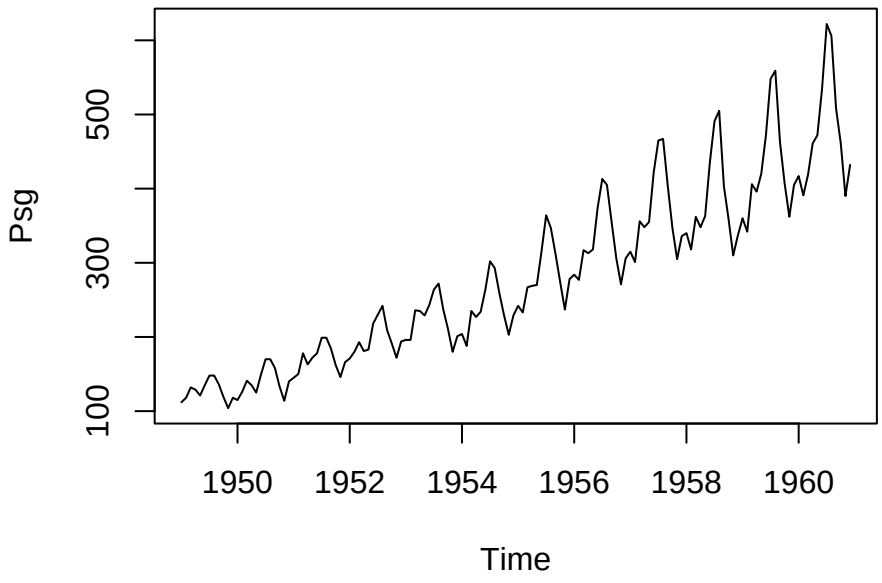
4. Gas suppliers - Variation about the average for the time of year depends on temperature and, to some extent, the wind speed. **Time series analysis** is used to forecast demand from the seasonal average with adjustments based on one-day-ahead weather forecasts.

5.1.2 An overview

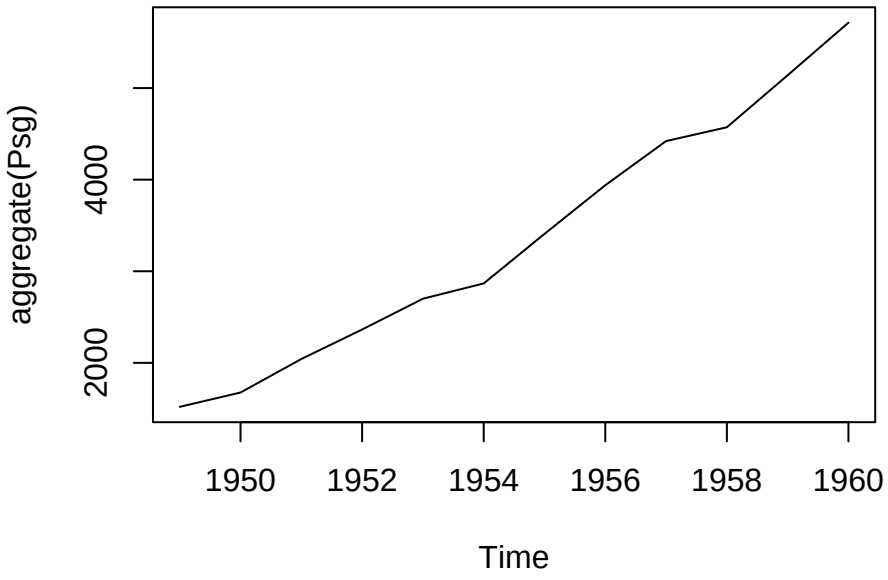
Basic methods

```
data(AirPassengers)
Psg <- AirPassengers
class(Psg)
## [1] "ts"
time(Psg)
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 1949 1949.000 1949.083 1949.167 1949.250 1949.333 1949.417 1949.500 1949.583
## 1950 1950.000 1950.083 1950.167 1950.250 1950.333 1950.417 1950.500 1950.583
## 1951 1951.000 1951.083 1951.167 1951.250 1951.333 1951.417 1951.500 1951.583
## 1952 1952.000 1952.083 1952.167 1952.250 1952.333 1952.417 1952.500 1952.583
## 1953 1953.000 1953.083 1953.167 1953.250 1953.333 1953.417 1953.500 1953.583
## 1954 1954.000 1954.083 1954.167 1954.250 1954.333 1954.417 1954.500 1954.583
## 1955 1955.000 1955.083 1955.167 1955.250 1955.333 1955.417 1955.500 1955.583
## 1956 1956.000 1956.083 1956.167 1956.250 1956.333 1956.417 1956.500 1956.583
## 1957 1957.000 1957.083 1957.167 1957.250 1957.333 1957.417 1957.500 1957.583
## 1958 1958.000 1958.083 1958.167 1958.250 1958.333 1958.417 1958.500 1958.583
## 1959 1959.000 1959.083 1959.167 1959.250 1959.333 1959.417 1959.500 1959.583
## 1960 1960.000 1960.083 1960.167 1960.250 1960.333 1960.417 1960.500 1960.583
```

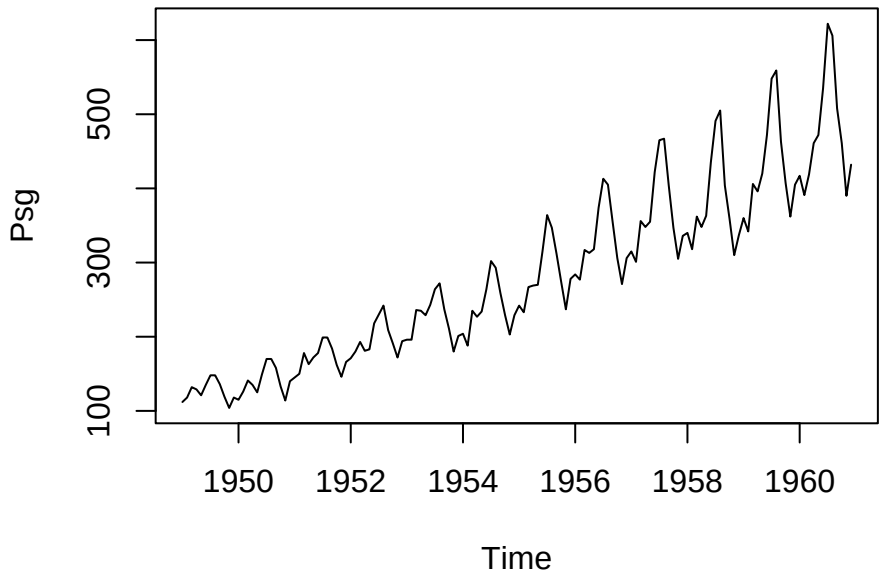
```
##           Sep      Oct      Nov      Dec
## 1949 1949.667 1949.750 1949.833 1949.917
## 1950 1950.667 1950.750 1950.833 1950.917
## 1951 1951.667 1951.750 1951.833 1951.917
## 1952 1952.667 1952.750 1952.833 1952.917
## 1953 1953.667 1953.750 1953.833 1953.917
## 1954 1954.667 1954.750 1954.833 1954.917
## 1955 1955.667 1955.750 1955.833 1955.917
## 1956 1956.667 1956.750 1956.833 1956.917
## 1957 1957.667 1957.750 1957.833 1957.917
## 1958 1958.667 1958.750 1958.833 1958.917
## 1959 1959.667 1959.750 1959.833 1959.917
## 1960 1960.667 1960.750 1960.833 1960.917
start(Psg)
## [1] 1949    1
end(Psg)
## [1] 1960   12
frequency(Psg)
## [1] 12
plot(Psg)
```




```
summary(Psg)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   104.0   180.0   265.5   280.3   360.5   622.0
aggregate(Psg)
## Time Series:
## Start = 1949
## End = 1960
## Frequency = 1
##  [1] 1520 1676 2042 2364 2700 2867 3408 3939 4421 4572 5140 5714
plot(aggregate(Psg))
```




```
ts.plot(Psg)
```



Advanced packages

Note: Install **TTR**, **forecast** packages

```
library(TTR)
library(forecast)
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

5.1.3 Forecasting

Forecasting is one of the principal applications in data science.

See Forecasting: Principles and Practice

Tell us what the future holds, so we may know that you are gods.

(Isaiah 41:23)

Forecasting is the set of techniques modelling how to predict the future as accurately as possible, given all of the information available, including historical data and any future data. These methods try to extrapolate trend and seasonal patterns

Applications of Forecasting

- Medicine, epidemiology,
- planning for the economy features of a country, financial institutions, pocily organizations,
- weather, global temperature changes
- scheduling in bussiness,
- marketing,
- predicting sales in a shop,
- finance and risk management,
- prediction of population in a country, in a region,
- seismic recordings
- planning th stock of product in a online shop,
- deciding whether to build another power generation plant,
- scheduling personal in a call centre depending of the number of calls,

- deciding capital investments,

See The statistical forecasting perspective

5.2 Time series analysis

The study of adjacent points using conventional statistical methods have problems.

- Conventional statistical methods are dependent on the assumption that these adjacent observations are independent and identically distributed.
- The temporal data appears temporal correlations and ad-hoc techniques have been developed to treat these data.

Basic steps

- Problem study
- Data collection
- Data analysis
- Model selection and fitting
- Model validation
- Forecasting model deployment
- Monitoring forecasting model performance

5.2.1 Time Series in R

Importing time series

To analyse your time series data we need to read it into R, and to plot the time series.

You can read data into R using **scan()**, which assumes that your data for successive time points is in a simple text file with one column.

The file *ts_k1.dat* contains data about average business sales of a product.

```
# Read from an URL
# Skip 1 line - comment
data.ts.k1 <- scan(
  here::here("data",
             "ts_k1.dat"),
  skip = 1)
head(data.ts.k1)
## [1] 60 43 67 50 56 42
```

The next step is to store the data in a time series object in R.

A time series is a list of numbers with temporal information stored as a ts object in R

There is a lot of packages and functions to deal with time series. Firstly, we will use **ts** function, which is used to create time-series objects.

The sintax of **ts** function is:

```
serie_1<-  ts(data, start, end, frequency)

# data - the  values in the time series

# start -  the first forecast observations in a time series

# end -  the last observation  in a time series

# frequency -  periods  (month, quarter, annual)

#     frequency=1 (by default) -  no periods or season in your data
#     frequency=12 - period is monthly
#     frequency=4  - period is quarterly
#Example

# data <- c(2, 8, 6, 10,
#           5, 14, 7, 14,
#           9, 23, 11, 29, 12)
# serie1 <- ts(data,start = c(2023,1),frequency = 12)

# plot(serie1)
# autoplot(serie1) - ggplot of the ts object
```

See ts-objects.

```
ts.k1 <- ts(data.ts.k1)
ts.k1
## Time Series:
## Start = 1
## End = 42
## Frequency = 1
##  [1] 60 43 67 50 56 42 50 65 68 43 65 34 47 34 49 41 13 35 53 56 16 43 69 59 48
```

```
## [26] 59 86 55 68 51 33 49 67 77 81 67 71 81 68 70 77 56
```

If the data set has values for periods less than one year, for example, monthly or quarterly, one can specify the number of times that data was collected per year by using the 'frequency' parameter in the **ts** function.

Dataset of unemployment in a city:

```
tsn1 <- scan(
    here::here("data",
               "ts_n1.dat"),
    skip = 1)
ts.n1 <- ts(tsn1, frequency = 12, start = c(1980, 1))
ts.n1
##         Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct
## 1980 26.663 23.598 26.931 24.740 25.806 24.364 24.477 23.901 23.175 23.227
## 1981 21.439 21.089 23.709 21.669 21.752 20.761 23.479 23.824 23.105 23.110
## 1982 21.937 20.035 23.590 21.672 22.222 22.123 23.950 23.504 22.238 23.142
## 1983 21.548 20.000 22.424 20.615 21.761 22.874 24.104 23.748 23.262 22.907
## 1984 22.604 20.894 24.677 23.673 25.320 23.583 24.671 24.454 24.122 24.252
## 1985 23.287 23.049 25.076 24.037 24.430 24.667 26.451 25.618 25.014 25.110
## 1986 23.798 22.270 24.775 22.646 23.988 24.737 26.276 25.816 25.210 25.199
## 1987 24.364 22.644 25.565 24.062 25.431 24.635 27.009 26.606 26.268 26.462
## 1988 24.657 23.304 26.982 26.199 27.210 26.122 26.706 26.878 26.152 26.379
## 1989 24.990 24.239 26.721 23.475 24.767 26.219 28.361 28.599 27.914 27.784
## 1990 26.217 24.218 27.914 26.975 28.527 27.139 28.982 28.169 28.056 29.136
## 1991 26.589 24.848 27.543 26.896 28.878 27.390 28.065 28.141 29.048 28.484
## 1992 27.132 24.924 28.963 26.589 27.931 28.009 29.229 28.759 28.405 27.945
## 1993 26.076 25.286 27.660 25.951 26.398 25.565 28.865 30.000 29.261 29.012
##         Nov    Dec
## 1980 21.672 21.870
## 1981 21.759 22.073
## 1982 21.059 21.573
## 1983 21.519 22.025
## 1984 22.084 22.991
## 1985 22.964 23.981
## 1986 23.162 24.707
## 1987 25.246 25.180
## 1988 24.712 25.688
## 1989 25.693 26.881
## 1990 26.291 26.987
## 1991 26.634 27.735
## 1992 25.912 26.619
## 1993 26.992 27.897
```

ts_f1.dat dataset contains monthly unemployment rates in a city.

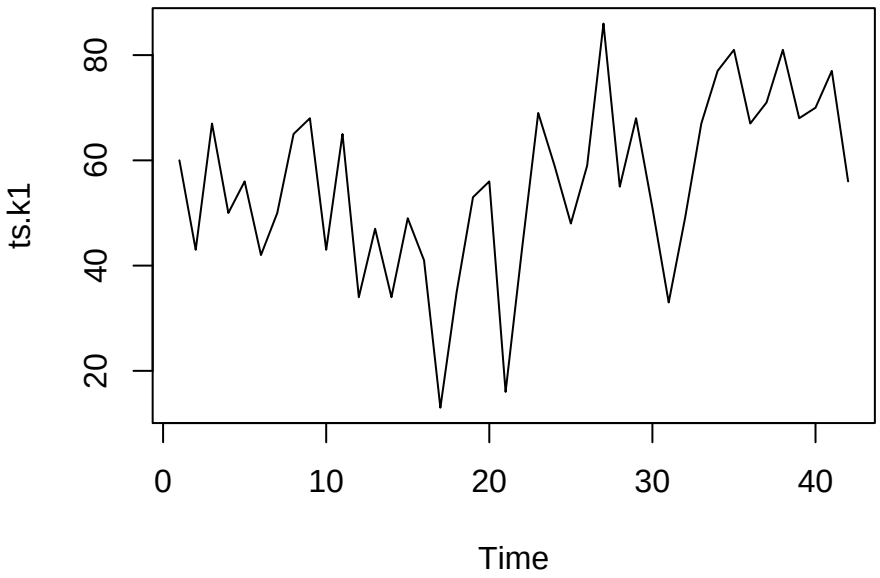
```
tsf1 <- scan(
    here::here("data",
               "ts_f1.dat"))
ts.f1 <- ts(tsf1, frequency = 12, start = c(1987, 1))
```

```
ts.f1
##             Jan       Feb       Mar       Apr       May       Jun       Jul
## 1987   1664.81   2397.53   2840.71   3547.29   3752.96   3714.74   4349.61
## 1988   2499.81   5198.24   7225.14   4806.03   5900.88   4951.34   6179.12
## 1989   4717.02   5702.63   9957.58   5304.78   6492.43   6630.80   7349.62
## 1990   5921.10   5814.58  12421.25   6369.77   7609.12   7224.75   8121.22
## 1991   4826.64   6470.23   9638.77   8821.17   8722.37  10209.48  11276.55
## 1992   7615.03   9849.69  14558.40  11587.33   9332.56  13082.09  16732.78
## 1993  10243.24  11266.88  21826.84  17357.33  15997.79  18601.53  26155.15
##             Aug       Sep       Oct       Nov       Dec
## 1987   3566.34   5021.82   6423.48   7600.60  19756.21
## 1988   4752.15   5496.43   5835.10  12600.08  28541.72
## 1989   8176.62   8573.17   9690.50  15151.84  34061.01
## 1990   7979.25   8093.06   8476.70  17914.66  30114.41
## 1991  12552.22  11637.39  13606.89  21822.11  45060.69
## 1992  19888.61  23933.38  25391.35  36024.80  80721.71
## 1993  28586.52  30505.41  30821.33  46634.38 104660.67
```

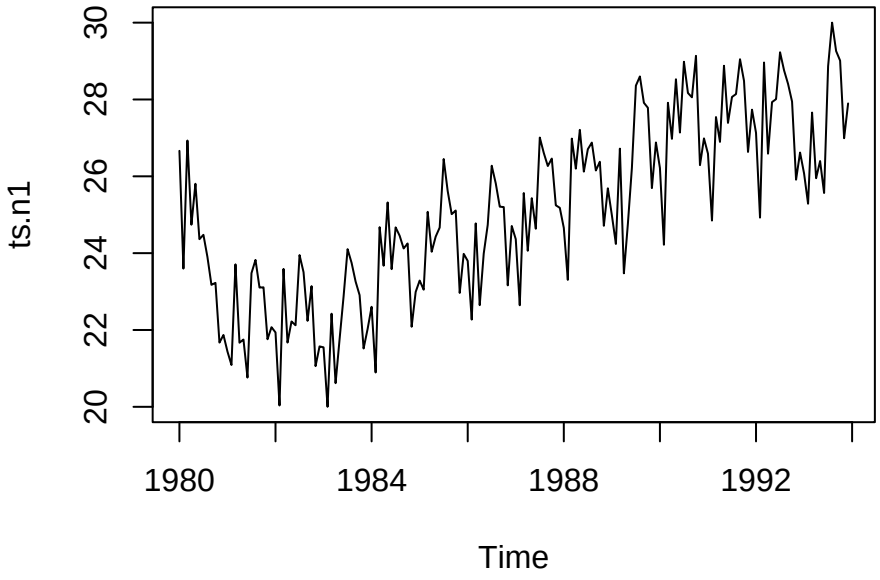
Displaying time series

Let us make a basic plot of the time series data (for the two first datasets):

```
plot.ts(ts.k1)
```



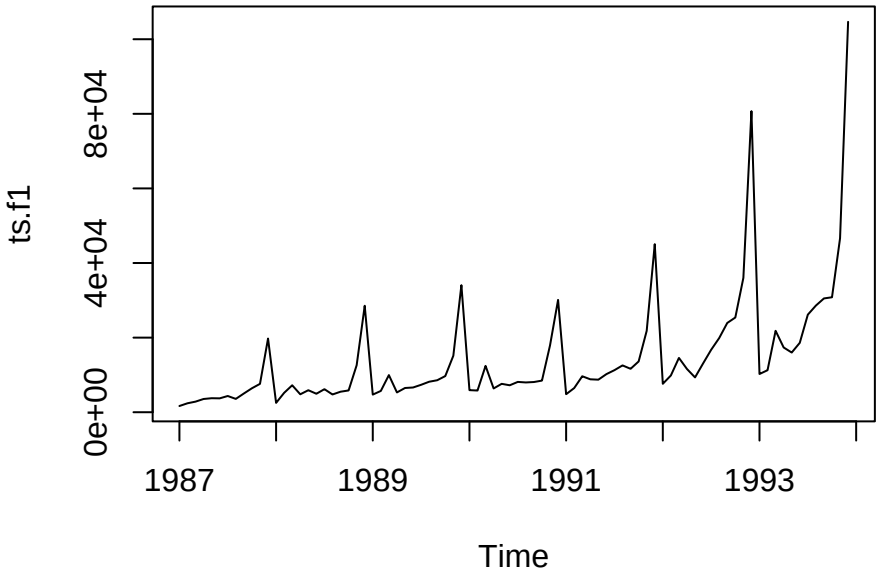

```
plot.ts(ts.n1)
```



In the second one, there seems to be seasonal variation in the number of unemployees per month: There is a peak every summer, and a trough every winter.

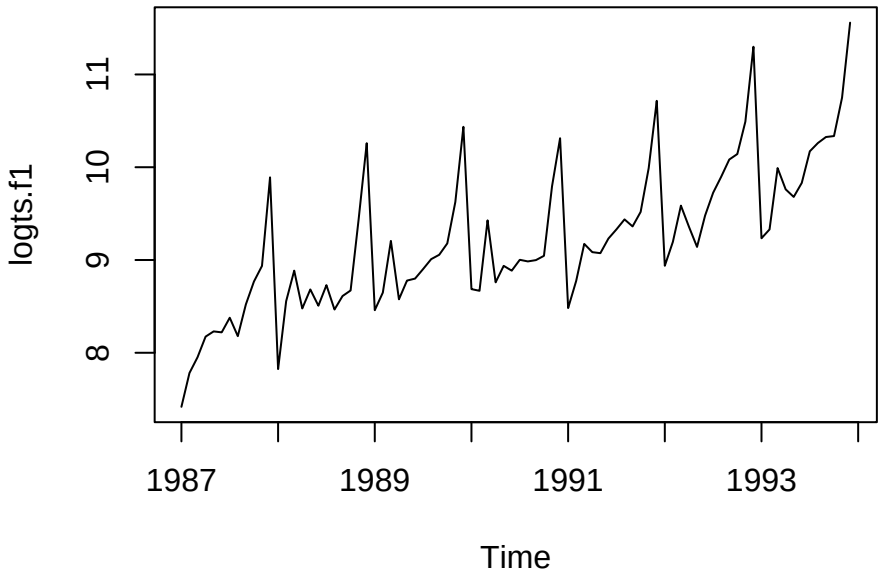
The plot for the third dataset is the following:

```
plot.ts(ts.f1)
```



The size of the seasonal fluctuations and random fluctuations seem to increase with the level of the time series. In these cases it is reasonable to transform the time series by calculating the natural logarithm of the original data.

```
logts.f1 <- log(ts.f1)
plot.ts(logts.f1)
```



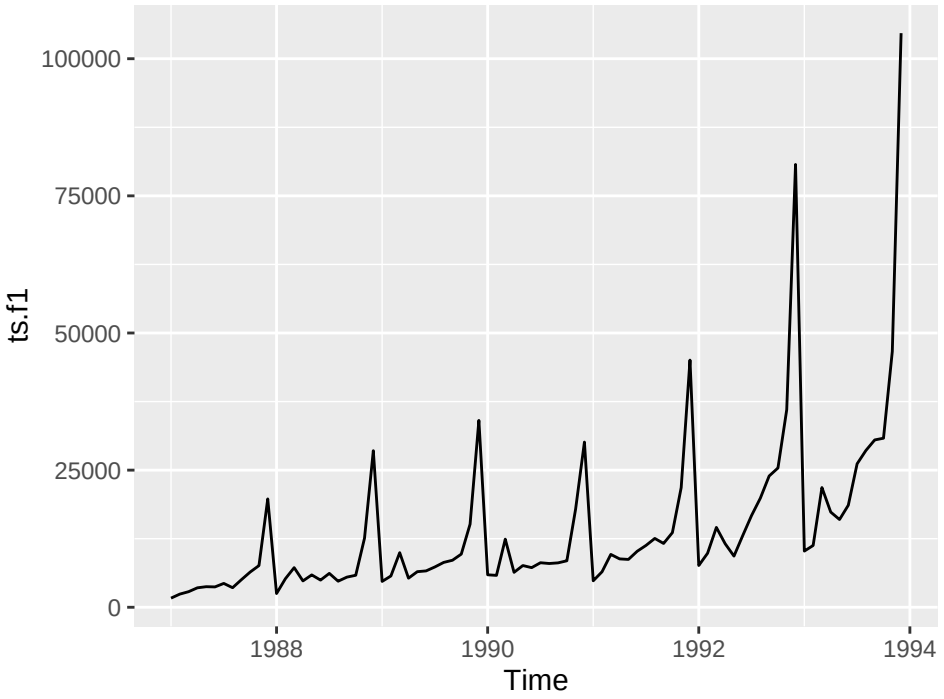
See time plots.

5.2.2 Time-dependent regression

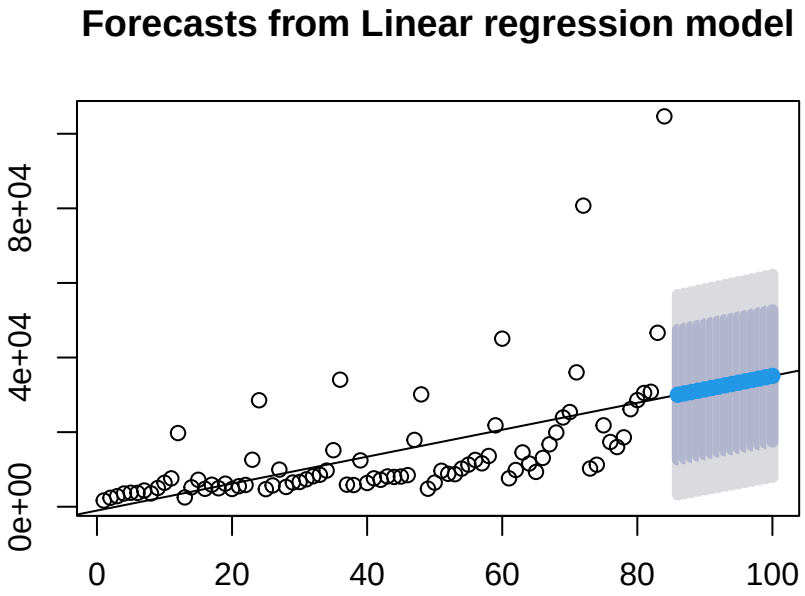
With one of the previously imported series (tsf1), we are fitting the model:

$$y = a + bx + cx^2$$

```
library(ggplot2)
library(forecast)
str(ts.f1)
##  Time-Series [1:84] from 1987 to 1994: 1665 2398 2841 3547 3753 ...
mydata <- data.frame(x=1:length(ts.f1), y=tsf1)
autoplot(ts.f1)
```




```
model <- lm( y ~ x , data=mydata)
summary(model)
##
## Call:
## lm(formula = y ~ x, data = mydata)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -15084  -7058  -1498   2394  75362
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1028.85    2892.21  -0.356    0.723
## x             361.05      59.11   6.108 3.22e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13140 on 82 degrees of freedom
## Multiple R-squared:  0.3127, Adjusted R-squared:  0.3043
## F-statistic: 37.31 on 1 and 82 DF,  p-value: 3.222e-08
prediction_model <- forecast::forecast(model, newdata = data.frame(x=86:100))
plot(prediction_model)
```

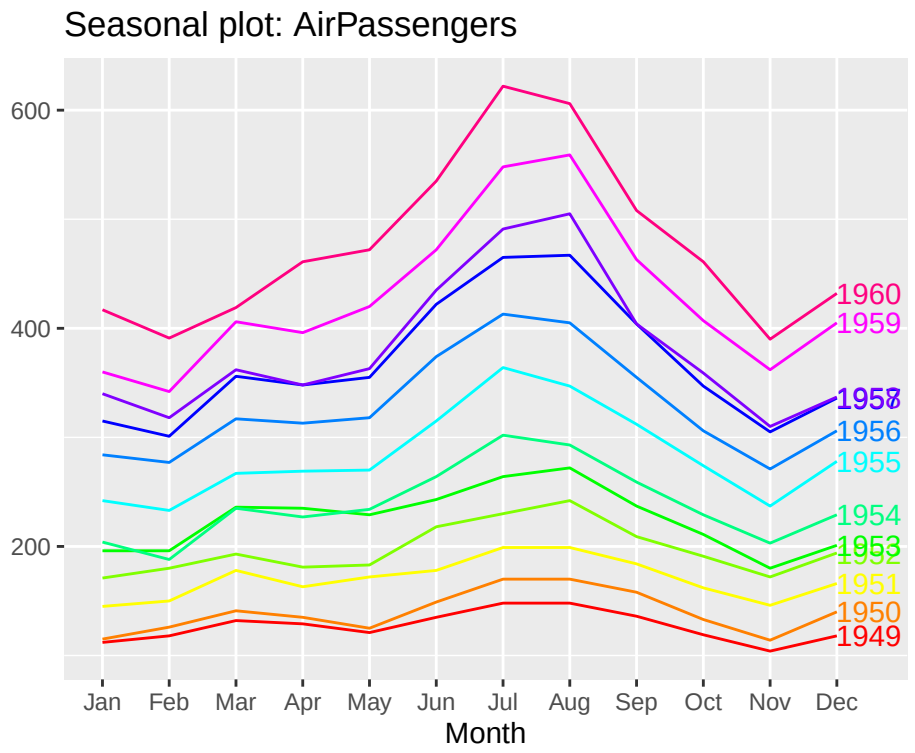


5.2.3 Time Series Components

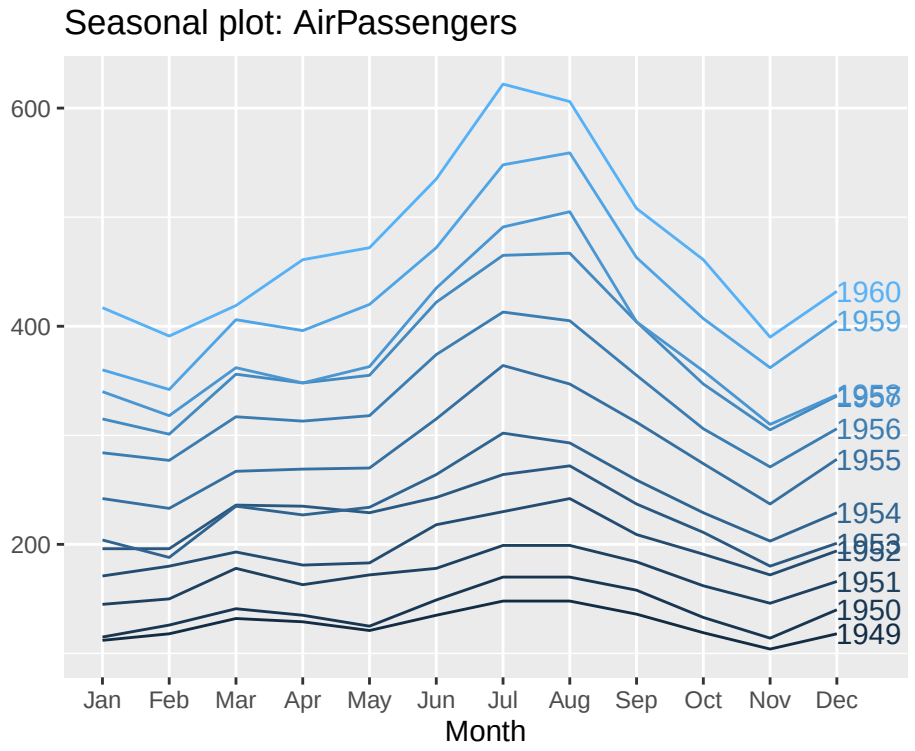
A time series has usually a trend component and an irregular component and, if it turns out to be a seasonal time series, a seasonal component as well.

See Time series patterns: Trend, Seasonal, Cyclic.

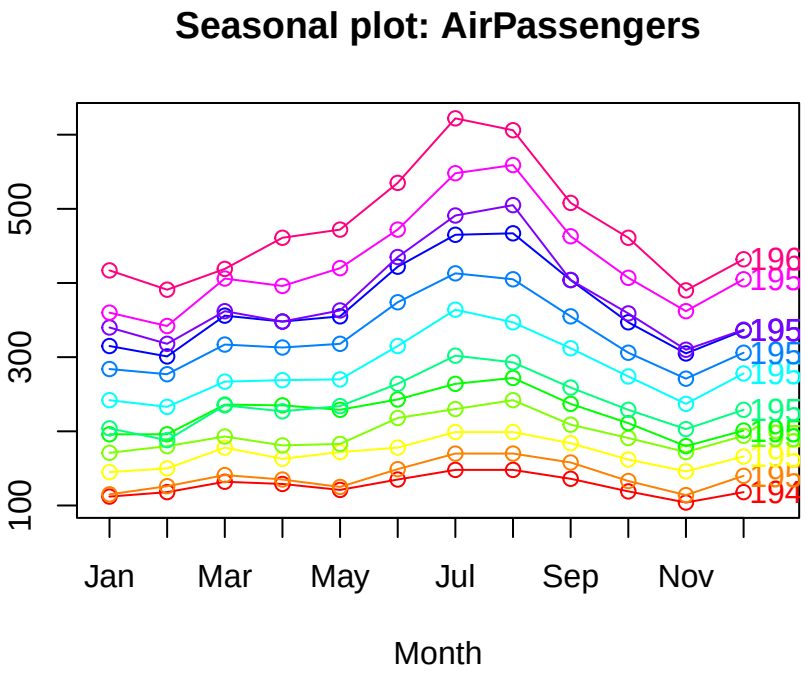
```
# https://www.rdocumentation.org/packages/forecast/versions/8.4/topics/ggseasonplot
library("forecast")
data("AirPassengers")
ggseasonplot(AirPassengers, col = rainbow(12), year.labels = TRUE)
```




```
ggseasonplot(AirPassengers, year.labels = TRUE, continuous = TRUE)
```

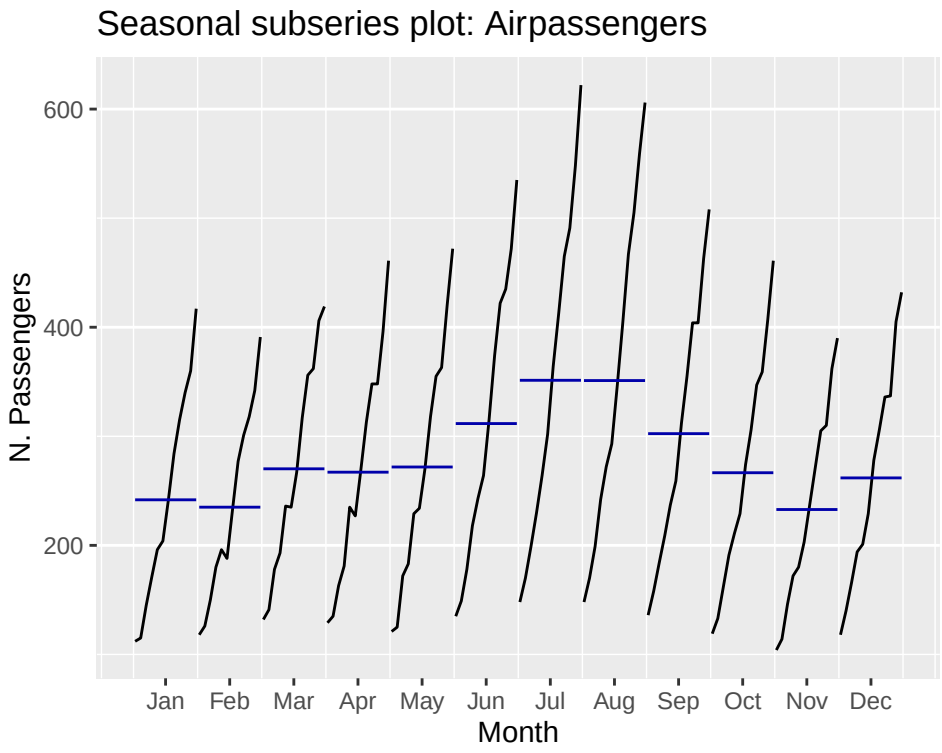



```
seasonplot(AirPassengers, col = rainbow(12), year.labels = TRUE)
```



Subseries plot

```
library(ggplot2)
ggsubseriesplot(AirPassengers) +
  ylab("N. Passengers") +
  ggtitle("Seasonal subseries plot: Airpassengers")
```



The mean is indicated by horizontal lines.

See Time series graphics: Scatterplots, lag plots, etc..

5.2.4 Other packages

- base R: times series class named ts. too restrictive, functiones associated limited
- zoo, xts packages: to represent time series
  - special structure for time series
  - many functions interestings

Exercise

- Make an .rmd with a mini-tutorial on the use of the zoo and xts packages, applying them to a time series (dataset with data from Spain) that you search on the web about electricity consumption, product consumption, etc.
- Hand in the homework of the time series topic.

Basic forecasting methods

We denote $\hat{y}_{T+h|T}$ the value of the variable y in the time $T+h$ based on $y_1,\dots,y_T$.

The residual will be computed by:

$$e_T = y_T - \hat{y}_T$$

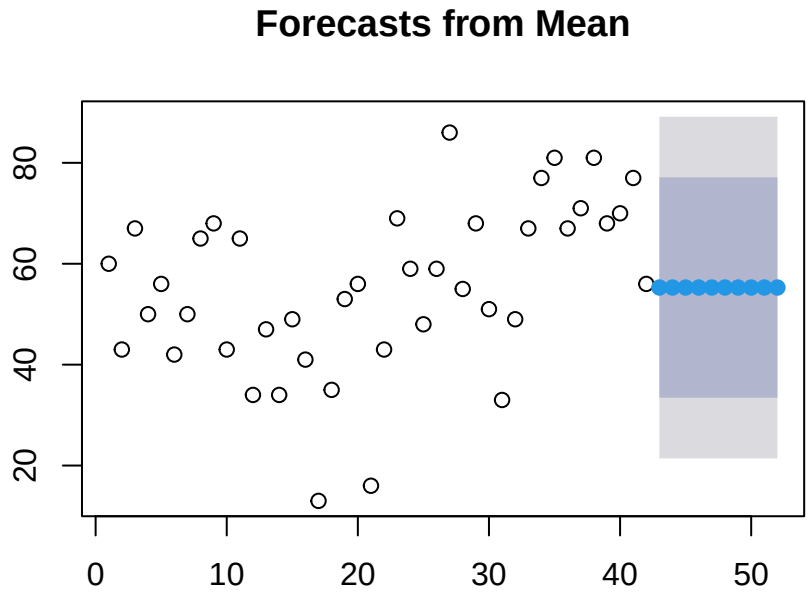
The forecast error is:

$$e_t(T) = y_t - \hat{y}_t(t-T)$$

Average method

$$\hat{y}_{T+h|T} = \frac{(y_1,\dots,y_T)}{T}$$

```
data.ts.k1 <- scan(
         here::here("data",
                    "ts_k1.dat"),
skip = 1)
mdata.ts.k1 <- meanf(data.ts.k1, 10)
plot(mdata.ts.k1)
```

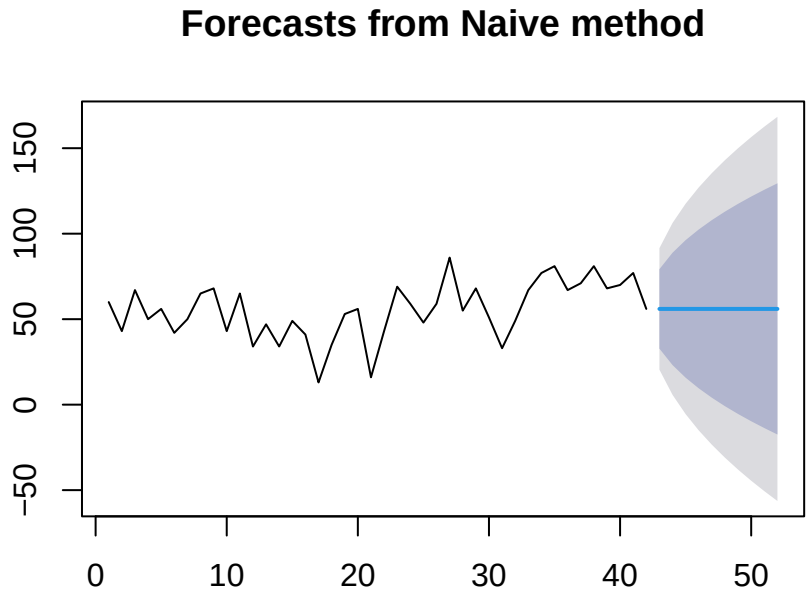


5.2.5 Naïve method

The prediction will be the last Y_T.

$$\hat{y}_{T+h|T} = y_T$$

```
data.ts.k1 <- scan(
  here::here("data",
  "ts_k1.dat"),
skip = 1)
h <- 10
mdata.ts.k1 <- naive(data.ts.k1, h)
plot(mdata.ts.k1)
```



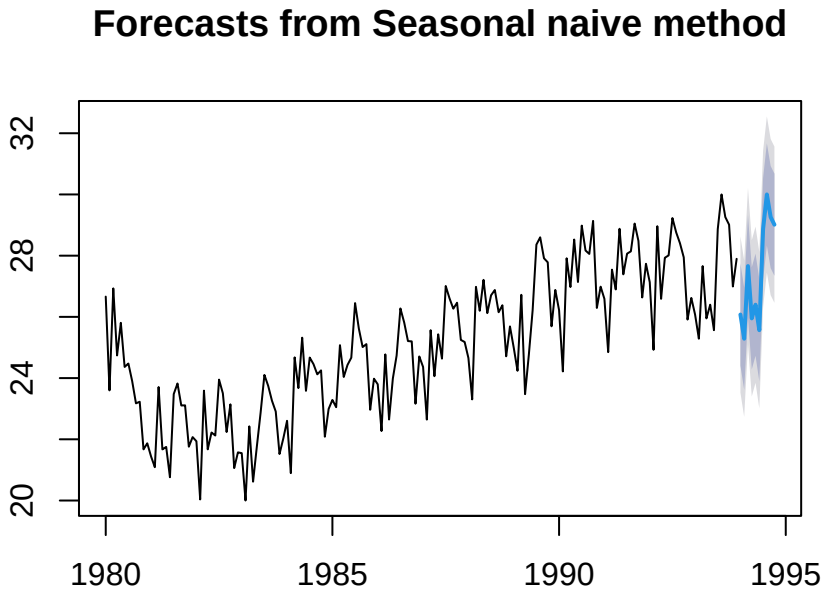
5.2.6 Seasonal naïve method

The prediction will be the same that the value of the variable in the last year in the same time.

$$\hat{y}_{T+h|T} = y_{T+h-m(k+1)}$$

(where m is the season in the year and k is the truncated value of $(h-1)/m$)

```
tsn1 <- scan(
  here::here("data",
  "ts_n1.dat"),
skip = 1)
ts.n1 <- ts(tsn1, frequency = 12, start = c(1980, 1))
h <- 10
mdata.ts.k1 <- snaive(ts.n1, h)
plot(mdata.ts.k1)
```

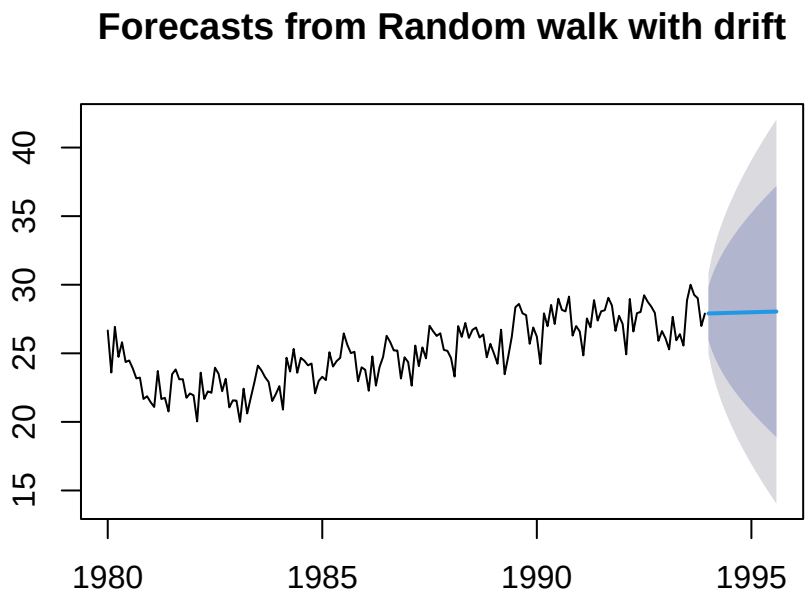


5.2.7 Drift method

It is based on naïve method where the amount of change in time (the drift) is set as the average change in the historical data:

$$\hat{y}_{T+h|T} = y_T + \frac{h}{T-1}\sum_{t=2}^{T}(y_t - t_{t-1}) = y_T + h\frac{y_T - y_1}{T-1}$$

```
h <- 20
mdata.ts.k1 <- rwf(ts.n1, h, drift = TRUE)
plot(mdata.ts.k1)
```



See all the methods in the same picture - Forecasts for quarterly beer production.

Note: It is necessary the package **fpp2** to execute the code in this page.

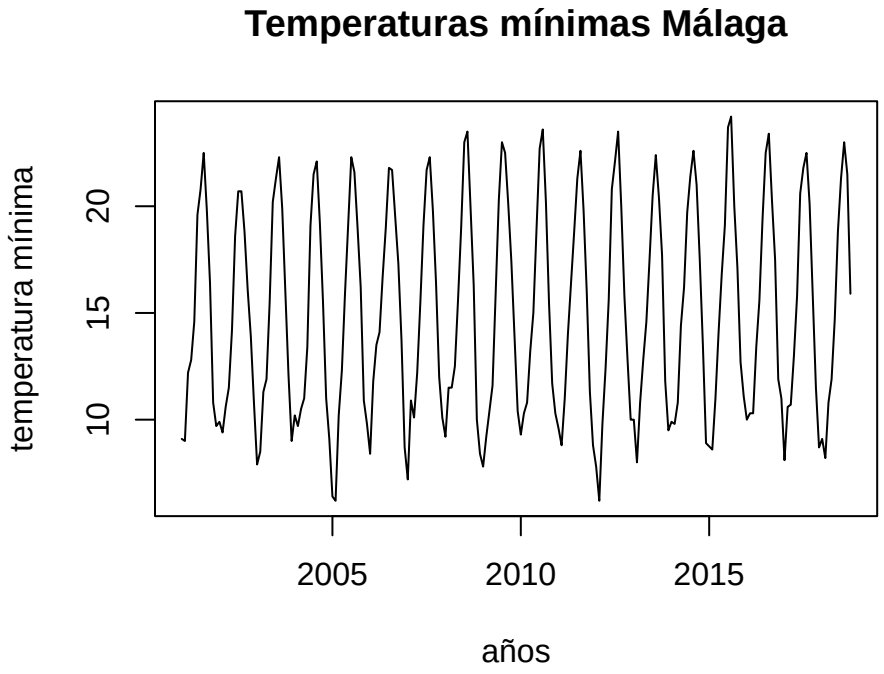
5.2.8 Example: Temperature forecast in Malaga - Basic forecasting methods

To analyse the evolution of minimum temperatures since 2001 in Malaga.

First of all we obtain a time-series object with the function **ts**, as our data are divided into months, we set the parameter *frecuency* to 12 and we indicate that we start in January 2001. After this, we draw the object obtained:

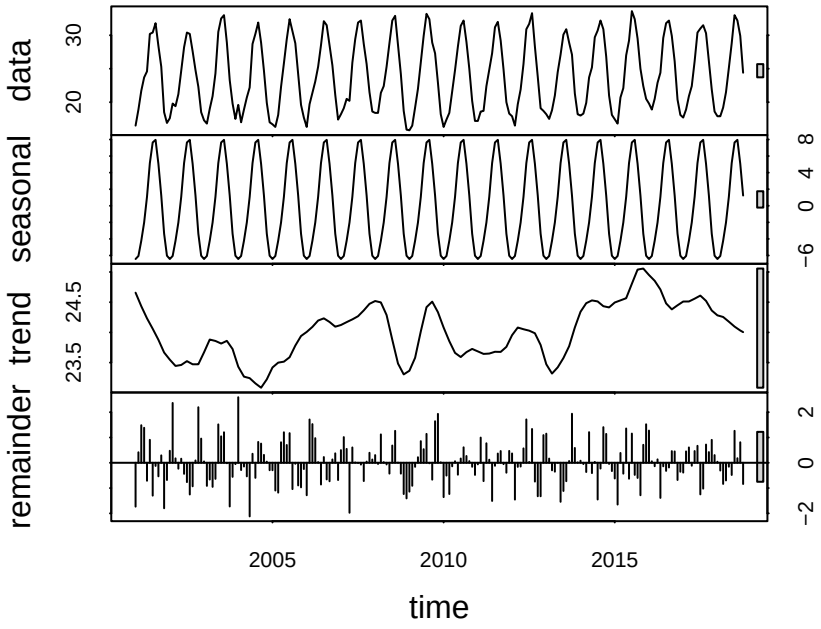
```
library('ggplot2')
library('TTR')
library('forecast')
library('tseries')
```

```
library(readr)
## Warning: package 'readr' was built under R version 4.3.1
temperaturas <- read_csv(
  here::here("data",
  "temp.csv"))
## Rows: 214 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): fecha, indicativo
## dbl (4): X1, X1_1, tm_max, tm_min
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
ts.p1 <- ts(temperaturas$tm_min, start = c(2001, 1), frequency = 12)
plot.ts(ts.p1, xlab = "años", ylab = "temperatura mínima", main = "Temperaturas mínimas M
```



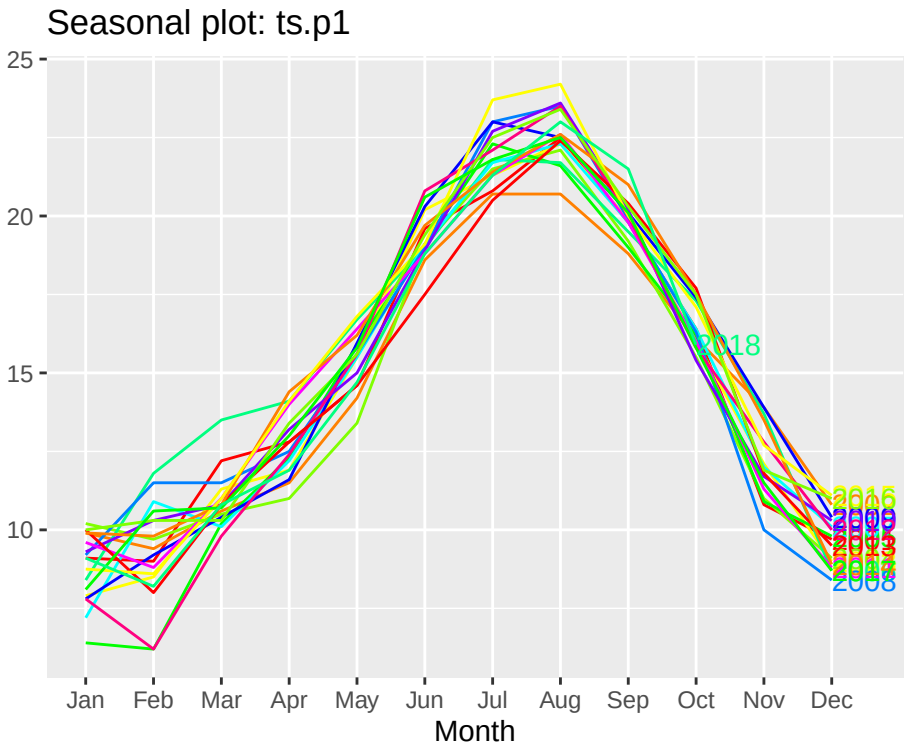
We decompose the data: *Seasonal component*, *Trend component*, *Cycle component* y *residual*.

```
# obtenemos una serie temporal con frecuencia 12 meses
obj.ts <- ts(na.omit(temperaturas$tm_max), frequency = 12, start = c(2001, 1))

# descompose
decomp <- stl(obj.ts, s.window = "periodic") # uses additive model by default
# decomposing the series and removing the seasonality can be done with seasadj() within t
deseasonal_cnt <- seasadj(decomp)
plot(decomp)
```



Let's see if it is seasonal, let's look at the data by year:

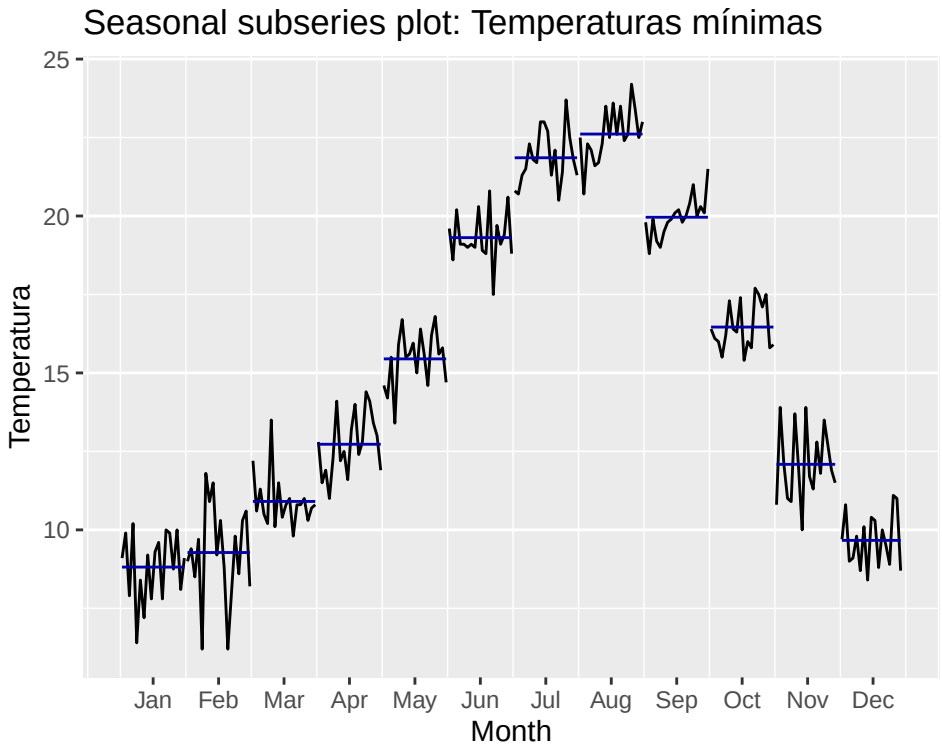
```
ggseasonplot(ts.p1, col = rainbow(12), year.labels = TRUE)
```



As we can see, all the years are identical, which indicates that the temperatures repeat annual and monthly cycles.

With these graphs we can see that our series is seasonal, at the beginning of the year it is down, during the year it rises and from August onwards it starts to fall. This cycle is repeated every year.

```
ggsubseriesplot(ts.p1) +
  ylab("Temperatura") +
  ggtitle("Seasonal subseries plot: Temperaturas mínimas")
```

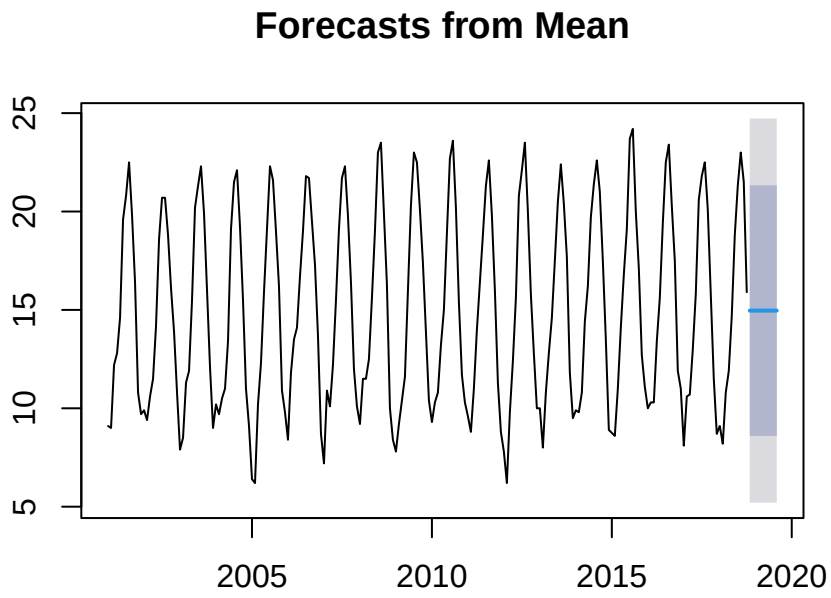


With this last graph we see the subseries by months. From it we see how each month moves in the same range. With this we confirm that it is a seasonal series.

Average method

With this method we see that the prediction is basically an average.

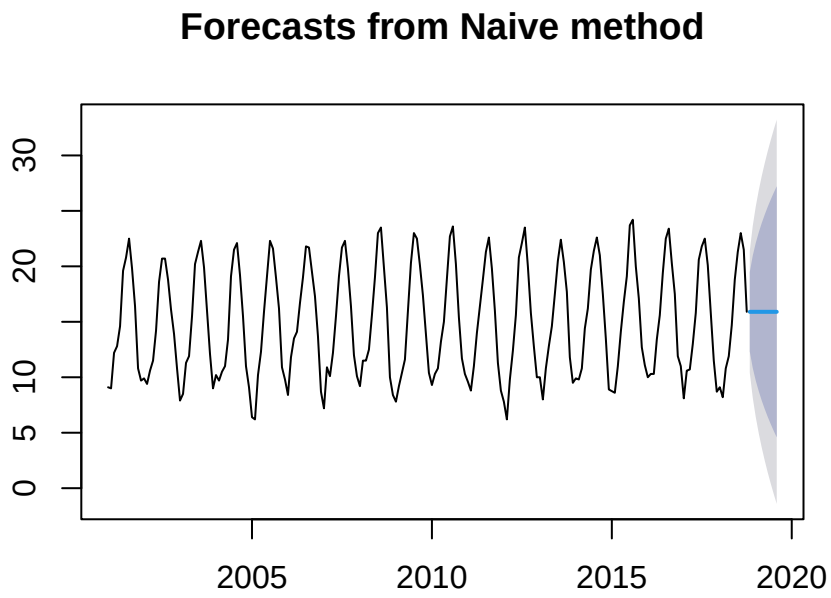
```
mdata.ts.p1 <- meanf(ts.p1,10)
plot(mdata.ts.p1)
```



Naïve method

This method uses the last value of y_t.

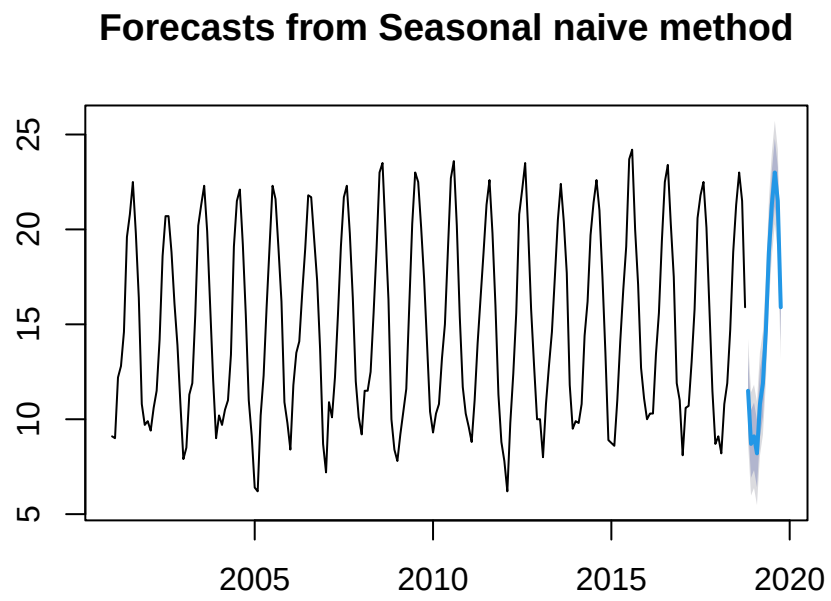
```
mdata.ts.p1 <- naive(ts.p1, 10)
plot(mdata.ts.p1)
```



Seasonal naïve method

This method uses the last year to predict. As we will see in the graph, it generates what has happened in the last year.

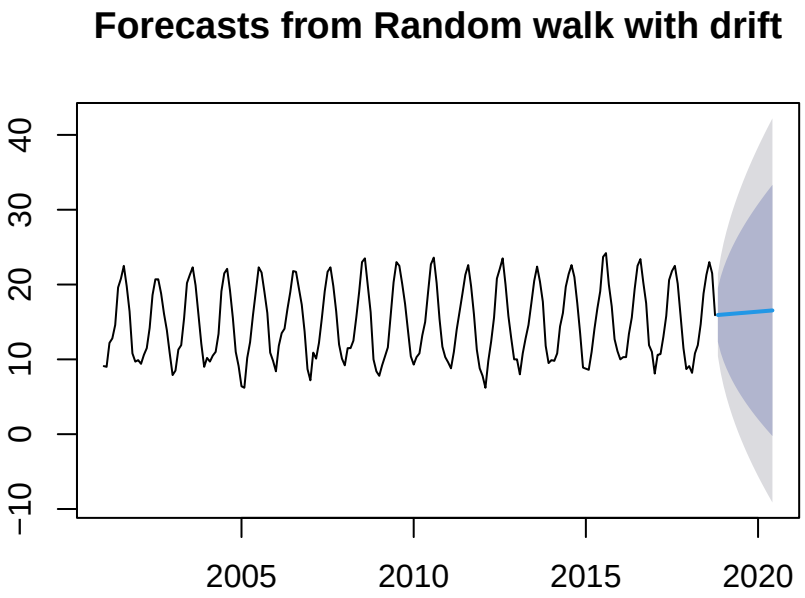
```
mdata.ts.p1 <- snaive(ts.p1, 12)
plot(mdata.ts.p1)
```



Drift method

This method is based on the previous one and that the amount of change over time is set as the average change in the historical data.

```
mdata.ts.p1 <- rwf(ts.p1, 20, drift = TRUE)
plot(mdata.ts.p1)
```



Conclusion

My data series as I have mentioned is for minimum temperatures, so these methods are too simple to predict well what will happen with the temperature in the following years. The only model that could be used is the *Seasonal naïve method* which uses the last year to see what will happen next year. But we still need to use better methods.

5.3 Decomposing Time Series

Decomposing a time series means separating it into its constituent components: trend, cycles, irregular, a seasonal component and a remainder component.

If S_t is the seasonal component, T_t is the trend-cycle component, and R_t is the remainder component, we can consider an **additive decomposition** or a **multiplicative decomposition** as follows:

Additive decomposition:

$$y_t = S_t + T_t + R_t$$

Multiplicative decomposition:

$$y_t = S_t \times T_t \times R_t$$

- Addditve decomposition is adequate when the seasonal part and the trend-cycle does not vary too much.
- Multiplicative decomposition is adequate when the amplitude of the seasonal part increases or decreases with the average level of the time series.

Multiplicative decomposition can be transformed into an additive one just by applying logarithms:

$$\log y_t = \log S_t + \log T_t + \log R_t$$

5.3.1 Additive decomposition

We describe how decompose a time series considering additive decomposition.

Note: Multiplicative decomposition is similar, except that the subtractions are replaced by divisions.

5.3.2 Non-Seasonal Data

- A non-seasonal time series consists of a trend component and an irregular component.
- To estimate the trend component of a non-seasonal time series that can be described using an *additive model*, it is common to use a smoothing method, such as calculating the Simple Moving Average (SMA) of the time series.

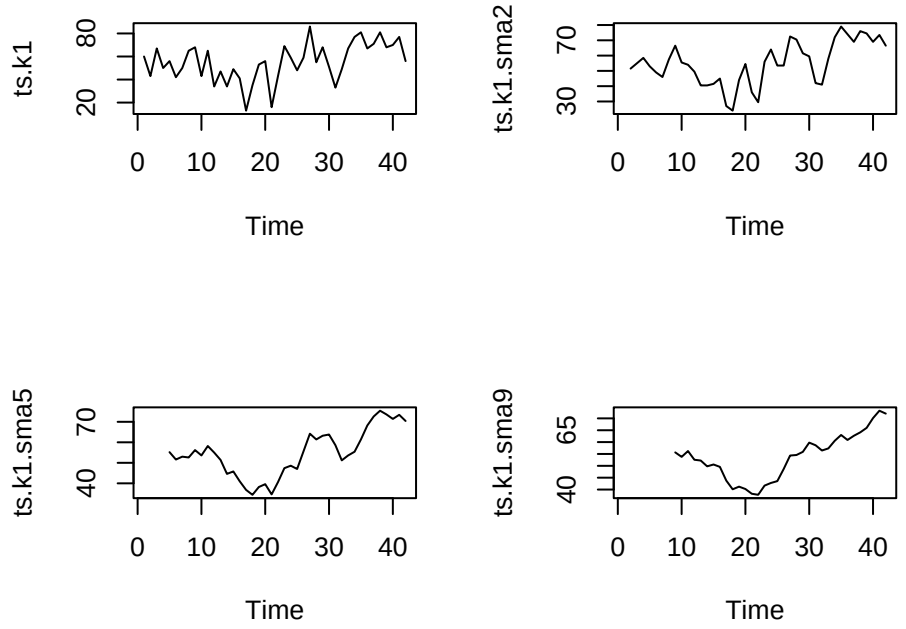
The **SMA()** function in the ***TTR*** package is used to smooth time series data using a simple moving average:

$$\hat{y}_t = \frac{1}{m} \sum_{j=-k}^{k} y_{t+j}$$

To compute the average eliminates randomness in the data.

To specify the order of the simple moving average, the parameter n will be used. For example, to calculate a simple moving average of order 2, we set $n = 2$ in the SMA() function.

```
library("TTR")
# simple moving average of order 2
# applied in the first dataset
data.ts.k1 <- scan(
  here::here("data",
  "ts_k1.dat"), skip = 1)
ts.k1 <- ts(data.ts.k1)

# Applying the MA of order 2,5,9
ts.k1.sma2 <- SMA(ts.k1, n = 2)
ts.k1.sma5 <- SMA(ts.k1, n = 5)
ts.k1.sma9 <- SMA(ts.k1, n = 9)

# Draw all the pictures together
par(mfrow = c(2, 2))
plot.ts(ts.k1)
plot.ts(ts.k1.sma2)
plot.ts(ts.k1.sma5)
plot.ts(ts.k1.sma9)
```



There still appears to be quite a lot of random fluctuations in the time series smoothed using a simple moving average of order 2.

Thus, to estimate the trend component more accurately, we might want to try smoothing the data with a simple moving average of a higher order. This takes a little bit of trial-and-error, to find the right amount of smoothing.

With MA of order 5, and order 9 the trend starts to be clearly identified in the plots.

5.3.3 Seasonal Data

A seasonal time series consists of a trend component, a seasonal component and an irregular component.

We can use the **decompose** function in R which estimates:

- the trend-cycle component using MA $\hat{y}_t$,
- the detrended serie $y_t - \hat{y}_t$
- the seasonal component S_t, computing the average of the detrended values for that season,

- the random components $R_t = y_t - \hat{y}_t - S_t$.

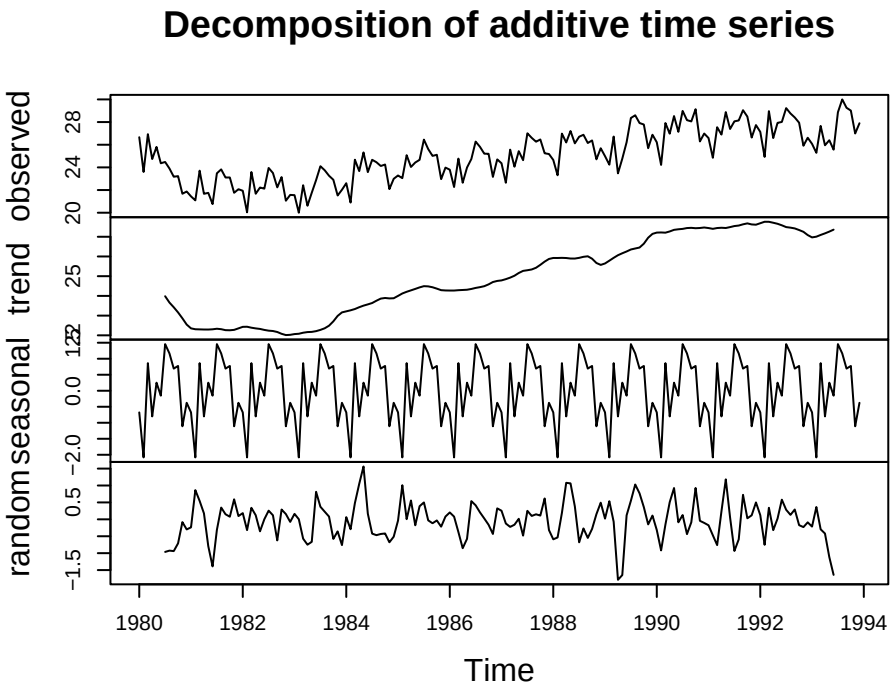
To compute the detrended serie in additive decomposition: $y_t - T_t = S_t + R_t$

Given the time series **ts.n1**, we will decompose the time serie using the following:

```
tsn1 <- scan(
  here::here("data",
  "ts_n1.dat"), skip = 1)
ts.n1 <- ts(tsn1, frequency = 12, start = c(1980, 1))
ts.n1.comp <- decompose(ts.n1)
```

The estimated seasonal factors are given for the months January-December, and are the same for each year.

```
plot(ts.n1.comp)
```



The different parts of the decomposed series can be managed as usual:

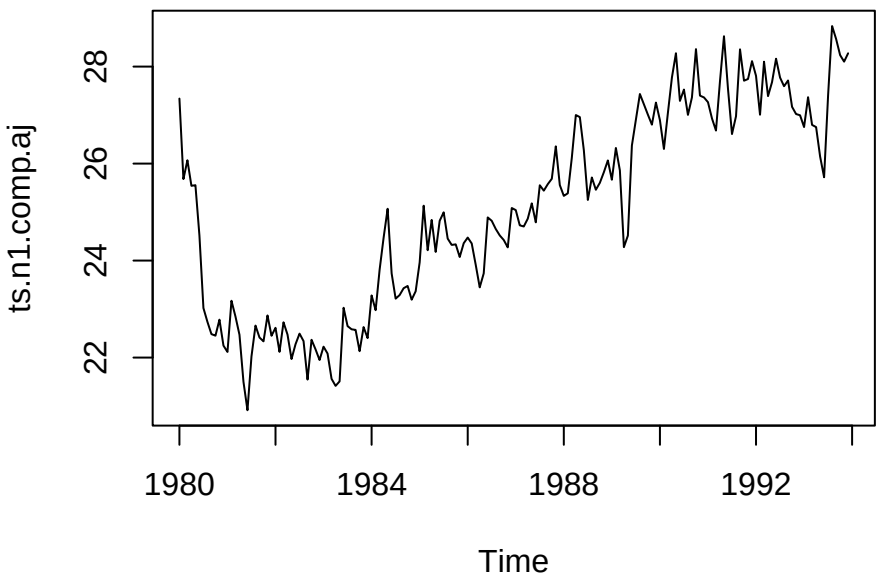
```
ts.n1.comp$trend
ts.n1.comp$seasonal
ts.n1.comp$random
```

5.3.4 Seasonal Adjusting

As we have explained, if you have a seasonal time series that can be described using an *additive model*, you can seasonally adjust the time series by estimating the seasonal component, and subtracting the estimated seasonal component from the original time series.

```
ts.n1.comp <- decompose(ts.n1)
ts.n1.comp.aj <- ts.n1 - ts.n1.comp$seasonal

ts.n1.comp.aj
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 1980 27.34019 25.68096 26.06848 25.54168 25.55435 24.51726 23.02095 22.73641
## 1981 22.11619 23.17196 22.84648 22.47068 21.50035 20.91426 22.02295 22.65941
## 1982 22.61419 22.11796 22.72748 22.47368 21.97035 22.27626 22.49395 22.33941
## 1983 22.22519 22.08296 21.56148 21.41668 21.50935 23.02726 22.64795 22.58341
## 1984 23.28119 22.97696 23.81448 24.47468 25.06835 23.73626 23.21495 23.28941
## 1985 23.96419 25.13196 24.21348 24.83868 24.17835 24.82026 24.99495 24.45341
## 1986 24.47519 24.35296 23.91248 23.44768 23.73635 24.89026 24.81995 24.65141
## 1987 25.04119 24.72696 24.70248 24.86368 25.17935 24.78826 25.55295 25.44141
## 1988 25.33419 25.38696 26.11948 27.00068 26.95835 26.27526 25.24995 25.71341
## 1989 25.66719 26.32196 25.85848 24.27668 24.51535 26.37226 26.90495 27.43441
## 1990 26.89419 26.30096 27.05148 27.77668 28.27535 27.29226 27.52595 27.00441
```

```
## 1991 27.26619 26.93096 26.68048 27.69768 28.62635 27.54326 26.60895 26.97641
## 1992 27.80919 27.00696 28.10048 27.39068 27.67935 28.16226 27.77295 27.59441
## 1993 26.75319 27.36896 26.79748 26.75268 26.14635 25.71826 27.40895 28.83541
##           Sep      Oct      Nov      Dec
## 1980 22.48338 22.45176 22.78177 22.24682
## 1981 22.41338 22.33476 22.86877 22.44982
## 1982 21.54638 22.36676 22.16877 21.94982
## 1983 22.57038 22.13176 22.62877 22.40182
## 1984 23.43038 23.47676 23.19377 23.36782
## 1985 24.32238 24.33476 24.07377 24.35782
## 1986 24.51838 24.42376 24.27177 25.08382
## 1987 25.57638 25.68676 26.35577 25.55682
## 1988 25.46038 25.60376 25.82177 26.06482
## 1989 27.22238 27.00876 26.80277 27.25782
## 1990 27.36438 28.36076 27.40077 27.36382
## 1991 28.35638 27.70876 27.74377 28.11182
## 1992 27.71338 27.16976 27.02177 26.99582
## 1993 28.56938 28.23676 28.10177 28.27382
plot(ts.n1.comp.aj)
```



The seasonal variation has been removed from the seasonally adjusted time series. The seasonally adjusted time series now just contains the trend component and an irregular component.

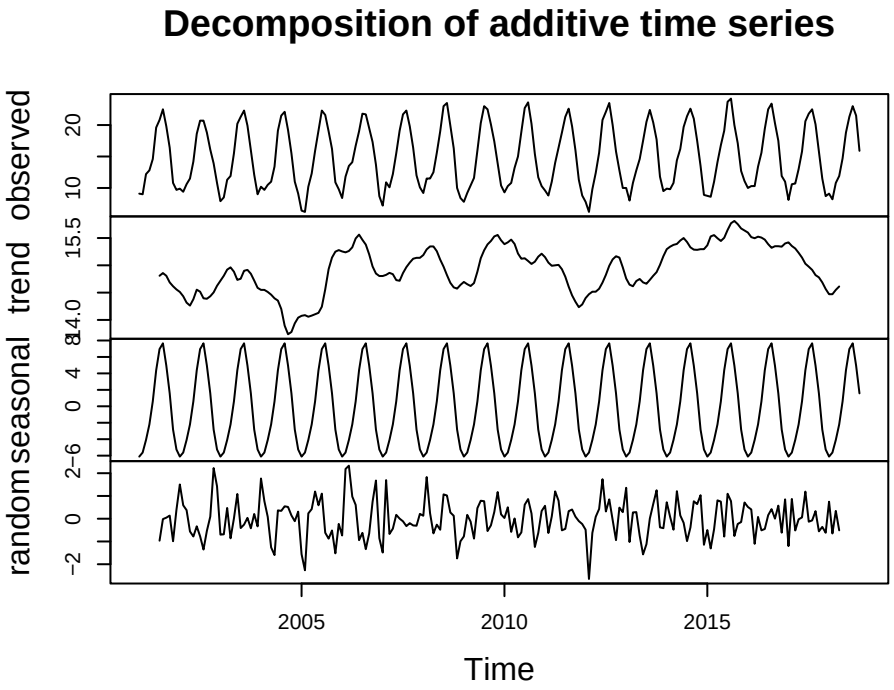
5.3.5 Example: Temperature forecast in Malaga - Decomposing Time Series

We have seen that the time series we use is seasonal so we are going to use the method of *Additive decomposition*.

Additive decomposition

A seasonal time series consists of a trend component, a seasonal component and an irregular component. To decompose it we are going to use **decompose**.

```
# we decompose the time series and draw it
ts.p1.comp <- decompose(ts.p1)
plot(ts.p1.comp)
```



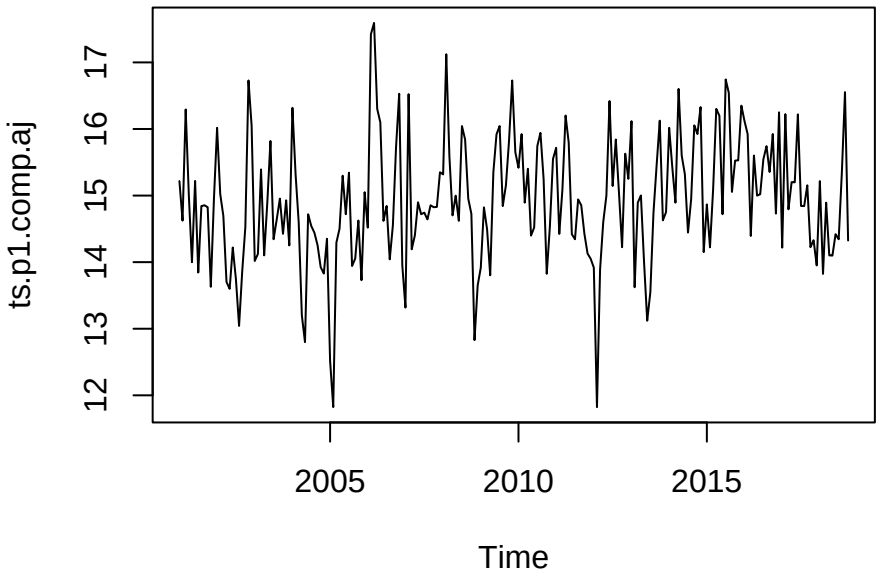
5.3.6 Seasonal Adjusting

If we have a seasonal time series that can be described using an additive model, we can seasonally adjust the time series by estimating the seasonal component and subtracting the estimated seasonal component from the original time series.

```
# we subtract the stationary component
ts.p1.comp.aj <- ts.p1 - ts.p1.comp$seasonal
ts.p1.comp.aj
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 2001 15.21772 14.62312 16.29322 15.00204 13.99887 15.21996 13.84395 14.84199
## 2002 16.01772 15.02312 14.69322 13.70204 13.59887 14.21996 13.74395 13.04199
## 2003 14.01772 14.12312 15.39322 14.10204 14.89887 15.81996 14.34395 14.64199
## 2004 16.31772 15.32312 14.59322 13.20204 12.79887 14.71996 14.54395 14.44199
## 2005 12.51772 11.82312 14.29322 14.50204 15.29887 14.71996 15.34395 13.94199
## 2006 14.51772 17.42312 17.59322 16.30204 16.09887 14.61996 14.84395 14.04199
## 2007 13.31772 16.52312 14.19322 14.40204 14.89887 14.71996 14.74395 14.64199
## 2008 15.31772 17.12312 15.59322 14.70204 14.99887 14.61996 16.04395 15.84199
## 2009 13.91772 14.82312 14.49322 13.80204 15.34887 15.91996 16.04395 14.84199
## 2010 15.41772 15.92312 14.89322 15.40204 14.39887 14.51996 15.74395 15.94199
## 2011 15.71772 14.42312 15.09322 16.20204 15.79887 14.41996 14.34395 14.94199
## 2012 13.91772 11.82312 13.89322 14.60204 14.99887 16.41996 15.14395 15.84199
## 2013 16.11772 13.62312 14.89322 15.00204 13.99887 13.11996 13.54395 14.74199
## 2014 16.01772 15.42312 14.89322 16.60204 15.59887 15.31996 14.44395 14.94199
## 2015 14.86772 14.22312 15.09322 16.30204 16.19887 14.71996 16.74395 16.54199
## 2016 16.11772 15.92312 14.39322 15.60204 14.99887 15.01996 15.54395 15.74199
```

```
## 2017 14.21772 16.22312 14.79322 15.20204 15.19887 16.21996 14.84395 14.84199
## 2018 15.21772 13.82312 14.89322 14.10204 14.09887 14.41996 14.34395 15.34199
##           Sep      Oct      Nov      Dec
## 2001 14.85424 14.82508 13.62900 14.95081
## 2002 13.85424 14.52508 16.72900 16.05081
## 2003 14.95424 14.42508 14.92900 14.25081
## 2004 14.25424 13.92508 13.82900 14.35081
## 2005 14.05424 14.62508 13.72900 15.05081
## 2006 14.55424 15.72508 16.52900 13.95081
## 2007 14.85424 14.82508 14.82900 15.35081
## 2008 14.95424 14.72508 12.82900 13.65081
## 2009 15.15424 15.82508 16.72900 15.65081
## 2010 15.25424 13.82508 14.52900 15.55081
## 2011 14.85424 14.42508 14.12900 14.05081
## 2012 15.05424 14.22508 15.62900 15.25081
## 2013 15.45424 16.12508 14.62900 14.75081
## 2014 16.05424 15.92508 16.32900 14.15081
## 2015 15.05424 15.52508 15.52900 16.35081
## 2016 15.35424 15.92508 14.72900 16.25081
## 2017 15.15424 14.22508 14.32900 13.95081
## 2018 16.55424 14.32508
```

```
# we draw the adjusted values
plot(ts.p1.comp.aj)
```



5.3.7 Forecasts using Exponential Smoothing

Exponential Smoothing considers weigths to the values of the variables: weighted averages of past observations. The weights decrease exponentially for distant values.

Taxonomy

5.3.8 Simple Exponential Smoothing

The idea is to consider a weighted average between the value of the variable and the previous one:

$$\hat{y}_{T+1|T} = \alpha y_T + (1-\alpha)\hat{y}_{T|T-1}$$

If we have into account more past values of the variable, after some computations, the forecast will be:

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + ...$$

The parameter α controls the rates at which the weights decrease. The weights decrease exponentially and this is the reason of the name of the method.

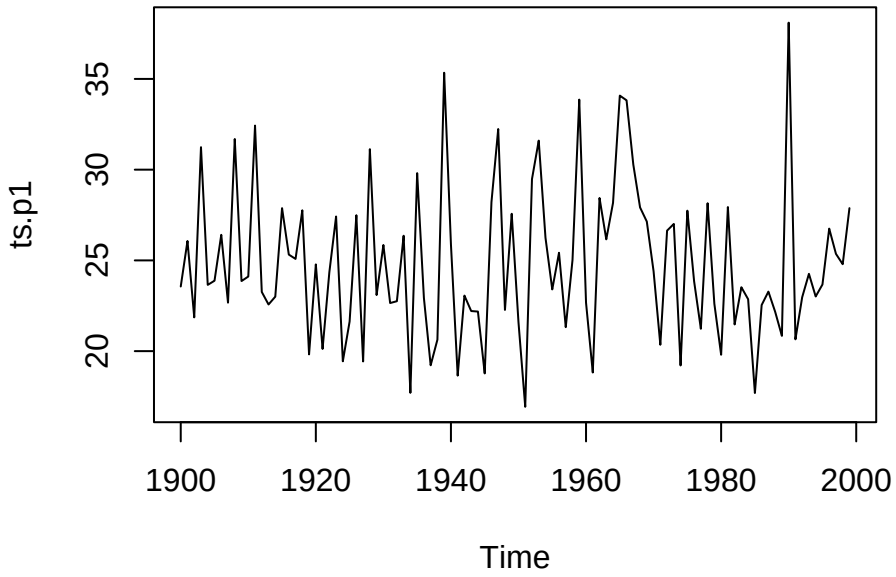
- α has values between 0 and 1.
- α close to 0, more important are the longer observations - slow learning (past observations have a large influence on forecasts).
- α close to 1, more important the more recent observation s- fast learning (that is, only the most recent values influence the forecasts).
- With $\alpha = 0$ or $\alpha = 1$ we have the extreme cases.

If you have a time series that can be described using an additive model with constant level (no clear trend) and no seasonality, you can use **simple exponential smoothing** to make short-term forecasts.

- The simple exponential smoothing method provides a way of estimating the level at the current time point.

The file **ts_p1.dat** contains total annual snowfall for a city.

```
tsp1 <- scan(
  here::here("data",
  "ts_p1.dat"), skip = 1)
ts.p1 <- ts(tsp1, start = c(1900))
plot.ts(ts.p1)
```



- The mean stays constant at about 25 - roughly constant level.
- The random fluctuations seem to be roughly constant.

Thus, we can make forecasts using simple exponential smoothing.

In R, we can fit a simple exponential smoothing predictive model using the **HoltWinters** function.

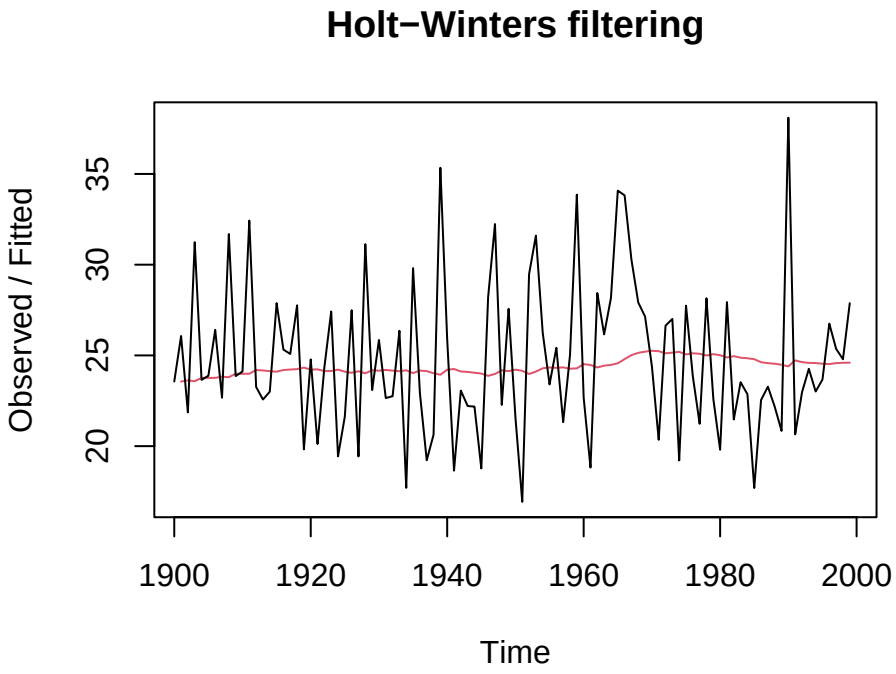
Main parameters:

- alpha: Smoothing is controlled by this parameter; the value of alpha lies between 0 and 1 (close to 0 mean that little weight is placed on the most recent observations when making forecasts of future values).
- beta=FALSE: the function will do exponential smoothing.
- gamma=FALSE: a non-seasonal model is fitted.
- seasonal: **additive by default** or multiplicative seasonal model.

```
ts.p1.forecasts <- HoltWinters(ts.p1, beta = FALSE, gamma = FALSE)
ts.p1.forecasts
## Holt-Winters exponential smoothing without trend and without seasonal component.
##
## Call:
## HoltWinters(x = ts.p1, beta = FALSE, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.02412151
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##       [,1]
## a 24.67819
```

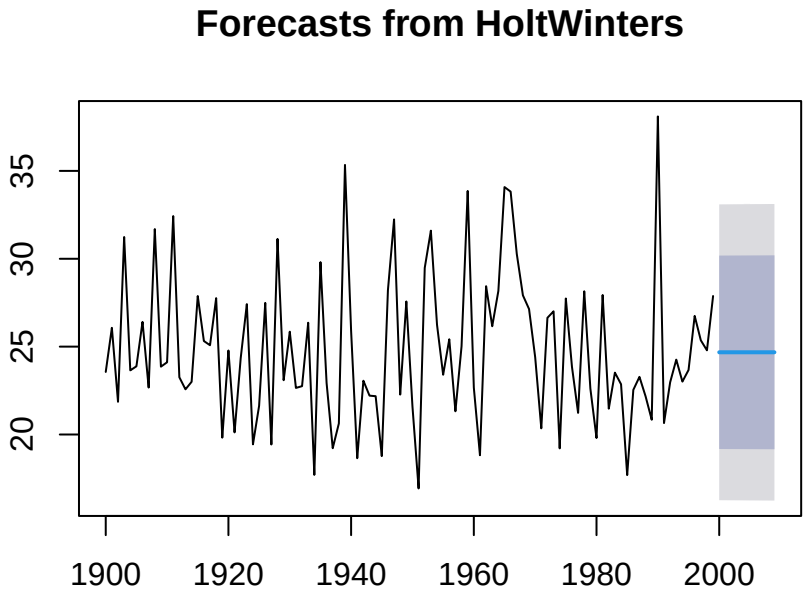
The output of `HoltWinters()` tells us that the estimated value of the alpha parameter is about 0.024. This is very close to zero, telling us that past observations have more influence on forecasts.

```
plot(ts.p1.forecasts)
```



The plot shows the original time series in black, and the forecasts of these values as a red line. The time series of forecasts is much smoother than the time series of the original data here.

```
plot(forecast(ts.p1.forecasts))
```



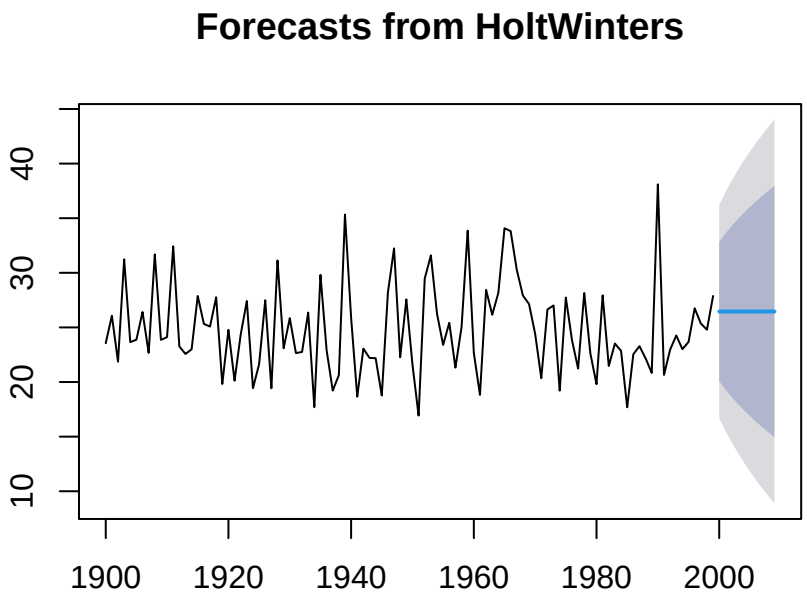
The function `forecast` predicts the value as the previous plot shows.

As a measure of the accuracy of the forecasts, we can calculate the sum of squared errors for the in-sample forecast errors, that is, the forecast errors for the time period covered by our original time series. The sum-of-squared- errors is stored in a named element of the list variable `ts.p1.forecasts` called "SSE", so we can get its value by typing:

```
ts.p1.forecasts$SSE
## [1] 1828.855
```

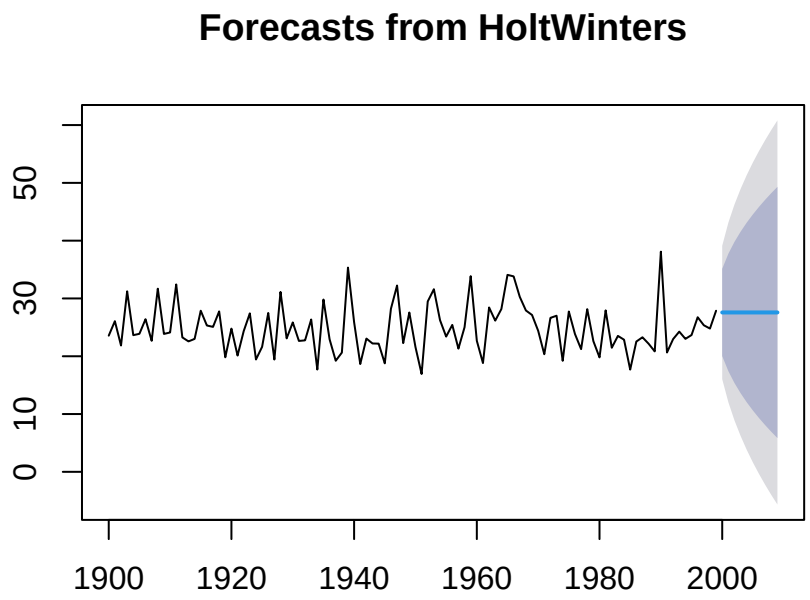
Finally, we try with different values of α and the errors are computed:

```
ts.p2.forecasts <- HoltWinters(ts.p1, alpha = 0.5, beta = FALSE, gamma = FALSE)
ts.p2.forecasts
## Holt-Winters exponential smoothing without trend and without seasonal component.
##
## Call:
## HoltWinters(x = ts.p1, alpha = 0.5, beta = FALSE, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.5
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##       [,1]
## a 26.45644
plot(forecast(ts.p2.forecasts))
```




```
ts.p2.forecasts$SSE
## [1] 2428.339
ts.p3.forecasts <- HoltWinters(ts.p1, alpha = 0.9, beta = FALSE, gamma = FALSE)
ts.p3.forecasts
## Holt-Winters exponential smoothing without trend and without seasonal component.
##
## Call:
## HoltWinters(x = ts.p1, alpha = 0.9, beta = FALSE, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.9
```

```
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##       [,1]
## a 27.57778
plot(forecast(ts.p3.forecasts))
```




```
ts.p3.forecasts$SSE
## [1] 3399.419
```

Note: The forecasts made by `HoltWinters` are stored in the list called fitted.

```
ts.p3.forecasts$fitted
## Time Series:
## Start = 1901
## End = 1999
## Frequency = 1
##          xhat    level
## 1901 23.56000 23.56000
## 1902 25.81900 25.81900
## 1903 22.25590 22.25590
## 1904 30.34159 30.34159
## 1905 24.31916 24.31916
## 1906 23.92392 23.92392
## 1907 26.16139 26.16139
## 1908 23.01914 23.01914
## 1909 30.82291 30.82291
## 1910 24.55629 24.55629
## 1911 24.15463 24.15463
## 1912 31.60246 31.60246
## 1913 24.09425 24.09425
## 1914 22.72242 22.72242
```

```
## 1915 22.97224 22.97224
## 1916 27.38922 27.38922
## 1917 25.52692 25.52692
## 1918 25.12469 25.12469
## 1919 27.49647 27.49647
## 1920 20.58765 20.58765
## 1921 24.36076 24.36076
## 1922 20.54408 20.54408
## 1923 23.96041 23.96041
## 1924 27.07404 27.07404
## 1925 20.20340 20.20340
## 1926 21.48734 21.48734
## 1927 26.88973 26.88973
## 1928 20.17597 20.17597
## 1929 30.03460 30.03460
## 1930 23.78446 23.78446
## 1931 25.64345 25.64345
## 1932 22.94934 22.94934
## 1933 22.76993 22.76993
## 1934 26.00099 26.00099
## 1935 18.53010 18.53010
## 1936 28.68201 28.68201
## 1937 23.50520 23.50520
## 1938 19.64852 19.64852
## 1939 20.53185 20.53185
## 1940 33.85919 33.85919
## 1941 26.68692 26.68692
## 1942 19.45369 19.45369
## 1943 22.69937 22.69937
## 1944 22.25894 22.25894
## 1945 22.18789 22.18789
## 1946 19.11179 19.11179
## 1947 27.30018 27.30018
## 1948 31.74602 31.74602
## 1949 23.21760 23.21760
## 1950 27.13476 27.13476
## 1951 22.14448 22.14448
## 1952 17.45145 17.45145
## 1953 28.27714 28.27714
## 1954 31.26771 31.26771
## 1955 26.75177 26.75177
## 1956 23.73518 23.73518
## 1957 25.25152 25.25152
## 1958 21.71315 21.71315
## 1959 24.68932 24.68932
## 1960 32.94293 32.94293
## 1961 23.69729 23.69729
## 1962 19.30773 19.30773
## 1963 27.52677 27.52677
```

```
## 1964 26.29668 26.29668
## 1965 27.98267 27.98267
## 1966 33.47027 33.47027
## 1967 33.78503 33.78503
## 1968 30.63050 30.63050
## 1969 28.19105 28.19105
## 1970 27.24511 27.24511
## 1971 24.68451 24.68451
## 1972 20.78345 20.78345
## 1973 26.05435 26.05435
## 1974 26.91443 26.91443
## 1975 19.98044 19.98044
## 1976 26.96404 26.96404
## 1977 24.16140 24.16140
## 1978 21.52314 21.52314
## 1979 27.48731 27.48731
## 1980 23.09773 23.09773
## 1981 20.12977 20.12977
## 1982 27.15898 27.15898
## 1983 22.03890 22.03890
## 1984 23.37189 23.37189
## 1985 22.91119 22.91119
## 1986 18.21212 18.21212
## 1987 22.10721 22.10721
## 1988 23.16272 23.16272
## 1989 22.26927 22.26927
## 1990 20.98293 20.98293
## 1991 36.38829 36.38829
## 1992 22.22383 22.22383
## 1993 22.89538 22.89538
## 1994 24.12354 24.12354
## 1995 23.12135 23.12135
## 1996 23.61514 23.61514
## 1997 26.43651 26.43651
## 1998 25.46765 25.46765
## 1999 24.85777 24.85777
```

5.3.9 Forecasting using forecast package

We can make forecasts for further time points by using the `forecast.HoltWinters` function in the R `forecast` package.

```
require("forecast")
library("forecast")
```

When using the `forecast.HoltWinters` function, as its first argument (input), you pass it the predictive model that you have already fitted using the `HoltWinters` function.

You specify how many further time points you want to make forecasts for by using the "h" parameter in `forecast.HoltWinters()`.

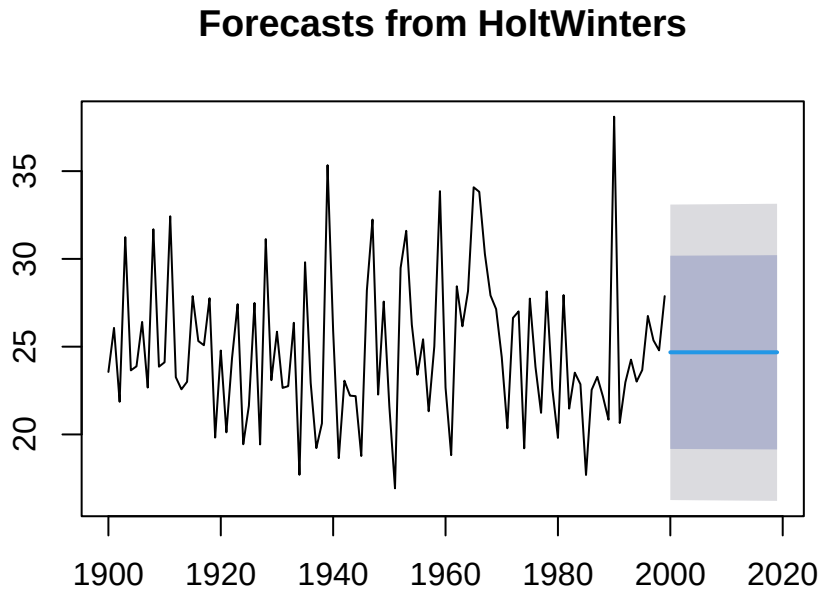
The output for the next 20 years is:

```
# ts.p1.forecasts2 <- forecast.HoltWinters(ts.p1.forecasts, h=20)

ts.p1.forecasts2 <- forecast:::forecast.HoltWinters(ts.p1.forecasts, h = 20)
ts.p1.forecasts2
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2000       24.67819 19.17493 30.18145 16.26169 33.09470
## 2001       24.67819 19.17333 30.18305 16.25924 33.09715
## 2002       24.67819 19.17173 30.18465 16.25679 33.09960
## 2003       24.67819 19.17013 30.18625 16.25434 33.10204
## 2004       24.67819 19.16853 30.18785 16.25190 33.10449
## 2005       24.67819 19.16694 30.18945 16.24945 33.10694
## 2006       24.67819 19.16534 30.19105 16.24701 33.10938
## 2007       24.67819 19.16374 30.19265 16.24456 33.11182
## 2008       24.67819 19.16214 30.19425 16.24212 33.11427
## 2009       24.67819 19.16054 30.19584 16.23968 33.11671
## 2010       24.67819 19.15895 30.19744 16.23724 33.11915
## 2011       24.67819 19.15735 30.19904 16.23479 33.12159
## 2012       24.67819 19.15576 30.20063 16.23235 33.12403
## 2013       24.67819 19.15416 30.20223 16.22991 33.12647
## 2014       24.67819 19.15257 30.20382 16.22748 33.12891
## 2015       24.67819 19.15097 30.20542 16.22504 33.13135
## 2016       24.67819 19.14938 30.20701 16.22260 33.13379
## 2017       24.67819 19.14778 30.20860 16.22016 33.13623
## 2018       24.67819 19.14619 30.21020 16.21773 33.13866
## 2019       24.67819 19.14460 30.21179 16.21529 33.14110
```

The forecast for a year, a 80% prediction interval for the forecast, and a 95% prediction interval for the forecast.

Plotting the predictions:

```
forecast:::plot.forecast(ts.p1.forecasts2)
```



The forecasts for 2000-2020 are plotted as a blue line, the 80% prediction interval as a dark blue shaded area, and the 95% prediction interval as a grey shaded area.

Forecast errors

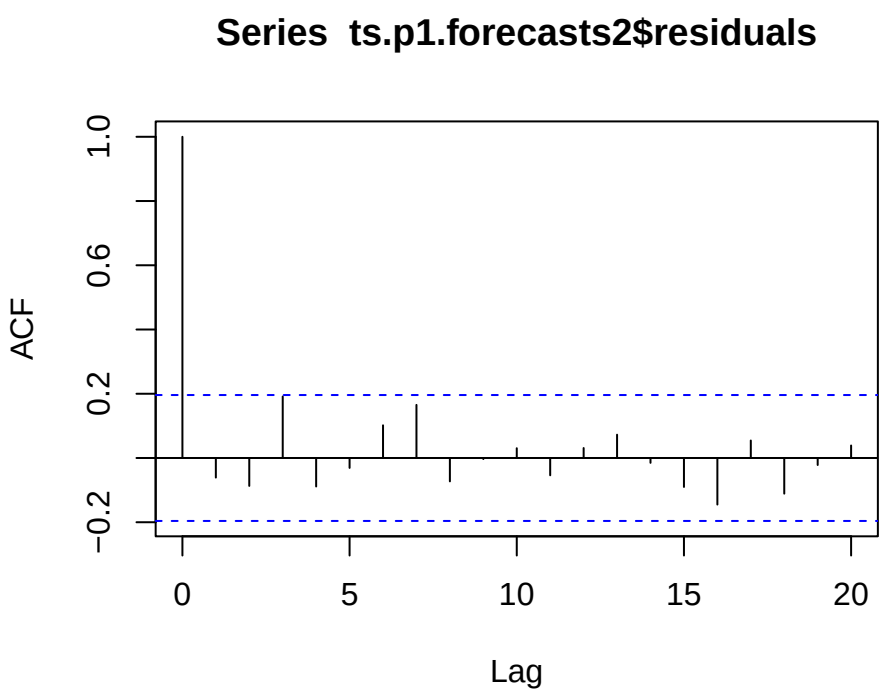
Forecast errors are stored in the element *residuals*.

- If the predictive model cannot be improved upon, there should be no correlations between forecast errors for successive predictions.
- If there are correlations between forecast errors for successive predictions, it is likely that the simple exponential smoothing forecasts could be improved upon by another forecasting technique.

To figure out whether this is the case, we can obtain a correlogram of the in-sample forecast errors. We can calculate a correlogram of the forecast errors using the `acf` function in R.

`acf` computes (and plots, by default) the estimation of the autocovariance or AutoCorrelation Function (ACF).

```
acf(ts.p1.forecasts2$residuals,na.action = na.pass, lag.max = 20)
```



The autocorrelation at lag 3 is just touching the significance bounds (significant evidence for non-zero correlations).

To test whether there is significant evidence for non-zero correlations at lags, we can carry out a Ljung- Box test. This can be done in R using the **Box.test** function.

```
Box.test(ts.p1.forecasts2$residuals, lag = 20, type = "Ljung-Box")
##
##  Box-Ljung test
##
## data:  ts.p1.forecasts2$residuals
## X-squared = 17.401, df = 20, p-value = 0.6268
```

The p-value is 0.6, so there is little evidence of non-zero autocorrelations in the in-sample forecast errors at lags 1-20.

5.3.10 ACF test

ACF (AutoCorrelation Function):

- Purpose: The ACF is used to study and visualize the autocorrelation in a time series. Autocorrelation measures the relationship between a data point and its past lags (previous observations in the series). It helps you understand the patterns, cycles, and seasonality in the data.
- How it works: The ACF computes and plots the autocorrelation coefficients for different lags. It shows the strength and direction (positive or negative) of the correlation between each observation and its lags.
- Interpretation: ACF can help you identify significant lags in the data, which can inform the choice of lag order in models like ARIMA (see below). Significant lags may indicate seasonality or other patterns in the data.

In an ACF (AutoCorrelation Function) test, if you observe a significant autocorrelation at lag 3, it means that there is a statistically meaningful relationship between data points that are separated by a lag of 3 time periods (lags). Here's what this implies:

- **Lag 3 Autocorrelation:** At lag 3, the ACF measures the correlation between the current data point and the data point that is three time periods (time steps) in the past. A positive autocorrelation suggests that as the current data point increases, the data point at lag 3 also tends to increase, and vice versa for a negative autocorrelation.
- **Significance:** When we say that the autocorrelation at lag 3 is "significant," it means that the correlation is unlikely to have occurred by random chance. In other words, the autocorrelation is strong enough to be considered a real relationship in the data.
- **Interpretation:** The significance of a lag in the ACF can provide insights into potential patterns or seasonality in the time series. For example, a significant positive autocorrelation at lag 3 might suggest that the time series exhibits a repeating pattern or cycle with a length of 3 time periods.
- **Modeling:** When building time series models like ARIMA, the presence of significant lags in the ACF can inform the choice of model parameters (e.g., the autoregressive order, denoted as 'p' in ARIMA) to capture these relationships and patterns.

In summary, a significant autocorrelation at lag 3 in the ACF suggests that there is a meaningful relationship between data points separated by a lag of 3 time periods, which can be indicative of repeating patterns or seasonality in the time series. It's an important finding when analyzing and modeling time series data.

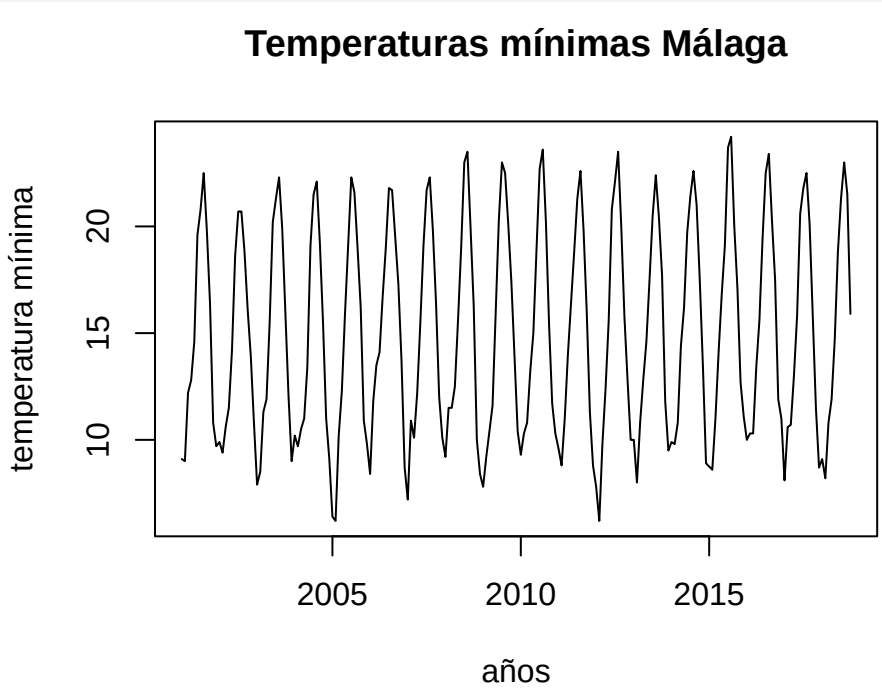
5.3.11 Example: Temperature forecast in Malaga - Exponential Smoothing

Analyse the evolution of minimum temperatures since 2001 in Malaga **with the methods seen in this document. Malaga** with the methods seen in this document**.

We obtain a time-series object with the `ts` function. data are divided into months, the parameter *frequency* is set to 12 and we indicate that we start in January 2001. After this, we draw the object:

```
library('ggplot2')
library('TTR')
library('forecast')
library('tseries')
```

```
library(readr)
temperaturas <- read_csv(
  here::here("data",
  "temp.csv"))
## Rows: 214 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): fecha, indicativo
## dbl (4): X1, X1_1, tm_max, tm_min
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
ts.p1 <- ts(temperaturas$tm_min, start = c(2001, 1), frequency = 12)
plot.ts(ts.p1, xlab = "años", ylab = "temperatura mínima", main = "Temperaturas mínimas M
```



Forecasts using Exponential Smoothing

Exponential Smoothing considers weights on variable values: weighted averages of past observations. The weights decrease exponentially for distant values.

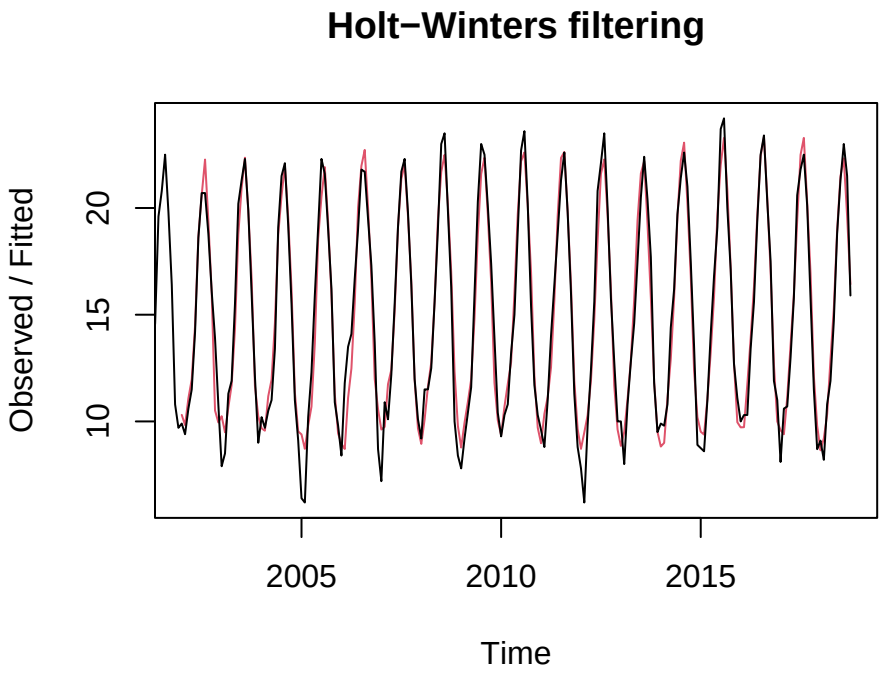
Simple Exponential Smoothing

- alpha parameter of Holt-Winters Filter.
- beta parameter of Holt-Winters Filter. If set to FALSE, the function will do exponential smoothing.
- gamma parameter used for the seasonal component. If set to FALSE, an non-seasonal model is fitted.

Looking at these parameters we have to assemble a model with *beta=FALSE* as we want exponential smoothing.

```
# we obtain the model
ts.p1.forecasts <- HoltWinters(ts.p1, beta = FALSE)
ts.p1.forecasts
## Holt-Winters exponential smoothing without trend and with additive seasonal component.
##
## Call:
## HoltWinters(x = ts.p1, beta = FALSE)
##
## Smoothing parameters:
##  alpha: 0.1412013
##  beta : FALSE
##  gamma: 0.2045694
##
## Coefficients:
##           [,1]
## a   14.8002193
## s1  -3.0179721
## s2  -5.3483612
## s3  -6.0676567
## s4  -5.8300885
## s5  -4.2971166
## s6  -2.0597869
## s7   0.5443399
## s8   4.3512303
## s9   6.9081866
## s10  7.9327353
## s11  5.3361482
## s12  1.4610327
```

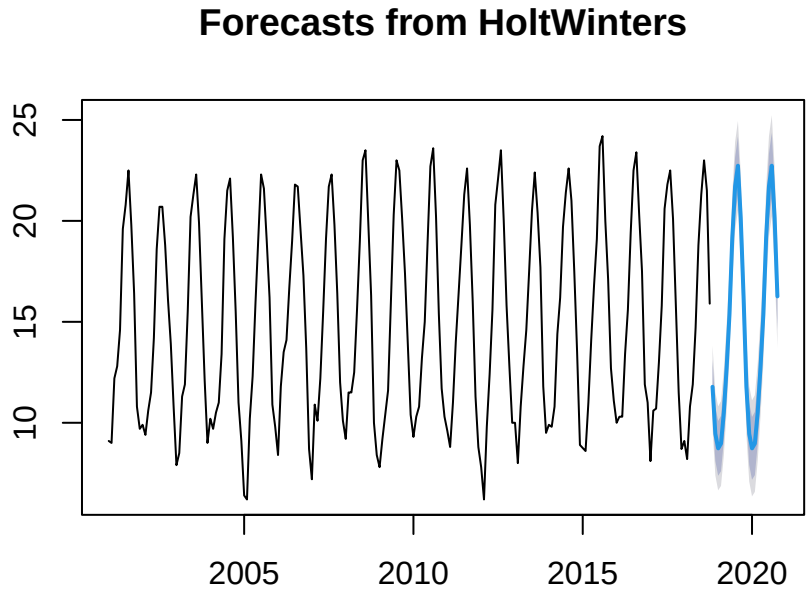
```
# we draw the model
plot(ts.p1.forecasts)
```



The graph shows the original time series in black, and forecasts of these values as a red line. The forecast time series is very close to the original data.

We now draw the forecast using `forecast`.

```
plot(forecast(ts.p1.forecasts))
```



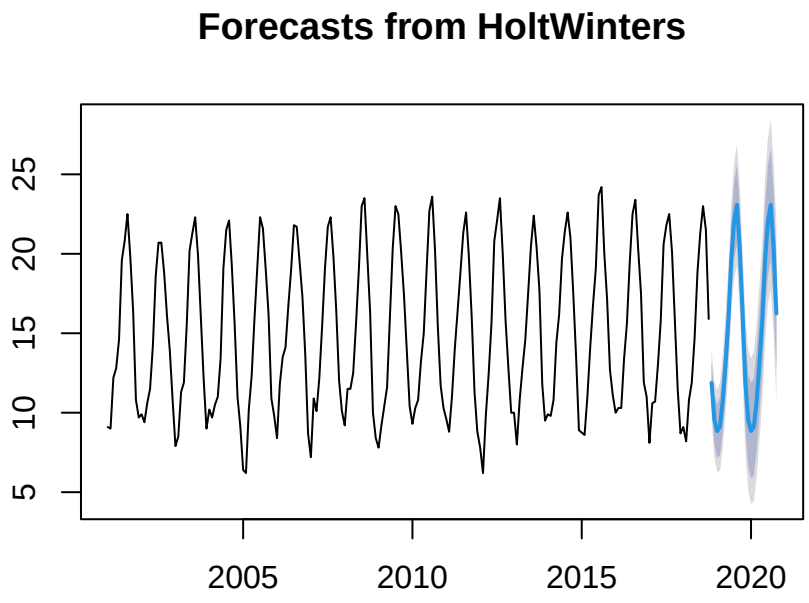
As we can see it fits quite well with what is happening in recent years.

Let's try a higher *alpha* value:

```
ts.p2.forecasts <- HoltWinters(ts.p1, alpha = 0.5, beta = FALSE)
ts.p2.forecasts
## Holt-Winters exponential smoothing without trend and with additive seasonal component.
##
## Call:
```

```
## HoltWinters(x = ts.p1, alpha = 0.5, beta = FALSE)
##
## Smoothing parameters:
##  alpha: 0.5
##  beta : FALSE
##  gamma: 0.3927417
##
## Coefficients:
##           [,1]
## a   15.0512589
## s1  -3.1697282
## s2  -5.4838631
## s3  -6.2075694
## s4  -5.9219038
## s5  -4.2667725
## s6  -1.8548302
## s7   0.8149946
## s8   4.5957651
## s9   7.0279317
## s10  8.0279557
## s11  5.3161216
## s12  1.1929326
```

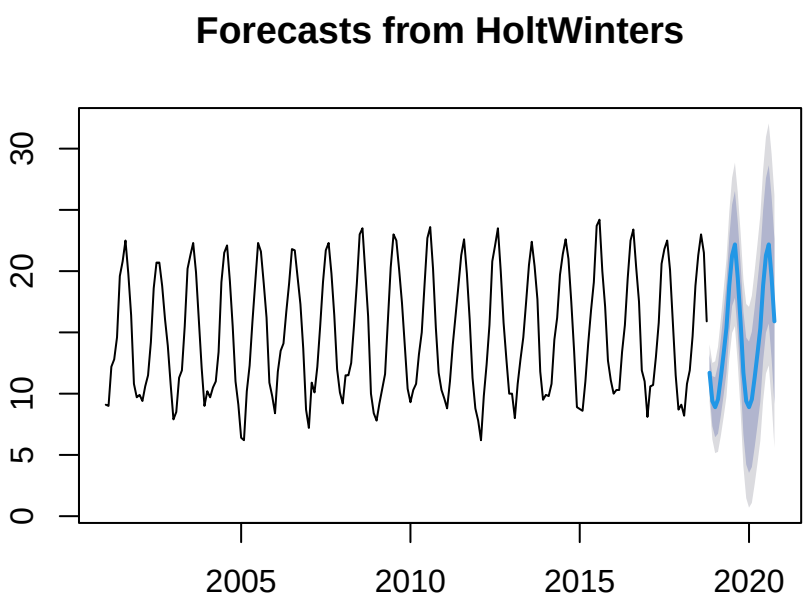
```
plot(forecast(ts.p2.forecasts))
```




```
ts.p3.forecasts <- HoltWinters(ts.p1, alpha = 0.9, beta = FALSE)
ts.p3.forecasts
## Holt-Winters exponential smoothing without trend and with additive seasonal component.
##
## Call:
## HoltWinters(x = ts.p1, alpha = 0.9, beta = FALSE)
##
## Smoothing parameters:
```

```
##  alpha: 0.9
##  beta : FALSE
##  gamma: 0.9999496
##
## Coefficients:
##           [,1]
## a   14.7654331
## s1  -3.0613814
## s2  -5.3688209
## s3  -5.8712039
## s4  -5.2291009
## s5  -3.4451379
## s6  -1.4794315
## s7   0.5904564
## s8   4.0760361
## s9   6.5257969
## s10  7.4170932
## s11  4.6994542
## s12  1.1345783
```

```
plot(forecast(ts.p3.forecasts))
```



As we can see there is not much difference between the first model and the other 2 created later, so the *alpha* value generated by the first model is the optimal one. We can see that the higher the *alpha* value, the narrower the fixed values, and the *gamma* values increase along with *alpha*.

5.3.12 Forecasting using forecast package

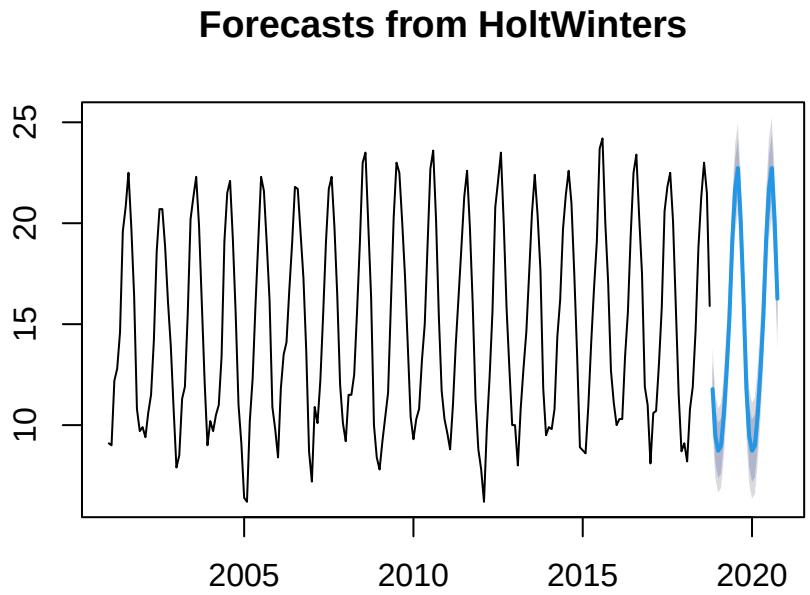
When we use the *forecast.HoltWinters* function, as the first argument, it is passed the predictive model that we have already set using the `HoltWinters` function.

To specify how many additional time points we want to predict, we use the parameter *"h"*.

The prediction for the next 2 years is:

```
# we indicate the parameter h=24 for 2 years, as we have the time series by months
ts.p1.forecasts2 <- forecast:::forecast.HoltWinters(ts.p1.forecasts, h = 24)
ts.p1.forecasts2
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## Nov 2018      11.782247 10.448881 13.11561  9.743039 13.82146
## Dec 2018       9.451858  8.105265 10.79845  7.392421 11.51129
## Jan 2019       8.732563  7.372871 10.09225  6.653094 10.81203
## Feb 2019       8.970131  7.597467 10.34279  6.870822 11.06944
## Mar 2019      10.503103  9.117587 11.88862  8.384139 12.62207
## Apr 2019      12.740432 11.342183 14.13868 10.601995 14.87887
## May 2019      15.344559 13.933692 16.75543 13.186823 17.50229
## Jun 2019      19.151450 17.728075 20.57482 16.974587 21.32831
## Jul 2019      21.708406 20.272634 23.14418 19.512582 23.90423
## Aug 2019      22.732955 21.284891 24.18102 20.518333 24.94758
## Sep 2019      20.136368 18.676116 21.59662 17.903106 22.36963
## Oct 2019      16.261252 14.788913 17.73359 14.009504 18.51300
## Nov 2019      11.782247 10.250481 13.31401  9.439613 14.12488
## Dec 2019       9.451858  7.908565 10.99515  7.091594 11.81212
## Jan 2020       8.732563  7.177827 10.28730  6.354800 11.11032
## Feb 2020       8.970131  7.404038 10.53622  6.574998 11.36526
## Mar 2020      10.503103  8.925733 12.08047  8.090724 12.91548
## Apr 2020      12.740432 11.151866 14.32900 10.310930 15.16993
## May 2020      15.344559 13.744875 16.94424 12.898054 17.79106
## Jun 2020      19.151450 17.540725 20.76217 16.688058 21.61484
## Jul 2020      21.708406 20.086715 23.33010 19.228243 24.18857
## Aug 2020      22.732955 21.100371 24.36554 20.236134 25.22978
## Sep 2020      20.136368 18.492964 21.77977 17.622999 22.64974
## Oct 2020      16.261252 14.607099 17.91541 13.731443 18.79106
```

```
forecast:::plot.forecast(ts.p1.forecasts2)
```



The forecasts for the next 2 years are represented as a blue line, the 80% prediction interval

as a dark grey shaded area and the 95% prediction interval as a grey shaded area.

Forecast errors

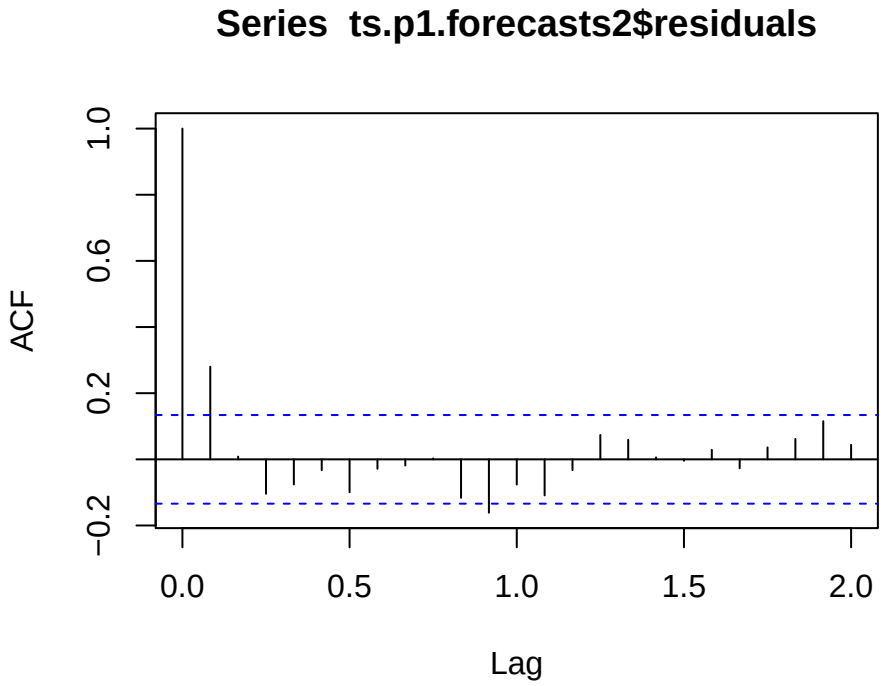
Errors are stored in `residuals`.

- If the model cannot be improved, there should be no correlations between forecast errors for successive predictions.
- If there are correlations between forecast errors for successive predictions, it is likely that the predictions can be improved with another technique.

To find out which is the case, we can obtain a correlation of the errors. We can calculate it using the `acf` function. `acf` calculates the estimate of the autocovariance or autocorrelation function *(ACF)*.

Let's see what happens, we use `acf` with residuals and lag max 24.

```
# na.action, to take or not to take into account the NA values
acf(ts.p1.forecasts2$residuals, na.action = na.pass, lag.max = 24)
```



We performed a Ljung Box text to see if there is significant evidence of non-zero correlations in the **lag**.

```
Box.test(ts.p1.forecasts2$residuals, lag = 24, type = "Ljung-Box")
##
##  Box-Ljung test
##
## data:  ts.p1.forecasts2$residuals
## X-squared = 41.723, df = 24, p-value = 0.01386
```

As we can see, the p-value is 0.01, which tells us that there is evidence of autocorrelations other than 0.

Note: The ACF (AutoCorrelation Function) chart is used to identify non-stationary time series. Stationary: if the ACF is rapidly declining

6. ARIMA models versus Holt-Winters models

The difference between exponential smoothing models and ARIMA models is that the former are based on a description of the trend and seasonality of the data, while **ARIMA models study the autocorrelations of the data.**

6.1 Characteristics

ARIMA (AutoRegressive Integrated Moving Average): - ARIMA decomposes a time series into three components: AutoRegressive (AR), Integrated (I), and Moving Average (MA). - ARIMA is particularly effective when dealing with stationary time series data, meaning that the statistical properties of the data do not change over time.

Holt-Winters: - Holt-Winters is a triple exponential smoothing method that decomposes a time series into three components: level, trend, and seasonality. - Holt-Winters is designed for time series data with trend and seasonality, making it a good choice for data that exhibits systematic patterns over time.

6.1.1 Stationarity

A stationary time series is one where the statistical properties of the data do not change over time, specifically with respect to time-dependent moments like mean, variance, and covariance.

```
 a. Constant Mean: The mean (average) of the time series data remains roughly constant ov
b. Constant Variance: The variance (spread or dispersion) of the data remains roughly con
c. Constant Autocovariance: The autocovariance (covariance between data points at differe
```

- A time series is said to be stationary when it does not depend on the instant of observation (white noise series).

  In a stationary time series, the behavior of the data is not influenced by when the data was collected. This means that the patterns and characteristics

of the data are consistent regardless of the time at which you look at it. White noise is a specific type of stationary time series where each data point is independently and identically distributed with a constant mean and constant variance. In other words, white noise is a stationary time series with no systematic patterns, trends, or correlations between data points.

- If a series has a trend or stationarity it is not stationary.
- A time series that has a cyclical component is stationary.

When a time series it exhibits these stationary characteristics, and it is often easier to analyze and model because the statistical properties do not change with time.

This is an important assumption in many time series analysis techniques, such as autoregressive integrated moving average (ARIMA) modeling.

If a time series is not stationary, it may require transformation (e.g., differencing) to make it stationary before applying certain analytical methods.

Stationarity with **ARIMA**?: - ARIMA models require that the time series be made stationary, often through differencing. - ARIMA is suitable for data with or without seasonality as long as it can be made stationary.

Stationarity with **Holt-Winters**?: - Holt-Winters can handle non-stationary data but is especially effective for time series with seasonality and trends.

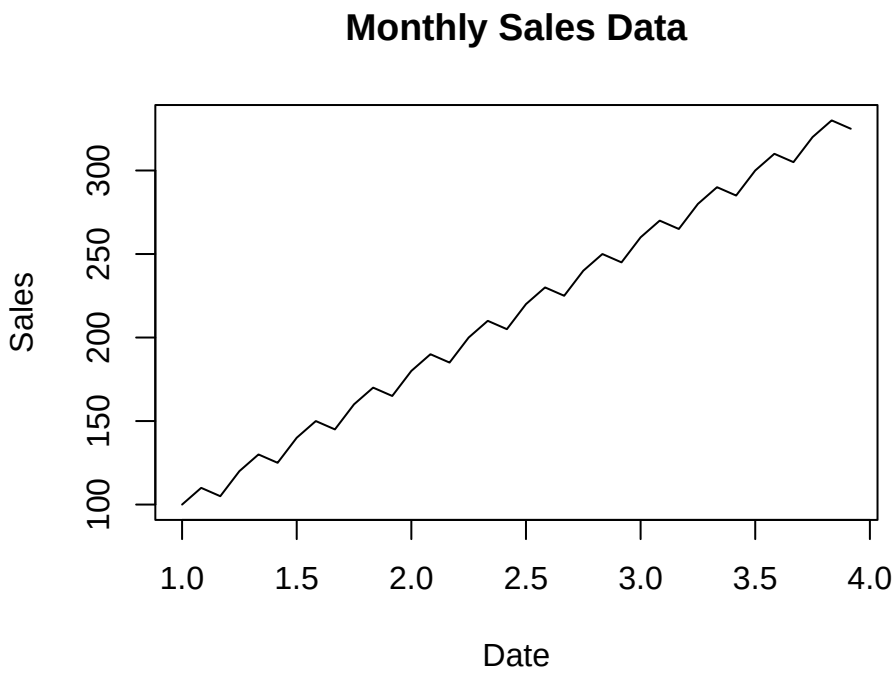
6.1.2 Stationary?

Example

From a dataset from sales, we check for stationarity using the Augmented Dickey-Fuller (ADF) test and fit an ARIMA model to the stationary data.

```
# Create a simple example dataset
date <- seq(as.Date("2020-01-01"), by = "1 month", length.out = 36)
sales <- c(100, 110, 105, 120, 130, 125, 140, 150, 145, 160, 170, 165,
           180, 190, 185, 200, 210, 205, 220, 230, 225, 240, 250, 245,
           260, 270, 265, 280, 290, 285, 300, 310, 305, 320, 330, 325)

# Create a data frame
sales_data <- data.frame(Date = date, Sales = sales)

# Save the data to a CSV file

library(tseries)
library(forecast)


# Convert the date column to a date object (adjust column name as needed)
sales_data$Date <- as.Date(sales_data$Date)

# Create a time series object
sales_ts <- ts(sales_data$Sales, frequency = 12)  # Assuming monthly data
```

```r
# Plot the time series
plot(sales_ts, main = "Monthly Sales Data", xlab = "Date", ylab = "Sales")
```




```r
# Check for stationarity
adf_test <- adf.test(sales_ts)
## Warning in adf.test(sales_ts): p-value smaller than printed p-value
if (adf_test$p.value < 0.05) {
  cat("The time series is stationary.\n")
} else {
  cat("The time series is not stationary. You may need to difference it.\n")
}
## The time series is stationary.
```

6.1.3 ADF test

2. ADF (Augmented Dickey-Fuller) Test:
    - Purpose: The ADF test is used to determine whether a time series is stationary or non-stationary. Stationarity is a critical assumption in time series modeling because many time series methods, like ARIMA, require data to be stationary.
    - How it works: The ADF test compares the observed data to a null hypothesis that the data has a unit root (non-stationary). It tests whether differencing the data (removing trends) makes it stationary.
    - Interpretation: The ADF test provides a test statistic and a p-value. If the p-value is less than a chosen significance level (e.g., 0.05), you can reject the null hypothesis and conclude that the data is stationary. If the p-value is greater, you may not be able to conclude stationarity.

The `adf.test` function will perform the ADF test on the `sales_ts` time series data. The results will include statistics and a p-value.

The p-value is a critical indicator to determine whether the time series is stationary. If the

p-value is less than your chosen significance level (e.g., 0.05), you can conclude that the time series is stationary.

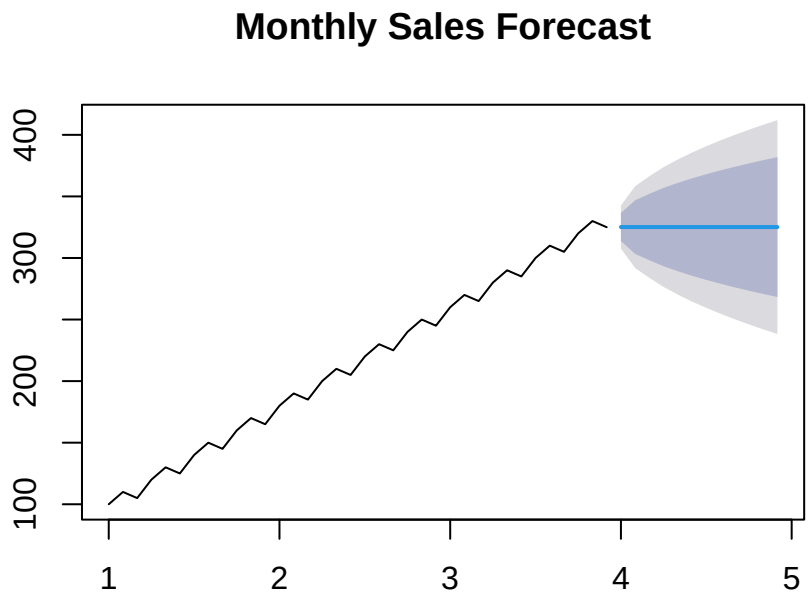
```
print(adf_test)
##
##  Augmented Dickey-Fuller Test
##
## data:  sales_ts
## Dickey-Fuller = -1.6224e+15, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

- p-value: The p-value is 0.01.
  This indicates that the stationarity of the time series is strong, and it doesn't require additional differencing.
- Alternative hypothesis: The alternative hypothesis is "stationary."
  The p-value is the key component to determine stationarity. In this case, the p-value is 0.01, which is less than the commonly chosen significance level of 0.05. When the p-value is less than your chosen significance level, you can reject the null hypothesis that the time series is non-stationary. Therefore, based on the low p-value of 0.01, you can conclude that your time series is stationary.
  The "Lag order" value of 3 in the Augmented Dickey-Fuller (ADF) test results indicates the number of lags used in the test to account for potential autocorrelation in the time series data. Specifically, it represents the number of past observations (time steps) that are considered when testing for stationarity.
  In this case, a lag order of 3 means that the test includes three lagged differences, which can be indicative of repeating patterns or seasonality in the time series. s

After fitting the model, we generate a forecast for future sales and plot the forecasted values.

```
# Fit an ARIMA model (adjust the order based on your analysis)
arima_model <- arima(sales_ts, order = c(1, 1, 1))

# Print model summary
summary(arima_model)
##
## Call:
## arima(x = sales_ts, order = c(1, 1, 1))
##
## Coefficients:
##           ar1     ma1
##       -0.3552  1.0000
## s.e.   0.1692  0.0792
##
## sigma^2 estimated as 77.43:  log likelihood = -127.21,  aic = 260.42
##
## Training set error measures:
##                    ME     RMSE      MAE      MPE     MAPE      MASE      ACF1
```

```
## Training set 4.232632 8.676369 5.934374 2.135967 3.156221 0.6020379 -0.5265618

# Forecast future sales (adjust the number of periods as needed)
forecast_periods <- 12  # Number of months to forecast
sales_forecast <- forecast(arima_model, h = forecast_periods)

# Plot the forecast
plot(sales_forecast, main = "Monthly Sales Forecast")
```




```
# Print the forecasted values
sales_forecast$mean
##         Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 4 325.1436 325.0926 325.1107 325.1043 325.1066 325.1058 325.1060 325.1059
##         Sep      Oct      Nov      Dec
## 4 325.1060 325.1060 325.1060 325.1060
```

6.1.4 Forecasting Approaches

ARIMA: - ARIMA models forecast future values based on past values, their lags, and moving averages of past forecast errors. - It models the autocorrelation between observations and their lags.

Holt-Winters: - Holt-Winters models account for three components: level, trend, and seasonality. - It uses a weighted average of these components to make forecasts and is particularly well-suited for data with strong seasonality.

6.1.5 Complexity Models

ARIMA: - ARIMA models are relatively simpler, as they involve setting orders for the AR, I, and MA components. - They can be adjusted by varying the orders (p, d, q) (see below).

Holt-Winters: - Holt-Winters models are more complex due to the triple exponential smoothing approach. - They require setting parameters for level, trend, and seasonality,

and potentially smoothing parameters.

6.1.6 Use Cases

ARIMA: - ARIMA is suitable for modeling time series data without strong seasonality, such as financial data or stock prices. - It is effective when the main focus is capturing lags and autocorrelation.

Holt-Winters: - Holt-Winters is ideal for data with clear seasonal patterns, like monthly sales, quarterly economic data, or daily weather observations. - It is particularly well-suited for forecasting when seasonality and trends are prominent.

In practice, the choice between ARIMA and Holt-Winters depends on the characteristics of your time series data.

It's important to analyze your data and determine whether it exhibits seasonality, trends, and stationarity, and then choose the method that best fits those characteristics.

Additionally, hybrid models, which combine ARIMA and Holt-Winters components, can also be used for more complex time series data.

6.2 ARIMA models

ARIMA is a popular time series forecasting method used to model and predict data that has temporal dependencies.

As we have already seen, the name ARIMA comes from *AutoRegressive Integrated Moving Average*.

- ***AR***: the study variable is analysed using the above values (*lag*).
  This component models the relationship between the current observation and a specified number of past observations. It captures the influence of past values on the present value. In ARIMA, this parameter is denoted as 'p.'
- **MA**: the regression error is a linear combination of current and past values of the signal.
  This component represents the number of differences needed to make the time series stationary. Differencing is the process of subtracting an observation from a lagged observation. In ARIMA, this parameter is denoted as 'd.'
- ***I***: replaces the value of the variable by the difference between the values and the previous values (one or more times).
  This component models the relationship between the current observation and a linear combination of past forecast errors. It accounts for the influence of past forecast errors on the present value. In ARIMA, this parameter is denoted as 'q.'

The combination of these three components (AR, I, and MA) determines the order of the ARIMA model, which is written as **ARIMA(p, d, q)**.

ARIMA models can be obtained following the **Box-Jenkins** method.

6.2.1 Examples

ARIMA(1, 0, 0)

Let's say we have monthly sales data for a store. To apply a simple ARIMA(1, 0, 0) model, we are using only the autoregressive component.

- AR component (p): 1
- No differencing (d): 0
- No moving average (q): 0

The model obtained will be:

$$Y_t = \phi_1 \cdot Y_{t-1} + C + \varepsilon_t$$

This means that the current observation depends only on the previous observation with coefficient ϕ_1.

This ARIMA model is relatively simple, and the forecast is based mainly on the previous observation.

```
library(forecast)

# Generate a simple example time series
sales_data <- c(100, 110, 105, 120, 130)

# Create an ARIMA(1,0,0) model
arima_model <- arima(sales_data, order = c(1, 0, 0))

# phi and C:
arima_model$coef
##         ar1   intercept
##    0.504016 113.577998

# Forecast future sales (adjust the number of periods as needed)
forecast_periods <- 3
sales_forecast <- forecast(arima_model, h = forecast_periods)

# Print the forecasted values
sales_forecast$mean
## Time Series:
## Start = 6
## End = 8
## Frequency = 1
## [1] 121.8549 117.7497 115.6806
```

In this example, we used only the autoregressive component to capture the relationship between the current sales and the previous month's sales.

ARIMA(0, 1, 1)

Now, let's consider a more complex ARIMA model with differencing and moving average components:

- AR component (p): 0

- One level of differencing (d): 1
  to make the time series stationary
- Moving Average component (q): 1
  to capture the influence of past forecast errors

The model obtained will be:

$$Y_t = Y_{t-1} + \phi_1 \cdot \varepsilon_{t-1} + \varepsilon_t$$

Coeficient ϕ_1 is the coeficient of the MA.

This ARIMA(0,1,1) model implies that future observations Y_t depend on past observations Y_{t-1} and a moving average component of order 1 $\phi_1 \cdot \varepsilon_{t-1}$.

The first order differentiation ($d = 1$) has been applied to make the series stationary before modelling. The error term (ε_t) is a residual component that captures the variability not explained by the past observations and the moving average term.

This ARIMA model is used to make forecasts of future temperatures based on past observations and the moving average component.

```
# Generate a simple example time series
temperature_data <- c(70, 72, 75, 78, 80, 82, 85, 88, 90, 92)

# Create an ARIMA(0,1,1) model
arima_model <- arima(temperature_data, order = c(0, 1, 1))

arima_model$coef
##       ma1
## 0.9999968
# Print the model summary
summary(arima_model)
##
## Call:
## arima(x = temperature_data, order = c(0, 1, 1))
##
## Coefficients:
##          ma1
##       1.0000
## s.e.  0.2901
##
## sigma^2 estimated as 1.733:  log likelihood = -16.4,  aic = 36.79
##
## Training set error measures:
##                     ME     RMSE      MAE      MPE     MAPE      MASE       ACF1
## Training set 1.151507 1.249193 1.151507 1.416502 1.416502 0.4710709 -0.0770721

# Forecast future temperatures (adjust the number of periods as needed)
forecast_periods <- 3
temperature_forecast <- forecast(arima_model, h = forecast_periods)

# Print the forecasted values
```

```
temperature_forecast$mean
## Time Series:
## Start = 11
## End = 13
## Frequency = 1
## [1] 93.19999 93.19999 93.19999
```

These examples illustrate how ARIMA models are constructed and applied to time series data for forecasting. The choice of ARIMA order (p, d, q) depends on the specific characteristics of the data and may require some analysis and experimentation.

Autoregressive models - AR(p)

The current value of the x_t series can be explained as a function of p past values, $x_{t-1}, x_{t-2}, x_{t-p}$, where p determines the number of past steps needed for the prediction.

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t$$

- ϵ_t: Gaussian white noise with mean 0 and variance $sigma_e^2psilon$.
- If the mean is non-zero, c takes the value $c = \mu(1 - \phi_1 - \cdots - \phi_p)$.

This is an autoregressive model of order p that applies to stationary models.

Moving average models

As an alternative to the autoregressive representation where x_t (on the left-hand side of the equation) are assumed to be linear combinations using moving averages of order q, $MA(q)$. We assume white noise $epsilon_t$. Model MA(q):

$$y_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}$$

As with autoregressive models, changing the parameters affects the time series patterns and changing the error term changes the scale.

Differentiation

To convert a non-stationary time series into a stationary time series, the differences between consecutive signal values are calculated (trend and seasonality are removed or reduced).

The following equation allows to calculate the differences between consecutive signal values:

$$y'_t = y_t - y_{t-1}$$

If the differentiated series is white noise:

$$y_t - y_{t-1} = \epsilon_t$$

(ϵ_t denotes white noise)

A closely related model allows the difference to have a non-zero mean.

$$y_t - y_{t-1} = c + \epsilon_t$$

where c is the average of the changes between consecutive observations.

Second order differencing

The difference of the data may have to be recalculated a second time to obtain a stationary series (the change in changes).

Seasonal differencing

A seasonal difference is the difference between one observation and the previous observation:

$$y'_t = y_t - y_{t-m}$$

where m is the number of seasons.

```
lynx <- read.csv(file =
  here::here("data",
  "annual-number-of-lynx-trapped-ma.csv"),
header = TRUE, quote = '\"')
lynx.ts <- ts(lynx$Lynx, frequency = 1, start = c(1821))
electricity <- read.csv(file =
  here::here("data",
  "monthly-electricity-production-i.csv"),
header = TRUE, quote = '\"')
electricity.ts <- ts(electricity$Million.kilowatt.hours, frequency = 12, start = c(1956,
```

```
plot.ts(lynx.ts, xlab = "Year", ylab = "Lynx Trapped")
```

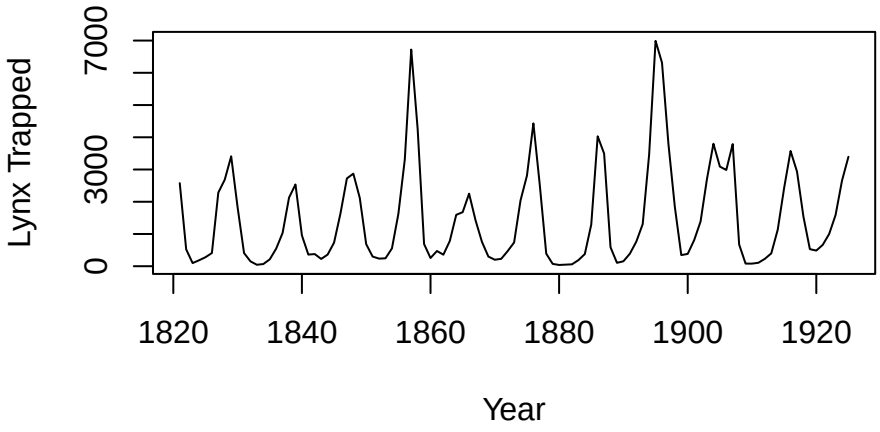


Figure 6.1.: Although there are cycles, they are not fixed in time.

```
plot.ts(electricity.ts, xlab = "Year", ylab = "kW/h")
```

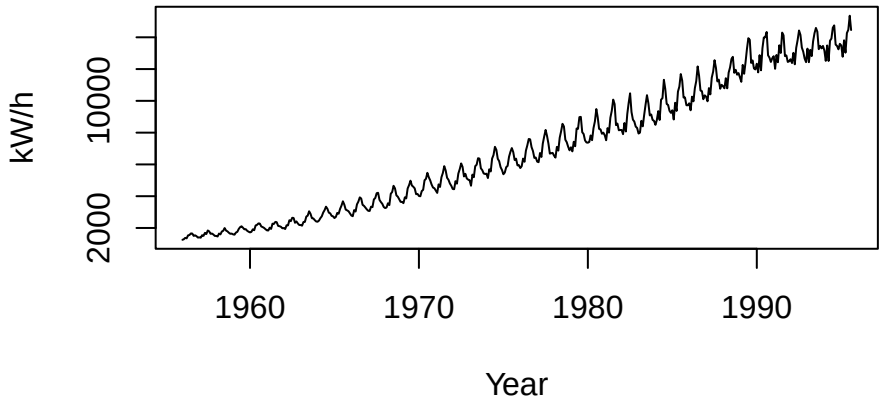


Figure 6.2.: Clear trend indicates this is not a stationary series.

```
par(mfrow = c(1, 2))
acf(lynx.ts)
acf(electricity.ts)
```

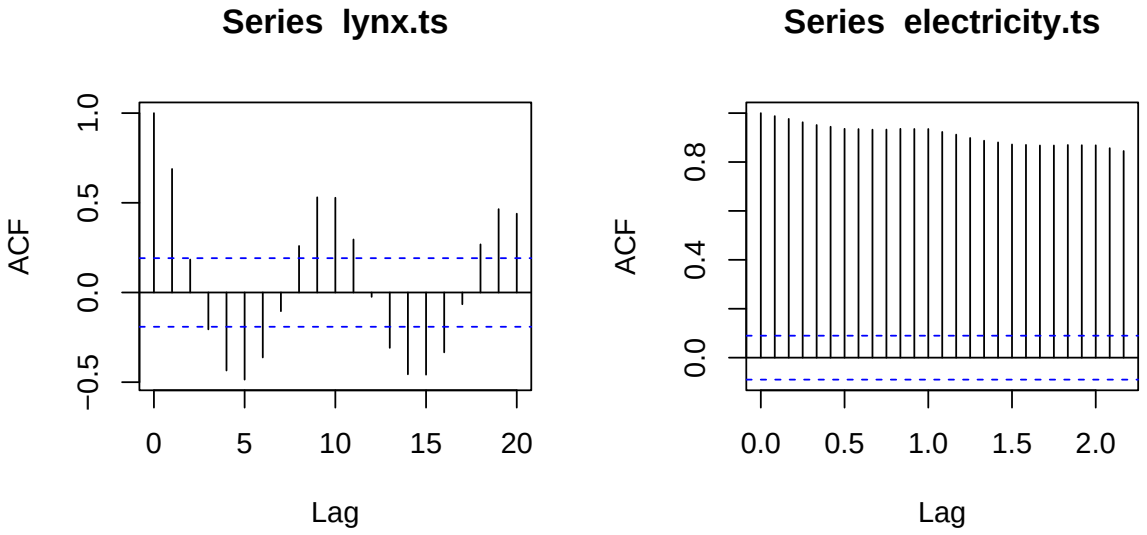


Figure 6.3.: Lynx trapped ACF plot (left) and kilowatts per hour in Australia ACF plot (right).

6.2.2 Parameters in ARIMA methods

Therefore, in an ARIMA model we will use three parameters: (p, d, q).

- **p**: refers to the use of past values of the series. It specifies the number of lags used in the model. For example, AR(2) or, ARIMA(2,0,0) specifies the use of two lags of the series.
- **d**: degree of differentiation. Differencing your current and previous values d times.
- **q**: model error as a combination of q previous error terms.

ARIMA works best for long and stable series.

6.2.3 Non-seasonal ARIMA models

If we combine differencing and autoregression we obtain a non-seasonal ARIMA model:

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t$$

where y'_t is the series to which differencing has been applied. These models are represented as ARIMA(p,d,q), where d represents the degree of differencing.

6.2.4 Auto-ARIMA model

In R we have functions that automatically calculate the parameter values of an ARIMA model.

The *`auto.arima()`* function is a modification of the **Hyndman-Khandakar** algorithm.

R's `auto.arima()` function combines tests, and minimisations to obtain an ARIMA model. By default, it can use initial approximations to speed up the model search (`approximation=TRUE/FALSE`).

If this does not allow to find a good model (minimum AICc) a larger set of models can be searched with the argument `stepwise=FALSE`. See [@forecasting] for more on this topic.

To choose a custom model for a non-seasonal series, we will use the `Arima()` function as follows:

1. Draw the dataset and identify outliers.
2. Transform the data to stabilise the variance.
3. If the data is not stationary use differencing until it is.
4. Analyse ACF/PACF for an ARIMA$(p,d,0)$ or ARIMA$(0,d,q)$ model.
5. Try to pick your model and use AICc to find the best one.
6. Check that the residuals behave like white noise:

- if this is not satisfied look for another model
- if this is true, we can use the model to predict

6.3 A use case

If your time series is not stationary, you should take steps to make it stationary before applying time series analysis techniques like ARIMA, Holt-Winters, or other methods. Stationarity is a fundamental assumption for many time series models, and non-stationary data can lead to unreliable results. Here's what you should do:

1. **Identify Non-Stationarity**:
   - First, ensure that you have identified the non-stationarity in your time series data. Common signs of non-stationarity include trends (long-term changes) and seasonality (repeating patterns).

For instance:

```
library(tseries)
adf_test_result <- adf.test(time_series_data)
```

or use Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF)

```
acf(time_series_data)
pacf(time_series_data)
```

or use seasonal decomposition techniques (e.g., stl()) to break down your time series into its trend, seasonal, and residual components. If you observe strong trends or seasonality, it suggests non-stationarity.

```
decomposed_data <- stl(time_series_data, s.window = "periodic")
plot(decomposed_data)
```

2. **Differencing**:
    - The most common method to achieve stationarity is differencing. You can difference the data by subtracting each data point from the previous one (first-order differencing). Repeat this process until the data becomes stationary. For example, you can use the `diff()` function in R.

For instance:

```
# Example time series data (replace with your actual data)
time_series_data <- c(100, 110, 105, 120, 130, 135, 140, 150)

# First-order differencing
first_order_difference <- diff(time_series_data, lag = 1)

# The 'first_order_difference' variable now contains the differenced values

# Second-order differencing
second_order_difference <- diff(time_series_data, lag = 1, differences = 2)

# The 'second_order_difference' variable contains the result of second-order differencing

# Print or visualize the differenced data
print(first_order_difference)
## [1] 10 -5 15 10  5  5 10
print(second_order_difference)
## [1] -15  20  -5  -5   0   5
# Adjust the lag and differences parameters according to the specific characteristics of
```

3. **Log Transformation**:
    - In some cases, taking the logarithm of the data can help stabilize variance and reduce non-stationarity, especially when you have exponential growth.

For instance:

```
log_transformed_data <- log(original_data)
```

4. **Seasonal Differencing**:
    - If your data exhibits seasonality, consider seasonal differencing by subtracting values from the same season in the previous year or period.

For instance:

```
seasonal_difference <- diff(time_series_data, lag = 12)
```

5. **Remove Trends**:
   - If you identify a trend in your data, remove it by fitting a trend model (e.g., linear or polynomial) and subtracting it from the data. You can also use rolling means or moving averages.

For instance:

```
library(tseries)
first_order_difference <- diff(time_series_data, lag = 1)
```

6. **Use Seasonal Decomposition**:
   - Seasonal decomposition techniques, like seasonal decomposition of time series (STL) or classical decomposition, can help in separating the trend and seasonality from the data.

For instance:

- Seasonal decomposition with STL

```
decomposed_data <- stl(time_series_data, s.window = "periodic")
detrended_data <- decomposed_data$time.series[, "trend"]
```

7. **Augmented Dickey-Fuller Test**:
   - Perform an Augmented Dickey-Fuller (ADF) test to quantitatively test for stationarity. The test determines whether your data is stationary or not. If the p-value is less than a chosen significance level (e.g., 0.05), you can consider the data stationary.
8. **Select Appropriate Parameters**:
   - Once the data is stationary, you can identify appropriate parameters (e.g., p, d, q) for ARIMA modeling.

   Visual Inspection Systematically try different combinations of parameter values (p, d, q) and evaluate model fit. Expert Knowledge Use auto.arima()
9. **Model Building and Forecasting**:
   - With stationary data and appropriate parameters, you can proceed with building your time series model (e.g., ARIMA) and make forecasts.
10. **Evaluate and Validate**:
    - After modeling, evaluate your results, validate the model, and use it for forecasting or analysis.

Remember that the process of making a time series stationary may involve trial and error. It's essential to understand the nature of your data and apply the appropriate transformations or differencing to achieve stationarity.

Additionally, some data may require more advanced techniques like seasonal adjustment, structural break analysis, or transformation of the underlying process to achieve stationarity.

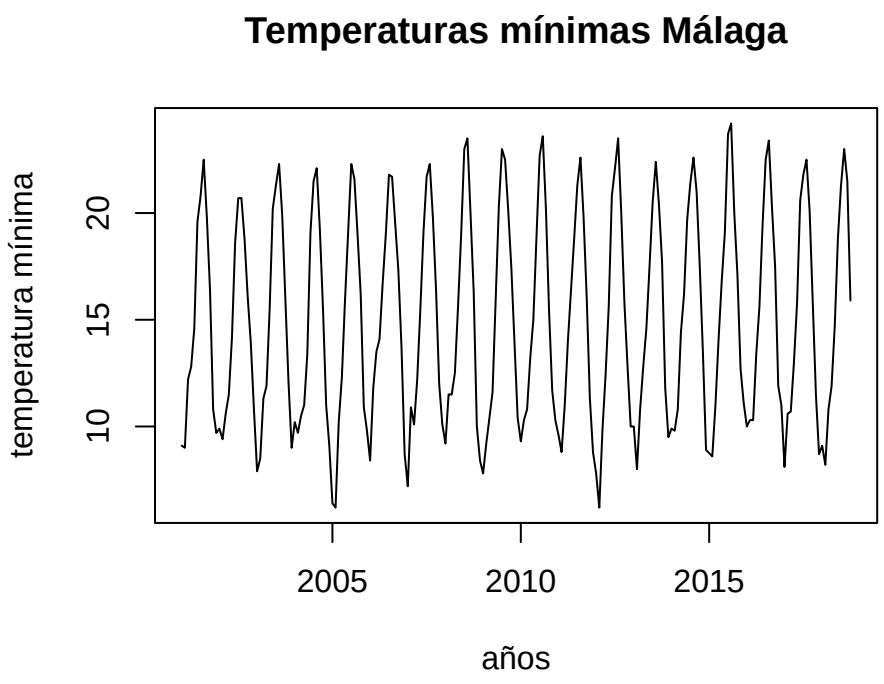
6.4 Example

Analyse the evolution of the minimum temperatures since 2001 in Malaga **with the methods seen in this document**.

We obtain a time-series object with the `ts` function, as our data are divided into months, we set the parameter *frequency* to 12 and we indicate that we start in January 2001. Then we draw the object obtained:

```
library('ggplot2')
library('TTR')
library('forecast')
library('tseries')
```

```
library(readr)
temperaturas <- read_csv(
  here::here("data",
  "temp.csv"))
## Rows: 214 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): fecha, indicativo
## dbl (4): X1, X1_1, tm_max, tm_min
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
ts.p1 <- ts(temperaturas$tm_min, start = c(2001, 1), frequency = 12)
plot.ts(ts.p1, xlab = "años", ylab = "temperatura mínima", main = "Temperaturas mínimas M
```



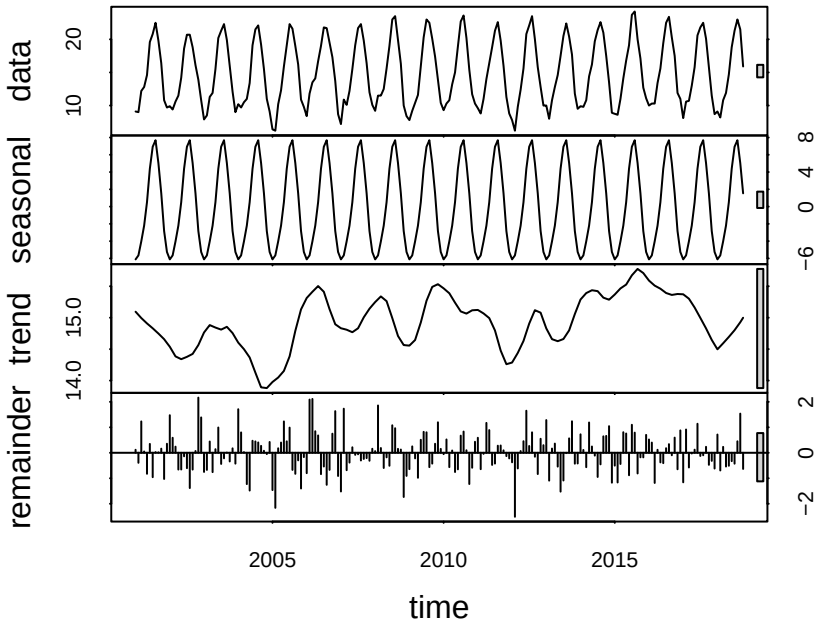
ARIMA models

We decompose the data: *Seasonal component*, *Trend component*, *Cycle component* y *residual*.

- we obtain a time series with a frequency of 12 months
- descompose
- decomposing the series and removing the seasonality can be done with seasadj()

within the forecast package

```
obj.ts <- ts(na.omit(temperaturas$tm_min), frequency = 12, start = c(2001, 1))
decomp <- stl(obj.ts, s.window = "periodic") # uses additive model by default
deseasonal_cnt <- seasadj(decomp)
plot(decomp)
```



Previously we have seen that the series is stationary, but to fit an ARIMA model requires the series to be stationary.

- with ADF test we can know if it is stationary or not.

```
adf.test(obj.ts, alternative = "stationary")
## Warning in adf.test(obj.ts, alternative = "stationary"): p-value smaller than
## printed p-value
##
##  Augmented Dickey-Fuller Test
##
## data:  obj.ts
## Dickey-Fuller = -14.33, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

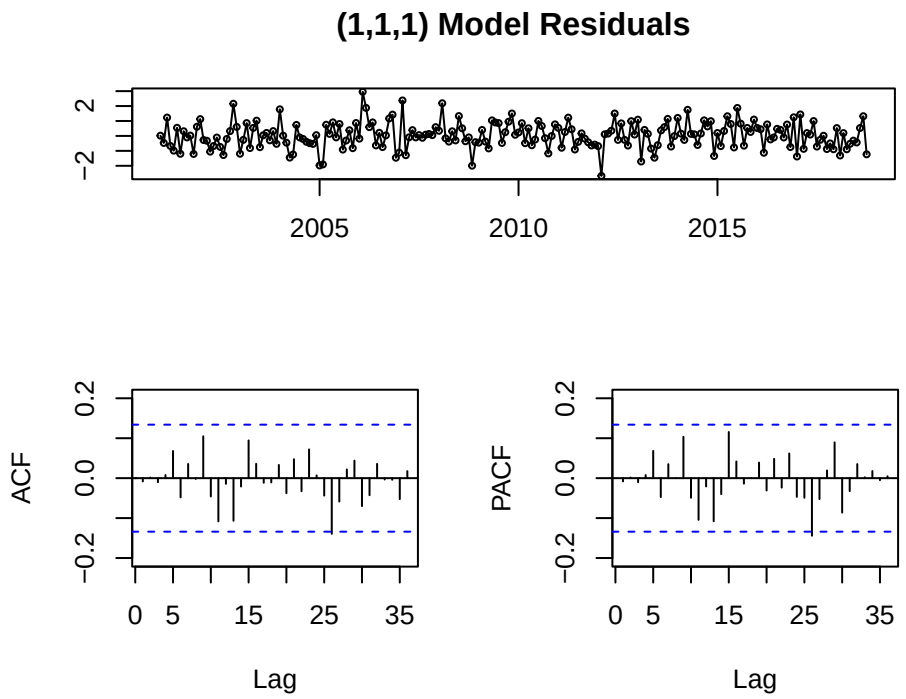
We see that it is stationary.

We fix an ARIMA model using auto.arima:

```
fit <- auto.arima(deseasonal_cnt, seasonal = FALSE)
fit
## Series: deseasonal_cnt
## ARIMA(1,1,1)
##
## Coefficients:
##          ar1      ma1
##       0.3684  -0.9799
```

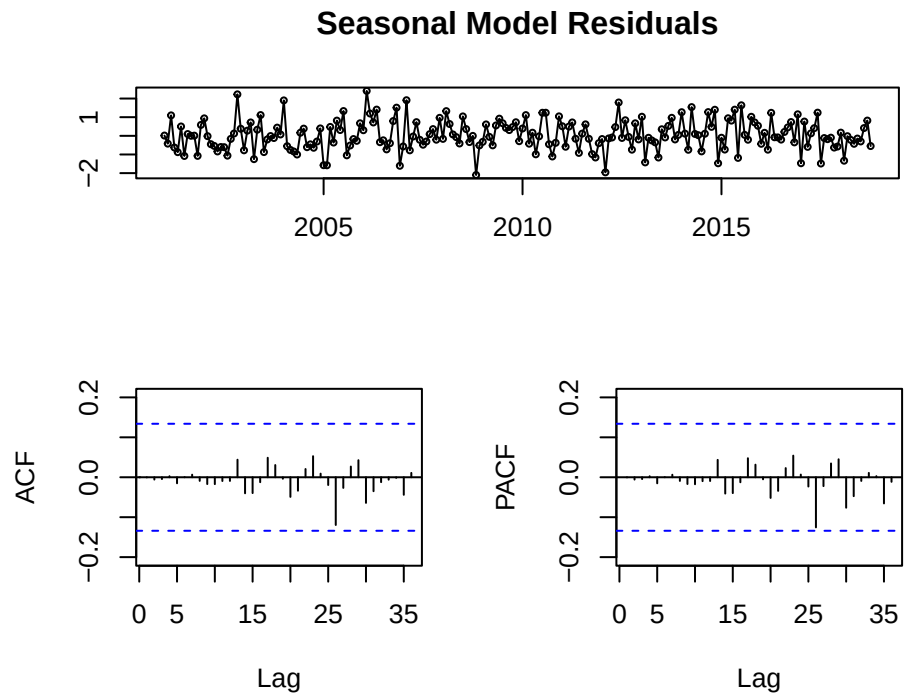
```
## s.e.  0.0668   0.0175
##
## sigma^2 = 0.7702:  log likelihood = -274.66
## AIC=555.32   AICc=555.44   BIC=565.41
```

```
tsdisplay(residuals(fit), main = "(1,1,1) Model Residuals")
```



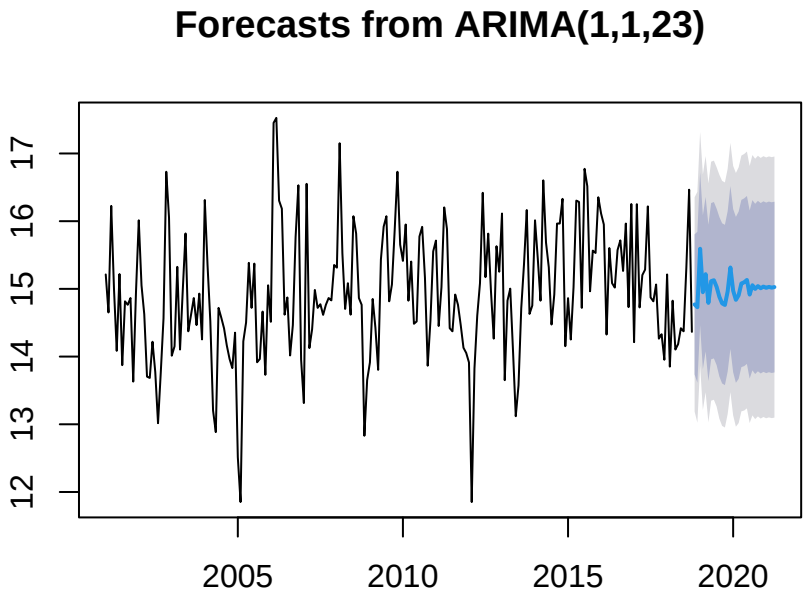
As we can see, even lag=23 is equal, so p or q is 23.

```
fit2 <- arima(deseasonal_cnt, order = c(1, 1, 23))

fit2
##
## Call:
## arima(x = deseasonal_cnt, order = c(1, 1, 23))
##
## Coefficients:
##           ar1      ma1      ma2      ma3      ma4      ma5      ma6      ma7
##       -0.7624  0.1627  -0.6937  -0.1923  -0.0753  0.0409  -0.1322  -0.0993
## s.e.   0.2475  0.2563   0.1611   0.1147   0.0880  0.0930   0.1067   0.1014
##           ma8      ma9     ma10     ma11     ma12     ma13     ma14     ma15     ma16
##       0.0485  0.2009  0.0295  -0.2316  -0.1224  -0.1785  0.0240  0.3970  0.1431
## s.e.  0.0936  0.0982  0.0963   0.1006   0.0939   0.0922  0.0978  0.1027  0.1168
##          ma17     ma18    ma19    ma20    ma21     ma22     ma23
##       -0.1927  -0.2143  0.0691  0.1088  0.1423  -0.0907  -0.0931
## s.e.   0.0902   0.1099  0.0969  0.1216  0.1053   0.1051   0.1117
##
## sigma^2 estimated as 0.6437:  log likelihood = -261.09,  aic = 572.19

tsdisplay(residuals(fit2), main = "Seasonal Model Residuals")
```



We forecast for the next 30 months:

```
fcast <- forecast(fit2, h = 30)
plot(fcast)
```



6.4.1 References

Information and images obtained from:

Forecasting: Principles and Practice. Rob J Hyndman and George Athanasopoulos. Monash University, Australia

Introducion to Forecasting with ARIMA in R

AEMET

II Rule-based methods

7. Association rules

- Unsupervised learning: a technique that allows learning from observations to extract patterns, trends and make predictions - **without prefixing a target**.
- Among the most relevant problems - **prediction of customer behaviour, shopping trends**: online shops, online services offering music, movies, etc.
- Current reality is that there is a lot of competition, many similar services, online marketplaces, etc.
- Objective: **Attracting customers is the key**.
- Means: data collected from consumers, service users, consumer characteristics extracted from the collected data, previous purchase or usage patterns, etc.
- Tools: Data Science methods - **search for frequent patterns, rules, etc.**.

Applications:

- Help personalise services and the shopping experience, guessing and suggesting shopping trends from *likes, dislikes*.
- Analyse combinations of products that people tend to buy together.
- Analyse reviews and prices that competitors offer for the same products.
- Discovering customer habits by finding relationships between different items that consumers place in their shopping basket in the items that consumers place in their shopping basket in large supermarkets.
- Analysis of purchases in large on-line shops - Amazon (thousands of products with millions of consumers), Google, etc., with the aim of (thousands of products with millions of consumers), Google, etc., with the objective of to offer recommendations.
- Detect relationships to provide suggestions between songs, series, films, etc. that a consumer is interested in, films, etc. that a user listens to/watches in an online music internet application, or online music, or video internet application - Spotify, Netflix (25 million users), HBO, etc.
- Control the peak hours of services.
- Marketing, catalogue design.

- Loan analysis, fraud detection, etc.

The problems in this area arise from the so-called **Market Basket Analysis** which deals with how to make product-based recommendations. Solutions for this problem can be used in other similar problems: recommendations of people's interests, etc.

The most important techniques are:

- Evaluation of the contingency matrix of the products.
- Generation of frequent itemsets.
- Extraction of association rules.

7.1 Detecting and predicting trends

- Trend: specific pattern or buying and selling behaviour that appears over a period of time in a shop.
- How to detect: Store all transactions that take place in the shop.
  - items purchased
  - stocks
  - combinations of items bought together
  - transaction of each sale made
- How to process the data:
  - pre-process
  - normalise
  - aggregate
- Apply algorithms: spot patterns and trends.
- Recommend: use patterns and trends to suggest new purchases.

The main method to achieve all this is based on the so-called **Market Basket Analysis** problem. These methods are based on probability-based statistics and probabilistic notions such as: * support * confidence * lift, etc.

Objective: **Which items (products) have been purchased most frequently**?

7.2 Fundamentals

7.2.1 Pattern search

Objective: to find relationships between data: market baskets, graphs, streams, etc.

- They are the core element in many techniques in many different areas from database, AI, data mining, machine learning, etc. database, AI, data mining, machine learning, etc.
- Frequent pattern search methods emerge in the area of data mining.
- The first algorithm to be highlighted was the so-called Apriori algorithm.

They arise with the study of the so-called Market Basket Analysis.

> Find coincidences between products purchased at the same time.
>
> Predict when the purchase of one product may increase the purchase of others.

Stories - Beer and Diapers

7.3 Structure of transactions

- Data: Typically a binary sparse D matrix of very large size.
- Item: Each dataset column. A piece of data in the dataset that can be an attribute, a product, a disease, a word, a document, a biomarker, etc.
- Transaction: Each row of the dataset typically representing a buyer, a pacient, a tweet, words in a document, genes, etc.

- Itemset: A collection of zero or more items.

$D_{i,j}$ represents item j in transaction i.

Objective: Given a new bit vector representing what the consumer has bought, the aim is to predict what other products the consumer might be interested in buying.

- Recommender systems.
- Modelling dependencies in complex sw complex systems: if the consumer is interested in the product, it is possible to predict what other products he/she might be interested in buying.

Example

```
FormalContext with 5 objects and 6 attributes.
    frankfurter  sausage  liver loaf  ham  meat  finished products
 t1                          X            X           X
 t2                 X
 t3                 X                 X               X
 t4                                         X
 t5                                                     X
```

7.3.1 First problem: Frequent Itemsets

The problem is to determine which sets of items occur jointly in the dataset. For instance, buying {beer, diapers, milk}.

This is a hard problem since we have to search a very complex space of possible itemsets.

But, we will have a good structure: a lattice. On the top we have the items (products), and in the second level combination of two-items, etc.

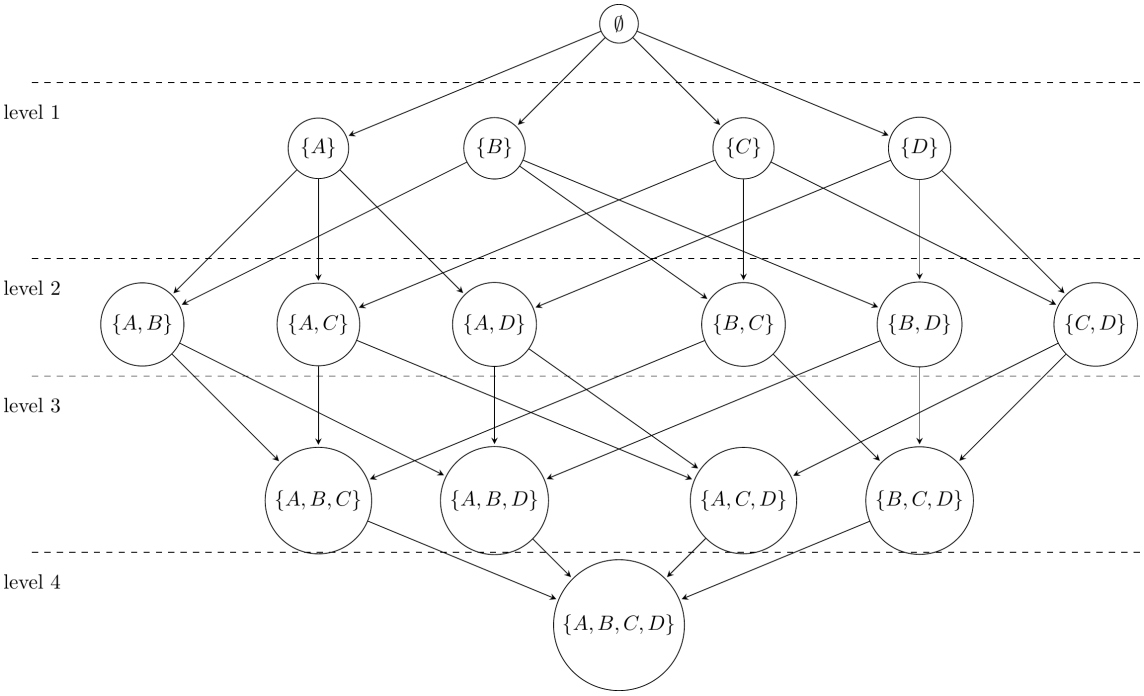


Figure 7.1.: Lattice

The idea is to simplify the search space by imposing a minimum support: **A measure of the percentage of the pattern found in relation to the size of the dataset.**

Notion: **Frequent itemset - Set a threshold for the support that must be satisfied by the itemset found.**

Support of an itemset is the proportion of transactions that have that itemset in the whole table:

$$Sup(X) = \frac{|X|}{|D|}$$

Prune: In this case, we can simplify the search, since if an itemset X is not frequent, any superset of X will not be frequent:

For instance, if A is not frequent we can prune the search space:

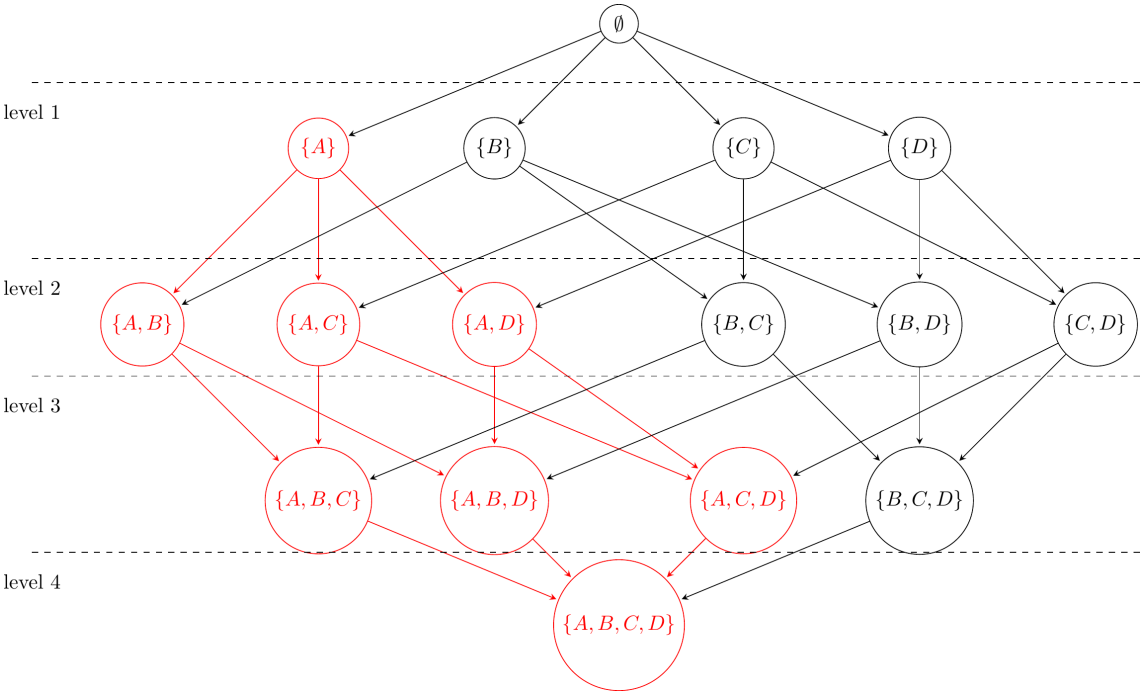


Figure 7.2.: Pruned Lattice

One of the earliest (if not the first) algorithm to compute frequent itemsets and association rules is `apriori()` (implemented in package `arules`).

It first creates the length 1 itemsets. Remove the infrequent ones (those not obtaining the minimum required support). then, from the _survivors, create *length 2* itemsets, and repeat.

7.3.2 Second problem: Association rules

An *association rule* is a relationship between attributes (items) representing a frequent pattern, i.e. a structural correlation in the table. We denote a rule as $X \to Y$. X is called *left hand side* or *antecedent* and Y is the *right hand side* or *consequent*.

Inside of the frequent patterns, we discover the rules.

$$Diapers, Milk, Beer$$

is a frequent pattern. What rules are interesting?

For instance,

$$Diapers, Milk \longrightarrow Beer$$

is a rule that can be read "if a person buys diapers and milk, very often he also buys beer".

Quality of a rule

We have to define some measures of the quality or the interestingness of a rule $X \to Y$:

- *Support*: $Sup(X \to Y) = Sup(X)$. It is the frequency of the left hand side. A high value indicates that the rule is applicable to a large part of the database. A low value indicates an infrequent rule. Rules with low support may be removed.

- *Confidence*: $Conf(X \to Y) = \dfrac{Sup(X \cup Y)}{Sup(X)} = \dfrac{|X \cup Y|}{|X|}$. It is an estimator of $P(Y|X)$, that is, of the probability in the dataset of having Y once we know we have X. Indicator of the reliability of the rule.

The confidence of a rule can be very high but the support of the consequent very low. The definition of confidence does not take into account how many times the consequent itemset appears in the table.

Other measures:

- *Lift*: $Lift(X \to Y) = \dfrac{Sup(X \cup Y)}{Sup(X)Sup(Y)}$. For a good rule, it should be greater than 1.
- *Conviction*: $Conv(X \to Y) = \dfrac{Sup(X)Sup(\overline{Y})}{Sup(X \cup \overline{Y})}$. It approaches 1 as X and Y are not related.

But, how do we compute association rules?

There are many possible rules ($3^d - 2^{d+1} + 1$, in a dataset with d items). Instead of using a brute force approach, we begin by using the already computed frequent itemsets.

In addition, we usually set a minimum support and minimum confidence for the extracted rules: this allows to reduce the number of rules to be extracted.

Procedure using A priori:

First, we find all frequent itemsets.

Then, for each itemset found:

1. Suppose the itemset is $X = \{x_1, x_2, \ldots, x_n\}$ where x_i are the different items inside the itemset X.
2. Consider all rules of the form:

$$\{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\} \to \{x_i\}$$

3. Compute support and confidence for each rule.
4. Those rules with less support or confidence are rejected.

Example

Suppose a frequent itemset $X = \{beer, diapers, milk\}$. We impose a minimum support of 0.4 and minimum confidence of 0.75.

The three possible rules are:

(1) $\{beer, diapers\} \to \{milk\}$
(2) $\{beer, milk\} \to \{diapers\}$
(3) $\{diapers, milk\} \to \{beer\}$

Suppose that we compute their support and confidence in our dataset:

$Sup(\{beer, diapers\} \to \{milk\}) = 0.8$, $Conf(\{beer, diapers\} \to \{milk\}) = 0.5$ $Sup(\{beer, diapers\} \to \{milk\}) = 0.3$, $Conf(\{beer, diapers\} \to \{milk\}) = 0.8$ $Sup(\{diapers, milk\} \to \{beer\}) = 0.55$, $Conf(\{diapers, milk\} \to \{beer\}) = 0.9$

With this information, we reject rules (1) and (2), and store the rule (3), since it verifies the requirements of support and confidence.

7.4 Patterns from binary transactions

Exploring the `Adult` dataset

```
library(arules)
data("Adult")
length(Adult)
## [1] 48842
dim(Adult)
## [1] 48842   115
Adult
## transactions in sparse format with
##  48842 transactions (rows) and
##  115 items (columns)
inspect(Adult[1:2])
##     items                                   transactionID
## [1] {age=Middle-aged,
##      workclass=State-gov,
##      education=Bachelors,
##      marital-status=Never-married,
##      occupation=Adm-clerical,
##      relationship=Not-in-family,
##      race=White,
##      sex=Male,
##      capital-gain=Low,
##      capital-loss=None,
##      hours-per-week=Full-time,
##      native-country=United-States,
##      income=small}                           1
## [2] {age=Senior,
##      workclass=Self-emp-not-inc,
##      education=Bachelors,
##      marital-status=Married-civ-spouse,
##      occupation=Exec-managerial,
##      relationship=Husband,
##      race=White,
##      sex=Male,
##      capital-gain=None,
##      capital-loss=None,
##      hours-per-week=Part-time,
##      native-country=United-States,
##      income=small}                           2
```

To compute the association rules:

```
data("Adult")
rules <- apriori(Adult, parameter = list(
  supp = 0.5, conf = 0.9,
  target = "frequent itemsets"
))
## Apriori
```

```
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##          NA    0.1    1 none FALSE            TRUE       5     0.5      1
##  maxlen            target  ext
##      10 frequent itemsets TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 24421
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[115 item(s), 48842 transaction(s)] done [0.02s].
## sorting and recoding items ... [9 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 done [0.00s].
## sorting transactions ... done [0.00s].
## writing ... [49 set(s)] done [0.00s].
## creating S4 object  ... done [0.00s].

rules <- Adult %>%
        apriori(parameter = list(
            supp = 0.5, conf = 0.9,
            target = "rules"
))
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.9    0.1    1 none FALSE            TRUE       5     0.5      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 24421
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[115 item(s), 48842 transaction(s)] done [0.02s].
## sorting and recoding items ... [9 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [52 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
summary(rules)
## set of 52 rules
##
## rule length distribution (lhs + rhs):sizes
##  1  2  3  4
##  2 13 24 13
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   2.923   3.250   4.000
##
## summary of quality measures:
##     support         confidence        coverage            lift
##  Min.   :0.5084   Min.   :0.9031   Min.   :0.5406   Min.   :0.9844
##  1st Qu.:0.5415   1st Qu.:0.9155   1st Qu.:0.5875   1st Qu.:0.9937
##  Median :0.5974   Median :0.9229   Median :0.6293   Median :0.9997
##  Mean   :0.6436   Mean   :0.9308   Mean   :0.6915   Mean   :1.0036
##  3rd Qu.:0.7426   3rd Qu.:0.9494   3rd Qu.:0.7945   3rd Qu.:1.0057
##  Max.   :0.9533   Max.   :0.9583   Max.   :1.0000   Max.   :1.0586
##      count
##  Min.   :24832
##  1st Qu.:26447
##  Median :29178
##  Mean   :31433
##  3rd Qu.:36269
##  Max.   :46560
##
## mining info:
##  data ntransactions support confidence
##     .         48842     0.5        0.9
##                                                                                call
##  apriori(data = ., parameter = list(supp = 0.5, conf = 0.9, target = "rules"))
inspect(head(rules))
##     lhs                             rhs                    support   confidence
## [1] {}                           => {capital-gain=None}    0.9173867 0.9173867
## [2] {}                           => {capital-loss=None}    0.9532779 0.9532779
## [3] {hours-per-week=Full-time} => {capital-gain=None}    0.5435895 0.9290688
## [4] {hours-per-week=Full-time} => {capital-loss=None}    0.5606650 0.9582531
## [5] {sex=Male}                   => {capital-gain=None}    0.6050735 0.9051455
## [6] {sex=Male}                   => {capital-loss=None}    0.6331027 0.9470750
##     coverage  lift      count
## [1] 1.0000000 1.0000000 44807
## [2] 1.0000000 1.0000000 46560
## [3] 0.5850907 1.0127342 26550
## [4] 0.5850907 1.0052191 27384
## [5] 0.6684820 0.9866565 29553
## [6] 0.6684820 0.9934931 30922
```

parameter: list of restrictions on the extraction process

- *supp* or *support*

- *conf* or *confidence*
- *minlen* - minimum number of items in itemset
- *maxlen* - maximum number of items in itemset
- *maxtime* - time limit
- *target* - indicate what kind of associations we want to extract: "rules", "frequent itemsets", "maximally frequent itemsets", "closed frequent itemsets".

```
rules <- apriori(Adult, parameter = list(supp = 0.1, conf = 1, minlen = 2))
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##           1    0.1    1 none FALSE            TRUE       5     0.1      2
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 4884
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[115 item(s), 48842 transaction(s)] done [0.02s].
## sorting and recoding items ... [31 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 5 6 7 8 9 done [0.05s].
## writing ... [72 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
inspect(tail(rules))
##     lhs                                      rhs          support confidence  coverage
## [1] {relationship=Husband,
##      race=White,
##      capital-gain=None,
##      capital-loss=None,
##      hours-per-week=Over-time,
##      native-country=United-States}       => {sex=Male} 0.1027394          1 0.1027394
## [2] {age=Senior,
##      marital-status=Married-civ-spouse,
##      relationship=Husband,
##      race=White,
##      capital-gain=None,
##      native-country=United-States}       => {sex=Male} 0.1083698          1 0.1083698
## [3] {age=Senior,
##      marital-status=Married-civ-spouse,
##      relationship=Husband,
##      race=White,
##      capital-loss=None,
##      native-country=United-States}       => {sex=Male} 0.1167233          1 0.1167233
## [4] {age=Senior,
```

```
##       marital-status=Married-civ-spouse,
##       relationship=Husband,
##       race=White,
##       capital-gain=None,
##       capital-loss=None}                    => {sex=Male} 0.1070800          1 0.1070800
## [5] {age=Senior,
##       marital-status=Married-civ-spouse,
##       relationship=Husband,
##       capital-gain=None,
##       capital-loss=None,
##       native-country=United-States}         => {sex=Male} 0.1072438          1 0.1072438
## [6] {marital-status=Married-civ-spouse,
##       relationship=Husband,
##       race=White,
##       capital-gain=None,
##       capital-loss=None,
##       hours-per-week=Over-time,
##       native-country=United-States}         => {sex=Male} 0.1026575          1 0.1026575
patterns <- apriori(Adult, parameter = list(supp = 0.5, conf = 0.9, maxlen = 10, target =
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##          NA    0.1    1 none FALSE            TRUE       5     0.5      1
##  maxlen            target  ext
##      10 frequent itemsets TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 24421
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[115 item(s), 48842 transaction(s)] done [0.02s].
## sorting and recoding items ... [9 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 done [0.00s].
## sorting transactions ... done [0.00s].
## writing ... [49 set(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
summary(patterns)
## set of 49 itemsets
##
## most frequent items:
##            capital-loss=None            capital-gain=None
##                           23                           21
## native-country=United-States                   race=White
##                           21                           19
```

```
##            workclass=Private                        (Other)
##                           14                             20
##
## element (itemset/transaction) length distribution:sizes
##  1  2  3  4
##  9 17 17  6
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   2.000   2.408   3.000   4.000
##
## summary of quality measures:
##     support           count
##  Min.   :0.5051   Min.   :24671
##  1st Qu.:0.5434   1st Qu.:26540
##  Median :0.5942   Median :29024
##  Mean   :0.6449   Mean   :31497
##  3rd Qu.:0.7404   3rd Qu.:36164
##  Max.   :0.9533   Max.   :46560
##
## includes transaction ID lists: FALSE
##
## mining info:
##   data ntransactions support confidence
##  Adult         48842     0.5          1
##
##  apriori(data = Adult, parameter = list(supp = 0.5, conf = 0.9, maxlen = 10, target =
inspect(tail(patterns))
##     items                              support count
## [1] {race=White,
##      sex=Male,
##      capital-loss=None,
##      native-country=United-States} 0.5113632 24976
## [2] {sex=Male,
##      capital-gain=None,
##      capital-loss=None,
##      native-country=United-States} 0.5084149 24832
## [3] {workclass=Private,
##      race=White,
##      capital-loss=None,
##      native-country=United-States} 0.5181401 25307
## [4] {workclass=Private,
##      race=White,
##      capital-gain=None,
##      capital-loss=None}            0.5204742 25421
## [5] {workclass=Private,
##      capital-gain=None,
##      capital-loss=None,
##      native-country=United-States} 0.5414807 26447
## [6] {race=White,
```

```
##       capital-gain=None,
##       capital-loss=None,
##       native-country=United-States} 0.6803980 33232
```

appearance: specify restrictions on the extraction of associations.

```
rules1 <- apriori(Adult,
  parameter = list(supp = 0.1, conf = 0.3),
  appearance = list(rhs = c("relationship=Husband"))
)
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.3    0.1    1 none FALSE            TRUE       5     0.1      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 4884
##
## set item appearances ...[1 item(s)] done [0.00s].
## set transactions ...[115 item(s), 48842 transaction(s)] done [0.02s].
## sorting and recoding items ... [31 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 5 6 7 8 9 done [0.05s].
## writing ... [660 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].

inspect(head(rules1))
##     lhs                                   rhs                          support confidence
## [1] {}                                 => {relationship=Husband} 0.4036690  0.4036690
## [2] {income=large}                     => {relationship=Husband} 0.1211662  0.7547507
## [3] {hours-per-week=Over-time}         => {relationship=Husband} 0.1472298  0.5672925
## [4] {age=Senior}                       => {relationship=Husband} 0.1479874  0.5673024
## [5] {education=HS-grad}                => {relationship=Husband} 0.1307891  0.4047136
## [6] {marital-status=Married-civ-spouse} => {relationship=Husband} 0.4034233  0.8804683

rules2 <- apriori(Adult,
  parameter = list(supp = 0.5, conf = 0.9),
  appearance = list(none = c("income=small", "sex=Male"))
)
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.9    0.1    1 none FALSE            TRUE       5     0.5      1
```

```
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 24421
##
## set item appearances ...[2 item(s)] done [0.00s].
## set transactions ...[115 item(s), 48842 transaction(s)] done [0.02s].
## sorting and recoding items ... [7 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [39 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].

inspect(head(rules2))
##     lhs                             rhs                  support   confidence
## [1] {}                          => {capital-gain=None} 0.9173867 0.9173867
## [2] {}                          => {capital-loss=None} 0.9532779 0.9532779
## [3] {hours-per-week=Full-time} => {capital-gain=None} 0.5435895 0.9290688
## [4] {hours-per-week=Full-time} => {capital-loss=None} 0.5606650 0.9582531
## [5] {workclass=Private}         => {capital-gain=None} 0.6413742 0.9239073
## [6] {workclass=Private}         => {capital-loss=None} 0.6639982 0.9564974
##     coverage  lift     count
## [1] 1.0000000 1.000000 44807
## [2] 1.0000000 1.000000 46560
## [3] 0.5850907 1.012734 26550
## [4] 0.5850907 1.005219 27384
## [5] 0.6941976 1.007108 31326
## [6] 0.6941976 1.003377 32431
```

7.5 Patterns from non binary datasets

Discretize

We are going to explor the `AdultUCI` dataset. It contains the data from the dataset originally called 'Census Income' Database in data.frame format.

The `Adult` dataset from the previous section has the data prepared for the computation of the rules. The data type of this dataset is **transactions** which is suitable for the arules package.

The `AdultUCI` dataset:

```
library(arules)
data("AdultUCI")
str(AdultUCI)
## 'data.frame':    48842 obs. of  15 variables:
```

```
##  $ age           : int  39 50 38 53 28 37 49 52 31 42 ...
##  $ workclass     : Factor w/ 8 levels "Federal-gov",..: 7 6 4 4 4 4 4 6 4 4 ...
##  $ fnlwgt        : int  77516 83311 215646 234721 338409 284582 160187 209642 45781 15
##  $ education     : Ord.factor w/ 16 levels "Preschool"<"1st-4th"<..: 14 14 9 7 14 15 5
##  $ education-num : int  13 13 9 7 13 14 5 9 14 13 ...
##  $ marital-status: Factor w/ 7 levels "Divorced","Married-AF-spouse",..: 5 3 1 3 3 3 4
##  $ occupation    : Factor w/ 14 levels "Adm-clerical",..: 1 4 6 6 10 4 8 4 10 4 ...
##  $ relationship  : Factor w/ 6 levels "Husband","Not-in-family",..: 2 1 2 1 6 6 2 1 2
##  $ race          : Factor w/ 5 levels "Amer-Indian-Eskimo",..: 5 5 5 3 3 5 3 5 5 5 ...
##  $ sex           : Factor w/ 2 levels "Female","Male": 2 2 2 2 1 1 1 2 1 2 ...
##  $ capital-gain  : int  2174 0 0 0 0 0 0 14084 5178 ...
##  $ capital-loss  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hours-per-week: int  40 13 40 40 40 40 16 45 50 40 ...
##  $ native-country: Factor w/ 41 levels "Cambodia","Canada",..: 39 39 39 39 5 39 23 39
##  $ income        : Ord.factor w/ 2 levels "small"<"large": 1 1 1 1 1 1 1 2 2 2 ...
```

For most datasets a first preprocessing step is necessary.

In the following, we will apply preprocessing to `AdultUCI` until it is converted into transactions that arules handles properly.

Discretisation of items

Delete some columns that are not interesting: `fnlwgt`, `education-num`

```
AdultUCI$fnlwgt <- NULL
## o  AdultUCI[["fnlwgt"]] <- NULL

AdultUCI$`education-num` <- NULL
```

Convert to discrete numerical values: `age`, `hours-per-week`, `capital-gain`, `capital-loss`.

We will use `cut()`, `ordered()` commands to do this. Here is an example of how these two commands work:

```
# ejemplo de funcionamiento de cut y ordered
v <- 1:100
v2 <- cut(v,
          c(0, 25, 50, 75, 100),
          labels = c("low", "medium", "high", "veryhigh"))
# ?ordered
v3 <- ordered(v2)
```

We apply these functions to `AdultUCI`:

```
AdultUCI$age <- ordered(cut(AdultUCI[["age"]], c(15, 25, 45, 65, 100)),
  labels = c("Young", "Middle-aged", "Senior", "Old")
)

AdultUCI[["hours-per-week"]] <- ordered(cut(
  AdultUCI[["hours-per-week"]],
  c(0, 25, 40, 60, 168)
```

```
),
labels = c("Part-time", "Full-time", "Over-time", "Workaholic")
)

AdultUCI[["capital-gain"]] <- ordered(cut(
  AdultUCI[["capital-gain"]],
  c(
    -Inf, 0, median(AdultUCI[["capital-gain"]][AdultUCI[["capital-gain"]] > 0]),
    Inf
  )
), labels = c("None", "Low", "High"))

AdultUCI[["capital-loss"]] <- ordered(cut(
  AdultUCI[["capital-loss"]],
  c(
    -Inf, 0, median(AdultUCI[["capital-loss"]][AdultUCI[["capital-loss"]] > 0]),
    Inf
  )
), labels = c("None", "Low", "High"))
```

We run `apriori()`:

```
reg <- apriori(AdultUCI)
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5     0.1      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 4884
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[115 item(s), 48842 transaction(s)] done [0.02s].
## sorting and recoding items ... [31 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 5 6 7 8 9 done [0.05s].
## writing ... [6137 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
inspect(head(reg))
##     lhs                         rhs                                support
## [1] {}                       => {race=White}                       0.8550428
## [2] {}                       => {native-country=United-States} 0.8974243
## [3] {}                       => {capital-gain=None}                0.9173867
```

```
## [4] {}                        => {capital-loss=None}            0.9532779
## [5] {relationship=Unmarried} => {capital-loss=None}            0.1019819
## [6] {occupation=Sales}        => {race=White}                   0.1005282
##     confidence coverage  lift     count
## [1] 0.8550428  1.0000000 1.000000 41762
## [2] 0.8974243  1.0000000 1.000000 43832
## [3] 0.9173867  1.0000000 1.000000 44807
## [4] 0.9532779  1.0000000 1.000000 46560
## [5] 0.9719024  0.1049302 1.019537  4981
## [6] 0.8920785  0.1126899 1.043314  4910
```

See this link.

We compare `AdultUCI` that we have processed with `Adult`..

```
Adult1 <- as(AdultUCI, "transactions")
class(Adult1)
## [1] "transactions"
## attr(,"package")
## [1] "arules"
length(Adult1)
## [1] 48842
dim(Adult1)
## [1] 48842   115
Adult1
## transactions in sparse format with
##  48842 transactions (rows) and
##  115 items (columns)
inspect(Adult1[1:2])
##     items                                    transactionID
## [1] {age=Middle-aged,
##      workclass=State-gov,
##      education=Bachelors,
##      marital-status=Never-married,
##      occupation=Adm-clerical,
##      relationship=Not-in-family,
##      race=White,
##      sex=Male,
##      capital-gain=Low,
##      capital-loss=None,
##      hours-per-week=Full-time,
##      native-country=United-States,
##      income=small}                                1
## [2] {age=Senior,
##      workclass=Self-emp-not-inc,
##      education=Bachelors,
##      marital-status=Married-civ-spouse,
##      occupation=Exec-managerial,
##      relationship=Husband,
##      race=White,
```

```
##       sex=Male,
##       capital-gain=None,
##       capital-loss=None,
##       hours-per-week=Part-time,
##       native-country=United-States,
##       income=small}                                    2
```

```
data("Adult")
class(Adult)
## [1] "transactions"
## attr(,"package")
## [1] "arules"
length(Adult)
## [1] 48842
dim(Adult)
## [1] 48842   115
Adult
## transactions in sparse format with
##  48842 transactions (rows) and
##  115 items (columns)
inspect(Adult[1:2])
##     items                                transactionID
## [1] {age=Middle-aged,
##      workclass=State-gov,
##      education=Bachelors,
##      marital-status=Never-married,
##      occupation=Adm-clerical,
##      relationship=Not-in-family,
##      race=White,
##      sex=Male,
##      capital-gain=Low,
##      capital-loss=None,
##      hours-per-week=Full-time,
##      native-country=United-States,
##      income=small}                                    1
## [2] {age=Senior,
##      workclass=Self-emp-not-inc,
##      education=Bachelors,
##      marital-status=Married-civ-spouse,
##      occupation=Exec-managerial,
##      relationship=Husband,
##      race=White,
##      sex=Male,
##      capital-gain=None,
##      capital-loss=None,
##      hours-per-week=Part-time,
##      native-country=United-States,
##      income=small}                                    2
```

7.6 Methods of arules package.

- summary(): Overview of the set of rules.
- length(): Number of rules.
- items(): Items involved.
- sort(): Sort
- subset(): Items involved. (see help(subset)). Select rules that meet certain criteria.
- union(), intersect(), setequal(), match() (use help(xxx)).

- write(): Writing more properly formatted rules.

Operations with rules:

```
data("Adult")
r1 <- apriori(Adult[1:1000], parameter = list(support = 0.5))
r2 <- apriori(Adult[1001:2000], parameter = list(support = 0.5))
# Convertir en un dataframe
dfr1 <- DATAFRAME(r1)
r_comb <- c(r1, r2)
duplicated(r_comb)
intersect(r1, r2)
union(r1, r2)
lhs(reglas1)
rhs(reglas1)
class(lhs(reglas1))
```

7.7 arulesViz package

Let's use the Groceries dataset to look at the most interesting association rule display commands.

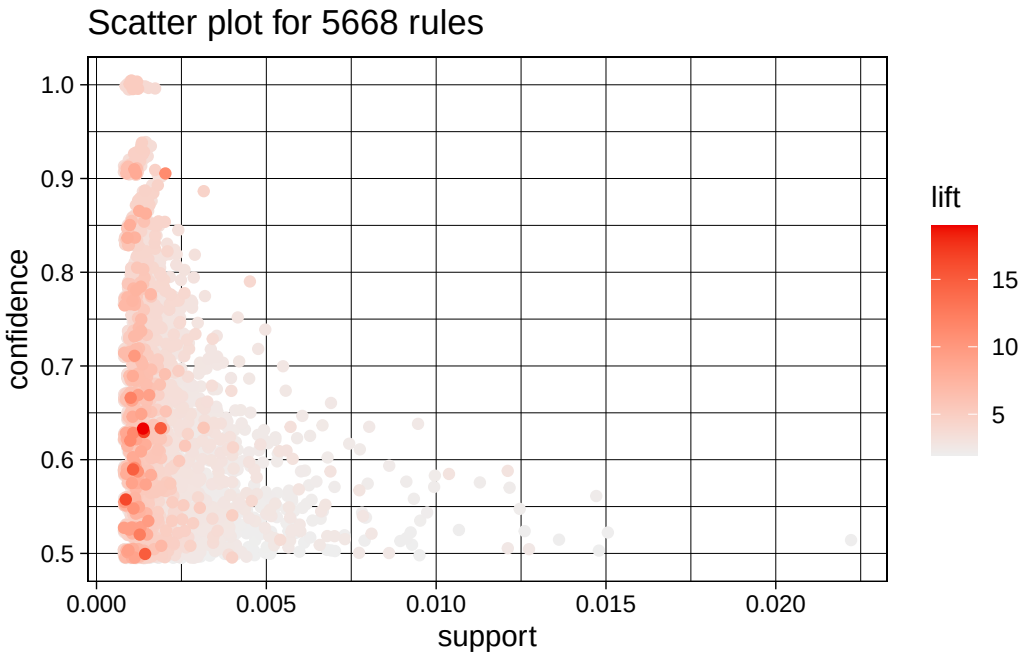
We extract the rules:

```
library(arules)
library(arulesViz)
data(Groceries)
rules <- apriori(Groceries, parameter = list(support = 0.001, confidence = 0.5))
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.5    0.1    1 none FALSE            TRUE       5   0.001      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 9
```

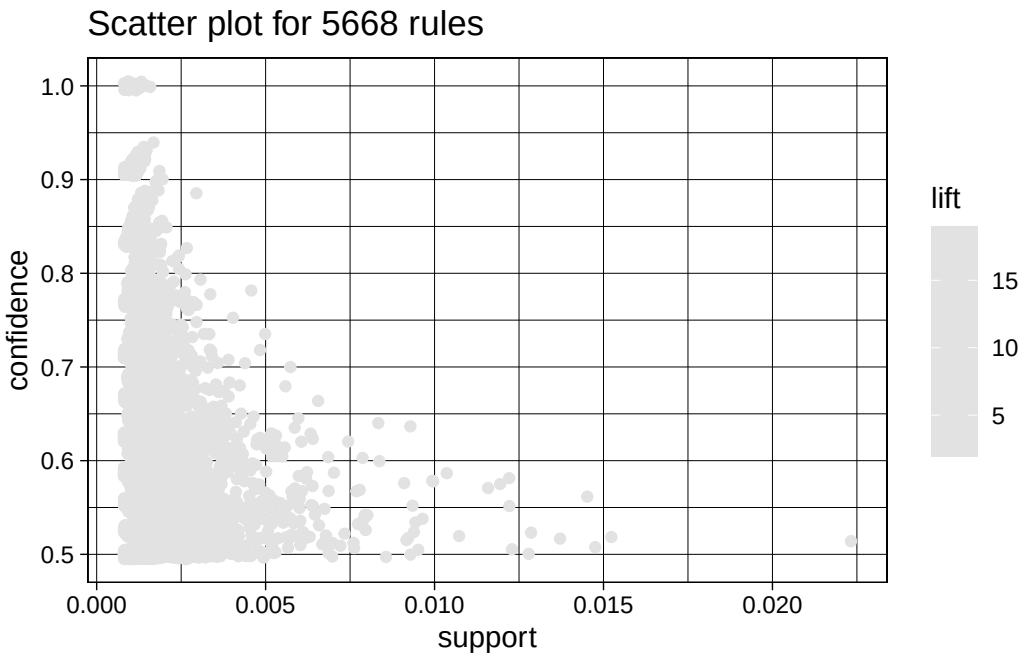
```
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [157 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 done [0.01s].
## writing ... [5668 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
rules
## set of 5668 rules
inspect(head(rules))
##     lhs                    rhs          support     confidence coverage
## [1] {honey}             => {whole milk} 0.001118454 0.7333333  0.001525165
## [2] {tidbits}           => {rolls/buns} 0.001220132 0.5217391  0.002338587
## [3] {cocoa drinks}      => {whole milk} 0.001321810 0.5909091  0.002236909
## [4] {pudding powder}    => {whole milk} 0.001321810 0.5652174  0.002338587
## [5] {cooking chocolate} => {whole milk} 0.001321810 0.5200000  0.002541942
## [6] {cereals}           => {whole milk} 0.003660397 0.6428571  0.005693950
##     lift     count
## [1] 2.870009 11
## [2] 2.836542 12
## [3] 2.312611 13
## [4] 2.212062 13
## [5] 2.035097 13
## [6] 2.515917 36
```

The basic rule visualisation command is `plot()`. We show different options for using `plot()` (colours).

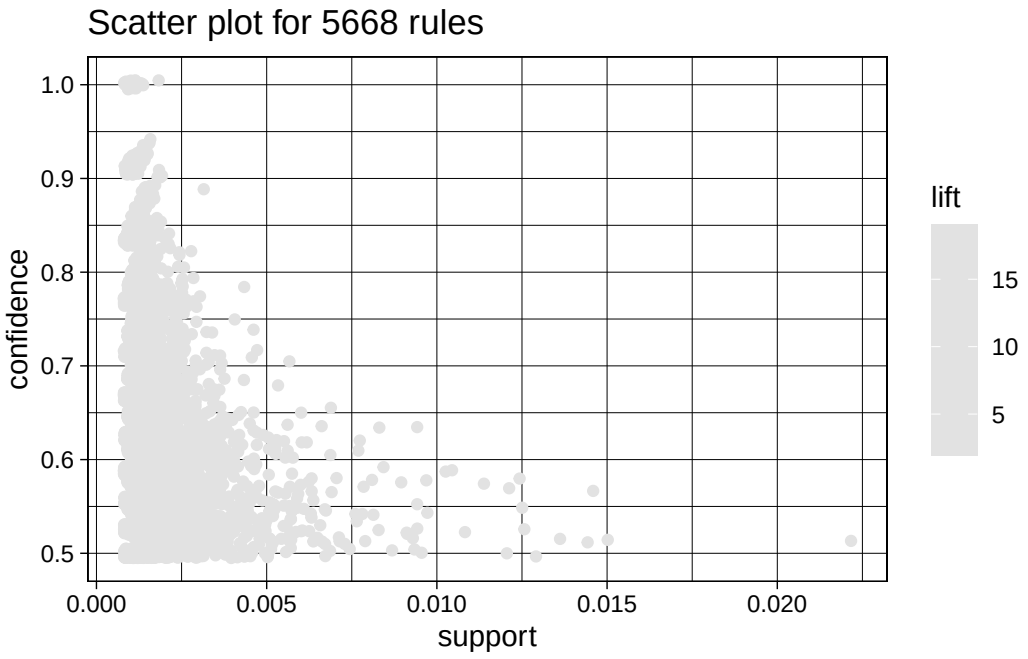
```
plot(rules)
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```



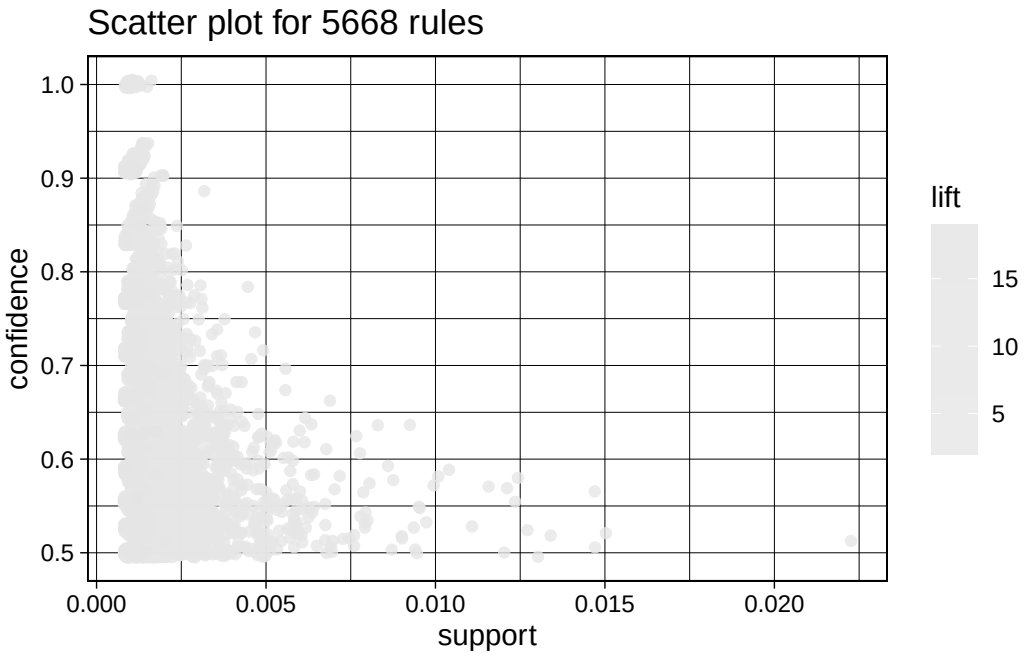
```
library(colorspace)
plot(rules, control = list(col = sequential_hcl(100)))
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```




```
plot(rules, col = sequential_hcl(100))
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```




```
plot(rules, col = grey.colors(50, alpha = .8))
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```



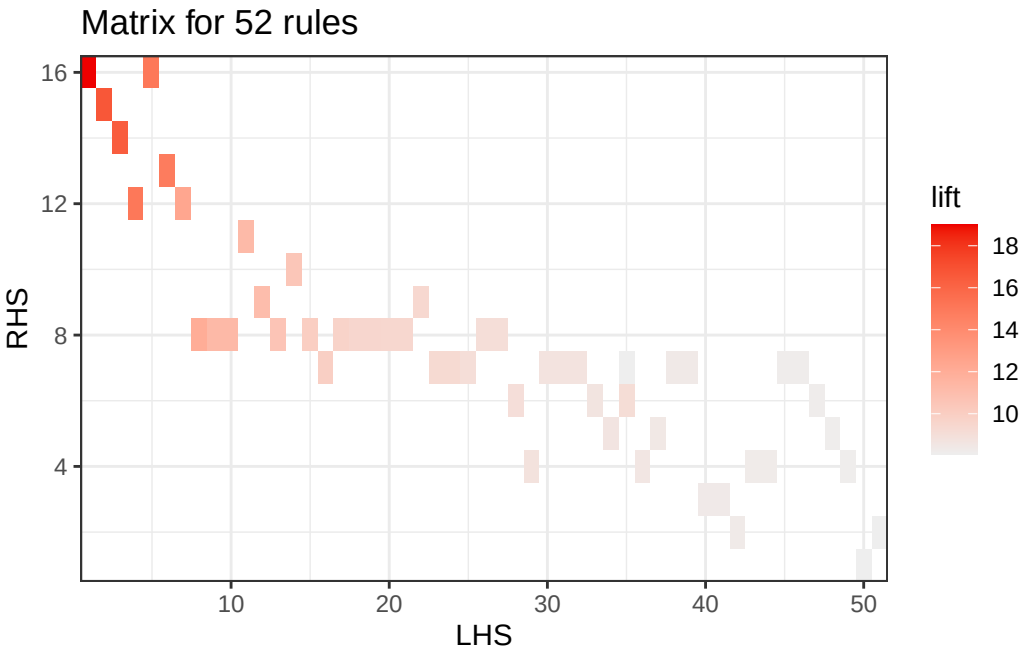
Interactive display option:

```
# Ejecutar en vuestro ordenador
sel <- plot(rules, engine = "interactive")
plot(rules, engine = "htmlwidget")
```

Matrix representation of rules:

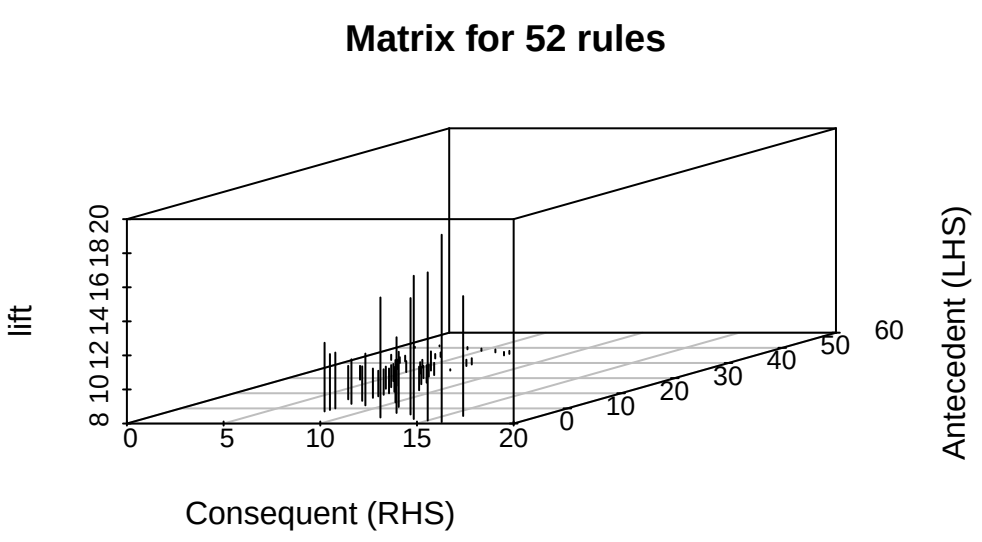
```
subrules <- subset(rules, lift > 8)
subrules
## set of 52 rules
plot(subrules, method = "matrix")
## Itemsets in Antecedent (LHS)
##  [1] "{Instant food products,soda}"
##  [2] "{soda,popcorn}"
##  [3] "{flour,baking powder}"
##  [4] "{ham,processed cheese}"
##  [5] "{whole milk,Instant food products}"
##  [6] "{other vegetables,curd,yogurt,whipped/sour cream}"
##  [7] "{processed cheese,domestic eggs}"
##  [8] "{tropical fruit,other vegetables,yogurt,white bread}"
##  [9] "{hamburger meat,yogurt,whipped/sour cream}"
## [10] "{tropical fruit,other vegetables,whole milk,yogurt,domestic eggs}"
## [11] "{liquor,red/blush wine}"
## [12] "{other vegetables,yogurt,whipped/sour cream,cream cheese }"
## [13] "{yogurt,whipped/sour cream,hard cheese}"
## [14] "{tropical fruit,root vegetables,other vegetables,whole milk,rolls/buns}"
## [15] "{tropical fruit,whole milk,yogurt,sliced cheese}"
## [16] "{other vegetables,butter,sugar}"
## [17] "{whole milk,whipped/sour cream,hard cheese}"
## [18] "{other vegetables,hard cheese,domestic eggs}"
## [19] "{tropical fruit,other vegetables,whipped/sour cream,fruit/vegetable juice}"
```

```
## [20] "{tropical fruit,onions,yogurt}"
## [21] "{tropical fruit,other vegetables,yogurt,domestic eggs}"
## [22] "{butter,yogurt,pastry}"
## [23] "{whole milk,butter,hard cheese}"
## [24] "{tropical fruit,other vegetables,butter,fruit/vegetable juice}"
## [25] "{whole milk,curd,yogurt,cream cheese }"
## [26] "{tropical fruit,other vegetables,hard cheese}"
## [27] "{other vegetables,whole milk,whipped/sour cream,napkins}"
## [28] "{citrus fruit,whole milk,cream cheese }"
## [29] "{tropical fruit,other vegetables,frozen fish}"
## [30] "{butter,yogurt,hard cheese}"
## [31] "{curd,yogurt,sugar}"
## [32] "{other vegetables,whole milk,butter,soda}"
## [33] "{whole milk,cream cheese ,sugar}"
## [34] "{frozen vegetables,specialty chocolate}"
## [35] "{citrus fruit,other vegetables,whole milk,cream cheese }"
## [36] "{tropical fruit,whipped/sour cream,shopping bags}"
## [37] "{citrus fruit,tropical fruit,grapes}"
## [38] "{other vegetables,butter,hard cheese}"
## [39] "{whole milk,butter,sliced cheese}"
## [40] "{citrus fruit,other vegetables,soda,fruit/vegetable juice}"
## [41] "{tropical fruit,other vegetables,whole milk,yogurt,oil}"
## [42] "{tropical fruit,grapes,fruit/vegetable juice}"
## [43] "{frankfurter,tropical fruit,domestic eggs}"
## [44] "{tropical fruit,whole milk,yogurt,frozen meals}"
## [45] "{other vegetables,curd,yogurt,cream cheese }"
## [46] "{root vegetables,whole milk,flour}"
## [47] "{citrus fruit,whole milk,sugar}"
## [48] "{tropical fruit,other vegetables,misc. beverages}"
## [49] "{ham,tropical fruit,other vegetables}"
## [50] "{citrus fruit,grapes,fruit/vegetable juice}"
## [51] "{whole milk,whipped/sour cream,rolls/buns,pastry}"
## Itemsets in Consequent (RHS)
##  [1] "{tropical fruit}"        "{citrus fruit}"
##  [3] "{root vegetables}"       "{pip fruit}"
##  [5] "{fruit/vegetable juice}" "{domestic eggs}"
##  [7] "{whipped/sour cream}"    "{butter}"
##  [9] "{curd}"                  "{beef}"
## [11] "{bottled beer}"          "{white bread}"
## [13] "{cream cheese }"         "{sugar}"
## [15] "{salty snack}"           "{hamburger meat}"
```




```
plot(subrules, method = "matrix", engine = "3d")
## Itemsets in Antecedent (LHS)
##  [1] "{Instant food products,soda}"
##  [2] "{soda,popcorn}"
##  [3] "{flour,baking powder}"
##  [4] "{ham,processed cheese}"
##  [5] "{whole milk,Instant food products}"
##  [6] "{other vegetables,curd,yogurt,whipped/sour cream}"
##  [7] "{processed cheese,domestic eggs}"
##  [8] "{tropical fruit,other vegetables,yogurt,white bread}"
##  [9] "{hamburger meat,yogurt,whipped/sour cream}"
## [10] "{tropical fruit,other vegetables,whole milk,yogurt,domestic eggs}"
## [11] "{liquor,red/blush wine}"
## [12] "{other vegetables,yogurt,whipped/sour cream,cream cheese }"
## [13] "{yogurt,whipped/sour cream,hard cheese}"
## [14] "{tropical fruit,root vegetables,other vegetables,whole milk,rolls/buns}"
## [15] "{tropical fruit,whole milk,yogurt,sliced cheese}"
## [16] "{other vegetables,butter,sugar}"
## [17] "{whole milk,whipped/sour cream,hard cheese}"
## [18] "{other vegetables,hard cheese,domestic eggs}"
## [19] "{tropical fruit,other vegetables,whipped/sour cream,fruit/vegetable juice}"
## [20] "{tropical fruit,onions,yogurt}"
## [21] "{tropical fruit,other vegetables,yogurt,domestic eggs}"
## [22] "{butter,yogurt,pastry}"
## [23] "{whole milk,butter,hard cheese}"
## [24] "{tropical fruit,other vegetables,butter,fruit/vegetable juice}"
## [25] "{whole milk,curd,yogurt,cream cheese }"
## [26] "{tropical fruit,other vegetables,hard cheese}"
## [27] "{other vegetables,whole milk,whipped/sour cream,napkins}"
## [28] "{citrus fruit,whole milk,cream cheese }"
## [29] "{tropical fruit,other vegetables,frozen fish}"
```

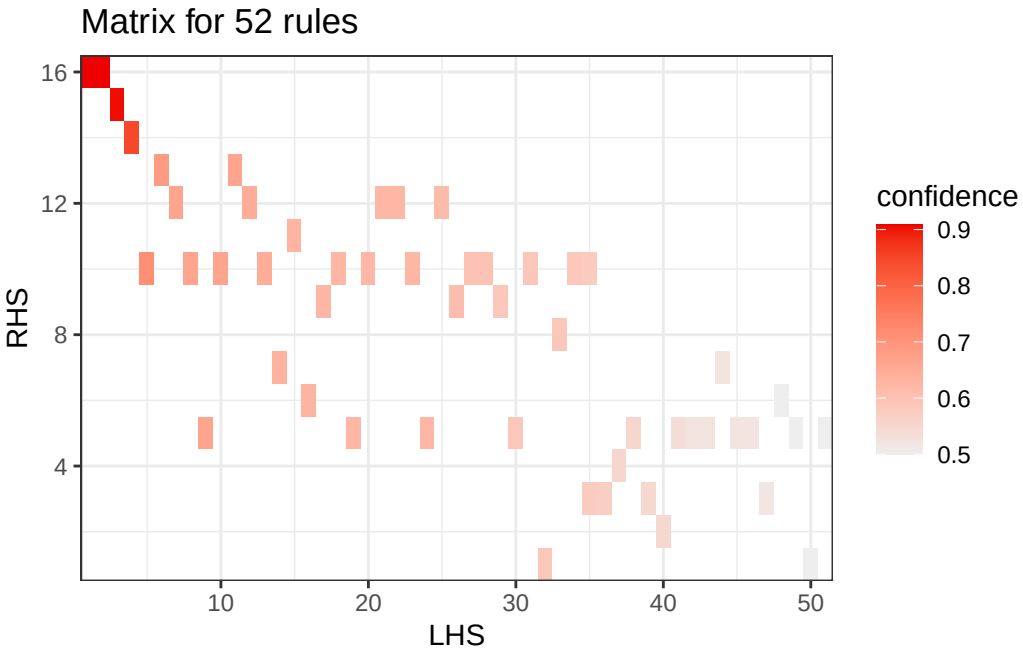
```
## [30] "{butter,yogurt,hard cheese}"
## [31] "{curd,yogurt,sugar}"
## [32] "{other vegetables,whole milk,butter,soda}"
## [33] "{whole milk,cream cheese ,sugar}"
## [34] "{frozen vegetables,specialty chocolate}"
## [35] "{citrus fruit,other vegetables,whole milk,cream cheese }"
## [36] "{tropical fruit,whipped/sour cream,shopping bags}"
## [37] "{citrus fruit,tropical fruit,grapes}"
## [38] "{other vegetables,butter,hard cheese}"
## [39] "{whole milk,butter,sliced cheese}"
## [40] "{citrus fruit,other vegetables,soda,fruit/vegetable juice}"
## [41] "{tropical fruit,other vegetables,whole milk,yogurt,oil}"
## [42] "{tropical fruit,grapes,fruit/vegetable juice}"
## [43] "{frankfurter,tropical fruit,domestic eggs}"
## [44] "{tropical fruit,whole milk,yogurt,frozen meals}"
## [45] "{other vegetables,curd,yogurt,cream cheese }"
## [46] "{root vegetables,whole milk,flour}"
## [47] "{citrus fruit,whole milk,sugar}"
## [48] "{tropical fruit,other vegetables,misc. beverages}"
## [49] "{ham,tropical fruit,other vegetables}"
## [50] "{citrus fruit,grapes,fruit/vegetable juice}"
## [51] "{whole milk,whipped/sour cream,rolls/buns,pastry}"
## Itemsets in Consequent (RHS)
##  [1] "{tropical fruit}"        "{citrus fruit}"
##  [3] "{root vegetables}"       "{pip fruit}"
##  [5] "{fruit/vegetable juice}" "{domestic eggs}"
##  [7] "{whipped/sour cream}"    "{butter}"
##  [9] "{curd}"                  "{beef}"
## [11] "{bottled beer}"          "{white bread}"
## [13] "{cream cheese }"         "{sugar}"
## [15] "{salty snack}"           "{hamburger meat}"
```




```
plot(subrules, method = "matrix",
     measure = "lift",
```

```
     shading = "confidence")
## Itemsets in Antecedent (LHS)
##  [1] "{citrus fruit,other vegetables,soda,fruit/vegetable juice}"
##  [2] "{tropical fruit,other vegetables,whole milk,yogurt,oil}"
##  [3] "{liquor,red/blush wine}"
##  [4] "{citrus fruit,grapes,fruit/vegetable juice}"
##  [5] "{other vegetables,butter,sugar}"
##  [6] "{tropical fruit,grapes,fruit/vegetable juice}"
##  [7] "{tropical fruit,other vegetables,frozen fish}"
##  [8] "{whole milk,butter,hard cheese}"
##  [9] "{tropical fruit,other vegetables,yogurt,white bread}"
## [10] "{tropical fruit,other vegetables,butter,fruit/vegetable juice}"
## [11] "{whole milk,whipped/sour cream,rolls/buns,pastry}"
## [12] "{tropical fruit,whipped/sour cream,shopping bags}"
## [13] "{whole milk,curd,yogurt,cream cheese }"
## [14] "{ham,processed cheese}"
## [15] "{soda,popcorn}"
## [16] "{Instant food products,soda}"
## [17] "{frozen vegetables,specialty chocolate}"
## [18] "{butter,yogurt,hard cheese}"
## [19] "{hamburger meat,yogurt,whipped/sour cream}"
## [20] "{curd,yogurt,sugar}"
## [21] "{frankfurter,tropical fruit,domestic eggs}"
## [22] "{tropical fruit,whole milk,yogurt,frozen meals}"
## [23] "{other vegetables,whole milk,butter,soda}"
## [24] "{tropical fruit,other vegetables,whole milk,yogurt,domestic eggs}"
## [25] "{ham,tropical fruit,other vegetables}"
## [26] "{citrus fruit,tropical fruit,grapes}"
## [27] "{other vegetables,butter,hard cheese}"
## [28] "{whole milk,butter,sliced cheese}"
## [29] "{tropical fruit,other vegetables,misc. beverages}"
## [30] "{yogurt,whipped/sour cream,hard cheese}"
## [31] "{other vegetables,curd,yogurt,cream cheese }"
## [32] "{other vegetables,yogurt,whipped/sour cream,cream cheese }"
## [33] "{other vegetables,curd,yogurt,whipped/sour cream}"
## [34] "{root vegetables,whole milk,flour}"
## [35] "{citrus fruit,other vegetables,whole milk,cream cheese }"
## [36] "{citrus fruit,whole milk,cream cheese }"
## [37] "{flour,baking powder}"
## [38] "{tropical fruit,whole milk,yogurt,sliced cheese}"
## [39] "{whole milk,cream cheese ,sugar}"
## [40] "{tropical fruit,root vegetables,other vegetables,whole milk,rolls/buns}"
## [41] "{whole milk,whipped/sour cream,hard cheese}"
## [42] "{other vegetables,hard cheese,domestic eggs}"
## [43] "{tropical fruit,other vegetables,whipped/sour cream,fruit/vegetable juice}"
## [44] "{processed cheese,domestic eggs}"
## [45] "{tropical fruit,onions,yogurt}"
## [46] "{tropical fruit,other vegetables,yogurt,domestic eggs}"
## [47] "{citrus fruit,whole milk,sugar}"
```

```
## [48] "{whole milk,Instant food products}"
## [49] "{tropical fruit,other vegetables,hard cheese}"
## [50] "{butter,yogurt,pastry}"
## [51] "{other vegetables,whole milk,whipped/sour cream,napkins}"
## Itemsets in Consequent (RHS)
##  [1] "{curd}"                    "{beef}"
##  [3] "{domestic eggs}"           "{sugar}"
##  [5] "{butter}"                  "{hamburger meat}"
##  [7] "{white bread}"             "{cream cheese }"
##  [9] "{fruit/vegetable juice}" "{whipped/sour cream}"
## [11] "{salty snack}"             "{pip fruit}"
## [13] "{citrus fruit}"            "{tropical fruit}"
## [15] "{bottled beer}"            "{root vegetables}"
```



And interactive:

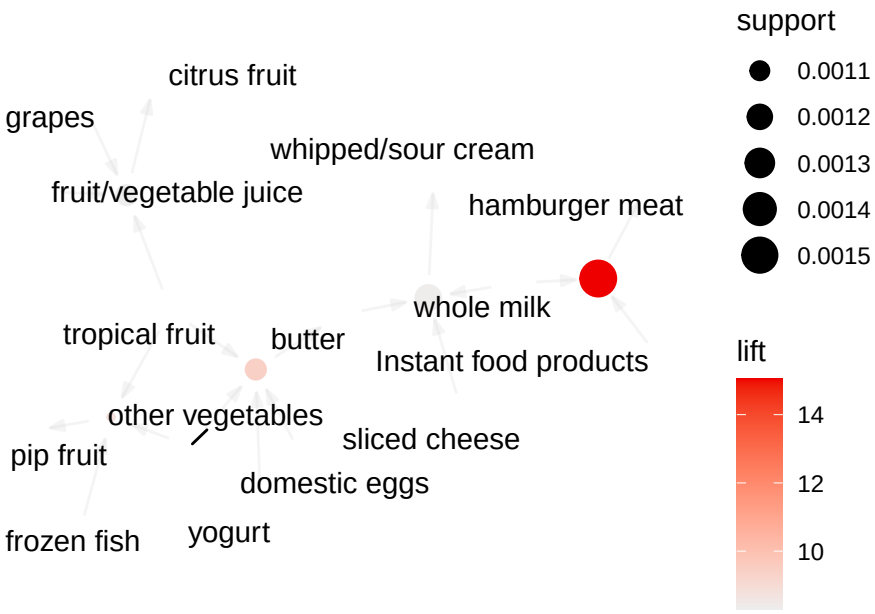
```
# Ejecutar en vuestro ordenador
plot(subrules, method = "matrix", engine = "interactive")
plot(subrules, method = "matrix", engine = "htmlwidget")
```

Matrix representation showing the items:

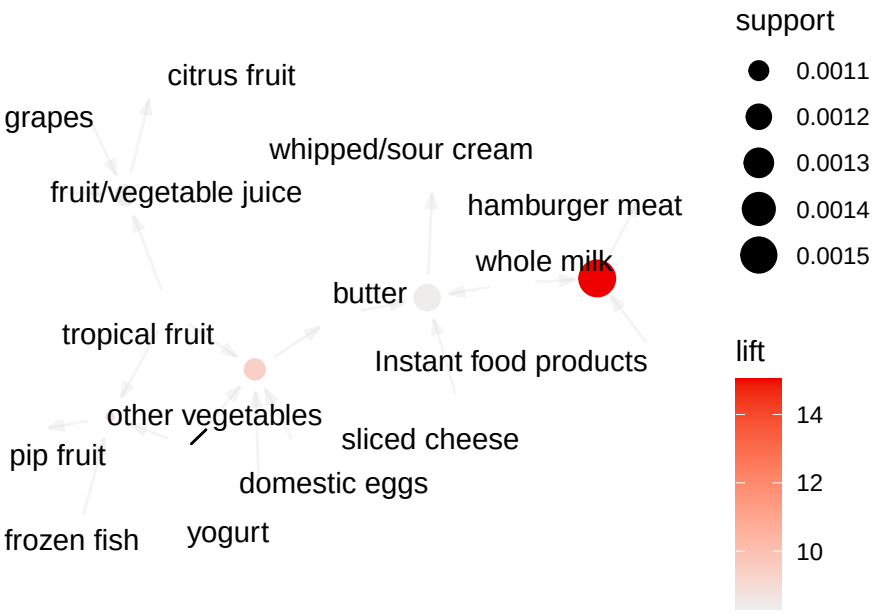
```
# Ejecutar en vuestro ordenador
plot(subrules, method = "grouped matrix")
plot(subrules,
  method = "grouped matrix",
  col = grey.colors(10),
  gp_labels = gpar(col = "blue", cex = 1, fontface = "italic")
)
# sel <- plot(rules, method="grouped", engine = "interactive")
```

Network representation of rules (only for small sets of rules):

```
subrules2 <- sample(subrules, 5)
plot(subrules2, method = "graph")
```

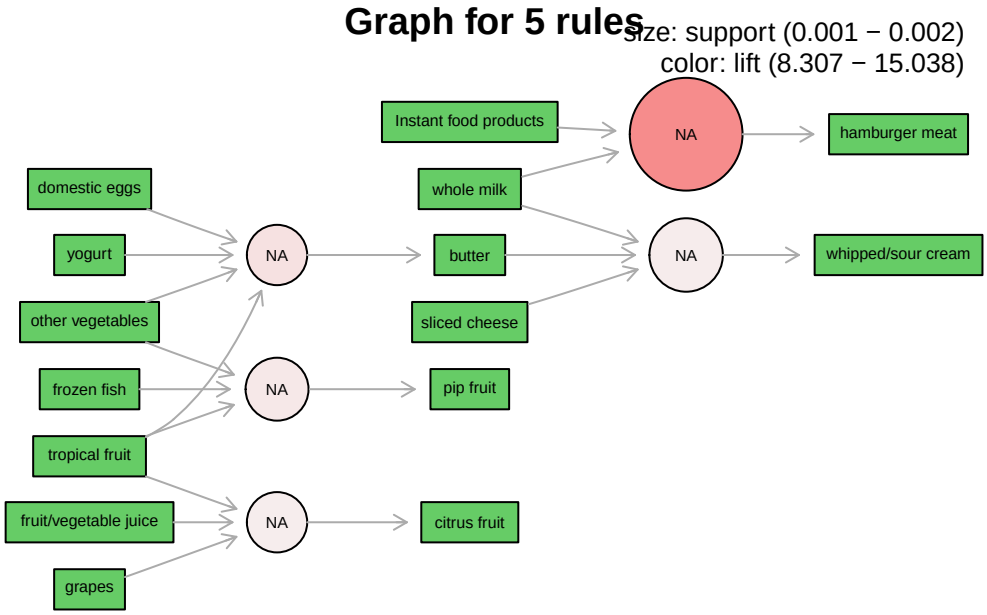



```
plot(subrules2,
  method = "graph",
  nodeCol = grey.colors(10), edgeCol = grey(.7), alpha = 1
)
## Warning: Unknown control parameters: nodeCol, edgeCol, alpha
## Available control parameters (with default values):
## layout    =  stress
## circular  =  FALSE
## ggraphdots    =  NULL
## edges     =  <environment>
## nodes     =  <environment>
## nodetext  =  <environment>
## colors    =  c("#EE0000FF", "#EEEEEEFF")
## engine    =  ggplot2
## max   =  100
## verbose   =  FALSE
```



The `Graphviz` package allows for better visualisation:

```
plot(subrules2, method = "graph", engine = "graphviz")
```



It allows dynamic visualisation:

```
# Ejecutar en local
plot(subrules2, method = "graph", engine = "htmlwidget")
plot(subrules2,
  method = "graph", engine = "htmlwidget",
  igraphLayout = "layout_in_circle"
)
```

7.8 A recommender system

Analysis of relationships between songs

The `lastfm.csv` dataset includes transactions collected from an online radio station that stores user ID, artist, user gender and country.

Objective: To build a system for recommending music groups to users from the above dataset.

```
library(arules)
lastfm <- read.csv("lastfm.csv")
lastfm[1:20, ]
##    user                  artist sex       country
## 1     1   red hot chili peppers   f       Germany
## 2     1 the black dahlia murder   f       Germany
## 3     1               goldfrapp   f       Germany
## 4     1        dropkick murphys   f       Germany
## 5     1                le tigre   f       Germany
## 6     1              schandmaul   f       Germany
## 7     1                  edguy   f       Germany
## 8     1           jack johnson   f       Germany
## 9     1              eluveitie   f       Germany
## 10    1             the killers   f       Germany
## 11    1            judas priest   f       Germany
## 12    1              rob zombie   f       Germany
## 13    1              john mayer   f       Germany
## 14    1                 the who   f       Germany
## 15    1              guano apes   f       Germany
## 16    1      the rolling stones   f       Germany
## 17    3        devendra banhart   m United States
## 18    3        boards of canada   m United States
## 19    3               cocorosie   m United States
## 20    3              aphex twin   m United States
length(lastfm$user) ## 289,955 filas
## [1] 289955
class(lastfm$user)
## [1] "integer"
# We need to convert this attribute to a factor in order to be able to analyse it with pa

lastfm$user <- factor(lastfm$user)
# Run on your computer
# levels(lastfm$user)  ## 15,000 users
# levels(lastfm$artist)  ## 1,004 artists
```

We call `apriori()`:

```
reglas1 <- apriori(lastfm, parameter = list(support = .01, confidence = .5))
## Warning: Column(s) 2, 3, 4 not logical or factor. Applying default
## discretization (see '? discretizeDF').
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.5    0.1    1 none FALSE            TRUE       5    0.01      1
```

```
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 2899
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[16165 item(s), 289955 transaction(s)] done [0.14s].
## sorting and recoding items ... [21 item(s)] done [0.00s].
## creating transaction tree ... done [0.04s].
## checking subsets of size 1 2 done [0.00s].
## writing ... [19 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
inspect(reglas1)
##      lhs                                rhs     support    confidence coverage
## [1]  {}                              => {sex=m} 0.73053750 0.7305375  1.00000000
## [2]  {country=Czech Republic}        => {sex=m} 0.01039472 0.8033049  0.01293994
## [3]  {country=Mexico}                => {sex=m} 0.01023607 0.7804365  0.01311583
## [4]  {country=Norway}                => {sex=m} 0.01239503 0.7744021  0.01600593
## [5]  {country=Turkey}                => {sex=m} 0.01088445 0.6627467  0.01642324
## [6]  {country=Italy}                 => {sex=m} 0.01501612 0.7615882  0.01971685
## [7]  {country=France}                => {sex=m} 0.01707161 0.8302583  0.02056181
## [8]  {country=Australia}             => {sex=m} 0.01497474 0.6776963  0.02209653
## [9]  {country=Canada}                => {sex=m} 0.01568174 0.6563222  0.02389336
## [10] {country=Spain}                 => {sex=m} 0.02482109 0.7720446  0.03214982
## [11] {country=Netherlands}           => {sex=m} 0.02690417 0.8064716  0.03336035
## [12] {country=Finland}               => {sex=m} 0.02661103 0.7596731  0.03502957
## [13] {country=Russian Federation}    => {sex=m} 0.03062544 0.7605344  0.04026832
## [14] {country=Brazil}                => {sex=m} 0.03001845 0.7300788  0.04111673
## [15] {country=Sweden}                => {sex=m} 0.03193254 0.7479603  0.04269283
## [16] {country=Poland}                => {sex=m} 0.03854391 0.6531471  0.05901261
## [17] {country=Germany}               => {sex=m} 0.06069907 0.7257433  0.08363712
## [18] {country=United Kingdom}        => {sex=m} 0.07730510 0.8110211  0.09531824
## [19] {country=United States}         => {sex=m} 0.13852839 0.6744182  0.20540429
##      lift      count
## [1]  1.0000000 211823
## [2]  1.0996080   3014
## [3]  1.0683045   2968
## [4]  1.0600442   3594
## [5]  0.9072043   3156
## [6]  1.0425040   4354
## [7]  1.1365033   4950
## [8]  0.9276680   4342
## [9]  0.8984100   4547
## [10] 1.0568172   7197
## [11] 1.1039428   7801
```

```
## [12] 1.0398825   7716
## [13] 1.0410615   8880
## [14] 0.9993722   8704
## [15] 1.0238492   9259
## [16] 0.8940638  11176
## [17] 0.9934374  17600
## [18] 1.1101703  22415
## [19] 0.9231808  40167
```

Comment: In previous versions of `arules` the previous command gave an error. We had to convert discrete variables to factor. It is a *live* package that is evolving day by day.

**What is the recommendation we can get with these rules?

It is not the kind of rules we want to get for our recommender system.

Rules for making recommendations

The data must be manipulated in order to find what we are interested in. We will use the commands: `split()`, `lapply()`.

First I keep a list of what each user listens to:

```
lista.musica.por.usuario <- split(x = lastfm[, "artist"], f = lastfm$user)
lista.musica.por.usuario[1:2]
## $`1`
##  [1] "red hot chili peppers"   "the black dahlia murder"
##  [3] "goldfrapp"               "dropkick murphys"
##  [5] "le tigre"                "schandmaul"
##  [7] "edguy"                   "jack johnson"
##  [9] "eluveitie"               "the killers"
## [11] "judas priest"            "rob zombie"
## [13] "john mayer"              "the who"
## [15] "guano apes"              "the rolling stones"
##
## $`3`
##  [1] "devendra banhart"    "boards of canada"    "cocorosie"
##  [4] "aphex twin"          "animal collective"   "atmosphere"
##  [7] "joanna newsom"       "air"                 "portishead"
## [10] "massive attack"      "broken social scene" "arcade fire"
## [13] "plaid"               "prefuse 73"          "m83"
## [16] "the flashbulb"       "pavement"            "goldfrapp"
## [19] "amon tobin"          "sage francis"        "four tet"
## [22] "max richter"         "autechre"            "radiohead"
## [25] "neutral milk hotel"  "beastie boys"        "aesop rock"
## [28] "mf doom"             "the books"
```

Next:

- A group/singer could be in a user twice: remove repeats
- Convert to transaction format
- Look at the music listened to by the first users

```
## Remove duplicates
lista.musica.por.usuario <- lapply(lista.musica.por.usuario, unique)

# We convert the music list into transactions.
lista.musica.por.usuario1 <- as(lista.musica.por.usuario, "transactions")

lista.musica.por.usuario[1:5]
## $`1`
##  [1] "red hot chili peppers"   "the black dahlia murder"
##  [3] "goldfrapp"               "dropkick murphys"
##  [5] "le tigre"                "schandmaul"
##  [7] "edguy"                   "jack johnson"
##  [9] "eluveitie"               "the killers"
## [11] "judas priest"            "rob zombie"
## [13] "john mayer"              "the who"
## [15] "guano apes"              "the rolling stones"
##
## $`3`
##  [1] "devendra banhart"    "boards of canada"    "cocorosie"
##  [4] "aphex twin"          "animal collective"   "atmosphere"
##  [7] "joanna newsom"       "air"                 "portishead"
## [10] "massive attack"      "broken social scene" "arcade fire"
## [13] "plaid"               "prefuse 73"          "m83"
## [16] "the flashbulb"       "pavement"            "goldfrapp"
## [19] "amon tobin"          "sage francis"        "four tet"
## [22] "max richter"         "autechre"            "radiohead"
## [25] "neutral milk hotel"  "beastie boys"        "aesop rock"
## [28] "mf doom"             "the books"
##
## $`4`
##  [1] "tv on the radio"         "tool"
##  [3] "kyuss"                   "dj shadow"
##  [5] "air"                     "a tribe called quest"
##  [7] "the cinematic orchestra" "beck"
##  [9] "bon iver"                "röyksopp"
## [11] "bonobo"                  "the decemberists"
## [13] "snow patrol"             "battles"
## [15] "the prodigy"             "pink floyd"
## [17] "rjd2"                    "the flaming lips"
## [19] "michael jackson"         "mgmt"
## [21] "the rolling stones"      "late of the pier"
## [23] "flight of the conchords" "simian mobile disco"
## [25] "muse"                    "fleetwood mac"
## [27] "led zeppelin"
##
## $`5`
##  [1] "dream theater"            "ac/dc"
##  [3] "metallica"                "iron maiden"
##  [5] "bob marley & the wailers" "megadeth"
```

```
##  [7] "children of bodom"        "trivium"
##  [9] "nightwish"                "sublime"
## [11] "volbeat"
##
## $`6`
##  [1] "lily allen"           "kanye west"           "sigur rós"
##  [4] "pink floyd"           "stevie wonder"        "metallica"
##  [7] "thievery corporation" "iron maiden"          "the streets"
## [10] "muse"                 "faith no more"        "manu chao"
## [13] "tenacious d"          "depeche mode"         "justin timberlake"
## [16] "green day"            "snow patrol"          "dream theater"
## [19] "u2"                   "jay-z"                "type o negative"
## [22] "pearl jam"            "queen"
```

We visualise what we have achieved so far:

```
str(lista.musica.por.usuario1)
## Formal class 'transactions' [package "arules"] with 3 slots
##   ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
##   .. .. ..@ i       : int [1:289953] 280 288 299 373 383 429 457 468 512 714 ...
##   .. .. ..@ p       : int [1:15001] 0 16 45 72 83 106 128 147 177 184 ...
##   .. .. ..@ Dim     : int [1:2] 1004 15000
##   .. .. ..@ Dimnames:List of 2
##   .. .. .. ..$ : NULL
##   .. .. .. ..$ : NULL
##   .. .. ..@ factors : list()
##   ..@ itemInfo   :'data.frame':  1004 obs. of  1 variable:
##   .. ..$ labels: chr [1:1004] "...and you will know us by the trail of dead" "[unknown
##   ..@ itemsetInfo:'data.frame':  15000 obs. of  1 variable:
##   .. ..$ transactionID: chr [1:15000] "1" "3" "4" "5" ...
write(head(lista.musica.por.usuario1))
## "dropkick murphys" "edguy" "eluveitie" "goldfrapp" "guano apes" "jack johnson" "john m
## "aesop rock" "air" "amon tobin" "animal collective" "aphex twin" "arcade fire" "atmosp
## "a tribe called quest" "air" "battles" "beck" "bon iver" "bonobo" "dj shadow" "fleetwo
## "ac/dc" "bob marley & the wailers" "children of bodom" "dream theater" "iron maiden" "
## "depeche mode" "dream theater" "faith no more" "green day" "iron maiden" "jay-z" "just
## "ac/dc" "aerosmith" "alice in chains" "audioslave" "buckethead" "camel" "disturbed" "d
write(head(lista.musica.por.usuario1), format = "single")
## "1" "dropkick murphys"
## "1" "edguy"
## "1" "eluveitie"
## "1" "goldfrapp"
## "1" "guano apes"
## "1" "jack johnson"
## "1" "john mayer"
## "1" "judas priest"
## "1" "le tigre"
## "1" "red hot chili peppers"
## "1" "rob zombie"
```

```
## "1" "schandmaul"
## "1" "the black dahlia murder"
## "1" "the killers"
## "1" "the rolling stones"
## "1" "the who"
## "3" "aesop rock"
## "3" "air"
## "3" "amon tobin"
## "3" "animal collective"
## "3" "aphex twin"
## "3" "arcade fire"
## "3" "atmosphere"
## "3" "autechre"
## "3" "beastie boys"
## "3" "boards of canada"
## "3" "broken social scene"
## "3" "cocorosie"
## "3" "devendra banhart"
## "3" "four tet"
## "3" "goldfrapp"
## "3" "joanna newsom"
## "3" "m83"
## "3" "massive attack"
## "3" "max richter"
## "3" "mf doom"
## "3" "neutral milk hotel"
## "3" "pavement"
## "3" "plaid"
## "3" "portishead"
## "3" "prefuse 73"
## "3" "radiohead"
## "3" "sage francis"
## "3" "the books"
## "3" "the flashbulb"
## "4" "a tribe called quest"
## "4" "air"
## "4" "battles"
## "4" "beck"
## "4" "bon iver"
## "4" "bonobo"
## "4" "dj shadow"
## "4" "fleetwood mac"
## "4" "flight of the conchords"
## "4" "kyuss"
## "4" "late of the pier"
## "4" "led zeppelin"
## "4" "mgmt"
## "4" "michael jackson"
## "4" "muse"
```

```
## "4" "pink floyd"
## "4" "rjd2"
## "4" "röyksopp"
## "4" "simian mobile disco"
## "4" "snow patrol"
## "4" "the cinematic orchestra"
## "4" "the decemberists"
## "4" "the flaming lips"
## "4" "the prodigy"
## "4" "the rolling stones"
## "4" "tool"
## "4" "tv on the radio"
## "5" "ac/dc"
## "5" "bob marley & the wailers"
## "5" "children of bodom"
## "5" "dream theater"
## "5" "iron maiden"
## "5" "megadeth"
## "5" "metallica"
## "5" "nightwish"
## "5" "sublime"
## "5" "trivium"
## "5" "volbeat"
## "6" "depeche mode"
## "6" "dream theater"
## "6" "faith no more"
## "6" "green day"
## "6" "iron maiden"
## "6" "jay-z"
## "6" "justin timberlake"
## "6" "kanye west"
## "6" "lily allen"
## "6" "manu chao"
## "6" "metallica"
## "6" "muse"
## "6" "pearl jam"
## "6" "pink floyd"
## "6" "queen"
## "6" "sigur rós"
## "6" "snow patrol"
## "6" "stevie wonder"
## "6" "tenacious d"
## "6" "the streets"
## "6" "thievery corporation"
## "6" "type o negative"
## "6" "u2"
## "7" "ac/dc"
## "7" "aerosmith"
## "7" "alice in chains"
```

```
## "7" "audioslave"
## "7" "buckethead"
## "7" "camel"
## "7" "disturbed"
## "7" "dream theater"
## "7" "jethro tull"
## "7" "king crimson"
## "7" "led zeppelin"
## "7" "oasis"
## "7" "pearl jam"
## "7" "pink floyd"
## "7" "porcupine tree"
## "7" "rammstein"
## "7" "rush"
## "7" "soundgarden"
## "7" "stone temple pilots"
## "7" "the verve"
## "7" "tool"
## "7" "type o negative"
```

It is a list of transactions - data type defined in arules. We calculate the relative frequency of the songs listened to:

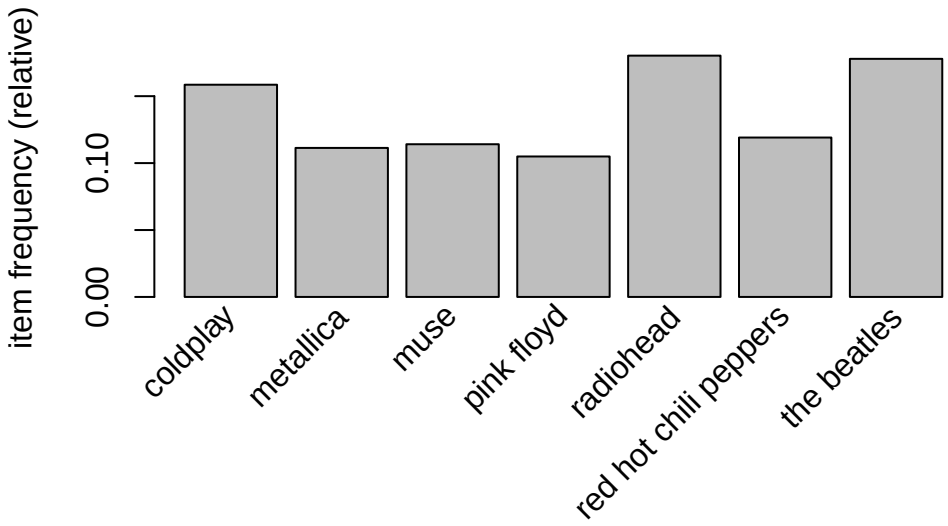
```
itfreq1 <- itemFrequency(lista.musica.por.usuario1)
head(itfreq1)
## ...and you will know us by the trail of dead
##                                   0.009800000
##                                     [unknown]
##                                   0.036866667
##                                          2pac
##                                   0.022733333
##                                  3 doors down
##                                   0.030933333
##                               30 seconds to mars
##                                   0.032800000
##                                           311
##                                   0.008333333
```

`itfreq1`:

- is a numeric vector
- the names of the list (`names(itfreq)` — the names of each group)
- each position therefore is the frequency of the group of that position

Draw the frequencies using the list of transactions obtained:

```
itemFrequencyPlot(lista.musica.por.usuario1, support = .1, cex.names = 1)
```



And we obtain the association rules with support 0.1 and confidence 0.5:

```
reglas2 <- apriori(lista.musica.por.usuario1,
  parameter =
    list(support = .01, confidence = .5)
)
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.5    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 150
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[1004 item(s), 15000 transaction(s)] done [0.03s].
## sorting and recoding items ... [655 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.01s].
## writing ... [50 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
reglas2
## set of 50 rules
inspect(reglas2)
##      lhs                                rhs             support
## [1]  {t.i.}                          => {kanye west}    0.01040000
## [2]  {the pussycat dolls}            => {rihanna}       0.01040000
## [3]  {the fray}                      => {coldplay}      0.01126667
## [4]  {sonata arctica}                => {nightwish}     0.01346667
```

```
## [5]  {judas priest}                          => {iron maiden}  0.01353333
## [6]  {the kinks}                             => {the beatles}  0.01360000
## [7]  {travis}                                => {coldplay}     0.01373333
## [8]  {the flaming lips}                      => {radiohead}    0.01306667
## [9]  {megadeth}                              => {metallica}    0.01626667
## [10] {simon & garfunkel}                     => {the beatles}  0.01540000
## [11] {broken social scene}                   => {radiohead}    0.01506667
## [12] {blur}                                  => {radiohead}    0.01753333
## [13] {keane}                                 => {coldplay}     0.02226667
## [14] {snow patrol}                           => {coldplay}     0.02646667
## [15] {beck}                                  => {radiohead}    0.02926667
## [16] {snow patrol, the killers}              => {coldplay}     0.01040000
## [17] {radiohead, snow patrol}                => {coldplay}     0.01006667
## [18] {death cab for cutie, the shins}        => {radiohead}    0.01006667
## [19] {the beatles, the shins}                => {radiohead}    0.01066667
## [20] {led zeppelin, the doors}               => {pink floyd}   0.01066667
## [21] {pink floyd, the doors}                 => {led zeppelin} 0.01066667
## [22] {pink floyd, the doors}                 => {the beatles}  0.01000000
## [23] {the beatles, the strokes}              => {radiohead}    0.01046667
## [24] {oasis, the killers}                    => {coldplay}     0.01113333
## [25] {oasis, the beatles}                    => {coldplay}     0.01060000
## [26] {oasis, radiohead}                      => {coldplay}     0.01273333
## [27] {beck, the beatles}                     => {radiohead}    0.01300000
## [28] {bob dylan, the rolling stones}         => {the beatles}  0.01146667
## [29] {david bowie, the rolling stones}       => {the beatles}  0.01000000
## [30] {led zeppelin, the rolling stones}      => {the beatles}  0.01066667
## [31] {radiohead, the rolling stones}         => {the beatles}  0.01060000
## [32] {coldplay, the smashing pumpkins}       => {radiohead}    0.01093333
## [33] {the beatles, the smashing pumpkins}    => {radiohead}    0.01146667
## [34] {radiohead, u2}                         => {coldplay}     0.01140000
## [35] {coldplay, sigur rós}                   => {radiohead}    0.01206667
## [36] {sigur rós, the beatles}                => {radiohead}    0.01046667
## [37] {bob dylan, pink floyd}                 => {the beatles}  0.01033333
## [38] {bob dylan, radiohead}                  => {the beatles}  0.01386667
## [39] {bloc party, the killers}               => {coldplay}     0.01106667
## [40] {david bowie, pink floyd}               => {the beatles}  0.01006667
## [41] {david bowie, radiohead}                => {the beatles}  0.01393333
## [42] {placebo, radiohead}                    => {muse}         0.01366667
## [43] {led zeppelin, radiohead}               => {the beatles}  0.01306667
## [44] {death cab for cutie, the killers}      => {coldplay}     0.01086667
## [45] {death cab for cutie, the beatles}      => {radiohead}    0.01246667
## [46] {muse, the killers}                     => {coldplay}     0.01513333
## [47] {red hot chili peppers, the killers}    => {coldplay}     0.01086667
## [48] {the beatles, the killers}              => {coldplay}     0.01253333
## [49] {radiohead, the killers}                => {coldplay}     0.01506667
## [50] {muse, the beatles}                     => {radiohead}    0.01380000
##      confidence coverage   lift      count
## [1]  0.5672727  0.01833333  8.854413 156
## [2]  0.5777778  0.01800000 13.415893 156
```

```
## [3]  0.5168196  0.02180000  3.260006 169
## [4]  0.5101010  0.02640000  8.236292 202
## [5]  0.5075000  0.02666667  8.562992 203
## [6]  0.5298701  0.02566667  2.979030 204
## [7]  0.5628415  0.02440000  3.550304 206
## [8]  0.5297297  0.02466667  2.938589 196
## [9]  0.5281385  0.03080000  4.743759 244
## [10] 0.5238095  0.02940000  2.944956 231
## [11] 0.5472155  0.02753333  3.035589 226
## [12] 0.5228628  0.03353333  2.900496 263
## [13] 0.6374046  0.03493333  4.020634 334
## [14] 0.5251323  0.05040000  3.312441 397
## [15] 0.5092807  0.05746667  2.825152 439
## [16] 0.5954198  0.01746667  3.755802 156
## [17] 0.6344538  0.01586667  4.002021 151
## [18] 0.5033333  0.02000000  2.792160 151
## [19] 0.5673759  0.01880000  3.147425 160
## [20] 0.5970149  0.01786667  5.689469 160
## [21] 0.5387205  0.01980000  6.802027 160
## [22] 0.5050505  0.01980000  2.839489 150
## [23] 0.5607143  0.01866667  3.110471 157
## [24] 0.6626984  0.01680000  4.180183 167
## [25] 0.5196078  0.02040000  3.277594 159
## [26] 0.5876923  0.02166667  3.707058 191
## [27] 0.5909091  0.02200000  3.277972 195
## [28] 0.5910653  0.01940000  3.323081 172
## [29] 0.5703422  0.01753333  3.206572 150
## [30] 0.5776173  0.01846667  3.247474 160
## [31] 0.5638298  0.01880000  3.169958 159
## [32] 0.6283525  0.01740000  3.485683 164
## [33] 0.6209386  0.01846667  3.444556 172
## [34] 0.5213415  0.02186667  3.288529 171
## [35] 0.5801282  0.02080000  3.218167 181
## [36] 0.6434426  0.01626667  3.569393 157
## [37] 0.6150794  0.01680000  3.458092 155
## [38] 0.5730028  0.02420000  3.221530 208
## [39] 0.5236593  0.02113333  3.303150 166
## [40] 0.5741445  0.01753333  3.227949 151
## [41] 0.5225000  0.02666667  2.937594 209
## [42] 0.5137845  0.02660000  4.504247 205
## [43] 0.5283019  0.02473333  2.970213 196
## [44] 0.5884477  0.01846667  3.711823 163
## [45] 0.5013405  0.02486667  2.781105 187
## [46] 0.5089686  0.02973333  3.210483 227
## [47] 0.5093750  0.02133333  3.213047 163
## [48] 0.5340909  0.02346667  3.368950 188
## [49] 0.5243619  0.02873333  3.307582 226
## [50] 0.5073529  0.02720000  2.814458 207
```

Recommendation system

First we keep the most interesting rules. We filter out those with lift greater than 1:

```
inspect(subset(reglas2, subset = lift > 3.5))
##      lhs                                    rhs             support    confidence
## [1]  {t.i.}                              => {kanye west}    0.01040000 0.5672727
## [2]  {the pussycat dolls}                => {rihanna}       0.01040000 0.5777778
## [3]  {sonata arctica}                    => {nightwish}     0.01346667 0.5101010
## [4]  {judas priest}                      => {iron maiden}   0.01353333 0.5075000
## [5]  {travis}                            => {coldplay}      0.01373333 0.5628415
## [6]  {megadeth}                          => {metallica}     0.01626667 0.5281385
## [7]  {keane}                             => {coldplay}      0.02226667 0.6374046
## [8]  {snow patrol, the killers}          => {coldplay}      0.01040000 0.5954198
## [9]  {radiohead, snow patrol}            => {coldplay}      0.01006667 0.6344538
## [10] {led zeppelin, the doors}           => {pink floyd}    0.01066667 0.5970149
## [11] {pink floyd, the doors}             => {led zeppelin}  0.01066667 0.5387205
## [12] {oasis, the killers}                => {coldplay}      0.01113333 0.6626984
## [13] {oasis, radiohead}                  => {coldplay}      0.01273333 0.5876923
## [14] {sigur rós, the beatles}            => {radiohead}     0.01046667 0.6434426
## [15] {placebo, radiohead}                => {muse}          0.01366667 0.5137845
## [16] {death cab for cutie, the killers} => {coldplay}      0.01086667 0.5884477
##      coverage   lift      count
## [1]  0.01833333  8.854413 156
## [2]  0.01800000 13.415893 156
## [3]  0.02640000  8.236292 202
## [4]  0.02666667  8.562992 203
## [5]  0.02440000  3.550304 206
## [6]  0.03080000  4.743759 244
## [7]  0.03493333  4.020634 334
## [8]  0.01746667  3.755802 156
## [9]  0.01586667  4.002021 151
## [10] 0.01786667  5.689469 160
## [11] 0.01980000  6.802027 160
## [12] 0.01680000  4.180183 167
## [13] 0.02166667  3.707058 191
## [14] 0.01626667  3.569393 157
## [15] 0.02660000  4.504247 205
## [16] 0.01846667  3.711823 163
```

We order by confidence these rules above:

```
inspect(sort(subset(reglas2, subset = lift > 1), by = "confidence"))
##      lhs                                    rhs             support
## [1]  {oasis, the killers}                => {coldplay}      0.01113333
## [2]  {sigur rós, the beatles}            => {radiohead}     0.01046667
## [3]  {keane}                             => {coldplay}      0.02226667
## [4]  {radiohead, snow patrol}            => {coldplay}      0.01006667
## [5]  {coldplay, the smashing pumpkins}   => {radiohead}     0.01093333
## [6]  {the beatles, the smashing pumpkins} => {radiohead}    0.01146667
## [7]  {bob dylan, pink floyd}             => {the beatles}   0.01033333
```

```
## [8]  {led zeppelin, the doors}            => {pink floyd}   0.01066667
## [9]  {snow patrol, the killers}           => {coldplay}     0.01040000
## [10] {bob dylan, the rolling stones}      => {the beatles}  0.01146667
## [11] {beck, the beatles}                  => {radiohead}    0.01300000
## [12] {death cab for cutie, the killers}   => {coldplay}     0.01086667
## [13] {oasis, radiohead}                   => {coldplay}     0.01273333
## [14] {coldplay, sigur rós}                => {radiohead}    0.01206667
## [15] {the pussycat dolls}                 => {rihanna}      0.01040000
## [16] {led zeppelin, the rolling stones}   => {the beatles}  0.01066667
## [17] {david bowie, pink floyd}            => {the beatles}  0.01006667
## [18] {bob dylan, radiohead}               => {the beatles}  0.01386667
## [19] {david bowie, the rolling stones}    => {the beatles}  0.01000000
## [20] {the beatles, the shins}             => {radiohead}    0.01066667
## [21] {t.i.}                               => {kanye west}   0.01040000
## [22] {radiohead, the rolling stones}      => {the beatles}  0.01060000
## [23] {travis}                             => {coldplay}     0.01373333
## [24] {the beatles, the strokes}           => {radiohead}    0.01046667
## [25] {broken social scene}                => {radiohead}    0.01506667
## [26] {pink floyd, the doors}              => {led zeppelin} 0.01066667
## [27] {the beatles, the killers}           => {coldplay}     0.01253333
## [28] {the kinks}                          => {the beatles}  0.01360000
## [29] {the flaming lips}                   => {radiohead}    0.01306667
## [30] {led zeppelin, radiohead}            => {the beatles}  0.01306667
## [31] {megadeth}                           => {metallica}    0.01626667
## [32] {snow patrol}                        => {coldplay}     0.02646667
## [33] {radiohead, the killers}             => {coldplay}     0.01506667
## [34] {simon & garfunkel}                  => {the beatles}  0.01540000
## [35] {bloc party, the killers}            => {coldplay}     0.01106667
## [36] {blur}                               => {radiohead}    0.01753333
## [37] {david bowie, radiohead}             => {the beatles}  0.01393333
## [38] {radiohead, u2}                      => {coldplay}     0.01140000
## [39] {oasis, the beatles}                 => {coldplay}     0.01060000
## [40] {the fray}                           => {coldplay}     0.01126667
## [41] {placebo, radiohead}                 => {muse}         0.01366667
## [42] {sonata arctica}                     => {nightwish}    0.01346667
## [43] {red hot chili peppers, the killers} => {coldplay}     0.01086667
## [44] {beck}                               => {radiohead}    0.02926667
## [45] {muse, the killers}                  => {coldplay}     0.01513333
## [46] {judas priest}                       => {iron maiden}  0.01353333
## [47] {muse, the beatles}                  => {radiohead}    0.01380000
## [48] {pink floyd, the doors}              => {the beatles}  0.01000000
## [49] {death cab for cutie, the shins}     => {radiohead}    0.01006667
## [50] {death cab for cutie, the beatles}   => {radiohead}    0.01246667
##      confidence coverage   lift      count
## [1]  0.6626984  0.01680000  4.180183 167
## [2]  0.6434426  0.01626667  3.569393 157
## [3]  0.6374046  0.03493333  4.020634 334
## [4]  0.6344538  0.01586667  4.002021 151
## [5]  0.6283525  0.01740000  3.485683 164
```

```
## [6]  0.6209386  0.01846667  3.444556 172
## [7]  0.6150794  0.01680000  3.458092 155
## [8]  0.5970149  0.01786667  5.689469 160
## [9]  0.5954198  0.01746667  3.755802 156
## [10] 0.5910653  0.01940000  3.323081 172
## [11] 0.5909091  0.02200000  3.277972 195
## [12] 0.5884477  0.01846667  3.711823 163
## [13] 0.5876923  0.02166667  3.707058 191
## [14] 0.5801282  0.02080000  3.218167 181
## [15] 0.5777778  0.01800000 13.415893 156
## [16] 0.5776173  0.01846667  3.247474 160
## [17] 0.5741445  0.01753333  3.227949 151
## [18] 0.5730028  0.02420000  3.221530 208
## [19] 0.5703422  0.01753333  3.206572 150
## [20] 0.5673759  0.01880000  3.147425 160
## [21] 0.5672727  0.01833333  8.854413 156
## [22] 0.5638298  0.01880000  3.169958 159
## [23] 0.5628415  0.02440000  3.550304 206
## [24] 0.5607143  0.01866667  3.110471 157
## [25] 0.5472155  0.02753333  3.035589 226
## [26] 0.5387205  0.01980000  6.802027 160
## [27] 0.5340909  0.02346667  3.368950 188
## [28] 0.5298701  0.02566667  2.979030 204
## [29] 0.5297297  0.02466667  2.938589 196
## [30] 0.5283019  0.02473333  2.970213 196
## [31] 0.5281385  0.03080000  4.743759 244
## [32] 0.5251323  0.05040000  3.312441 397
## [33] 0.5243619  0.02873333  3.307582 226
## [34] 0.5238095  0.02940000  2.944956 231
## [35] 0.5236593  0.02113333  3.303150 166
## [36] 0.5228628  0.03353333  2.900496 263
## [37] 0.5225000  0.02666667  2.937594 209
## [38] 0.5213415  0.02186667  3.288529 171
## [39] 0.5196078  0.02040000  3.277594 159
## [40] 0.5168196  0.02180000  3.260006 169
## [41] 0.5137845  0.02660000  4.504247 205
## [42] 0.5101010  0.02640000  8.236292 202
## [43] 0.5093750  0.02133333  3.213047 163
## [44] 0.5092807  0.05746667  2.825152 439
## [45] 0.5089686  0.02973333  3.210483 227
## [46] 0.5075000  0.02666667  8.562992 203
## [47] 0.5073529  0.02720000  2.814458 207
## [48] 0.5050505  0.01980000  2.839489 150
## [49] 0.5033333  0.02000000  2.792160 151
## [50] 0.5013405  0.02486667  2.781105 187
```

Recommendation to users who listen to Coldplay?:

```
r1 <- subset(reglas2, subset = lhs %pin%
  c("cold"))
inspect(r1)
##     lhs                                    rhs          support    confidence
## [1] {coldplay, the smashing pumpkins} => {radiohead} 0.01093333 0.6283525
## [2] {coldplay, sigur rós}             => {radiohead} 0.01206667 0.5801282
##     coverage lift     count
## [1] 0.0174   3.485683 164
## [2] 0.0208   3.218167 181
```

Try other groups.

8. Formal concept analysis

8.1 Formal Concept Analysis in `R`

FCA is a mathematical tool based on lattice theory and logic. **It is the forgotten sister in data science.**

However, FCA allows to discover hidden knowledge in a dataset in a similar (or even more powerful) way than some very famous Machine Learning techniques such as, for example, association rules.

Why develop an `R` package for FCA?

- Helping to disseminate FCA as a new knowledge discovery tool
- Help to integrate methods from different areas that do not really have empty intersection
- Help the more theoretical FCA community to calculate concepts, implications, minimal generators, etc.
- Applying FCA to real problems: development of recommender systems in tourism and medicine

For us it is useful to:

- disseminate our theoretical works to a wider audience not only in the FCA community
- close the cycle: **theory-practice-theory**.

8.2 Any rule management packages?

- `arules`
- `frbs`

- `RKEEL`

We highlight **arules** well known in the data science community for extracting and managing patterns and association rules.

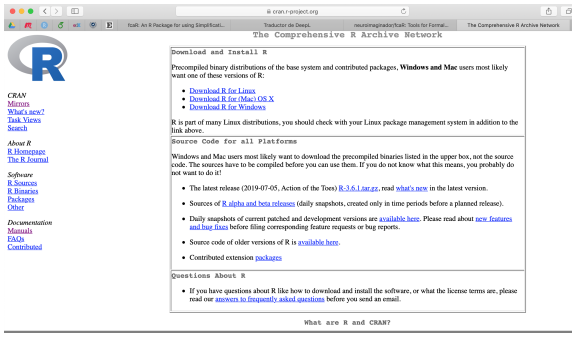
8.3 The fcaR package

The package is in a stable phase in a repository on Github and on CRAN.

- Unit tests
- Bullets with demos
- Status:
  - lifecycle: stable
  - CRAN: 1.1.1
  - build: passing
  - downloads: ~13K
- Installation:

```
# Latest version
remotes::install_github("Malaga-FCA-group/fcaR", build_vignettes = TRUE)

# Stable version
install.packages("fcaR")
```

The package is also available from CRAN.



8.4 Extensibility

- Object-oriented programming: extensible classes.
- Use of registry to extend functionality:
  - Redundancy elimination methods.
  - Methods for constructing the concept lattice.
  - Techniques for finding the implication base.

8.5 fcaR

The first important issue regarding transaction datasets used in association rules: in association rules the datasets are either binary or with categorical or discrete information that can be passed to binary.

FCA is able to work with binary or fuzzy information stored in the datasets and extract implications (association rules with confidence one) and also a concept lattice.

8.5.1 Formal Context

We will use the following binary dataset called `planets` which appears in:

> Wille R (1982). "Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts." In Ordered Sets, pp. 445–470. Springer.

```
knitr::kable(planets, format = "html", booktabs = TRUE)
```

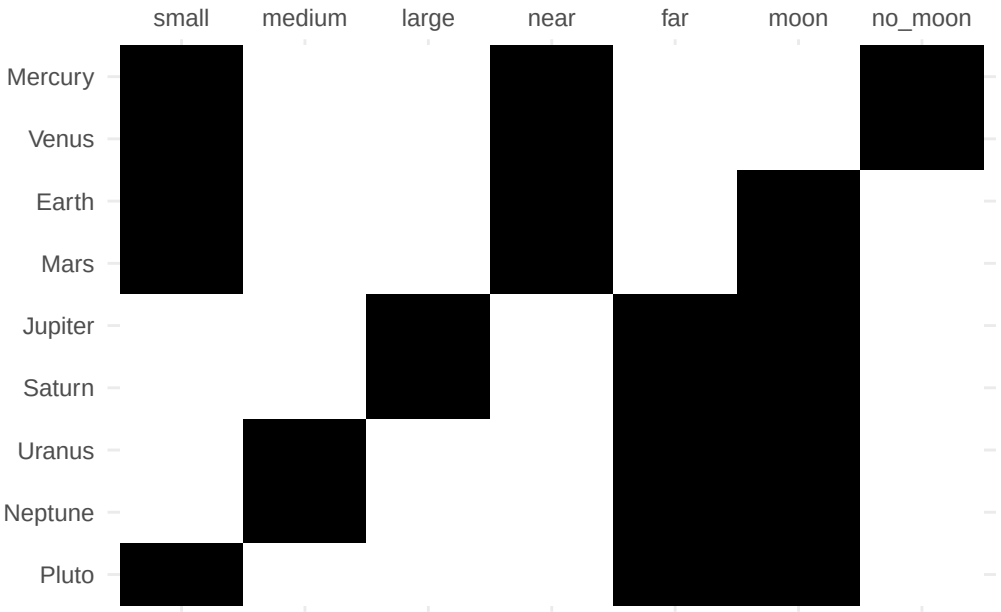
	small	medium	large	near	far	moon	no_moon
Mercury	1	0	0	1	0	0	1
Venus	1	0	0	1	0	0	1
Earth	1	0	0	1	0	1	0
Mars	1	0	0	1	0	1	0
Jupiter	0	0	1	0	1	1	0
Saturn	0	0	1	0	1	1	0
Uranus	0	1	0	0	1	1	0
Neptune	0	1	0	0	1	1	0
Pluto	1	0	0	0	1	1	0

We create a formal context from the above dataset using the `fcaR` package:

```
library(fcaR)
fc_planets <- FormalContext$new(planets)
fc_planets$print()
## FormalContext with 9 objects and 7 attributes.
##          small  medium  large  near  far  moon  no_moon
##  Mercury   X                     X                 X
##    Venus   X                     X                 X
##    Earth   X                     X          X
##     Mars   X                     X          X
##  Jupiter                  X           X     X
##   Saturn                  X           X     X
##   Uranus           X                  X     X
##  Neptune           X                  X     X
##    Pluto   X                         X     X
```

- In the formal context `fc_planets` we have some properties and methods that can be used to extract hidden knowledge in the dataset.

```
fc_planets$dim()
## [1] 9 7
fc_planets$attributes
## [1] "small"   "medium"  "large"   "near"    "far"     "moon"    "no_moon"
fc_planets$objects
## [1] "Mercury" "Venus"   "Earth"   "Mars"    "Jupiter" "Saturn"  "Uranus"
## [8] "Neptune" "Pluto"
fc_planets$plot()
```



For Latex users:

```
fc_planets$to_latex(table = FALSE)
## \begin{tabular}{r|ccccccc}
##   & $small$ & $medium$ & $large$ & $near$ & $far$ & $moon$ & $no\_moon$\\
##  \hline
##  $Mercury$ & $\times$ &  &  & $\times$ &  &  & $\times$ \\
## $Venus$ & $\times$ &  &  & $\times$ &  &  & $\times$ \\
## $Earth$ & $\times$ &  &  & $\times$ &  & $\times$ &  \\
## $Mars$ & $\times$ &  &  & $\times$ &  & $\times$ &  \\
## $Jupiter$ &  &  & $\times$ &  & $\times$ & $\times$ &  \\
## $Saturn$ &  &  & $\times$ &  & $\times$ & $\times$ &  \\
## $Uranus$ &  & $\times$ &  &  & $\times$ & $\times$ &  \\
## $Neptune$ &  & $\times$ &  &  & $\times$ & $\times$ &  \\
## $Pluto$ & $\times$ &  &  &  & $\times$ & $\times$ &
##  \end{tabular}
```

8.5.2 Concepts

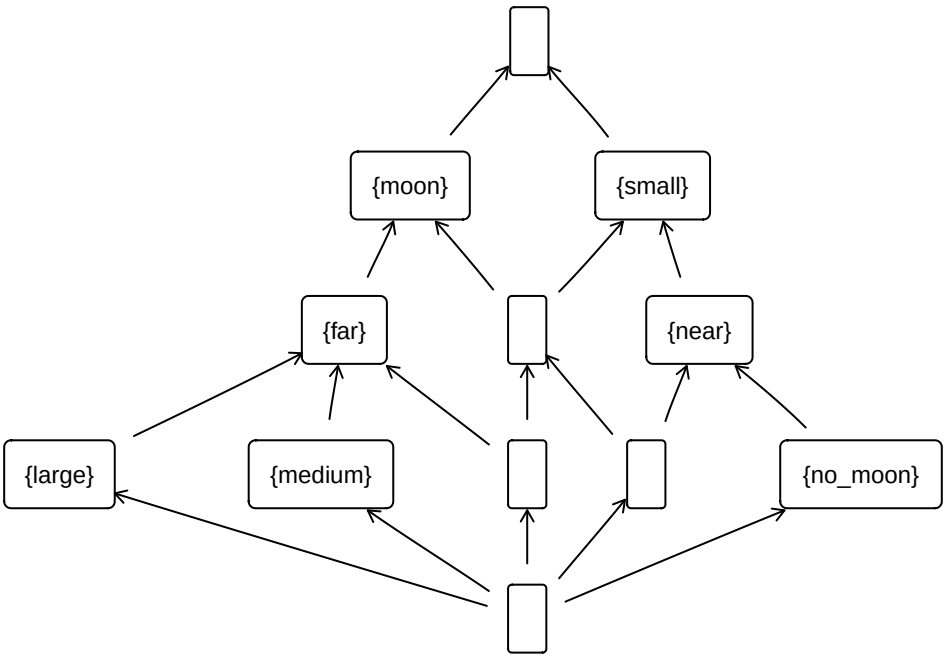
We can calculate concepts using Ganter and Wille's **NextClosure** algorithm, implemented in the `find_concepts()` method, which is developed in `C` behind `R`.

```
fc_planets$find_concepts()
fc_planets$concepts$size()
## [1] 12
head(fc_planets$concepts)
## A set of 6 concepts:
## 1: ({Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto}, {})
## 2: ({Earth, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto}, {moon})
## 3: ({Jupiter, Saturn, Uranus, Neptune, Pluto}, {far, moon})
## 4: ({Jupiter, Saturn}, {large, far, moon})
## 5: ({Uranus, Neptune}, {medium, far, moon})
## 6: ({Mercury, Venus, Earth, Mars, Pluto}, {small})
```

- FCA showed that there is an ordering relationship in the concepts.
- A concept lattice (concept taxonomy) has actually been calculated.

Plot of the concept lattice

```
fc_planets$concepts$plot(object_names = FALSE)
```




```
fc_planets$concepts$sub(2)
## {{Earth, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto}, {moon}}
```

There is a set of planets with common characteristics - we have discovered hidden knowledge: does this concept already have a name?

A concept (A, B), contains a set of objects A (the *extent*) and a set of attributes B (the *intent*) such that:

- the *extent* A consists of all objects that share the attributes in B.
- the *intent* B consists of all the attributes shared by the objects in A.

Closures

The basic operation in FCA is the calculation of closures from a set of attributes or from a set of objects. Two mathematical functions (two derivation operators) are used, which we have called in our library `extent()` (from objects), `intent()` (from attributes).

The `extent()` of a set of objects is the set of their common attributes:

```
# Define a set of objects
S <- Set$new(attributes = fc_planets$objects)
S$assign(Earth = 1, Mars = 1)
S
## {Earth, Mars}

# Compute the intent of S
```

```
fc_planets$intent(S)
## {small, near, moon}
```

With this we can see what common properties Earth and Mars have in common.

Analogously, the `intent()` of an attribute set is the set of objects that possesses all the attributes in the set:

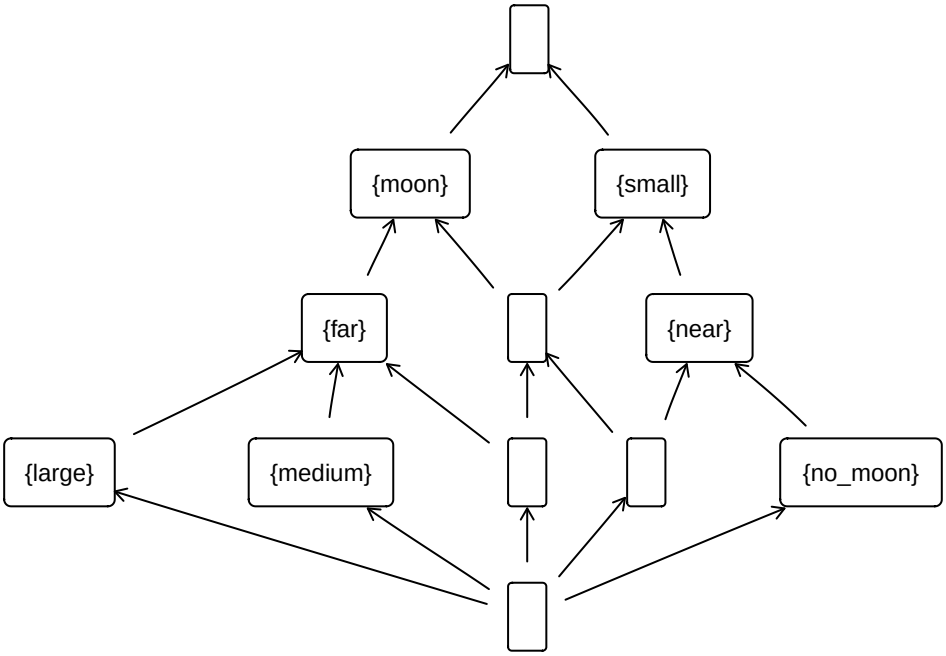
```
# Define a set of objects
S <- Set$new(attributes = fc_planets$attributes)
S$assign(moon = 1, large = 1)
S
## {large, moon}

# Compute the extent of S
fc_planets$extent(S)
## {Jupiter, Saturn}
```

Planets that have the characteristics indicated.

Operations with the lattice

All basic functions developed in the FCA theory by Ganter and Wille have been implemented:

```
fc_planets$concepts$support()
##  [1] 1.0000000 0.7777778 0.5555556 0.2222222 0.2222222 0.5555556 0.3333333
##  [8] 0.1111111 0.4444444 0.2222222 0.2222222 0.0000000

# Get the index of those concepts with support
# greater than the threshold
idx <- which(fc_planets$concepts$support() > 0.2)
# Build the sublattice
sublattice <- fc_planets$concepts$sublattice(idx)
sublattice
## A set of 12 concepts:
## 1: ({Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto}, {})
## 2: ({Earth, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto}, {moon})
## 3: ({Jupiter, Saturn, Uranus, Neptune, Pluto}, {far, moon})
## 4: ({Jupiter, Saturn}, {large, far, moon})
## 5: ({Uranus, Neptune}, {medium, far, moon})
## 6: ({Mercury, Venus, Earth, Mars, Pluto}, {small})
## 7: ({Earth, Mars, Pluto}, {small, moon})
## 8: ({Pluto}, {small, far, moon})
## 9: ({Mercury, Venus, Earth, Mars}, {small, near})
## 10: ({Mercury, Venus}, {small, near, no_moon})
## 11: ({Earth, Mars}, {small, near, moon})
## 12: ({}, {small, medium, large, near, far, moon, no_moon})
sublattice$plot()
```




```
# The fifth concept
C <- fc_planets$concepts$sub(5)
C
## ({Uranus, Neptune}, {medium, far, moon})
# Its subconcepts:
fc_planets$concepts$subconcepts(C)
## A set of 2 concepts:
## 1: ({Uranus, Neptune}, {medium, far, moon})
## 2: ({}, {small, medium, large, near, far, moon, no_moon})
# And its superconcepts:
fc_planets$concepts$superconcepts(C)
## A set of 4 concepts:
## 1: ({Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto}, {})
## 2: ({Earth, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto}, {moon})
## 3: ({Jupiter, Saturn, Uranus, Neptune, Pluto}, {far, moon})
## 4: ({Uranus, Neptune}, {medium, far, moon})
# A list of concepts
C <- fc_planets$concepts[5:7]
C
## A set of 3 concepts:
## 1: ({Uranus, Neptune}, {medium, far, moon})
## 2: ({Mercury, Venus, Earth, Mars, Pluto}, {small})
## 3: ({Earth, Mars, Pluto}, {small, moon})
# Supremum of the concepts in C
fc_planets$concepts$supremum(C)
## ({Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto}, {})
# Infimum of the concepts in C
fc_planets$concepts$infimum(C)
## ({}, {small, medium, large, near, far, moon, no_moon})
fc_planets$concepts$join_irreducibles()
## A set of 5 concepts:
```

```
## 1: ({Jupiter, Saturn}, {large, far, moon})
## 2: ({Uranus, Neptune}, {medium, far, moon})
## 3: ({Pluto}, {small, far, moon})
## 4: ({Mercury, Venus}, {small, near, no_moon})
## 5: ({Earth, Mars}, {small, near, moon})
fc_planets$concepts$meet_irreducibles()
## A set of 7 concepts:
## 1: ({Earth, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto}, {moon})
## 2: ({Jupiter, Saturn, Uranus, Neptune, Pluto}, {far, moon})
## 3: ({Jupiter, Saturn}, {large, far, moon})
## 4: ({Uranus, Neptune}, {medium, far, moon})
## 5: ({Mercury, Venus, Earth, Mars, Pluto}, {small})
## 6: ({Mercury, Venus, Earth, Mars}, {small, near})
## 7: ({Mercury, Venus}, {small, near, no_moon})
```

8.5.3 Implications. Duquenne-Guigues basis

The **NextClosure** algorithm is used to compute at the same cost not only concepts but also implications.

The `find_implications()` method is applied to a *formal context object (of class `FormalContext`).

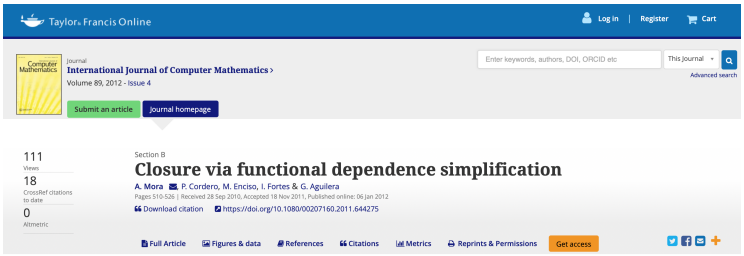
```
fc_planets$find_implications()
```

The result (a set of association rules with confidence 1) is stored in `fc_planets$implications`.

```
fc_planets$implications$cardinality()
## [1] 10
fc_planets$implications
## Implication set with 10 implications.
## Rule 1: {no_moon} -> {small, near}
## Rule 2: {far} -> {moon}
## Rule 3: {near} -> {small}
## Rule 4: {large} -> {far, moon}
## Rule 5: {medium} -> {far, moon}
## Rule 6: {medium, large, far, moon} -> {small, near, no_moon}
## Rule 7: {small, near, moon, no_moon} -> {medium, large, far}
## Rule 8: {small, near, far, moon} -> {medium, large, no_moon}
## Rule 9: {small, large, far, moon} -> {medium, near, no_moon}
## Rule 10: {small, medium, far, moon} -> {large, near, no_moon}
fc_planets$implications[2:4]
## Implication set with 3 implications.
## Rule 1: {far} -> {moon}
## Rule 2: {near} -> {small}
## Rule 3: {large} -> {far, moon}
```

Simplification Logic in R

We propose Simplification Logic as a tool for developing automatic reasoning methods.



For the case of binary datasets, we have proposed (Ángel Mora, Pablo Cordero, Manuel Enciso, *et. al.*) the **Logic of Simplifications**, in 2012.

- **Axiom**: infer $A \cup B \Rightarrow A$

- **Multiplication**: from $A \Rightarrow B$ infer $c^* \otimes A \Rightarrow c^* \otimes B$
- **Simplification**: from $A \Rightarrow B$ and $C \Rightarrow D$ infer $(C \setminus B) \Rightarrow (D \setminus B)$ if $A \subseteq C$
- **Rsimplification**: from $A \Rightarrow B$ and $C \Rightarrow D$ infer $C \Rightarrow (D \setminus B)$ if $A \subseteq (C \cup D)$

The **Simplifications Logic** is the engine of automatic methods for:

- Elimination of redundancy,
- Calculation of attribute closures,
- Minimal key calculation,
- Obtaining recommendations, etc.

In `fcaR` the following equivalence rules have been implemented, based on the axioms of logic:

- **Reduction**: from $A \Rightarrow B$, infer $A \Rightarrow B \setminus A$
- **Generalization**: from $A \Rightarrow B$ and $C \Rightarrow D$, if $A \subseteq C$ and $D \subseteq B$, remove $C \Rightarrow D$.
- **Composition**: if $A \Rightarrow B$ and $A \Rightarrow C$, replace both implications by $A \Rightarrow B \cup C$.
- **Simplification**: if $A \Rightarrow B$ and $C \Rightarrow D$, with $A \subset C$ and $A \cap B = \varnothing$, replace $C \Rightarrow D$ by $C \setminus B \Rightarrow D \setminus B$.

```
fc_planets$plot()
# Average number of attributes on left and right hand side
colSums(fc_planets$implications$size())
# We apply the logic of simplification
fc_planets$implications$apply_rules(c("reduction",
                                      "composition",
                                      "generalization",
                                      "simplification",
                                      "rsimplification"))
# Average number of attributes in left and right hand side, after simplification
colSums(fc_planets$implications$size())
```

8.5.4 Support for the `arules` package

- Import/Export `transactions` from `arules`.
- Import rules from `arules`.
- Manipulate the rules with our algorithms (use Simplification Logic, closures, etc.).
- Subsequently export to `arules`.

```
library(arules)
## Loading required package: Matrix
## Warning: package 'Matrix' was built under R version 4.3.1
##
## Attaching package: 'Matrix'
## The following object is masked from 'package:fcaR':
##
##     %&%
##
## Attaching package: 'arules'
## The following objects are masked from 'package:base':
##
##     abbreviate, write
data("Mushroom")
rules_apriori <- apriori(Mushroom, parameter = list(support = 0.3, confidence = 1))
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##           1    0.1    1 none FALSE            TRUE       5     0.3      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 2437
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[114 item(s), 8124 transaction(s)] done [0.00s].
## sorting and recoding items ... [26 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 7 8 9 done [0.00s].
## writing ... [4059 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
inspect(head(rules_apriori))
##     lhs                      rhs                   support   confidence coverage
## [1] {}                    => {VeilType=partial}  1.0000000 1          1.0000000
## [2] {GillSize=narrow}     => {RingNumber=one}    0.3092073 1          0.3092073
## [3] {GillSize=narrow}     => {GillAttached=free} 0.3092073 1          0.3092073
## [4] {GillSize=narrow}     => {VeilType=partial}  0.3092073 1          0.3092073
## [5] {CapSurf=smooth}      => {VeilType=partial}  0.3146233 1          0.3146233
## [6] {RingType=evanescent} => {GillAttached=free} 0.3417036 1          0.3417036
##     lift     count
## [1] 1.000000 8124
## [2] 1.084936 2512
## [3] 1.026535 2512
## [4] 1.000000 2512
```

```
## [5] 1.000000 2556
## [6] 1.026535 2776
```

arules to fcaR

```
fc_Mushroom <- FormalContext$new(Mushroom)
fc_Mushroom$implications$add(rules_apriori)
fc_Mushroom$implications[2:5]
## Implication set with 4 implications.
## Rule 1: {GillSize=narrow} -> {RingNumber=one}
## Rule 2: {GillSize=narrow} -> {GillAttached=free}
## Rule 3: {GillSize=narrow} -> {VeilType=partial}
## Rule 4: {CapSurf=smooth} -> {VeilType=partial}
fc_Mushroom$implications$cardinality()
## [1] 4059
fc_Mushroom$implications$apply_rules(c(
  "reduction",
  "composition",
  "generalization",
  "simplification",
  "rsimplification"
))
## Processing batch
## --> Reduction: from 4059 to 4059 in 0 secs.
## --> Composition: from 4059 to 2168 in 0.241 secs.
## --> Generalization: from 2168 to 71 in 0.177 secs.
## --> Simplification: from 71 to 55 in 0.045 secs.
## --> Right Simplification: from 55 to 55 in 0.05 secs.
## Batch took 0.514 secs.
fc_Mushroom$implications$cardinality()
## [1] 55
```

fcaR to arules

```
arules_rules <- fc_Mushroom$implications$to_arules()
arules_rules
## set of 55 rules
class(arules_rules)
## [1] "rules"
## attr(,"package")
## [1] "arules"
```

8.5.5 Recommender system for medical diagnostics

Dataset `cobre32`

A diagnostic system from a formal context containing fuzzy medical data.

The dataset `cobre32` has:

- 105 objects (patients) in rows.

- 30 attributes related to signs or symptoms, and 2 attributes related to diagnosis: `dx_ss` (strict schizophrenia) and `dx_other` (bipolar or schizoaffective disorder).

For each symptom-related attribute, there are several possible grades, from *absent* to *extreme*, through *minimal*, *mild*, *moderate*, *moderate severe* and *severe*, which are mapped into the interval $[0,1]$.

```
data("cobre32")
colnames(cobre32)
##  [1] "COSAS_1"   "COSAS_2"   "COSAS_3"   "COSAS_4"   "COSAS_5"   "COSAS_6"
##  [7] "COSAS_7"   "FICAL_1"   "FICAL_2"   "FICAL_3"   "FICAL_4"   "FICAL_5"
## [13] "FICAL_6"   "FICAL_7"   "FICAL_8"   "FICAL_9"   "SCIDII_10" "SCIDII_11"
## [19] "SCIDII_12" "SCIDII_13" "SCIDII_14" "SCIDII_15" "SCIDII_16" "SCIDII_17"
## [25] "SCIDII_18" "SCIDII_19" "SCIDII_20" "SCIDII_21" "SCIDII_22" "SCIDII_23"
## [31] "dx_ss"     "dx_other"
```

```
# We create the formal context
fc <- FormalContext$new(cobre32)

# We extract both the concept lattice and the set of implications.
fc$find_implications()
fc$concepts$size()
## [1] 14706
fc$implications$cardinality()
## [1] 985

# We can apply equivalences to get a simpler set of implications.
fc$implications$apply_rules(rules = c("composition", "simplification"))
## Processing batch
## --> Composition: from 985 to 985 in 0.001 secs.
## --> Simplification: from 985 to 985 in 0.138 secs.
## Batch took 0.139 secs.
fc$implications$cardinality()
## [1] 985
```

Recommendations

From the extracted set of rules, we can define a function that encapsulates the behaviour of the recommender system:

```
# Here S is the set of known attributes for a new individual.
# Taking the implications, the consequent containing one of the possible diagnoses is cal
diagnose <- function(S) {
              fc$implications$recommend(S = S,
                                        attribute_filter = c("dx_ss","dx_other"))

}
```

Let's see how it works with some examples.

Let us consider a first patient:

```
# We create a vector (set) for the patient's attributes
Patient1 <- Set$new(attributes = fc$attributes)

# We add those that are known
# (absent = 0, extreme = 1,...)
Patient1$assign(COSAS_1 = 0.5,
                COSAS_2 = 1,
                COSAS_3 = 0.5,
                COSAS_4 = 0.166667,
                COSAS_5 = 0.5,
                COSAS_6 = 1)

# Finally, we can run the recommender system
diagnose(Patient1)
##    dx_ss dx_other
##        1        0
```

The result tells us that this patient should be diagnosed as strict schizophrenia.

A second patient:

```
# Empty set
Patient2 <- Set$new(attributes = fc$attributes)

# We add the symptoms or signs:
Patient2$assign(FICAL_2 = 1)

# And we diagnose using the implication-based system.
diagnose(Patient2)
##    dx_ss dx_other
##        0        0
```

In this case, the sign that has been evaluated in this patient does not indicate (according to the set of implications that model the knowledge of the problem) any diagnosis. The problem remains open, and we will see how to solve it later.

Finally, a third patient:

```
Patient3 <- Set$new(attributes = fc$attributes)

Patient3$assign(COSAS_4 = 0.6666667,
                FICAL_3 = 0.5,
                FICAL_5 = 0.5,
                FICAL_8 = 0.5)


diagnose(Patient3)
##    dx_ss dx_other
##        0        1
```

In this case, the patient could be diagnosed with a disorder other than strict schizophrenia.

Closure with Simplification Logic - more knowledge

When we calculate the closure of a set of attributes using Simplification Logic, we also reduce the set of implications that contain important knowledge (this is the core of other automated methods).

This therefore provides us with additional knowledge that helps us to solve the problem that appeared earlier in `Patient2`.

```
# Here we get not only the closure, but a simplified subset of the implications that are
cl <- fc$implications$closure(Patient2,
                              reduce = TRUE)

# We write the first rules
new_rules <- cl$implications$filter(rhs = c("dx_ss", "dx_other"),
                                    drop = TRUE)

new_rules[c(2, 5, 12:16)]
## Implication set with 7 implications.
## Rule 1: {SCIDII_19 [0.33]} -> {dx_ss, dx_other}
## Rule 2: {SCIDII_13 [0.33]} -> {dx_ss, dx_other}
## Rule 3: {SCIDII_14} -> {dx_ss}
## Rule 4: {SCIDII_14, SCIDII_18} -> {dx_other}
## Rule 5: {FICAL_1 [0.33], FICAL_4 [0.33], SCIDII_10 [0.5], SCIDII_18 [0.5],
##   dx_ss} -> {dx_other}
## Rule 6: {COSAS_7 [0.33]} -> {dx_ss, dx_other}
## Rule 7: {COSAS_6 [0.5], SCIDII_20 [0.5], dx_ss} -> {dx_other}
```

Some of these inferred rules can help physicians make the diagnosis.

A. Practice: Hypothesis Testing

The objective of this practice is to consolidate the knowledge acquired in parametric and non-parametric hypothesis testing, apply various statistical tests to different types of data, and interpret the results of these tests to make data-driven decisions.

A.1 Deliverables

You must submit a PDF document containing the entire data analysis process related to the questions posed in the **Exercises** section below.

To generate the PDF document, you must use **R Markdown** or **Quarto**. This allows you to easily interweave explanations and code with results.

The PDF document will be uploaded to a task in the virtual campus.

A.2 Suggested Workflow

For each dataset, it is recommended to follow these analysis steps:

1. **Data exploration:** Use functions like `summary`, `str`, `head`, `tail`, and graphs to familiarize yourself with the data.
2. **Data cleaning:** Identify and handle missing values, outliers, and any other data issues. *Note:* this step is not necessary for this practice.
3. **Hypothesis formulation:** Clearly define the null and alternative hypotheses you want to test.
4. **Statistical test selection:** Choose the appropriate test based on the data type and hypotheses.
5. **Test execution:** Use R functions to perform the statistical test.
6. **Result interpretation:** Analyze the confidence interval and other parameters to make a decision about the null hypothesis.

7. **Visualization:** Use graphs to effectively communicate the results.

A.3 Datasets

This section describes the datasets used in this practice, as well as their online availability.

A.3.1 Titanic

A classic dataset that offers a rich variety of categorical and numerical variables.

The Titanic dataset is a collection of information about the passengers and crew aboard the famous transatlantic liner that sank in the North Atlantic in 1912. This dataset has become a classic in the field of data science and machine learning, frequently used to teach data analysis techniques and to explore research questions related to survival, socioeconomic factors, and decision-making in critical situations.

What does the dataset contain?

The Titanic dataset typically includes variables such as:

- Passenger information: Name, age, sex, social class (first, second, third), whether they had a partner or family members on board, ticket fare, port of embarkation, among others.
- Survival information: Whether the passenger survived the shipwreck or not.

What is the dataset used for?

This dataset is used for various purposes, including:

- **Exploratory data analysis:** To understand the characteristics of the passengers and their relationship to survival.
- **Predictive modeling:** To build models that predict the probability of a passenger surviving based on their characteristics.
- **Data visualization:** To create graphs and visualizations that illustrate the relationships between variables and tell the story of the disaster.

Download:

Available from [https://www.kaggle.com/c/titanic/data].

A.3.2 HR Analytics Job Prediction

This dataset is an invaluable tool for analyzing various factors that influence job satisfaction and employee turnover in an organization. Like the Titanic dataset, it has become a benchmark for those who want to learn and practice data analysis techniques in the field of human resources.

What does the dataset contain?

The HR Analytics Job Prediction dataset typically includes variables such as:

- Employee information: Age, gender, education level, department, role, years of experience, salary, etc.
- Performance information: Performance evaluation, number of projects, hours worked, etc.

- Satisfaction information: Level of job satisfaction, satisfaction and dissatisfaction factors, etc.
- Turnover information: Whether the employee left the company or not.

What is the dataset used for?

This dataset is used for various purposes, including:

- **Exploratory data analysis:** To understand the characteristics of employees and their relationship to job satisfaction and turnover.
- **Predictive modeling:** To build models that predict the probability of an employee leaving the company based on their characteristics.
- **Human resources optimization:** To identify the factors that most influence job satisfaction and take measures to improve the work environment.

Download:

Available at this Kaggle page.

A.4 Functions to use

The following functions will be the basics for performing the proposed analyses.

- `t.test`: To compare means of one or two samples.
- `wilcox.test`: To compare means of one or two samples when normality is not assumed.
- `var.test`: To compare variances of two samples.
- `prop.test`: To compare proportions of one or two samples.
- `binom.test`: To test hypotheses about a proportion of a binomial sample.
- `chisq.test`: To test the independence between two categorical variables or goodness of fit.
- `ks.test`: To test if two samples come from the same distribution.

It is recommended that you use R's help to learn how to use them, and seek help online (or from the professor) on when and how to use each function.

A.5 Exercises

Here are some questions that can be answered using the hypothesis testing tools seen in class.

The idea is that each question translates into an analysis (hypothesis test) and this translates into a series of R commands, using the functions already mentioned and other auxiliary ones.

A.5.1 Titanic

Analyse the following questions.

Survival

1. Is the proportion of survivors significantly different from 50%? *Hint: Use* `binom.test`.
2. Is there a significant difference in the average age of survivors and the deceased? *Hint: Use* `t.test` *or* `wilcox.test` *as appropriate.*

3. Are the variances of age equal between survivors and deceased? *Hint: Use* `var.test`.

Social class

1. Is the distribution of social class among passengers uniform? *Hint: Use* `chisq.test`.
2. Is there a significant difference in the proportion of passengers traveling alone among the different social classes? *Hint: Use* `prop.test`.

Other questions of interest

1. Comparison of ages: Is there a significant difference in the average age of passengers traveling in first and third class?
2. Survival by gender and class: Does the interaction between gender and social class influence the probability of survival?
3. Comparison of fares: Is there a significant difference in the average fares paid by passengers who survived and those who died?

A.5.2 HR Analytics Job Prediction

This dataset allows us to analyze various factors that influence job satisfaction and employee turnover.

Satisfaction and performance evaluation

1. Is there a significant difference in the level of satisfaction between employees with a high performance evaluation and those with a low evaluation?
2. Do employees with an outstanding performance evaluation have a significantly higher probability of being satisfied with their job?

Hours worked and satisfaction

1. Are employees who work more hours significantly less satisfied with their job?
2. Is there a significant difference in the level of satisfaction between employees who work full-time and those who work part-time?

Number of projects and satisfaction

1. Is there a negative relationship between the number of projects assigned to an employee and their level of satisfaction?
2. Are employees who work on more than three projects simultaneously significantly less satisfied?

Employee turnover and satisfaction

1. Did employees who decided to leave the company have a significantly lower level of satisfaction before leaving?
2. Is there a significant difference in the level of satisfaction between employees who stayed with the company and those who left?

A.6 Additional tasks

As a complement to the previous discussions, it is advisable to consider the following:

1. **Visualizations:** Use the {ggplot2} package to create histograms, boxplots, and other graphs to help visualize the data and the results of the contrasts.

2. **Confidence intervals:** Construct confidence intervals for the estimated parameters to support the explanations given.

B. Practice: Regression

The objective of this practice is to consolidate the knowledge acquired about regression.

B.1 Deliverables

You must submit a PDF document containing the entire data analysis process related to the questions posed in the **Exercises** section below.

To generate the PDF document, you must use **R Markdown** or **Quarto**. This allows you to easily interweave explanations and code with results.

The PDF document will be uploaded to a task in the virtual campus.

B.2 Dataset

We will explore a *dataset* of volumetric brain measurements of a number of individuals, which fall into two groups (according to the variable `CLASS`): the value `HEALTHY` indicates a healthy individual, while the value `AD` indicates Alzheimer's disease (AD stands for *Alzheimer's Disease*).

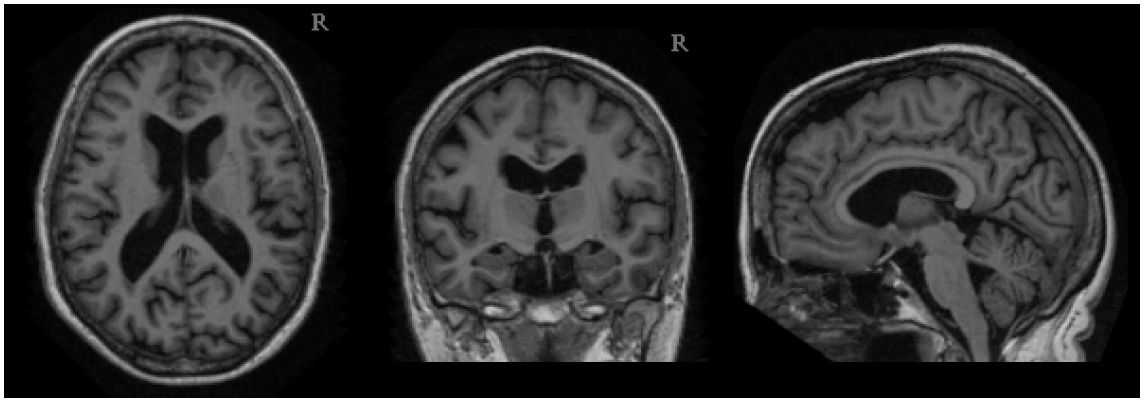


Figure B.1.: Brain images from the OASIS website.

These data come from calculating around 230 morphometric estimates (volume and thickness of anatomical brain regions) of some 260 individuals, belonging to the OASIS project.

B.3 Objective

The purpose of this project is to use the advanced statistical techniques seen in class to:

- Establish the existing relationship between volumetric variables and the age and sex of each individual.
- Discover *markers* among volumetric measures of cognitive status (healthy, Alzheimer's)
- Generate a classifier model of cognitive status to predict, from the volumetric variables, whether an individual has Alzheimer's or is healthy.
- Conduct a small Bayesian analysis of the epidemiology/incidence of Alzheimer's disease.

B.4 Exercises

Import the *dataset* `Alzheimer.csv`. (**Help**: use `read_csv()` from the `readr` package).

Answer the following questions and accompany them with explanatory graphs where appropriate.

B.4.1 Descriptive statistics

The first step is to understand the *dataset* and become familiar with it.

- What types of variables appear in the dataset? For numeric variables, perform a brief descriptive statistic (**Help**: use `summary()`).

B.4.2 Hypothesis testing

We aim to establish markers or indicators of an individual's condition. Basically, we want to test whether brain atrophy is specific to Alzheimer's disease, compared to healthy subjects.

- For the two groups, `HEALTHY` and `AD`, you want to test (by appropriate hypothesis tests) whether the brain volume variable (`BRAIN_VOLUME`) is significantly lower in

subjects with Alzheimer's disease than in healthy subjects (perform a separate analysis for each of the sexes in the dataset). Accompany with a `boxplot()` to observe the differences. (**Help**: use the `filter()` function of the `dplyr` package to filter the dataset to leave only one sex, perform the hypothesis test, and repeat with the other sex).
- Repeat the above analysis, but to test whether the grey matter or white matter volume variables are different between healthy subjects and subjects with AD (perform one test for each variable). (**Help**: Simply repeat the previous step, with minimal changes).

Is the incidence of the disease different in patients of different sexes?

- Are the variables `SEX` and `CLASS` independent? Use a χ^2 test to test for independence and correlation between the two.

B.4.3 Regression

We are considering establishing a pattern of annual (i.e. age-dependent) atrophy in the brain. How does the volume of the brain vary with age?

- We want to establish a regression model that uses as predictors the variables *sex* and *age* to estimate the value of brain volume (variable `BRAIN_VOLUME`). Use simple linear and polynomial regression and compare the models using `anova()`, detecting if all variables are significant and commenting and interpreting the known goodness-of-fit and error measures.
- Repeat the previous step to find models to estimate the variable `GM_VOLUME` (grey matter volume) and models to estimate `WM_VOLUME` (white matter volume) from age and sex.

On the other hand, we want to know whether the volumetric parameters of the brain are good indicators of age.

- The inverse analysis to the previous ones is proposed. Using as predictors the variables of brain volume, grey and white matter, as well as sex, study regression models (at least 3, taking into account possible interactions and creating a polynomial model, as a minimum) where the age of the individual in question is estimated. Also, accompany with graphs. Establish which of the models is the best fit using the most appropriate functions and measures.

B.4.4 Logistic regression and classification

We want to be able to decide, on the basis of the brain volumetry data we are looking at, whether a new subject may have Alzheimer's disease.

To do so, we will build logistic regression and classification models to solve this problem.

- We need to split the dataset into a training set with 80% of the data, randomly selected, and the remaining 20% to validate the models.
- Create a suitable logistic regression model (especially one in which the predictors are significant) for the `CLASS` variable. How accurate is this model if we run it (using `predict()`) on the validation set?

Index