University of BRISTOL

How to organise your projects with Rstudio

IEU Introduction to R (part 2)

Marina Vabistsevits

MRC Integrative Epidemiology Unit

What this session is about

1. Organising your projects with .Rproj aka **project-oriented workflow**
2. Rstudio efficiency tips
3. (bonus) .Rproj with git

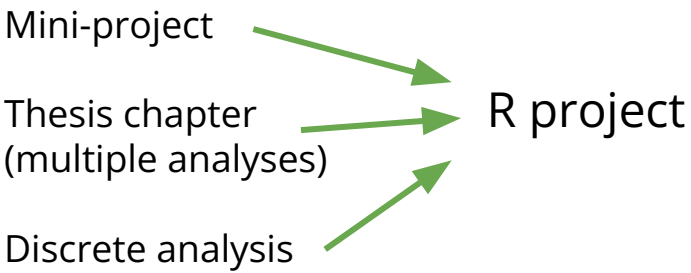
Feel free to follow along as we go or try thing out in mini-breaks after each part

Icon for R

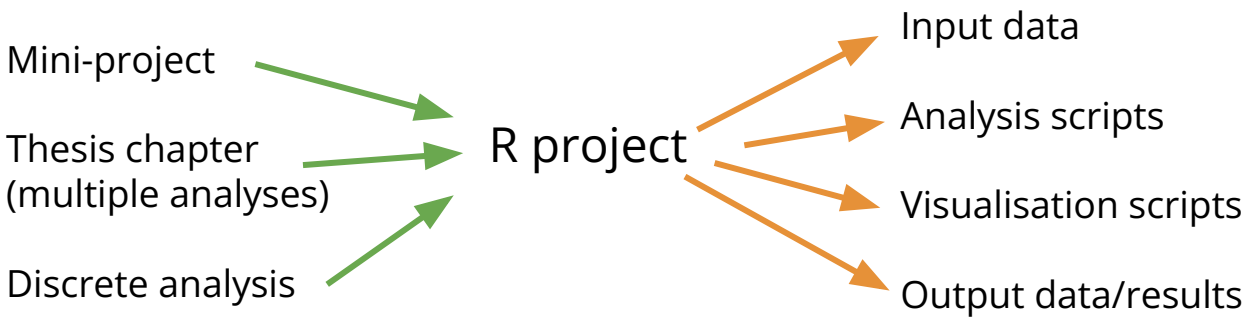
Icon for RStudio

1. Project-oriented workflow

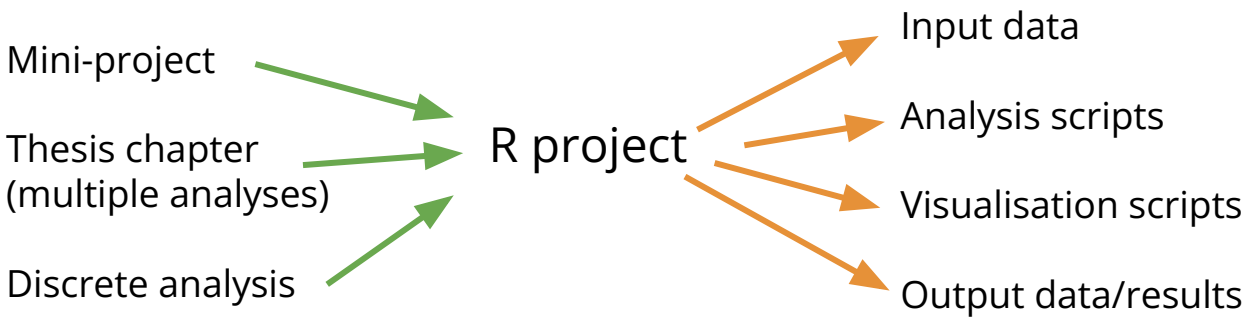
R projects


Mini-project
Thesis chapter
(multiple analyses)
Discrete analysis
R project

R projects


Mini-project
Thesis chapter
(multiple analyses)
Discrete analysis
R project
Input data
Analysis scripts
Visualisation scripts
Output data/results

R projects



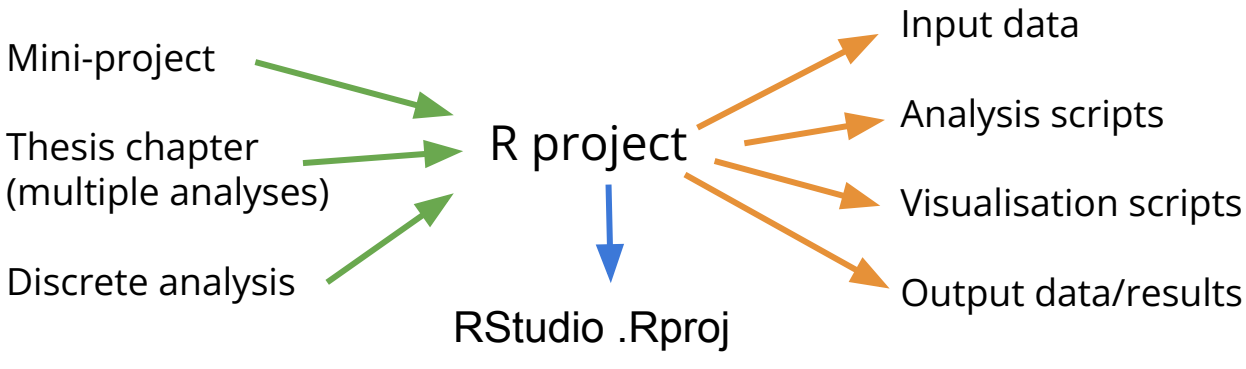
How you do organise your projects? scripts? folders?

Where do you store and access your data? results? figures?

Does it matter? Are your analysis/projects reproducible?

Can you organise your projects better and make life easier for future self/colleagues?

R projects



How you do organise your projects? scripts? folders?

Where do you store and access your data? results? figures?

Does it matter? Are your analysis/projects reproducible?

Can you organise your projects better and make life easier for future self/colleagues?

Reproducibility can be enhanced through intentionally organising projects with

Project-oriented workflow

Photo by secumem

2017/12/12

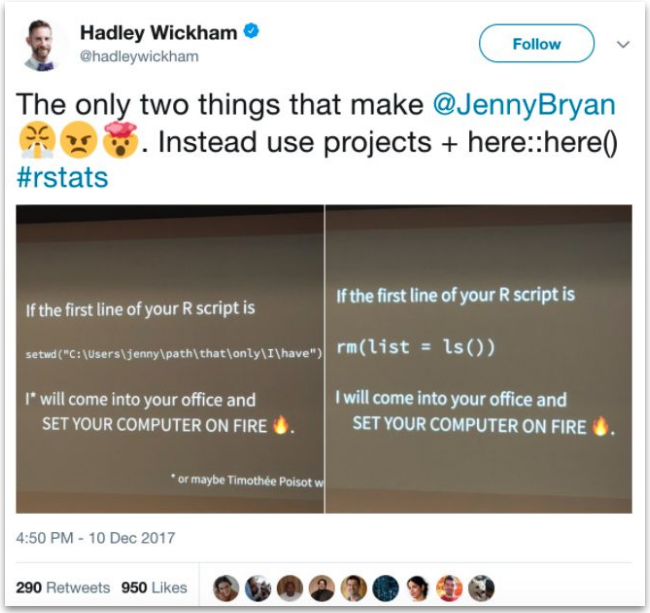
Jenny Bryan

I was honored to speak this week at the IASC-ARS/NZSA Conference, hosted by the Stats Department at The University of Auckland. One of the conference themes is to celebrate the accomplishments of Ross Ihaka, who got R started back in 1992, along with Robert Gentleman. My talk included advice on setting up your R life to maximize effectiveness and reduce frustration.

Two specific slides generated much discussion and consternation in #rstats Twitter:

> *If the first line of your R script is*
>
> `setwd("C:\Users\jenny\path\that\only\I\have")`
>
> *I will come into your office and SET YOUR COMPUTER ON FIRE 🔥.*

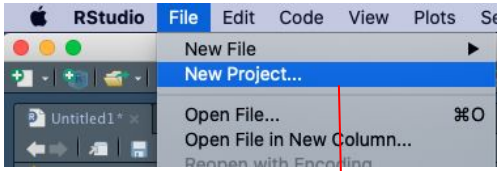
> *If the first line of your R script is*
>
> `rm(list = ls())`
>
> *I will come into your office and SET YOUR COMPUTER ON FIRE 🔥.*

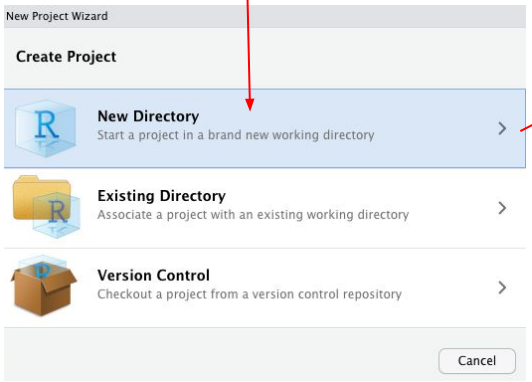


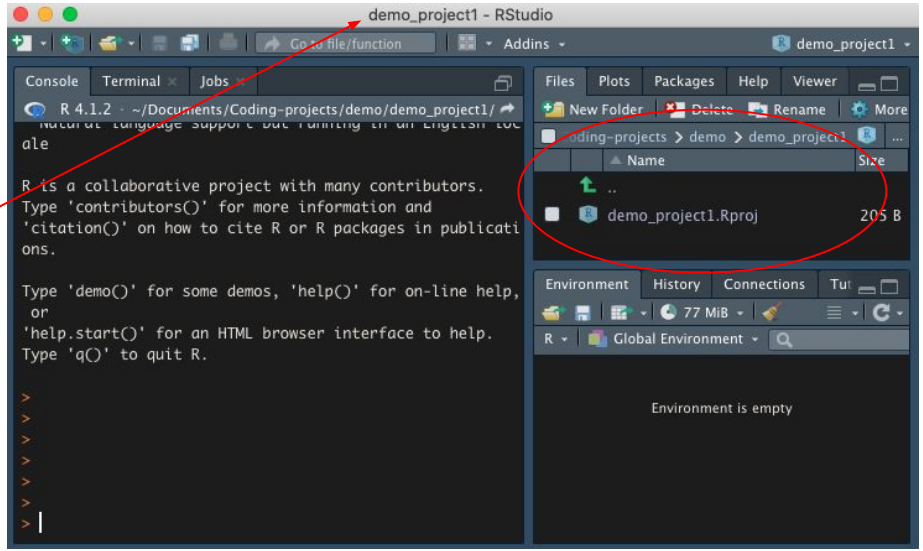
https://www.tidyverse.org/blog/2017/12/workflow-vs-script/

Project-oriented workflow

Use Rstudio / .Rproj for your data analysis projects





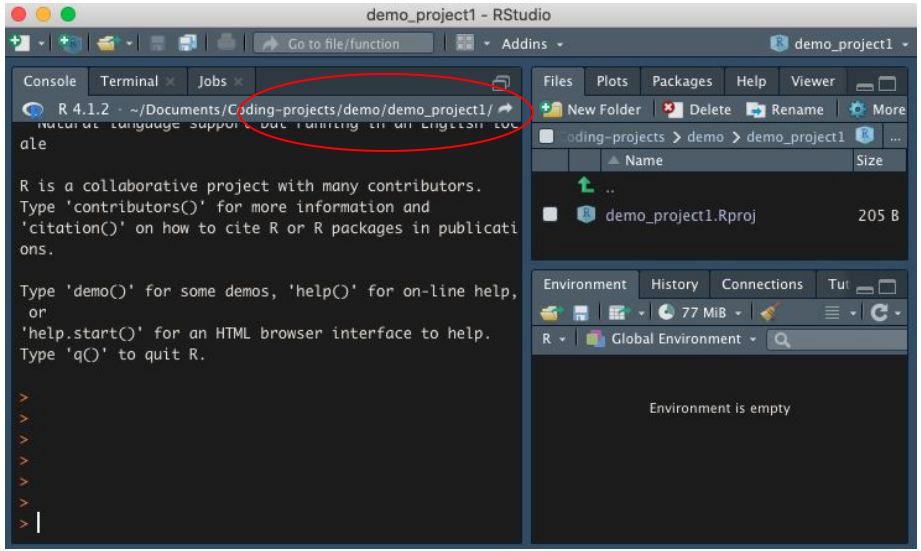


This means that you are essentially compartmentalizing your current project

Project-oriented workflow

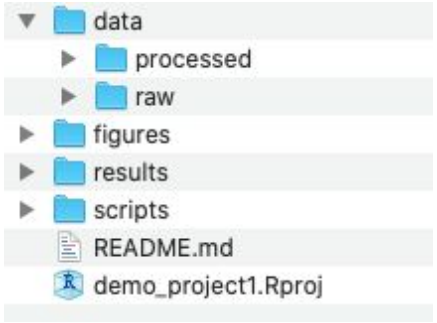
Use Rstudio / .Rproj for your data analysis projects

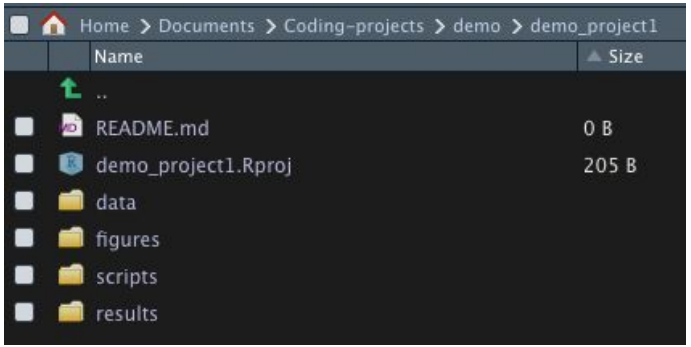
- Project directory stores all your data, scripts
- The working directory is set to the project directory (e.g. demo_project1), so don't need to specify full paths to data (only internal subfolders)



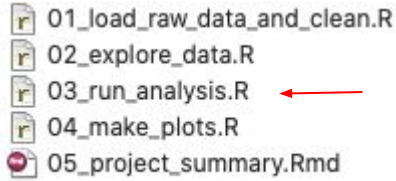
- The project creates everything it needs, within its workspace/folder, and touches nothing it did not create
- Any scripts are written assuming they will be run from a fresh R session within the project
- The project folder can be moved _anywhere_, and everything will still work (no paths will be broken)

Organise your projects intentionally

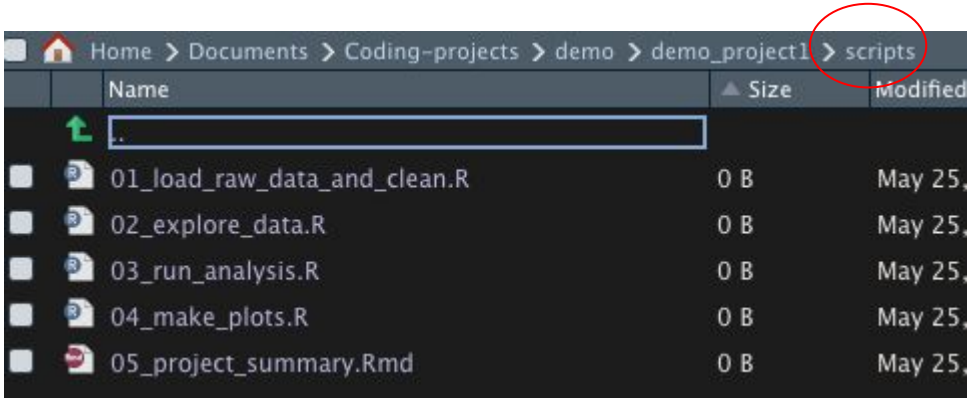




Take advantage of default ordering



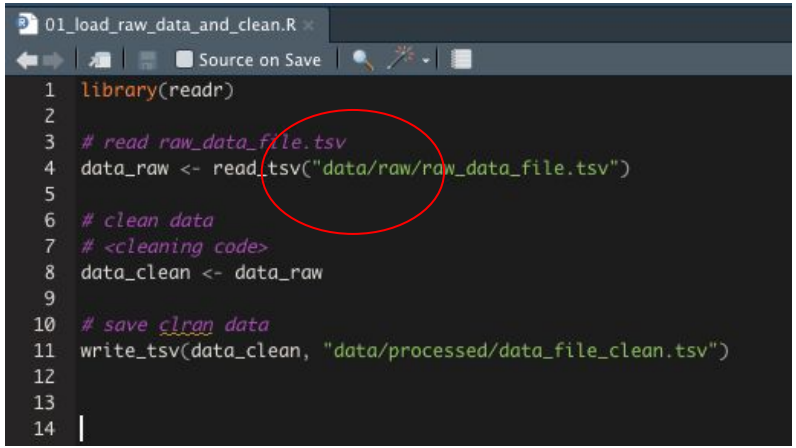
Can have many parts of the analysis separately - save interim results as files and re-read then in the next script

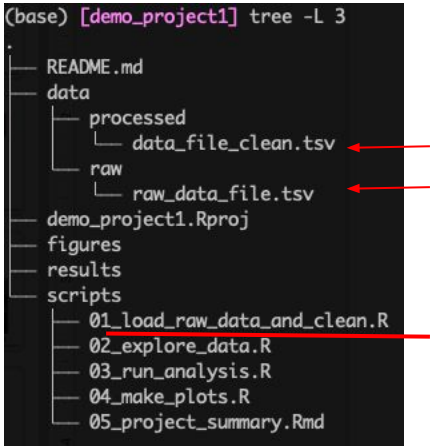


Don't use `setwd()`

Keeping your work as an **.Rproj** will help you manage your file paths

```
setwd("path/that/only/works/on/my/machine")
```



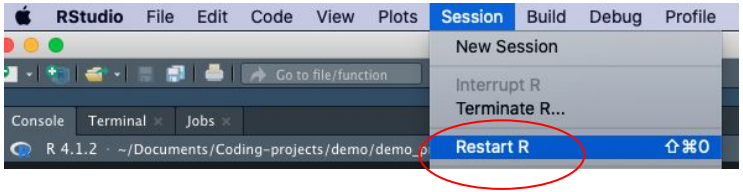


No need to hardcode paths when using Rproj

Don't use `rm(list = ls())`

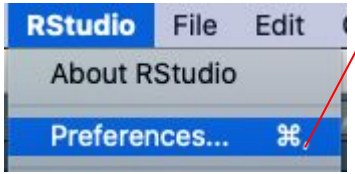
rm(list=ls()) does not fully clean your env!

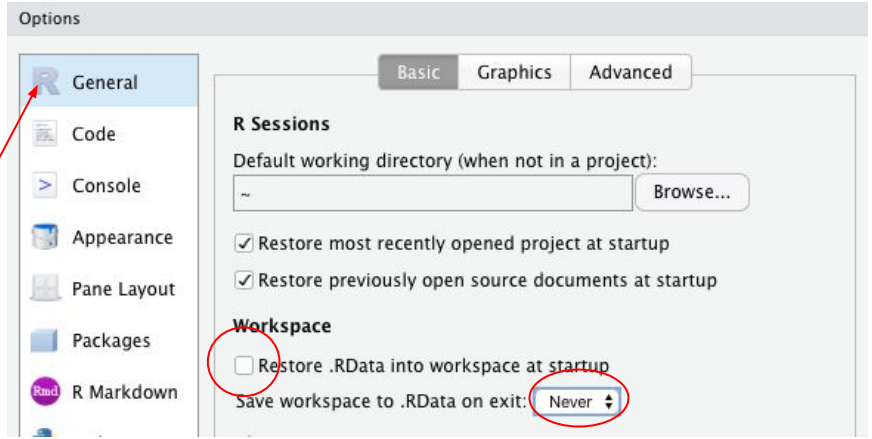
Restart R daily (or every time you start working after a break) to ensure a clean environment



And !! do not save your .Rdata workspace

(untick and select 'never')



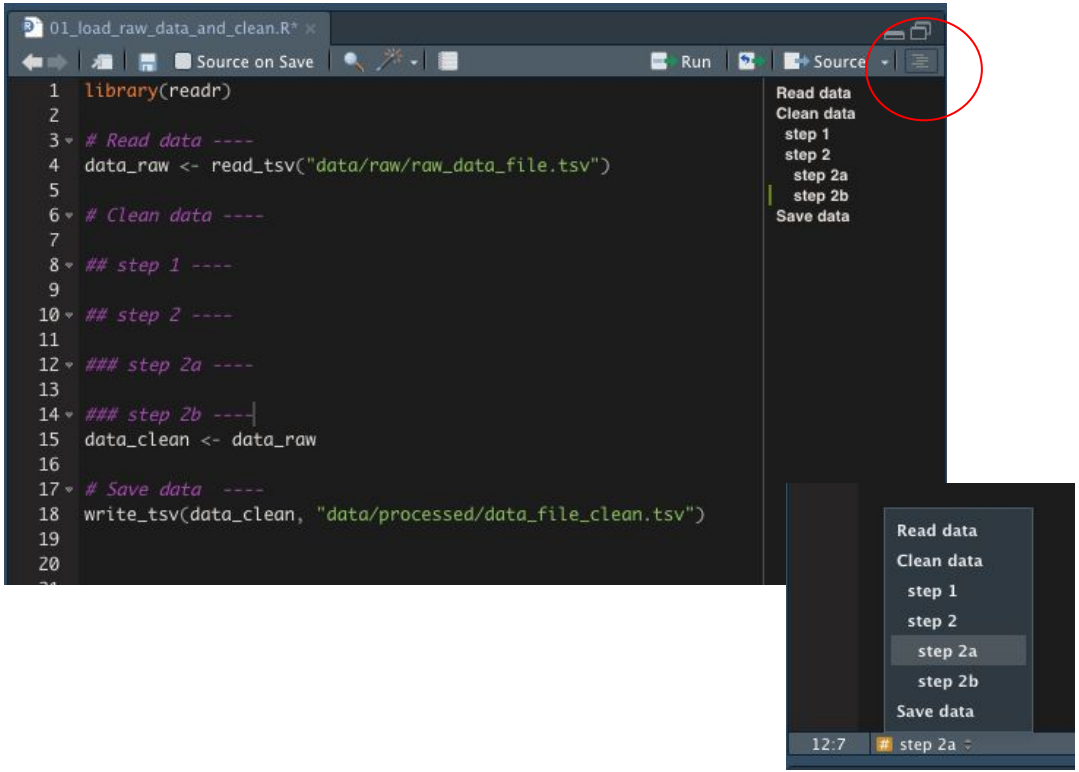


Save your 'real' work, delete the rest

5 mins to try it

2. Rstudio / Rproj efficiency tips

Name your code sections and use then for quick navigation



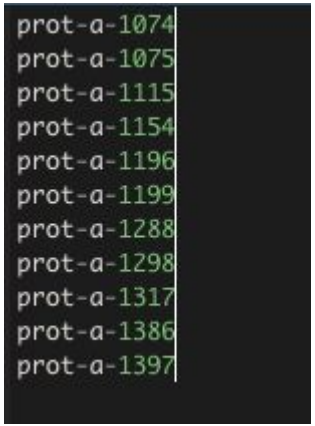
- Use section headings:

```
# section  ----

## subsection  ----

### subsubsection  ----
```

- Great for navigating in long scripts
- Can fold sections

Vertical selection

(hold *option* or *alt* and drag cursor down to select vertically)



Great for e.g.

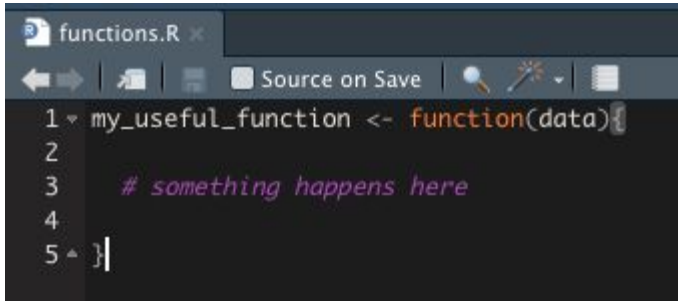
- commenting out a block of code with #
- adding “ “ around a column of ids

Jump to function definition or open data frame

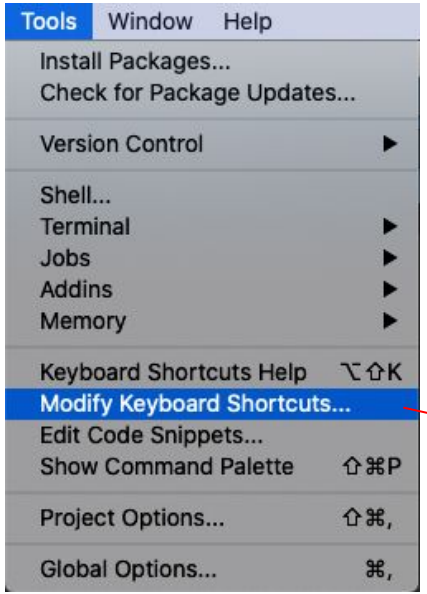
```
## step 2 ----

output <- my_useful_function(input)
```

Cmd + mouse click on the name

(opens in a new window)

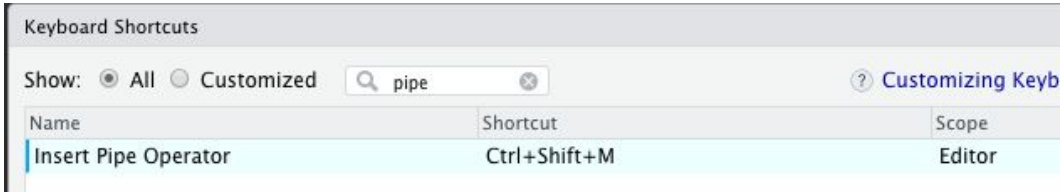


Keyboard shortcuts

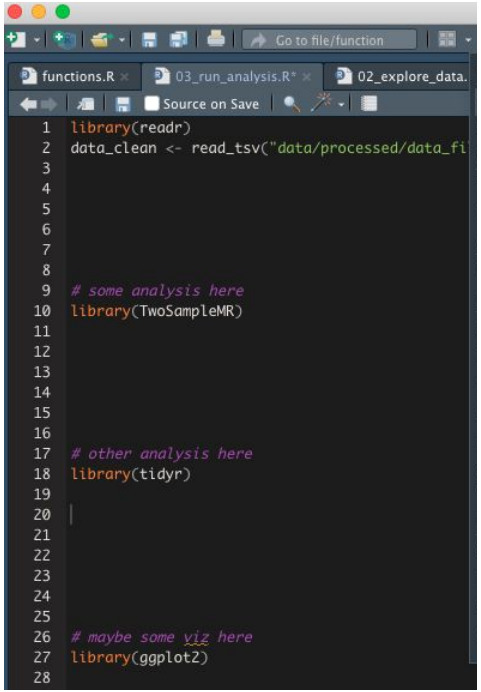


- Run (option + Enter)
- <- (option/alt + " - ")
- %>% (control + shift + M)
- ```{r} (control + shift + I)

  ```


  ```

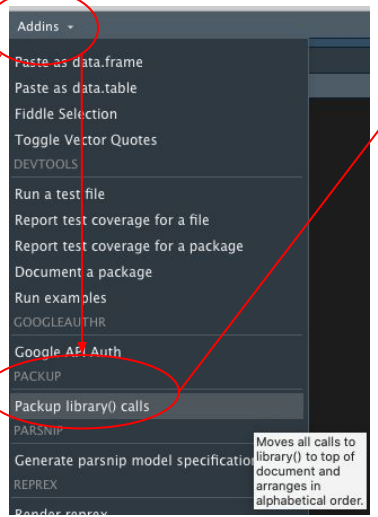
Move all libraries to the top

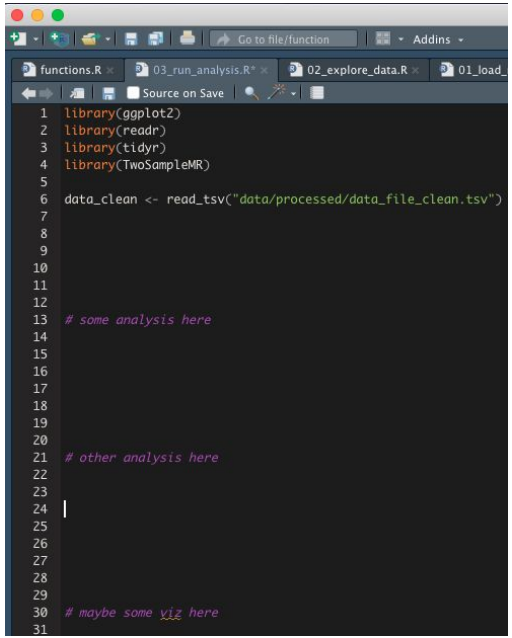


Install *packup* add-in:

```
devtools::install_github("milesmcbain/packup")
```

Call it from Addins menu:





https://github.com/MilesMcBain/packup

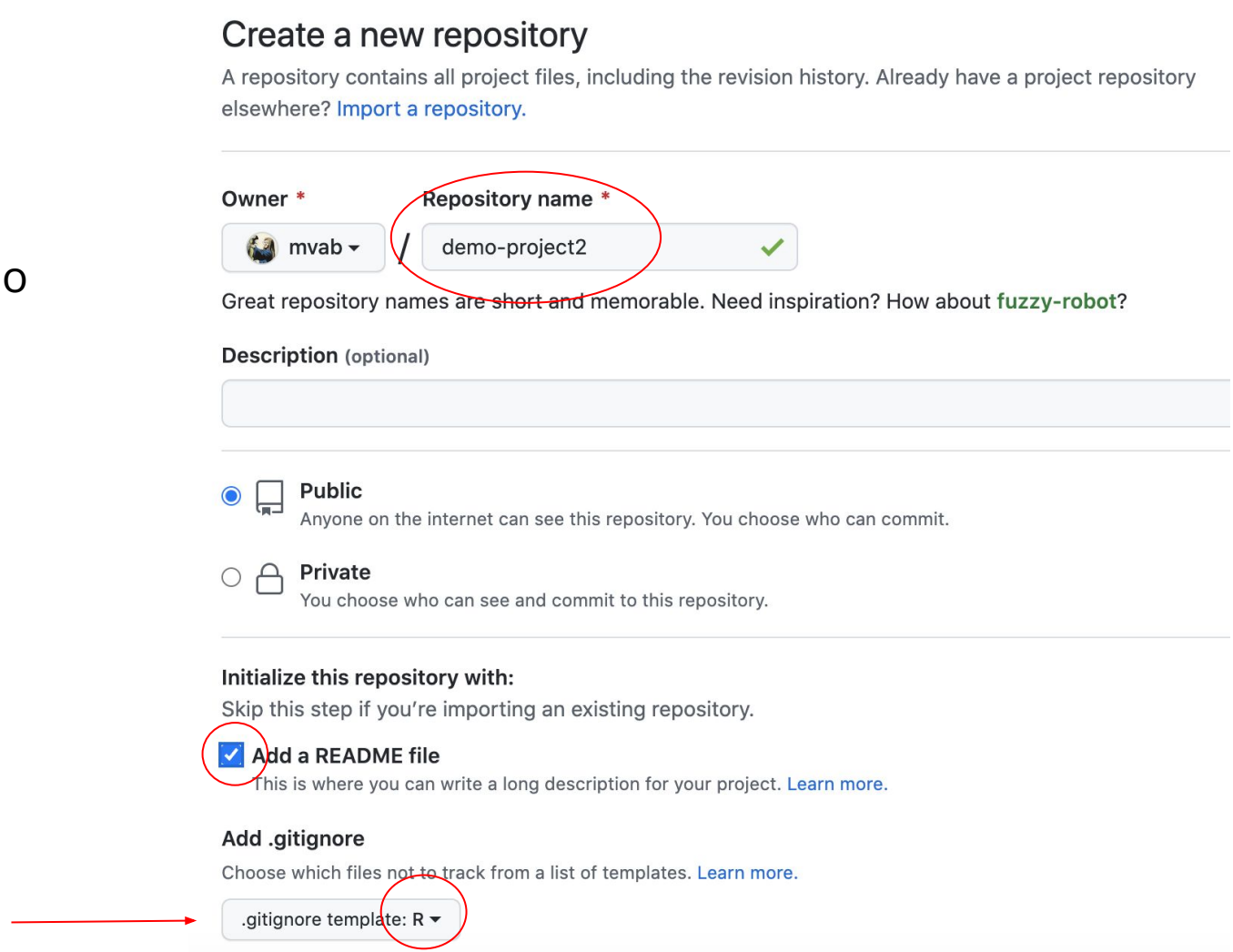
Any other ‘life-saver’ tricks to share?

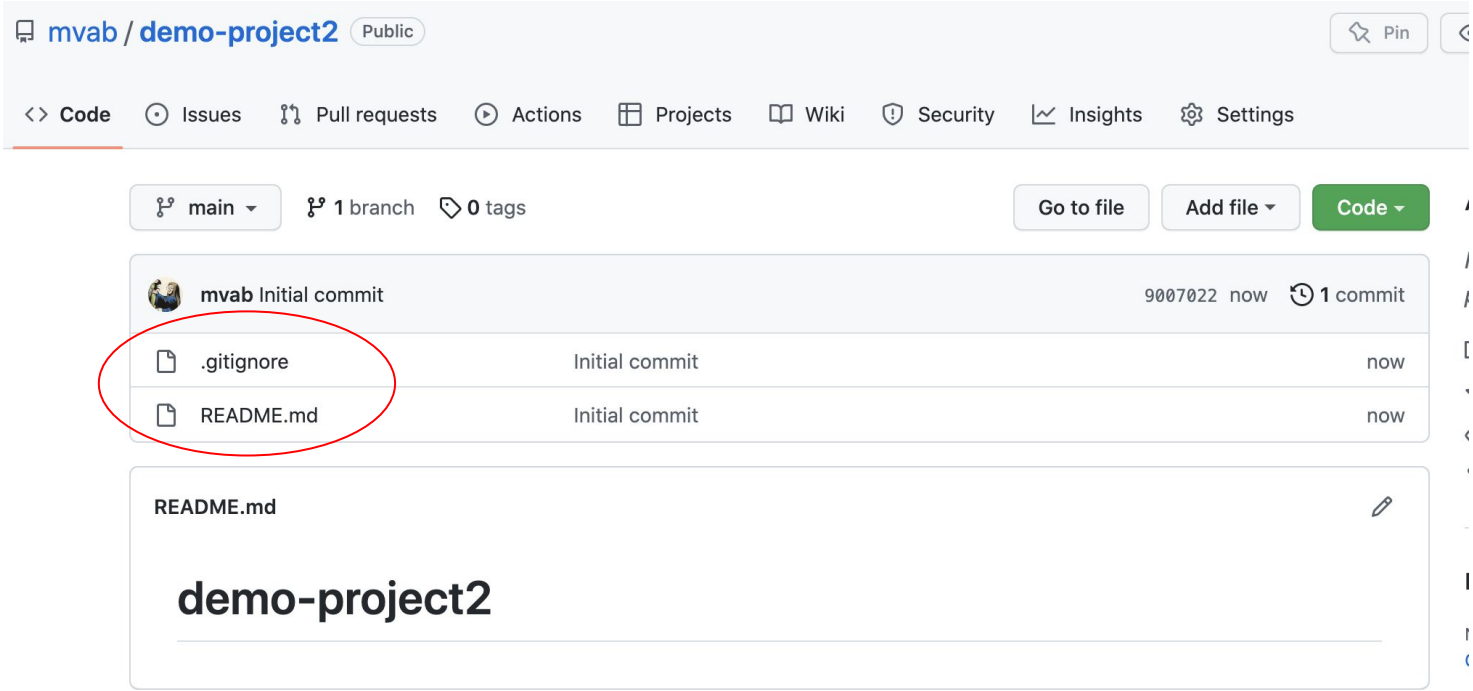
3. Rproj for git users

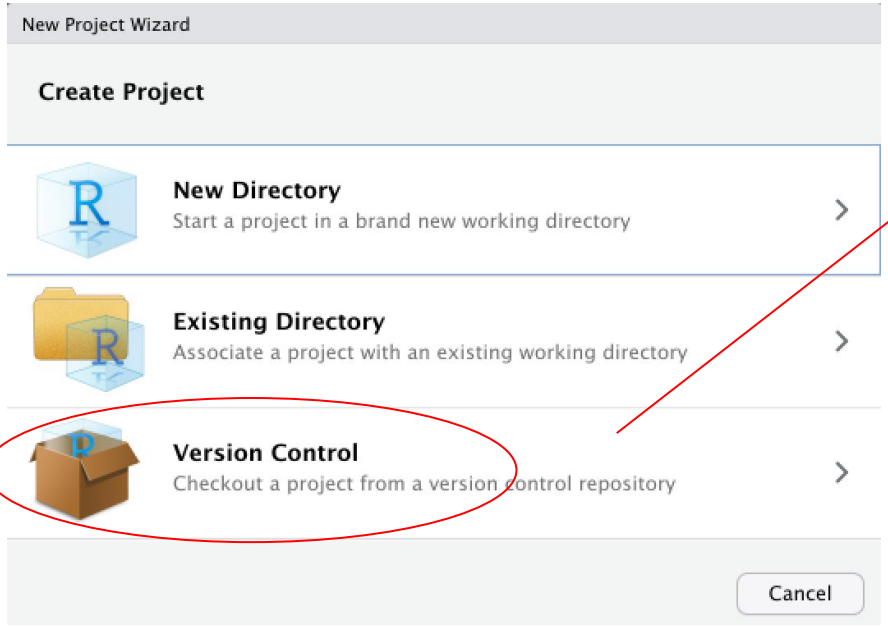
https://happygitwithr.com/

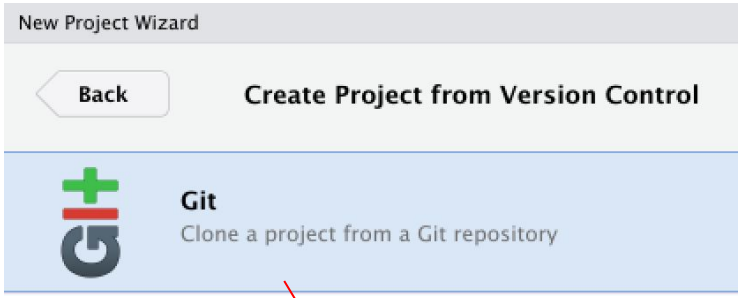
Create new repo on github:

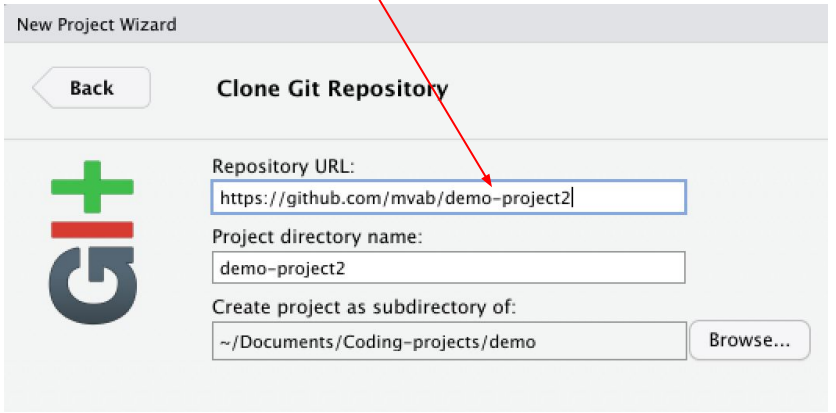
Repositories -> new

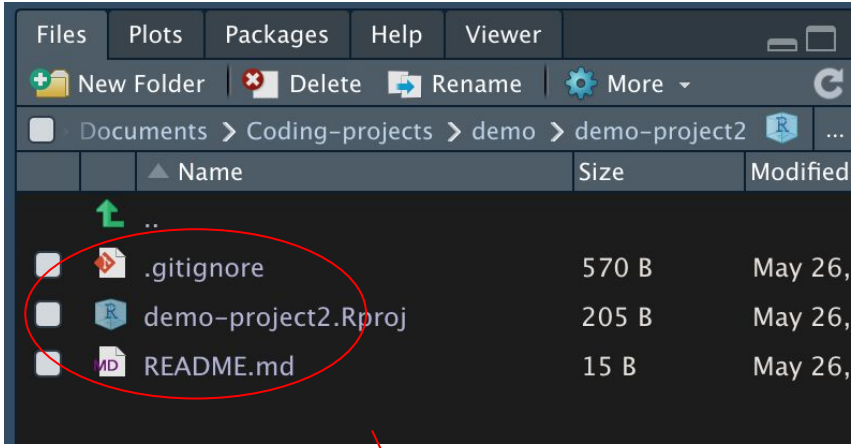



mvab / demo-project2
Public
Pin
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
mvab Initial commit
9007022 now
1 commit
.gitignore
Initial commit
now
README.md
Initial commit
now
README.md
demo-project2


New Project Wizard
Create Project
New Directory
Start a project in a brand new working directory
Existing Directory
Associate a project with an existing working directory
Version Control
Checkout a project from a version control repository
Cancel

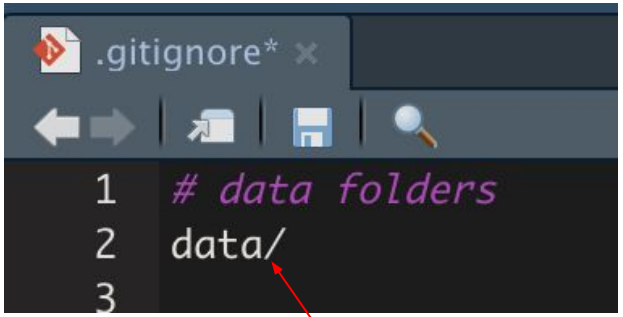

New Project Wizard
Back
Create Project from Version Control
Git
Clone a project from a Git repository


New Project Wizard
Back
Clone Git Repository
Repository URL:
https://github.com/mvab/demo-project2
Project directory name:
demo-project2
Create project as subdirectory of:
~/Documents/Coding-projects/demo
Browse...



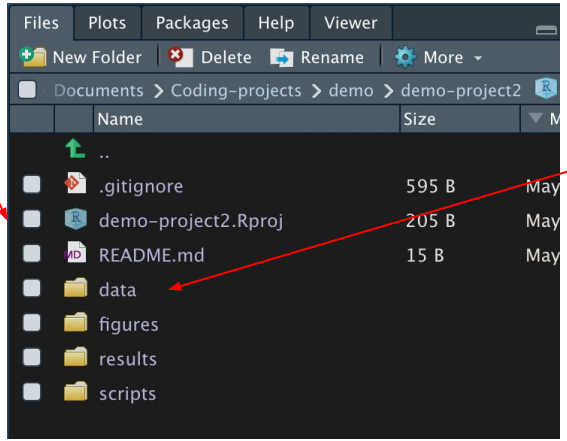
1

.gitignore and *README* come from github; *Rproj* was added by creating an R project



3

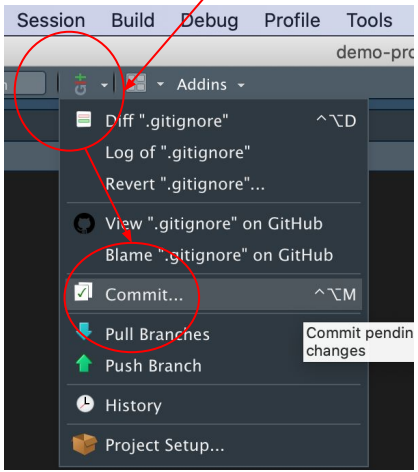
Add **data/** folder to *.gitignore* file so that your data files (if large or sensitive) are not commited to your project repo on Github



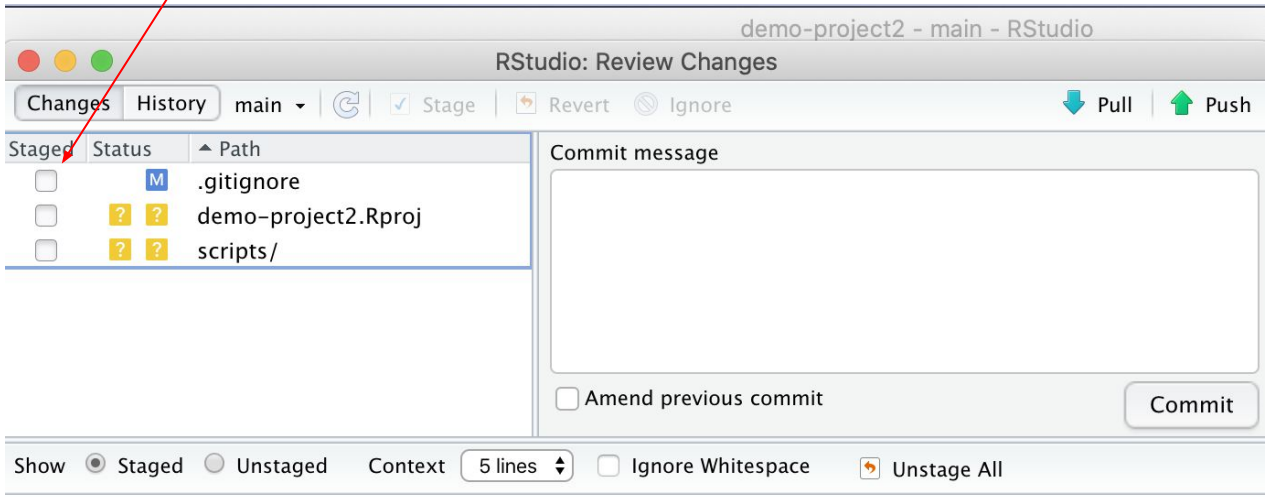
2

Add folders etc

Git button - >
Commit

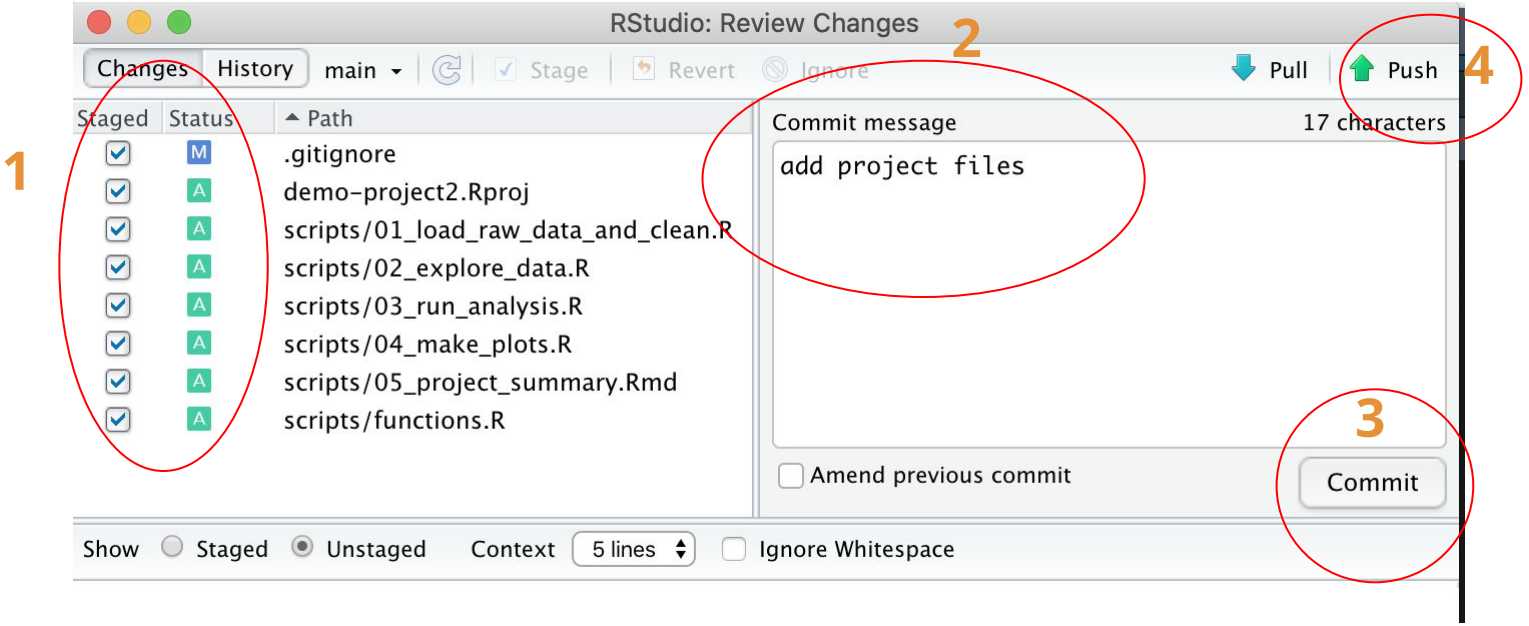


Tick files to commit:

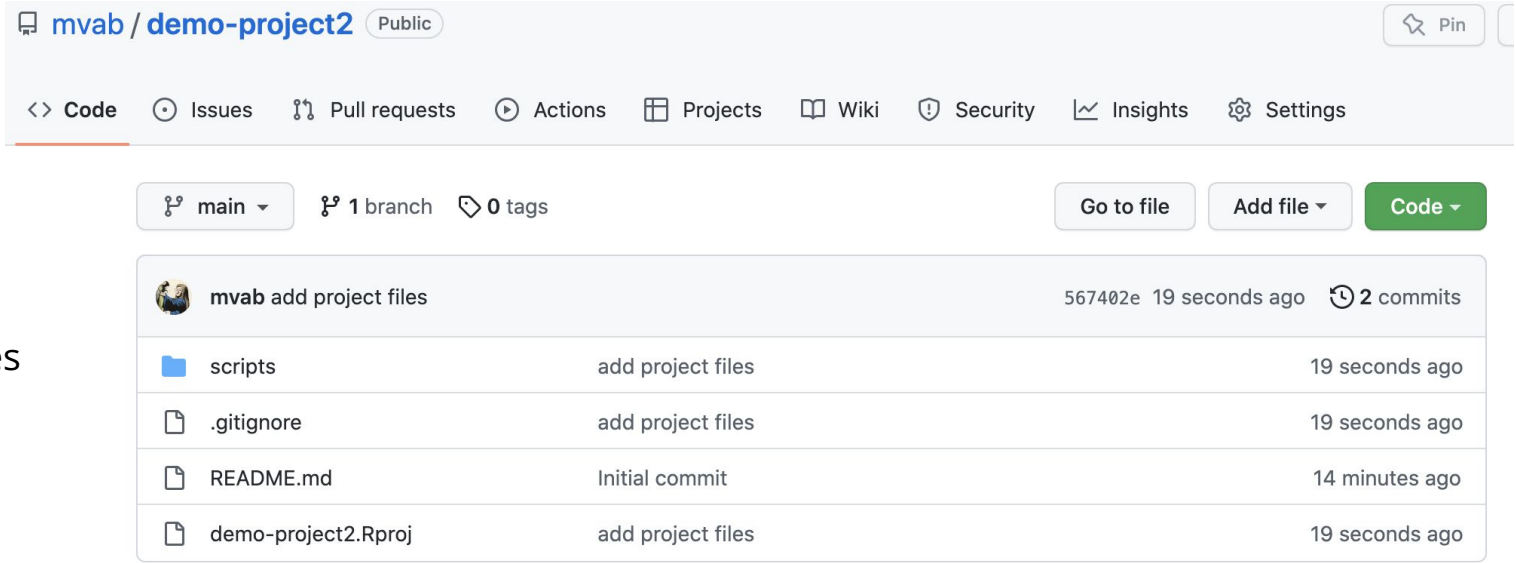


Commit changes:

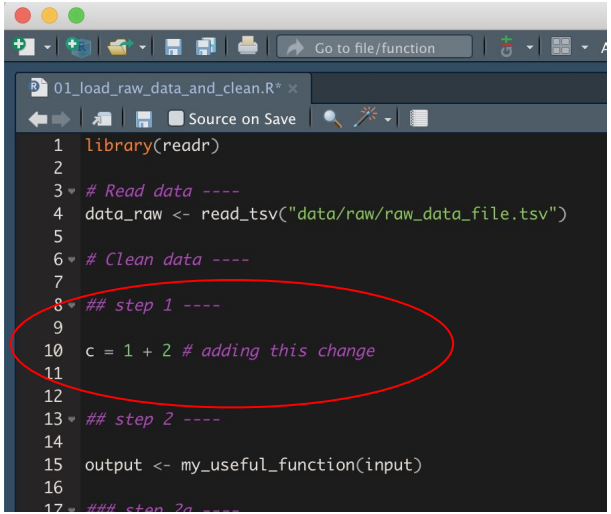
Add message:

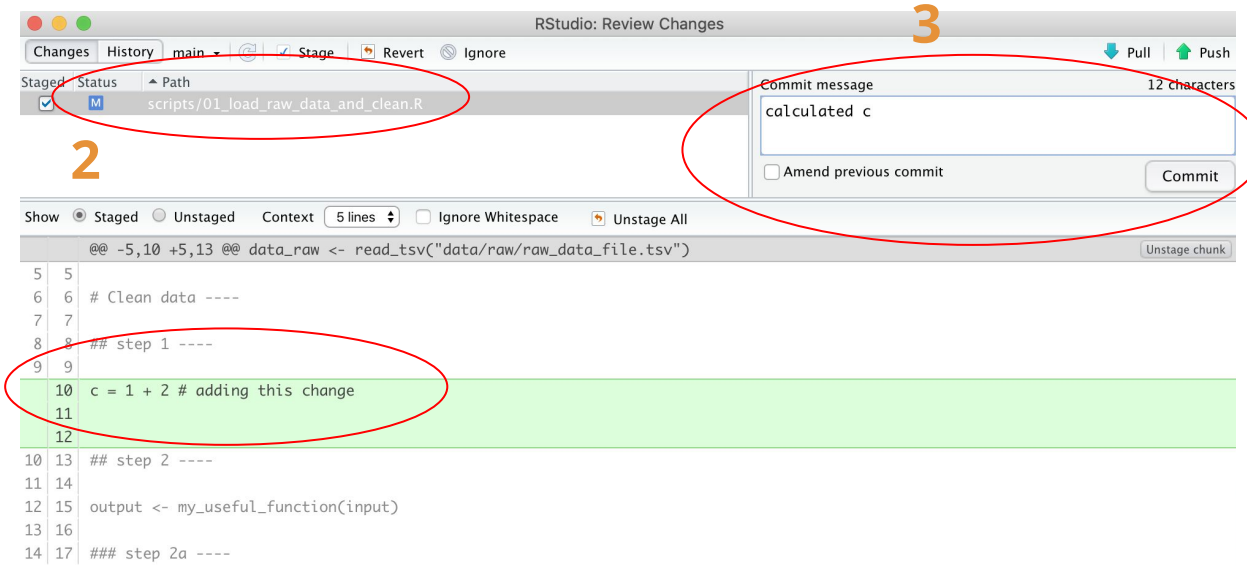


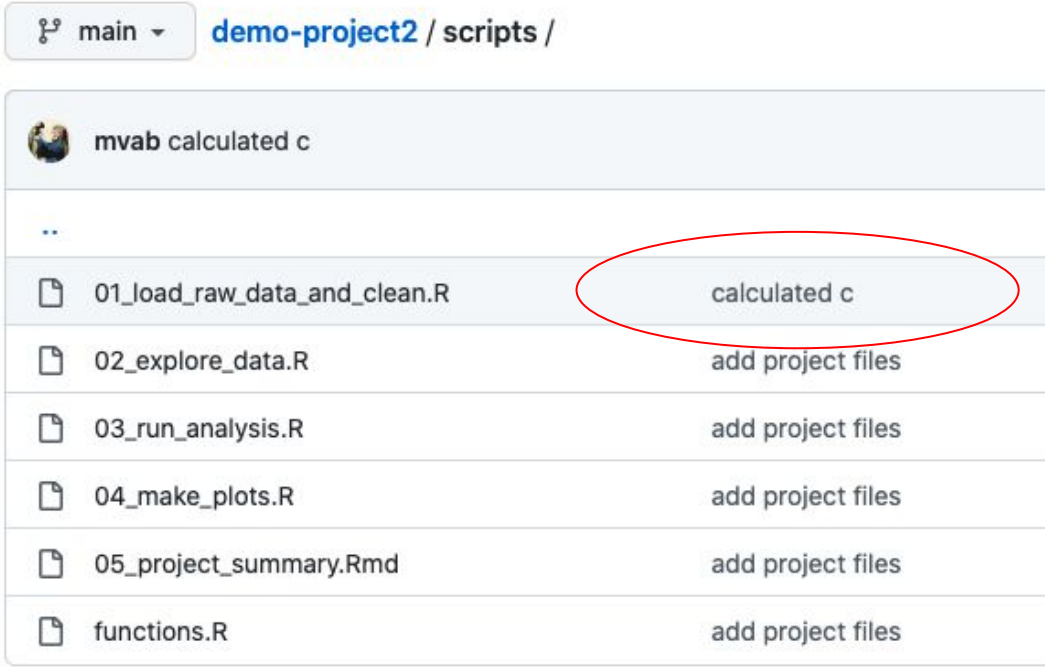
Your changes on Github:



Adding a specific change:






main
demo-project2 / scripts /
mvab calculated c
..
01_load_raw_data_and_clean.R
calculated c
02_explore_data.R
add project files
03_run_analysis.R
add project files
04_make_plots.R
add project files
05_project_summary.Rmd
add project files
functions.R
add project files

5 mins to try it

Using .Rproj for organising work

- “Work in a project” means:
  - **File system discipline:** all files related to a single project are stored in a designated folder;
  - **Working directory discipline**: intentionally work in project directory when opening Rproj
  - **File path discipline:** all paths are relative to the project directory (not hard-coded full paths!)
  - **Daily work habit:** Restarting R very often and re-run your under-development script from the top will help you catch issues early on

- Practising these habits together will give you the biggest pay-off
  - Reproducing your analyses will be easy
  - Organising your projects will help you make sense of them in 6/12/etc months
  - Can move your project anywhere or share it with anyone without changing paths

https://rstats.wtf/project-oriented-workflow.html#work-in-a-project

Final thoughts / disclaimers

- Project-oriented workflow is not suitable/applicable to every scenario
  - Sometimes data is stored externally and can't be/too big to move (so can't use within-project paths)
- Not all work is done interactively in Rstudio
  - Some people use R from the terminal on the server (e.g. BlueCrystal) - again, because of data access/size
  - Some analyses are computation-heavy and require to be submitted as scripts / run in parallel on server

- If your current workflow with `setwd()` works for you and your colleagues, consider future-proofing! ;)

Recommended and used resources

https://www.tidyverse.org/blog/2017/12/workflow-vs-script/

https://richpauloo.github.io/2018-10-17-How-to-keep-your-R-projects-organized/

https://www.rforecology.com/post/organizing-your-r-studio-projects/

https://kkulma.github.io/2018-03-18-Prime-Hints-for-Running-a-data-project-in-R/

https://rstats.wtf/project-oriented-workflow.html

https://appsilon.com/rstudio-shortcuts-and-tips/

https://datacornering.com/my-favorite-rstudio-tips-and-tricks/

https://happygitwithr.com/