Introduction to R

Stefan Richter

25. April 2024

Remarks

- Each input yielda an output in form of an object (often: a vector)
- *R* is fast with vector and matrix operations but not with loops $\rightarrow$ for fast calculations, they have to be written in form of vector and matrix operations
- *R* is not an computer algebra system
- *R* is mainly for doing statistics, many commands from statistics are already implements and additional packets for more commands are available
- During executing R, one works in a specific working directory. This can be changed ('Tools' / 'Verschiedenes'). It is important to know where you are if you want to read in data.
- Commands in the console are executed with 'Enter', it is not needed to have ';' at the end of your commands.
- Typically, spaces do not matter!

Basic operations

- $2 + 5$, $5 * 5$, $5 + 5$, 5^2, $\log(5) = \log_e(5)$, $\sin(4)$, $\sqrt{10}$ (Wurzel)

Variables

- $a = 4$: Variable a gets the value 4.
- a: Get the value of the variable a
- $b = 5$
- $a + 4$
- $c = a - b$
- $a = a + 10$

Vectors

Standard command: $c(...)$: concatenates elements in ... to one vector.

- $b1 = c(2, 3, 4)$
- $b2 = c(4, 5, 6)$
- $b3 = c(b1, b2)$

Creation of standard vectors: $a : b$ produces vector $(a, a + 1, a + 2, ..., b - 1, b)$.

- $a = 7 : 12$

Create vectors by repetition: rep(number/vector,howoften)

- rep(0,5)

access elements of a vector x: $x[...]$ yields elements No. ... of x

- $a[2]$
- $a[c(2, 3)]$
- $a[4 : 6]$.
- $a[-2]$.

length of a vector x: length(x)

- length(a)

Computations with vectors

Standard rule: all is component-wise!

- $d = 1:5$
- $d - 5$
- $(d-5)^2$
- $n = 100, \quad (1:n)/n$
- $a = c(5,3,4,2,1)$
- $a + d$

summation of elements of x: $sum(x)$

- $sum(1:10)$

Task: Calculate with R:

- the length and sum of a vector $(1,2,3,4,5,6)^T$.
- $\frac{1}{n}\sum_{i=1}^{n}(i^3+i^2)$ for $n = 100$.

Matrices

Creation of a matrix:

- x = matrix([elements as vector], nrow=[number rows], ncol=[number columns])

Examples:

- x = matrix(c(1,2,3,4), nrow=2, ncol=2)
- x = matrix(c(1,2,3,4), nrow=2, ncol=2, byrow=FALSE)
- x = matrix(c(1,2,3,4,5,6), nrow=2, ncol=3)
- y = matrix(0, nrow=5,ncol=5)
- z = diag(1,4)

Access to elements of a matrix x by $x[zeile, spalte]$:

- x[2,2]
- x[1:2,3]
- x[1:2,2:3]

Matrix operations

Transpose of a matrix x: t(x)

- t(x)

Standard rule: All operations work component-wise!

- 5 * x
- x^2

Matrix multiplication: x %*% y, Matrix inversion: solve(x)

- x = matrix(c(1,2,3,4),nrow=2)
- y = matrix(c(5,6,7,8),nrow=2)
- x*y
- x %*% y
- solve(x)

Task: Compute with *R*:

- $\begin{pmatrix} 2 & 3 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 7 & 2 \\ 5 & 1 \end{pmatrix}$, $7 * \begin{pmatrix} 2 & 3 \\ 1 & 1 \end{pmatrix}$, $\det \begin{pmatrix} 2 & 3 \\ 1 & 1 \end{pmatrix}$, $\begin{pmatrix} 2 & 3 \\ 1 & 1 \end{pmatrix}^{-1}$

Random numbers

Create n i.i.d. $N(\mu, \sigma^2)$-distributed random variables: rnorm(n,μ,σ)
Create n i.i.d. $U[a, b]$-uniform distributed random variables: runif(n,a,b)

- n = 100
- rnorm(n,0,1)
- runif(n,0,1)

Random sampling of n elements of a given vector x: sample(x, size=n, replace=FALSE)

- x = 1:100
- sample(x,10,replace=FALSE)
- sample(x,100,replace=TRUE)

Task: For $i = 1, ..., n = 100$ let $x_i = \frac{i}{n}$, $\varepsilon_i \overset{iid}{\sim} N(0, 1)$ and

$$y_i = \beta_1 * x_i + \beta_2 * x_i^2 + \varepsilon_i,$$

where $\beta_1 = 5$ and $\beta_2 = 7$. Create the vectors $x = (x_1, ..., x_n)^T$, $y = (y_1, ..., y_n)^T$.

Plots

Plots of points (x,y) of vectors x,y: plot(x,y)

- plot(x,y)

Additional options:
plot(x,y,type="p"/"l"/"o", col="color(english)")

- plot(x,y,type="l",col="red")
- plot(x,y,type="o",col="blue")

Add additional lines given by coordinates x,y to the plot:
lines(x,y, col="color(englisch)",lty=1/2/...)

- yneu = beta1*x + beta2*x^2
- lines(x,yneu,col="red")

Task:

- Plot the function $g(x) = sin(x)$ on the interval $[-5, 5]$ with points / and with lines.
- Plot 100 randomly created points in $[0, 1]^2$ (use uniform distribution for the x- and y-components)

Use external R-scripts

2 possibilities:

- Menu 'Ablage' / 'File' → Execute R Script
- source("path/to/skript.r",chdir=TRUE)

Create a file 'skript.r' with an editor (or e.g. Text editor / Notepad)

Loops / If-Else

With a vector x:
Syntax: for (i in x) { repeat this for each element i of x}

- for (i in 1:10) { print(i) }
- x = 0
- y = 1:10
- for (i in 1:10) { x = x + y[i] }
- x

Syntax: if (Condition) { *Then do that* } else { *else do that* }

- x = 5
- if (x > 3) { print("yes") } else { print("no") }
- if (x == 5) { print("x is 5") } else { print("x is not 5")}

Task: Loop through the numbers from 1 to 100 and only print out the prime numbers
(Command a%%b yields the remainder by division a by b)

Lists

Lists save objects of different types and give a name to the corresponding objects

- x = list(number = 5, text = "hallo", datapoints = c(1,2,3,4))

Access to elements of a list x: x[[...]] with ... being the number of the element OR x$name with 'name' the name of the object

- x[[2]]
- x$zahl
- x$daten

Read-In of external data

→ Download: File 'test.csv' from ILIAS

Command to read-in data: read.table or more specific: read.csv:

read.csv(ßource/file name", header = TRUE, sep = "separator of different columns", dec = "decimal point")

Result is an object of type 'Data Frame' - access to elements like a matrix, but each column can have a different element type

- x = read.csv("test.csv", header = TRUE, sep = ";", dec = ",")
- names(x)
- x[1,]
- x[1:20,]
- x[1:20,5]
- x$ARZTBES[1:100]

Read-in external data

as.matrix(dataframe): change dataframe to matrix.

- data = read.csv("test.csv", header = TRUE, sep = ";", dec = ",")
- n = length(data[,1])
- d = length(data[1,])-1
- x = as.matrix(data[,1:9])
- x = cbind(rep(1,n),x)
- y = as.matrix(data[,10])
- beta = solve(t(x) %*% x) %*% t(x) %*% y
- beta
- round(beta,2)

beta ist the LS-estimator!

R functions

Apply a linear regression (with additional component $X_{0d} = 1$) to data frame a dataframe 'data' aus:
res = lm("y ~ .", data=data)
coefficients = coef(res)

- data = read.csv("test.csv", header = TRUE, sep = ";", dec = ",")
- res = lm("ZUFRIEDENHEIT ~ .", data=data)
- beta = coef(erg)

beta is the LS-estimator!

Calculate the ridge estimator for different λ

- lambda = 0.7^(0:30)
- beta = matrix(0,nrow=length(lambda),ncol=d)
- unitmatrix = diag(c(0,rep(1,d-1)),d)
- for (i in 1:length(lambda)) {
- beta[i,] = (solve(t(x) %*% x) + lambda*n*unitmatrix) %*% t(x) %*% y
- }

Task: Plot the profile lines of a ridge estimator based on the data in the above matrix beta!