README

- Practice instructions for the introduction to R lectures
- Emacs + ESS + Org-mode and R must be installed
- Upload the completed file as a class work assignment (Canvas link)

TIME TABLE

PRACTICE	MIN
Install R	30
Find R	15
Run scripts	15
Explore shell	30
Change dir	10
Change prompt	10
Compute/comment	10
R Packages	15
Customization	15
TOTAL	150

TODO IDENTIFY YOURSELF

- Update the `#+AUTHOR:` information in the header
- Add this on a line to the header of this file : `#+STARTUP: overview hideblocks indent`
- With the cursor on theline, activate the header line with `C-c C-c`.
- Put your cursor on the headline of this section, and type `S <LEFT>` until you see `DONE` instead of `TODO` next to the title.
- Perform this last step each time you complete a section.

TODO DOWNLOAD AND INSTALL R (own PC only)

Time: 30 min

- You can only do this on a computer where you have adminstrative rights. Hopefully, you've brought your laptop to the session (if you have one).
- You find the latest version of R, Emacs + ESS + Org-mode pre-installed on all computer lab PCs.
- Detailed instructions are available in the FAQ on GitHub.

TODO FIND R PROGRAMS ON YOUR COMPUTER

Time: 15 min

There is only one R program on your Windows box, but there are multiple ways of using it.

1. Find the programs `Rterm` and `Rgui` on your Windows box (they are likely in `C:\Program Files\R\R-4.1.3\bin\x64` (R version may differ).
2. Start both programs from the File Explorer
3. Open a CMD line terminal and type R at the prompt.
4. In each shell program, execute the command `plot(rnorm(1000))` at the command prompt (`>`). This should open a graphics device with a plot of one-thousand random points.
5. In Emacs, enter `M-x R` - a buffer named `*R*` should open
6. At the `>` prompt, enter the command `plot(rnorm(1000))`.

TODO RUN R SCRIPTS ON THE COMMAND LINE

Time: 15 min

You can save R instruction sets as scripts and execute them on the CLI. In our sample script, we plot one-thousand random points and print the structure of a built-in data frame, `mtcars`.

1. Put the following two lines into a script `test.R`:

```
plot(rnorm(1000))
str(mtcars)
```

2. Open a Windows CMD line terminal/shell in Emacs with `M-x shell RET`.
3. Make sure that you're operating in the directory where `test.R` is located by listing the file. Enter at the prompt: `DIR test.R` - you should now see a listing of the file.
4. Run the script on the shell with `Rscript test.R`. You should see the output of the second command on the screen.
5. Open a `Dired` buffer, and press `s` to see the last files that were created. You should see a file `Rplots.pdf`. It contains the output of the graph command. In a Windows Emacs, you cannot open it (on MacOS and Linux Emacs, you can): find it in the File Explorer and open it to see the familiar random points plot.
6. Delete the file `Rplots.pdf`. (`D` in the `Dired` buffer).
7. To test batch mode, run this command on the shell: `R CMD BATCH test.R`. This time you only get a confirmation of the end of the batch process.
8. Go back to the `Dired` buffer and open the new file `test.Rout`. It contains the non-graphical results as part of a whole R session.
9. There is a new file `Rplots.pdf` that contains the PDF with the plot.

TODO EXPLORING THE R CONSOLE

Time: 30 min

1. You can open as many R instances as you like. Open a (new) R instance inside Emacs now (`M-x R`).
2. Close the instance with the command `q()` (confirm with `n`)
3. You see the R shell startup screen. It contains information about:
   - The *version* of the installed R program
   - The *license* and copyrights of R
   - The R *project* and *citation*
   - The *help* you can get on the shell
   - How to *quit* the shell
   - R's current *working directory*

4. You can check R's version at any point by entering the command `version` at the prompt. When you enter it, you get a lot more information than you saw before - to find out more, follow this footnote by putting your cursor on the label and entering `C-c C-o`:[1].
5. Enter `license()` at the R program prompt `>` for license information[2].

6. What is the "GNU General Public License" mentioned here? You can find this out with the links given by `license()`. Copy the answer in the following `quote` block:

   > Answer: "The GNU General Public License is … a free, copyleft license for software and other kinds of works." From the GPT-3 Preamble - open with the command `RShowDoc("GPL-3")`.[3]

7. Compare the R startup screen with the GNU Emacs startup screen by entering `M-x about-emacs`. The Linux kernel is another famous program licensed under the GPL.
8. Call the function `contributors()`. Check the output to satisfy your curiosity and to marvel at the community effort. Anybody can contribute!

9. Call the function `citation()`. What is the purpose of this information? What are `BibTex` and `LaTeX`? When should you cite the R Project?

   > Answers: (1) **LaTeX is …** a typesetting software used especially for the preparation of scientific documents. (2) **BibTeX is …** a tool for formatting lists of references in LaTeX. (3) **When cite R?** You should cite R in any publication that uses R for data analysis.

10. Follow the instructions on the startup screen and type `demo()` to see all demo categories, and then `demo(graphics)`. Click in the opening graphics window to advance the demo. You can stop it by typing in `C-c C-c` inside Emacs.

11. For online-help, type `help.start()`. The help menu opens in a browser. Where is this information coming from?

    > The help information is located … on your computer. The URL `127.0.0.1:14307` is an alias for `'localhost'` on port `14307`.

TODO CHANGE WORKING DIRECTORY

Time: 10 min

1. Start another R shell inside Emacs. In the mini-buffer, you'll be prompted for the `starting directory` associated with the new R shell. If you're still in the directory `Downloads`, accept the choice, otherwise enter `~/Downloads`.
2. In your new R shell, you should see a `setwd` function call to the directory you just accepted. However, since `setwd` interacts with the operating system (e.g. Windows), the path to `~/Downloads` depends on the OS.
3. Display the current working directory using the function `getwd`[4].
4. Using an Emacs `Dired` buffer (`C-x d`), create a new directory called `Test` (enter `+` in `Dired`).
5. Go back to the R buffer and set the new directory to `~/Downloads/Test`, and confirm the new working directory.
6. Set the new working directory to your user directory (for me, on Windows, that's `C:/Users/birkenkrahe`[5]. You can do this either using an *absolute* path, like `'c:/Users/birkenkrahe'` as the argument, or a *relative* path, like `'../../'`.
7. On the shell, the period operator `.` stands for the current directory. Use it as an argument for `setwd()`.
8. Confirm that your working directory is unchanged.

TODO CHANGE R SHELL PROMPT

Time: 10 min

1. Open an R shell in Emacs.
2. Change the prompt of your R shell to your own name, like this: `Marcus>`
3. Display the value of the prompt with the command: `options()$prompt`
4. Change the value of the prompt back to `'> '`.
5. Display the new value of the prompt.
6. Take a look at the documentation for the function `options` by entering either `help(options)` or the equivalent `?options`

TODO COMPUTE AND COMMENT

Time: 10 min

1. Open an R shell and compute `2+2`
2. Pass the operation `2+2` as argument to the `print` function
3. Run both computations again, but this time with an inline comment. The result should be the same
4. Put the code into an R script and save it as `test.R`
5. Inside Emacs, open a Linux-type shell with `M-x eshell`
6. Run the script with `Rscript`

TODO R PACKAGE COMMANDS

Time: 15 min

1. Open an R shell with `M-x R`
2. Install the `MASS` package with `install.packages`
3. List all data sets in `MASS` with `data(package="MASS")`

4. Open the help for the data set `MASS::Boston` - how many rows (observations) and columns (variables) does it have?

   The `Boston` data set has … rows and … columns

5. Load the `MASS` package into your current R session
6. Load the data set `Boston` into your current R session
7. List all loaded packages with `search()`
8. List all loaded objects with `ls()`
9. Display the structure of `Boston` with `str(Boston)`
10. Display the first **three** rows of `Boston` with `head`
11. Remove all loaded objects with `rm(list=ls())`
12. Detach the `MASS` package with `detach("package:MASS")` and list the loaded packages again.

TODO CUSTOMIZING AT STARTUP

Time: 15 min

1. Find out which directory Emacs (and R) consider to be `$HOME`:
   - Open an `eshell` in Emacs
   - Enter `echo $HOME` - the output is your Emacs/R home directory

- Open your `$HOME` directory in Emacs by opening `~/`

2. Create a file `.Rprofile` in your Emacs `$HOME` directory and put the following lines into it[6]:

```
options(repos=c("https://mirrors.nics.utk.edu/cran/"))
options(crayon.enabled = FALSE)
message("*** Loaded .Rprofile ***")
```

3. Open a new R shell and display the value of `options()$repos` that you just reset. Every time a new R shell is started, `.Rprofile` is read. Make sure that the `message` is displayed.
4. Re-install the `MASS` package from the new location.

Footnotes:

[1] Your first footnote! The `version` command returns platform and os (operating system) information: what you see is the processor (e.g. `x86_64`) and the OS/operating system, e.g. `linux` or `mingw32`, which isn't an OS at all but a C compiler (minimal GNU C compiler, 32-bit), which was used to *build* your R program for your computer only. You also see more detailed version information. You can now return from this footnote by entering `C-c &` (or `C-c C-o` when your cursor is on the footnote link itself).

[2] The parentheses imply that you just called a function. Note that the version information was not obtained with a function call. Try entering `version()` (with parentheses) and `license` (without parentheses) to see what happens. Note that you can cycle through previously entered commands with `M-n` and `M-p`.

[3] The only possibly obscure term here is "copyleft" - this means that the software may be used, modified and distributed freely on condition that anything derived from it is bound by the same condition (namely subject to the GPL). This mechanism stops software from being commercialized to the point that a company can stop you from using, modifying or distributing it freely. In essence, it means that the company cannot base its business model on licensing software but only on selling services (to install, maintain, develop etc.).

[4] `setwd` requires an argument, while `getwd` does not. You can find this out by entering only the name of the function.

[5] Note that Emacs and R require you to give pathnames using the forward slash `/` instead of Windows' backward slash `\`.

[6] The first line determines where packages are installed from. The second line allows Emacs to print so-called "tibbles", data frames in the "Tidyverse", and the last one prints a startup message.

Author: [YourName]
Created: 2022-09-08 Thu 14:14