

What They Forgot to Teach You About R



rstudio::conf
SAN FRANCISCO // JANUARY 27 - 30, 2020

from RStudio

This work is licensed under a **Creative Commons**
Attribution-ShareAlike 4.0 International License.

To view a copy of this license, visit
<http://creativecommons.org/licenses/by-sa/4.0/>

rstd.io/wtf-2020-rsc

Day 1, morning

Project-oriented workflow

Kara Woo



@karawoo



@kara_woo

Jenny Bryan



@jennybc



@JennyBryan

What *Did* They Forget
to Teach You?

Everything else

Statistical
analysis

*Deep
Thoughts*



Be organized

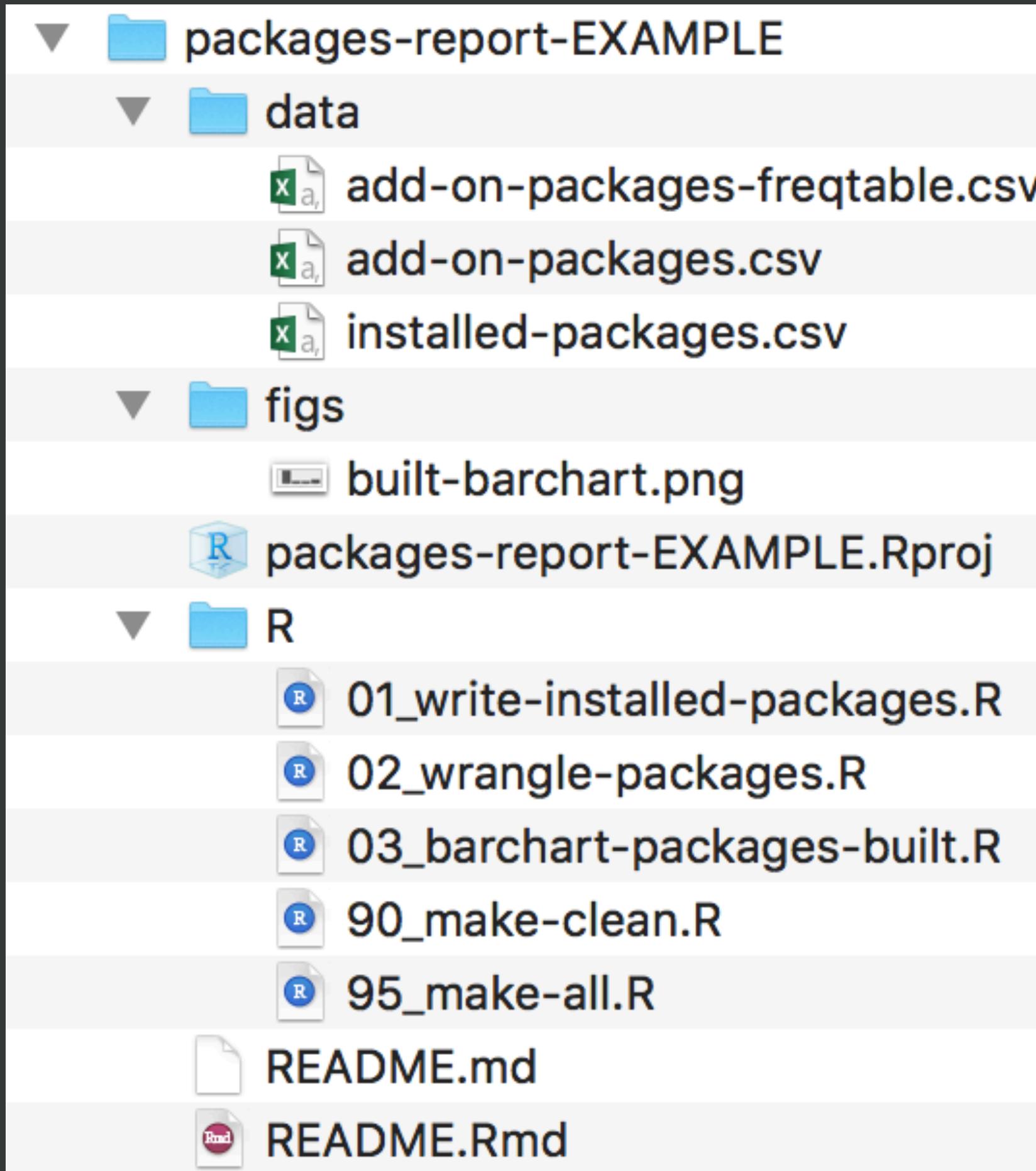
do this as you go, not "tomorrow"

but also don't fret over past mistakes
raise the bar for *new* work



Be organized

self-explaining >>> wordy, needy explainers



>>>

file salad
+ an out-of-date README

Good enough practices in scientific computing

Wilson, Bryan, Cranston, Kitzes, Nederbragt, Teal

<https://doi.org/10.1371/journal.pcbi.1005510>

<http://bit.ly/good-enuff>



Day 1 Practical Example: Explore your R installation

R package = the natural unit for distributing R code

base R \approx 14 base + 15 recommended packages
ship with all binary distributions of R

can use right out of the box:

`library(lattice)`

CRAN has ~15K additional packages
install, then attach:

```
install.packages("devtools")
library(devtools)
```

And then there's GitHub ...

install via devtools, then attach:

```
devtools::install_github("jimhester/lookup")
library(lookup)
```

Where do packages live locally?

By default, in the default library

. Library

All libraries for current session:

.libPaths()

All installed packages:

installed.packages()

```
install.packages("usethis")
library(usethis)
use_course("rstudio/wtf-explore-libraries")
```

```
install.packages("usethis")
library(usethis)
use_course("rstd.io/wtf-explore-libraries")
```

Pick one to open and flesh out:

01_explore-libraries_spartan.R
01_explore-libraries_comfy.R*

* worst case, there's always jenny

```
install.packages("usethis")
library(usethis)
use_course("rstd.io/wtf-explore-libraries")
```

Stay relaxed.

We will refine this code and where it lives soon enough.

You want to leave rough edges and gaps to address later.

work on challenge

*Deep
Thoughts*



Adopt a project-oriented workflow

Why?

- work on more than 1 thing at a time
- collaborate, communicate, distribute
- start and stop

Adopt a project-oriented workflow

How?

- dedicated directory
- RStudio Project
- Git repo, probably syncing to a remote

If you do this at the top of your scripts, I might set
your computer on 🔥:

```
setwd("C:\Users\jenny\path\that\only\I\have")  
rm(list = ls())
```

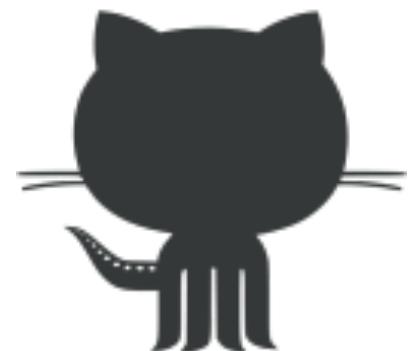
Project-oriented workflow designs this away:

<https://www.tidyverse.org/articles/2017/12/workflow-vs-script/>

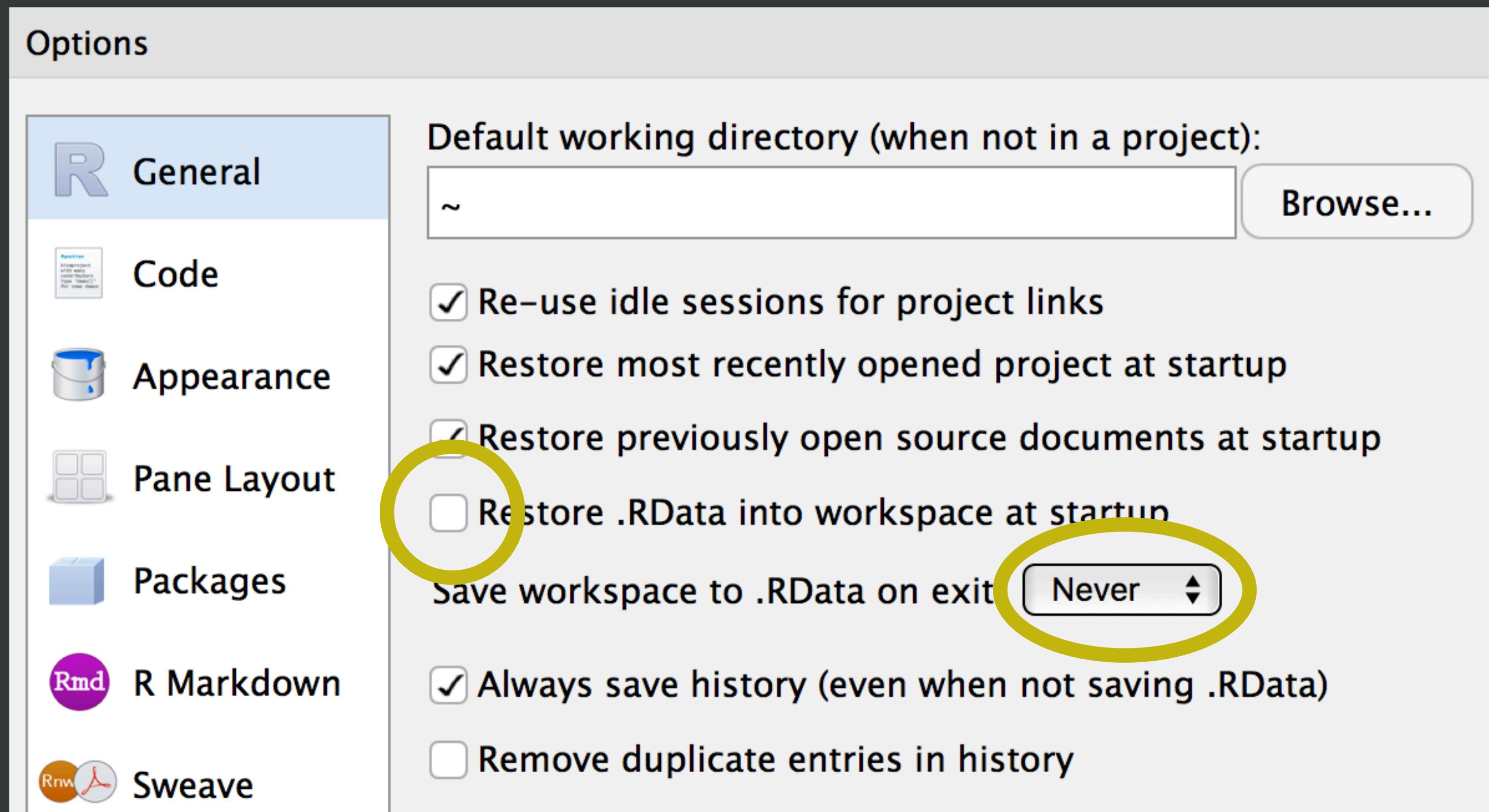
<https://whattheyforgot.org> ← see "A holistic workflow"

What does it mean to be an RStudio Project?

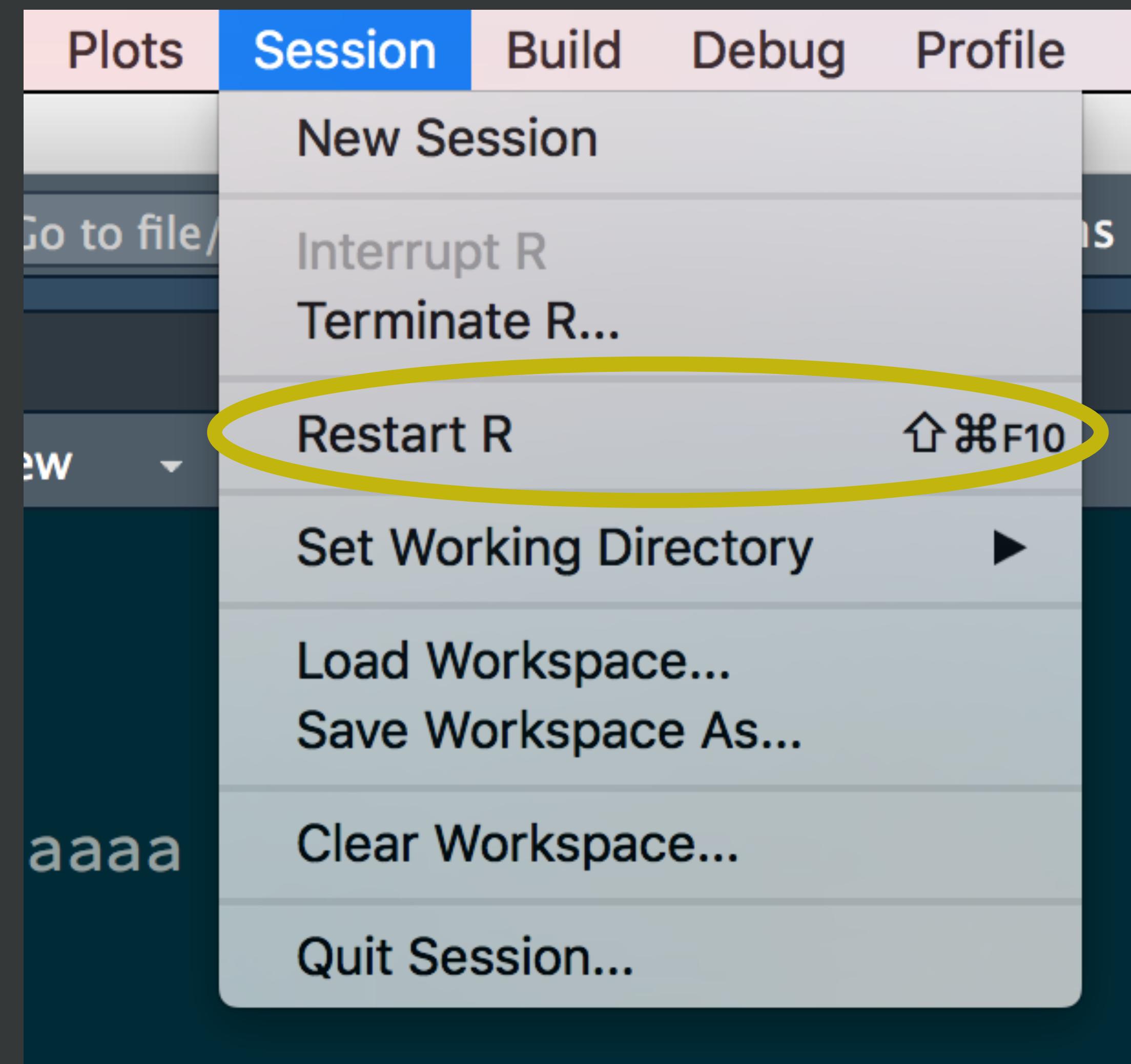
- RStudio leaves notes to itself in `foo.Rproj`
- Open Project = dedicated instance of RStudio
 - dedicated R process
 - file browser pointed at Project directory
 - working directory set to Project directory



Use a "blank slate"



Restart R often



Project initiation: the local case

Pick one and
do this now!

New folder + make it an RStudio Project

- `usethis::create_project("~/i_am_new")`
- RStudio > New Project... > New Directory > New Project

Make existing folder into an RStudio Project

- `usethis::create_project("~/i_exist")`
- RStudio > New Project... > Existing Directory

break here?

*Deep
Thoughts*



Practice "safe paths"

Practice "safe paths"

relative to a **stable base**

use **file system functions**,

not `paste()`, `strsplit()`, etc.

Do you know where your files are?



What is working directory?

We Sent Freddie to the Stratosphere
preserved thanks to the Wayback Machine



Our second launch was supposed to be The One, but we ran into a critical issue with some of our code.... The code worked fine as we were testing on command line since the location of PHP was known by our shell, but once the code was added to cron for automation, the location of PHP wasn't known, and the scripts continuously failed to send.



```
install.packages("fs")  
install.packages("here")
```

fs = filepath handling

here = project-relative paths

Examples of a stable base

- Project directory
 - `here::here("data", "raw-data.csv")`
 - `here::here("data/raw-data.csv")`
- User's home directory
 - `file.path("~/", ...)`
 - `fs::path_home(...)`
- Official location for installed s/w
 - `library(thingy)`
 - `system.file(..., package = "thingy")`

I have nothing against absolute paths.

Some of my best friends are absolute paths!

But don't hard-wire them into your scripts.

Instead, form at runtime relative to a stable base.

```
> (GOOD <- fs::path_home("tmp/test.csv")  
[1] "/Users/jenny/tmp/test.csv"
```

```
> (BAD <- "/Users/jenny/tmp/test.csv")  
[1] "/Users/jenny/tmp/test.csv"
```

Practice "safe paths"

Use the `here` package to build paths inside a project.

Leave working directory at top-level at all times,
during development.

Absolute paths are formed at runtime.

```
library(here)
#> here() starts at <snip, snip>/here_here
system("tree")
#> .
#>   └── one
#>     └── two
#>       └── awesome.txt

here("one", "two", "awesome.txt")
#> [1] "<snip, snip>/here_here/one/two/awesome.txt"
cat(readLines(here("one", "two", "awesome.txt")))
#> OMG this is so awesome!

setwd(here("one"))
getwd()
#> [1] "<snip, snip>/here_here/one"
here("one", "two", "awesome.txt")
#> [1] "<snip, snip>/here_here/one/two/awesome.txt"
cat(readLines(here("one", "two", "awesome.txt")))
#> OMG this is so awesome!
```

```
ggsave(here("figs", "built-barchart.png"))
```

Works on my machine, works on yours!

Works even if working directory is in a sub-folder

Works for RStudio projects, Git repos, R packages, ...

Works with knitr / rmarkdown

The `here` package is designed to work inside a project, where that could mean:

- RStudio Project
- Git repo
- R package
- Folder with a file named `.here`

`here()` does not create directories; that's your job.

Practice calling `here()` in a project to get a feel for it.

Or break here?

*Deep
Thoughts*



Names matter

machine readable

human readable

sort nicely



myabstract.docx

Joe's Filenames Use Spaces and Punctuation.xlsx

figure 1.png

homework1.R

JW7d^(2sl@deletethisandyourcareerisoverWx2*.txt



2018-01_bryan-abstract-rstudio-conf.docx

joes-filenames-are-getting-better.xlsx

fig01_scatterplot-talk-length-vs-interest.png

bryan_hw01.R

1986-01-28_raw-data-from-challenger-o-rings.txt

"machine readable"

regular expression and globbing friendly

- avoid spaces, punctuation, accented characters, case sensitivity

easy to compute on

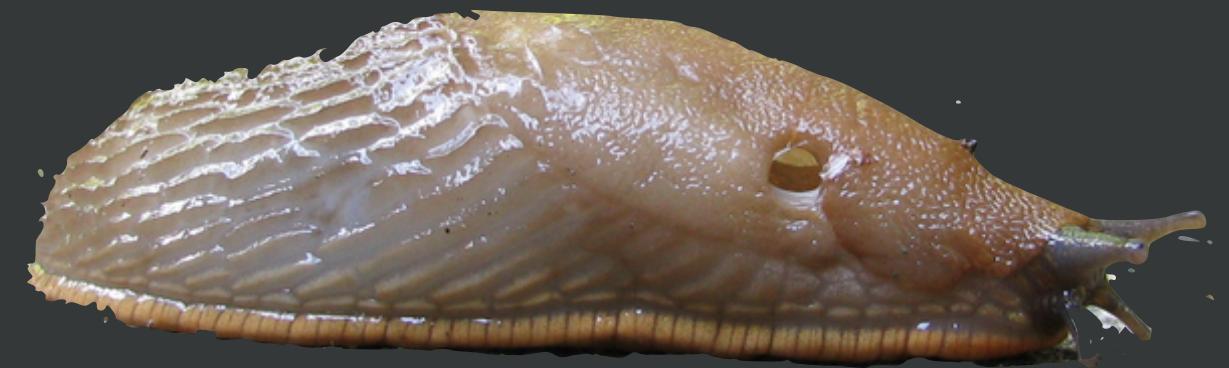
- deliberate use of delimiters

"human readable"

name contains info on content

name anticipates context

concept of a slug from user-friendly URLs



1986-01-28_raw-data-from-challenger-o-rings.txt

"sort nicely"

put something numeric in there

left pad with zeros for constant width, nice sorting

use the ISO 8601 standard for dates

order = chronological or ... consider common sense

```
> ft <- tibble(files = dir_ls(glob = "*.R"))
> ft
# A tibble: 6 x 1
  files
  <fs::path>
1 00_filesystem-practice_comfy.R
2 00_filesystem-practice_jenny.R
3 00_filesystem-practice_spartan.R
4 01_explore-libraries_comfy.R
5 01_explore-libraries_jenny.R
6 01_explore-libraries_spartan.R
```

👍 file names

```
> ft <- tibble(files = dir_ls(glob = "*.R"))
> ft
# A tibble: 6 x 1
  files
  <fs::path>
1 00_filesystem-practice_comfy.R
2 00_filesystem-practice_jenny.R
3 00_filesystem-practice_spartan.R
4 01_explore-libraries_comfy.R
5 01_explore-libraries_jenny.R
6 01_explore-libraries_spartan.R
```

Anyone can guess at file's purpose

```
> ft %>%  
+   filter(str_detect(files, "explore"))  
# A tibble: 3 × 1  
  files  
  <fs::path>  
1 01_explore-libraries_comfy.R  
2 01_explore-libraries_jenny.R  
3 01_explore-libraries_spartan.R
```

Easy to filter in R (or the shell or whatever)

```
> ft %>%  
+   mutate(files = path_ext_remove(files)) %>%  
+   separate(files, into = c("i", "topic", "flavor"), sep = "_")  
# A tibble: 6 x 3  
  i      topic        flavor  
* <chr> <chr>        <chr>  
1 00    filesystem-practice comfy  
2 00    filesystem-practice jenny  
3 00    filesystem-practice spartan  
4 01    explore-libraries   comfy  
5 01    explore-libraries   jenny  
6 01    explore-libraries   spartan
```

Intentional use of delimiters = meta-data easy to recover

"_" delimits fields

"-" delimits words so my eyes don't bleed

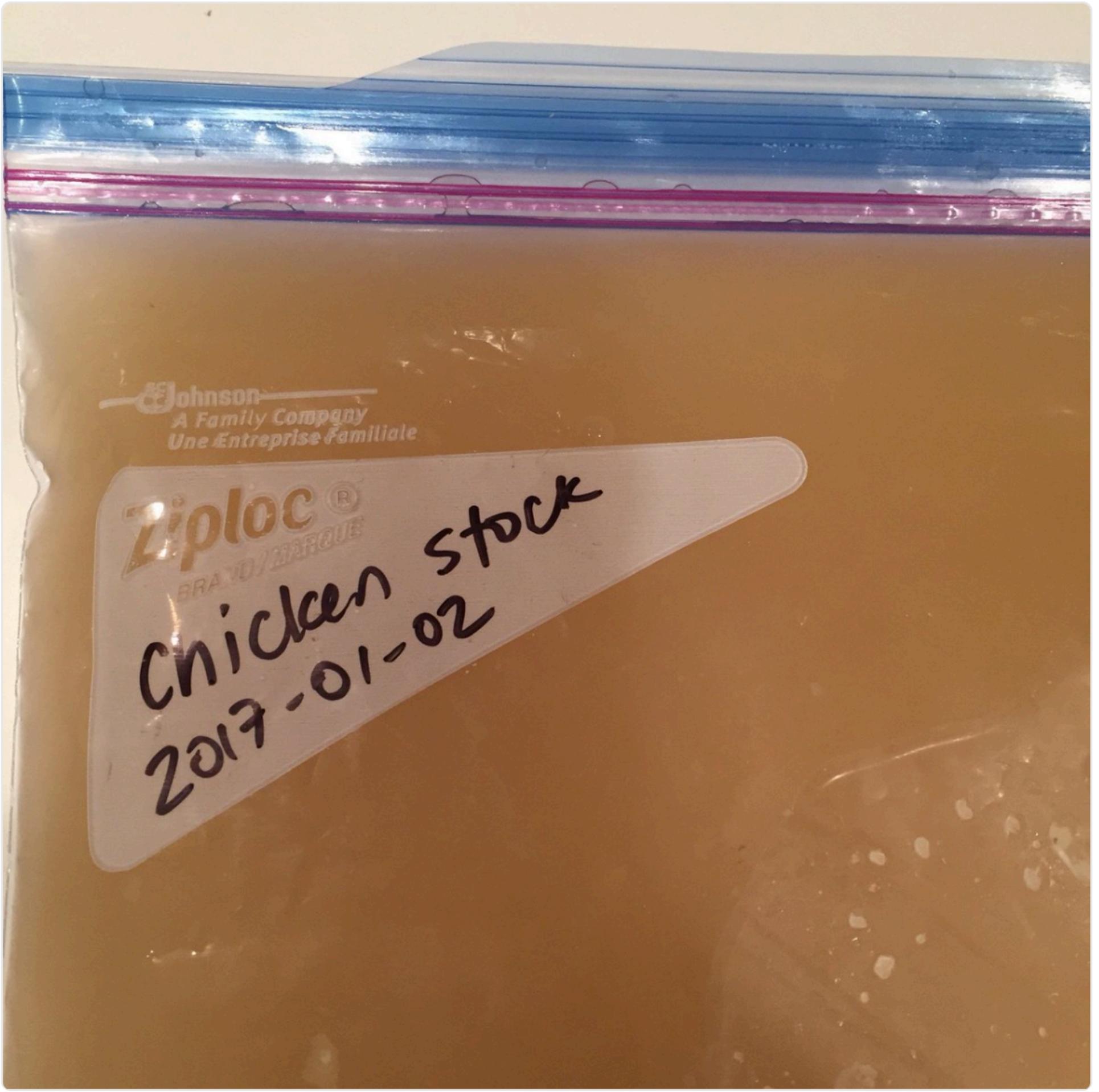
```
> dirs <- dir_ls(path_home("Desktop"), type = "directory")
> (dt <- tibble(dirs = path_file(dirs)))
# A tibble: 2 × 1
  dirs
  <fs::path>
1 day1_s1_explore-libraries
2 day1_s2_copy-files
```

Sorts in the same order as you
experience in real life



Jenny Bryan
@JennyBryan

I have an unwavering commitment to the ISO 8601 date standard. People of all nations can parse my freezer.



YYYY-MM-DD

ISO 8601

Names matter

machine readable

human readable

sort nicely

Names matter

easy to implement NOW

payoffs accumulate as your skills
evolve and projects get more complex

*Deep
Thoughts*



beware of
monoliths



break logic &
output into
pieces



smell-test.R

wrangle.R

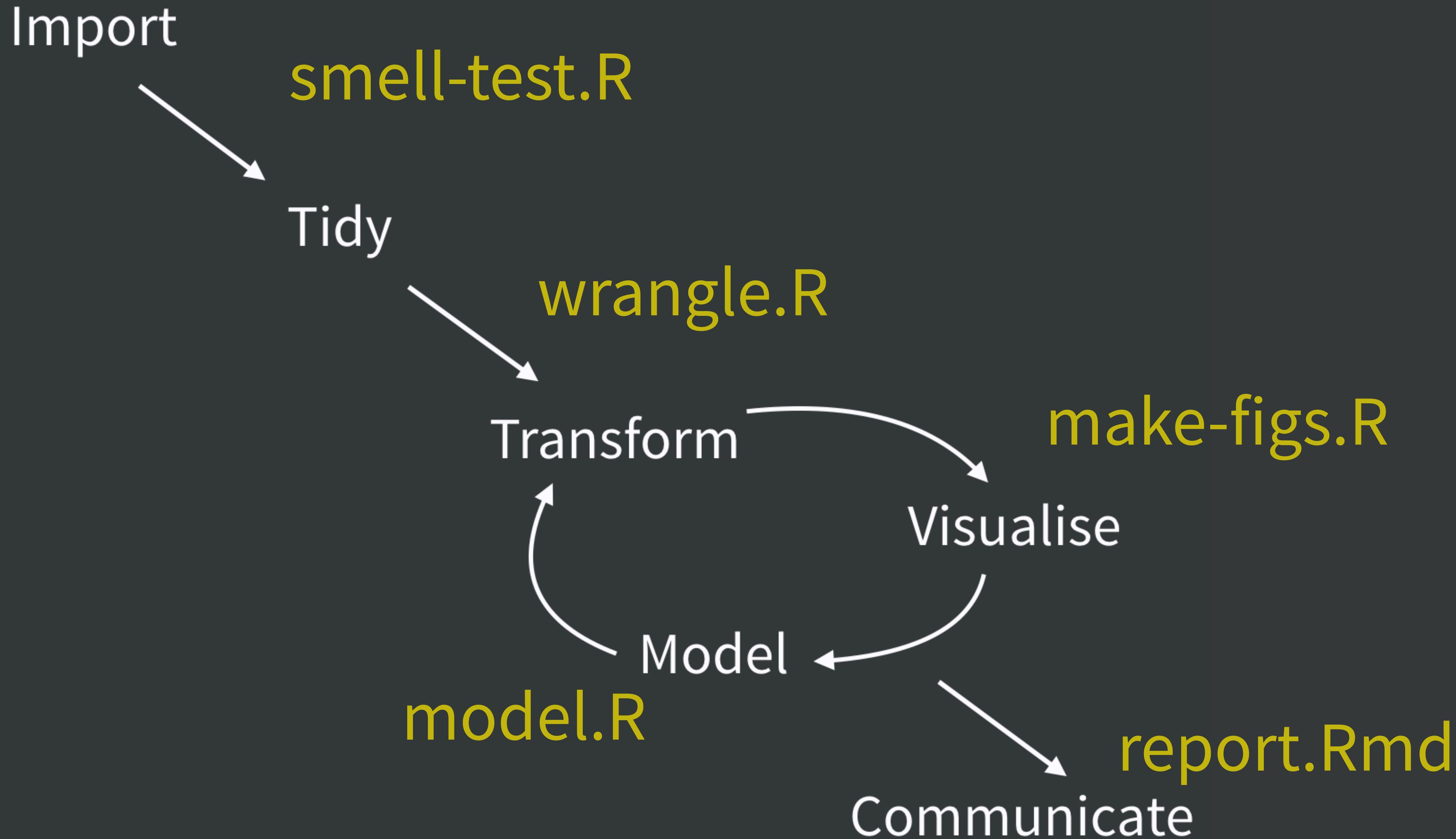
model.R

>>>

everything.R

make-figs.R

report.Rmd



raw-data.xlsx

data.csv

fits.rds

ests.CSV

>>>

.Rdata

raw-data.xlsx

Import



Tidy
data.csv

Transform

fits.rds

ests.csv



figs/hist.png
figs/dot.png

Visualise

Communicate

Input	Code	Output
raw data	smell-test.R	wisdom
raw data	wrangle.R	data.csv
data.csv	model.R	fits.rds ests.csv
data.csv fits.rds ests.csv	make-figs.R	figs/*
figs/* ests.csv	report.Rmd	report.html report.docx report.pdf

a humane API
for your analysis

```
library(usethis)  
use_course("rstd.io/wtf-packages-report")
```

- you know the drill
- download to same location as previous
- should open as RStudio project
- if not, you can make it an RStudio project

Project initiation strategies, the local case

New folder + make it an RStudio Project

- `usethis::create_project("~/i_am_new")`
- RStudio > New Project... > New Directory > New Project

Make existing folder into an RStudio Project

- `usethis::create_project("~/i_exist")`
- RStudio > New Project... > Existing Directory

You should now be in RStudio,
in the new RStudio Project named
wtf-packages-report.

How to launch an RStudio Project?

- double-click on .Rproj file
- use *File > Open Project* and friends
- use Project drop down in upper right corner
- Alfred trick 😊

We are here!

The screenshot shows the RStudio interface with the following details:

- File Explorer:** Shows the directory structure of the GitHub repository `wtf-packages-report`. The contents include `..`, `.gitignore`, `.Rhistory`, `data`, `figs`, `R`, `README.md`, `shhh-secret-for-later-README.Rmd`, and `wtf-packages-report.Rproj`.
- Global Environment:** Displays the message "Environment is empty".
- Code Editor:** Contains an `R` script titled `03_barchart-packages-built.R` with the following content:

```
1 # wtf-packages-report
2
3 * Have a look around the files here. Where are the R scripts?
  What are the directories `data` and `figs` for? Do the names
  and structure help you find things?
4 * Open each R script, finish it, and run it. Remember to
  restart R as you go, so you are certain each file is complete,
  i.e. data flows through explicit write/read, not the global
  workspace.
5   - `R/01_write-installed-packages.R`
6   - `R/02_wrangle-packages.R`
7   - `R/03_barchart-packages-built.R`
8 * It's OK if you don't finish! We can keep working on this
  later.
9 * If you finish quickly, write an R script to run the whole
```
- Console:** Displays the output of running the script:

```
1: I agree
2: No way
3: Not now

Selection: 1
✓ Creating GitHub repository
✓ Adding GitHub remote
✓ Pushing to GitHub and setting remote tracking branch
✓ Opening URL 'https://github.com/jennybc/wtf-packages-report'
```

work on challenge
README.md has instructions