R – A NON-TECHNICAL INTRODUCTION TO BIG DATA TECHNIQUES, TEAM WORK AND INTERACTIVE VISUALIZATION WITH APPLICATIONS TO MARKETING
@ UZH

EXERCISE OVERVIEW

DAY 3

- ***17 – Why version control?***

1. Install Git on your computer: https://git-scm.com.
2. Create a Github account: https://github.com.

- ***18 –Let's get started with Git***

1. Create an RSA key in RStudio and add it to your GitHub account.
2. Create a new public repository on GitHub.
3. Create a RStudio project with version control, that is connected to the new repository on GitHub.
4. Create a test file, add some changes, and push it to your GitHub repository.
5. Clone the following repository to your computer: https://github.com/octocat/Spoon-Knife

- ***19 – Advanced Git features (HOMESTUDY)***

1. Fork the following repository on GitHub and then clone it: https://github.com/octocat/Spoon-Knife
2. Add a new file to your project, i.e., an empty text file, and change the file to your name. Commit and push your change and create (on Github.com) a pull request.

- ***20 – Why create a package?***

1. Build your own RFM package based on your RFM function from LE 1-3.
2. Use your RFM package to calculate RFM scores (weights: 20, 20, 60).

- ***21 – Keep calm and read the manual***

1. Install the roxygen2 package and prepare RStudio for it.
2. Document the RFM function with roxygen2 in your package and build it again. (Hint: Have a look at the roxygen_template.R)
3. Check out the new help file.

- ***22 – Publish your package (HOMESTUDY)***

1. Build a source package of your RFM package
2. Publish your package on GitHub (public repository).

3. Perform a check of your package. What would you need to change to publish your package on CRAN?

- ***23 – Debugging – Find the mistake***

1. Create a function called `addFirstTwo(vec)` that adds the first two elements of a vector.
2. Run the following code: `addFirstTwo(c(1, "z"))`. What happens?
3. Now let's debug. First, add `print()` statements inside the function that show the class of the first two elements of vec. *Hint: use the function `class()`.*
4. Instead of the print statements, use the function `browser()`. Place this inside the `AddFirstTwo()` function. Call the function again by running `addFirstTwo(c(1, "z"))`. Have a look at the Environment tab in RStudio. What do you notice? Then type n in the console and see how the next line of code gets executed.
5. Run the following code: `mean(1:10, trim = NA)`. Use traceback to see which sub-function is causing the error.

- ***24 – Unit testing – Cover all the bases***

1. Add the function `addFirstTwo()` to you RFM package. If you didn't manage to build the RFM package, just create a new package and use that one.
2. Create a unit test framework by running the function `usethis::use_testthat()`.
3. Write tests for your `addFirstTwo()` function. Create at least two test files with at least three tests each. Don't forget to put the files in the "`testthat`" directory and define a context in each file! *Hint: have a look at the `usethis::use_test()` function*.
4. Run the tests using the function `devtools::test()`

- ***25 – Continuous integration services (HOMESTUDY)***

1. Measure the coverage of your tests. Do this by loading the `covr` package and running the function `report()`
2. Push your package to a GitHub repo and set up Travis CI.