

How to organise your projects with Rstudio

IEU Introduction to R (part 2)
Marina Vabistsevits

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*



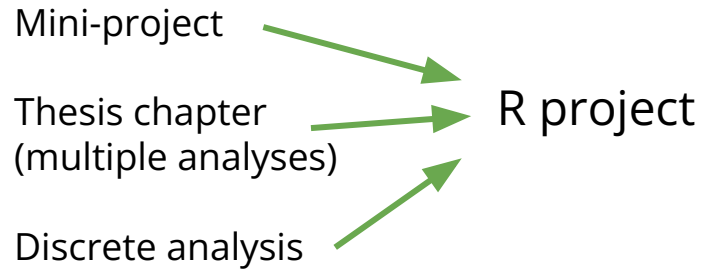
Icon for R



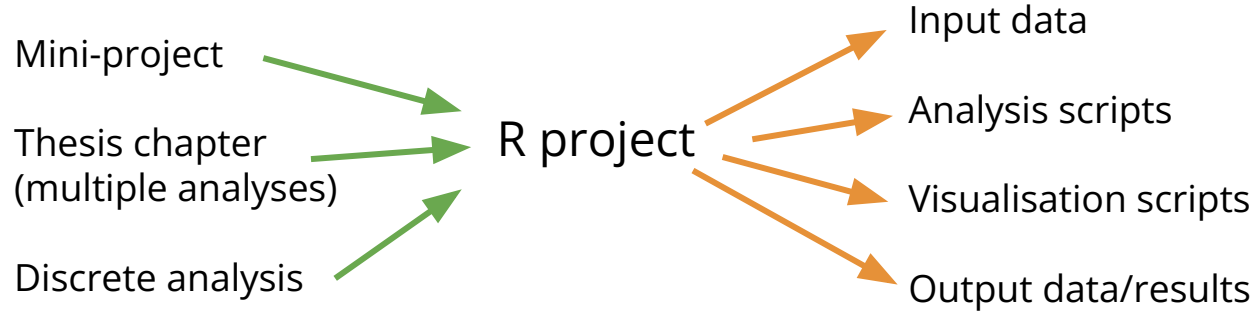
Icon for RStudio

1. Project-oriented workflow

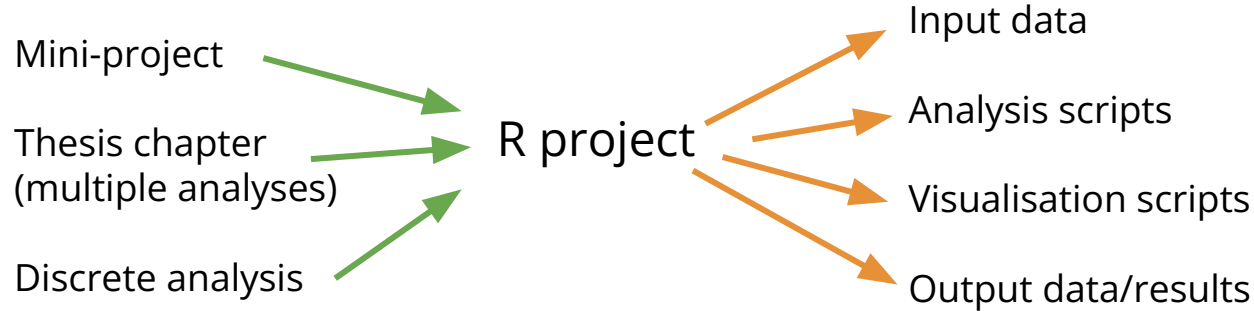
R projects



R projects



R projects



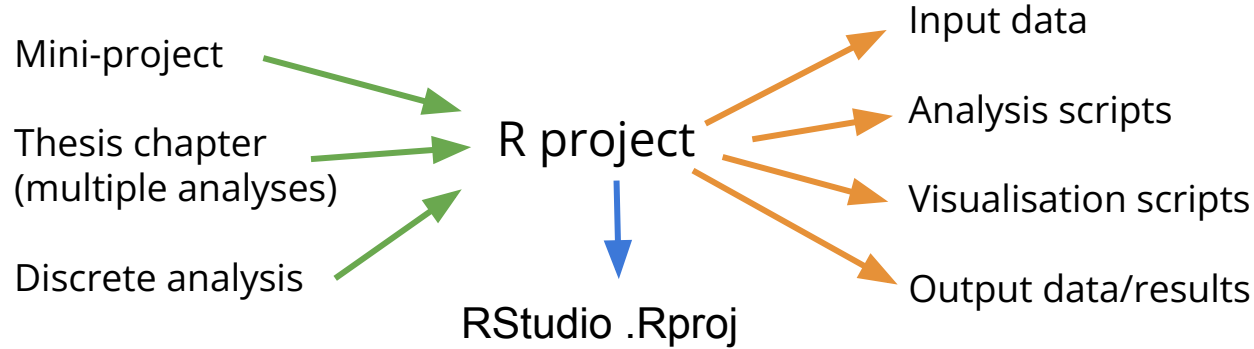
How you do organise your projects? scripts? folders?

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())
```

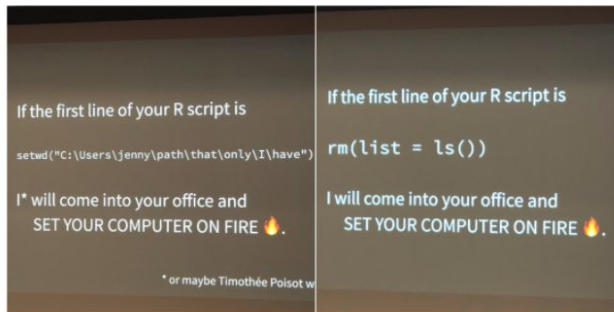
I will come into your office and SET YOUR COMPUTER ON FIRE 🔥.

Twitter profile card for Jenny Bryan (@JennyBryan). It includes her profile picture, a 'Following' button, and her bio: 'Software engineer @rstudio, humane #rstats, adjunct prof @UBC where I created @STAT545, part of @ropensci, she/her'. Location is 'Vancouver, BC' and website is 'jennybryan.org'.

Hadley Wickham
@hadleywickham

Follow

The only two things that make @JennyBryan 🤔😡💩. Instead use projects + here::here() #rstats



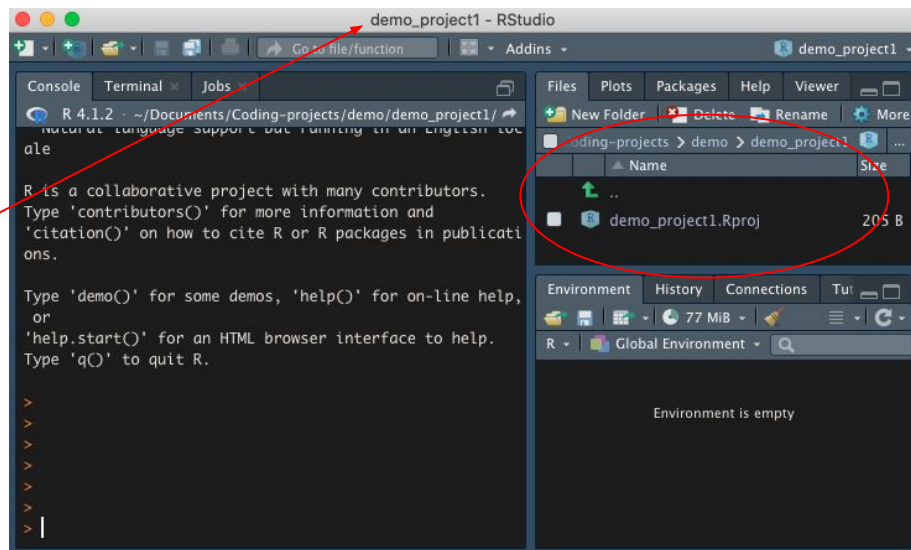
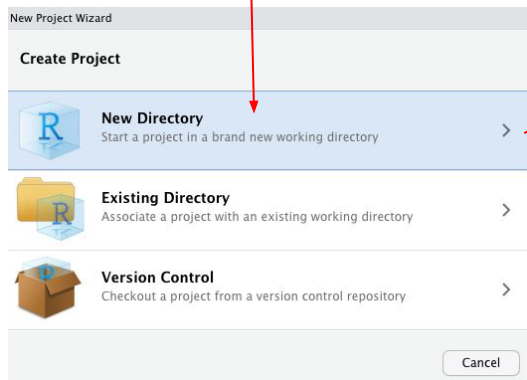
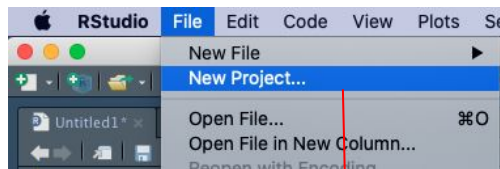
4:50 PM - 10 Dec 2017

290 Retweets 950 Likes



Project-oriented workflow

Use Rstudio / .Rproj for your data analysis projects



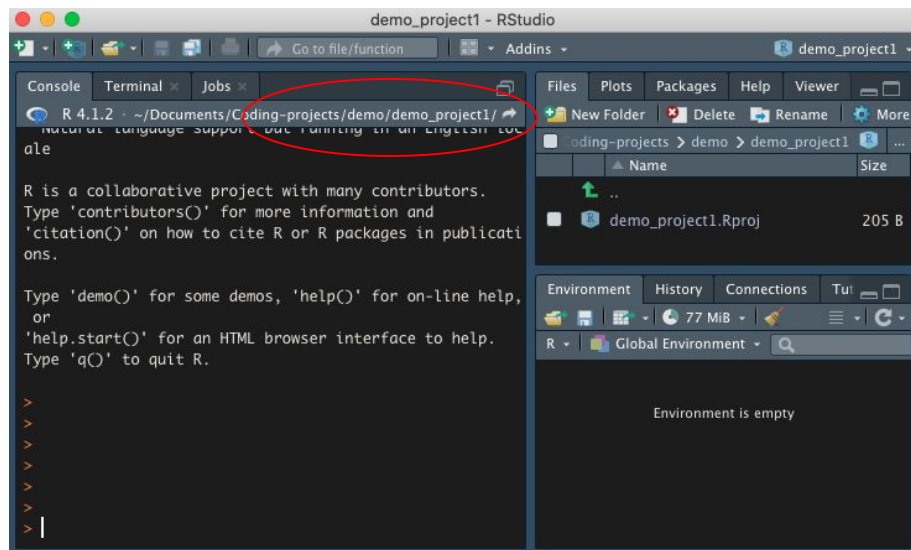
This means that you are essentially compartmentalizing your current project



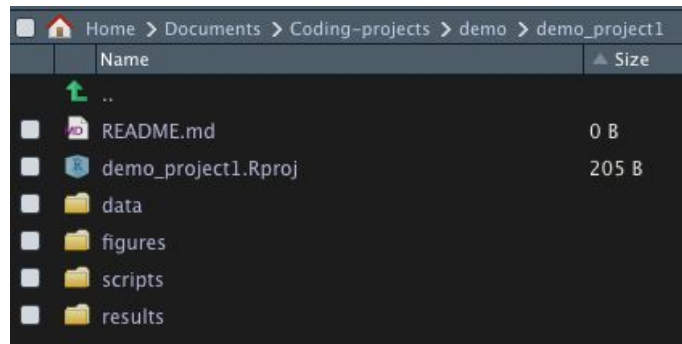
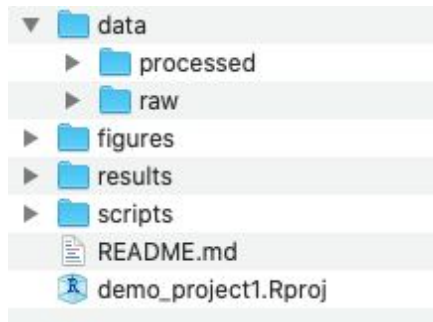
Project-oriented workflow

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)



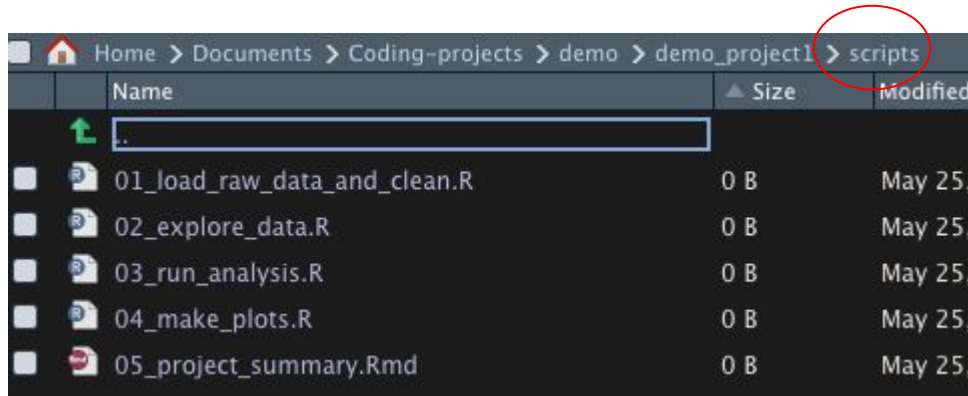
Organise your projects intentionally



Take advantage of default ordering

- 01_load_raw_data_and_clean.R
- 02_explore_data.R
- 03_run_analysis.R
- 04_make_plots.R
- 05_project_summary.Rmd

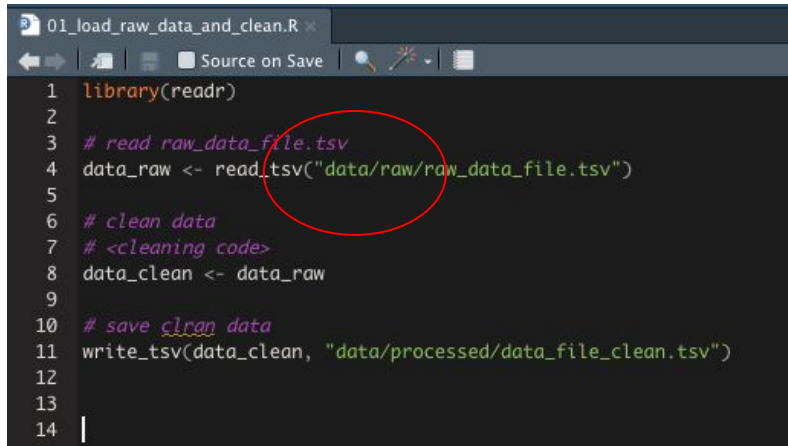
Can have many parts of the analysis separately - save interim results as files and re-read then in the next script



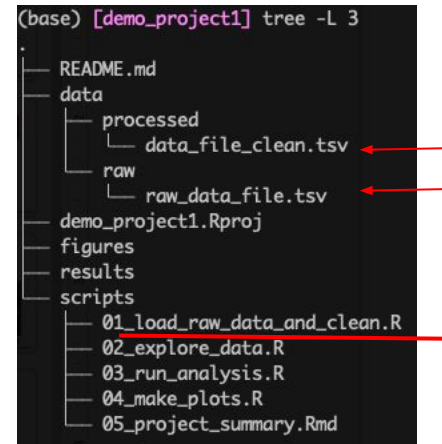
Don't use `setwd()`

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

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



```
01_load_raw_data_and_clean.R
1 library(readr)
2
3 # read raw_data_file.tsv
4 data_raw <- read_tsv("data/raw/raw_data_file.tsv")
5
6 # clean data
7 # <cleaning code>
8 data_clean <- data_raw
9
10 # save clean data
11 write_tsv(data_clean, "data/processed/data_file_clean.tsv")
12
13
14
```



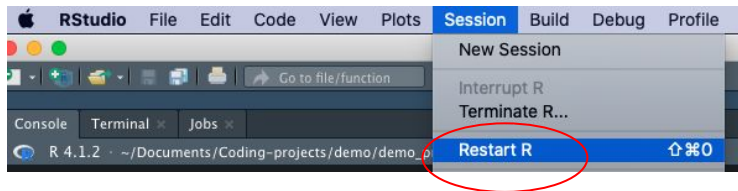
```
(base) [demo_project1] tree -L 3
.
├── README.md
├── data
│   ├── processed
│   │   └── data_file_clean.tsv
│   └── raw
│       └── raw_data_file.tsv
├── demo_project1.Rproj
├── figures
├── results
├── scripts
│   ├── 01_load_raw_data_and_clean.R
│   ├── 02_explore_data.R
│   ├── 03_run_analysis.R
│   ├── 04_make_plots.R
│   └── 05_project_summary.Rmd
```

No need to hardcode paths when using Rproj

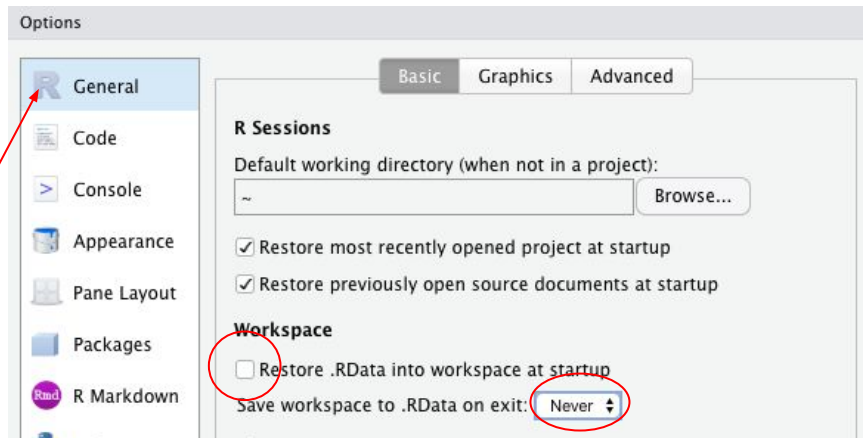
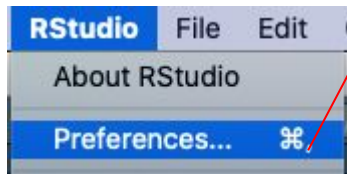
Don't use `rm(list = ls())`

`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')

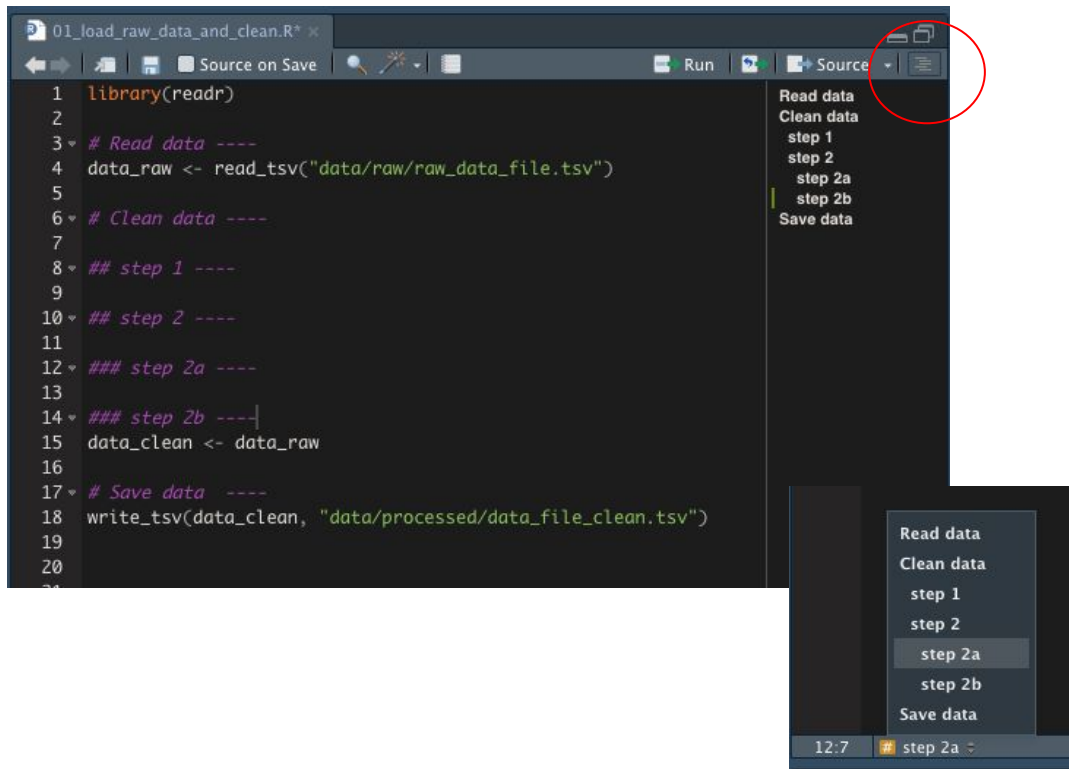


Save your 'real' work, delete the rest

5 mins to try it

2. Rstudio / Rproj efficiency tips

Name your code sections and use them for quick navigation



- Use section headings:

section ----

subsection ----

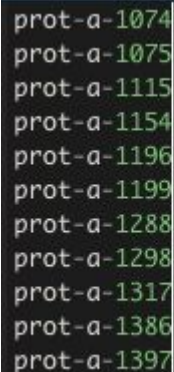
subsubsection ----

- Great for navigating in long scripts
- Can fold sections



Vertical selection

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



```
prot-a-1074  
prot-a-1075  
prot-a-1115  
prot-a-1154  
prot-a-1196  
prot-a-1199  
prot-a-1288  
prot-a-1298  
prot-a-1317  
prot-a-1386  
prot-a-1397
```

A screenshot of a code editor with a dark background. It shows a list of identifiers, each on a new line: 'prot-a-1074', 'prot-a-1075', 'prot-a-1115', 'prot-a-1154', 'prot-a-1196', 'prot-a-1199', 'prot-a-1288', 'prot-a-1298', 'prot-a-1317', 'prot-a-1386', and 'prot-a-1397'. A vertical selection is active, indicated by a thin white line and a light blue highlight, covering the ID numbers (1074, 1075, 1115, 1154, 1196, 1199, 1288, 1298, 1317, 1386, 1397) in the last column of each line.

Great for e.g.

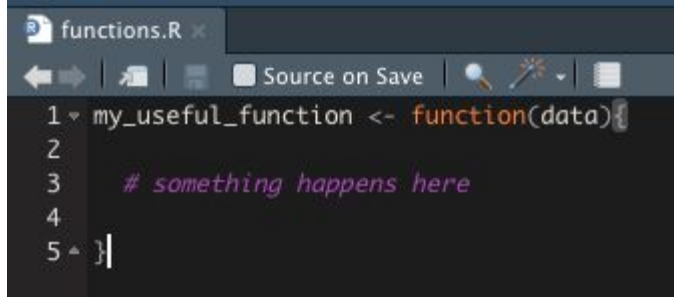
- commenting out a block of code with #
- adding " " around a column of ids

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)





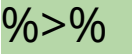
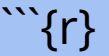
The screenshot shows an RStudio editor window titled 'functions.R'. The code editor contains the following R code:

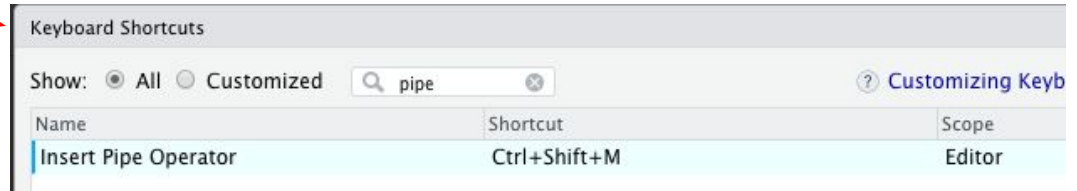
```
1 my_useful_function <- function(data){  
2  
3   # something happens here  
4  
5 }
```

The code is syntax-highlighted, with the function name 'my_useful_function' in orange, the keyword 'function' in orange, and the comment '# something happens here' in purple. The cursor is at the end of line 5.

Keyboard shortcuts



-  (option + Enter)
-  (option/alt + " - ")
-  (control + shift + M)
-  (control + shift + I)

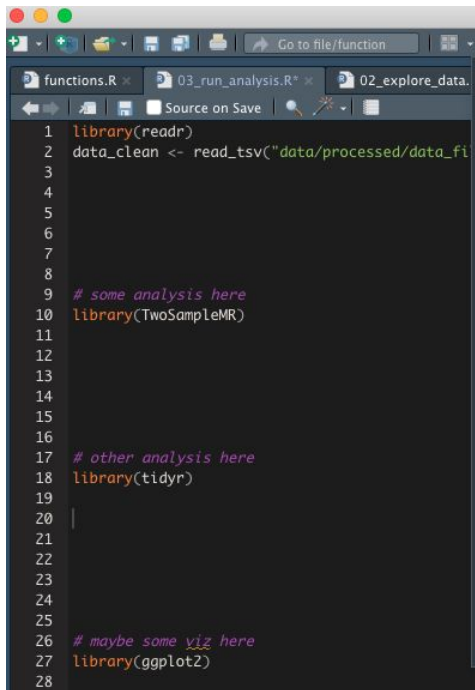
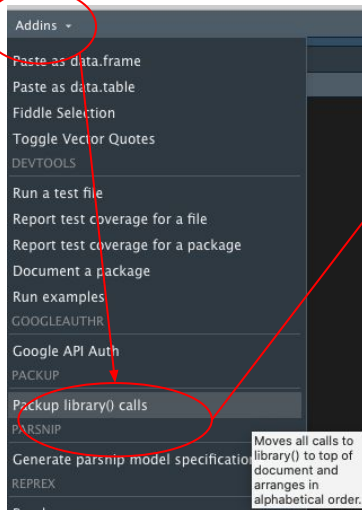


Move all libraries to the top

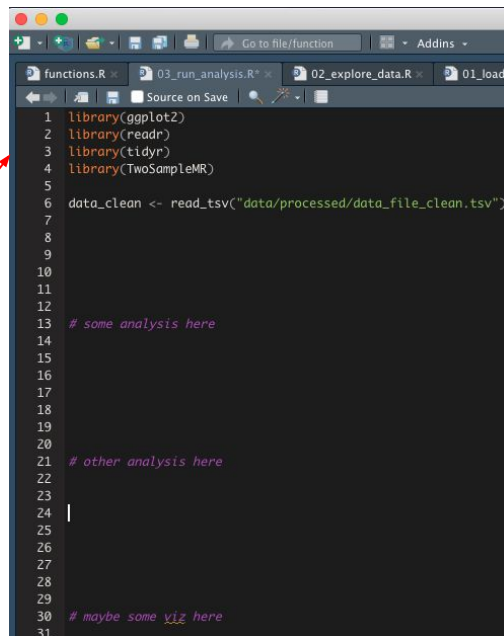
Install *packup* add-in:

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

Call it from Addins menu:



```
1 library(readr)
2 data_clean <- read_tsv("data/processed/data_file_clean.tsv")
3
4
5
6
7
8
9 # some analysis here
10 library(TwoSampleMR)
11
12
13
14
15
16
17 # other analysis here
18 library(tidyr)
19
20
21
22
23
24
25
26 # maybe some viz here
27 library(ggplot2)
28
```



```
1 library(ggplot2)
2 library(readr)
3 library(tidyr)
4 library(TwoSampleMR)
5
6 data_clean <- read_tsv("data/processed/data_file_clean.tsv")
7
8
9
10
11
12
13 # some analysis here
14
15
16
17
18
19
20
21 # other analysis here
22
23
24
25
26
27
28
29
30 # maybe some viz here
31
```

Any other 'life-saver' tricks to share?

3. Rproj for git users

<https://happygitwithr.com/>

Create new repo on github:

Repositories -> new

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *



mvab ▾

Repository name *

/ demo-project2 ✓

Great repository names are short and memorable. Need inspiration? How about **fuzzy-robot?**

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.



Add a README file

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: R ▾

main 1 branch 0 tags

Go to file

Add file

Code



mvab Initial commit

9007022 now 1 commit



.gitignore

Initial commit

now



README.md

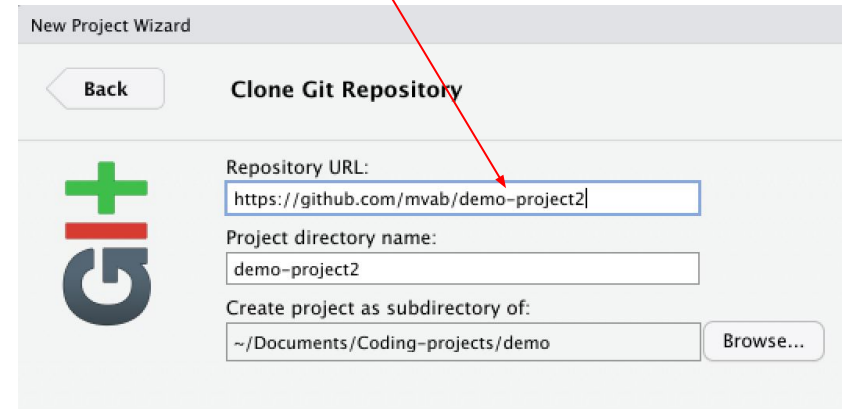
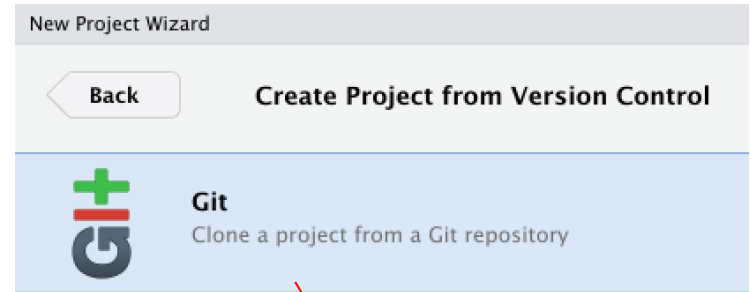
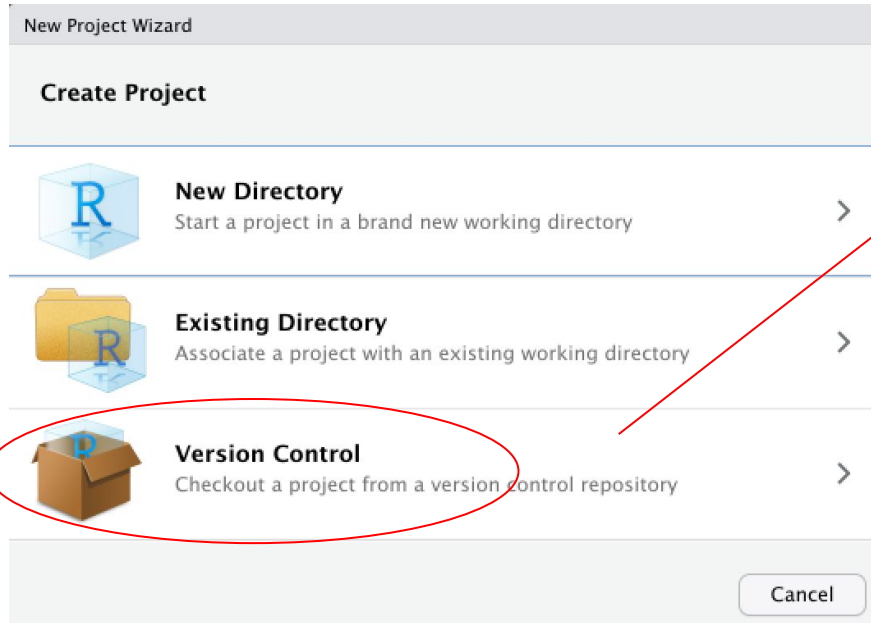
Initial commit

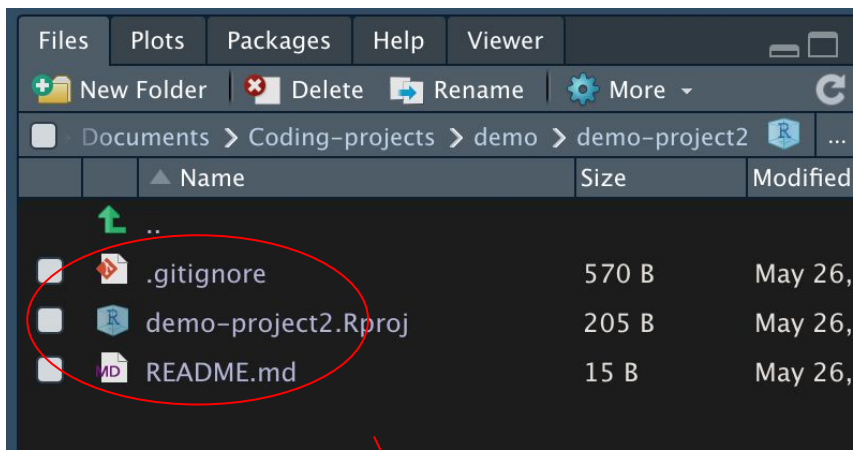
now

README.md



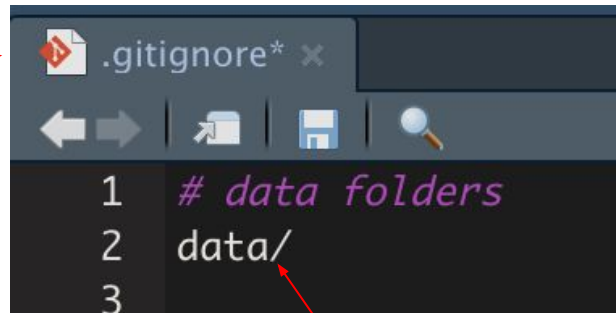
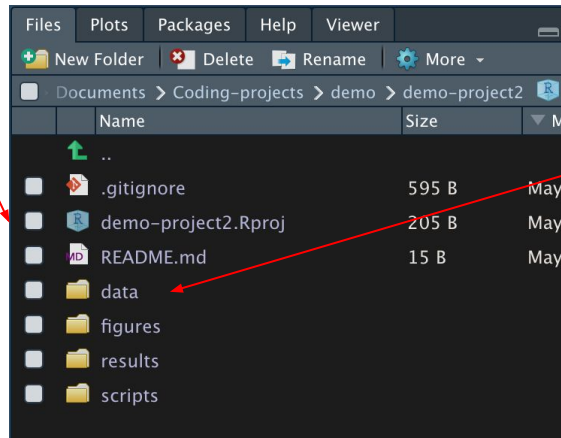
demo-project2





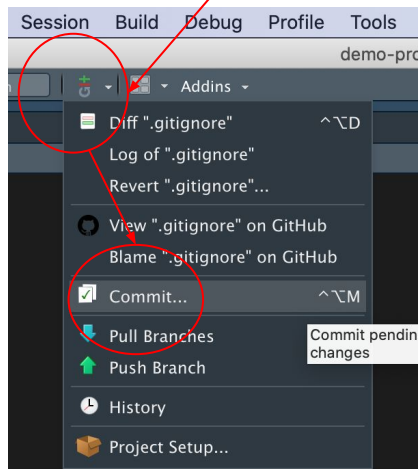
.gitignore and *README* come from github;
Rproj was added by creating an R project

2 Add folders etc

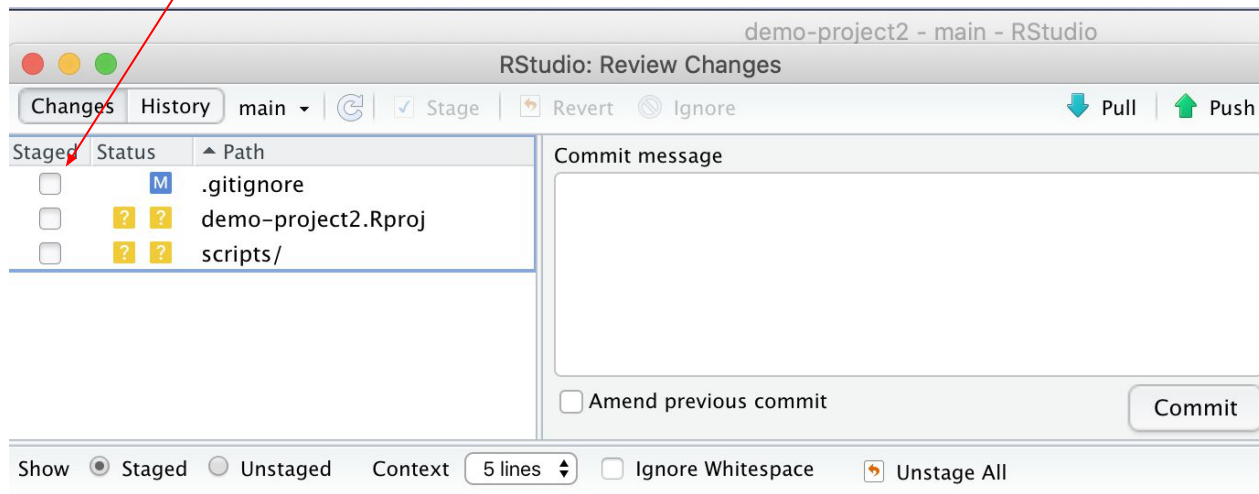


Add **data/** folder to *.gitignore* file so that your data files (if large or sensitive) are not committed to your project repo on Github

Git button ->
Commit

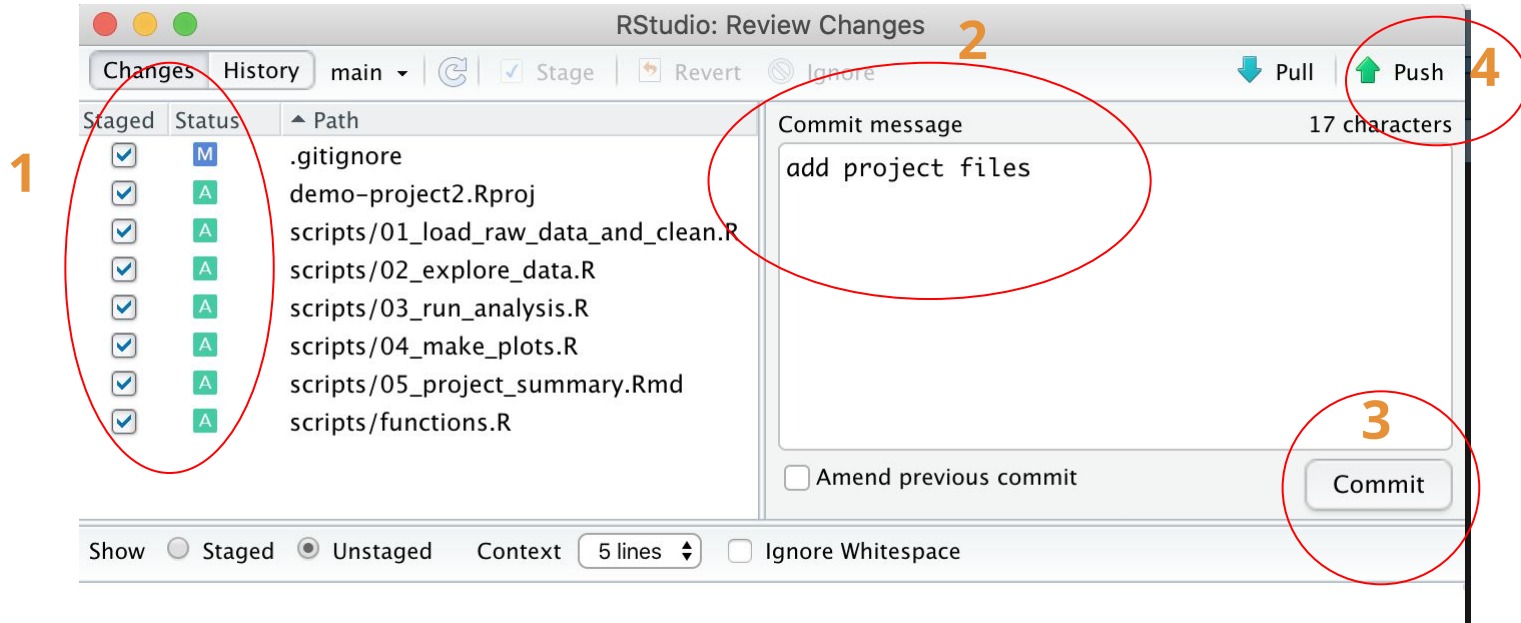


Tick files to commit:




Commit changes:

Add message:




Your changes
on Github:





 **mvab / demo-project2** Public

[Pin](#)

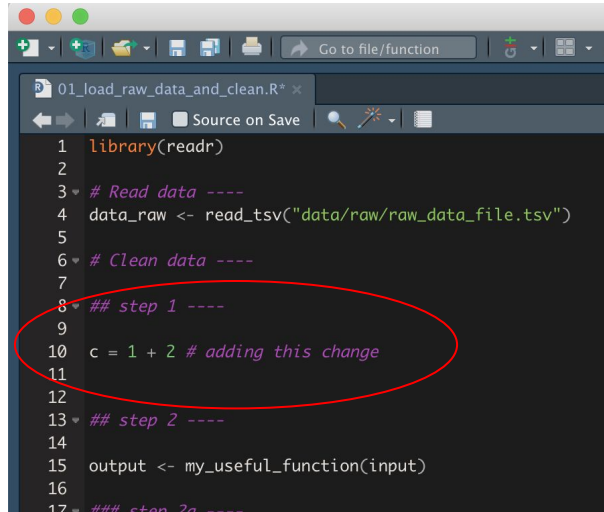
[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

[main](#) [1 branch](#) [0 tags](#) [Go to file](#) [Add file](#) [Code](#)

 **mvab** add project files 567402e 19 seconds ago 🕒 2 commits

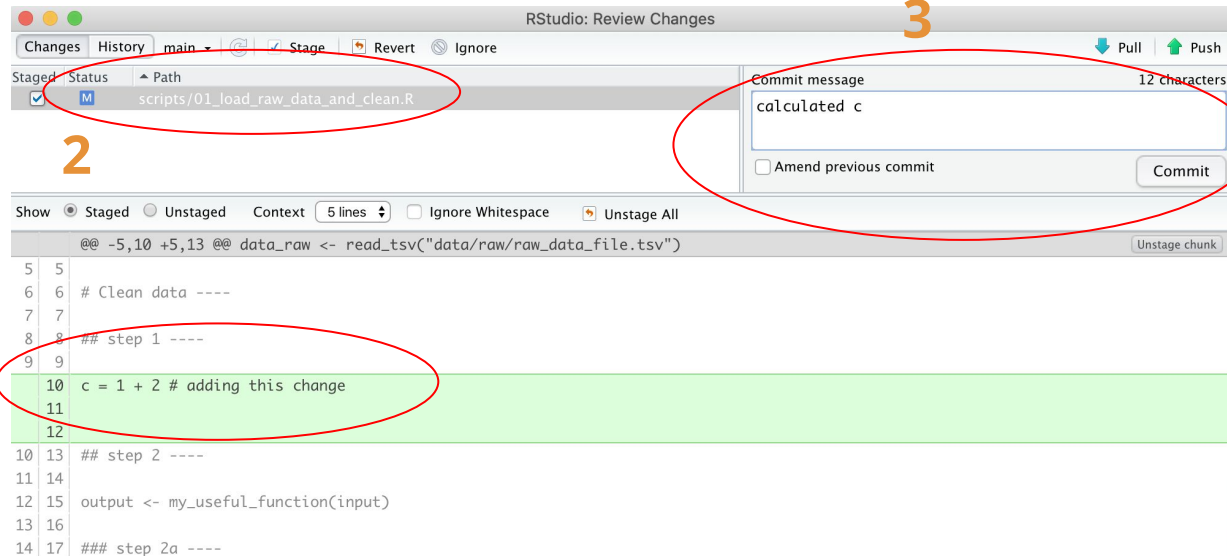
 scripts	add project files	19 seconds ago
 .gitignore	add project files	19 seconds ago
 README.md	Initial commit	14 minutes ago
 demo-project2.Rproj	add project files	19 seconds ago

Adding a specific change:



1 `library(readr)`
2
3 `# Read data ----`
4 `data_raw <- read_tsv("data/raw/raw_data_file.tsv")`
5
6 `# Clean data ----`
7
8 `## step 1 ----`
9
10 `c = 1 + 2 # adding this change`
11
12
13 `## step 2 ----`
14
15 `output <- my_useful_function(input)`
16
17 `### step 2a ----`

The line `c = 1 + 2 # adding this change` is circled in red.



RStudio: Review Changes

Changes History main Stage Revert Ignore Pull Push

Staged Status Path
[x] M scripts/01_load_raw_data_and_clean.R

Commit message 12 characters
calculated c
☐ Amend previous commit Commit

Show Staged Unstaged Context 5 lines Ignore Whitespace Unstage All

@@ -5,10 +5,13 @@ data_raw <- read_tsv("data/raw/raw_data_file.tsv")

5 5
6 6 # Clean data ----
7 7
8 8 ## step 1 ----
9 9
10 10 c = 1 + 2 # adding this change
11 11
12 12
10 13 ## step 2 ----
11 14
12 15 output <- my_useful_function(input)
13 16
14 17 ### step 2a ----

The file path `scripts/01_load_raw_data_and_clean.R` is circled in red. The commit message `calculated c` is circled in red. The change `c = 1 + 2 # adding this change` is circled in red.

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

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/>