Introduction to useful software and workflows

Introduction to useful software and workflows

Git: Idea and concept

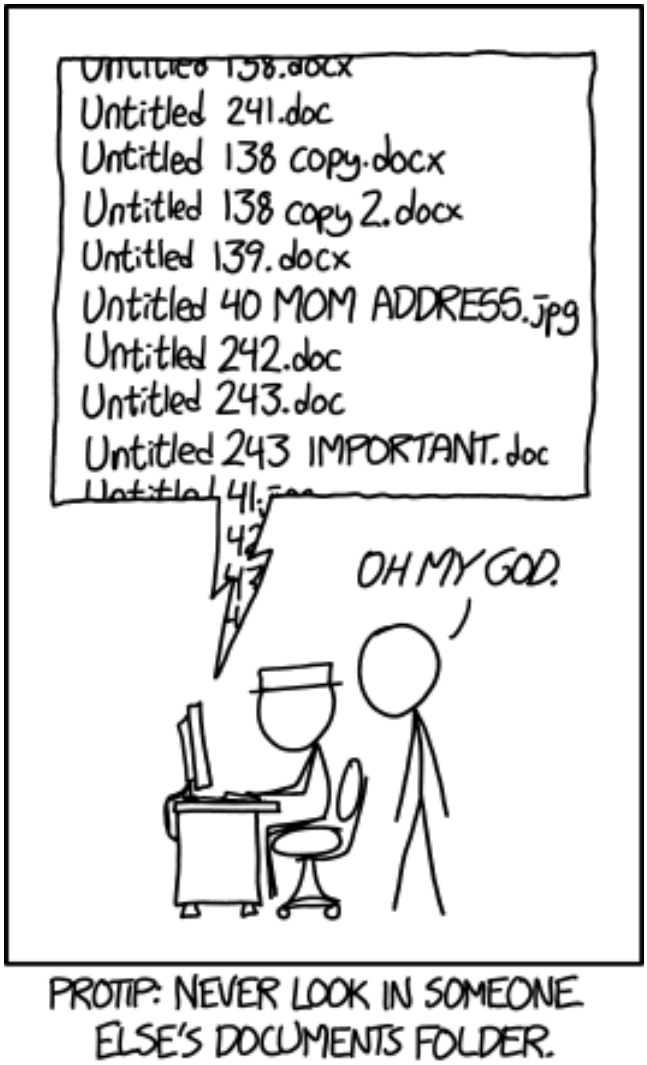


Figure 1: bg left:40% 75%

- distributed version control system

- track and document changes in code
- compare and find differences
- “time travel maschine”
- helps to imagine changes as smaller tasks
- collaborate on projects, share and publish them

Git: How to use it

	Ease of use	Efficiency	Nerdiness
CLI	●○○○○	●●○○○	●●●●●
GUI	●●●○○	●●●○○	●●○○○
RStudio	●●●○○	●●●●○	●●●○○
VSCode	●●●●●	●●●●○	●●●●○

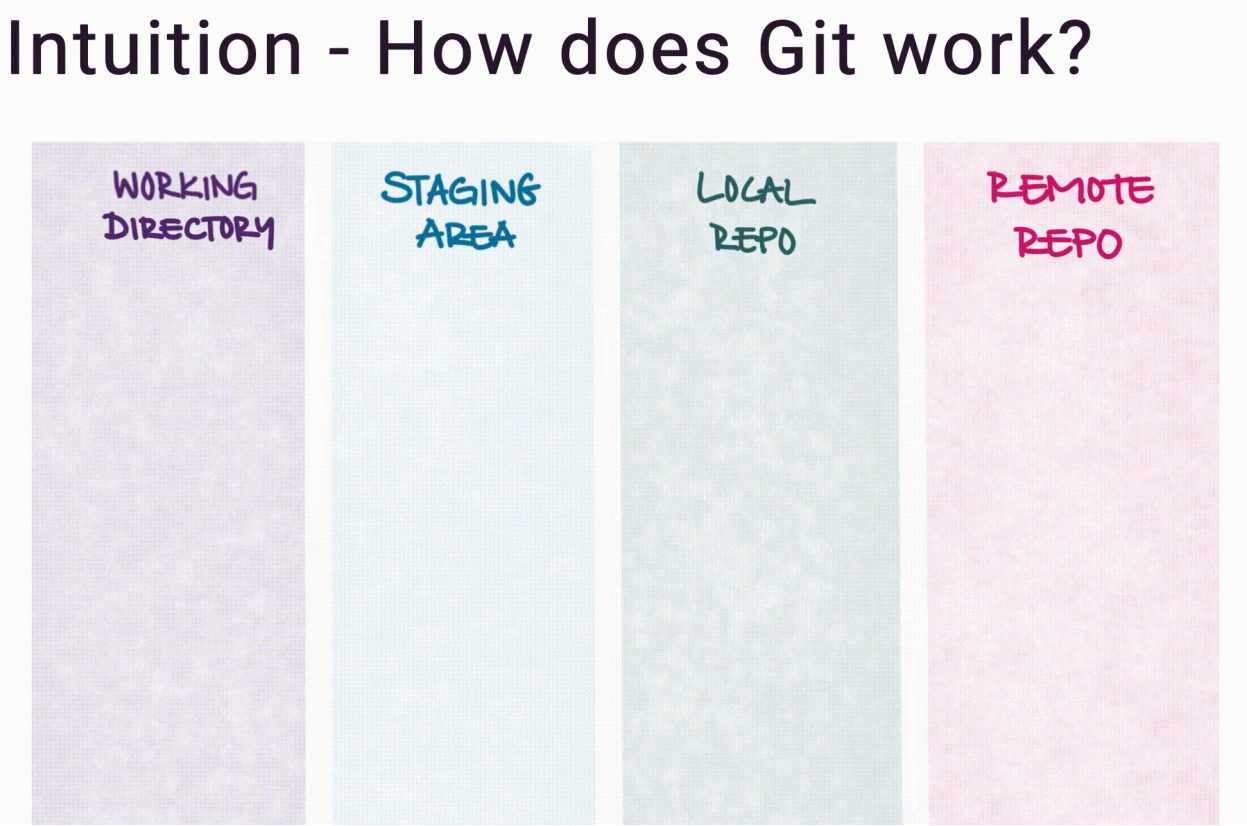


Figure 2: bg 70%

Git: First and basic steps

```
$ git config --global user.name <your name>
```

```
$ git init <your repository name>
$ git status

$ git add <file-name-1> <file-name-2> OR --all
$ git commit -m "<commit-message>"
OR BOTH IN ONE
$ git commit -am "<commit-message>"
```

GitHub: Remote and cooperative workflow

```
$ git clone <git-repo-url>

$ git branch < branch-name>
$ git checkout <name-of-your-branch>
OR BOTH IN ONE
$ git checkout -b <name-of-your-branch>

$ git push

& git fetch
& git merge <branch-name>
OR BOTH IN ONE
$ git pull <remote> (<branch-name>)

$ git fork
```

GitHub: Credentials

- for GitHub remote commands: Account and personal access token needed
- Settings → Developer Setting → Personal Access Tokens → Generate new token
- on local machine enter username and PAT every time, or store:
    1. in CLI: `git config credential.helper store`
    2. in RStudio: `credentials::set_github_pat("YourPAT")`

GitHub: .gitignore

How it works: text file with folder names and files (patterns) not to track

Use case: sensitive data; temp and old files; big data files; outputs → usually track just plain text files (e.g. R scripts, TeX source, etc.)

Get startet: 2 approaches (online tool creates .gitignore content for you)

```
/data/old
passworts.txt
*.doc
------OR------
/*
!.gitignore
!/scripts
```

Git: In RStudio

1. you need to start a new project in RStudio (clone from repository): File → New Project → Version Control → Git → Add URL and Folder
2. new in upper right corner of the screen: **Git**
3. add, commit and push your changes directly in RStudio to GitHub

Git: Resources

- Intro to Git, for the Social Scientist
- Git for Social Scientists
- Git for Students in the Social Sciences
- GitHub - The Perks of Collaboration
- AI tool suggesting git command
- Configure GitHub for Rstudio

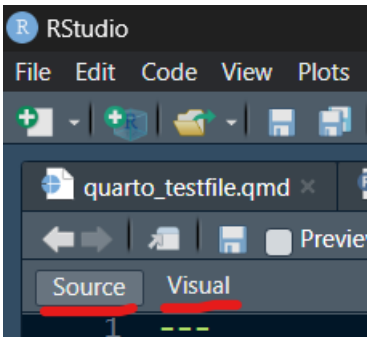
Exercise: Course material

0. find in groups
1. install git & create a GitHub account
2. become collaborator (tell me your username)
3. clone our course material **repository**
4. add a personal folder and test file in exercises
5. push this changes to the remote repository
6. pull changes of the other participants

Installation: Quarto

On Tuesday and Wednesday, we are going to use Quarto Markdown Documents, instead of R scripts. Quarto should be pre-installed in RStudio. Please check whether it is by opening the file "day1_r_git/quarto_testfile.qmd" with RStudio.

Also make sure that you see the "Source" and "Visual" buttons in the top left (see image).



If it is not installed, please update your RStudio version!

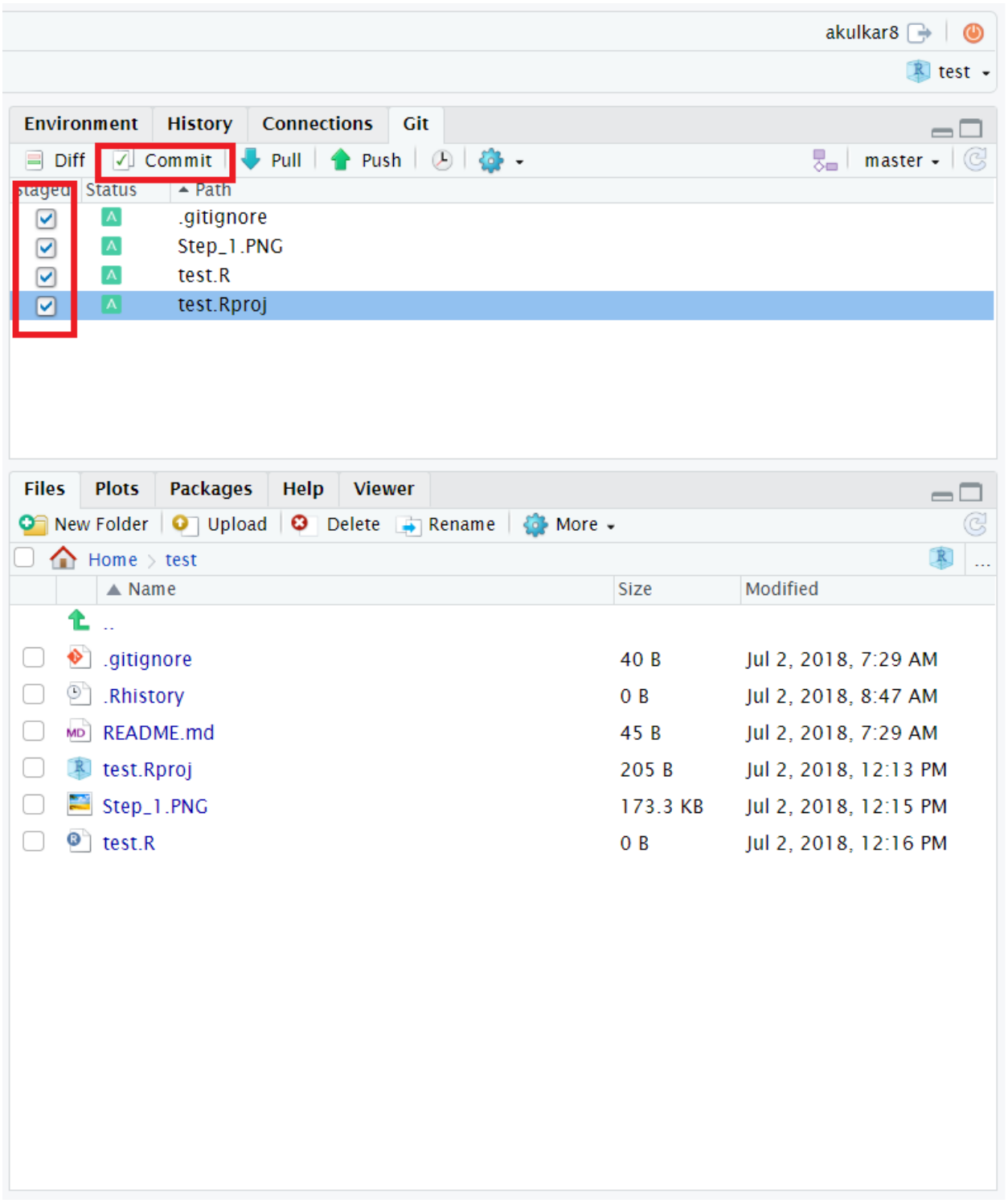


Figure 3: bg left:40% 90%

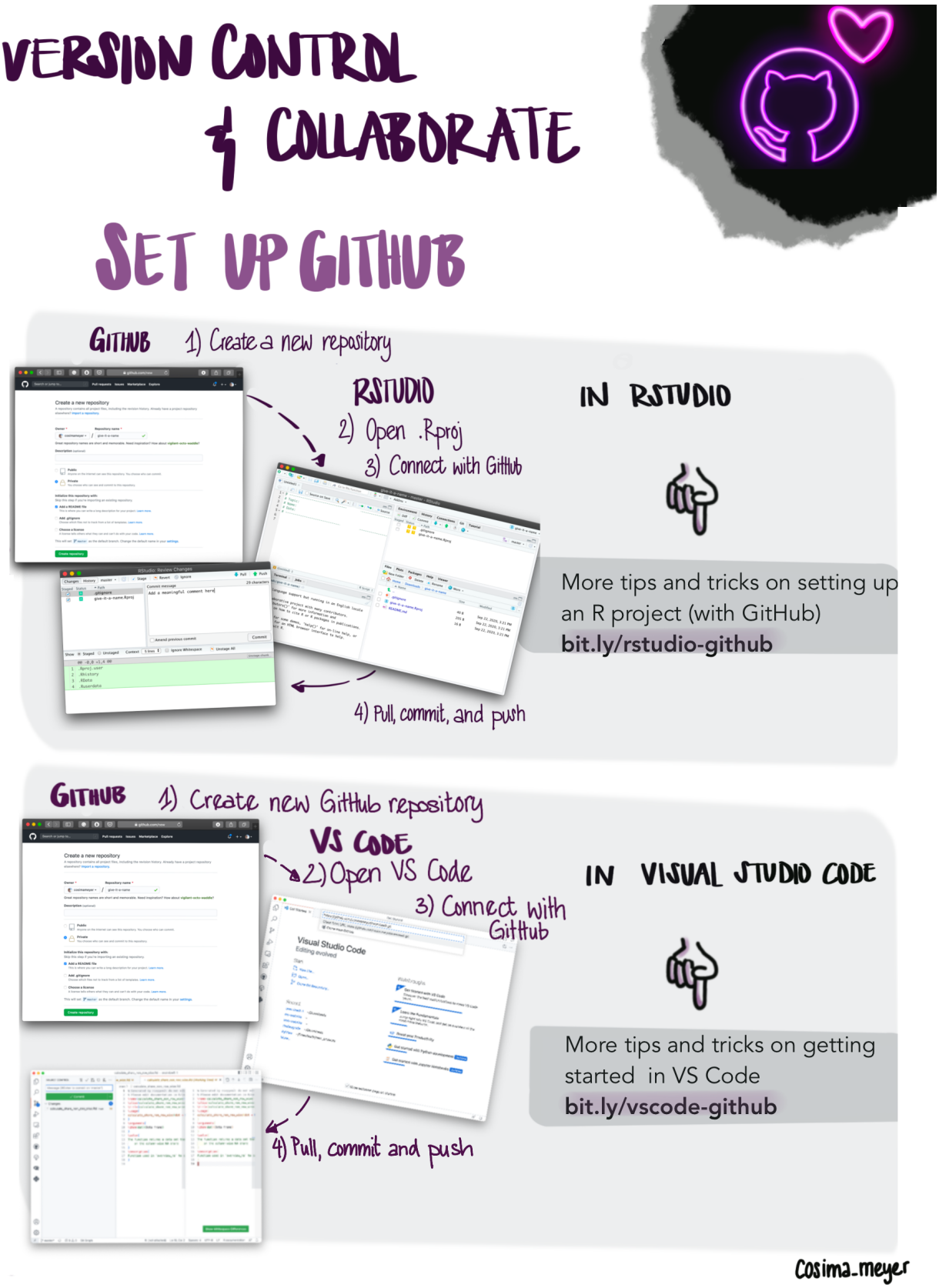


Figure 4: bg left:40% 90%

Installation: RSelenium

You will need to follow the steps described in this Video.

For Everyone:

- Chromedrivers from RSelenium are outdated, download an up to date chromedriver yourself. Check the version of your Chrome browser, and then download the matching chromedriver from here. See this video for a step-by-step guide.

For Apple users:

- When selecting the "Architecture" for the Java SDK on Azul, you need to know which version to choose. You find that out by pressing the Apple button and selecting "About this Mac". With M1 or M2, select "Arm 64 bit", or with Intel, select "x86 64 bit". To find the libray: Finder → Go to (top of screen) → Option → Library

---

```
install.packages(c("RSelenium", "wdman", "netstat", "binman"))

library(RSelenium)
library(wdman)

selenium()

selenium_object <- selenium(retcommand = TRUE,
                            check = FALSE)
```

```
binman::list_versions("chromedriver")

# The following command should open a browser window (you might need to adjust the version!)
remote_driver <- rsDriver(browser = "chrome",
                          chromever = "126.0.6478.127",
                          verbose = FALSE,
                          port = free_port())


# close the server
remote_driver$server$stop()

# If you start it a few times, but never close the server there might be no empty port left.
# You can run the following to kill all java processes
system("taskkill /im java.exe /f", intern=FALSE, ignore.stdout=FALSE)
```

If you manage to start your chrome browser with the above script, RSelenium is installed properly.

Installion: Python Anaconda

- Anaconda is a free and open-source distribution of the Python programming language
- includes also Jupyter Notebooks, Conda (Package Manager) and over 1500 pre-installed data science related packages
- will be useful to interact with LLM models with Python (from RStudio)

Decide which one to download: - Anaconda (extensive and effortless) - Miniconda (slim and customizable)

Installation: Transformers

Install python package transformers with conda package manager:

1. open anaconda/miniconda prompt
2. `conda install transformers`
3. check installed packages: `conda list`

If you use Anaconda, you can also try the GUI “Anaconda Navigator”.

Code Editor: VSCode

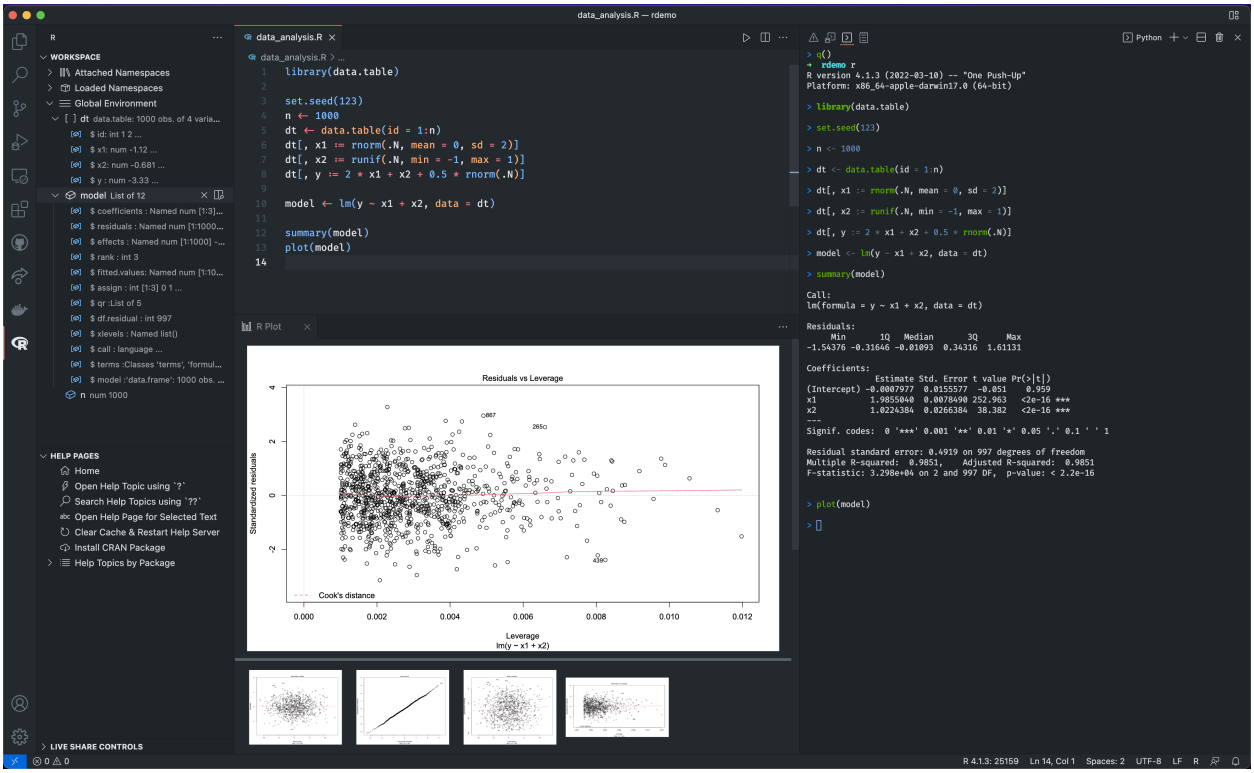


Figure 5: h:530 drop-shadow:0,10px,20px,rgba(0,0,0,.4)

Code Editor: Benefits

- Swiss army knife for coding and file management
  - search (and replace) in whole project folder
  - side-by-side editor windows
  - better file and folder management
  - customizable (with extensions)
- multiple languages supported (e.g. R, Python, Notebooks, LaTeX, Markdown)
- easy Git(Hub) integration for better workflow
- with R:
  - run multiple R Sessions in parallel
  - scripts still editable if process is busy

Code Editor: Resources

- https://code.visualstudio.com/docs/languages/r
- https://renkun.me/2019/12/11/writing-r-in-vscode-a-fresh-start/
- https://schiff.co.nz/blog/r-and-vscode/
- https://rolkra.github.io/R-VSCode/

Best practice: Folder and file structure

1. use separate folders for scripts, data, output, and reports
2. if to many files (~10), use subfolders
3. separate raw data files from processed data
4. use clear and consistent names for script, data, and output files:
   - numbering, lowercase, connect words with underscores or hyphens
   - if date is necessary, put at the end, sort by YYYYMMDD
5. multiple script files for different (sub) tasks (max 100 lines)

Best practice: Efficient R scripts

1) define libraries, default variables, source code at top of script
2) comment and structure sections (# —- headline —-)
3) use pipe operator |> (magritter: %>%) for combining functions
4) use indentations and spaces for readability
5) max line length of 80 characters
6) DRY - use lists, lapply, vectorization, and functions
7) use relative paths for data and output
8) avoid hard coded subsetting and indexing

Best practice: Resources

- Best Coding Practices for R
- Structuring R projects

- R Best Practices
- Tips for organising your R code
- Nice R Code
- Repeating things: looping and the apply family