­­R упражнения материали 2019/2020

----------------------------------------------- Упражнение 1 -----------------------------------------------------

install.packages("UsingR") # Инсталиране на пакети

# Преди да инсталираме пакети, желателно е да проверим дали пакетът вече не е инсталиран.

# Това става с реда

installed.packages()

# Резултатът е матрица, която съдържа името на пакета, директорията на пакета, версията

# и т.н. Ако само се нуждаем от имената на инсталираните пакети, то най-добре е да изпозлваме

# командата по-долу

row.names(installed.packages())

# Връща ни вектор с имената. Векторът може да се присвои на променлива

library(UsingR) # Зареждане на пакета

# - Как можем да потърсим помощ в R?

# - С помощта на ?името\_на\_функцията и help("името\_на\_функцията"). Ако не се сещаме

# името на функцията, която ни трябва, то можем да напишем част от това, която ни трябва

?mean

help("sd")

??linear # Надяваме се да намерим някаква функция за линейна регресия. Намерихме я - stats::lm

# Начините за присвояване на обект към променлива са: "=", "<-" или "->". Последняит е

# напълно излишен но го има

x = 5

y <- 2\*x + 6

"Austria" -> z

x; y; z

# Основни структури от данни

# 1. Вектор/масив

# С комaндата "c(...)" се създава масив от елементи. В един векторът може да се

# съдържат елементи от различни тип - числа, стрингове и дори листа/обекти.

# Ако имаме наличие на лист или обект, то векторът се превръща в лист.

v1 <- c() # Създаване на правен вектор

v1 <- c(1, 4, 6., 4, 7, 12., -17) # Създаване на вектор, който съдържа числа

v1

# За да добавим нови елементи към вектора, най-лесно е да използваме отново командата "c(...)"

v2 <- c("a", "b", "c")

v2

v12 <- c(v1, v2) # Новият вектор съдържа елементите на двата вектора v1 и v2

v12

# Какъв е типа на елементите в новия вектор? Това се проверява с функцията "str(...)"

str(v1) # Числов вектор

str(v2) # Стрингов вектор

str(v12) # Автоматично е cast-нат е до стрингов вектор

# Добре, а какво би станало, ако използваме дати?

v3 <- as.Date(c("2015-01-01", "2016-07-08"))

# С функцията as.Date конвертираме стрингов вектор (или числов) във вектор с дати

?as.Date

v13 <- c(v1 ,v3)

v13

str(v13) # Числов вектор

# Забележете, че датите са integer числа, които броят дните от 1970-01-01 до посочената дата

# - А можем ли да кажем на R, че искаме непременно векторът да бъде в числов вид

# - Да и не само в числов. Можем да му кажем, че искаме да го cast-нем в стрингов или

# във вектор, съдържащ дати. Това, разбира се, може да доводе до загуба на информация

as.numeric(v12)

# Стойността "NA" в последните три елемента от вектора, показва, че имаме липса на

# информация. Това се дължи на факта, че се опитахме да представим стрингов вектор в

# числова форма.

as.character(v1) # Тук нямаме проблем, защото всеки елемент може да се представи в стрингова форма.

# Да пробваме да преобразуваме числов вектор във вектор с дати

as.Date(v1)

# Дава грешка. Защо?

# Защото не сме посочили начална дата, от която да тръгне броенето на дните. Началната

# дата се задава с параметъра "origin"

as.Date(v1, origin = "2011-01-01")

v1

> v1 <- as.Date(c("2015-01-01", "2016-07-08"))

> str(v1)

Date[1:2], format: "2015-01-01" "2016-07-08"

> v1 <- c(1, 4, 6., 4, 7, 12., -17)

> as.Date(v1)

Error in as.Date.numeric(v1) : 'origin' must be supplied

> as.Date(v1, origin = "2019-03-22")

[1] "2019-03-23" "2019-03-26" "2019-03-28" "2019-03-26" "2019-03-29" "2019-04-03"

[7] "2019-03-05"

# length(...) - взима дължината на вектор

length(v1)

# Взимане на елемент от вектор

v1[1] # Индексацията започва от 1

v1[c(1, length(v1))] # Нов вектор, който съдържа първия и последния елемент от вектора

v1[c(3, 3, 3, 3)] # Вектор, който има дължина 4 и стойностите му са третия елемент от v1

> v1 <- c(1, 4, 6., 4, 7, 12., -17)

> v1[c(3, 3, 3, 3)]

[1] 6 6 6 6

1:8 # Създаваме редица, която съдържа елементите от 1 до 8

8:1 # Създаваме редица, която съдържа елементите от 8 до 1

# Горните два реда са еквивалентни съответно на долните два

seq(from = 1, to = 8, by = 1)

> seq(1,8,1)

[1] 1 2 3 4 5 6 7 8

seq(from = 8, to = 1, by = -1)

# Искаме да вектор, който да съдържа 3, 6, 1, 2, 3, 4 елементи от вектора v1

v1[c(3, 6, 1:4)]

> v1 <- seq(1,8,1)

> v1[c(3, 6, 1:4)]

[1] 3 6 1 2 3 4

> v2

[1] 2 3 4 5 6

> v2[c(3, 6)]

[1] 4 NA

> v2[c(3, 5)]

[1] 4 6

> v2[c(3, 5,4:1)]

[1] 4 6 5 4 3 2

# В R можем да създадем подвектор на v1, като му кажем кои стойности НЕ ИСКАМЕ да присъстват

> v1[-c(3, 6)]

[1] 1 2 4 5 7 8

> v1[-(3:6)]

[1] 1 2 7 8

# Можем да добавяме/умножаваме с число даден вектор. Също така можем да събираме и

# умножаваме два вектора.

# За да не ни се налага да се чудим какви числа да измисляме всеки път, то най-добре е

# всичко до оставим в ръцете на "съдбата". Тоест да генерираме нашите числа на случаен принцип.

# Създаваме 3 вектора с генерирани псевдослучайни величини. За да получаваме винаги една и

# съща редица от числа, то трябва винаги да стартираме от една и съща начална позиция. За тази

# цел използваме командата set.seed(...).

# Искаме числата, които се генерират в трите вектора, да се падат с равни вероятности. Тоест

# P(1) = P(2) = ... = P(n-1) = P(n), където P(x) е вероятнсотта да се падне числото x.

# В упражненията ще учите подробно различните вероятностни разпределения, но за момента е

# достатъчно да знаем, че този вид разпределение се нарича "равномерно".

# На долния ред е показан пример на равномерно разпределение

hist(trunc(runif(10^3, 1, 5.9999)), col = "red", main = "Histogram",

xlab = "Pseudo random numbers")

A screenshot of a cell phone

Description automatically generated

#

> set.seed(1806)

> v4 <- trunc(runif(n = 20, min = 1, max = 40.99999))

>

> v4

[1] 36 26 16 6 18 30 23 37 4 2 25 24 7 13 29 10 13 5 13 33

> set.seed(2713)

> v5 <- round(runif(n = length(v4), min = 1, max = 40.99999))

>

> v5

[1] 12 15 9 1 27 11 12 31 17 7 23 4 7 29 19 7 1 9 13 20

> set.seed(189)

> v6 <- round(runif(n = length(v4) - 7, min = 1, max = 40.99999))

>

> v6

[1] 29 25 24 29 10 3 17 19 20 9 35 30 28

# runif - функция за генериране на псевдо случайни равномерно разпределени числа.

# Първият параметър е за броя на случайните числа. Вторият и третият показват обхвата на възможните числа.

# Препоръчвам да я разгледате, защото се използва при Монте Карло методите за оптимизации

v4 + 3 – добавя 3 на всеки елемент във вектора, не променя самия вектор, принти само резултата

> v4 + 3

[1] 39 29 19 9 21 33 26 40 7 5 28 27 10 16 32

[16] 13 16 8 16 36

2\*v4 – умножава всеки елемент

> 2\*v4

[1] 72 52 32 12 36 60 46 74 8 4 50 48 14 26 58

[16] 20 26 10 26 66

v4/7 + 11

> v4/7+11

[1] 16.14286 14.71429 13.28571 11.85714 13.57143

[6] 15.28571 14.28571 16.28571 11.57143 11.28571

[11] 14.57143 14.42857 12.00000 12.85714 15.14286

[16] 12.42857 12.85714 11.71429 12.85714 15.71429

v4\*v5 # Скаларно произведение

> v4\*v5

[1] 432 390 144 6 486 330 276 1147 68

[10] 14 575 96 49 377 551 70 13 45

[19] 169 660

v4\*v6

> v4\*v6

[1] 1044 650 384 174 180 90 391 703 80

[10] 18 875 720 196 377 725 240 377 50

[19] 39 561

Warning message:

In v4 \* v6 :

longer object length is not a multiple of shorter object length

# Дава предупреждение, защото дължината на втория вектор не е кратна на първия.

# Ето защо започва умножението отначало.

> v1 <- c(1,2,3)

> v2 <- c(2,3,4)

> v1\*v2

[1] 2 6 12

> v1 <- c(1,2,3)

> v2 <- c(2,3)

> v1\*v2

[1] 2 6 6 - > започва отново да умножава от 2 на 2рия вектор

Warning message:

In v1 \* v2 :

longer object length is not a multiple of shorter object length

# - А друго какво мога да правя? Мога ли да сменям числа във вектора

# - Може, разбира се.

v4.prime <- v4

v4[c(1, 2, 3)] <- c(3, 2, 1)

v4 <- v4.prime

v4[c(1, 2, 3, 4)] <- c(100, -100)

> v4

[1] 36 26 16 6 18 30 23 37 4 2 25 24 7 13 29 10 13 5 13 33

> v4[c(1, 2, 3, 4)] <- c(100, -100)

> v4

[1] 100 -100 100 -100 18 30 23 37 4 2 25 24 7 13 29

[16] 10 13 5 13 33

# -------

# Функции

# име\_на\_функцията <- function(параметри) {

#

# }

func1 <- function(a, b, c) {

return(a + b + c)

}

func1(1, 2, 3)

# Функциите в R могат да имат стойности по подразбиране

func2 <- function(a, b = 0, c = 0) {

return(a + b + c)

}

func2(1)

func2(1, 2)

func2(1, c = 3)

# Функциите в R могат да приемат параметрите си в различен ред

func2(a = 1, c = 3, b = 2)

func2(b = 2, c = 3)

# Връща грешка, защото не сме задали стойност по подразбиране на параметъра "a" във функцията

# Това лесно може да се избегне с едно условие във функцията

func3 <- function(a, b = 0, c = 0) {

if(missing(a)) {

print("You don't enter value for \"a\". The function generate normal distributed value")

a <- rnorm(n = 1)

}

return(a + b + c)

}

# Командата "missing" проверява дали имаме стойност за параметъра "a"

# С "print()" извеждаме съобщение

# С "rnorm(n = 1)" генерираме една (n = 1) нормално разпределена случайна величина

# с очакване 0 и стандартно отклонение 1

func3(a = 1, b = 2, c = 3)

func3(b = 2, c = 3)

func4 <- function(a, b = 2, c = 3) {

if(missing(a)) {a <- NA}

Obj <- NULL

Obj$number1 <- a

Obj$number2 <- b

Obj$number3 <- c

Obj # return(Obj)

}

func4(1, 2, 3)

# if-else условия

# if() {

#

# } else if() {

#

# } else {

#

# }

# Цикли

# for(...) {}

# Цикълът for представлява foreach итерация. Конструкцията е проста -

# променлива %in% вектор

v7 <- c("a", "b", "c", "d", "e")

for(i in v7) {print(i)}

for(i in 1:length(v7)) {print(v7[i])}

for(i in length(7):1) {print(v7[i])}

for(i in 1:length(v7)) {

if(i == 4) {

print("----MISS")

next

}

print(v7[i])

}

# С командата "next" пропускаме итерация

counter <- 0

while(counter < 5) {

counter <- counter + 1

print(counter)

}

# do - while

counter <- 0

repeat {

counter <- counter + 1

print(counter)

if(counter > 5) {

break

}

}

# С командата "break" излизаме от най-близкия цикъл

# Циклите в R са бавни. Ето защо е подходящо в някои случаи да използвате методите

# apply, lapply, sapply, vapply и tapply. Тези методи ще ги представим след малко.

# -------

# 2. Матрица

# С функцията "matrix" в R създаваме матрица от предварително зададено множество от стойности - вектор или лист със стойности. Освен множеството с елементи, трябва да

посочим и формата на матрицата - брой редове и колони. Стойностите на матрицата се

пълнят по колони. За да напълним матрицата със стойности по редове, трябва да използваме параметъра byrow = TRUE

l1 <- lapply(1:12, function(x) {x}) # лист

> l1 <- lapply(1:12, function(x) {x+8})

> l1

[[1]]

[1] 9

[[2]]

[1] 10

[[3]]

[1] 11

[[4]]

[1] 12

[[5]]

[1] 13

[[6]]

[1] 14

[[7]]

[1] 15

[[8]]

[1] 16

[[9]]

[1] 17

[[10]]

[1] 18

[[11]]

[1] 19

[[12]]

[1] 20

M1 <- matrix(data = l1, nrow = 4, ncol = 3)

> M1 <- matrix(data = l1, nrow = 4, ncol = 3)

> M1

[,1] [,2] [,3]

[1,] 9 13 17

[2,] 10 14 18

[3,] 11 15 19

[4,] 12 16 20

M1 <- matrix(data = 1:12, nrow = 4, ncol = 3)

> M1 <- matrix(data = 1:12, nrow = 4, ncol = 3)

> M1

[,1] [,2] [,3]

[1,] 1 5 9

[2,] 2 6 10

[3,] 3 7 11

[4,] 4 8 12

# Горните два реда са еквивалентни

M2 <- matrix(data = 1:12, nrow = 4, ncol = 3, byrow = T)

> M2 <- matrix(data = 1:12, nrow = 4, ncol = 3, byrow = T)

> M2

[,1] [,2] [,3]

[1,] 1 2 3

[2,] 4 5 6

[3,] 7 8 9

[4,] 10 11 12

M1

M2

# Взимане на елемент, ред, колона и подматрица от матрица

M3 <- matrix(data = c(1:28), nrow = 7, ncol = 4, byrow = TRUE)

> M3 <- matrix(data = c(1:28), nrow = 7, ncol = 4, byrow = TRUE)

> M3

[,1] [,2] [,3] [,4]

[1,] 1 2 3 4

[2,] 5 6 7 8

[3,] 9 10 11 12

[4,] 13 14 15 16

[5,] 17 18 19 20

[6,] 21 22 23 24

[7,] 25 26 27 28

M3[2, ] # Взимане на ред

> M3[2, ]

[1] 5 6 7 8

M3[, 3] # Взимане на колона

> M3[, 3]

[1] 3 7 11 15 19 23 27

M3[1, 3] # Взимане на елемент

> M3[1, 3]

[1] 3

M3[c(1, 2), 3] # Взимаме 1 и 2 елемент от 3 колона

> M3[c(1, 2), 3]

[1] 3 7

M3[c(1, 2), c(3, 4)] # Взимаме подматрица

> M3[c(1, 2), c(3, 4)]

[,1] [,2]

[1,] 3 4

[2,] 7 8

# Операции с матрици

M3 + 4 # Добавяме число към матрицата

> M3+4

[,1] [,2] [,3] [,4]

[1,] 5 6 7 8

[2,] 9 10 11 12

[3,] 13 14 15 16

[4,] 17 18 19 20

[5,] 21 22 23 24

[6,] 25 26 27 28

[7,] 29 30 31 32

M3 + 5\*c(1:7) # Добавяме по вектор към всеки от четирите колони

- добавя 5 на 1вата колона, 10 на втората, 15 на третата и т.н до 7мата колона

> M3 + 5\*c(1:7) #

[,1] [,2] [,3] [,4]

[1,] 6 7 8 9

[2,] 15 16 17 18

[3,] 24 25 26 27

[4,] 33 34 35 36

[5,] 42 43 44 45

[6,] 51 52 53 54

[7,] 60 61 62 63

M3 + 5\*c(1:4) # Добавяме ред вектор всеки от седемте реда на матрицата

> M3 + 5\*c(1:4)

[,1] [,2] [,3] [,4]

[1,] 6 22 18 14

[2,] 15 11 27 23

[3,] 24 20 16 32

[4,] 33 29 25 21

[5,] 22 38 34 30

[6,] 31 27 43 39

[7,] 40 36 32 48

> M3 + 5\*c(1:2) # R автоматично удвоява вектора до дължина 4 и добавяме вектора към всеки ред

[,1] [,2] [,3] [,4]

[1,] 6 12 8 14

[2,] 15 11 17 13

[3,] 14 20 16 22

[4,] 23 19 25 21

[5,] 22 28 24 30

[6,] 31 27 33 29

[7,] 30 36 32 38

# Аналогично е и при умножение на матрица с вектор

> M4 <- matrix(1:8, nrow = 4, ncol = 2)

> M34 <- M3 %\*% M4 # Стандартно умножение на две матрици

> M34

[,1] [,2]

[1,] 30 70

[2,] 70 174

[3,] 110 278

[4,] 150 382

[5,] 190 486

[6,] 230 590

[7,] 270 694

> # M(7x4) \* M(4x2) = M(7x2)

> dim(M34) # Връща размера на матрицата

[1] 7 2

nrow(M34) # Връща броя на редовете

ncol(M34) # Връща броя на колоните

# Функцията apply се прилага върху матрици, data.frame и други производни структури.

# Първият параметър е множеството от данни, вторият показва по редове (1) или колони (2) искаме да направим трансформациите, а третият е самата функция

> M3

[,1] [,2] [,3] [,4]

[1,] 1 2 3 4

[2,] 5 6 7 8

[3,] 9 10 11 12

[4,] 13 14 15 16

[5,] 17 18 19 20

[6,] 21 22 23 24

[7,] 25 26 27 28

> apply(M3, 2, function(x, a) {sum(x)}, a)

[1] 91 98 105 112

> apply(M3, 1, function(x) {sum(x)})

[1] 10 26 42 58 74 90 106

APPLY - > MARGIN(1) за редове, 2 за колони

**Usage**

apply(X, MARGIN, FUN, ...)

**Arguments**

|  |  |
| --- | --- |
| X | an array, including a matrix. |
| MARGIN | a vector giving the subscripts which the function will be applied over. E.g., for a matrix 1 indicates rows, 2 indicates columns, c(1, 2) indicates rows and columns. Where X has named dimnames, it can be a character vector selecting dimension names. |
| FUN | the function to be applied: see ‘Details’. In the case of functions like +, %\*%, etc., the function name must be backquoted or quoted. |

----------------------------------------------- Упражнение 2 -----------------------------------------------------

# Data frame

# Друга структура за съхраняване на данни в R е data frame. Разликата с матрицата е,

че елементите в матрицата е вектор от вектори и всички елементи са от един тип, докато

data frame-a е лист от вектори и елементите на data frame-а са от един тип, само в рамките на отделния вектор.

car\_brand <- c('volkswagen', 'chevrolet', 'peugot', 'volvo', 'porsche', 'chevrolet', 'mitsubishi', 'mazda',

'toyota', 'subaru', 'isuzu', 'volvo', 'mercury', 'honda', 'peugot', 'porsche', 'dodge',

'volkswagen', 'bmw', 'mercedes-benz', 'toyota', 'isuzu', 'volkswagen', 'mazda', 'plymouth',

'honda', 'mitsubishi', 'plymouth', 'volkswagen', 'volkswagen', 'honda', 'toyota', 'nissan',

'nissan', 'mitsubishi', 'volvo', 'toyota', 'jaguar', 'volkswagen', 'dodge')

fuel\_type <- c('gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas',

'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'diesel', 'gas', 'gas', 'diesel', 'gas', 'gas', 'gas',

'gas', 'diesel', 'gas', 'gas', 'gas', 'gas', 'diesel', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas')

horsepower <- c(111, 154, 102, 115, 110, 140, 160, 101, 121, 182, 48, 70, 68, 102, 88, 145, 58, 76, 60, 86,

101, 100, 78, 70, 90, 176, 262, 68, 101, 135, 84, 64, 120, 72, 123, 155, 184, 175, 68, 102)

turbo\_aspiration <- c(FALSE, TRUE, FALSE, FALSE, TRUE, FALSE, FALSE, TRUE, TRUE, FALSE, FALSE, FALSE, TRUE, FALSE,

FALSE, FALSE, FALSE, FALSE, TRUE, FALSE, FALSE, TRUE, FALSE, FALSE, TRUE, FALSE, FALSE,

FALSE, TRUE, TRUE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, TRUE, TRUE, FALSE, TRUE)

date <- c('2019-01-09', '2019-03-01', '2019-01-16', '2018-12-13', '2019-02-16', '2019-02-15', '2019-02-19',

'2019-02-27', '2019-01-19', '2019-01-22', '2019-03-05', '2018-12-19', '2019-01-13', '2019-02-04',

'2019-01-30', '2019-01-12', '2019-02-28', '2018-12-15', '2018-12-31', '2018-12-20', '2019-01-21',

'2019-01-24', '2019-02-05', '2019-02-17', '2018-12-19', '2018-12-26', '2019-02-26', '2018-12-26',

'2018-12-19', '2018-12-17', '2019-03-04', '2019-03-07', '2019-01-26', '2019-02-06', '2018-12-20',

'2018-12-13', '2018-12-15', '2018-12-24', '2018-12-30', '2019-03-03')

cars <- data.frame(

date = as.Date(date), # Дати

car\_brand, fuel\_type, # Стрингове

turbo\_aspiration, # Булева променлива

horsepower # Числа

)

rm(list = c("car\_brand", "fuel\_type", "horsepower", "turbo\_aspiration", "date"))

# функцията rm() премахва обектите, които описани в масива, присвоен на параметъра "list"

### Usage

remove(..., list = character(), pos = -1,

envir = as.environment(pos), inherits = FALSE)

rm (..., list = character(), pos = -1,

envir = as.environment(pos), inherits = FALSE)

str(cars)

# Както виждаме, data frame-а може да съдържа данни от различен тип - дата, стринг,

булев или числов.

# В R можем да вземем първите и последните няколко реда с помощта на функциите head() и tail()

head(x = cars)

head(x = cars, n = 3) # n - колко наблюдения искаме да видим.

По подразбиране, стойността на n е 6

tail(x = cars, n = 3)

# Създаване на извадки

# Искаме да създадем вектор, който съдържа стойностите TRUE и FALSE

> N <- 30

> set.seed(1234)

> sample(x = c(T, F), size = N, replace = TRUE, prob = c(0.9, 0.1))

[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

[14] FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

[27] TRUE FALSE TRUE TRUE

### Description

sample takes a sample of the specified size from the elements of x using either with or without replacement.

### Usage

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

> set.seed(4321)

> round(runif(n = N, min = 1, max = 10))

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

[27] 9 3 2 2

# Генериране на равномерно разпределена величина с N на брой стойности в интервала 1, 10

# sample(x, size, replace, prob) - на случаен принцип се избират N на брой елементи от предварително зададен вектор

# x - вектор, от който се избират елементите,

# size - размерът на желаната извадка (в нашия случай 30),

# replace - опцията дали да се преизбират наново елементи. Ако size > x, то задължително трябва replace = TRUE

# prob - вектор с вероятностите за избиране на случаен елемент. Трябва дължината на prob да бъде равна на дължината на x и сборът да бъде равен на 1

# round(x, digits = 0) - закръгля числата до digits цифри след десетичната запетая.

# Обединяване на матрици и data frame-ове по редове и колони

# Ще покажем случая с матрици. С data frame е абсолютно същото.

# Създаваме две матрици

M1 <- matrix(data = 1:20, nrow = 10, byrow = T)

M2 <- matrix(data = 21:30, nrow = 2, byrow = T)

M1; M2

# Функциите за обединяване на две матрици/data frame-ове са:

# - rbind - обединение по редове - долепя втората матрица под първата

# - cbind - обединение по колони - долепя втората матрица след първата

rbind(M1, M2)

# Връща грешка, защото, когато обединяваме две матрици по редове, то размерът на колоните трябва да бъде еднакъв. Същото се отнася и при обединяване по колони -

# искаме редовете да са еднавни на брой.

# Проверка може да се извърши с nrow(), ncol() и dim()

dim(M1)

dim(M2)

# Нека да транспонираме втората матрица (транспонирането е операция, за която редовете се превръщат в колони). Функцията за транспониране в R е t()

M2.t <- t(M2)

dim(M2.t)

rbind(M1, M2.t)

# Пример за обединение по колони

M3 <- matrix(data = 21:40, nrow = 10, byrow = T)

cbind(M1, M3)

# Функциите apply, lapply, sapply и т.н. могат да се прилагат и върху data frame-ове

# Взимане на ред, колона и елемент от data frame

# Всеки от начините, които споменахме при взиамне на елементи от матрица, са приложими и за data frame, като отделно имаме и допълнителни.

# dataframe[избор на редове, избор на колони]

cars[1, ] # Първи ред и всички колони (индексът за колоните е празен)

cars[, 2] # Втора колона и всички редове (индексът за редовете е празен)

cars[1:nrow(cars), 2] # еквивалентен на горния ред

cars[5, c(2, 3)] # Пети ред, втора и трета колона

cars[5, 4] # 5 ред и 4-та колона

# С функцията row.names можем както да взимаме имената на редовете, така можем и да променяме вече съществуващи имена

row.names(cars) <- paste("car", 1:nrow(cars), sep = "\_")

> row.names(cars)

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

[16] "16" "17" "18" "19" "20" "21" "22" "23" "24" "25" "26" "27" "28" "29" "30"

[31] "31" "32" "33" "34" "35" "36" "37" "38" "39" "40"

> row.names(cars) <- paste("car", 1:nrow(cars), sep = "\_")

> row.names(cars)

[1] "car\_1" "car\_2" "car\_3" "car\_4" "car\_5" "car\_6" "car\_7" "car\_8"

[9] "car\_9" "car\_10" "car\_11" "car\_12" "car\_13" "car\_14" "car\_15" "car\_16"

[17] "car\_17" "car\_18" "car\_19" "car\_20" "car\_21" "car\_22" "car\_23" "car\_24"

[25] "car\_25" "car\_26" "car\_27" "car\_28" "car\_29" "car\_30" "car\_31" "car\_32"

[33] "car\_33" "car\_34" "car\_35" "car\_36" "car\_37" "car\_38" "car\_39" "car\_40"

ПРОМЕНЯ ГО В DATAFRAME-A

# С paste() обединяваме вектори със стрингове (n-тия елемент на единия вектор с n-тия елемент на друг вектор)

# Имената трябва да са уникални

# При data frame-а можем да вземем елементи и посредством техните имена

# - по колони

cars$car\_brand # С "$" взимаме елементи от листа. Data frame-а е ЛИСТ от ВЕКТОРИ => взимаме вектора

cars[, "car\_brand"]

# Еквивалентни са на cars[, 2]

# - по редове

cars["car\_1", ]

# Еквивалентхно е на cars[1, ]

# - по редове и колони

cars["car\_1", c("car\_brand", "turbo\_aspiration")]

# Следващата стъпка е филтрирането на данни и взимането на подгрупи по зададен признак

# & - логическо "и", | - логическо "или", == - проверка за равенстов

# which() - Връща вектор с индекстите на елементите, за които булевото условие връща TRUE.

# Тоест взимаме индексите на елементите, които ни вълнуват по някакъв начин.

# Нека да проверим кои от всички 40 модела коли имат turbo aspiration.

> which(cars$turbo\_aspiration)

[1] 2 5 8 9 13 19 22 25 29 30 37 38 40

cars[which(cars$turbo\_aspiration), ] # Извеждаме самите наблюдения

> cars[which(cars$turbo\_aspiration), ] # Извеждаме самите наблюдения

date car\_brand fuel\_type turbo\_aspiration horsepower

car\_2 2019-03-01 chevrolet gas TRUE 154

car\_5 2019-02-16 porsche gas TRUE 110

car\_8 2019-02-27 mazda gas TRUE 101

car\_9 2019-01-19 toyota gas TRUE 121

car\_13 2019-01-13 mercury gas TRUE 68

car\_19 2018-12-31 bmw gas TRUE 60

car\_22 2019-01-24 isuzu gas TRUE 100

car\_25 2018-12-19 plymouth gas TRUE 90

car\_29 2018-12-19 volkswagen diesel TRUE 101

car\_30 2018-12-17 volkswagen gas TRUE 135

car\_37 2018-12-15 toyota gas TRUE 184

car\_38 2018-12-24 jaguar gas TRUE 175

car\_40 2019-03-03 dodge gas TRUE 102

# cars$turbo\_aspiration е булева променлива => which(cars$turbo\_aspiration) ще ни върне индексите на наблюденията, които изпълняват това условие => имайки индексите с cars[индексите, ], извеждаме и самите резултати

cars[which(cars$turbo\_aspiration), "car\_brand"] # Имената на момичетата, които са приели

> cars[which(cars$turbo\_aspiration), "car\_brand"]

[1] chevrolet porsche mazda toyota mercury bmw isuzu

[8] plymouth volkswagen volkswagen toyota jaguar dodge

18 Levels: bmw chevrolet dodge honda isuzu jaguar mazda mercedes-benz ... volvo

cars[which(cars$horsepower >= 102), ] # коли, които имат над 102 конски сили

> cars[which(cars$horsepower >= 102), ]

date car\_brand fuel\_type turbo\_aspiration horsepower

car\_1 2019-01-09 volkswagen gas FALSE 111

car\_2 2019-03-01 chevrolet gas TRUE 154

car\_3 2019-01-16 peugot gas FALSE 102

car\_4 2018-12-13 volvo gas FALSE 115

car\_5 2019-02-16 porsche gas TRUE 110

car\_6 2019-02-15 chevrolet gas FALSE 140

car\_7 2019-02-19 mitsubishi gas FALSE 160

car\_9 2019-01-19 toyota gas TRUE 121

car\_10 2019-01-22 subaru gas FALSE 182

car\_14 2019-02-04 honda gas FALSE 102

car\_16 2019-01-12 porsche gas FALSE 145

car\_26 2018-12-26 honda gas FALSE 176

car\_27 2019-02-26 mitsubishi gas FALSE 262

car\_30 2018-12-17 volkswagen gas TRUE 135

car\_33 2019-01-26 nissan gas FALSE 120

car\_35 2018-12-20 mitsubishi gas FALSE 123

car\_36 2018-12-13 volvo gas FALSE 155

car\_37 2018-12-15 toyota gas TRUE 184

car\_38 2018-12-24 jaguar gas TRUE 175

car\_40 2019-03-03 dodge gas TRUE 102

# Векторът cars$horsepower съдържа стойностите 111, 154, 102, 115, ..., 184, 175,102.

# С условието cars$horsepower >= 102 ще върне T, T, T, T, ..., F, T, T, T, T, F, T =>

# which(cars$horsepower >= 102) ще върне вектор с индекси 1, 2, 3, 4, 5, 6, ..., 36, 37 38, 40 =>

# cars[which(cars$horsepower >= 102), ] ще върне информацията за колите, които имат конски сили поне 102

cars[which(cars$car\_brand %in% c("chevrolet", "porsche")), ]

# първи/а масив/вектор/лист/стойност %in% втори/а масив/вектор/лист/стойност -

# проверяваме кои елементи от първата структура присъстват във втората.

> cars[which(cars$car\_brand %in% c("chevrolet", "porsche")), ]

date car\_brand fuel\_type turbo\_aspiration horsepower

car\_2 2019-03-01 chevrolet gas TRUE 154

car\_5 2019-02-16 porsche gas TRUE 110

car\_6 2019-02-15 chevrolet gas FALSE 140

car\_16 2019-01-12 porsche gas FALSE 145

# Как да вземем моделите на колите, които имат turbo aspiration и са продадени

# през последната седмица? Дали са има такива?

attach(cars)

car\_brand

detach(cars)

car\_brand

# С това приключваме с въведението за трите типа структури данни в R.

# О П И С А Т Е Л Н А С Т А Т И С Т И К А

# 1. Променливи

# Най-простата дефиниция за променливи е характеристика, която се варира от един човек/обект до друг. Променливите са категоризирани в два типа - количествени или числови (представят се с числа)и категорийни или качествени (нечислово изобразяване).

# Категорийните променливи се разбиват на два подтипа - номинални (без подредба) и ординални

# (с подредба). Примери за стойности на НОМИНАЛНИ променливи са: видове заведения, различни

# държави и организции, видове алкохол, раси и т.н. Примери за стойности на ОРДИНАРНИ

# променливи са: степени на образование/квалификация, нива на социалните общества и т.н.

# Количествените се разбиват на два подтипа - непрекъснати (стойностите могат да варират

# в даден числов интервал) и дискретни (възможните стойности могат да бъдат изброени).

# Примери за стойности на НЕПРЕКЪСНАТИ променливи са: възраст, различни съотношения, тегло

# и т.н. Примери за стойности на ДИСКРЕТНИ променливи са: брой катастрофи на шофьори, брой на

# деца и т.н.

# 2. Разпределения

# Разпределение - това е таблица, графика или формула, съдържаща стойностите на наблюденията

# и колко често се случват. Разпределенията се характеризират с форма, модалност (modality),

# локация, размах, симетричност/асиметричност и ексцес.

# На графиката по-долу са представени няколко от формите на плътността на разпределенията

N <- 2\*10^4

distributions <- c("Камбановидна", "J-", "Правоъгълна", "Дясно асиметрична", "Двумодална")

par(mfrow = c(2, 3))

for(distr in distributions) {

distribution\_values <- switch(distr, "Камбановидна" = rnorm(N), "J-" = rexp(N),

"Правоъгълна" = round(runif(N), 2),

"Дясно асиметрична" = rgamma(N, shape = 5),

"Двумодална" = c(rnorm(N, mean = -2, sd = 2), rnorm(N, mean = 3, sd = 1)))

distr\_density <- density(distribution\_values)

hist(distribution\_values, col = "pink", prob = T, main = paste(distr, "форма на разпределението"),

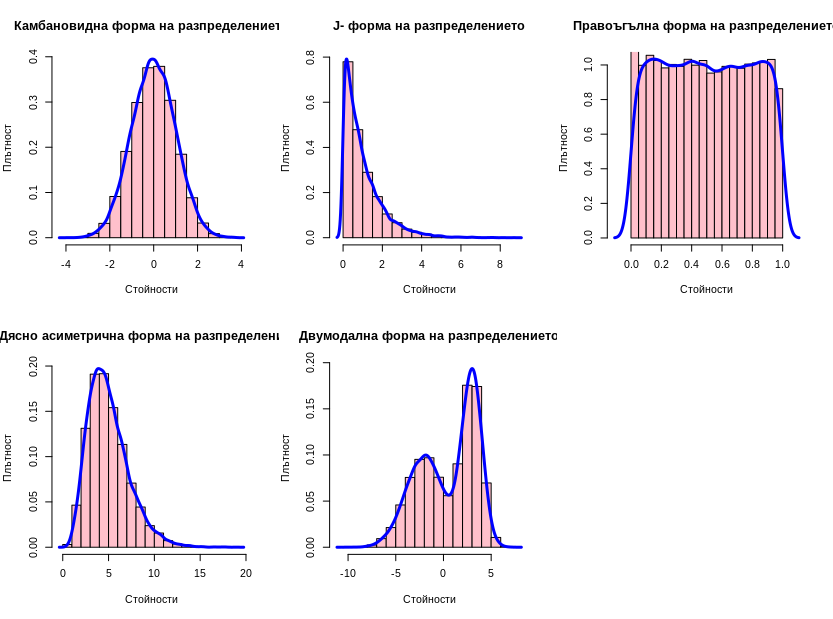
ylab = "Плътност", xlab = "Стойности", ylim = range(distr\_density$y),

xlim = range(distr\_density$x))

lines(distr\_density, lwd = 3, col = "blue")

}

par(mfrow = c(1, 1))



# За видовете разпределения, симулациите, квантили и плътност ще говорим по-нататъка.

country = c('Switzerland', 'Norway', 'Iceland', 'Denmark', 'Finland', 'Netherlands', 'Germany',

'Ireland', 'Sweden', 'United Kingdom', 'Austria', 'Belgium', 'France', 'Italy',

'Spain', 'Malta', 'Slovenia', 'Estonia', 'Czech Republic', 'Portugal', 'Slovakia',

'Poland', 'Croatia', 'Lithuania', 'Greece', 'Latvia', 'Hungary', 'Romania',

'Bulgaria', 'Russia', 'Bosnia And Herzegovina', 'Serbia', 'Belarus', 'Albania',

'Macedonia', 'Ukraine', 'Moldova')

avg\_sal\_in\_USD = c(4959.48, 3416.1, 3136.95, 2975.8, 2583.43, 2575.2, 2538.92, 2521.51, 2494.39,

2394.78, 2219.23, 2162.23, 2140.64, 1675.32, 1480.64, 1278.04, 1180.66,

1158.76, 1069.03, 925.84, 894.2, 884.81, 852.23, 804.68, 795.34, 771.91,

708.91, 649.68, 611.41, 558.01, 481.17, 412.22, 409.62, 348.56, 346.15,

293.26, 284.94)

salaryDF <- data.frame(country, avg\_sal\_in\_USD)

# Проучването е правено за 2018-та година за европейските страни

Country <- c("Austria", "Belgium", "Croatia", "Czechia", "Denmark", "Estonia", "Finland",

"France", "Germany", "Greece", "Hungary", "Ireland", "Italy", "Latvia",

"Lithuania", "Luxembourg", "Netherlands", "Norway", "Poland", "Portugal",

"Romania", "Slovakia", "Slovenia", "Spain", "Sweden", "Switzerland", "UK")

Car <- c("Volkswagen", "Volkswagen", "Skoda", "Skoda", "Peugeot", "Toyota", "Nissan",

"Renault", "Volkswagen", "Toyota", "Suzuki", "Hyundai", "Fiat", "Volkswagen",

"Fiat", "Volkswagen", "Volkswagen", "Nissan", "Skoda", "Renault", "Dacia",

"Skoda", "Renault", "Seat", "Volvo", "Skoda", "Ford")

carsDF <- data.frame(Country, Car)

> carsDF

Country Car

1 Austria Volkswagen

2 Belgium Volkswagen

3 Croatia Skoda

4 Czechia Skoda

5 Denmark Peugeot

6 Estonia Toyota

7 Finland Nissan

8 France Renault

9 Germany Volkswagen

10 Greece Toyota

11 Hungary Suzuki

12 Ireland Hyundai

13 Italy Fiat

14 Latvia Volkswagen

15 Lithuania Fiat

16 Luxembourg Volkswagen

17 Netherlands Volkswagen

18 Norway Nissan

19 Poland Skoda

20 Portugal Renault

21 Romania Dacia

22 Slovakia Skoda

23 Slovenia Renault

24 Spain Seat

25 Sweden Volvo

26 Switzerland Skoda

27 UK Ford

# А Н А Л И З Н А Е Д Н О М Е Р Н А П Р О М Е Н Л И В А

# (UNIVARIATE ANALYSIS)

# 3. Категорийни променливи

# Категорийните проенливи като цяло носят по-малко информация отколкото числовите променливи.

# Често обаче, те носят по-голяма стабилност за прогнозните моделите отколкото числовите.

# Освен графично представяне на данните (ще го разгледаме по-късно), най-добре честотата се

# вижда посредством таблици. Функцията, която ни трябва е table()

head(carsDF)

tt <- table(carsDF$Car)

tt # Взимаме честотното разпределение

> tt <- table(carsDF$Car)

> tt

Dacia Fiat Ford Hyundai Nissan Peugeot Renault

1 2 1 1 2 1 3

Seat Skoda Suzuki Toyota Volkswagen Volvo

1 5 1 2 6 1

round(prop.table(tt)\*100, 2) # Процентното разпределение

# С prop.table() взимаме процентите от вече направена таблица

> round(prop.table(tt)\*100, 2)

Dacia Fiat Ford Hyundai Nissan Peugeot Renault

3.70 7.41 3.70 3.70 7.41 3.70 11.11

Seat Skoda Suzuki Toyota Volkswagen Volvo

3.70 18.52 3.70 7.41 22.22 3.70

# Volkswagen държи най-голям пазарен дял по страни, следван от Skoda

# 4. Числови променливи

# 4.1. Оценка на центъра (локацията) на разпределение

# 4.1.1. Средна стойност (Очакване)

meanFunction <- function(x) {

sum(x)/length(x)

}

meanFunction(salaryDF$avg\_sal\_in\_USD)

mean(salaryDF$avg\_sal\_in\_USD)

# 4.1.2. Медиана

# Подреждаме данните във вариационен ред. Взимаме средната стойност (при нечетен брой)

# или средната стойност на средните два елемента (при четен брой елементи).

medianFunction <- function(x) {

x\_sorted <- sort(x)

nn <- length(x\_sorted)

if(nn %% 2 == 0) {

return(mean(x\_sorted[nn/2 + c(0, 1)]))

} else {

return(x\_sorted[round(nn/2 + 0.25)])

}

}

# %% == mod

# %/% == div

medianFunction(salaryDF$avg\_sal\_in\_USD)

median(salaryDF$avg\_sal\_in\_USD)

medianFunction(1:10)

median(1:10)

# 4.1.3. Мода - най-често срещаната стойност

modeFunction <- function(x) {

tt <- table(x)

return(names(tt)[tt == max(tt)])

}

> modeFunction(carsDF$Car)

[1] "Volkswagen"

# Модата може да има повече от една стойност

# table() - честотоното разпределение на променлива (коя стойност колко пъти се с)

# which.max() - връща индекса на първото число, което приема МАКСИМАЛНА стойност

# which.min() - връща индекса на първото число, което приема МИНИМАЛНА стойност

# names() - връща имената на стойностите

modeFunction(round(salaryDF$avg\_sal\_in\_USD/100))

summary(salaryDF$avg\_sal\_in\_USD) # Описателна статистика за центъра на разпределението

quantile(salaryDF$avg\_sal\_in\_USD, prob = seq(0.1, 0.9, by = 0.1))

# квантили

# Задача 1 - Ежедневни инциденти с мотоциклети

# Шотландският изпълнителен директор в отдел "Аналитични услуги" на Транспортна

# статистика събира данни за произшествията с мотоциклети. В таблицата по-долу са

# представени, броят на инцидентите с мотоциклети в Шотландия по пътища с ограничение

# до 30 и над 30 мили в час, случили се по дни от седмицата.

Day <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")

Built\_up <- c(88, 100, 76, 98, 103, 85, 69)

Non\_built\_up <- c(70, 58, 59, 53, 56, 94, 102)

motorcycleAccidentsDF <- data.frame(Day, Built\_up, Non\_built\_up)

# а. Каква е средната стойност и медианата на броя на произшествията за двата вида пътища

# б. Каква е формата на разпределението по различните пътища. Интересува ни само модалността

# Задача 2 - Инвестиции в акциите "LMT" и "FB"

Date <- c('2015-09', '2015-10', '2015-11', '2015-12', '2016-01', '2016-02', '2016-03', '2016-04',

'2016-05', '2016-06', '2016-07', '2016-08', '2016-09', '2016-10', '2016-11', '2016-12',

'2017-01', '2017-02', '2017-03', '2017-04', '2017-05', '2017-06', '2017-07', '2017-08',

'2017-09', '2017-10', '2017-11', '2017-12', '2018-01', '2018-02', '2018-03', '2018-04',

'2018-05', '2018-06', '2018-07', '2018-08', '2018-09')

FB <- c(89.9, 101.97, 104.24,

104.66, 112.21, 106.92, 114.1, 117.58, 118.81, 114.28, 123.94, 126.12, 128.27, 130.99,

118.42, 115.05, 130.32, 135.54, 142.05, 150.25, 151.46, 150.98, 169.25, 171.97, 170.87,

180.06, 177.18, 176.46, 186.89, 178.32, 159.79, 172, 191.78, 194.32, 172.58, 175.73,

164.46)

LMT <- c(207.31, 219.83, 219.16, 217.15, 211, 215.79, 221.5, 232.38, 236.23, 248.17, 252.73,

242.97, 239.72, 246.38, 265.25, 249.94, 251.33, 266.58, 267.6, 269.45, 281.13, 277.61,

292.13, 305.39, 310.29, 308.16, 319.12, 321.05, 354.85, 352.44, 337.93, 320.84, 314.54,

295.43, 326.1, 320.41, 345.96)

stocksDF <- data.frame(Date, LMT, FB)

# Анализът трябва да се извърши върху възвръщаемостите на цените, получени по diff(log(x)),

# където "x" е цената на актива

# а. Какви са средните стойности и медианите. Според вас как се интерпретират тези числа

# б. Можете ли да кажете в кой актив бихте инвестирали при наличието само на тази информация

----------------------------------------------- Упражнение 3 -----------------------------------------------------

# Вариация

team1 <- c(72, 73, 76, 76, 78)

team2 <- c(67, 72, 76, 76, 84)

basketball\_teams <- data.frame(team1, team2)

# 3. Оценка на вариацията на разпределение

# Преди да започнем с изследването на разсейването, първо ще видим какво е очакването

colMeans(basketball\_teams) # Взимаме средните на стойностите по колони

apply(basketball\_teams, 2, mean) # Еквивалентно на горния ред

# Като цело се избягват заключенията само на база средните стойности или други оценки за

# локацията на разпределенията (ще наблегнем на това при оценките на хипотезите).

# И при двета отбора средните стойности са равни - 75. това не означава, че двата отбора са близки относно разпределението на височината им. В частност, височината при играчите на втория отбор варира много повече от тази на първия. В следващата секция ще разгледаме как можем да оженим вариацията.

# 3.1. Обхват (Range) - максималната стойност - минималната стойност

rangeFunction <- function(x) {

max(x) - min(x)

}

> rangeFunction(basketball\_teams$team1)

[1] 6

> rangeFunction(basketball\_teams$team2)

[1] 17

# както можем да видим от резултатите, при първия отбор имаме разлика от 6 инча, докато при втория - 17

# 3.2. Вариация (дисперсия) и стандартно отклонение

# За разлика от обхвата, стандартното отклонение взема под внимамнеи всички наблюдения.

# Стандартното отклонение е оценка на вариацията, която показва колко далече са наблюденията от очакването

# Стандартното отклонение е предпочитана оценка за вариацията, когато среднатото се използва за оценка

# на локацията (центъра) на разпределението.

# Вариацията (дисперсията) се изчислява по формулата по-долу:

variationFunction <- function(x) {

x\_mean <- sum(x)/length(x)

x\_minus\_xMean <- x - x\_mean

x\_minus\_xMean\_2 <- x\_minus\_xMean^2

sum(x\_minus\_xMean\_2) / (length(x) - 1)

}

# Вариацията има функция в базовия пакет на R

variationFunction(basketball\_teams$team1)

var(basketball\_teams$team1)

var(basketball\_teams$team2)

# При тестването на хипотези и при определянето на доверителните интервали се използва

# стандардартното отклонение. Стандартното отклонение е производно на вариацяита и представлява

# корен квадратен от дисперсията.

sqrt(variationFunction(basketball\_teams$team1))

sd(basketball\_teams$team1)

sd(basketball\_teams$team2)

# Правилото на Чебишев, което е валидно за всички множества, ни казва, че 89% от наблюденията

# лежат в интервала (X\_mean - 3\*X\_std; X\_mean + 3\*X\_std), където X\_mean - средната стойност и

# X\_std - стандартното отклонение.

# При камбановидна форма на разпределението, този процент достига до 99.7

N <- 10^4

set.seed(94171)

dist1 <- rnorm(N)

dist2 <- rgamma(N, 3)

par(mfrow = c(1, 2))

plot(density(dist1), lwd = 2, main = "Плътност", xlab = "Нормално разпределение",

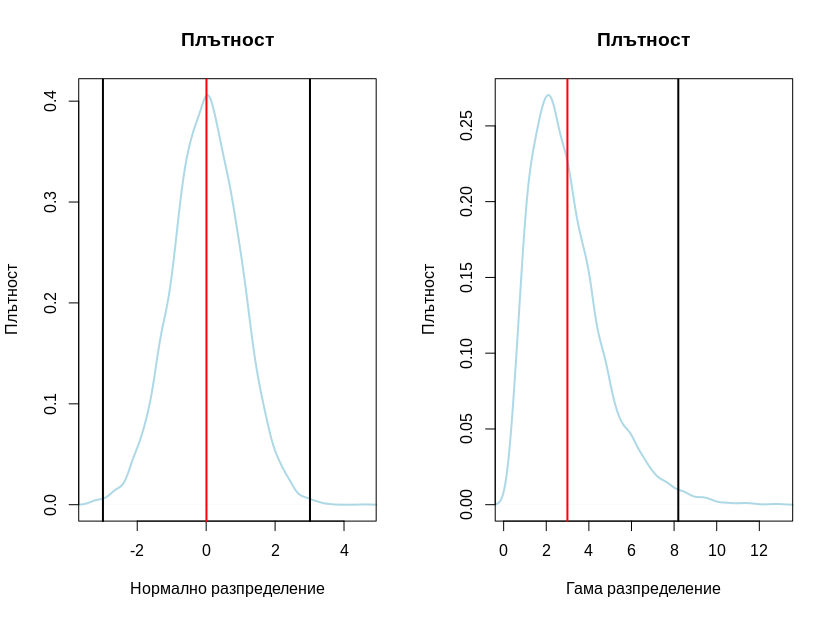
ylab = "Плътност", col = "lightblue", xlim = range(dist1))

mu <- mean(dist1); sigma <- sd(dist1)

abline(v = c(mu - 3\*sigma, mu, mu + 3\*sigma), lwd = 2, col = c("black", "red", "black"))

plot(density(dist2), lwd = 2, main = "Плътност", xlab = "Гама разпределение",

ylab = "Плътност", col = "lightblue", xlim = range(dist2))



mu <- mean(dist2); sigma <- sd(dist2)

abline(v = mu + c(-3, 0, 3)\*sigma, lwd = 2, col = c("black", "red", "black"))

par(mfrow = c(1, 1))

mu <- mean(dist2); sigma <- sd(dist2)

round(sum(dist2 >= mu - 3\*sigma & dist2 <= mu + 3\*sigma)\*100/N, 2)

# 3.3. The five number summary

# тази статтистика най-често показва минималната стойност, 1-ви квартил, медиана (2-ри квартил),

# 3-ти квартил и максималната стойност

# В R използваме фунцкциите summary() и fivenum()

summary(dist1)

fivenum(dist1)

par(mfrow = c(1, 2))

plot(density(dist1), lwd = 2, main = "Плътност", xlab = "Нормално разпределение",

ylab = "Плътност", col = "lightblue", xlim = range(dist1))

abline(v = fivenum(dist1), lwd = c(1.5, rep(2, 3), 1.5), col = c("black", "red", "red", "red", "black"),

lty = c(1, rep(3, 3), 1))

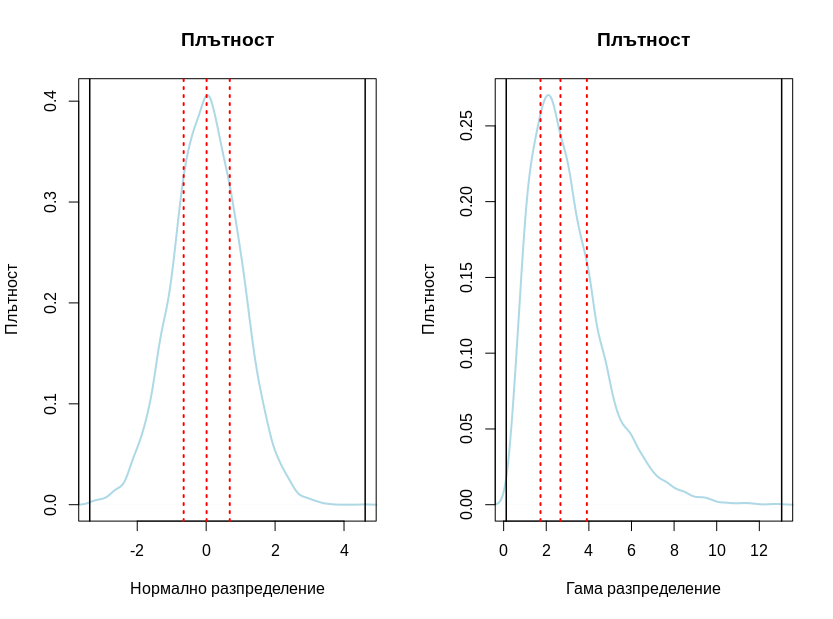
plot(density(dist2), lwd = 2, main = "Плътност", xlab = "Гама разпределение",

ylab = "Плътност", col = "lightblue", xlim = range(dist2))

abline(v = fivenum(dist2), lwd = c(1.5, rep(2, 3), 1.5), col = c("black", "red", "red", "red", "black"),

lty = c(1, rep(3, 3), 1))

par(mfrow = c(1, 1))



# 3.4. Interquartile Range и MAD

# Тези два вида оценки на дисперсията е препоръчително да се използват, когато за оценка на центъра

# на разпределението се използва медианата. И двете оценки се водят "стабилни" към екстремалните стойности

# Nielsen Company е публикувала информация колко часа седмично американците прекарват пред телевизора.

# Това е извадка от 20 човека

tv\_viewing\_times <- c(25, 41, 27, 32, 43, 66, 35, 31, 15, 5, 34, 26, 32, 38, 16, 30, 38, 30, 20, 21)

# За да покажем как екстремалните стойности влияят върху част от оценките ще добавим голяма стойност,

# например 240 часа

tv\_viewing\_times\_new <- c(tv\_viewing\_times, 240)

# Интерквартилния обхват се изчислява като разлика между 3-ти квартил и 1-ви квартил

summary(tv\_viewing\_times)

summary(tv\_viewing\_times)[c(2, 5)]

diff(summary(tv\_viewing\_times)[c(2, 5)])

> tv\_viewing\_times <- c(25, 41, 27, 32, 43, 66, 35, 31, 15, 5, 34, 26, 32, 38, 16, 30, 38, 30, 20, 21)

> tv\_viewing\_times\_new <- c(tv\_viewing\_times, 240)

> summary(tv\_viewing\_times)

Min. 1st Qu. Median

5.00 24.00 30.50

Mean 3rd Qu. Max.

30.25 35.75 66.00

> summary(tv\_viewing\_times)[c(2, 5)]

1st Qu. 3rd Qu.

24.00 35.75

> diff(summary(tv\_viewing\_times)[c(2, 5)])

3rd Qu.

11.75

# Базовата функция в R се казва IQR()

IQR(tv\_viewing\_times)

> IQR(tv\_viewing\_times)

[1] 11.75

IQR(tv\_viewing\_times)

IQR(tv\_viewing\_times\_new)

> IQR(tv\_viewing\_times\_new)

[1] 13

# Както се вижда няма кой знае колко голяма промяна след добаяването на екстремалната стойност

# Какво обаче би станало, ако използваме стандартното отклонение?

> sd(tv\_viewing\_times)

[1] 12.64859

> sd(tv\_viewing\_times\_new)

[1] 47.40243

# Разликата скача в пъти

# Ето защо при наличието на екстремуми е по-разумно да използваме медианата за оценка на центъра и

# IQR или mad за оценка на дисперсията

# MAD

# Оценката MAD представлява медианата на вектора с абсолютните стойности от разлики от стойността и

# медианата на самия вектор. Резултатът е умножен по 1.4826

# Формулата е записана по-долу

X\_median <- median(tv\_viewing\_times\_new)

X\_median

X\_diff <- abs(tv\_viewing\_times\_new - X\_median)

X\_diff

median(X\_diff)\*1.4826

mad(tv\_viewing\_times\_new)

> mad(tv\_viewing\_times\_new)

[1] 10.3782

# - Добре, при оценката на вариацията имаме значима промяна. Как ли стоят нещата с оценките за центъра?

# - Екстремумите оказват влияние и при оценката за центъра. Ето защо, при наличие на такива стойности,

# предпочитаме да използваме медианата, вместо средната стойност.

> mean(tv\_viewing\_times)

[1] 30.25

> mean(tv\_viewing\_times\_new)

[1] 40.2381

> median(tv\_viewing\_times)

[1] 30.5

> median(tv\_viewing\_times\_new)

[1] 31

# Разликата е очевидна

# - Между другото имаме и други опции при наличието на екстремални стойности - bootstrap метод и

# trimmed mean. За съжаление, няма да можем да се запознаем в курса с bootstrap, но ви го препоръчвам.

# Какво прави trim опцията? Тя премахва по част от най-големите и най-малките стойности.

# В нашия случай, ние сме посочили, че искаме да махмен 5% от най-големите и най-малките стойности.

# Тоест ще вземем 5/2 = 2.5% от най-малките стойности и 2.5 от най-големите.

> mean(tv\_viewing\_times, trim = 0.05)

[1] 29.66667

> mean(tv\_viewing\_times\_new, trim = 0.05)

[1] 31.57895

# Както виждаме стойностите са близки

# 4. Графично представяне на разпределение

# 4.1. Barplot

# Използваме barplot, когато искаме да представим честотното разпределение на категорийни променливи

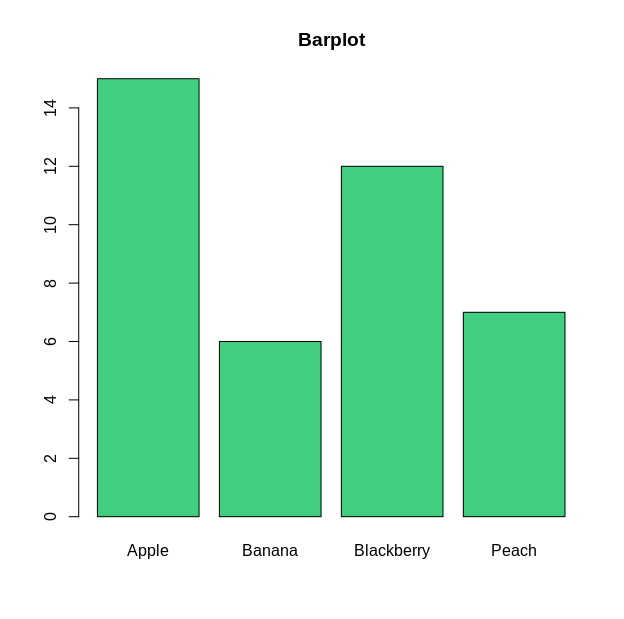
set.seed(4012)

fruits <- sample(x = c("Apple", "Banana", "Blackberry", "Peach"), size = 40, replace = T,

prob = c(0.4, 0.1, 0.3, 0.2))

tt <- table(fruits); tt

barplot(height = tt, col = "seagreen3", main = "Barplot")



?barplot

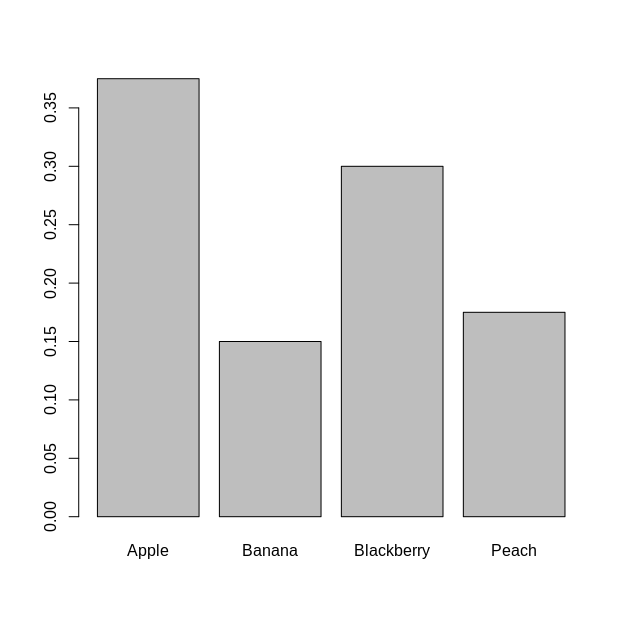
# height - приема вектор или матрица с числови стойности като вход. Стойностите могат да бъдат и отрицателни

# main - заглавие на графиката

# col - цвят на стълбовете

# Тези параметри са основни и ги има и при другите графики

barplot(prop.table(tt))



# 4.2. Хистограма

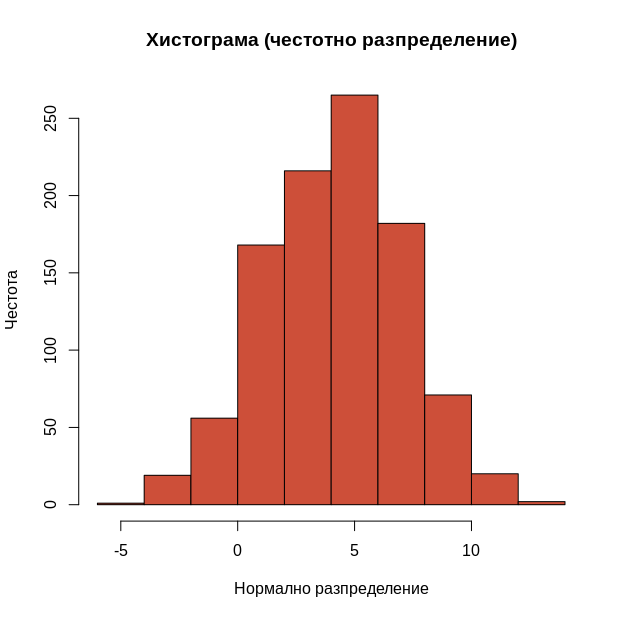
# Използваме хистограма, когато искаме да представим разпределението на непрекъснати променливи

set.seed(7821)

r1 <- rnorm(n = 10^3, mean = 4, sd = 3)

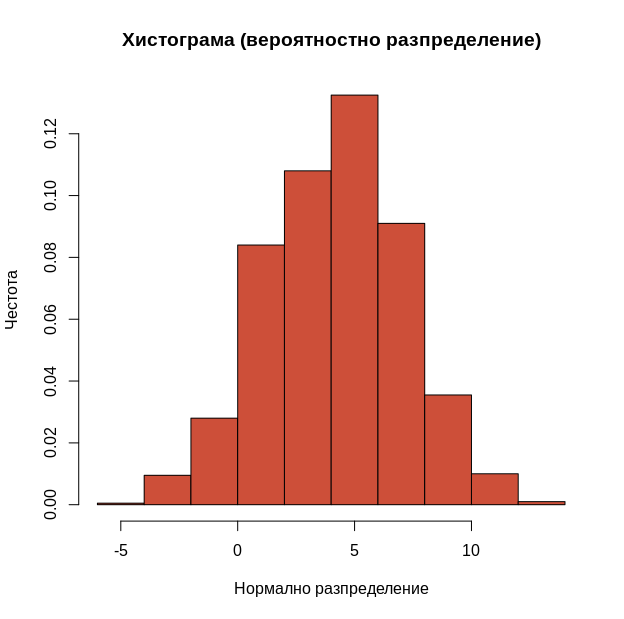
hist(r1, main = "Хистограма (честотно разпределение)", xlab = "Нормално разпределение", ylab = "Честота",

col = "tomato3")



hist(r1, main = "Хистограма (вероятностно разпределение)", xlab = "Нормално разпределение", ylab = "Честота",

col = "tomato3", prob = T)



colors() # различни видове цветове, които се подържат от базовия пакет в R

# 4.3. Piechart

# Използваме piechart-, когато боравим с категорийни променливи и искаме да

# изобразим процентното им разпределение

cities <- c(rep("London", 14), rep("New York", 49), rep("Singapore", 28), rep("Mumbai", 36))

cities.table <- table(cities)

> cities.table

cities

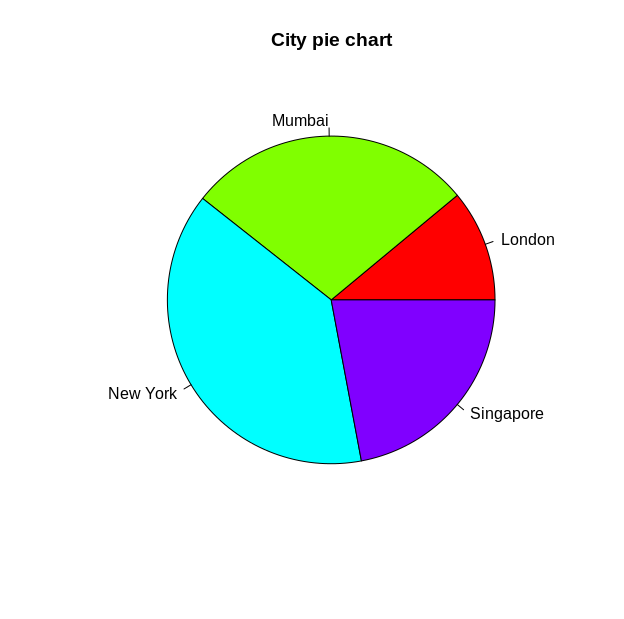
London Mumbai New York

14 36 49

Singapore

28

pie(cities.table, main = "City pie chart", col = rainbow(length(cities.table)))



# Броя на цветовете е хубаво да бъде равен на броя на категориите. В противен случай два сигмента ще бъдат оцветени в един и същи цвят.

piepercent<- round(100\*cities.table/sum(cities.table), 1)

pie(cities.table, labels = piepercent, main = "City pie chart", col = rainbow(n = length(cities.table)))

# rainbow(n) - връща n на брой цветове, произтичащи от дъгата

legend(x = "topright", legend = c("London","New York","Singapore","Mumbai"), cex = 0.8,

fill = rainbow(length(cities.table)))

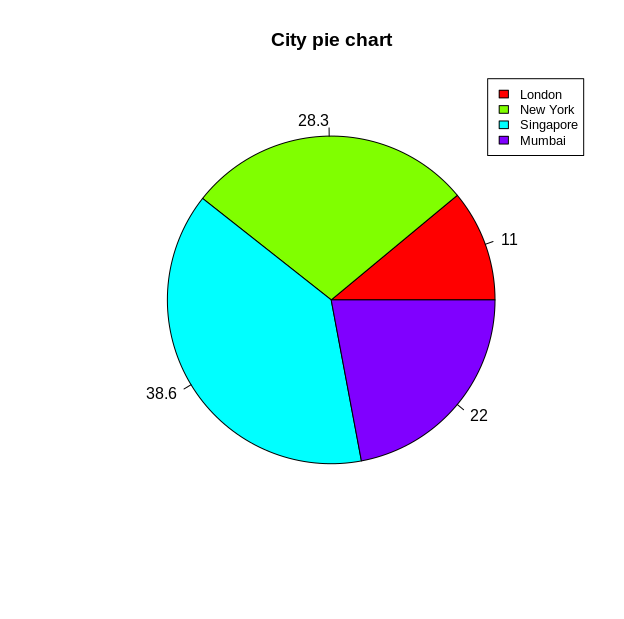
?legend

# x - разположение на графиката - може да слагате както координати, така и да описвате позицията

# legend - имената на категориите

# cex - големината на текста

# fill - цвета, на който отговаря текста в легендата



# 4.4. Boxplot

# При едномерния анализ, boxplot-а се използва, за да откриване на потенциални outlier-и.

tv\_viewing\_times <- c(25, 41, 27, 32, 43, 66, 35, 31, 15, 5, 34, 26, 32, 38, 16, 30, 38, 30, 20, 21)

tv\_viewing\_times\_new <- c(tv\_viewing\_times, 240)

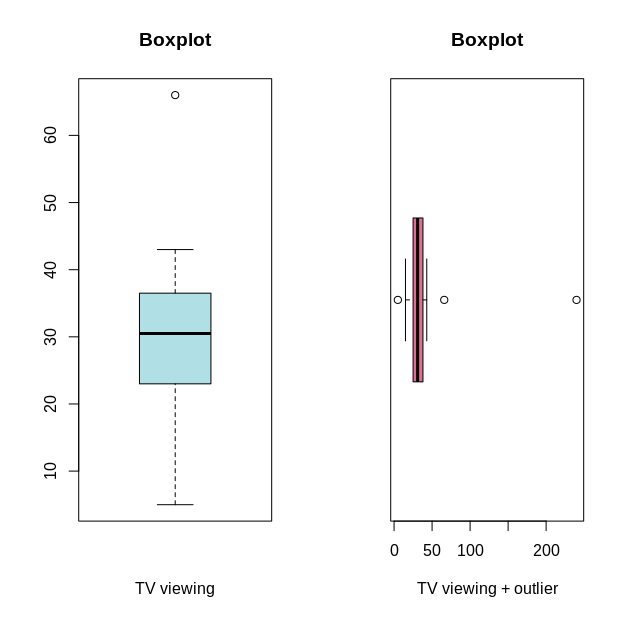
par(mfrow = c(1, 2))

boxplot(tv\_viewing\_times, col = "powderblue", main = "Boxplot", xlab = "TV viewing")

boxplot(tv\_viewing\_times\_new, horizontal = T, col = "palevioletred", main = "Boxplot",

xlab = "TV viewing + outlier")

par(mfrow = c(1, 1))



# 4.5. Q-Q plot

# Проверяваме дали стойностите на наблюдаваната променлива се доближават до теоретичните стойности на някое разпределение.

emp <- c(19.14, 6.29, 17.02, 6.13, 1.63, 18.78, 9.43, 11.21, 2.89, 9.52, 9.49, 4.83, 13.26, -0.96,

5.12, 1.39, 6.76, 2.1, 4.32, 1.38, 10.7, 9.01, 4.73, 11.59, 7.22, 1.53, 8.36, 10.91, 6.49,

3.69, 2.06, 15.92, 16.76, 18.13, 10.22, 19.25, 9.65, 17.75, 2.52, 1.24, 18.51, 11.52, 14.67,

12.65, 11.22, 27.78, 1.76, 9.64, 11.42, 12.29)

d1 <- rnorm(n = 10^2, mean = mean(emp), sd = sd(emp))

d2 <- rcauchy(n = 10^2, location = mean(emp), scale = sd(emp))

par(mfrow = c(1, 2))

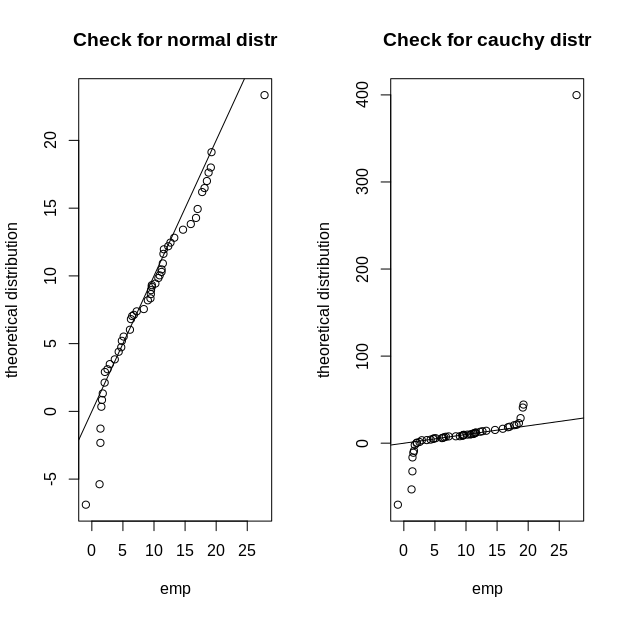
qqplot(emp, d1, ylab = "theoretical distribution", main = "Check for normal distr")

abline(a = 0, b = 1)

qqplot(emp, d2, ylab = "theoretical distribution", main = "Check for cauchy distr")

abline(a = 0, b = 1)

par(mfrow = c(1, 1))



# abline() - чертае права линия

# a - изместване по Х, b - ъгъл на правата, v - вертикална линия, h - хоризонтална линия

----------------------------------------------- Упражнение 4 -----------------------------------------------------

# N-мерни променливи и изследване на връзките между тях

# Какво предсталяват "n"-мерните данни?

# Едномерните данни, това са масиви/листа от обекти (числа, стрингове, дати, друг тип обекти).

# При двумерните данни имаме колекция от едномерни данни. Тоест, представянето е във формата на

# матрици, data frame-ове или друга подобна структура, при която най-често по редове са представени

# примерите/елементите, а по колони техните признаци (променливите).

# Пример за многомерни данни е

data("mtcars")

head(mtcars)

# Нека да изследваме обема на двигателя за въпроснтие коли. Първо ще построим хистограма

hist(x = mtcars$disp, col = "red", xlab = "Displacement (u.in.)", main = "Histogram")

summary(mtcars$disp)

sd(mtcars$disp)

abline(v = mean(mtcars$disp), lwd = 2, lty = 4)

abline(v = median(mtcars$disp), lwd = 2, lty = 3, col = "blue")

A screenshot of a cell phone

Description automatically generated

# От хистограмата се вижда, че имаме два пика. Тоест, разпределението на променливата е

# бимодално. Черната вертикалана прекъсната линия показва къде се намира средната стойност,

# а синята прекъсната - медината. И в двата случая, малко трудно можем да приемем, че тпва е

# очакването на разпределението.

# Нека сега да проверим, какво би станало, ако групираме данните по броя на цилиндрите

disp\_cyl4 <- mtcars$disp[which(mtcars$cyl == 4)]

disp\_cyl6 <- mtcars$disp[which(mtcars$cyl == 6)]

disp\_cyl8 <- mtcars$disp[which(mtcars$cyl == 8)]

par(mfrow = c(2, 2))

hist(x = disp\_cyl4, col = "red", xlab = "4 cylinders", main = "Histogram of displacement (u.in.)")

hist(x = disp\_cyl6, col = "lightblue", xlab = "6 cylinders", main = "Histogram of displacement (u.in.)")

hist(x = disp\_cyl8, col = "forestgreen", xlab = "8 cylinders", main = "Histogram of displacement (u.in.)")

par(mfrow = c(1, 1))

A screenshot of a cell phone

Description automatically generated

> summary(disp\_cyl4)

Min. 1st Qu. Median Mean 3rd Qu. Max.

71.10 78.85 108.00 105.14 120.65 146.70

> sd(disp\_cyl4)

[1] 26.87159

>

> summary(disp\_cyl6)

Min. 1st Qu. Median Mean 3rd Qu. Max.

145.0 160.0 167.6 183.3 196.3 258.0

> sd(disp\_cyl6)

[1] 41.56246

>

> summary(disp\_cyl8)

Min. 1st Qu. Median Mean 3rd Qu. Max.

275.8 301.8 350.5 353.1 390.0 472.0

> sd(disp\_cyl8)

[1] 67.77132

# За групата на двигетелите, които имат 4 цилиндъра, все още не може да получим добра оценка

# за очакването, no за другите две групи - можем, защото имаме по един връх.

# Анализирайки зависимостите на една променлива от други променливи, ние успяваме да подобрим оценките на параметрите, които са ни неогходи. По този начин правим прогнозите си по-точни.

# Изследване на двумерни данни

# 1. Категорийни (обясняващи) VS категорийни (зависими)

# Връзките между тези променливи най-лесно се виждат с помощта на cross таблици и barplot-ове.

# Пример: Направили сме хипотетично прочуване, което измерва дали студентите, които пушат,

# учат по-малко.

smokes <- c("Y", "N", "N", "Y", "N", "Y", "Y", "Y", "N", "Y")

amount <- c("0 - 5 hours", "5 - 10 hours", "5 - 10 hours", "more than 10 hours",

"more than 10 hours", "0 - 5 hours", "5 - 10 hours", "0 - 5 hours",

"more than 10 hours", "5 - 10 hours")

table(amount, smokes)

> table(amount, smokes)

smokes

amount N Y

0 - 5 hours 0 3

5 - 10 hours 2 2

more than 10 hours 2 1

# Данните показват, че пушачите учат по-малко от непушачите. Нека да разгледаме резулатите

# не като честоти, а като проценти. За целта изпозлвае командата

prop.table(x = table(amount, smokes))

> prop.table(x = table(amount, smokes))

smokes

amount N Y

0 - 5 hours 0.0 0.3

5 - 10 hours 0.2 0.2

more than 10 hours 0.2 0.1

# Показва ни в коя група, колко процента от данните попадат.

prop.table(x = table(amount, smokes), margin = 1)

> prop.table(x = table(amount, smokes), margin = 1)

smokes

amount N Y

0 - 5 hours 0.0000000 1.0000000

5 - 10 hours 0.5000000 0.5000000

more than 10 hours 0.6666667 0.3333333

# Параметърът "margin" задава как желаем да изчисляваме процентите - по редове или по колони.

# От данните виждаме, че имаме нарастване в процента на непушещите студентите, спрямо броя на bчасовете, които отделят за учене.

# Сега ще разгледаме графичното представяне на данните.

barplot(table(smokes, amount))

# Малко трудно бихме видяли разликите, освен ако не са фрапиращи.

# В долния код ще се опитаме да нормализираме стойностите като използваме процентните

# съотношения. При този подход, ясно се вижда превъзходствата на едни признаци в една група,

# спрямо друга.

barplot(prop.table(x = table(smokes, amount), margin = 2))

A screenshot of a cell phone

Description automatically generated

# Друг подход е описаният по-долу

barplot(table(smokes, amount), beside = TRUE, legend.text = T)

# При този подход съще лесно се забелязват разликите в отделните групи. В сегашния barplot сме задали и легенда

A screenshot of a cell phone

Description automatically generated

# Освен че можем да изведем легенда на графиката (legend.text = TRUE), то можем и да я попълним със стойности, които ни трябват. Попълването е показано в примера по-долу.

barplot(table(amount, smokes), main = "table(amount, smokes)", beside = TRUE,

legend.text = c("less than 5", "5 - 10", "more than 10"))

A screenshot of a cell phone

Description automatically generated

# 2. Категорийни (обясняващи) VS числови (зависими)

# Когато имаме такава конфигурация при връзките, то най-удачно е да използваме One-way ANOVA

# и t-test или техните непараемтрични еквиваленти. Тези анализи ще ги учим по-нататък в курса по статистика. Ако искаме да ги изследваме графично, удачно решение е boxplot графиките.

amount <- c(5, 5, 5, 13, 7, 11, 11, 9, 8, 9, 11, 8, 4, 5, 9, 5, 10, 5, 4, 10)

category <- c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2)

tt <- boxplot(amount ~ category)

A screenshot of a social media post

Description automatically generated

# Както се вижда, лесно могат да се сравнят двете категории. Средната дебела линия във всеки един boxplot е медианата, страните на правоъгълника са 1 и 3-ти квартил, а дължината на опашките са минималната и максималните стойности, като са изключени потенциалните outlier-и. Тоест интерпретацията на тази графика е, че стойнсотите на първата група като цяло са по-големи защото и медианата и третия квартил за първата група са по-големи от тези на втората. Отделно, минималната стойност и първия квартил за първата група съвпадат ( = 5), докато минималната стойност на втората група е 4.

# 3. Числови (обясняващи) VS категорийни (зависими)

# Този случай на връзка е сходен с горния. Затова тук също можем да използваме One-way ANOVA или

# t-test, непараметричните им екваваленти и boxplot-ове. Също така можем да използваме и

# логистичната регресия.

# Този тип връзка ще бъде обяснена по-нататък

# 4. Числови (обясняващи) VS числови (зависими)

# Това е може би групата, за която съществуват най-много похвати за анализи. Променливите от числов тип могат да бъдат превърнати в категорийни и следователно за тях важат горните типове анализи. Това, разбира се, би довело до загуба на информация, но в определени случаи е по-подходящо заради по-голямата стабилност на моделите. Похватите, които са характерни за изследването на този тип връзки са най-често корелационен анализ и регресионен анализ, както и dotplot (графично представяне на връзката). Почти винаги изследването на този тип връзка следва последователността dotplot, корелационен анализ и регресионен анализ.

# Да разгледаме пример от данните "mtcars". Интересуват ни променливите disp (обем на двигателя)

# и wt (тегло)

plot(mtcars$disp, mtcars$wt)

# От графиката се вижда, че съществува положителна линейна връзка. Тоест с нарастване на обема на двигателя, нараства и теглото на автомобила. Следователно, можем да използваме линеен модел, за да моделираме връзката.

# ИНФОРМАЦИЯ, КОЯТО НЯМА ДА Я ИМА НА ИЗПИТА/КОНТРОЛНИТЕ

# Преди да продължим с корелационния и регресионния анализ, нека да разгледаме друг пример. Този път данните ще бъдат симулирани. И връзката няма да бъде линейна, а кубична.

set.seed(4455)

x <- runif(1000, -3, 3)

y <- x^3 - 3 + rnorm(length(x), sd = 2)

plot(x, y)

# Както се вижда от графиката, този тип връзка не прилича на линейна. Но чрез подходяща трансформация, връзката може да се представи като линейна. Например, ако създадем нова променлива

x3 <- x^3

# Тогава, новата променлива x3 е в линейна зависимост с променливата y

par(mfrow = c(1, 2))

plot(x, y)

plot(x3, y)

par(mfrow = c(1, 1))

# Корелационен анализ

# Корелационният анализ измерва силата на линейна връзка между две променливи. Коефициентът на корелация (rho) принадлежи на интервала [-1, 1]. Силата на връзката се определя от абсолютната стойност на rho. Въпреки, че силата на връзката с субективна, все пак можем да определим някакви нива.

N <- 1000

# abs(rho) = 1 - Детерминистична връзка (y = f(x)). За една стойност на x имаме точно една единствена стойност на y

set.seed(3654)

x1 <- runif(N)

y1 <- 3\*x1 + 4

rho1 <- round(cor(x1, y1), 3)

# 0.9 <= abs(rho) < 1 - Много силна корелация на между x иy

set.seed(3654)

x2 <- runif(N)

y2 <- 3\*x2+ 4 + rnorm(N, sd = 0.2)

rho2 <- round(cor(x2, y2), 3)

# 0.75 <= abs(rho) < 0.9 - Силна корелация на между x и y

set.seed(3654)

x3 <- runif(N)

y3 <- -3\*x3 + 4 + rnorm(N, sd = 0.5)

rho3 <- round(cor(x3, y3), 3)

# 0.5 <= abs(rho) < 0.75 - Средна корелация на между x и y

set.seed(3654)

x4 <- runif(N)

y4 <- -3\*x4 + 4 + 1\*rnorm(N)

rho4 <- round(cor(x4, y4), 3)

# 0 <= abs(rho) < 0.5 - Слаба корелация на между x и y

set.seed(3654)

x5 <- runif(N)

y5 <- 3\*x4 + 4 + 3\*rnorm(N)

rho5 <- round(cor(x5, y5), 3)

par(mfrow = c(2, 3))

plot(x1, y1, main = paste("rho:", rho1))

abline(a = 4, b = 3, col = "red", lwd = 2)

plot(x2, y2, main = paste("rho:", rho2))

abline(a = 4, b = 3, col = "red", lwd = 2)

plot(x3, y3, main = paste("rho:", rho3))

abline(a = 4, b = -3, col = "red", lwd = 2)

plot(x4, y4, main = paste("rho:", rho4))

abline(a = 4, b = -3, col = "red", lwd = 2)

plot(x5, y5, main = paste("rho:", rho5))

abline(a = 4, b = 3, col = "red", lwd = 2)

par(mfrow = c(1, 1))

# От графиките се вижда, че колко по-разпръснати са наблюденията около правата,

# толкова корелацията намалява

# Командата за корелация е cor. С командата може да се изследват както връзките

# между две променливи, така и връзките между N-мерни ЧИСЛОВИ данни.

# Формулата за коралация ще я опишем с примера по-долу

X <- x3; Y <- y3

X\_mean <- mean(X); Y\_mean <- mean(Y)

XY <- (X - X\_mean)\*(Y - Y\_mean)

XX <- (X - X\_mean)^2; YY <- (Y - Y\_mean)^2

sum(XY)/sqrt(sum(XX)\*sum(YY)) # Стойността на корелацията

cor(x3, y3)

cor(mtcars$mpg, y = mtcars$hp)

# Връща ни само едно число - корелацията между двете променливи

cor(mtcars[, c("mpg", "disp", "hp", "drat", "wt", "qsec")])

# Връща СИМЕТРИЧНА матрица (A[i, j] == A[j, i]) с корелациите между отделните

# променливи.

# Интересно е, че, по главния диагонал, всички стойности са единици. Това е

# следствие от формулата

# Съществуват три основни вида корелации - Pearson, Spearman и Kendall. Първата

# корелация е параметрична оценка на връзката между две променливи, докато останалите

# две - непараметрични.

# Тоест корелацията на Pearson е по-точна, но е неустойчива при наличието на outlier-и

# Останалите две корелации са по-стабилни и не толкова точни.

# Най-лесно това ще го демонстрираме с примера по-долу

set.seed(4413)

x <- sort(rnorm(200, mean = 2))

y <- x + sqrt(1 - 0.8^2)\*rnorm(length(x))

plot(x, y, main = paste("Pearson's rho:", round(cor(x, y), 2))) # корелацията е 0.85

# Нека обаче да добавим няколко outlier-а

x1 <- c(x, 3.4, 3, 3.8, 3.5, 4, 4.1)

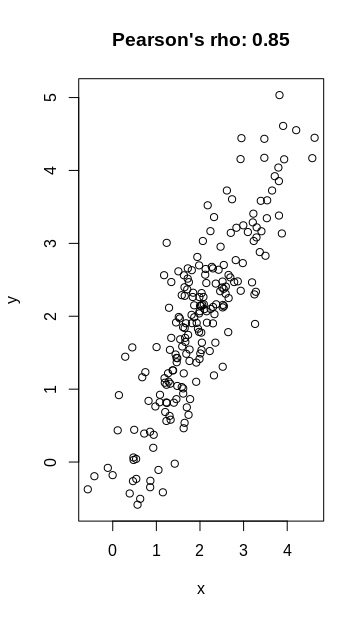
y1 <- c(y, 17, 18.5, 19.2, 19, 20, 22)

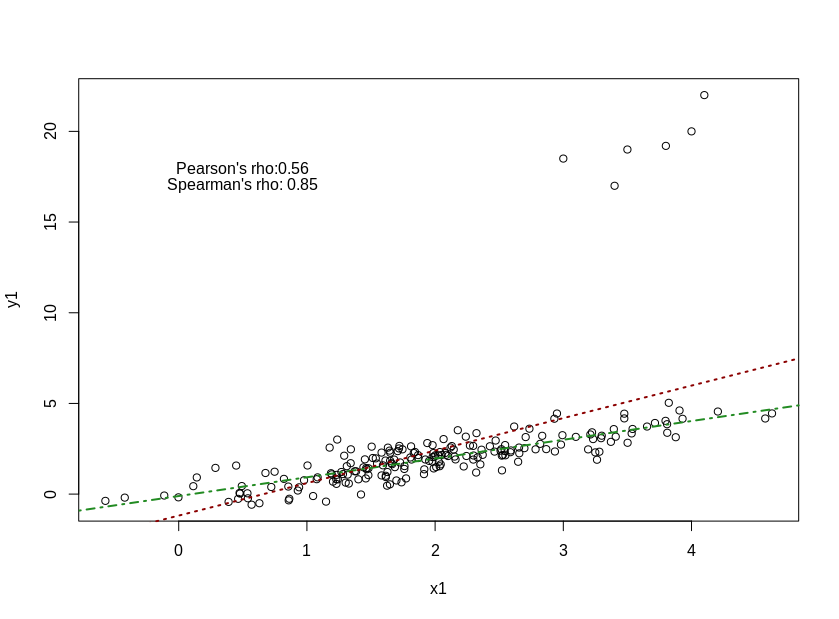
plot(x1, y1)

abline(lm(y ~ x), col = "forestgreen", lwd = 2, lty = 4)

abline(lm(y1 ~ x1), col = "darkred", lwd = 2, lty = 3)

text(x = 0.5, y = 18, labels = paste0("Pearson's rho:", round(cor(x1, y1), 2)))

text(x = 0.5, y = 17, labels = paste0("Spearman's rho: ", round(cor(x1, y1, method = "spearman"), 2)))



----------------------------------------------- Упражнение 5 -----------------------------------------------------

# Разпределения

############################

# Преди да започнем с характеристиките на разпределенията трябва да знаем, че в R всяко разпределение има разновидност, която стартите с d-(име на разпределение),

# p-(име на разпределение), q-(име на разпределение) и r-(име на разпределение). Тези

# означения показват какво искаме да правим:

# d- - изчислява вероятността за сбъдване на събитието x

# p- - изчислява вероятността на предварително зададен квантил за разпределението

# q- - изчислява квантилът на предварително зададена вероятност за разпределението

# r- - генериране на случайни величини за разпределението

# p- е обратна функция на q-. Например qnorm(pnorm(0.9)) = 0.9, pnorm(qnorm(0.9)) = 0.9

############################

# 1. Биномно разпределение

# Биномното разпределение е дисктретно разпределение, което брои успехите в редица

# от n независими опити. Биномното разпределение приема параметри "n" - броя на опитите

# и "p" - вероятността за настъпване на успех.

# Вероятността за настъпване на събитие "x" е P(x) = C(n, x)\*p^x\*(1-p)^(n-x),

# за x = 0, ..., n

# В R, функциите за биномно разпределение са rbinom(), dbinom(), pbinom(), qbinom()

# n - броят на симулациите, size - големианта на редицата от независимите опити

# p - вероятността за настъпване на успех от един опит

# Частния случай, когато size = 1 поражда Бернулиево разпределение. Тоест,

# искаме да генерираме само един опит с вероятност за успех "p"

> rbinom(n = 1, size = 1, p = 0.2)

[1] 0

> rbinom(n = 10, size = 1, p = 0.2)

[1] 0 1 0 0 1 0 1 0 1 0

# Биномно разпределение

> rbinom(n = 1, size = 6, p = 0.2)

[1] 0

> rbinom(n = 10, size = 6, p = 0.2)

[1] 3 0 1 1 1 2 4 1 2 2

> set.seed(4442)

> rbd <- rbinom(n = 100, size = 6, p = 0.2)

>

> rbd

[1] 1 1 0 1 1 2 2 1 2 3 0 0 1 1 0 1 0 0 0 0 2 2 0 1 1 0 1 1 3 2 3 3 1 2 0 1 2 2

[39] 2 0 0 3 1 2 2 2 0 0 1 1 1 0 2 1 1 0 2 0 1 1 1 1 1 1 1 3 2 1 0 1 1 1 2 2 2 0

[77] 1 1 1 1 1 0 3 1 2 2 1 4 1 1 1 1 3 2 1 0 1 1 1 0

# Средната стойност и медианата са равни на size\*p, къдетo "size" и "p" са броят на

# опитите в редицата от експерименти и вероятността за успех.

# Вариацията е равна на size\*p\*(1 - p)

> summary(rbd)

Min. 1st Qu. Median Mean 3rd Qu. Max.

0.00 1.00 1.00 1.18 2.00 4.00

> var(rbd)

[1] 0.8359596

# 2. Геометрично разпределение

# Геометричното разпределение е дискретно разпределение, което показва броя на

# опитите до настъпването на успех.

# Вероятността е P(x) = p\*(1-p)^x, за x = 0, ..., n

> set.seed(46871)

> rgd <- rgeom(n = 100, prob = 0.2)

>

> rgd

[1] 0 5 1 8 23 5 4 1 4 0 10 1 0 3 7 28 2 5 12 6 8 1 5 0 3

[26] 4 10 9 6 6 1 2 11 4 0 0 5 1 9 13 4 4 3 5 0 2 10 7 0 0

[51] 1 11 5 6 6 2 9 2 1 3 4 8 0 0 1 1 0 10 3 0 1 1 1 14 0

[76] 1 10 3 17 0 0 9 0 7 3 0 11 1 1 0 4 0 3 7 0 2 3 9 0 6

# Средната стойност са равни на 1/p, "p" е вероятността за успех.

# Вариацията е равна на (1-p)/(p^2)

> summary(rgd)

Min. 1st Qu. Median Mean 3rd Qu. Max.

0.00 1.00 3.00 4.45 7.00 28.00

> var(rgd)

[1] 24.51263

# 3. Отрицателно биномно разпределение

# Отрицателното биномно разпределение е дискретно разпределение, което измерва броя на успехите в редица от бернулиеви опити до настъпване на r на брой неуспеха. Разпределението има два параметъра r (броя на неуспехите) и p (вероятността за успех за всеки опит). При r = 1 имаме геометрично разпределение.

# Функцията на масата е P(x) = C(x+r-1, x)\*(1-p)^r\*p^x, за x = 0, 1, ...

# Очакването на отрицателното биномно разпределение е r\*(1-p)/p, а вариацията е r\*(1-p)/p^2

> set.seed(4457)

> N <- 1000

> nbd <- rnbinom(n = N, size = 30, prob = 0.2)

> # size - броят на успехите

> hist(nbd)

A screenshot of a cell phone

Description automatically generated

> summary(nbd)

Min. 1st Qu. Median Mean 3rd Qu. Max.

49.0 104.0 119.0 120.5 136.0 209.0

> var(nbd)

[1] 605.6595

# 4. Поасоново разпределение

# Поасоновото разпределение е дискретно разпределение, което измерва вероятността "lambda" (n\*p) на брой независими събития да се случат в определен интервал от време. При Поасоновото разпределение "n" клони към безкрайност, a "p" клони към нула. Това е и връзката между биномно разпределение и поасоново.

# Вероятността е P(x) = exp(-lambda)\*lambda^x/(x!)

# При поасоновото разпределение, средната стойност и вариацията са равни на lambda

> set.seed(1477)

> rpd <- rpois(n = 100, lambda = 3.4)

> summary(rpd)

Min. 1st Qu. Median Mean 3rd Qu. Max.

0.0 2.0 3.0 3.1 4.0 10.0

> var(rpd)

[1] 3.888889

# 5. Равномерно непрекъснато разпределение

# Равномерното непрекъснато разпределение много наподобява равнометрното дискретно.

# Разликата е в това, че вместо дискретни стойности, при непрекъснатото може да

# приема в целия интервал.

# Вероятността P(X < x) = (x - a) / (b - a)

# Равномерното непрекъснато разпределение приема параметри a и b, които са

# минимална и максимална стойности.

# Геенрирането на равномерно разпределение става с функцията runif(n, a, b)

> runif(1, 0, 2)

[1] 1.705281

> runif(5, 0, 2)

[1] 0.5429592 0.9797916 0.2432638 1.4614488 0.4039031

>

> set.seed(6674)

> x <- runif(1000) # get the random numbers

> hist(x, prob = T, breaks = 10)

A screenshot of a social media post

Description automatically generated

# 6. Нормално разпределение

# Нормалното разпределение е непрекъснато разпеделение, което има два основни

# параметъра mu (средна стойност) и sigma (стандартно отклонение).

> set.seed(96637)

> rnd <- rnorm(n = 1000, mean = 3, sd = 4)

> summary(rnd)

Min. 1st Qu. Median Mean 3rd Qu. Max.

-9.9389 0.2999 3.0284 3.0739 5.7400 17.2789

> var(rnd)

[1] 16.45538

# Както не един път сме споменавали, нормалното разпределение има формата на камбана

> hist(rnd, prob = TRUE)

A screenshot of a cell phone

Description automatically generated

# Всяко едно нормално разпределение, с очакване "mu" и стандартно отклонение

# "sigma", може да се стандартизира в нормално разпределение с очакване 0 и стандартно отклонение 1. Стандартизацията се нарича Z-score, а формулата е Z = (x – m ean(x))/sd(x)

rnd1 <- (rnd - mean(rnd)) / sd(rnd)

# R притежава функция "scale", която го прави автоматично. Функцията е притежава

# параметрите center (булева или числова променлива) и scale (булева или числова

# променлива)

rnd2 <- scale(rnd)

all(rnd1 == rnd2) # Проверяваме дали всички стойности на rnd1 и rnd2 са равни

# 7. Експоненциално

# Друго важно непрекъснато разпределение е екопоненциалното. Това разпределение

# e подходящо за употреба в случаи, когато имаме работа с промеливи, свързани с време. Експоненциалното разпределение има само един параметър lambda.

# Експоненциалното разпределение се свързва с Поасоновото разпределение и като с него се оценява времето между настъпванията на две събития. Това разпределение се разглежда и като непрекъснат аналог на геометричното разпределение.

# Средната стойност на експоненциалното разпределение е равна на 1/lambda, а

# вариацията = 1/lambda^2.

> set.seed(7114)

> lambda <- 4

> red <- rexp(1000, rate = 1/lambda)

> hist(red)

>

> summary(red)

Min. 1st Qu. Median Mean 3rd Qu. Max.

0.000144 1.168403 2.878353 3.999495 5.685135 29.931341

> var(red)

[1] 15.54045

# 8. t разпределение

# Друго важно непрекъснато разпределение е t разпределението. Това разпределение се използва за оценка на параметрите на популацията, когато размерът на извадката е малък или когато стандартното отклонение на популацията е неизвестно. T разпределението има един параметър nu = n - 1 (n - броят на наблюденията), който представлява степените на свобода.

# Средната стойност и медианата на t разпределението са 0, а вариацията nu/(nu-2)

set.seed(7114)

df <- 14

td <- rt(10^4, df = df)

hist(td)

summary(td)

var(td)

# 9. Chi квадрат разпределение

# Chi квадрат разпределението е непрекъснато разпределение, което представлява сума на k

# на брой независими стандартно нормално разпределени величини. Използваме го за тестване на

# хипотези свързани с дисперсията и при определяне на доверителните интервали. Разпределението

# намира приложение и при изследването на категорийните модели и по-точно до колко прогнозите

# на един модел съответстват на реалните стойности.

# Разпределенеито има един параметър (k), показващ степените на свобода.

# Очакването на Chi квадрат е k, а дисперсията - 2\*k

set.seed(11478)

N <- 1000

rch <- rchisq(n = N, df = 30)

hist(rch)

summary(rch)

var(rch)

# 10. Хипергеометрично разпределение

# Това е дискретно разпределение, което описва вероятността от k успеха в n на брой извадка,

# без замествания, взета от крайна популация с размер N и съдържаща K на брой успеха. Разпределението

# намира приложение при изследването дали една популация е overrepresented или underrepresented.

# Функцията на масата е P(x) = C(K, k)\*C((N - K), (n - k)) / C(N, n)

# Очакването е n\*K/N, а вариацията - n\*(K/N)\*((N-K)/N)\*((N-n)/(N-1))

# 11. Boostrap

# Booostrap е метод, който представлява създаване на извадка с големина, равна на

# големината на вектора X и всеки елемент от X може да участва пвоече от веднъж в новата извадка.

data(faithful)

names(faithful)

eruptions <- faithful[, "eruptions"]

sample(eruptions, 10, replace = TRUE)

par(mfrow = c(1, 2))

hist(eruptions, breaks = 25)

hist(sample(eruptions, length(eruptions), replace = TRUE), breaks = 25)

par(mfrow = c(1, 1))

# Едно от приложенията на bootstrap метода е при определянето на локацията на разпределение с тежки опашки и/или изразена асиметрия. Целта е да се включат в анализа и наблюдения, които се считат за "outlier"-и.

# Асиметрия

# За да видим дали имаме асиметрия в разпределението, ще изчислим статистиката skewness. В R има доста пакети, които предлагат тази опция ("DistributionUtils", "fBasics", "moments", "e1071"), но за целта ще използваме наша собствена

Skewness <- function(x) {

x\_centred <- x - mean(x)

n <- length(x\_centred)

y <- sqrt(n)\*sum(x\_centred^3) / (sum(x\_centred^2)^(3/2))

list(estim = y\*((1 - 1/n))^(3/2), se = sqrt((6\*n\*(n-1)) / (n-2) / (n+1) / (n+3)))

}

N <- 1000

set.seed(4741)

x1 <- -rexp(N, rate = 1/3); x1\_skewness <- Skewness(x1)

x3 <- rgamma(N, shape = 2, rate = 1/6); x3\_skewness <- Skewness(x3)

x2 <- rnorm(N, mean = 4, sd = 3); x2\_skewness <- Skewness(x2)

par(mfrow = c(1, 3))

hist(x1, xlab = "Values", main = paste("Skewness:", round(x1\_skewness$estim, 3)), col = "red")

hist(x2, xlab = "Values", main = paste("Skewness:", round(x2\_skewness$estim, 3)), col = "forestgreen")

hist(x3, xlab = "Values", main = paste("Skewness:", round(x3\_skewness$estim, 3)), col = "blue")

par(mfrow = c(1, 1))

# Лесно се забелязва, че при отрицателна стойност на skewness имаме асиметрия, при която лявата опашка е по-дълга. И обратното - при положителна стойност имаме по-дълга дясна опашка

# Ако стойността е близка до 0, тогава нямаме доказана асиметрия в разпределението.

# - Но какво означава, стойност близка до 0-та, при положение, че в статистиката разлика от 10 може да бъде незначима, а разлика от 0.00001 да бъде?

# - Това е така и ето защо има и начин за определянето на значимостта ?. Най-лесно съотношението

# abs(estim/se) трябва да бъде по-малко от 2

x1\_skewness$estim / x1\_skewness$se

x2\_skewness$estim / x2\_skewness$se

x3\_skewness$estim / x3\_skewness$se

# Да се върнем към bootstrap алгоритъма за изчисляване на локацията на разпределението. Той се състои в следните стъпки:

# 1. Определяме броя (M) на извадките (извадките са с повторения и имат дължина, равна на вектора, от който сме взели извадката).

# 2. Изчисляваме средната стойност на всички M извадки и ги съхраняваме в нов вектор.

# 3. От централната гранична теорема (ЦГТ, CLT) следва, че този вектор е с нормално разпределение

# 4. Взимаме средната стойност на този вектор го приемаме за локация на разпределението, а с помощта

# на стандартното отклонение, можем да определоим доверителни интервали

> N <- 400

> set.seed(9504)

> X <- rgamma(n = N, shape = 2, rate = 1/20)

> hist(X)

> summary(X)

Min. 1st Qu. Median Mean 3rd Qu. Max.

0.8918 17.8385 32.2251 39.7475 53.5784 170.3280

> mean(X, trim = 0.05)

[1] 37.18985

M <- 500

Mat <- matrix(nrow = M, ncol = length(X))

A screenshot of a cell phone

Description automatically generated

# Създаваме матрица, която има M на брой редове и length(X) брой колони

# В тази матрица, по редове ще съхраняваме извадките

set.seed(10)

for(row\_index in 1:nrow(Mat)) {

Mat[row\_index, ] <- sample(x = X, size = length(X), replace = TRUE)

}

# С долния ред взичмаме всички средни стойности по редове

mu\_array <- apply(Mat, 1, FUN = mean)

par(mfrow = c(1, 2))

hist(mu\_array, main = "Histrogram of mu's distribution", xlab = "Values", col = "pink")

qqnorm(mu\_array); qqline(mu\_array)

par(mfrow = c(1, 1))

# Проверката на хипотези е важна част от анализа на данни. В сегашния случай ще проверим хипотезата, че разпределението е нормално. За целта ще използваме тест, който проверява хипотезата

shapiro.test(mu\_array)

# Стойността p-value > 0.05 => разпределението може да го приемем за нормално.

# Понеже не сме учили тестовете, към настоящия момент ще ви се наложи да ми се доверите.

# По-нататък в упражненията ще бъдат засегнати тестовете подробно.

summary(X)

mean(X, trim = 0.05)

(loc <- mean(mu\_array))

# Предимството на този тест е, че можем да определим някакви неблагоприятни стойности, които да използваме вместо реалното очакване.

# Така например, ако искам в 95% от случаите да съм познал средната си стойност, тогава взимам 5% квантил.

(a <- quantile(mu\_array, prob = 0.05))

hist(mu\_array, main = "Histrogram of mu's distribution", xlab = "Values", col = "orange")

abline(v = loc, lwd = 3, col = "blue")

abline(v = a, lwd = 3, lty = 3, col = "forestgreen")

----------------------------------------------- Упражнение 6 -----------------------------------------------------

ip <- installed.packages()[, 1] # показва инсталираните пакети

pfi <- setdiff(c("ggplot2", "ggpubr", "nortest"), ip)

# Форният ред показва кои пакети не са инсталирани

if(length(pfi) > 0) {

install.packages(pfi)

}

library(ggplot2)

library(ggpubr)

library(nortest) # Тестове за проверка на нормално разпределение

# Статистически заключения

# Статистическите заклюения, основаващи се на случайни извадки, позволяват значително да се намалят разходите за статистически изследвания на големи по обем съвкупности.

# Информацията, получена от извадките почти винаги удовлетворява потребностите на

# проучващите.

# Статистическите заключения имат две основни направления:

# - статистическо оценяване

# - проверка на хипотези

# И двете направления имат вероятностен характер и са свързани помежду си.

# 1. Статистическо оценяване

# 1.1. Точкови оценки

# Точковата оценка представлява отделна величнина, получена от данните на случайна извадка, която може да се доближава в различна степен до съответния параметър на популацията (генералната съвкупност). Примери за точкови оценки са оценката на локацията и вариацията.

set.seed(950411)

x <- rnorm(n = 200, 10, 10)

x\_sample <- sample(x, 30)

mean(x) # 10.357

mean(x\_sample) # 9.1

set.seed(620331)

y <- rbinom(n = 400, size = 1, prob = 0.3)

y\_sample <- sample(y, 50)

mean(y) # 0.3175

mean(y\_sample) # 0.34

# 1.2. Интервални оценки

# Всяка оценка, получена от случайна извавдка е обременена със случайна грешка. Основният недостатък на точковата оценка се състои в това, че не позволява да се формират изводи за размера на тази грешка, за точността на нейното изчисляване по отношение по отношение на обема на вариацията на разпределението ѝ. Тази информация се съдържа в интервалната оценка.

#

alpha <- 0.05 # ниво на съгласие

k <- qnorm(1 - alpha/2) # квантил

n <- length(x\_sample) # размер на извадката

mean(x\_sample) + k\*c(-1, 1)\*sd(x\_sample)/sqrt(n) # доверителен интервал

n <- length(y\_sample) # размер на извадката

mean(y\_sample) + k\*c(-1, 1)\*sd(y\_sample)/sqrt(n) # доверителен интервал

# 1.3. Обем на извадката

# Обемът на извадката е един от най-важните фактори за точността на тези оценки

mu <- mean(x)

d <- density(x)

ss <- c(34, 7, 21)

counter <- 0

par(mfrow = c(2, 2))

for(i in c(10, 30, 100)) {

counter <- counter + 1

plot(d, main = paste("Density plot - ", i, "obs"), xlab = "x", lwd = 2)

set.seed(ss[counter])

xx <- sample(x, i)

abline(v = mu)

abline(v = mean(xx) + k\*c(-1, 0, 1)\*sd(xx)/sqrt(i), col = "red", lwd = 1.5, lty = 2)

}

par(mfrow = c(1, 1))

# 2. Тестване на хипотези

# 2.1. Хипотези. Видове хипотези

# При статистическите хипотези се проверява правдоподобността на предварително

# формулирани предположения относно праметрите или вида на неизвестното разпределение в популцията. Заключенията, основаващи се на хипотезите имат вероятностен храктер.

# Проверката на хипотеза се извършва в няколко стъпки. Стартира (първа стъпка) с

# формулирането на две хипотези - нулева (H0) и алтернативна (H1). Двете хипотези са

# взаимоизключващи се.

# Съществуват три вида хипотези:

# - Двустранна: H0: параметър = C / H1: параметър != C

# - Едностранна (лявостранна): H0: параметър >= C / H1: параметър < C

# - Едностранна (дясностранна): H0: параметър <= C / H1: параметър > C

# Втората стъпка е да се избере нивото на съгласие (alpha). Това е вероятност, която

# определя зоната за отхвърляне на нулевата хипотеза. Alpha се определя предварително в съответствие с целите и здачите на изследването. Най-често нивото н съгласие е 0.05.

x <- seq(-4, 4, by = 0.01)

d <- dnorm(x)

alpha <- 0.05

rej <- paste0("Отхвърлям (alpha = ", alpha, ")")

criteria <- factor(rep("Не отхвърлям", length(x)), levels = c("Не отхвърлям", rej))

criteria[which(x < qnorm(alpha))] <- rej

hypothesis\_greater <- qplot(x, d, geom = c("path", "area"), fill = criteria, xlab = "Z",

ylab = "Плътност", main = "H0: Параметър > C") +

scale\_fill\_manual(values = c("darkgreen", "red"))

hypothesis\_greater

criteria <- factor(rep("Не отхвърлям", length(x)), levels = c("Не отхвърлям", rej))

criteria[which(x > qnorm(1 - alpha))] <- rej

hypothesis\_less <- qplot(x, d, geom = c("path", "area"), fill = criteria, xlab = "Z",

ylab = "Плътност", main = "H0: Параметър < C") +

scale\_fill\_manual(values = c("darkgreen", "red"))

hypothesis\_less

rej1 <- paste0("Отхвърлям (<) (alpha = ", alpha/2, ")")

rej2 <- paste0("Отхвърлям (>) (alpha = ", alpha/2, ")")

criteria <- factor(rep("Не отхвърлям", length(x)), levels = c("Не отхвърлям", rej1, rej2))

criteria[which(x > qnorm(1 - alpha/2))] <- rej2 #| x < qnorm(0.025))] <- "Отхвърлям"

criteria[which(x < qnorm(alpha/2))] <- rej1 #

hypothesis\_two\_sided <- qplot(x, d, geom = c("path", "area"), fill = criteria, xlab = "Z",

ylab = "Плътност", main = "H0: Параметър = C") +

scale\_fill\_manual(values = c("darkgreen", "red", "darkred"))

hypothesis\_two\_sided

# В червено са изобразени критичните области, при които нулевата хипотеза се отхвърля

# Нивото на съглсие alpha = 0.05

# Трета и четвърта стъпка са да се определят се емперичната характеристика и след това да се провери дали попада в критичната област. По-лесният вариант е да се види стойността

# на p-value (significance).

# Какво по-точно представлява p-value?

# Най-грубо казано, с подхода p-value първо оценяваме колко вероятно е емпиричнаата стойност, получена от статистическия тест при положение, че нулевата хипотеза е вярна.

# Критерият за взимане на решение дали да се отхвърли H0 включва сравнение натзи вероятност

# с определеното ниво на съгласие alpha.

# 2.2. Тестове за локция/очакване на разпределението

# Пример

# Георги (наскоро формиран баровец) казал на Гергана, че средното разстояние, което изминава топката за голф при негов удар е 247 метра. Естестено, Гергана (учила през живота си поне един курс по статистика) е скептична и му иска доказателство. Така не Георги му се наложило

# да направи 25 опита, които той стриктно си записал във вектора

golf\_driving\_distances <- c(239, 229, 223, 224, 267, 235, 264, 235, 239, 251, 200,

191, 254, 253, 238, 216, 256, 228, 247, 219, 245, 251, 235, 246, 266)

# Както не веднъж сме споменавали, оценките на статистиките биват параметрични и непараметрични,

# в зависимост от вида на разпределението, за което ги изчисляваме. Параметричната статистика се

# изпозлва при наличие на НОРМАЛНО разпределение или поне симетрично разпределение, за което нямаме

# голям брой екстремални стойности. Ето защо, първата задача е да изследваме вида на разпределението

# Как можем да проверим едно разпределение дали е нормално или не?

par(mfrow = c(2, 2))

qqnorm(golf\_driving\_distances); qqline(golf\_driving\_distances)

d <- density(golf\_driving\_distances)

hist(golf\_driving\_distances, main = "Хистограма", col = "red", xlab = "Golf driving distances",

prob = T, ylim = c(0, max(d$y)))

lines(d, lw = 2)

x\_axis <- seq(0.9\*min(golf\_driving\_distances), 1.11\*max(golf\_driving\_distances), length = 300)

y\_axis <- dnorm(x\_axis, mean = mean(golf\_driving\_distances), sd = sd(golf\_driving\_distances))

lines(x\_axis, y\_axis, col = "blue", lw = 2)

boxplot(golf\_driving\_distances, horizontal = TRUE)

par(mfrow = c(1, 1))

ggqqplot(golf\_driving\_distances) # Друг начин за Q-Q plot

shapiro.test(golf\_driving\_distances)

# Нулевата хипотеза на теста (H0) е, че разпределението е нормално

# Стойността на p-value = 0.4157 => не можем да отхвърлим H0 =>

# приемаме, че разпределението e нормално

gdd <- golf\_driving\_distances

y <- rnorm(n = length(gdd), mean = mean(gdd), sd = sd(gdd))

ks.test(x = golf\_driving\_distances, y = y)

ks.test(x = scale(golf\_driving\_distances), y = "pnorm")

ad.test(x = golf\_driving\_distances)

# Тестовете и графиките показват, че разпределението е нормално. Следователно най-добре е

# да използваме параметрични тестове, т.е. student t тест

# 2.1.1. Параметрични тестове за една извадка

# H0: mean(x) = 247

# H1: mean(x) != 247

# Определяме ниво на съгласие alpha = 0.05

# x - приема вектор (задължителен параметър)

# y - приема вектор (не е задължителен)

# alternative - отговаря за типа на хипотезата и приема стойностите c("two.sided", "less", "greater")

# mu - константа, с която искаме да тестваме нулевата хипотеза

t.test(x = golf\_driving\_distances, mu = 247, alternative = "two.sided")

# Стойността на p-value < alpha => отхвърляме нулевата хипотеза H0. Тоест средната

# не е равна на 247 метра.

# Всички t тестове ни показват и доверителните интервали на очакването. Ако проверяваната

# стойност (mu) е извън този доверителен интервал, то отхвърляме H0 в полза на H1.

# Можем да порменяме големината на доверителните интервали с помощта на параметъра

# conf.level, където посочваме с каква вероятност искаме да присъства очакването в него

t.test(x = golf\_driving\_distances, mu = 247, alternative = "two.sided",

conf.level = 0.9)

# Пример 2

# Службата за вътрешни приходи (IRS) публикува данни за федералните данъчни декларации за

# доходите на физическите лица. Извадка от 12 лица от последната година показа коригираните

# брутни доходи в хиляди долари, които са записани във вектора

incomes <- c(9.7, 93.1, 33.0, 21.2, 81.4, 51.1, 43.5, 10.6, 12.8, 7.8, 18.1, 12.7)

# Искаме да проверим дали физическите лица получават годишно поне 20 000 долара?

qqnorm(incomes); qqline(incomes)

# От Q-Q plot-а се вижда, че данните не са нормално разпределени. Ето защо ще използваме

# непараметрични тестове

# 2.1.2. Непараметрични тестове за една извадка

# Непараметричният еквивалент на Student t тест е Wilcoxon signed rank test

# H0: E[x] = 20

# H1: E[x] > 20

wilcox.test(x = incomes, alternative = "greater", conf.int = TRUE, mu = 20)

# Стойността на p-value e 0.19 > alpha = 0.05 => не можем да отхвърлим H0. Доверителният

# интервал съдържа стойността 20 (14.35, Inf)

# Параметрите в Wilcoxon теста са сходни с тези на Student t тест. Единствената разлика е

# параметърът conf.int, който отговаря за показването на доверителния интервал.

# Пример 3

# Американската асоциация на университетските преподаватели (AAUP) провежда проучвания

# за заплатите на професори от колежи и публикува резултатите си в годишния доклад на AAUP

# за икономическото състояние на професията. Да предположим, че искаме да решат дали

# средните заплати на преподавателите в частни и публични институции са различни. Резултатите

# са представени във векторите по-долу

private\_institutions <- c(87.3, 75.9, 108.8, 83.9, 56.6, 99.2, 54.9, 73.1, 90.6, 89.3, 84.9,

84.4, 129.3, 98.8, 148.1, 132.4, 75.0, 98.2, 106.3, 131.5, 41.4,

115.6, 60.6, 64.6, 59.9, 105.4, 74.6, 82.0, 87.2, 45.1, 116.6,

106.7, 66.0, 99.6, 53.0)

public\_institutions <- c(49.9, 105.7, 116.1, 40.3, 123.1, 79.3, 72.5, 57.1, 50.7, 69.9, 40.1,

71.7, 73.9, 92.5, 99.9, 95.1, 57.9, 97.5, 44.9, 31.5, 49.5, 55.9,

66.9, 56.9, 75.9, 103.9, 60.3, 80.1, 89.7, 86.7)

# Първо ще започнем с изследването дали разпределенията са нормално разпределени. Ако и

# при два вектора имаме нормални разпределения, то ще изпозлваме параметрична статистика.

# Но, ако поне за единия вектор разпределенеито не е нормално, тогава е по-удачно да се спрем

# на непараметрични тестове.

shapiro.test(private\_institutions)

shapiro.test(public\_institutions)

# Минималната стойност на p-value за двата вектора е 0.6798 > alpha = 0.05 => разпределенията и

# на двата вектора ги приемаме за нормални.

# 2.1.3. Параметрични тестове за две извадки - Indipendent Two Sample t test и

# Welch Two sample t test

# Имаме два параметрични теста за проверка на локацията на две извавдки. Разликата между двата

# теста е предположението, че вариациите на двете извадки са с равни вариации (Independent Two

# Sample t test) или че не са - Welch Two Sample t test.

# Independent Two Sample t test е по-точен от Welch Two Sample t test

# Тест за сравняване на вариациите на две извадки от нормално разпределена популация

var.test(x = private\_institutions, y = public\_institutions)

# Нулевата хипотеза H0 е, че двете извадки имат равна вариация.

# В нашия случай, стойността на p-value = 0.6253 и следователно

# H0: mean(x) - mean(y) = 0

# H1: mean(x) - mean(y) != 0

t.test(x = private\_institutions, y = public\_institutions, var.equal = TRUE)

t.test(x = private\_institutions, y = public\_institutions)

# И двата теста отхвърлят нулевата хипотеза, че имаме равенство между средните стойности на

# двете извадки (p-value = 0.0196 и p-value = 0.0188). Тоест съществува статистически значима разлика между годишните заплащания на професорите в частните и публичните колежи. Разликата е в полза на частните колежи.

# Доверителният интервал е построен върху разликата от средните стойности на двете извавaдки претеглена сума на вариациите.

# Пример 4

data("mtcars")

# Искаме да изследваме дали средната мощност на колата, измерена в конски сили hp се различава за различните трансмисии. Данните са взети от "mtcars".

nortest::ad.test(mtcars$hp[which(mtcars$am == 0)])

nortest::ad.test(mtcars$hp[which(mtcars$am == 1)])

# Тестът за нормалност на разпределението отхвърля H0 при ръчните скорости (p-value = 0.00149).

# Следователно ще използваме теста на Wilcoxon за две извадки

# 2.1.4. Непараметрични тестове за две извадки

# H0: E(x) - E(y) = 0

# H1: E(x) - E(y) != 0

wilcox.test(mpg ~ am, data = mtcars, conf.int = TRUE, exact = FALSE)

# Стойността на p-value за теста е 0.001871 < 0.05 = alpha => Отхвърляме H0. Тоест

# Съществува статистически значима разлика между очакваните мощности при колите с ръчна и автоматична трансмисии. По-мощни са колите с ръчна трансмисия.

# Доверителният интервал е построен по-много интересна формула, която няма да я обясняваме, но я има :). Достатъчно е да знаем, че разликата (mu = 0) не попада в интервала.

install.packages("gplots")

library(gplots)

# Изследване на локациите на разпределеняита при повече от две групи

# Пример

# Взета е извадка от месечни наеми на апартаметни в различни региони в САЩ (в долари)

Northeast <- c(1005, 898, 948, 1181, 1244)

Midwest <- c(870, 748, 699, 814, 721, 606)

South <- c(891, 630, 861, 1036)

West <- c(1025, 1012, 1090, 926, 1269)

# Искаме да изследваме дали между някой от регионите съществува значима

# разлика в очакването за цените в наемите.

# Данните трябва да ги обединим в един data frame

rent\_data <- data.frame(rent = c(Northeast, Midwest, South, West),

region = c(rep("Northeast", length(Northeast)),

rep("Midwest", length(Midwest)),

rep("South", length(South)),

rep("West", length(West))))

# В предишното упражнение използвахме Student t тест и Wilcoxon тест, за да изследваме средните стойности и медианите на една извадката или между две групи от наблюдения. За изследването на разлика между локациите на повече от две групи трябва да използваме One-way ANOVA (параметричен тест) или Kruskal тест (непараметричния еквавалент на ANOVA). Нека имаме n на брой вектора X1, X2, ..., Xn. Тогава имаме нулевата хипотеза Н0: E[X1] = E[X2] = ... = E[Xn] и алтернатива H1: поне при една от двойките E[Xi] != E[Xj] за i != j.

# Като всеки един параметричне тест и One-way ANOVA има своите първоначални предположения, които, ако бъдат нарушени, то трябва да използваме Kruskal тест

# Предположения

# 1. За всяка една група, разпределението на стойностите трябва да бъде нормално разпределена

# 2. Статистически еднаква дисперсия при всички групи (хомогенност на дисперсиите).

# Ще започнем с изследване на разпределението на данните по различните

# групи. Най-лесно проверката ще стане с помощта на функцията aggregate.

# Като фунцкия за агрегация ще използваме теста на Shapiro-wilk

aggregate(rent ~ region, data = rent\_data, FUN = function(x) {shapiro.test(x)$p.value})

# Минималната стойност p-value за четирите групи е 0.456 > 0.05 = alpha =>

# не можем да отхвърлим H0 => приемаме, че и четирите групи са нормално разпределени

# Хомогенността на дисперсиите ще проверим с помощта на теста на Бarlett,

# с нулева хипотеза за равемство на дисперсиите между различните групи

bartlett.test(rent ~ region, data = rent\_data)

# P-value = 0.6957 > 0.05 = alpha => имаме статистически равни дисперсии

# One-way ANOVA

summary(rent\_anova <- aov(formula = rent ~ region, data = rent\_data))

# С помощта на фунцкията "aov" прилагаме One-way ANOVA. Функцията съдържа

# параметрите formula и data.

# Стойността на p-value = 0.0023 < 0.05 = alpha => отхвърляме H0 в полза на H1 =>

# съществува статистически значима разлика поне в някоя от двойките.

# Остана да видим къде между кои групи са разликите. Това лесно става графично

# с помощта на функцията plotmeans()

plotmeans(formula = rent ~ region, data = rent\_data)

# Друга опция е използването на така наречените Post-hoc pairwise контрасти,

# които изследват взаимодействието на една група спрямо останалите.

# Съществуват различни методи за изследването им, но ние ще се спрем само на

# Tukey HSD. Върнатият резултат представлява тества на разликите между

# всички възможни две групи, където нулевата хипотеза е, че двете локации са

# статистически равни (или, че разликата им е = 0)

(tukey <- TukeyHSD(rent\_anova))

# Съществените разлики при групите се забелязват в последната колона

# "p adj" (p-value), където искаме стойността на p-value < alpha - нивото на съгласие

# Тоест групите между, които имаме разлика са (Northest, Midwest) и (West, Midwest)

plot(tukey) # Графично представяне на разликите между отделните групи

# Други методи за анализ на Post hoc pairwise са

pairwise.t.test(rent\_data$rent, rent\_data$region, p.adj = "bonf")

pairwise.t.test(rent\_data$rent, rent\_data$region, p.adj = "holm")

# ! Различните тестове, дават различни резултати при анализа. Ето защо е важно

# да се избере най-подходящия алгоритъм за конкретната задача.

# Горните два теста връщат директно стойността на p-value

# Пример

# изследване на връзката между месец в годината и средните стойности на озона

# за Ню Йорк.

data(airquality)

# Изследване за нормално разпределние в различните групи.

aggregate(Ozone ~ Month, data = airquality, FUN = function(x) {round(shapiro.test(x)$p.value, 3)})

# Имаме нарушение на условието за нормално разпределение на стойностите (Май и Септември)

kruskal.test(Ozone ~ Month, data = airquality)

# Стойността на p-value за теста е 6.901e-06 << alpha = 0.05 => съществува

# статистически значима разлика между групите.

# Post-hoc анализ за Kruskal-Wallis тест

pairwise.wilcox.test(airquality$Ozone, airquality$Month,

p.adjust.method = "BH", exact = FALSE)

# В получената табличка са записани стойностите на p-value при изследването на

# разликите между групите. Така статистически значима разлика получаваме при месеците

# (5, 7), (5, 8), (6, 7) и т.н.

----------------------------------------------- Упражнение 7 -----------------------------------------------------

# Линейни модели

# Линейна регресия

NN <- 300

set.seed(73391)

x1 <- round(runif(NN, 0, 5), 1)

x2 <- round(runif(NN, 2, 6), 1)

y <- 3 + 2.5\*x1 + rnorm(NN)

DF <- data.frame(x1, x2, y)

# Какво представлява линейната регресия

# Линейната регресия е статистически метод, който ни позволява да проучим и обобщим връзките между две множества от непрекъснати променливи - X и y:

# - в множеството X се намират обясняващите променливи (наречени още предиктори или независими променливи) и на върху тях се основават нашите прогнози;

# - в множеството y се съдържа една променлива (вектор), наричаща се зависима променлива

# променливаи или резултат, която искаме да прогнозираме.

# Ако приемем, че размерът на вектора y e N, a размерът на множеството X е N x p, то връзката между двете множества е y = b(0) + b(1)\*x(1) + ... + b(p)\*x(p) + error, където

b(0) е константа, а b(1), ..., b(p) са параметрите, които обясняват влиянието на X над y.

Линейната регресия ни позволява да оценим стойностите на тези p+1 коефцициенти.

Има няколко начина за намирането на тези коефициенти, но най-често използваният е OLS

# (метод на най-малките квадрати)

# В R, функцията за линейна регресия е lm()

# Два начина за извикване на линейна регресия. Първият, ако виждаме нужните ни променливите

# в средата на R. Лесно можем да проверим дали променливите са заредени в среда на R с функцията

# ls().

model1 <- lm(y ~ x1)

model1

rm(list = c("x1", "x2", "y"))

# Вторият начин е като посочим data frame-а или матрицата, който(която) съдържа необходимите

# променливи.

model2 <- lm(y ~ x1, data = DF)

model2

rm(list = "model2")

# Когато изследваме връзка между една зависима и една обясняваща променлива, тогава линейната

# регресия е едномерна (или проста). При наличието на повече предиктори (обясняващи променливи),

# тогава имаме многомерна линейна регресия.

# Горните два модела са пример за проста линейна регресия

# Многомерната линейна регресия ще бъде разгледана по-подробно по-нататък в курса.

# След като сме построили линееен модел, следващата стъпка е да проверим до колко този модел

# описва добре данни и какви са оценките на коефициенти му.

summary(model1)

# -------

# Хипотези и проверка на хипотези

# Накратко, статистическата хипотеза е предположение за параметър на извадката/популацията.

# Това предположение може да бъде вярно или невярно. Ето защо съществуват две взаимоизключващи се

# хипотези - нулева (H0) и алтернативна (H1).

# Имаме три типа хипотези:

# 1. H0: параметър = число, H1: параметър != число

# 2. H0: параметър <= число, H1: параметър > число

# 3. H0: параметър >= число, H1: параметър < число

# Проверката на хипотезите става с помощта на тестове. На база вида на теста, искаме да

# отхвърлим или не нулевата хипотеза H0. Дали нулевата хипотеза е отхвърлена се определя от

# стойност, наречена "p-value". Стандартно, една H0 се отхвърля при стойност на p-value < 0.05.

# При отхвърляне на нулева хипотеза, за вярна се приема алтернативната H1.

# -------

# Първо ще проверим дали коефициентите са статистически значими, тоест дали е необходимо да

# участват в анализа. За всеки един коефициент проверяваме хипотезата дали коефициентът е равен

# на 0 (b(i) ?= 0). За да бъде един коефициент значим, то трябва за него да отхвърлим горната

# хипотеза. Както беше споменато по-горе, за да се отхвърли H0, то стойността на p-vaue трябва да

# бъде по-малка от 0.05. Стойностите p-value се намират в колоната "Pr(>|t|)". Стойностите на

# p-value за двата параметъра е 2e-16 << 0,05 и следователно двата параметъра са статистически

# значими.

# Преди да продължим с изследването на регресията, нека да видим случай, когато коефициентите

# не са значими.

summary(model3 <- lm(y ~ x2, data = DF))

rm("model3")

# Да разгледаме оценките пред коефицнета x2. Оценката на коефициента е -0,1505. Но въпреки, че

# стойността му е различна от 0, то той е статистически незначим. Защо? Защото стойността на

# p-value e 0,414 > 0,05. Тоест, този коефициент може да отпадне от анализа.

# Следващата стъпка е да проверим до колко модела описва добре данните. За целта ще използваме

# статистиките "Multiple R-squared" или "Adjusted R-squared". Статистиката "Multiple R-squared"

# приема стойности в интервала [0-1]. Колкото тази статистика се приближава до единица, толкова

# моделът е по-добър. И обратното, колкото стойността на R2 клони към 0, толкова моделът не се

# справя с описването на данните. Моделите, които имат стойности за R2 под 0.5, ги приемаме за

# слаби.

# Препоръчително е да се използва обаче статистиката Adjusted R-squared, защото тя "наказва",

# когато използваме ненужни променливи. По принцип, тази статистика също приема стойности в

# интервала [0-1], но когато използваме само статистически незначими, тогава Adjusted R-squared

# може да приеме и отрицателни стойности.

summary(model1)

# Какво можем да кажем за model2? Стойността на Adjusted R2 е 0.9284. Тоест моделът описва

# много добре данните.

# - Какво представлява R2 и как можем да го изчислим?

# - За целта първо ще разгледаме начините да изчисляваме прогнози и остатъците (residuals).

# Регресионното уравнение придобива вида y = 3.027 + 2.247\*x1.

# Линейната регресия позволява не само да се оценят връзките между отделните обясняващи

# променливи и резулатата, но както споменахме по-горе, позволява да се правят прогнози. В R

# използваме функцията "predict" за прогнозиране. Функцията съдържа два основни параметъра object

# (построения модел) и newdata (данни, за които искаме да направим прогноза)

model1.predictions <- predict(object = model1, newdata = DF)

model1.predictions.alt <- model1$coefficients[1] + model1$coefficients[2]\*DF$x1 # Алтернативен начин

all(model1.predictions == model1.predictions.alt)

rm("model1.predictions.alt")

# Нека да видим на графика как изглеждат прогнозите спрямо реалните стойности

plot(model1.predictions, DF$y)

abline(a = 0, b = 1, col = "red", lwd = 2)

# От графиката се вижда, че прогнозите и реалните стойности се движат около ъглополовящата на

# първи квадрант (x = y)

# Остатъците са разликата между наблюдаваната стойност и направената прогноза. За целта ще

# използваме функцията residuals(). Параметърът object приема стойността на модела, за когото

# желаем да оценим остатъците.

res <- residuals(object = model1)

res.alt <- DF$y - model1.predictions

all(round(res, 10) == round(res.alt, 10))

rm("res.alt")

# Нека сега се върнм към Multiple R-squared. В своята същност R2 представлява

# 1 - съотношението на вариацията на остатъците и общата вариация. Колкото един модел е по-добър,

# толкова остатъците му следва да бъдат по-малки, а от там и вариацията им. Тъй като общата

# вариация е константа, то можем да използваме тази статистика при сравняването на моделите и

# избора на по-добрия.

summary(model1)$r.squared

1 - var(res)/var(DF$y)

# Условия

# Не на последно място, остава да се проверят дали линейанта регресия (а и всички останали

# линейни модели или ML алгоритми) отговарят на три необходими условия

# 1. Константна вариация на грешките (Хомоскедастичност)

# Това е най-важното условие при линейните модели. Целта на константната вариация е около

# регресионната линия да се изгради "тунел" и да може да се определи (с някаква вероятност), в

# какви граници се намира прогнозата. При нарушение на това условие, за определени интервали

# грешката ще бъде по-малка от очакваното, а за други - по-голяма. Проблемът е, че ако очакваме

# определени неблагоприятни сценарии, те може да се окажат още по-лоши.

# Това условие най-лесно се проверява графично. По остта X изобразяваме прогнозите, а по Y -

# остатъците

plot(model1.predictions, res)

abline(h = 1.96\*c(-1, 1)\*round(sd(res, 2)), col = "red", lty = 4)

# За да имаме хомоскедастичност, то остатъците трябва да бъдат разпръснати равномерно по

# цялата графика. Между двете червени

# линии хипотетично се намират 95% от остатъците.

# Примери за хетероскедастичност (неконстантна вариация на грешките)

sigmaFunction <- function(x) {

thresholds <- unname(quantile(x, prob = seq(0, 1, by = 0.1)))

thresholds[length(thresholds)] <- thresholds[length(thresholds)] + 0.001

findInterval(x, thresholds)

}

NN <- 400

set.seed(6335)

a <- 0.1; b <- 4

predictions <- 4 + 5\*runif(NN, a, b)

noise <- rnorm(NN, sd = 0.25)

SI <- sigmaFunction(predictions)

r <- cbind(SI, (11 - SI), (1 + 2\*abs(mean(SI) - SI)), (11 - (1 + 2\*abs(mean(SI) - SI))))\*noise

par(mfrow = c(2, 2))

for(i in 1:4) {

plot(predictions, r[, i], xlab = "Predictions", ylab = "Residuals")

#abline(h = 0, col = "red", lwd = 2)

abline(h = 1.96\*c(-1, 1)\*round(sd(r[, i]), 2), col = "red", lty = 4)

}

par(mfrow = c(1, 1))

rm(list = c("a", "b", "i", "noise", "predictions", "r", "SI", "sigmaFunction"))

# На графиката са показани четирите основни типа хетероскедастичност. Между двете червени

# линии хипотетично се намират 95% от остатъците.

# 2. Липса на автокорелация на грешките

# Следващото важно условие е между остатъците да нямаме наличие на автокорелация. Тоест

# всяка следваща грешка да не зависи от предходната грешка. Най-лесно е да проверим с теста

# на Durbin-Watson. Този тест се намира в пакета "lmtest", който трябва да го инсталираме и

# заредим.

# Функцията за теста на Durbin-Watson е dwtest(). Теста приема като параметър самия модел.

# Нулевата хипотеза e, че не съществува автокорелация. Тоест, целта ни е ДА НЕ отхвърлим H0.

install.packages("lmtest")

library(lmtest)

dwtest(model1)

# Стойността на p-value е 0.767 > 0.05. Следователно няма да отхвърлим хипотезата.

# Следователно нямаме автокорелация при грешките.

# Пример за автокорелация при грешките на линеен модел

NN <- 300

set.seed(6621)

x1 <- runif(NN, 1, 5)

noise <- rnorm(NN)

rho <- 0.8

for(i in 2:NN) { noise[i] <- rho\*noise[i-1] + sqrt(1 - rho^2)\*noise[i] }

# Задаваме автокорелация равна на 0.8

y1 <- 3 + 2\*x1 + noise

model4 <- lm(y1 ~ x1)

summary(model4)

dwtest(model4)

# Стойността на p-value e 0, следователно имаме наличие на автокорелация. Както и очаквахме

rm(list = c("i", "model4", "noise", "rho", "x1", "y1"))

# 3. нормално разпределение на грешките

# Последното условие е грешката да има нормално разпределение. Когато това условие е

# изпълнено, тогава имаме най-добрите оценки на коефициентите на линейната регресия. Проверката

# на това условие става с помощта на теста на Shapiro-Wilk (H0: нормално разпределение) и

# Q-Q plot.

shapiro.test(res)

# Стойността на p-value e 0.632 => грешката е нормално разпределена.

qqnorm(res); qqline(res)

# На тази графика търсим за тежки опашки (стойностите в краищата са на голямо разстояние от

# линията). Както се вижда, няма тежки опашки