P01 – Basics

12. April 2021

Contents

Erstelle pro Aufgabe eine R-Code-Datei und benenne diese nach dem Schema `P<Woche>-<Aufgabe>.R` also hier `P01-1.R`, `P01-2.R`, `P01-3.R` und `P01-4.R` Schreibe den Code zur Lösung einer Aufgabe in die jeweilige Datei.

Es ist erlaubt (aber nicht verpflichtend) zu zweit abzugeben. Abgaben in Gruppen von drei oder mehr Personen sind nicht erlaubt. Diese Gruppierung gilt nur für die Abgabe der Programmierprobleme, nicht für die Live-Übungen.

Bei Abgaben zu zweit gibt nur eine der beiden Person ab. Dabei müssen in **jeder** abgegebenen Datei in der **ersten Zeile** als Kommentar **beide** Namen stehen also zB

```
# Ada Lovelace, Charles Babbage

1+1
# ...
```

Die Abgabe der einzelnen Dateien (kein Archiv wie `.zip`) erfolgt über Moodle im Element namens `P01`. Die Abgabe muss bis spätestens Sonntag, 18. April 2021, 23:59 erfolgen.

1 atomare Vektoren (10 Punkte)

1. Erzeuge eine atomaren Vektor $x = (1.3, \frac{1}{2}, 42, -8 \cdot 10^{-5})$.
2. Invertiere die Reihenfolge der Elemente des Vektors x, ohne ihn neu zu erzeugen.
3. Setze alle Werte von x, die kleiner als 1 sind, auf -2 mit einem möglichst kurzen Befehl.
4. Berechne $\sum_{i=1}^{4} x_i^2$.
5. Erzeuge den atomaren Vektor $q = (1, 0, 3, 0, 5, 0, ..., 97, 0, 99, 0)$.

2 Euklidische Distanz und Listen (10 Punkte)

Schreibe eine Funktion `eucl_dist()` mit zwei Argumenten `x` und `y`, die die euklidische Distanz $\|x - y\| = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$ für zwei Vektoren der Länge $n \in \mathbb{N}$ berechnet. Dieser Wert soll als zweiter Eintrag einer Liste zurückgegeben werden. Der erste Eintrag soll den Text `"ok"` enthalten, falls die übergebenen Vektoren die selbe Länge haben. Ist dies nicht der Fall, soll der Text stattdessen `"lengths differ: A vs B"` lauten, wobei an Stelle von `A` und `B` die entsprechenden Längen stehen sollen. Der zweite Listeneintrag ist dann auf `NaN` zu setzen.

```r
x <- c(1, 2)
y <- c(4, -2)
eucl_dist(x, y)
## [[1]]
## [1] "ok"
##
## [[2]]
## [1] 5
eucl_dist(1:4, 1:5)
## [[1]]
## [1] "lengths differ: 4 vs 5"
##
## [[2]]
## [1] NaN
```

Hinweis:

```r
x <- NaN
x
## [1] NaN

txt <- sprintf("bla %d blub %d!!", 42, -15)
txt
## [1] "bla 42 blub -15!!"
```

3 Primzahlen (20 Punkte)

Schreibe eine Funktion `find_primes()` mit einem Argument `count`, die einen atomaren Vektor der ersten `count` Primzahlen zurückgibt.

```r
find_primes(5)
## [1]  2  3  5  7 11
find_primes(10)
##  [1]  2  3  5  7 11 13 17 19 23 29
```

4 Matrizen und lineare Regression (20 Punkte)

Wir erzeugen Daten $(x_1, y_1), \ldots, (x_n, y_n)$ aus einem linearen Modell, dh $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$, $i = 1, \ldots, n$. Hierbei sind $\beta_0, \beta_1 \in \mathbb{R}$, $\epsilon_i \overset{iid}{\sim} \mathcal{N}(0, \frac{1}{4})$ sowie $x_i \overset{iid}{\sim} \text{Unif}(0, 1)$, wobei $(\epsilon_i)_{i=1,\ldots,n}$, $(x_i)_{i=1,\ldots,n}$ unabhängig sind (**iid** steht für *independent and identically distributed* also *unabhängig und identisch verteilt*).

Wir notieren

$$X := \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}, \quad y := \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \quad \epsilon := \begin{pmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{pmatrix}, \quad \beta := \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix}.$$

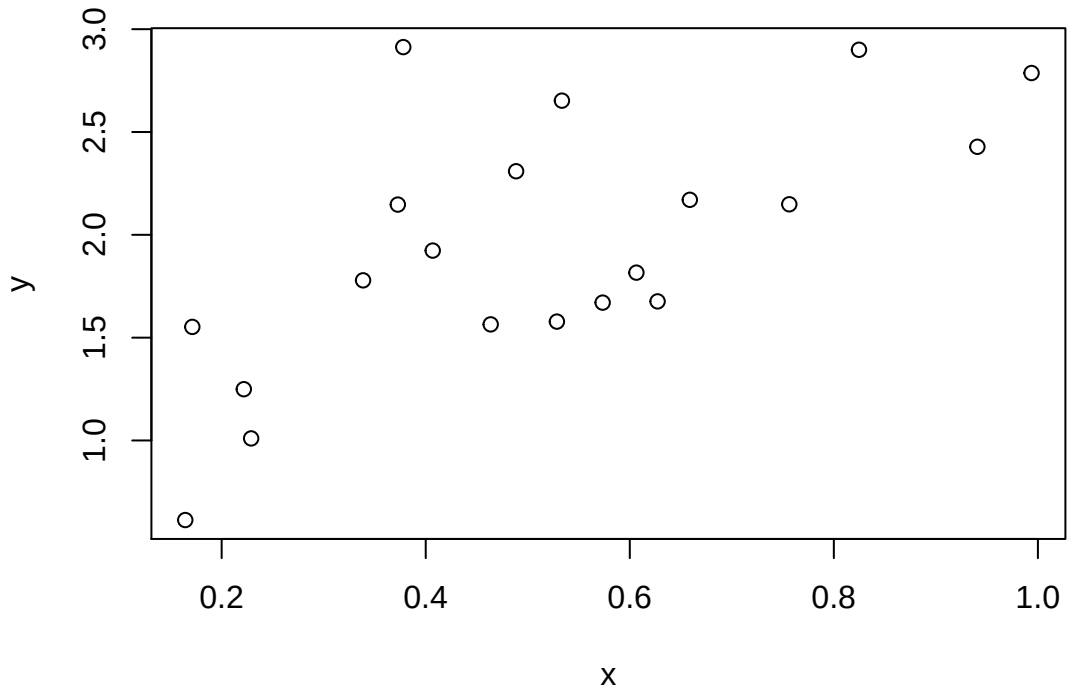
Dann gilt $y = X\beta + \epsilon$.

```r
# create data ----------------------------------------------------------
beta <- c(1, 2)
n <- 20
x <- runif(n)
X <- matrix(c(rep(1, n), x), ncol=2)
noise <- rnorm(n, mean=0, sd=0.5) # normally distributed noise
```

```r
# sd = 0.5  =>  var = 0.5^2 = 0.25
y <- X %*% beta + noise # data from linear model with noise

# show created objects ------------------------------------------------
beta
## [1] 1 2
str(x) # atomic vector length 20
##  num [1:20] 0.222 0.529 0.994 0.825 0.373 ...
str(X) # 20x2 matrix
##  num [1:20, 1:2] 1 1 1 1 1 1 1 1 1 1 ...
str(noise) # atomic vector length 20
##  num [1:20] -0.194 -0.479 -0.201 0.251 0.402 ...
str(y) # 20x1 matrix (~ vector length 20)
##  num [1:20, 1] 1.25 1.58 2.79 2.9 2.15 ...

# plot data -----------------------------------------------------------
plot(x, y)
```



Wir betrachten nun β als unbekannt und möchten es aus den Daten $(x_i, y_i)_{i=1,\dots,n}$ schätzen. Dazu nutzen wir den Kleinste-Quadrate-Schätzer

$$\hat{\beta} \in \arg\min_{b \in \mathbb{R}^2} \sum_{i=1}^{n} (y_i - b_0 + b_1 x_i)^2 = \arg\min_{b \in \mathbb{R}^2} \|y - Xb\|^2 .$$

Falls $X^t X$ invertierbar ist, gilt $\hat{\beta} = (X^t X)^{-1} X^t y$.

Schreibe eine Funktion `least_squares_estimator()` mit zwei Argumenten `X` und `y`, die $\hat{\beta} = (X^t X)^{-1} X^t y$

berechnet (Wir gehen hier fahrlässigerweise davon aus, dass X^tX invertierbar ist). Führe die Funktion mit den angegebenen Daten `X` und `y` aus, um $\hat{\beta}$ zu berechnen. Zeichne einen Plot, der die Datenpunkte sowie die Geraden $x \mapsto \beta_0 + \beta_1 x$ und $x \mapsto \hat{\beta}_0 + \hat{\beta}_1 x$ enthält. Um die Geraden unterscheiden zu können, sollen diese in unterschiedlichen Farben gezeichnet werden.

Hinweis: `?solve`