1. Use Make to automate your work

1. What is Make?
Make is a tool we use to automate our research projects and make it reproducible. If you update a certain file and want to re-run the project, you can easily do so by simply running "make" in the terminal.

2. Get your setup ready!
Make sure you properly installed make by following the instructions on this page: https://tilburgsciencehub.com/get/make

3. A Makefile consists of a set of rules:

```
targets[1]: prequisites[2]
    commands to build[3]
```

→

```
plot.pdf: plot.R table.csv
  R --vanilla < plot.R
```

→ Build the plot.pdf (1) using the plot.R script and table.csv file (2). The command line opens R and runs the plot.R script (3)

Remember the structure of a Makefile (let's call it a recipe):

- **Targets:**	Files that you want to build
- **Prerequisites:**	Files you need to build the targets
- **Commands to build:**	Series of steps to build the targets (indented with a tab!)

2. How to use Make

1. Has somebody else already written a Makefile?
Check the directory and see whether there is a Makefile (without any file extension!). If it's there? Sit back, relax, open your command prompt or terminal and type… make

2. No Makefile present? Follow the following steps to create one!
The first step is to make a proper directory structure. Below is an example we use in the tutorial:

- `data` → Store raw data files
- `src` → Stores source code to build the project. Within this folder, use subdirectories:
  - `data preparation:` Cleaning datasets
  - `analysis:` Analyzing cleaned data
- `gen` → Store generated files. Again, use subdirectories to structure your project
  - `temp:` Temporary files that still need transformations
  - `output:` Final documents (e.g., datasets or tables and figures from analysis)

3. Place the files in the corresponding folder
First, create a Makefile in each subdirectory of your src folder. You can create a Makefile by opening a new script in R and renaming it to "makefile" (without the .R!). For a project with subdirectories src/data-preparation and src/analysis, the file structure should look as follows:

```
/src/data-preparation      /src/analysis
-   download.R             -   plot.R
-   clean.R                -   makefile
-   makefile
```

Don't forget to create the (sub)directories when referring to them in the respective script (e.g., type dir.create("../../gen/output") and save an output file as ../../gen/output/{file_name}

4. Writing your Makefile(s)
Within the Makefiles, write the necessary rules to run the code. An example for the data-preparation folder could look like this:

```
all: ../../data/reviews.csv ../../gen/temp/aggregate_df.csv

../../data/reviews: download.R
    R --vanilla < download.R

../../gen/temp/aggregated_df.csv: ../../data/reviews.csv clean.R
    R --vanilla < clean.R
```

Besides using R --vanilla, which produces the full output, there are several other ways to call programs, such as:

- Rscript file.R → (no output on the screen, unless requested)
- python file.py → (executes a python file)
- Rscript –e "rmarkdown::render("file.rmd") → (build html files from .Rmd files)

If you have multiple prerequisites, make sure to separate each by a space!

5. Specifying long paths using variables [Optional – but recommended!]
If you noticed in step 4, constantly writing ../../ is quite cumbersome. Therefore, we introduce variables:

```
TEMP = ../../gen/temp
DATA = ../../data
```

In the Makefile, refer to these variables using $(VARIABLE) (e.g., $(TEMP)). These variables make your script less prone to errors. The "all" target from step 3 now looks as follows:

```
all: $(DATA)/reviews.csv $(TEMP)/aggregate_df.csv
```

The "all" target tells make which files to build. Without this target, make will simply start running the first rule, followed by the second etc.

Recall: ../../ means go up 2 directories. For temp, you then cd into the gen/temp folder.

6. Finalizing your automation task
Finally, create an overall Makefile that triggers the two Makefiles in their respective subfolder. Place this Makefile in the root directory.

```
all: analysis data-preparation

data-preparation:
     make –C src/data-preparation

analysis: data-preparation
     make –C src/analysis
```

Finally, type make in the terminal and see your project build itself! If you make any changes to a certain script, you only need to type make in the terminal to see your research project being rebuilt again, thus being a very efficient way to automate your work and make it reproducible!

For more content and cheatsheets, check out the course website!

Tilburg ScienceHub