Organising your projects in Rstudio for reproducibility

Marina Vabistsevits

What this session is about

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

Icon for R

Icon for RStudio

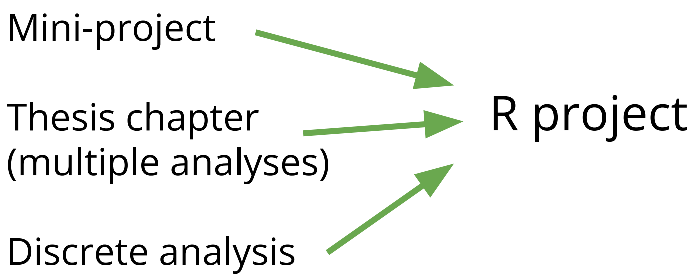
software company who develop Rstudio

We are going to try things out in mini-breaks after each part

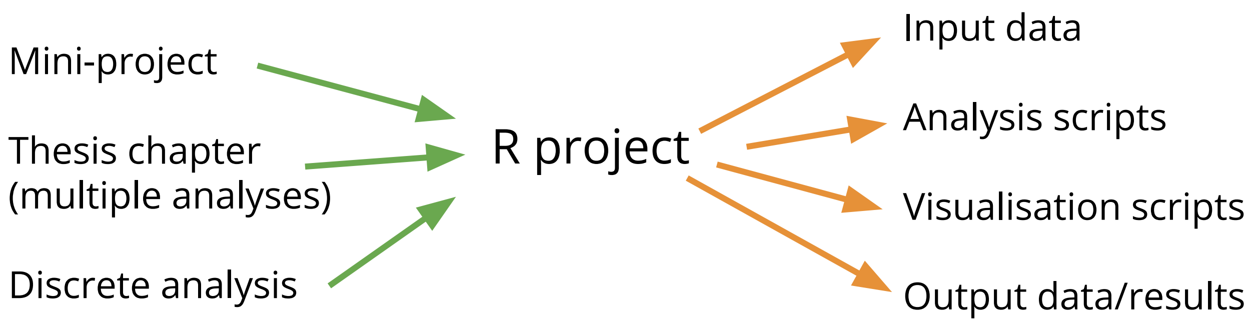
The slides will be shared after!

1. Project-oriented workflow

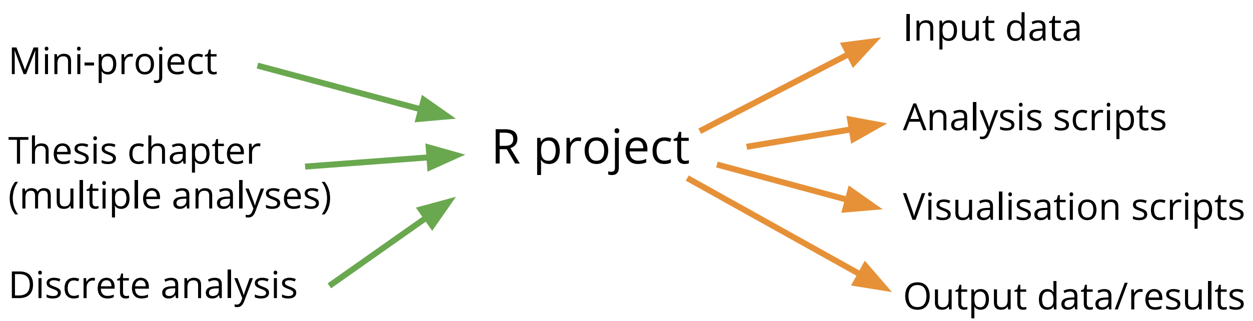
R projects


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



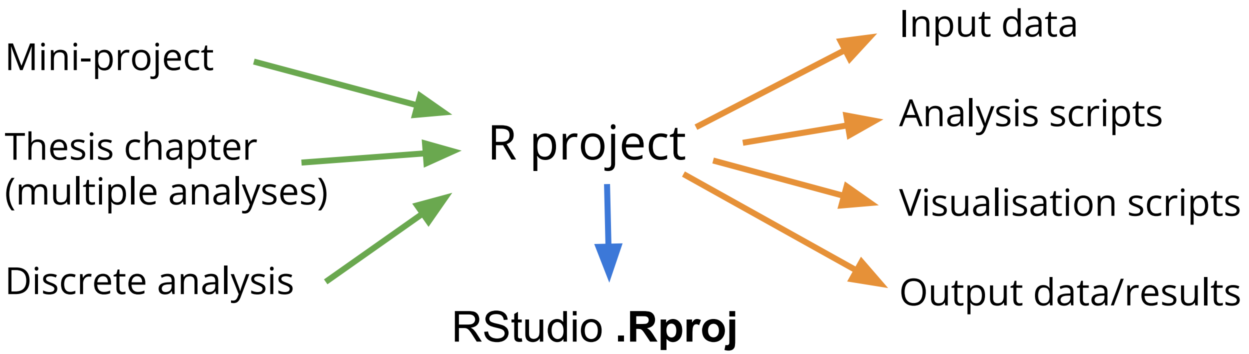
How you do organise your projects? scripts? folders?

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



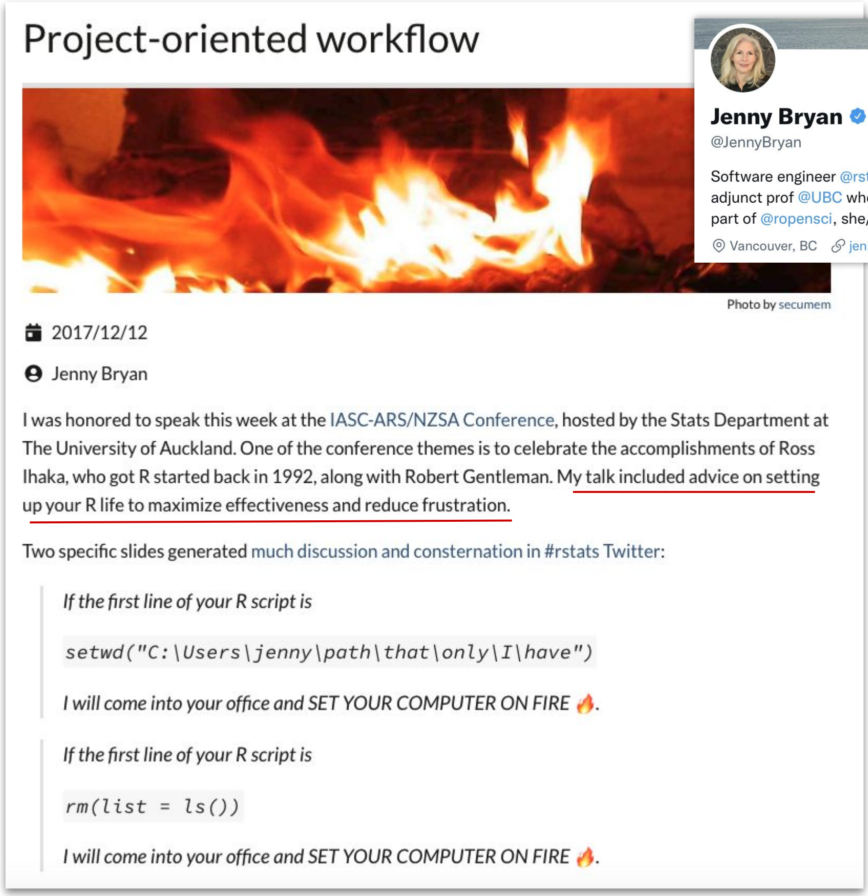
How you do organise your projects? scripts? folders?

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

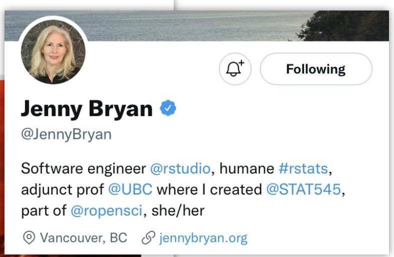
Jenny Bryan

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* 🔥.



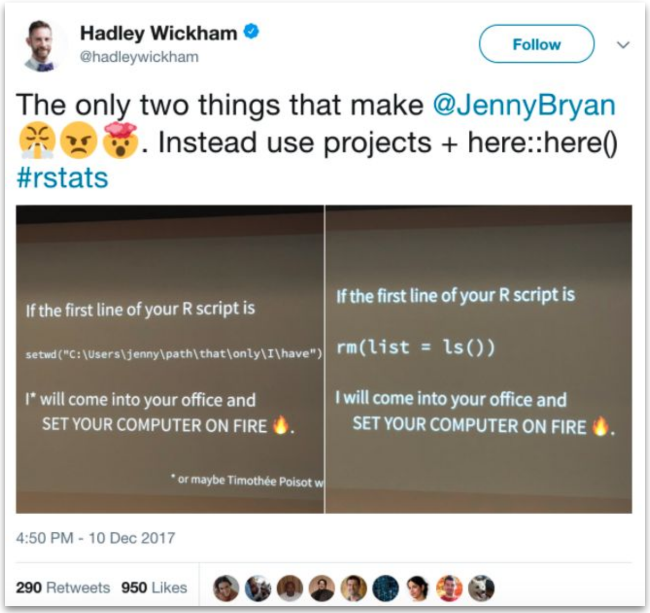
Following

Jenny Bryan

@JennyBryan

Software engineer @rstudio, humane #rstats, adjunct prof @UBC where I created @STAT545, part of @ropensci, she/her

Vancouver, BC jennybryan.org



Hadley Wickham

@hadleywickham

Follow

The only two things that make @JennyBryan 😤😠🤯. Instead use projects + here::here() #rstats

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 🔥.

* or maybe Timothée Poisot w

If the first line of your R script is

`rm(list = ls())`

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

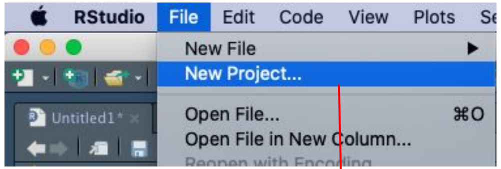
4:50 PM - 10 Dec 2017

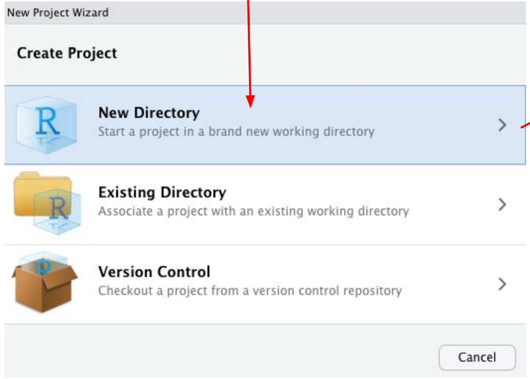
290 Retweets 950 Likes

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

Project-oriented workflow

Use Rstudio / .Rproj for your data analysis projects





New Project Wizard

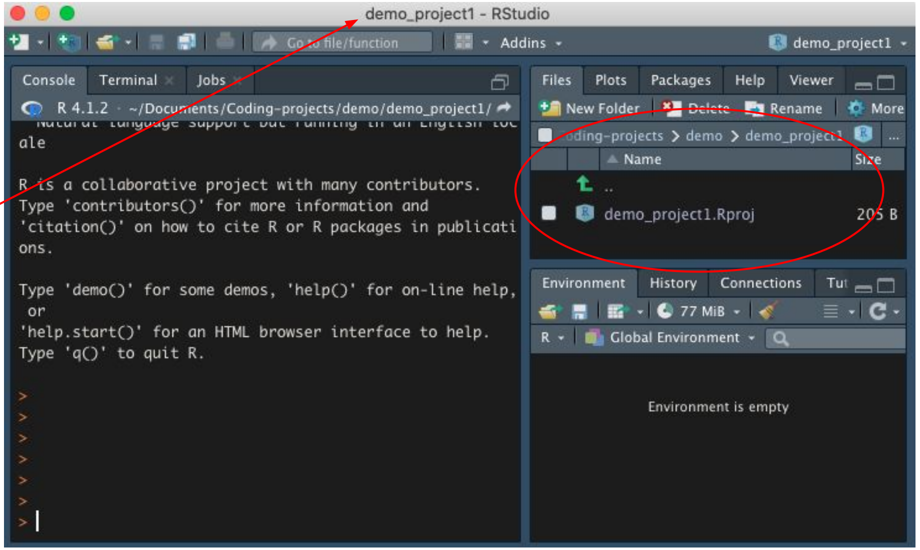
Create Project

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



```
R 4.1.2 · ~/Documents/Coding-projects/demo/demo_project1/
Natural language support but running in an English loc
ale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publicati
ons.

Type 'demo()' for some demos, 'help()' for on-line help,
 or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
>
>
>
>
>
>
```

```
oding-projects > demo > demo_project1
Name                      Size
..
demo_project1.Rproj       205 B
```

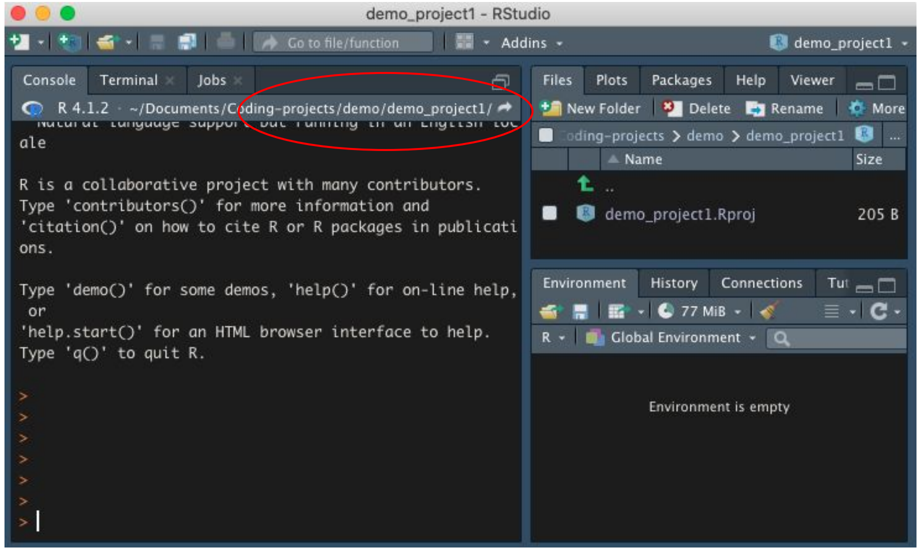
This means that you are essentially compartmentalizing your current project

Project-oriented workflow

Use Rstudio / .Rproj for your data analysis projects

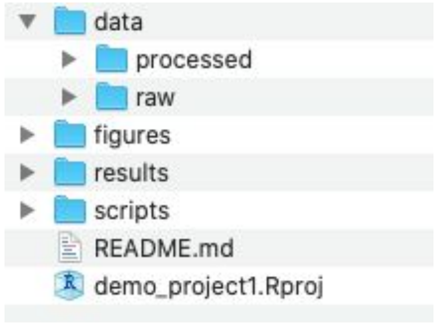
- Project directory stores all your data, scripts
- The working directory is set to the project directory (e.g. demo_project1), so don't need to specify full paths to data (only internal subfolders)

- 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)

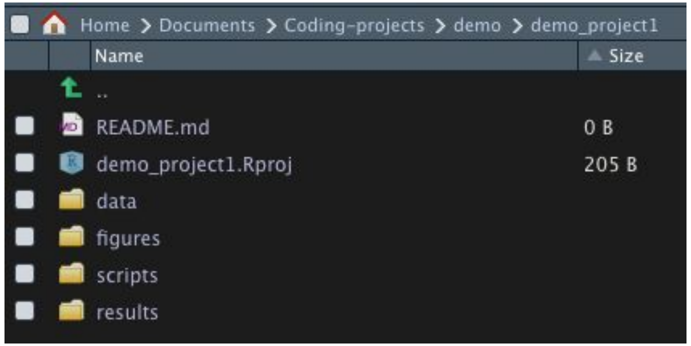


```
R 4.1.2 · ~/Documents/Coding-projects/demo/demo_project1/
Natural language support but running in an English loc
ale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publicati
ons.

Type 'demo()' for some demos, 'help()' for on-line help,
 or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
>
>
>
>
>
> |
```

Organise your projects intentionally



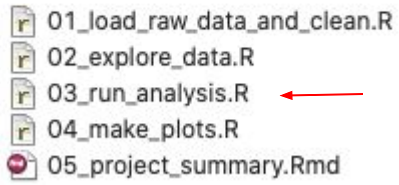
```
▼ data
    ▶ processed
    ▶ raw
▶ figures
▶ results
▶ scripts
  README.md
  demo_project1.Rproj
```



Home > Documents > Coding-projects > demo > demo_project1

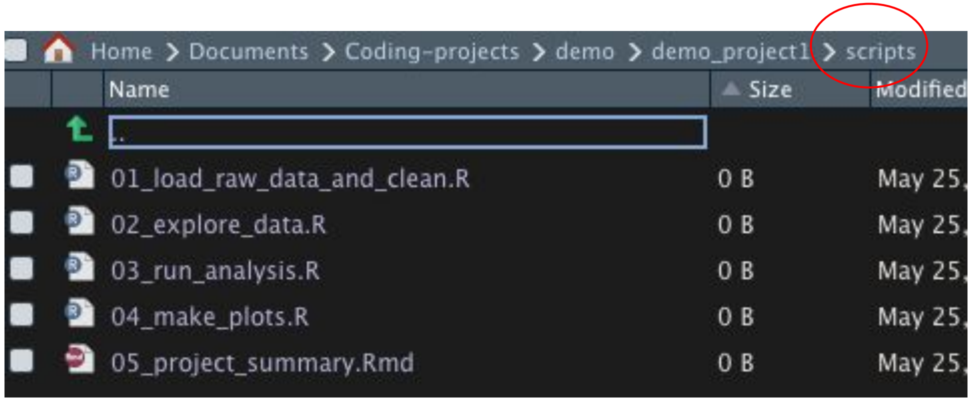
Name	Size
..	
README.md	0 B
demo_project1.Rproj	205 B
data	
figures	
scripts	
results	

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



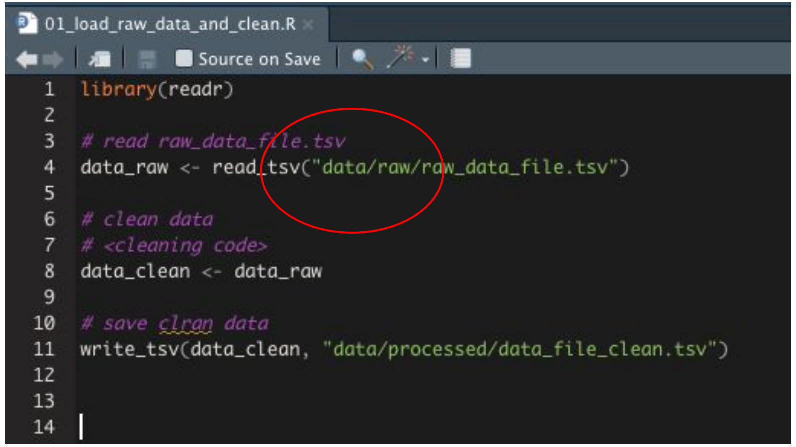
Home > Documents > Coding-projects > demo > demo_project1 > scripts

```
Name                              Size    Modified
..
01_load_raw_data_and_clean.R      0 B     May 25,
02_explore_data.R                 0 B     May 25,
03_run_analysis.R                 0 B     May 25,
04_make_plots.R                   0 B     May 25,
05_project_summary.Rmd            0 B     May 25,
```

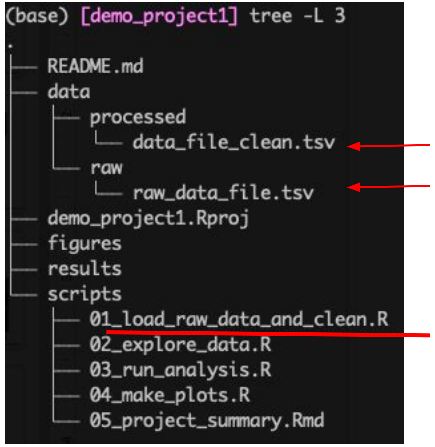
Don't use `setwd()`

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

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



```r
library(readr)

# read raw_data_file.tsv
data_raw <- read_tsv("data/raw/raw_data_file.tsv")

# clean data
# <cleaning code>
data_clean <- data_raw

# save clran data
write_tsv(data_clean, "data/processed/data_file_clean.tsv")


```



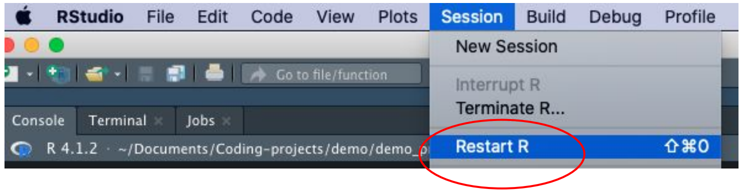
```
(base) [demo_project1] tree -L 3
.
├── README.md
├── data
│   ├── processed
│   │   └── data_file_clean.tsv
│   └── raw
│       └── raw_data_file.tsv
├── demo_project1.Rproj
├── figures
├── results
└── scripts
    ├── 01_load_raw_data_and_clean.R
    ├── 02_explore_data.R
    ├── 03_run_analysis.R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

No need to hardcode paths when using Rproj

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

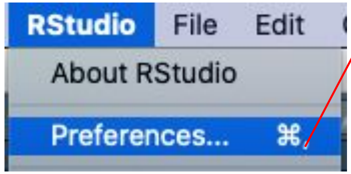
rm(list=ls()) does not fully clean your env!

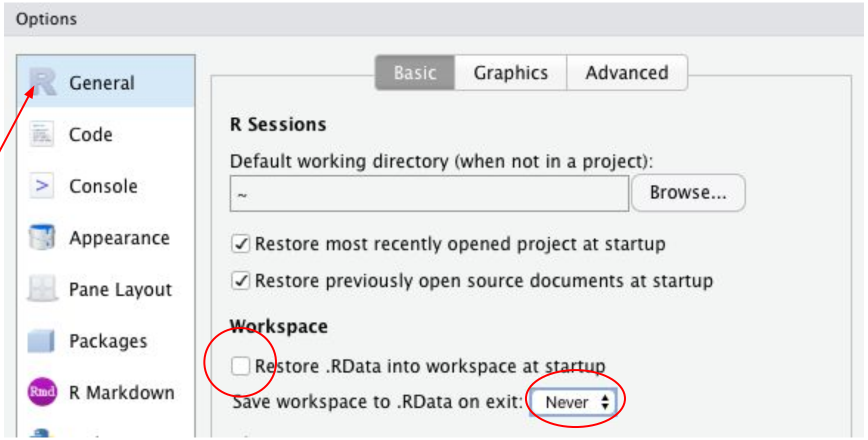
Restart R daily (or every time you start working after a break) to ensure a clean environment



And !! do not save your .Rdata workspace

(untick and select 'never')





R Sessions

Default working directory (when not in a project): ~

[✓] Restore most recently opened project at startup

[✓] Restore previously open source documents at startup

Workspace

[ ] Restore .RData into workspace at startup

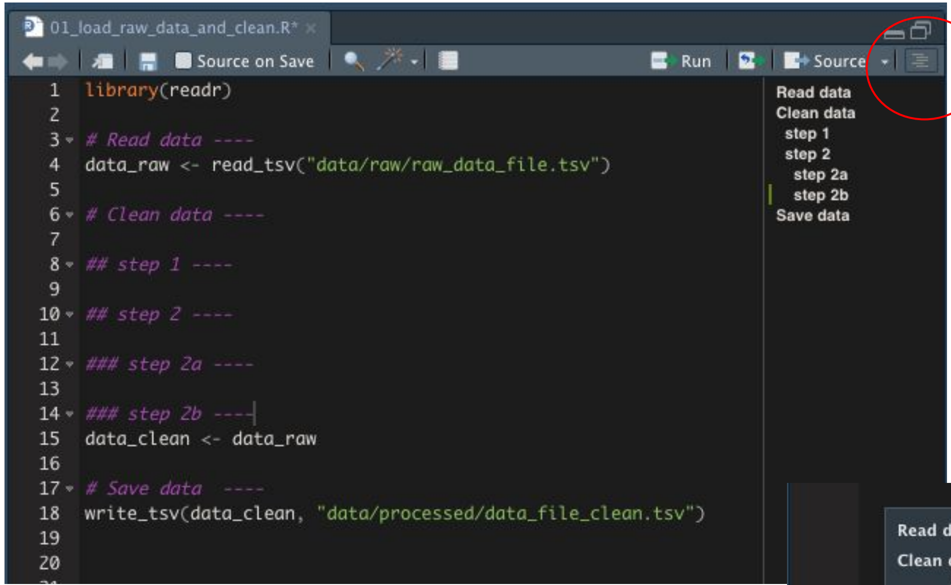
Save workspace to .RData on exit: Never

Save your 'real' work, delete the rest

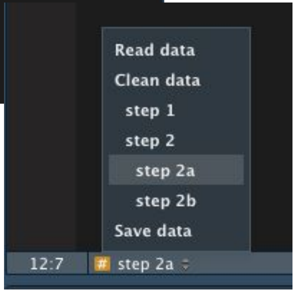
5 mins to try it

2. Various efficiency tips for R

Name your code sections and use then for quick navigation



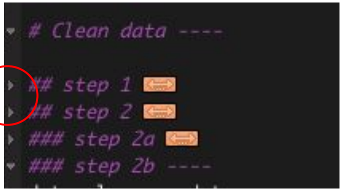
```r
library(readr)

# Read data ----
data_raw <- read_tsv("data/raw/raw_data_file.tsv")

# Clean data ----

## step 1 ----

## step 2 ----

### step 2a ----

### step 2b ----
data_clean <- data_raw

# Save data ----
write_tsv(data_clean, "data/processed/data_file_clean.tsv")
```



- Use section headings:

```
# section ----

## subsection ----

### subsubsection ----
```

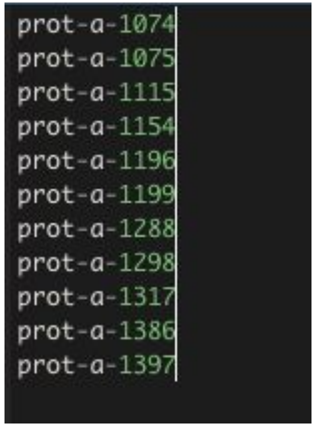
- Great for navigating in long scripts
- Can fold sections



```r
# Clean data ----

## step 1
## step 2
### step 2a
### step 2b ----
```

Vertical selection

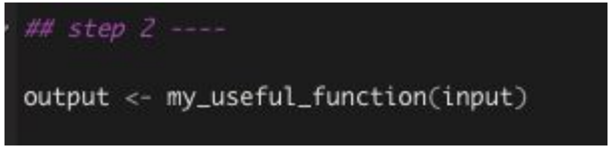
(hold *option* or *alt* and drag cursor down to select vertically)


```
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
```

Great for e.g.

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

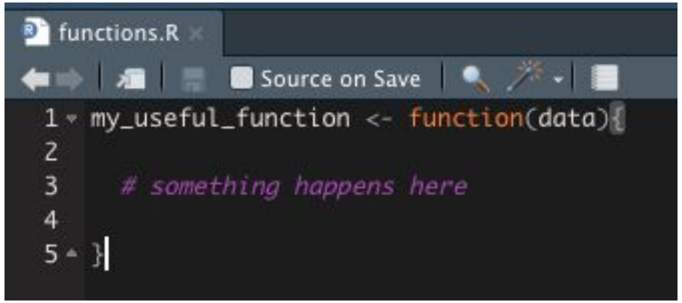
Jump to function definition or open data frame



```
## step 2 ----

output <- my_useful_function(input)
```

Cmd + mouse click on the name

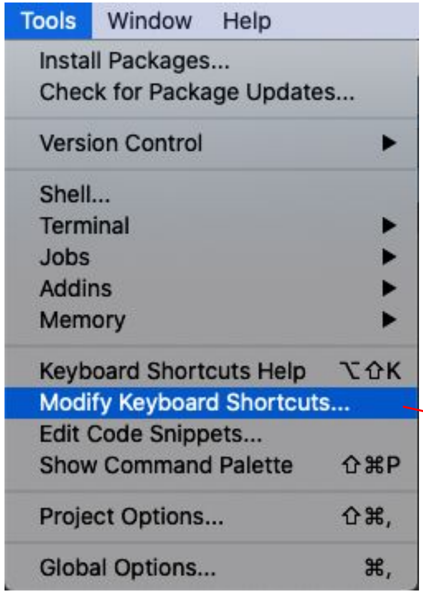
(opens in a new window)

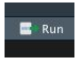


```
my_useful_function <- function(data){

  # something happens here

}
```

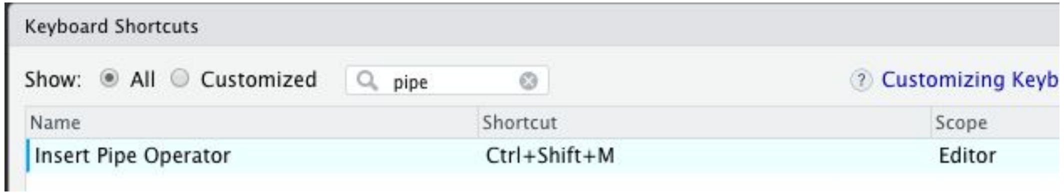
> also works on functions from external packages (if you want to check what they do internally)

Keyboard shortcuts


Tools Window Help
Install Packages...
Check for Package Updates...
Version Control
Shell...
Terminal
Jobs
Addins
Memory
Keyboard Shortcuts Help ⌥⇧K
Modify Keyboard Shortcuts...
Edit Code Snippets...
Show Command Palette ⇧⌘P
Project Options... ⇧⌘,
Global Options... ⌘,

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

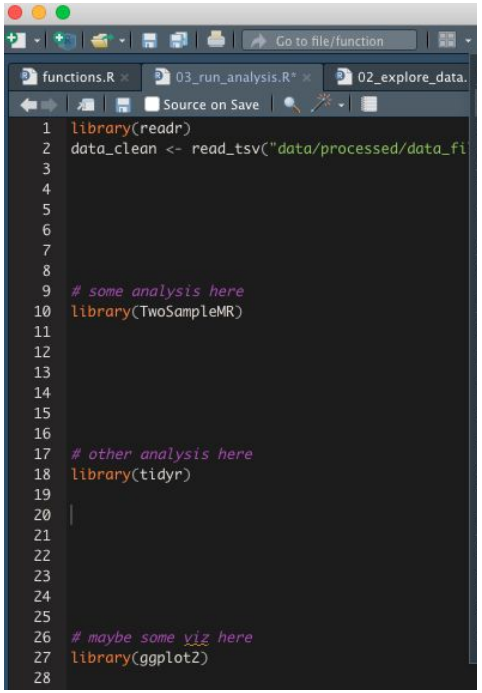
  ```


Keyboard Shortcuts

Show: [✓ All] Customized pipe Customizing Keyb

Name	Shortcut	Scope
Insert Pipe Operator	Ctrl+Shift+M	Editor
  ```

Move all libraries to the top

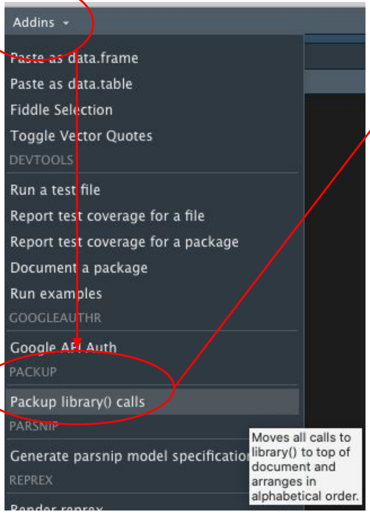


```r
library(readr)
data_clean <- read_tsv("data/processed/data_fi






# some analysis here
library(TwoSampleMR)






# other analysis here
library(tidyr)








# maybe some viz here
library(ggplot2)

```

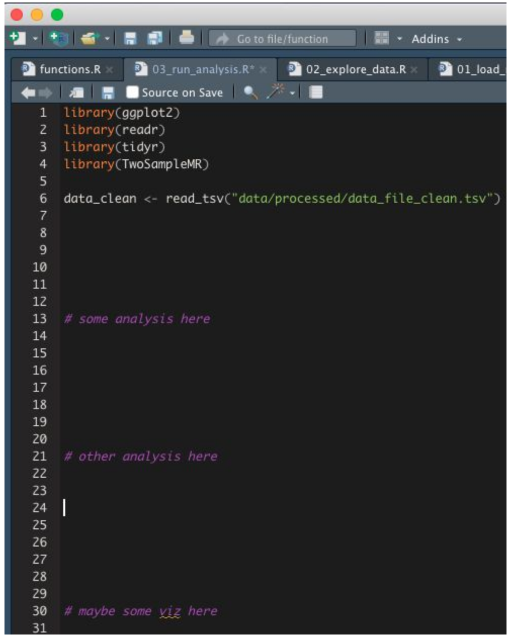
Install *packup* add-in:

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

Call it from Addins menu:



Moves all calls to library() to top of document and arranges in alphabetical order.



```r
library(ggplot2)
library(readr)
library(tidyr)
library(TwoSampleMR)

data_clean <- read_tsv("data/processed/data_file_clean.tsv")






# some analysis here







# other analysis here








# maybe some viz here

```

https://github.com/MilesMcBain/packup

Any other R tricks to share?

3. Rproj for git users

https://happygitwithr.com/

Git brief intro

Git - tool for code version control

- Tracking code changes
- Keeping older versions of the script
- Coding collaboration (with others or yourself in multiple locations)
- Tracking who and when made changes

If used consistently!

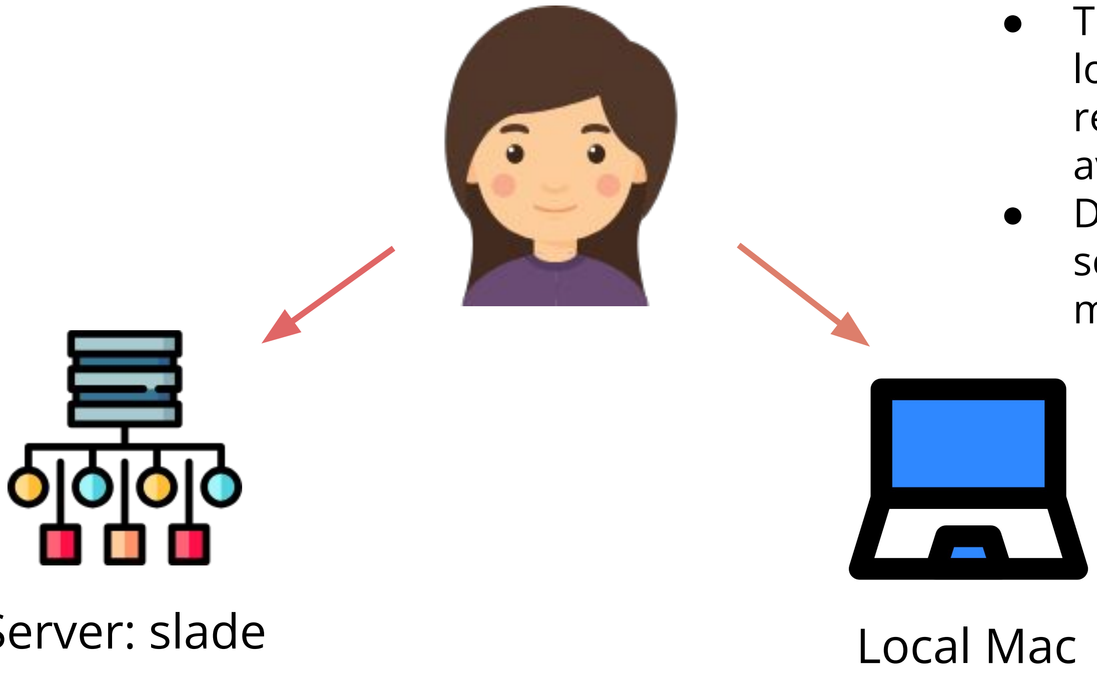
Why use it?

- Can facilitate data/code integrity
- Improves reproducibility (e.g. keeps record of changes)
- Enables code sharing with colleagues / as a part of publication
- Important skill for anyone working with data (e.g. can showcase your work during job applications)

Github / Gitlab etc

- Platforms where you can store your coding projects using git

Why might you want to use git?



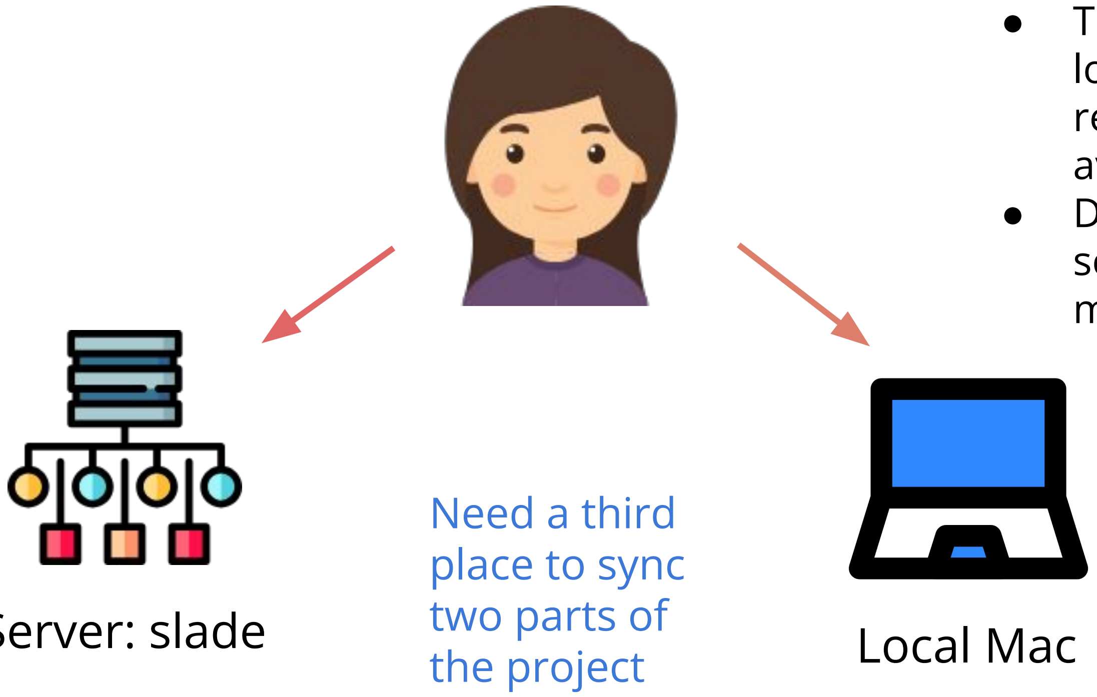
Problem:

- The same project is both locations (e.g due to data restrictions / tool availability)
- Don’t want to copy over scripts all the time, as it might get messy!

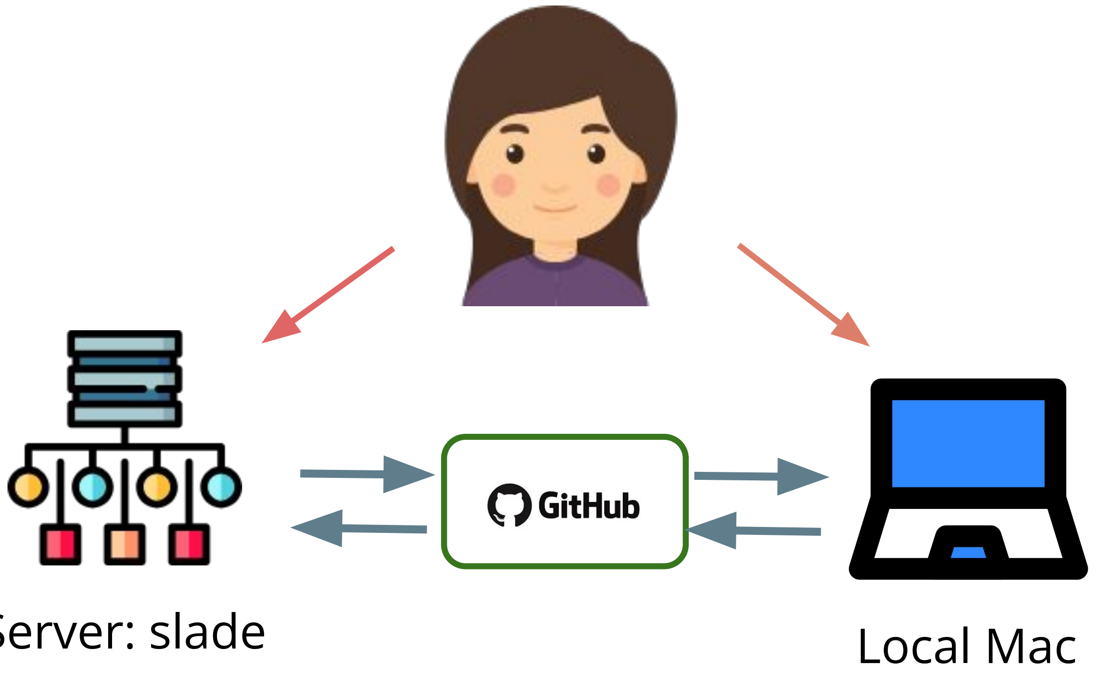
Why might you want to use git?

Problem:

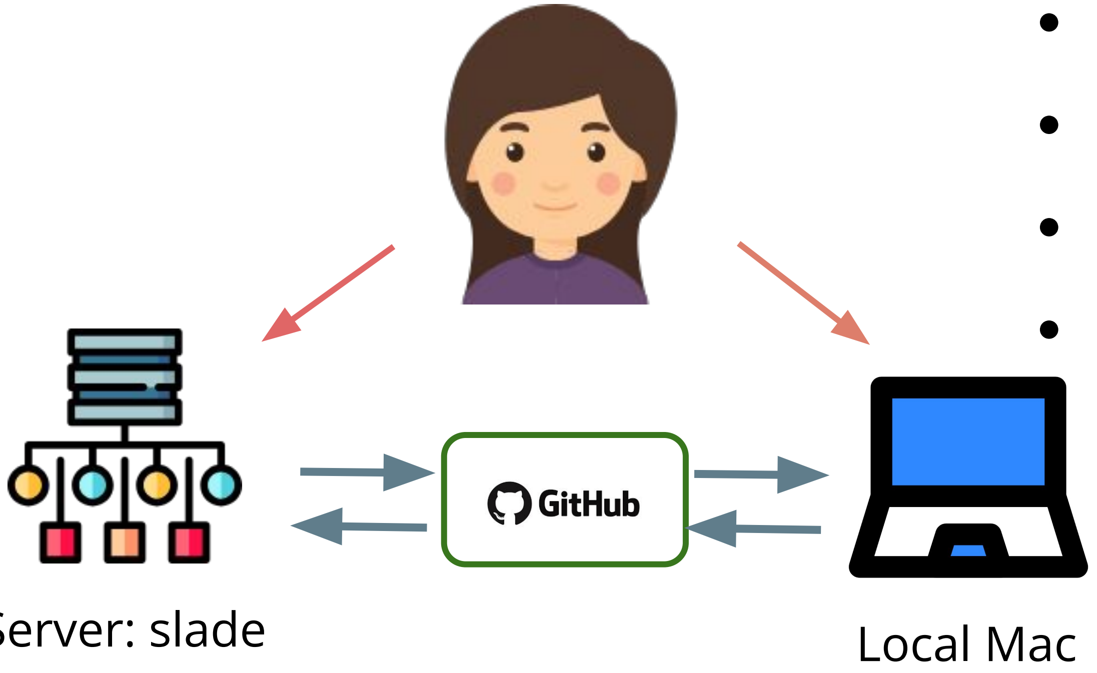
- The same project is both locations (e.g due to data restrictions / tool availability)
- Don't want to copy over scripts all the time, as it might get messy!



Why might you want to use git?



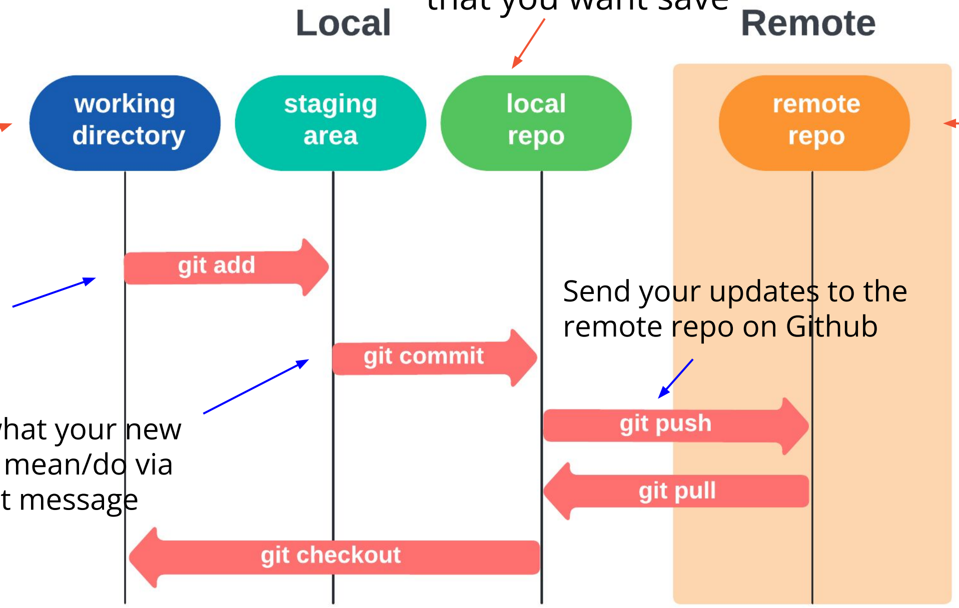
Why might you want to use git?



Solution:

- Use git to “collaborate with yourself”
- This enhances project reproducibility
- Keeps everything in one project folder
- Compatible with .Rproj

Basic git commands / actions



git local repo will keep a record of the new changes that you want save

Analysis folder on your local computer

Your analysis folder repo on Github remote repo with all changes tracked, with a commit message that explains what has changes

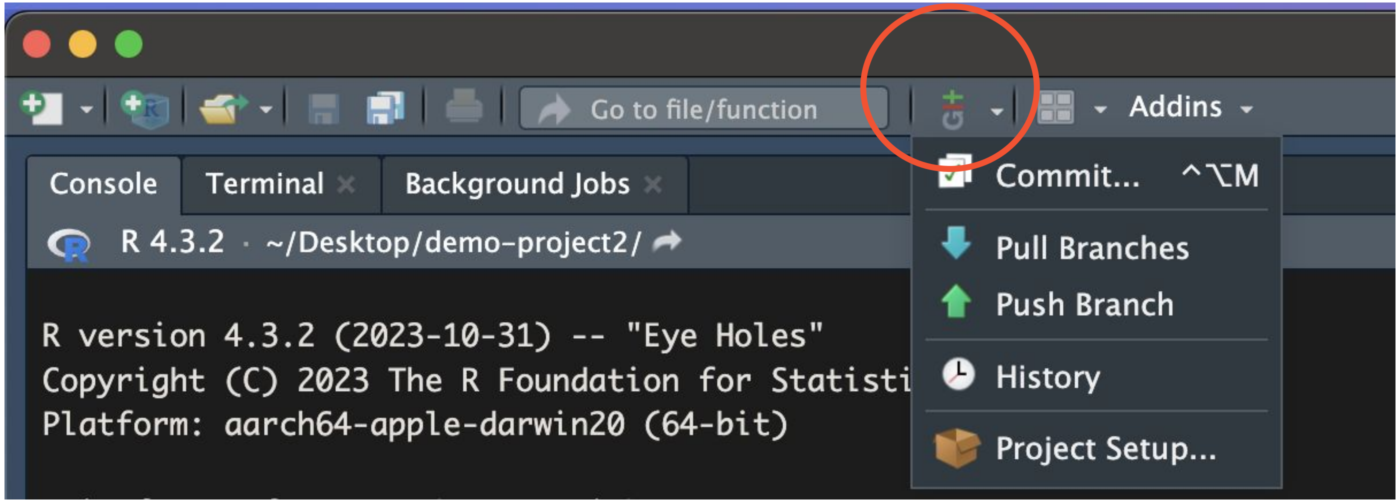
Tell git which files you changed/created and want to track with git

Tell git what your new changes mean/do via a commit message

Send your updates to the remote repo on Github

Using git with your .Rproj

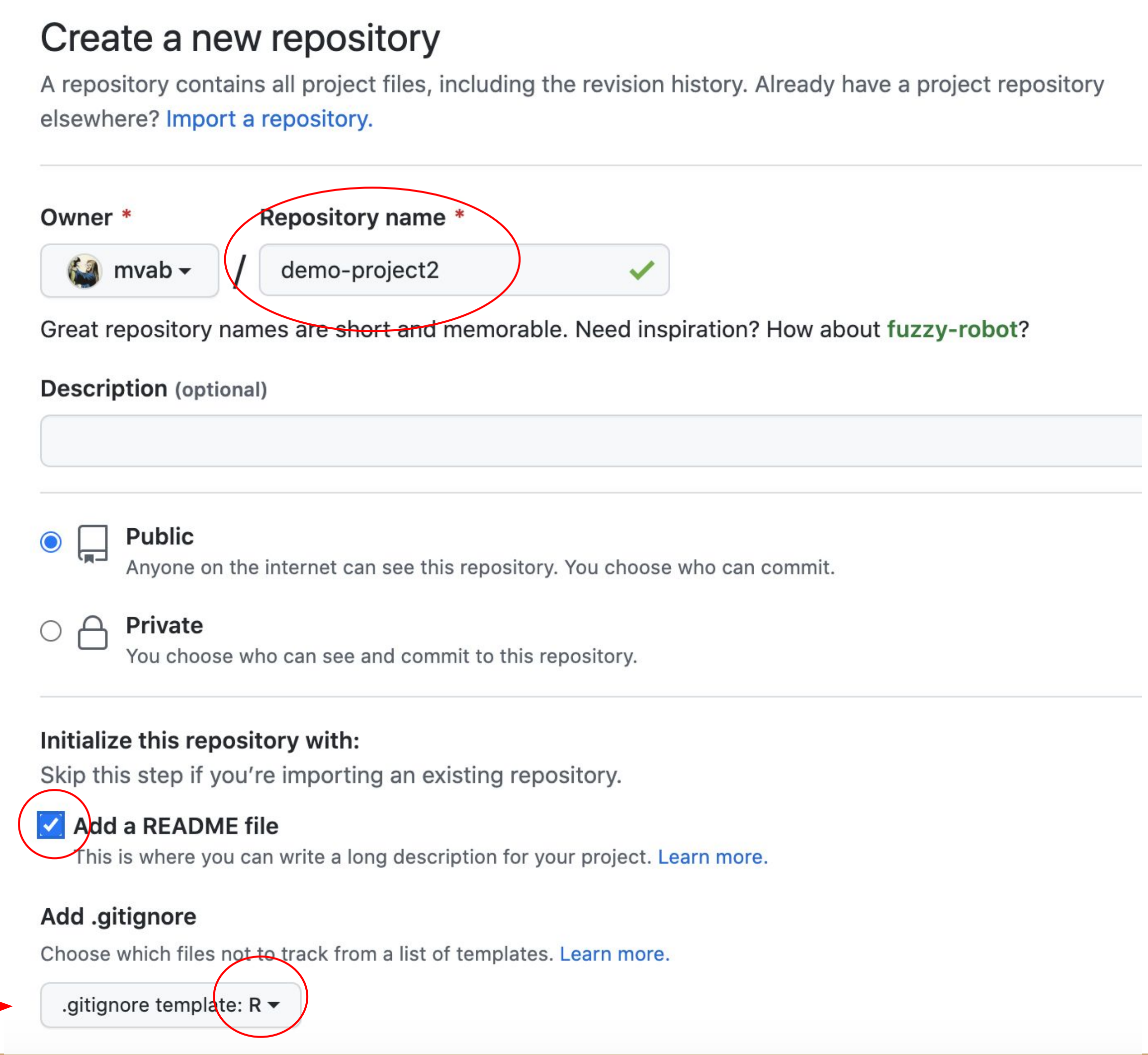
- Rstudio user interface makes it easy to get started with git
- (but also can use the terminal to run git commands)



```
R version 4.3.2 (2023-10-31) -- "Eye Holes"
Copyright (C) 2023 The R Foundation for Statisti
Platform: aarch64-apple-darwin20 (64-bit)
```

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.

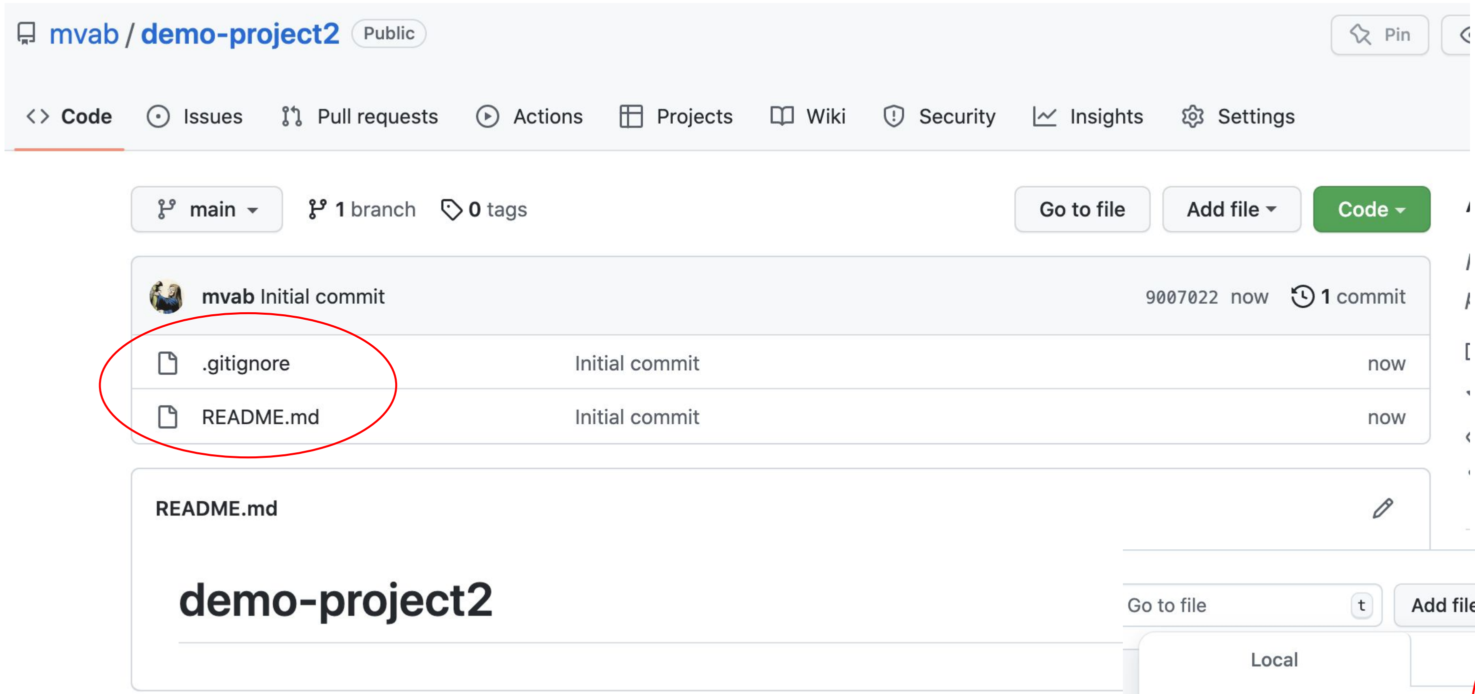
○ Private
You choose who can see and commit to this repository.

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


mvab / demo-project2 Public

Pin

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

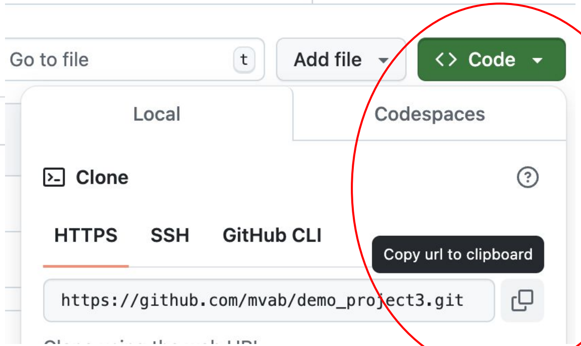
main 1 branch 0 tags Go to file Add file Code

mvab Initial commit 9007022 now 1 commit

.gitignore	Initial commit	now
README.md	Initial commit	now

README.md

demo-project2


Go to file t Add file Code

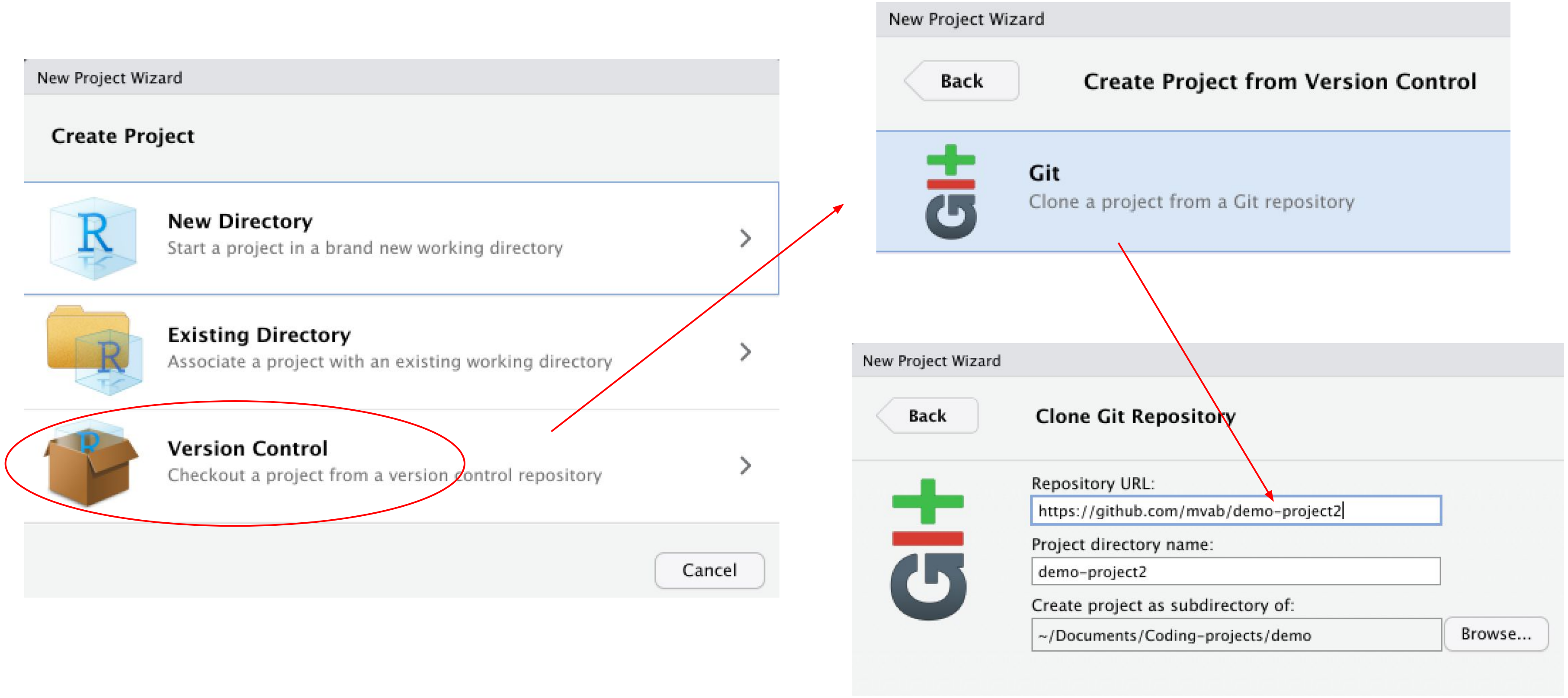
Local Codespaces

Clone

HTTPS SSH GitHub CLI

Copy url to clipboard

```
https://github.com/mvab/demo_project3.git
```



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

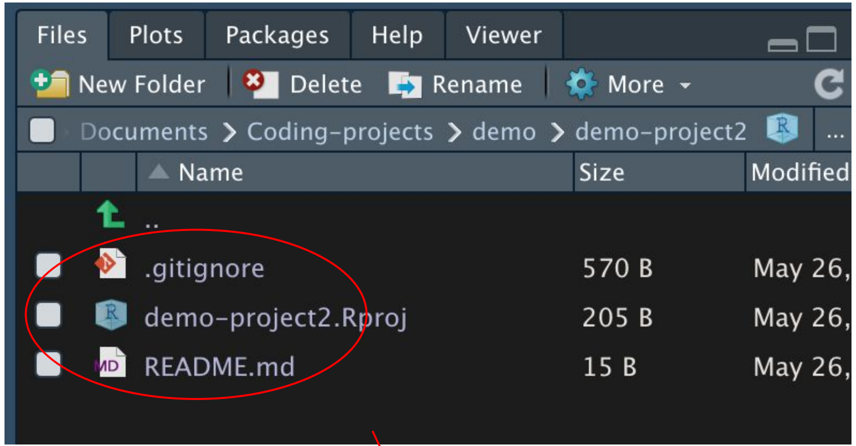
Repository URL:
https://github.com/mvab/demo-project2

Project directory name:
demo-project2

Create project as subdirectory of:
~/Documents/Coding-projects/demo

Browse...

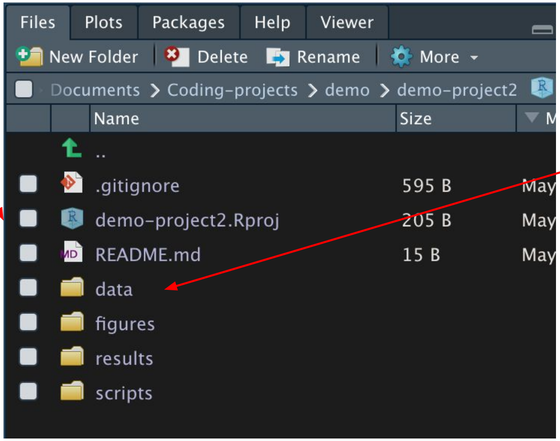
1



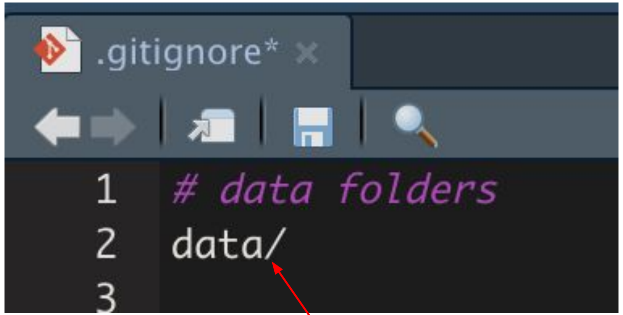
Name	Size	Modified
..		
.gitignore	570 B	May 26,
demo-project2.Rproj	205 B	May 26,
README.md	15 B	May 26,

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

2 Add folders etc



Name	Size	M
..		
.gitignore	595 B	May
demo-project2.Rproj	205 B	May
README.md	15 B	May
data		
figures		
results		
scripts		

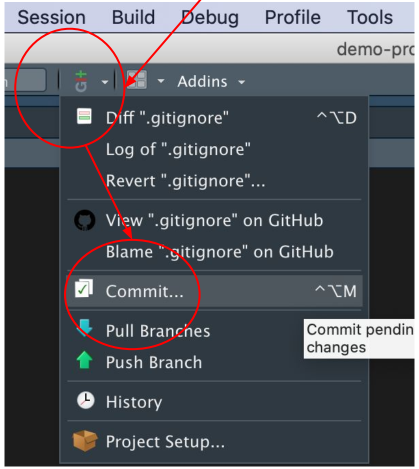


```
# data folders
data/

```

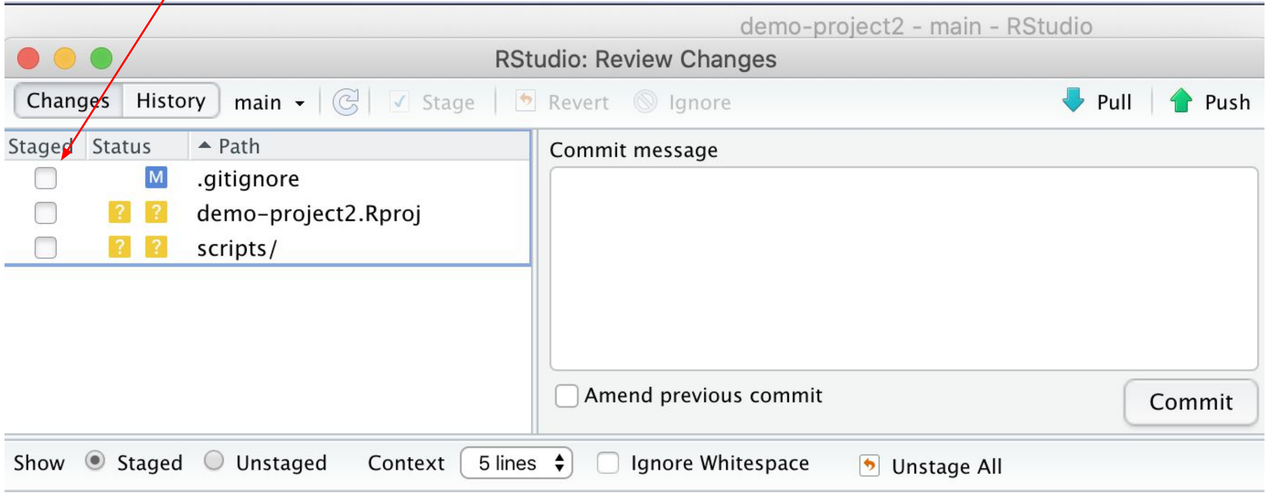
3

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

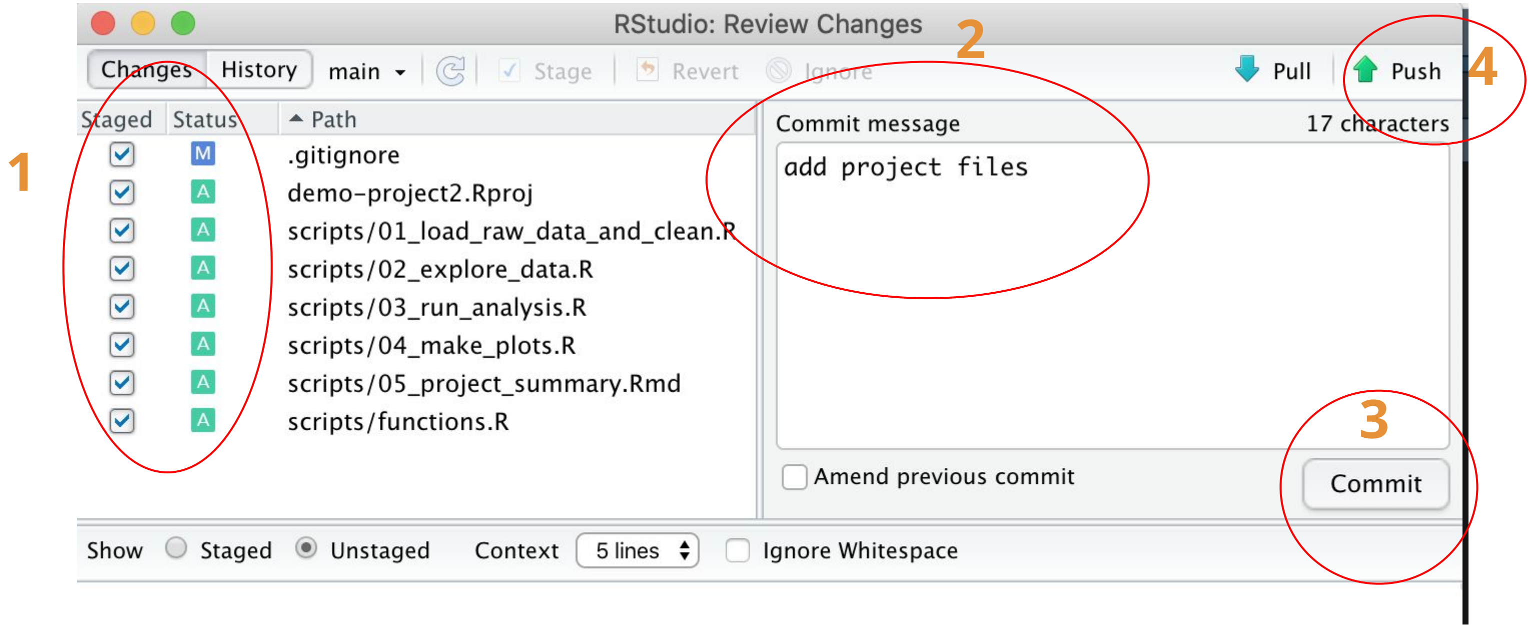


Tick files to commit:



Staged	Status	Path
☐	M	.gitignore
☐	? ?	demo-project2.Rproj
☐	? ?	scripts/

Commit changes: Add message:


RStudio: Review Changes

Changes | History | main | Stage | Revert | Ignore | Pull | Push

Staged	Status	Path
✓	M	.gitignore
✓	A	demo-project2.Rproj
✓	A	scripts/01_load_raw_data_and_clean.R
✓	A	scripts/02_explore_data.R
✓	A	scripts/03_run_analysis.R
✓	A	scripts/04_make_plots.R
✓	A	scripts/05_project_summary.Rmd
✓	A	scripts/functions.R

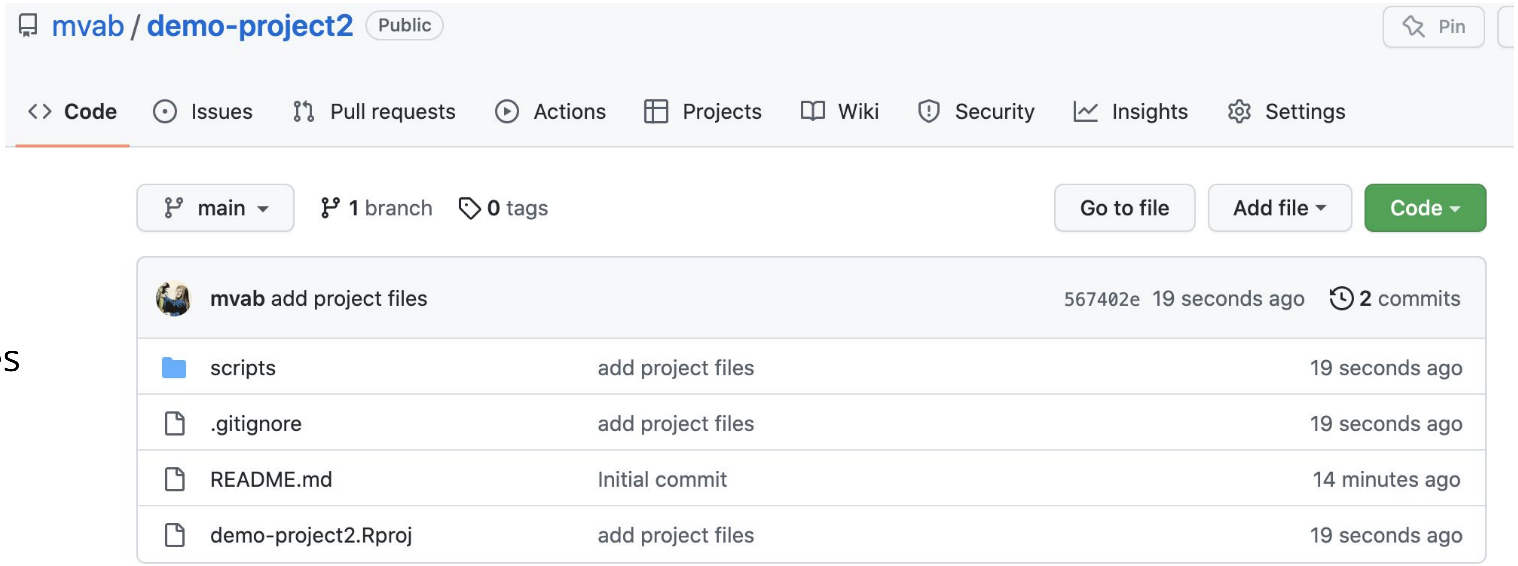
Commit message — 17 characters

```
add project files
```

☐ Amend previous commit

Show ○ Staged [● Unstaged] Context 5 lines ☐ Ignore Whitespace

Your changes
on Github:

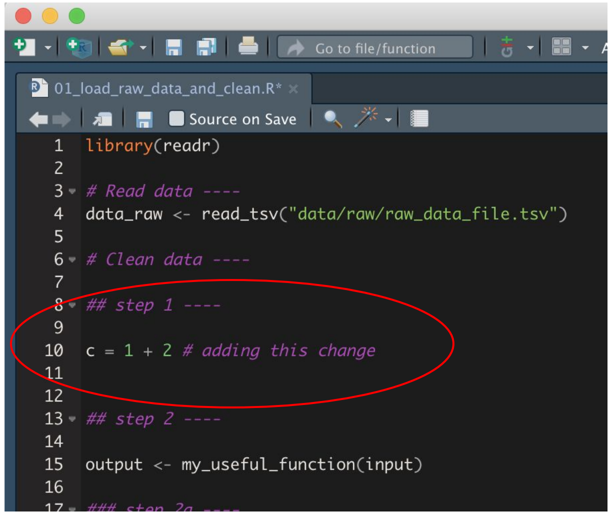


mvab add project files 567402e 19 seconds ago 2 commits

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:

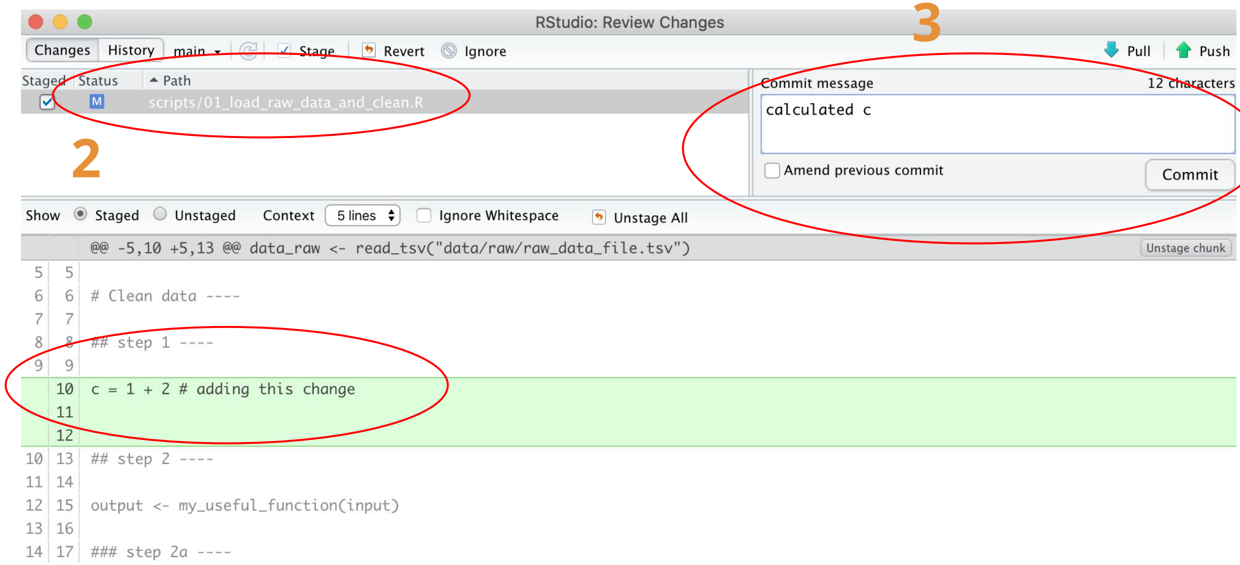
1



```
library(readr)

# Read data ----
data_raw <- read_tsv("data/raw/raw_data_file.tsv")

# Clean data ----

## step 1 ----

c = 1 + 2 # adding this change


## step 2 ----

output <- my_useful_function(input)

### step 2a ----
```

2

3



RStudio: Review Changes

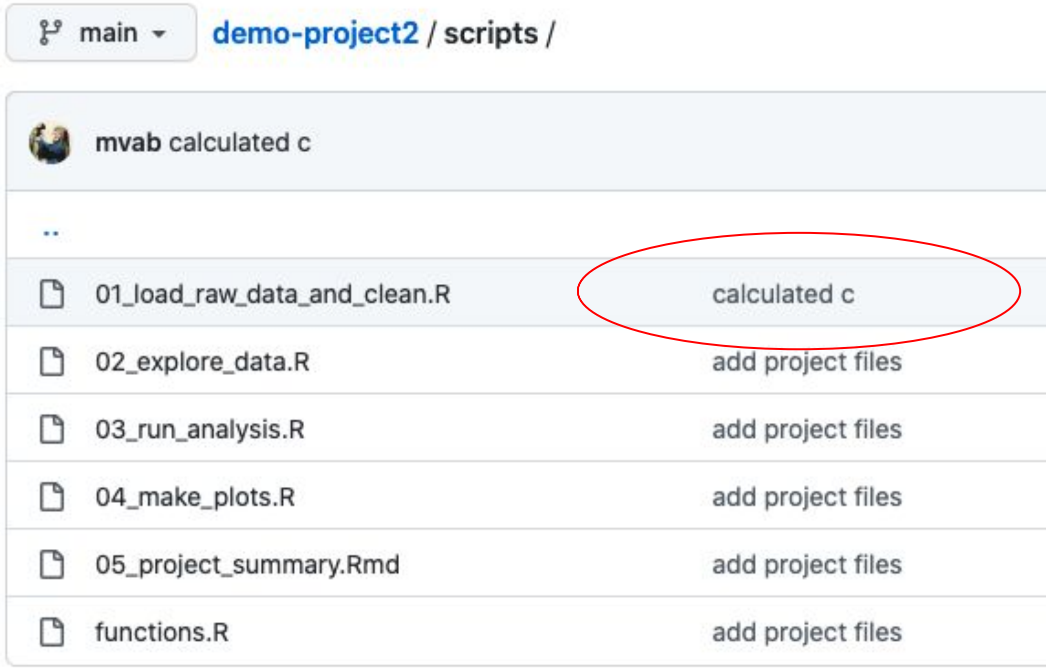
M scripts/01_load_raw_data_and_clean.R

Commit message — 12 characters

calculated c

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   c = 1 + 2 # adding this change
    11
    12
10  13   ## step 2 ----
11  14
12  15   output <- my_useful_function(input)
13  16
14  17   ### step 2a ----
```



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-running your script under development from the top will help you catch issues early on
  - **Using git for version control**

- 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 in R 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 - 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

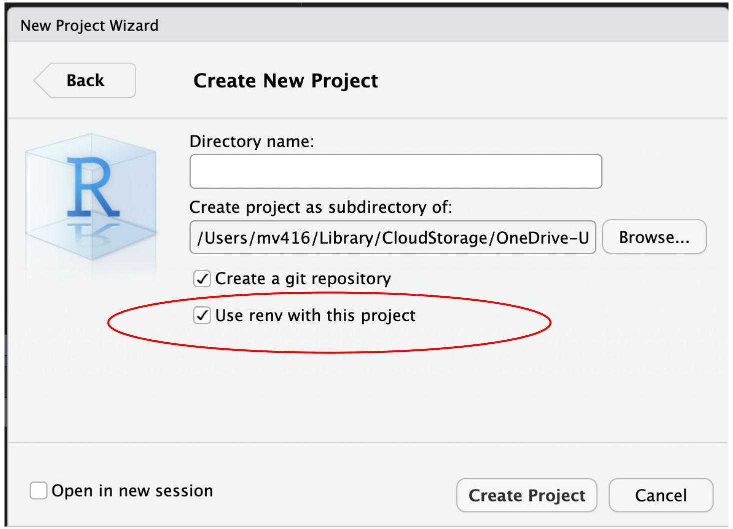
https://appsilon.com/rstudio-shortcuts-and-tips/

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

https://happygitwithr.com/

https://rstudio.github.io/renv/articles/renv.html

.renv for managing package versions within projects



New Project Wizard

Back

Create New Project

Directory name:

Create project as subdirectory of:

/Users/mv416/Library/CloudStorage/OneDrive-U

Browse...

[✓] Create a git repository

[✓] Use renv with this project

[ ] Open in new session

Create Project

Cancel

renv package helps you create reproducible environments for your R projects

- Keeps track of all your installed packages
- Installs them in a new location (i.e. easy to share)
- Useful for working on DNAnexus/AoU platforms

See tutorial - https://rstudio.github.io/renv/articles/renv.html