R PROGRAMMING

for environmental health research

Brooke Anderson
Colorado State University

April 9, 2019

Today's plan

ORGANIZE

TRACK

PACKAGE

COLLECT

Homework?!

https://bit.ly/2WQV6XT

PREREQUISITES

Setting up

Install RStudio Desktop

https://www.rstudio.com/

Install git

https://git-scm.com/downloads

Create GitHub account

https://github.com/

Download example project

https://github.com/geanders/columbia_env_health_examples

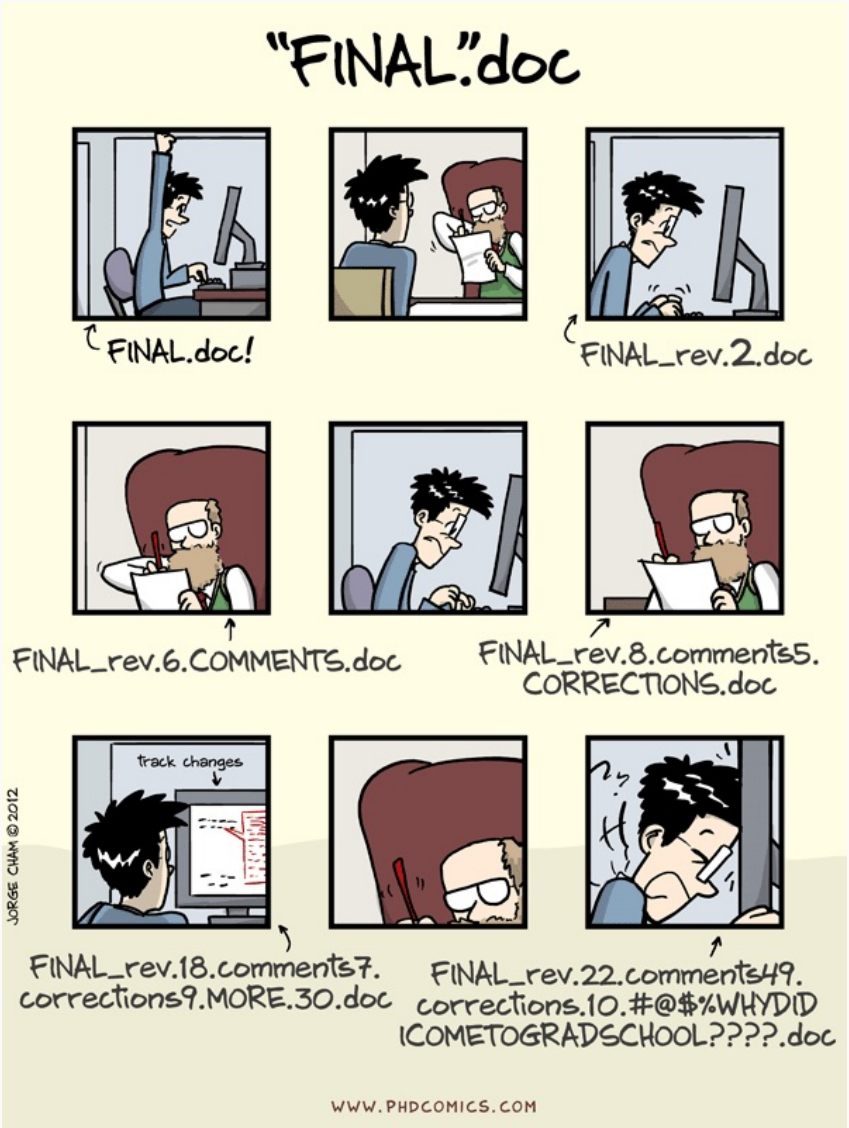
ORGANIZE

Using **R projects** to organize research files

One file : One version

Rule #1 of research project file organization

Organizing research project files

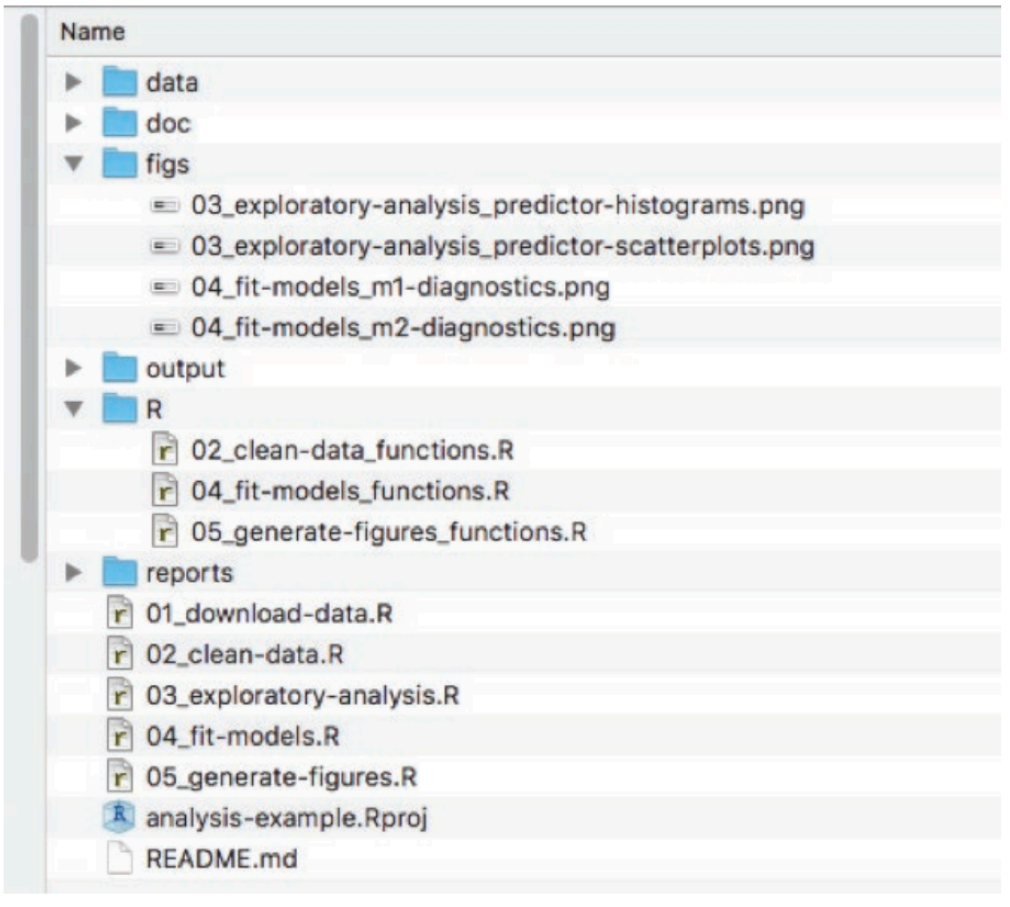


Source: PhDComics

One project : One directory

Rule #2 of research project file organization

Example project directory



Source: “A Guide to Reproducible Code in Ecology and Evolution”. British Ecological Society.

Use consistent names

Rule #3 of research project file organization

Common project subdirectories

data-raw Raw data and R scripts to clean the raw data.

data Cleaned data, often saved as `.RData` after being generated by a script in `data-raw`.

R Code for any functions used in analysis.

figures Figures created from R code.

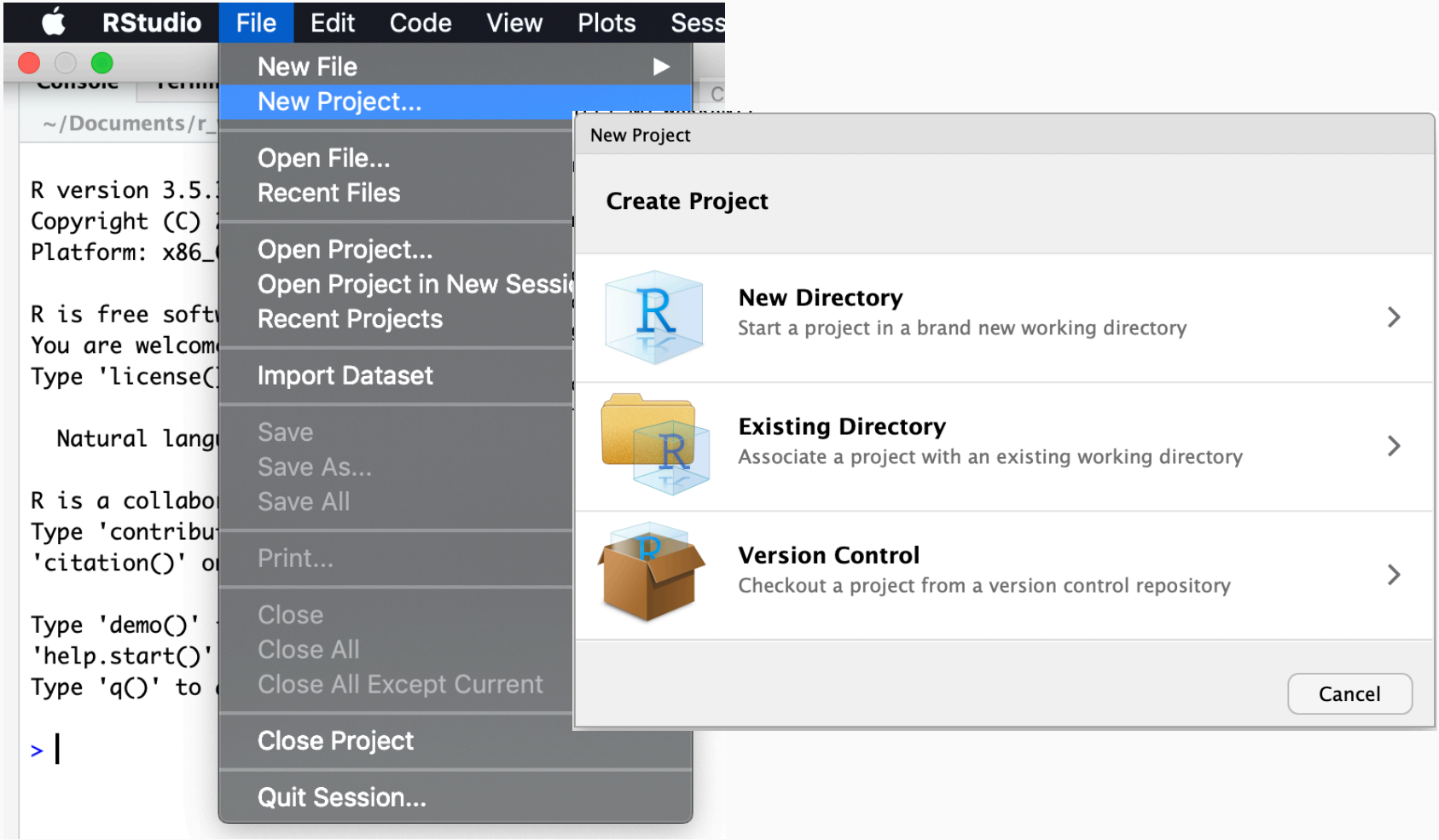
reports R Markdown files and products rendered from those files (e.g., paper drafts, presentations).

Use relative filenames

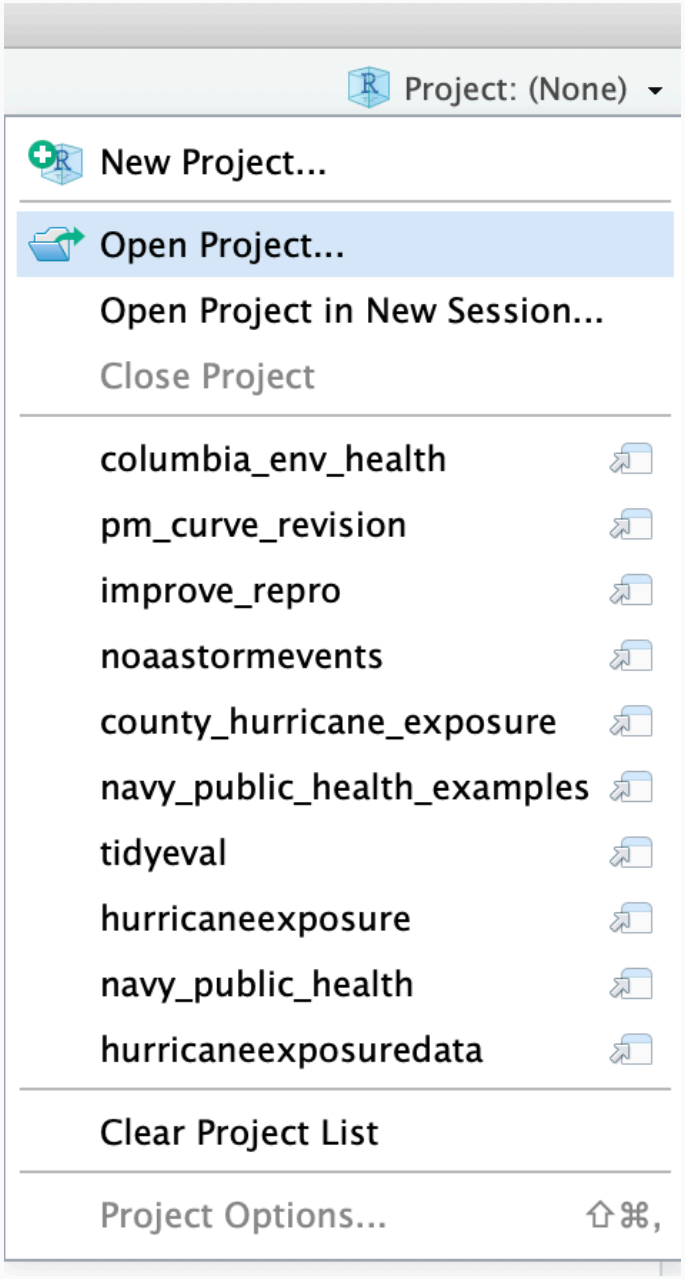
Rule #4 of research project file organization

```
data/my_data_file.csv
figures/figure_1.png
```

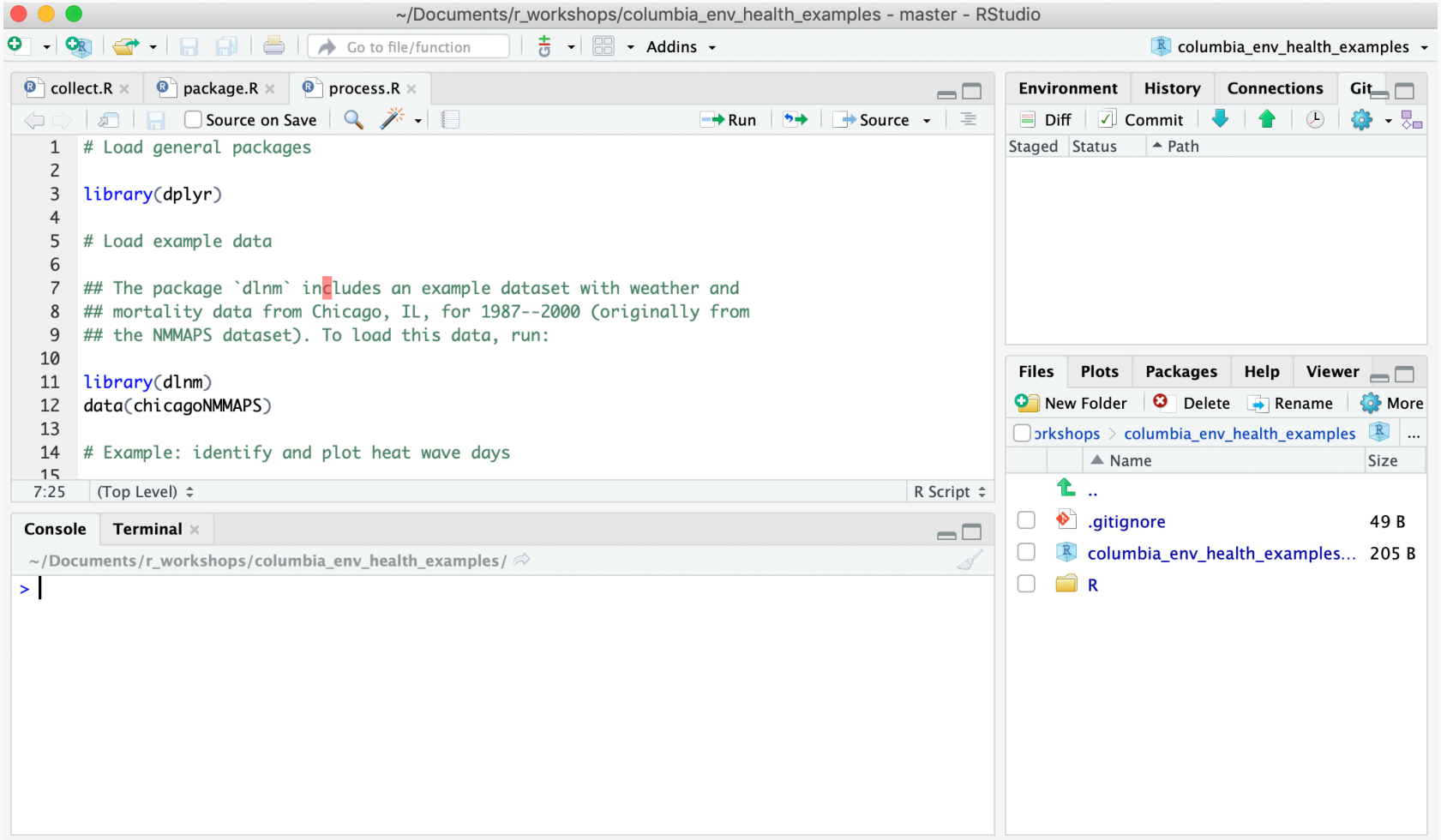
Create R project



Navigating R Projects



Navigating R Projects

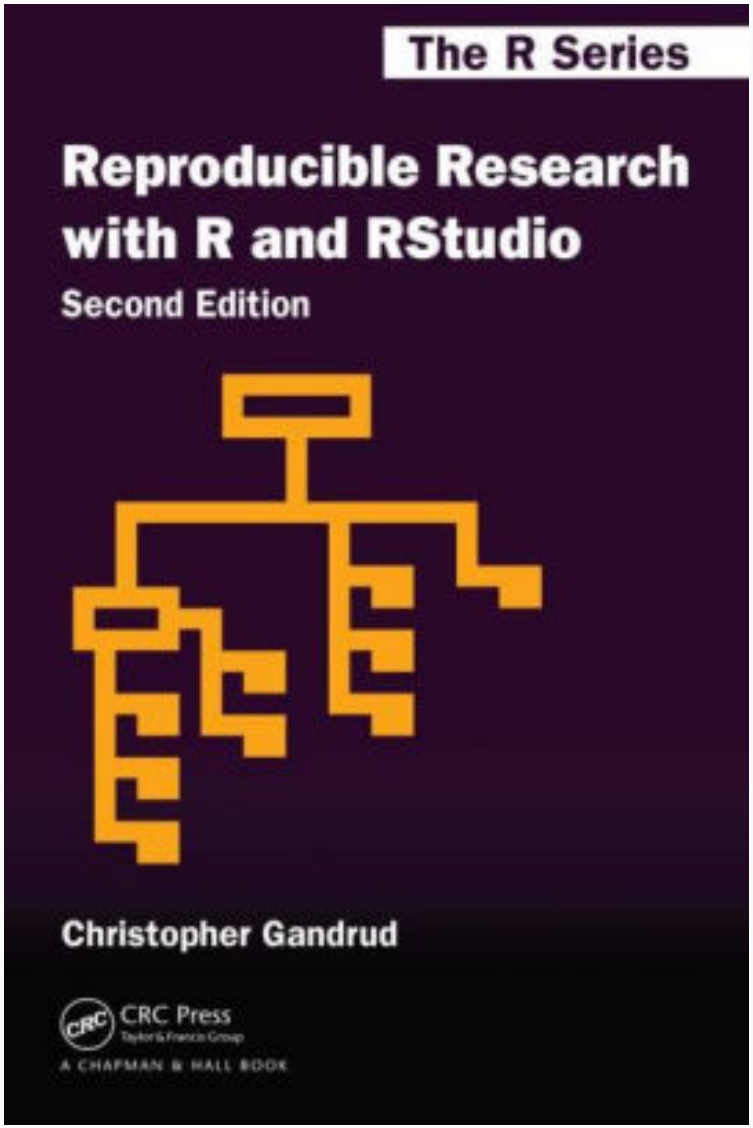


```
Georgianas-MacBook-Pro:columbia_env_health georgianaanderson$ ls -a
.                               _build.sh
..                              _output.yml
.DS_Store                       _workshop_slides
.Rproj.user                     book.bib
.git                            columbia_env_health.Rproj
.gitignore                      columbia_env_health.log
.nojekyll                       data
01-organize.Rmd                 docs
02-track.Rmd                    flexdashboard
03-package.Rmd                  images
04-collect.Rmd                  index.Rmd
05-process.Rmd                  irma_fatalities.pdf
06-summary.Rmd                  now.json
DESCRIPTION                     old_data
LICENSE                         packages.bib
R                               preamble.tex
README.md                       skeleton.bib
_bookdown.yml                   style.css
_bookdown_files                 toc.css
```

.Rproj/

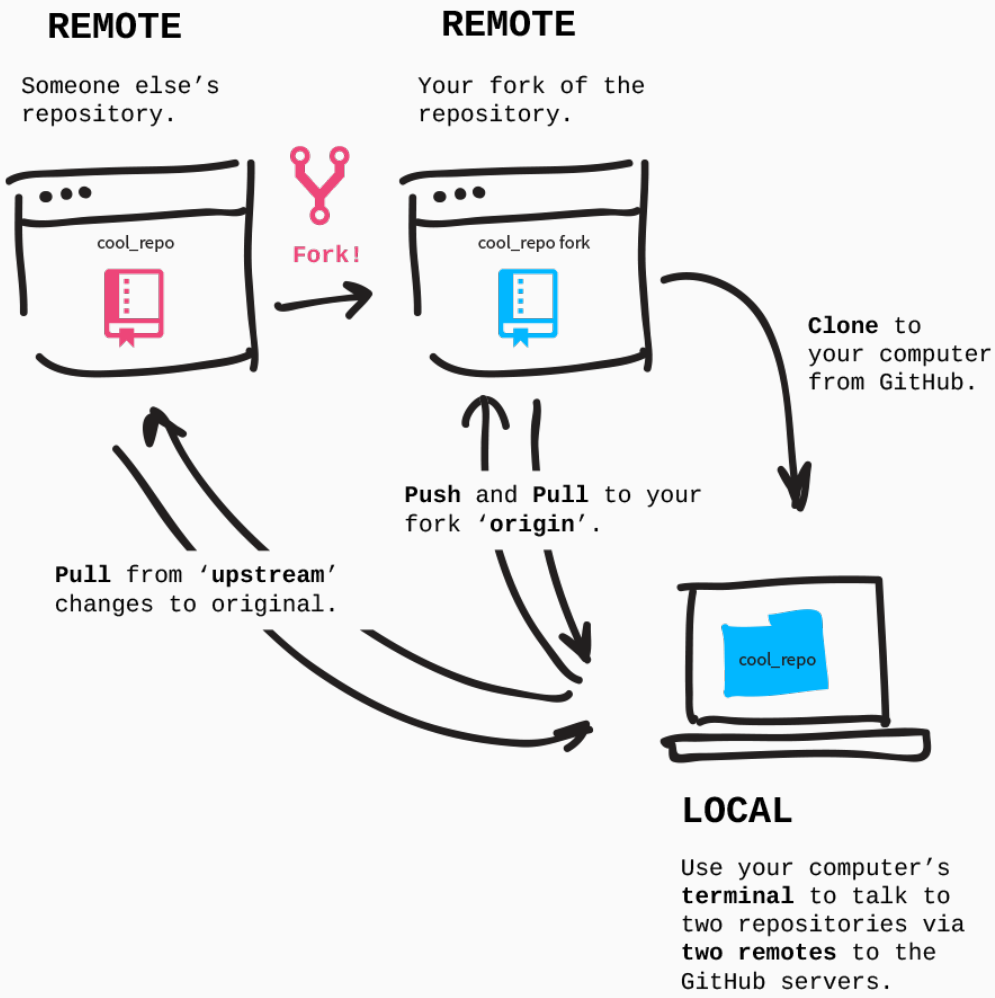
[Live coding example]

Resources


The R Series
Reproducible Research with R and RStudio
Second Edition
Christopher Gandrud
CRC Press
A CHAPMAN & HALL BOOK

TRACK

git and GitHub for version control



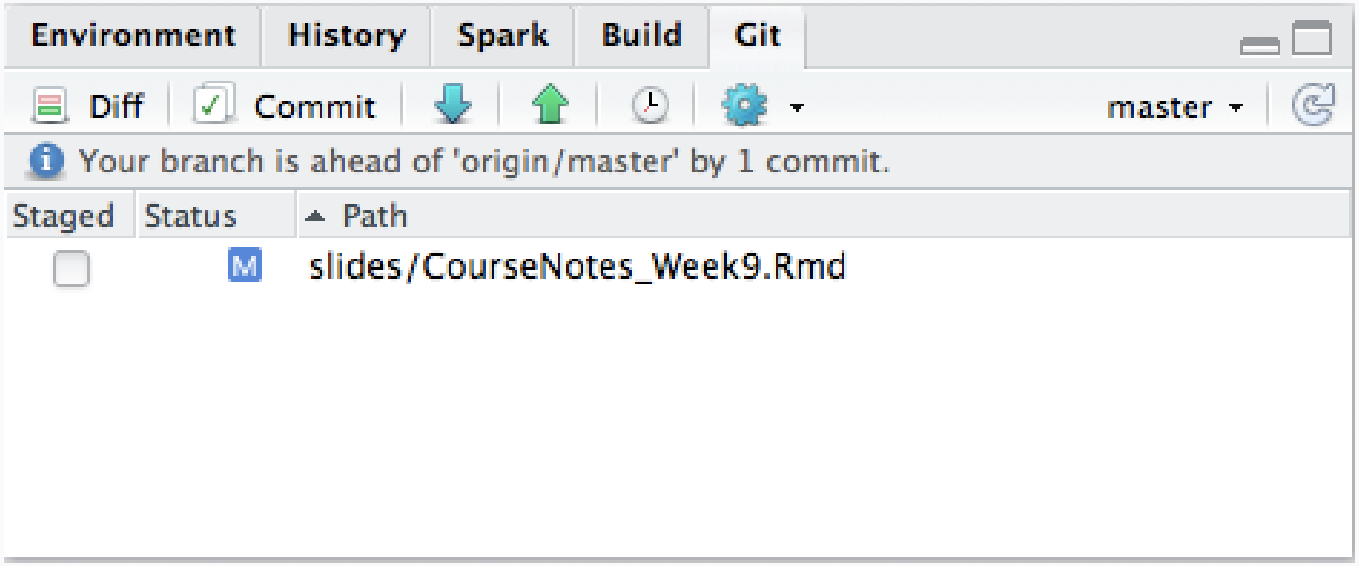
Source: GitHub

```
Georgianas-MacBook-Pro:columbia_env_health georgianaanderson$ ls -a
.                               _build.sh
..                              _output.yml
.DS_Store                       _workshop_slides
.Rproj.user                     book.bib
.git                            columbia_env_health.Rproj
.gitignore                      columbia_env_health.log
.nojekyll                       docs
01-organize.Rmd                 flexdashboard
02-track.Rmd                    images
03-package.Rmd                  index.Rmd
04-collect.Rmd                  irma_fatalities.pdf
05-process.Rmd                  now.json
06-summary.Rmd                  old_data
DESCRIPTION                     packages.bib
LICENSE                         preamble.tex
R                               skeleton.bib
README.md                       style.css
_bookdown.yml                   toc.css
_bookdown_files
```

.git/

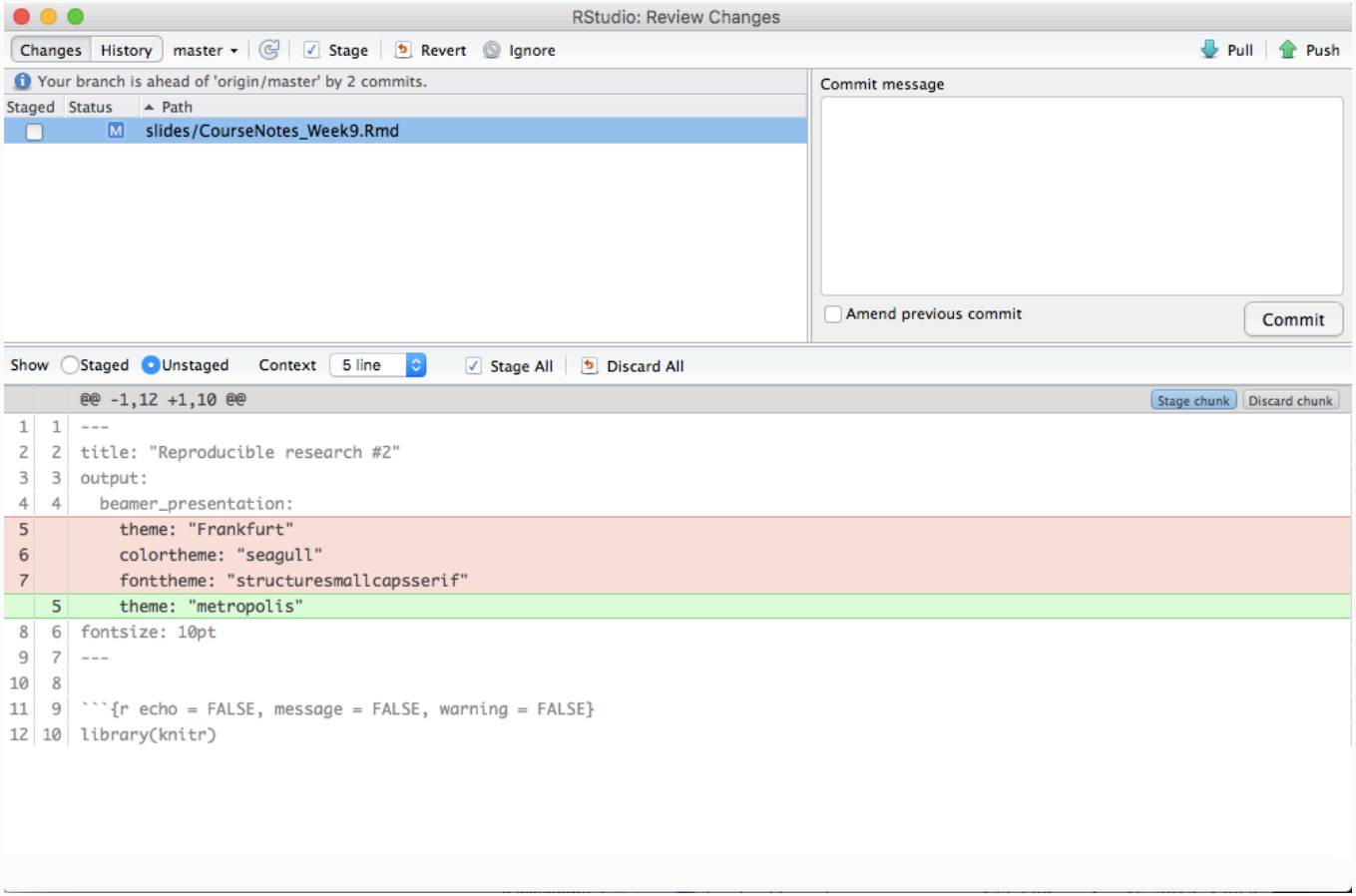
Git commits in RStudio

Once you initialize the project as a git repository, you should have a “Git” window in one of your RStudio panes (top right pane by default).



Git commits in RStudio

When you click on“Commit”, it opens a separate commit window :

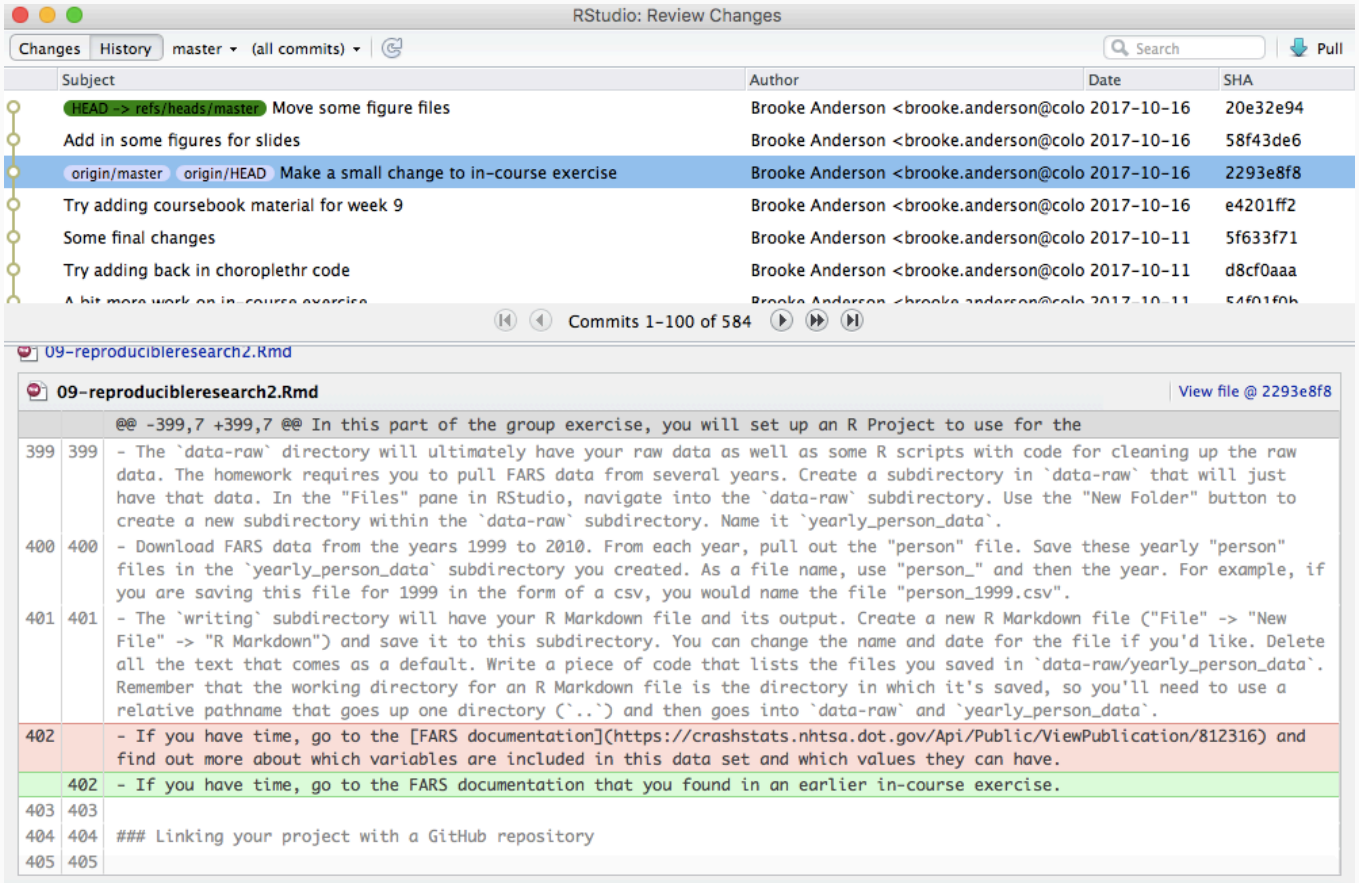


Commit messages

Source: xkcd

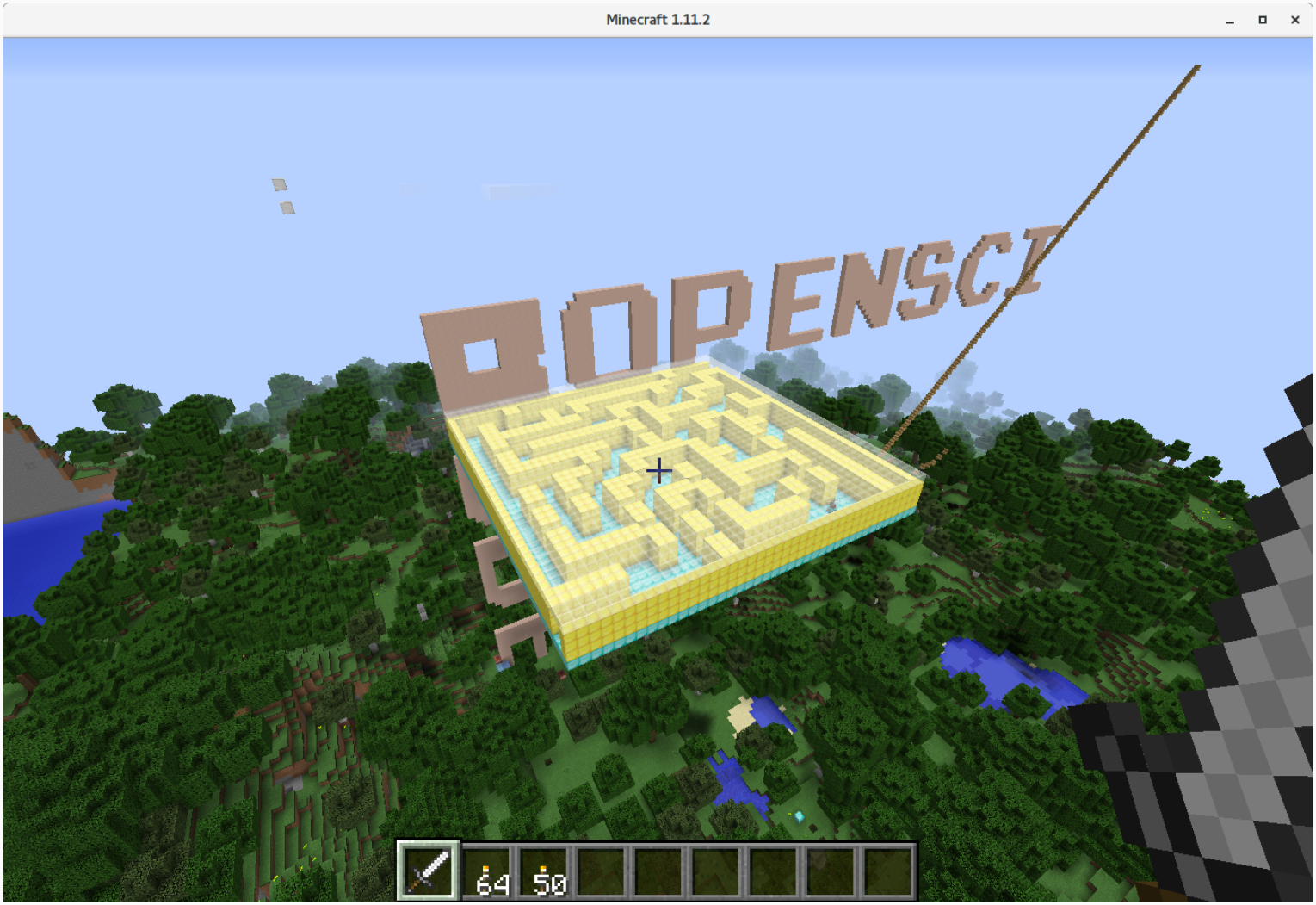
Browsing history

On the top left of the Commit window, you can toggle to “History”:



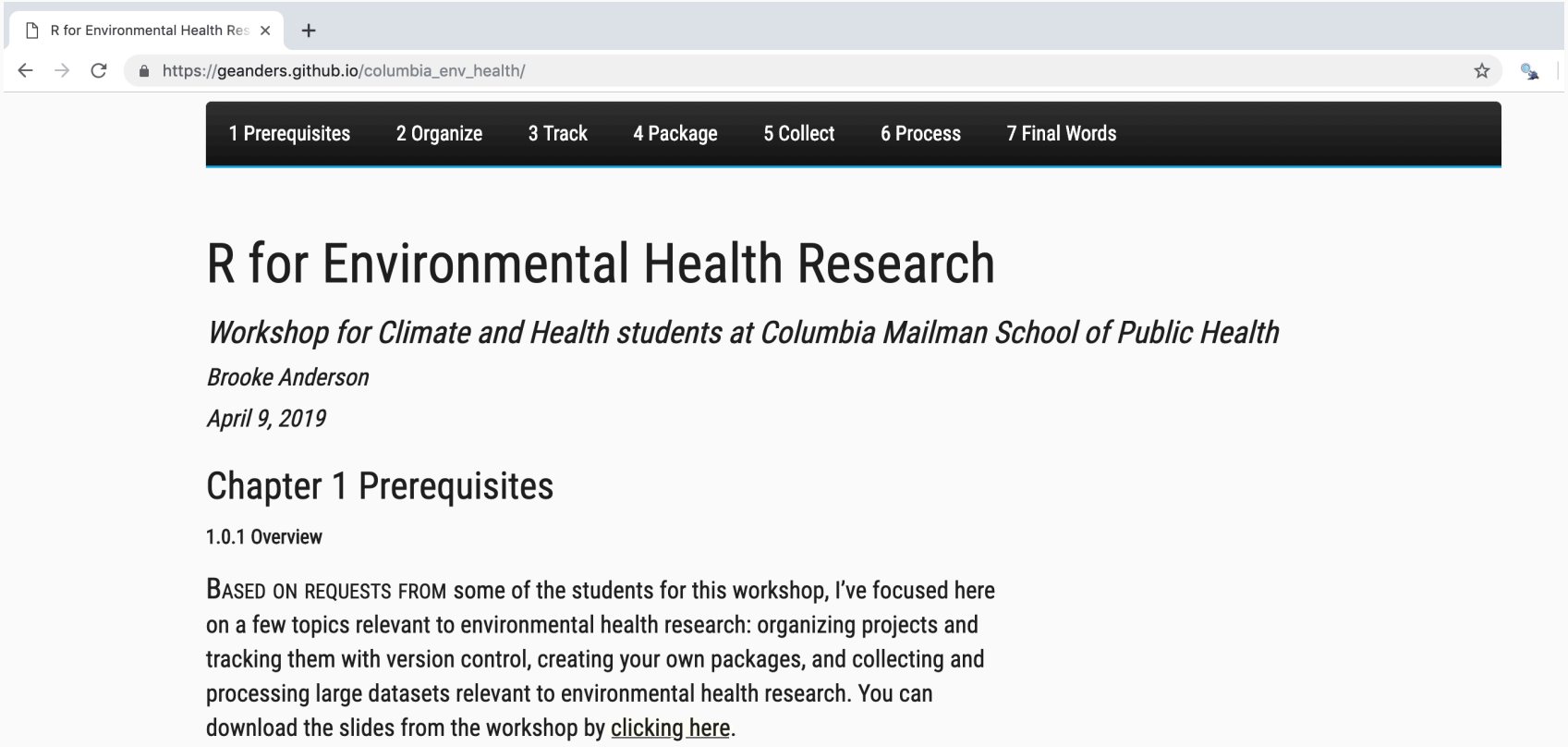
[Live coding example]

Using GitHub to collaborate



https://github.com/ropenscilabs/miner

Hosting content with GitHub Pages



PACKAGE

Collect R functions in **packages**

Why write R packages

Software development in biostatistics

So I have a new policy when evaluating CV's of candidates for jobs, or when I'm reading a paper as a referee. If the paper is about a new statistical method or machine learning algorithm and there is no software available for that method - I simply mentally cross it off the CV. If I'm reading a data analysis and there isn't code that reproduces their analysis - I mentally cross it off. In my mind, new methods/analyses without software are just vapor ware. Now, you'd definitely have to cross a few papers off my CV, based on this principle. I do that. But I'm trying really hard going forward to make sure nothing gets crossed off.

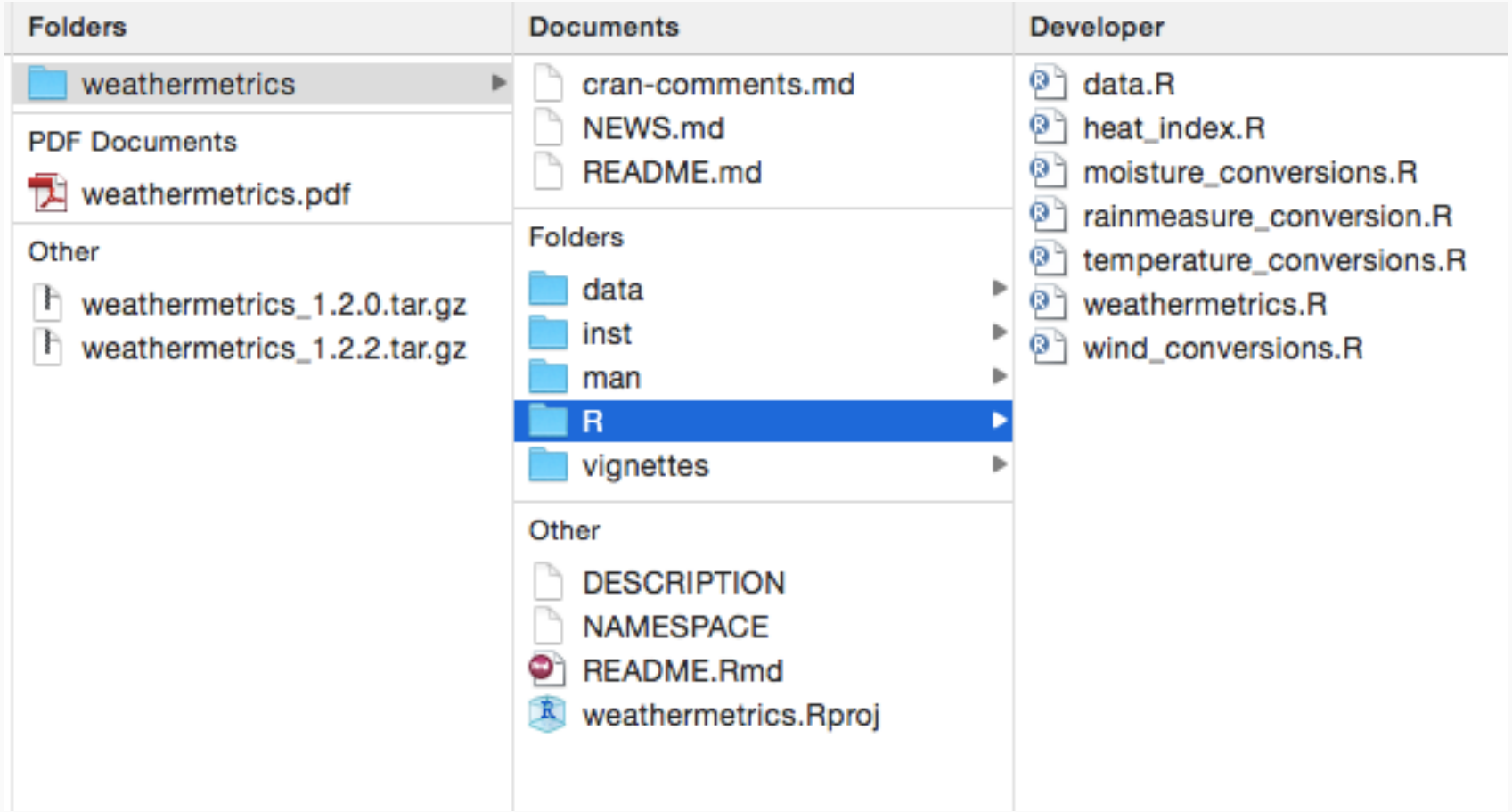
Source: Jeff Leek, Simply Statistics

Why write R packages

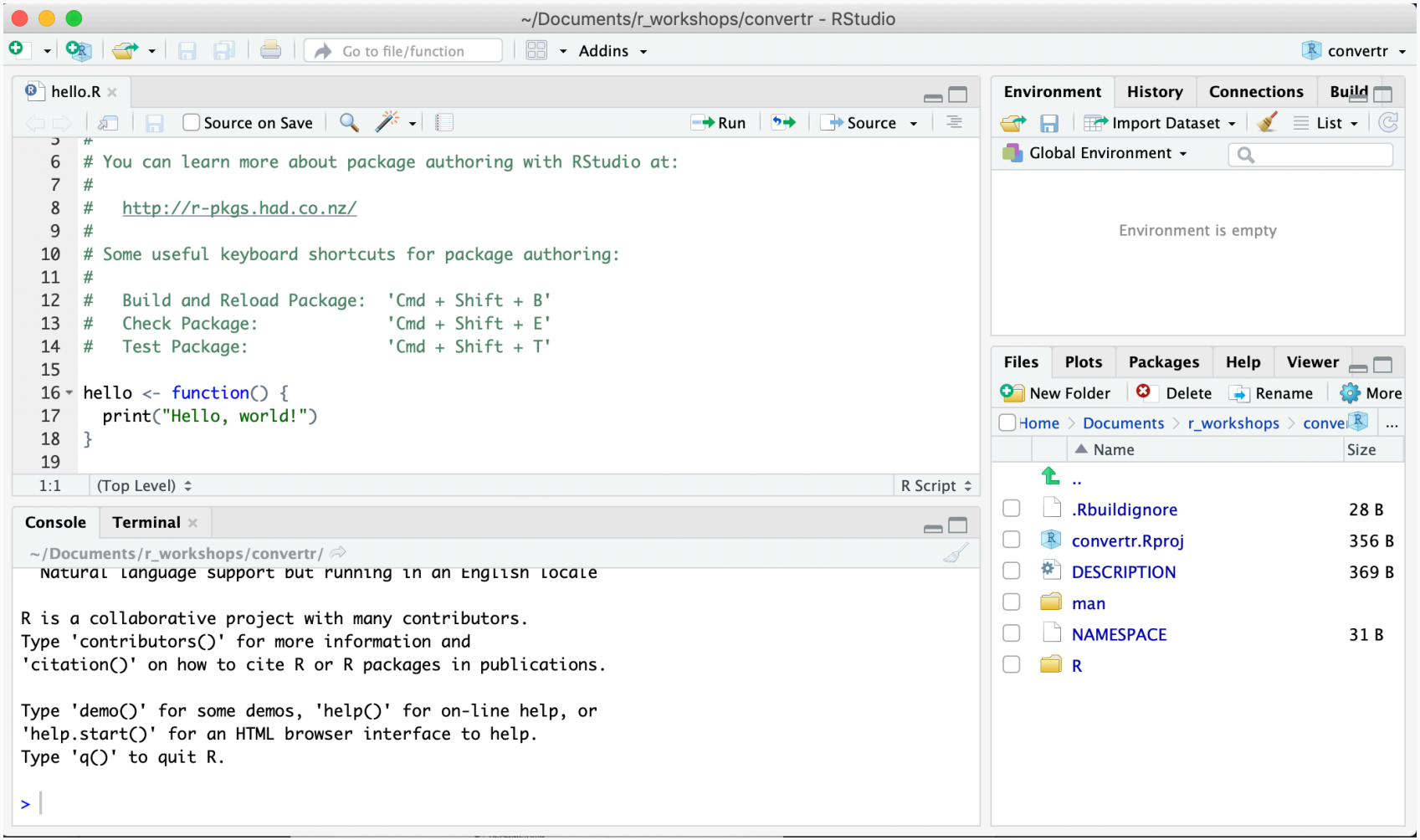
Research impacts of NMMAPS package (*Source: Barnett, Huang, and Turner, "Benefits of Publicly Available Data", Epidemiology 2012*):

> As of November 2011, 67 publications had been published using this data, with 1,781 citations to these papers
>
> Research using NMMAPS has been used by the US EPA in creating regulatory impact statements for air pollution (particulates and ozone)
>
> "Thanks to NMMAPS, there is probably no other country in the world with a greater understanding of the health effects of air pollution and heat waves in its population."

What an R package looks like



R package template



Required files

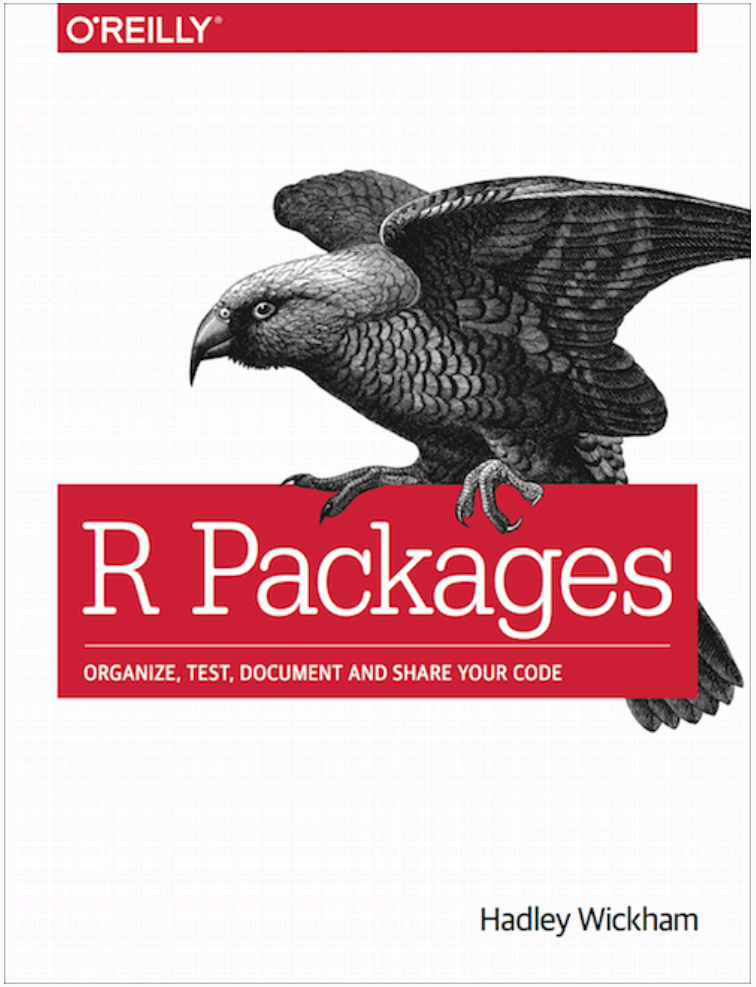
R/ or **data/** If you don't have one of these, your package won't do anything

DESCRIPTION Needed, but you can't keep the template version as-is

NAMESPACE Needed, but you can't keep the template version as-is

[Live coding example]

Resources



http://r-pkgs.had.co.nz/

Resources

Package Development: : CHEAT SHEET

Package Structure

A package is a convention for organizing files into directories.

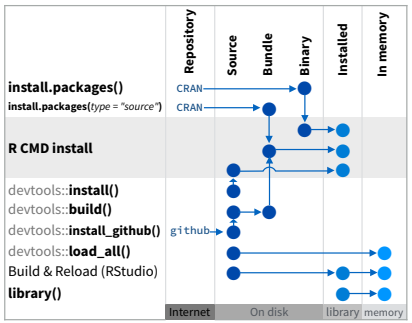
This sheet shows how to work with the 7 most common parts of an R package:

Package	
DESCRIPTION	SETUP
R/	WRITE CODE
tests/	TEST
man/	DOCUMENT
vignettes/	TEACH
data/	ADD DATA
NAMESPACE	ORGANIZE

The contents of a package can be stored on disk as a:

- **source** - a directory with sub-directories (as above)
- **bundle** - a single compressed file (*.tar.gz*)
- **binary** - a single compressed file optimized for a specific OS

Or installed into an R library (loaded into memory during an R session) or archived online in a repository. Use the functions below to move between these states.



devtools::**use_build_ignore**(*"file"*)
Adds file to .Rbuildignore, a list of files that will not be included when package is built.

Setup (DESCRIPTION)

The DESCRIPTION file describes your work, sets up how your package will work with other packages, and applies a copyright.

- You must have a DESCRIPTION file
- Add the packages that yours relies on with
  devtools::**use_package()**
  Adds a package to the Imports or Suggests field

CC0	MIT	GPL-2
No strings attached.	MIT license applies to your code if re-shared.	GPL-2 license applies to your code, *and all code anyone bundles with it*, if re-shared.

```
Package: mypackage
Title: Title of Package
Version: 0.1.0
Authors@R: person("Hadley", "Wickham", email =
    "hadley@me.com", role = c("aut", "cre"))
Description: What the package does (one paragraph)
Depends: R (>= 3.1.0)
License: GPL-2
LazyData: true
Imports:
    dplyr (>= 0.4.0),
    ggvis (>= 0.2)
Suggests:
    knitr (>= 0.1.0)
```

Import packages that your package *must have* to work. R will install them when it installs your package.

Suggest packages that are not very essential to yours. Users can install them manually, or not, as they like.

Write Code (R/)

All of the R code in your package goes in R/. A package with just an R/ directory is still a very useful package.

- Create a new package project with
  devtools::**create**("path/to/name")
  Create a template to develop into a package.
- Save your code in R/ as scripts (extension .R)

WORKFLOW

1. Edit your code.
2. Load your code with one of
   devtools::**load_all()**
   Re-loads all *saved* files in R/ into memory.
   Ctrl/Cmd + Shift + L (keyboard shortcut)
   Saves all open files then calls load_all().
3. Experiment in the console.
4. Repeat.

- Use consistent style with **r-pkgs.had.co.nz/r.html#style**
- Click on a function and press **F2** to open its definition
- Search for a function with **Ctrl + .**

Visit **r-pkgs.had.co.nz** to learn much more about writing and publishing packages for R

Test (tests/)

Use tests/ to store tests that will alert you if your code breaks.

- Add a **tests/** directory
- Import **testthat** with devtools::**use_testthat()**, which sets up package to use automated tests with testthat
- Write tests with **context()**, **test()**, and expect statements
- Save your tests as .R files in **tests/testthat/**

WORKFLOW

1. Modify your code or tests.
2. Test your code with one of
   devtools::**test()**
   Runs all tests in tests/
   Ctrl/Cmd + Shift + T
   (keyboard shortcut)
3. Repeat until all tests pass

Example Test

```
context("Arithmetic")

test_that("Math works", {
  expect_equal(1 + 1, 2)
  expect_equal(1 + 2, 3)
  expect_equal(1 + 3, 4)
})
```

Expect statement	Tests
expect_equal()	is equal within small numerical tolerance?
expect_identical()	is exactly equal?
expect_match()	matches specified string or regular
expect_output()	prints specified output?
expect_message()	displays specified message?
expect_warning()	displays specified warning?
expect_error()	throws specified error?
expect_is()	output inherits from certain class?
expect_false()	returns FALSE?
expect_true()	returns TRUE?

RStudio

https://www.rstudio.com/resources/cheatsheets/

Resources

COLLECT

Leverage **open data** tools for collecting data

Data packages

```
library(hurricaneexposure)
county_wind(counties = "36061",
            start_year = 1988, end_year = 2015,
            wind_limit = 17.5) %>%
  select(storm_id, vmax_sust, storm_dist, closest_date)
```

```
##      storm_id vmax_sust storm_dist closest_date
## 1    Bob-1991  18.19559 161.571830   1991-08-19
## 2 Bertha-1996  28.95496  16.966013   1996-07-13
## 3  Floyd-1999  20.50178  45.408483   1999-09-16
## 4  Hanna-2008  19.25390  29.916672   2008-09-06
## 5  Irene-2011  25.68553   5.796733   2011-08-28
## 6  Sandy-2012  21.99213 158.040788   2012-10-29
```

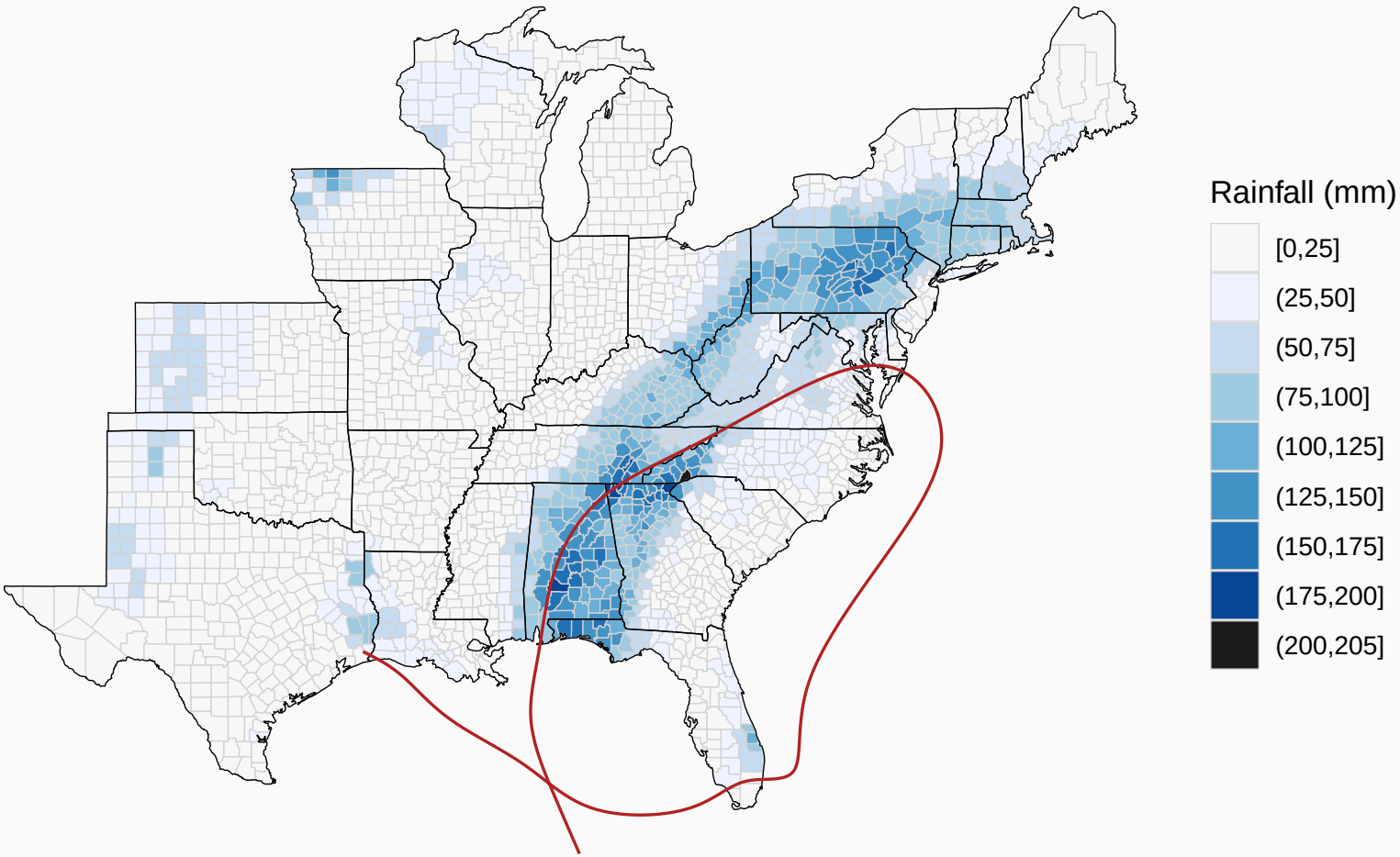
Data packages

```
county_events(counties = "36061",
              start_year = 1988, end_year = 2015,
              event_type = "flood") %>%
  select(storm_id, storm_dist, closest_date)
```

```
##       storm_id storm_dist closest_date
## 1   Floyd-1999  45.408483   1999-09-16
## 2 Allison-2001 158.909890   2001-06-17
## 3 Frances-2004 379.343696   2004-09-09
## 4    Ivan-2004 311.346881   2004-09-18
## 5  Jeanne-2004 222.900157   2004-09-29
## 6   Beryl-2006 207.358443   2006-07-20
## 7   Barry-2007 148.251718   2007-06-04
## 8   Irene-2011   5.796733   2011-08-28
## 9  Andrea-2013  92.381282   2013-06-08
```

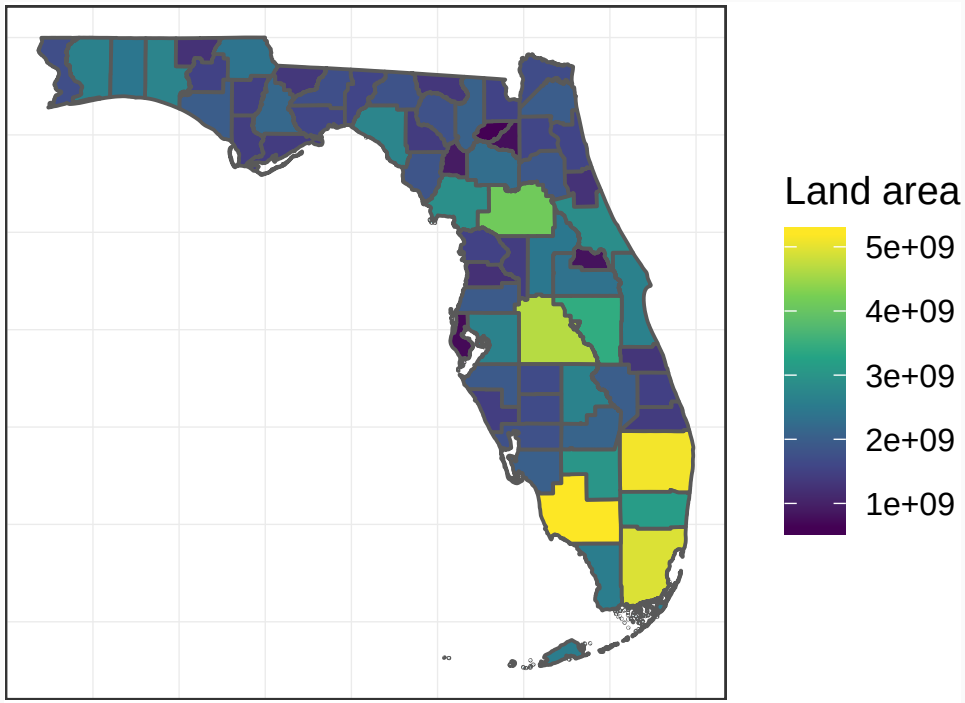
Data packages

```
map_counties(storm = "Ivan-2004", metric = "rainfall")
```



Open Data APIs

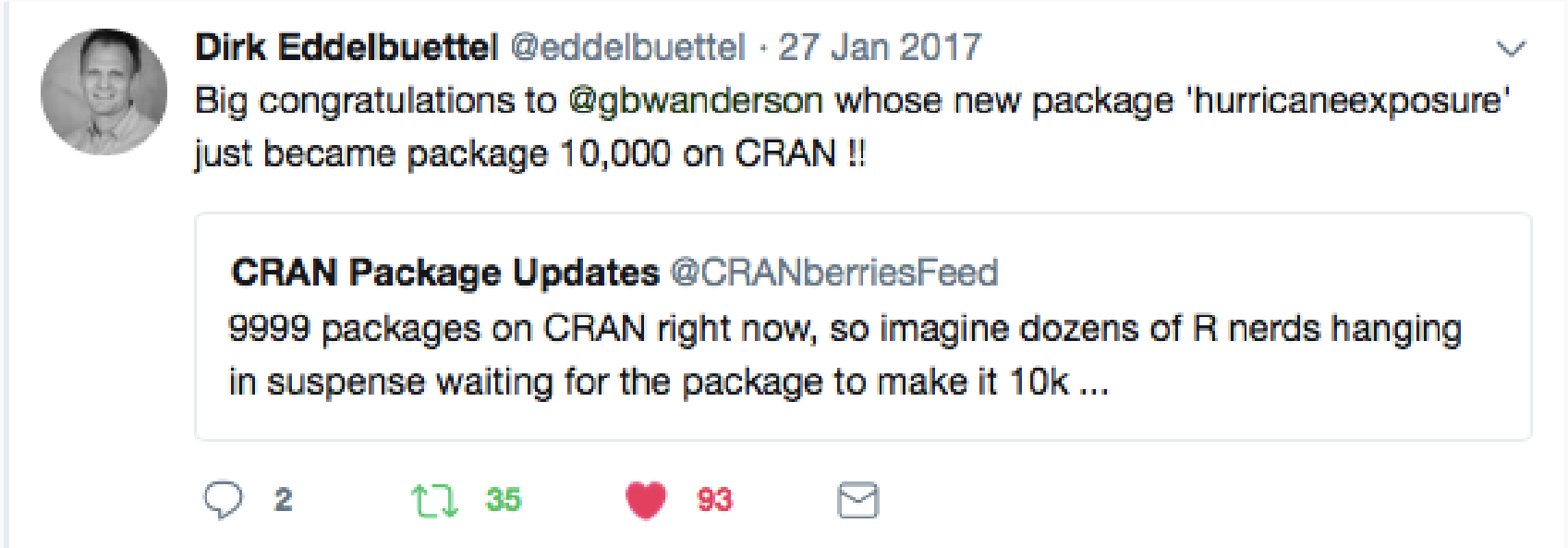
```
library(tigris)
fl_counties <- counties(state = "FL",
                        class = "sf")
```



[Live coding example]

Resources

#rstats



Resources

ROpenSci

Homework!!

https://bit.ly/2WQV6XT