Exercise #3

Fortgeschrittene Statistische Software für NF - SS 2022/23

Sungho Park (12469817), Jonas Reinhard (12442176), ...

2023-06-09

General Remarks

- You can submit your solutions in teams of up to 3 students.
- Include all your team-member's names and student numbers (Martrikelnummern) in the `authors` field.
- Please use the exercise template document to work on and submit your results.
- Use a level 2 heading for each new exercise and answer each subtask next to it's bullet point or use a new level 3 heading if you want.
- Always render the R code for your solutions and make sure to include the resulting data in your rendered document.
  - Make sure to not print more than 10 rows of data (unless specifically instructed to).
- Always submit both the rendered document(s) as well as your source Rmarkdown document. Submit the files separately on moodle, **not** as a zip archive.

Exercise 1: Initializing git (4 Points)

For this whole exercise sheet we will be tracking all our changes to it in git.

a) Start by initializing a new R project with git support, called `exeRcise-sheet-3`. If you forgot how to do this, you can follow this guide.
b) Commit the files generated by Rstudio.
c) For all of the following tasks in this exercise sheet we ask you to always commit your changes after finishing each subtask e.g. create a commit after task *1d*, *1e* etc.

   Note: This applies only to answers that have text or code as their answer. If you complete tasks in a different order or forget to commit one, this is no problem. If you change your answers you can just create multiple commits to track the changes.

d) Name 2 strengths and 2 weaknesses of git. (Don't forget to create a commit after this answer, see *1c*).
   -strength.
   Branching and Merging: Git provides powerful branching and merging capabilities. Developers can create multiple branches to work on different features or experiments independently, and later merge them back into the main codebase. This enables parallel development, isolation of changes, and facilitates collaboration among team members.
   Version Control: Git excels at managing versions of a project's source code. It tracks changes at a granular level, allowing you to view the history of modifications, compare different versions, and easily revert to a previous state if needed. This capability is particularly useful in collaborative development environments.

-weakness.
Complexity with Advanced Features: While Git's core functionality is relatively straightforward, it can become more complex when dealing with advanced features such as rebasing, resolving conflicts, or handling submodules. These operations may require a deeper understanding of Git's internal workings, and mistakes can potentially lead to data loss or confusion if not executed properly.
Steep Learning Curve: Git has a reputation for having a steep learning curve, especially for users who are new to version control systems or have only worked with centralized systems in the past. The command-line interface and the multitude of commands and options can be overwhelming for beginners. However, there are also user-friendly Git clients available that provide graphical interfaces and simplify the learning process.

e) Knit this exercise sheet. Some new files will automatically be generated when knitting the sheet e.g. the HTML page. Ignore these files, as we only want to track the source files themselves.

Exercise 2: Putting your Repository on GitHub (3.5 Points)

For this task you will upload your solution to GitHub.

a) Create a new repository on GitHub in your account named `exeRcise-sheet-3`. Make sure you create a **public repository** so we are able to see it for grading. Add the link to the repository below:
b) Push your code to this new repository by copying and executing the snippet on github listed under `...or push an existing repository from the command line`.
c) Regularly push your latest changes to GitHub again and especially do so when you are finished with this sheet.

Exercise 3: Baby-Names in Munich (4.5 Points)

Download the latest open datasets on given names ("Vornamen") from the open data repository of the city of Munich for the years `2022` and `2021`.

Link: https://opendata.muenchen.de/dataset/vornamen-von-neugeborenen

a) Download the data for both years and track it in git. For small datasets like these adding them to git is not a problem.

```
vornamen_2021 <- read.csv("vornamen_2021.csv")
vornamen_2022 <- read.csv("vornamen_2022.csv")
```

b) Load the data for both years into R. Check the type of the count variable ("Anzahl") and look into the data to determine why it is not numeric? Fix the problem in an appropriate manner, it is OK if some of the counts are inaccurate because of this. Explain your solution and the repercussions.
-The Variable "Anzahl" is not numeric, because the var "Anzahl" is abbreviated ,which has count of first name under 4. If the "Anzahl" has as Value '4 oder weniger', that Value is changed in a new Variable "count" to 2, and Variables "count" have now Data Type "Numeric".

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
```

```
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
vornamen_2021 <- vornamen_2021 %>% mutate(count = ifelse(vornamen_2021$Anzahl=='4 oder weniger',2,vornam
vornamen_2021$count <- as.numeric(vornamen_2021$count)
vornamen_2022 <- vornamen_2022 %>% mutate(count = ifelse(vornamen_2022$Anzahl=='4 oder weniger',2,vornam
vornamen_2022$count <- as.numeric(vornamen_2022$count)
```

c) Calculate the total number of babies born in Munich in 2022 and 2021. Which year had the bigger baby-boom?
Year 2021 has bigger baby-boom then Year 2022. But the Values are not an exact Value ,because some Values in Column "Anzahl" are not pricise detemined,e.g "4 oder weniger".

```
total_babies_2021 <- sum(vornamen_2021$count)
total_babies_2022 <- sum(vornamen_2022$count)
```

d) Add a new column `year` to both datasets which holds the correct year for each.

```
vornamen_2021 <- vornamen_2021 %>% mutate(year=2021)
vornamen_2022 <- vornamen_2022 %>% mutate(year=2022)
```

e) Combine both datasets into one using `bind_rows()`.

```
df<-bind_rows(vornamen_2021,vornamen_2022)
```

f) Combine the counts for same names to determine the most popular names across both years. Print out the top 10 names in a nicely formatted table for both years. Include a table caption.

```
library(knitr)
popular_names <- df %>% group_by(Vorname) %>% summarize(total_count=sum(count))
popular_names <- popular_names %>%
  arrange(desc(total_count))
popular_names <- head(popular_names,10)
kable(popular_names,caption = "the top 10 names in 2021 and 2022")
```

Table 1: the top 10 names in 2021 and 2022

Vorname	total_count
Maximilian	240
Emilia	234
Felix	220
Anton	206
Emma	199
Leon	195
Noah	185

Vorname	total_count
Jakob	180
Anna	178
Lukas	173

Exercise 4: Chat GPT + apply (3 points)

For this task: Specifically use ChatGPT to solve the task and submit your prompts in addition to the solution

a) The code below does not work because the wrong apply function has been used. Find out which apply function would be correct and why it did not work. Correct the code. Also calculate the rowwise means.

###Create a sample data frame

tax_data <- data.frame(Name = c("Munich GmbH", "ABC Inc.", "Backpacks 1980", "Bavarian Circus"), Tax_2019 = c(5000, 4000, 6000, 3500), Tax_2020 = c(4800, 4200, 5800, 3700), Tax_2021 = c(5200, 3800, 5900, 3400))

###Calculate column-wise means

column_means <- lapply(tax_data

$$, -1$$

, 2, mean)

column_means

```
##Create a sample data frame

tax_data <- data.frame( Name = c("Munich GmbH", "ABC Inc.", "Backpacks
1980", "Bavarian Circus"), Tax_2019 = c(5000, 4000, 6000, 3500),
Tax_2020 = c(4800, 4200, 5800, 3700), Tax_2021 = c(5200, 3800, 5900,
3400) )

# prompt : column_means <- lapply(tax_data[, -1], 2, mean) column_means
# right answer from Chat GPT
# The code you provided calculates the mean value for each column of the "tax_data" data frame, excludi

#1. `tax_data[, -1]` selects all columns of the "tax_data" data frame except for the first column ("Nam

#2. `lapply()` is a function in R that applies a given function to each element of a list or vector. In

#Therefore, the code `column_means <- lapply(tax_data[, -1], mean)` calculates the mean value for each
column_means <- lapply(tax_data[, -1], mean)
column_means
```

```
## $Tax_2019
## [1] 4625
##
## $Tax_2020
## [1] 4625
##
```

```
## $Tax_2021
## [1] 4575
```

```
## prompt : calculate rowwise means
## answer from Chat Gpt
# Calculating row-wise means
row_means <- rowMeans(tax_data[, -1])
```

b) Using ChatGPT try to understand what the rapply() function does. Create an easy example with mock data where the function is used and explain it in your words.

```
# prompt: What is rapply() in R?
# explain from GPT: The rapply() function is particularly useful when dealing with complex nested struc
# example
my_list <- list(a = 1:3, b = list(c = 4:6, d = 7:9))
result <- rapply(my_list, function(x) x * 2)
print(result)
```

```
##   a1   a2   a3 b.c1 b.c2 b.c3 b.d1 b.d2 b.d3
##    2    4    6    8   10   12   14   16   18
```

```
# explain in our words:
# list a has values 1,2,3 , so function(x) calculate x*2 , we get the result 2,4,6. And B is nested Lis
```

Final Note

Make sure to push all your commits and changes to GitHub before submittining the exercise sheet.