Eva-Maria Maier
Elena Ivanova

Chair of Statistics
Humboldt-Universität zu Berlin

Statistical Programming Languages 2020/21

Take-home exam

This take-home exam consists of two parts:

1. **Package Report** [60 points]

   Prepare a presentation and write a report on the `R` package(s) `glm2`. Only the report is graded. However, the presentation is mandatory to pass the course. Respect the guidelines of this course. In particular, the report

   - has to be written in R Markdown[1] (Rmd file around 10,000 characters without white spaces),
   - should enable someone without any prior knowledge of the package to acquire basic knowledge of its usage,
   - should contain examples on how to use the package, illustrated on an appropriate self-chosen data set,
   - has to comply with scientific standards, i.e., write in your own words and specify your sources.

   You have to submit the following files (at least two) related to this part:

   - the report as Rmd and pdf file, named in the scheme "SPL2021_*package*_*surname*_*name*.Rmd" and "SPL2021_*package*_*surname*_*name*.pdf", respectively, where *package* is replaced by the name of your assigned package and *surname* and *name* are replaced by your surname and name. The report must contain your complete name and student ID (Matrikelnummer).
   - all external files (e.g. data files) which are required to run your Rmd file

2. **Programming task** [115 points]

   Solve the exercises below. Use the file SPL2021_surname_name.R for your solutions and rename it accordingly. Furthermore, fill in your names and student ID (Matrikelnummer) at the designated place in the file. Requirements for the `R` code:

   - must solve the exercises without errors or warnings (except for intended ones) when executed from top to bottom (`R` session will be restarted before every exercise)
   - should only include one solution per (sub)task; in case of several solutions only the first one will be graded
   - should be comprehensible and efficient
   - should include reasonable comments; also to structure the code (subtasks) and to answer specific questions
   - must not use any packages (unless demanded in a specific exercise); violation will result in 0 points for the respective task(s)!
   - should only include pure functions (exception: functions that are only used as argument of an `apply()` family function may depend on variables defined outside of them)

You have to submit your solutions for both parts of this take-home exam as one zip file until **31 January 2021, 11.59 p.m.** via Moodle.

[1] You may use the following templates: presentation_template.zip and report_template.zip

Exercise 1:

Session Info [2 points]

Start a new `R` session and execute the following code:

```
> sink("my_session.txt")
> sessionInfo()
> sink()
```

Briefly describe in your own words, what kind of information the function `sessionInfo()` provides. Include the resulting file "my_session.txt" in your submission, i.e., in the zip file containing your solutions.

Exercise 2:

COVID-19 [22 points]

The file corona.csv contains information on the worldwide development of the COVID-19 pandemic[2]. Amongst others, the following variables were observed in the data set:

Variable (original name)	New name	Description
`dateRep`	`date`	Observation date
`cases`		Number of newly infected people on this date
`deaths`		Number of people who died on this date
`countriesAndTerritories`		Country/Territory of the observation
`popData2019`		Population (as of 2019)
`continentExp`	`continent`	Continent where the Country/Territory is located
`Cumulative_number_for_14_days_of_COVID.19_cases_per_100000`	`indicator14`	Number of COVID-19 cases per 100,000 residents over the last 14 days (including current date)

a) Set your working directory appropriately and read the data conveniently into a data frame `corona`. Delete all variables not contained in the table above and rename the remaining ones according to "New name", if "New name" is not empty for this variable. Transform the variable `date` to class `POSIXct`. Are the classes of the other variables adequately represented? Briefly justify your answer. [4.5]

b) How many observations in the data set contain missing values? [2]

c) How many observations in the data set counted more than 200 `cases`? What's their share on the total number of observations? [1.5]

d) Use an appropriate test to decide whether the average value of `indicator14` differs in the countries Italy and Spain in the month March (significance level: 5%). Briefly justify your decision on whether the average value differs. [2]

e) Compute the total number of `deaths` per `continent` in the month October and sort them in ascending order. Is the order of the continents the same for the number of `cases` per `continent`? [3]

f) In Germany, a 7 day indicator is popular instead of `indicator14`. It is the number of `cases` per 100,000 residents over the last 7 days (including the current date). Create a new numeric variable `indicator7` containing this 7 day indicator. [6.5]

g) Create a new factor variable `alert_level` which is

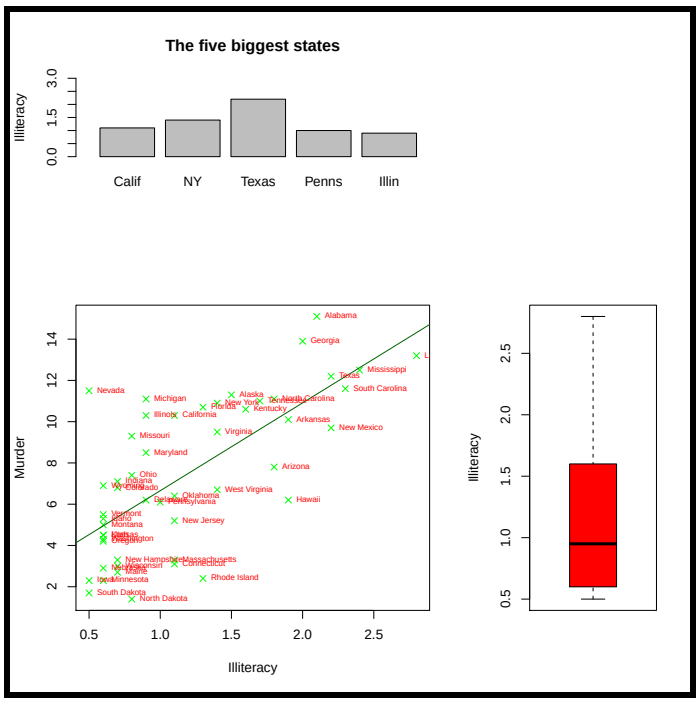
- "green" if `indicator7` is less than 35,
- "yellow" if `indicator7` is at least 35 but less than 50,
- "red" if `indicator7` is at least 50 but less than 100,
- "darkred" if `indicator7` is at least 100.

If you did not manage to solve exercise f) use the variable `indicator14` instead, but double the threshold values (e.g., use 70 as upper bound for "green"). [2.5]

Exercise 3:

State Graphic [13 points]

a) Construct the following graphic, which includes everything inside the thick black frame (except the frame itself), based on the built-in dataset `state.x77`. Use the variables `Population`, `Illiteracy`, and `Murder` as well as the row names, providing the state name. For the barplot at the top consider only the 5 largest states with respect to `Population`. In order to have the barplot labels centered, use the returned value of `barplot()`, see `?barplot`. For the rest of the plots please use the whole data set. The line in the plot at the bottom left is the regression line corresponding to a linear model with dependent variable `Murder` and independent variable `Illiteracy`.



Pay attention to the details, so your graphic looks as similar as possible. In particular,

[2]source: https://data.europa.eu/euodp/de/data/dataset/covid-19-coronavirus-data as of 15 November 2020

- arrangement of the plots and their proportion to each other
- axis labels and scaling
- colors [10]

b) Save the graphic as pdf file in your working directory. [3]

Exercise 4:

RSS feeds [17 points]

The files bbc.txt and guardian.txt contain information obtained from RSS feeds of *BBC* and *The Guardian*[3]. Both files are UTF-8 encoded and contain the names of the variables in their first line. To solve this exercise you are allowed to use the package `stringr`.

a) Set your working directory appropriately and read the files conveniently into two data frames `bbc` and `guardian`, respectively. Make sure that your data frames contain no leading or trailing white space and that strings have class `character`. [3.5]

b) Combine both data frames to one new data frame `rss_feed`, which satisfies the following requirements:

- Its variables are the ones contained in both, `bbc` and `guardian`.
- It contains all observations from both, `bbc` and `guardian`, sorted by a news' publication date `item_pub_date` (ascending order). [4]

The following exercises refer to the data frame `rss_feed` created in exercise b). Always consider both versions of a word – starting with an uppercase or a lowercase letter.

c) The variable `feed_title` contains the title of the RSS feed. Modify it by removing the character "-", precedent white space, and all subsequent characters using an appropriate regular expression. [2]

d) For each of the RSS feeds, compute the share of observations containing at least one of the following words in the variable `item_title`:

"covid", "corona", "pandemic", "lockdown", "wave", "vaccine"

Which observations containing at least one of these words in the variable `item_title` also contain one of these words in the variable `item_description`? Compute the maximum number of appearances of the words above in one observation of the variable `item_description`. [6]

e) How many observations of the variable `item_description` start with the words "a" or "the"? How many contain a number? [1.5]

Exercise 5:

[3]Sources: http://feeds.bbci.co.uk/news/uk/rss.xml, https://www.theguardian.com/uk/rss as of 11 November 2020, 14:30.

Simulation: Linear Model [19 points]

The aim of this exercise is to examine the normal distribution of coefficient estimates in the linear model (assuming normally distributed disturbance terms). For this purpose, implement the following functions:

a)
```
> simulate_lm <- function(x, beta_0, beta_1, sigma) {
+   # Your part
+ }
```

which

- computes a vector `eps` of the same length `n` as the vector `x`, containing `n` random observations from a normal distribution with mean 0 and variance `sigma`,
- computes a vector `y` of length `n` given by $\texttt{y}_\texttt{i} = \texttt{beta\_0} + \texttt{beta\_1} \cdot \texttt{x}_\texttt{i} + \texttt{eps}_\texttt{i}$ for $\texttt{i} = 1, \ldots, \texttt{n}$,
- fits a simple linear model with dependent variable `y` and independent variable `x`.

The function returns a vector `coef` of length 2 containing the estimated coefficients of the model. [4.5]

b)
```
> repeat_lm <- function(n_rep, n, beta_0, beta_1, sigma) {
+   # Your part
+ }
```

which

- computes a vector `x` of length `n` containing values obtained by drawing `n` random observations from a beta distribution with first shape parameter 3 and second shape parameter 3.5 and multiplying each random observation by 8 (Hint: `?rbeta`),
- calls the function `simulate_lm` `n_rep` times with this vector `x` and stores the result, i.e., the estimated coefficients of all `n_rep` models, in a $2 \times$ `n_rep` matrix `coef_matrix`,
- creates the design matrix `X` of dimension `n_rep` $\times 2$ containing ones in the first column and `x` in the second one,
- computes the theoretical covariance matrix `Sigma` (dimension 2×2) of the coefficient vector, given by $(\texttt{sigma})^2 \cdot (\texttt{X}^\top \texttt{X})^{-1}$

The function returns a named list of 2 elements:

- the $2 \times$ `n_rep` matrix `coef_matrix`,
- a vector `coef_sd` of length 2 containing the square root of the diagonal elements of `Sigma`. [6]

Now perform the following simulations. Make sure that your results are reproducible.

c) Execute the function `repeat_lm` with the following parameters:

- `n_rep` $= 200$, `n` $= 100$, `beta_0` $= 5$, `beta_1` $= 2$, `sigma` $= 1$
- `n_rep` $= 200$, `n` $= 500$, `beta_0` $= 0.5$, `beta_1` $= -3$, `sigma` $= 2$

For both function calls create two histograms (each with total area 1) to illustrate the results – one for the estimated values of `beta_0` and one for the estimated values of `beta_1`. Add to both histograms the following densities:

- the corresponding kernel density estimator as solid red line (Hint: `?density`)
- the density of a normal distribution as dashed blue line with `beta_0` or `beta_1` as mean and the first or the second value of `coef_sd` as standard deviation, respectively. Note that this corresponds to the theoretical distribution of the coefficient estimates.

Pay attention to choosing a reasonable title for each plot. [8.5]

Exercise 6:

Connect Four [42 points]

The game *Connect Four* is played by two persons, who take turns dropping one disc (each player has a different color) from the top into a vertically suspended grid (usually 6×7). The pieces fall straight down, occupying the lowest available space within the column. The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own discs. The game ends in a tie, if there is no free cell left and none of the players has a line of four.

Write a function `connect_four` that enables to play Connect Four via the `R` console on a playing field of arbitrary dimension. It should fulfill the following:

- The arguments of the function correspond to the number of rows (`n_row`), to the number of columns (`n_col`) and the number of players (`n_player`; usually 2). The basic structure is:

```
> connect_four <- function(n_row = 6, n_col = 7, n_player = 2) {
+   # Your part
+ }
```

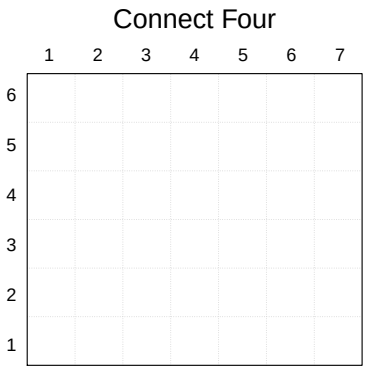
- The number of players can be 1 or 2 and corresponds to the number of real players. If it's equal to 1, then the real player (communicating via the console) is Player 1 while the moves of Player 2 are chosen by the function ("playing against computer"). The function stops with the following error, if `n_player` is neither 1 nor 2.

```
Error in connect_four(): n_player must be 1 or 2.
```

- The player who begins is selected randomly. The following message is printed in the console (appropriately adapted, if Player 2 starts) and an empty playing field of appropriate dimension is plotted (in the Plots-tab of RStudio).

```
Player 1 starts!
In each move you have to choose one column.
```

The empty playing field for 6 rows and 7 columns should look like this:



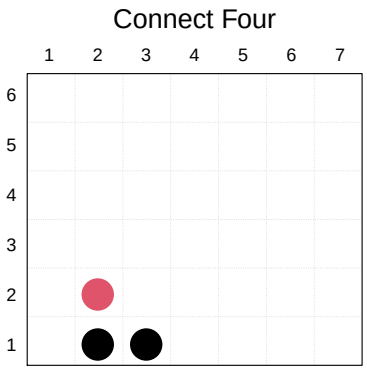
- A real player executes a move by inserting the number of the column in which the disc should be dropped when requested by the console. (Hint: `?scan`)
- If there is only one real player (Player 1), Player 2 chooses to drop their disc as follows:
  1. If Player 2 has the possibility to win with a vertical line of 4, i.e., if in one column the three topmost discs are their own with a free cell left above, Player 2 selects this column.
  2. If 1. is not fulfilled, but Player 1 has the possibility to win with a vertical line of 4, Player 2 prevents it by choosing this column.
  3. If neither 1. nor 2. are fulfilled, Player 2 draws a column randomly (equal probability for all valid columns).

  The following message (appropriatly adapted) is printed in the console:

  `Player 2 chose column 2.`
- The (partial) output in the console during the playing process should be as follows (appropriately adapted)
  - For 2 real players:

    ```
    Player 1 starts!
    In each move you have to choose one column.
    Player 1:
    1: 2
    Player 2:
    1: 2
    Player 1:
    1: 3
    Player 2:
    1:
    ```
  - For 1 real player:

    ```
    Player 1 starts!
    In each move you have to choose one column.
    Player 1:
    1: 2
    Player 2 chose column 2
    Player 1:
    1: 3
    ```
- After every move, the plot of the playing field is updated accordingly. After the three moves given in the example above, it should look like this:



- In each column the maximum number of discs to drop is the number of rows.
- Every move must be valid. Otherwise, the function reacts with the following message and the respective player has to insert a column, again.

```
Player 1:
1: 8
Column not valid. Again:
1:
```

- The game ends (i.e., the function terminates) as soon as one player wins or if there is no free cell left on the playing field. The following messages (appropriately adapted) should be printed in the console, respectively:

```
[1] "Player 1 wins!"
```

or

```
[1] "Game ends in a tie!"
```

- In particluar, after every move it must be tested, whether one player won. I.e., the line of four must include the last disc dropped.
- Hints:
  - Swap (self-contained) subproblems in separate functions, which are then called in the main function `connect_four`.
  - Test the function with different calls.