

# 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](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

Name
▶ data
▶ doc
▼ figs
└ 03_exploratory-analysis_predictor-histograms.png
└ 03_exploratory-analysis_predictor-scatterplots.png
└ 04_fit-models_m1-diagnostics.png
└ 04_fit-models_m2-diagnostics.png
▶ output
▼ R
└ 02_clean-data_functions.R
└ 04_fit-models_functions.R
└ 05_generate-figures_functions.R
▶ reports
└ 01_download-data.R
└ 02_clean-data.R
└ 03_exploratory-analysis.R
└ 04_fit-models.R
└ 05_generate-figures.R
└ analysis-example.Rproj
└ README.md

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).

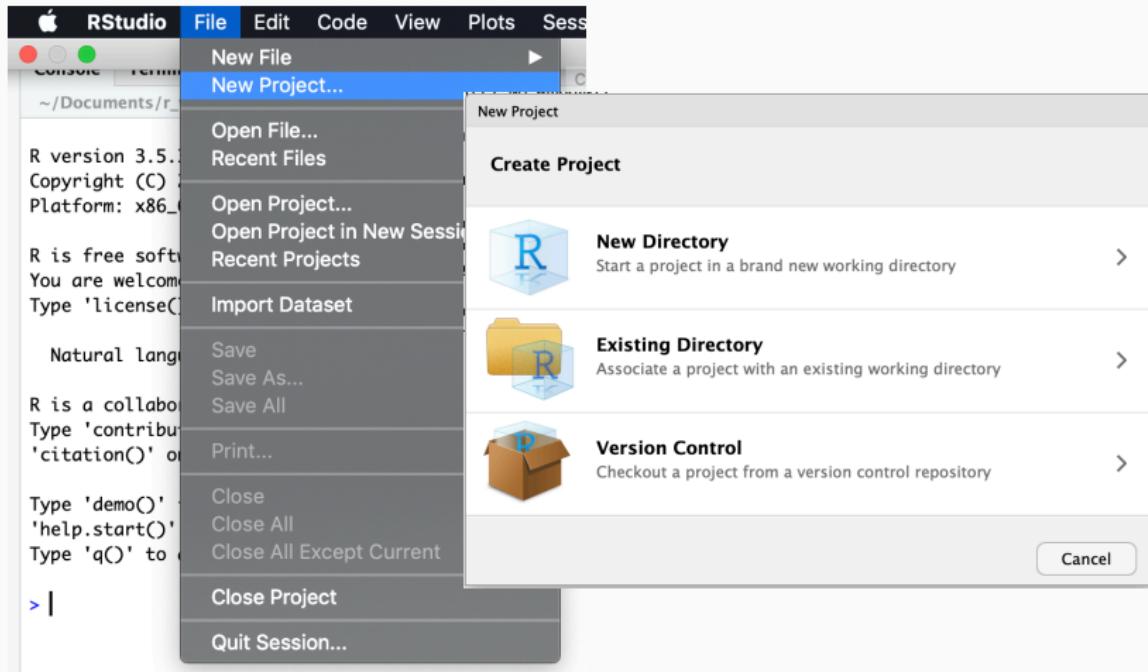
# Use relative filenames

**Rule #4** of research project file organization

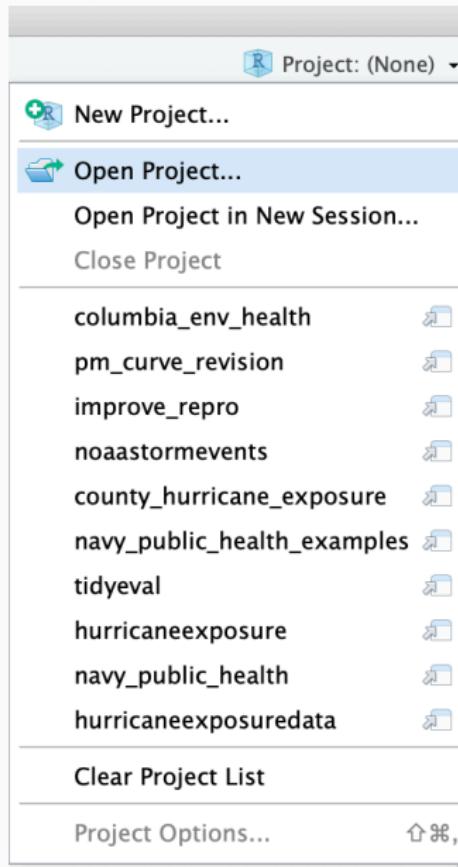
data/my\_data\_file.csv

figures/figure\_1.png

# Create R project



# Navigating R Projects



# Navigating R Projects

The screenshot shows the RStudio interface with the following details:

- Title Bar:** ~Documents/r\_workshops/columbia\_env\_health\_examples - master - RStudio
- File Explorer:** Shows the project structure under 'columbia\_env\_health\_examples'. It includes files like '.gitignore', 'columbia\_env\_health\_examples.Rproj', and a folder 'R'.
- Code Editor:** Displays three R script files: 'collect.R', 'package.R', and 'process.R'. The code in 'collect.R' is as follows:

```
1 # Load general packages
2
3 library(dplyr)
4
5 # Load example data
6
7 ## The package `dlm` includes an example dataset with weather and
8 ## mortality data from Chicago, IL, for 1987--2000 (originally from
9 ## the NMMAPS dataset). To load this data, run:
10
11 library(dlm)
12 data(chicagoNMMAPS)
13
14 # Example: identify and plot heat wave days
15
```

- Console:** Shows the command: ~/Documents/r\_workshops/columbia\_env\_health\_examples/
- Terminal:** Shows the command: > |
- Environment Tab:** Contains tabs for Environment, History, Connections, and Git.
- Plots Tab:** Shows a small preview of a scatter plot.
- Packages Tab:** Shows a list of installed packages.
- Help Tab:** Shows help documentation for 'lapply'.
- Viewer Tab:** Shows a preview of a data frame.

```
[Georgianas-MacBook-Pro:columbia_env_health georgianaanderson$ ls -a
.
..
.DS_Store
.Rproj.user
.git
.gitignore
.nojekyll
01-organize.Rmd
02-track.Rmd
03-package.Rmd
04-collect.Rmd
05-process.Rmd
06-summary.Rmd
DESCRIPTION
LICENSE
R
README.md
_bookdown.yml
_bookdown_files
```

# .Rproj/

```
_build.sh
_output.yml
_workshop_slides
book.bib
columbia_env_health.Rproj
columbia_env_health.log
data
flexdashboard
images
index.Rmd
irma_fatalities.pdf
now.json
old_data
packages.bib
preamble.tex
skeleton.bib
style.css
toc.css
```

```
irma_week_accs <- fl_accidents %>%  
  group_by(fips) %>%  
  summarize(fatals = sum(fatals))
```

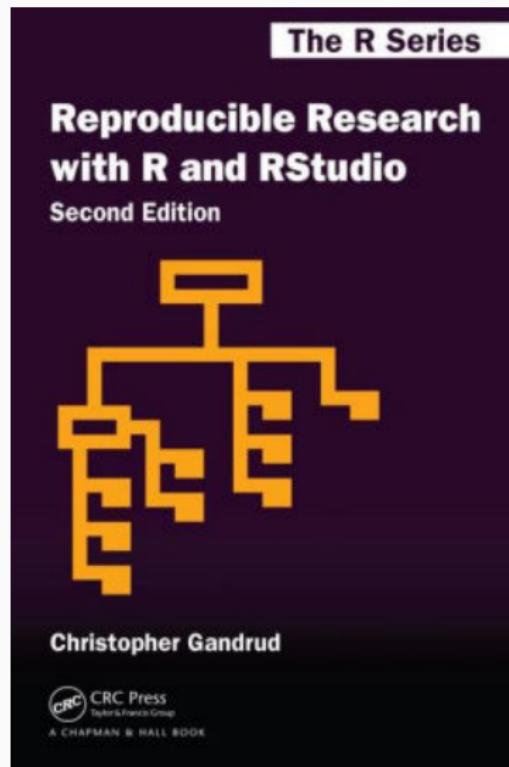
# [Live coding example]

```
irma_accs <- fl_counties %>%  
  full_join(irma_week_accs, by = c("GEOID" ~ "fips")) %>%  
  mutate(fatals = ifelse(is.na(fatals), 0, fatals))  
  
fl_accidents <- fl_accidents %>%  
  st_as_sf(coords = c("longitud", "latitude")) %>%  
  st_set_crs(st_crs(st_read(dsn, layer, ...)))
```

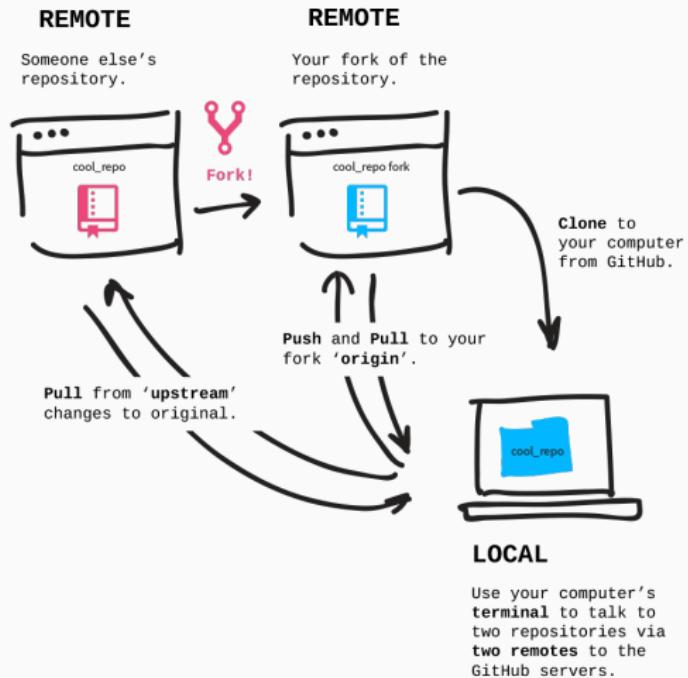
```
irma_track <- st_read("data/al112017_best_track",  
  layer = "al112017_lin") %>%  
  st_transform(crs = st_crs(irma_accs))
```

# Resources



# TRACK

**git and GitHub** for version control



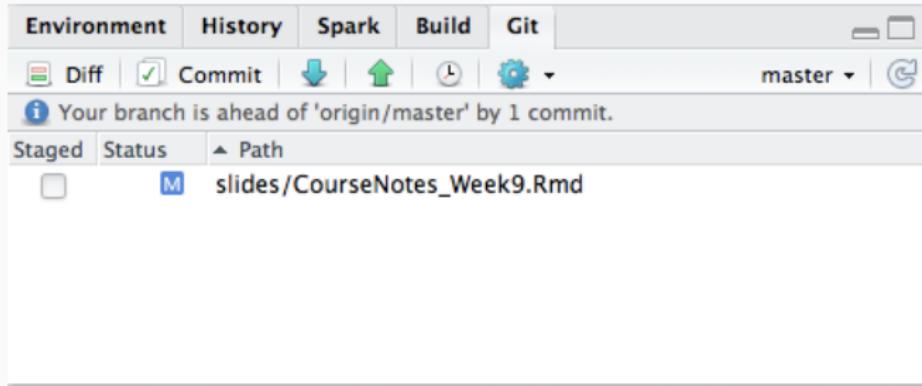
Source: GitHub

```
[Georgianas-MacBook-Pro:columbia_env_health georgianaanderson$ ls -a
.
..
.DS_Store
.Rproj.user
.git
.gitignore
.nojekyll
01-organize.Rmd
02-track.Rmd
03-package.Rmd
04-collect.Rmd
05-process.Rmd
06-summary.Rmd
DESCRIPTION
LICENSE
R
README.md
_bookdown.yml
_bookdown_files
```



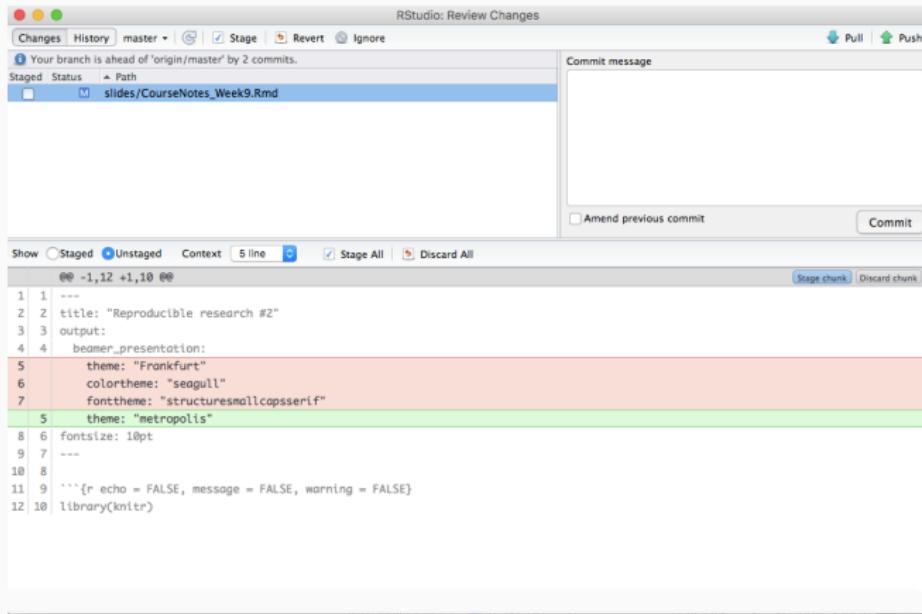
# Git commits in RStudio

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 :



The screenshot shows the RStudio interface for managing git changes. The top navigation bar includes tabs for 'Changes' (which is selected), 'History', and 'master'. Below the tabs, a status message indicates 'Your branch is ahead of 'origin/master' by 2 commits.' The main area displays a file named 'slides/CourseNotes\_Week9.Rmd' in the 'Staged' status. A large text area below shows the code changes:

```
@@ -1,12 +1,10 @@
1: ---
2: 2 title: "Reproducible research #2"
3: 3 output:
4: 4   beamer_presentation:
5:     theme: "Frankfurt"
6:     colortheme: "seagull"
7:     fonttheme: "structuresmallcapsserif"
8: 5   theme: "metropolis"
9: 6   fontsize: 10pt
10: 7   ---
11: 8
12: 9   ```{r echo = FALSE, message = FALSE, warning = FALSE}
13: 10  library(knitr)
```

At the bottom of the code view, there are buttons for 'Stage chunk' and 'Discard chunk'. To the right of the code view is a 'Commit message' field with a 'Commit' button at the bottom.

# Commit messages

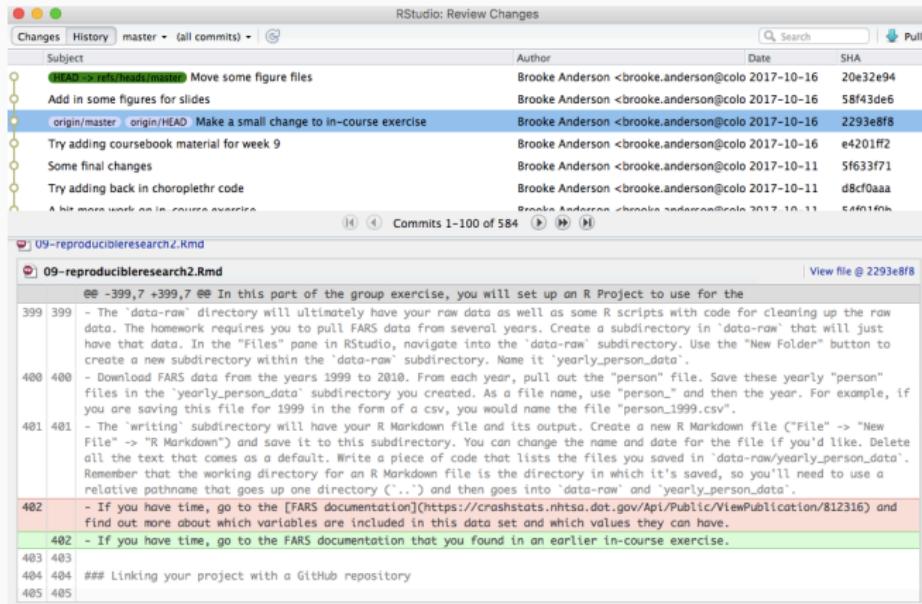
	COMMENT	DATE
O	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
O	ENABLED CONFIG FILE PARSING	9 HOURS AGO
O	MISC BUGFIXES	5 HOURS AGO
O	CODE ADDITIONS/EDITS	4 HOURS AGO
O	MORE CODE	4 HOURS AGO
O	HERE HAVE CODE	4 HOURS AGO
O	AAAAAAA	3 HOURS AGO
O	ADKFJSLKDFJSDFKLJF	3 HOURS AGO
O	MY HANDS ARE TYPING WORDS	2 HOURS AGO
O	HAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Source: xkcd

# Browsing history

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



The screenshot shows the RStudio interface with the "Review Changes" window open. The "History" tab is selected at the top left. The main pane displays a list of commits from a repository named "09-reproducibleresearch2.Rmd". The commits are listed in chronological order, with the most recent at the top. Each commit includes the subject, author, date, and SHA. The commit "origin/master > origin/HEAD Make a small change to in-course exercise" is highlighted with a blue selection bar. Below the commit list, there is a preview of the file "09-reproduciblerearch2.Rmd". The preview shows several lines of R code and some explanatory text. The code includes variable definitions like `@ 09-reproduciblerearch2.Rmd` and `## Linking your project with a GitHub repository`. The preview area has a light gray background with some colored highlights (blue, green, red) around specific lines of code.

Subject	Author	Date	SHA
(HEAD => refs/heads/master) Move some figure files	Brooke Anderson <brooke.anderson@colo>	2017-10-16	20e32e94
Add in some figures for slides	Brooke Anderson <brooke.anderson@colo>	2017-10-16	58f43de6
origin/master > origin/HEAD Make a small change to in-course exercise	Brooke Anderson <brooke.anderson@colo>	2017-10-16	2293e8f8
Try adding coursebook material for week 9	Brooke Anderson <brooke.anderson@colo>	2017-10-16	e4201ff2
Some final changes	Brooke Anderson <brooke.anderson@colo>	2017-10-11	f633cf71
Try adding back in choropleth code	Brooke Anderson <brooke.anderson@colo>	2017-10-11	d8cf0aaa
A bit more work on in-course exercises	Brooke Anderson <brooke.anderson@colo>	2017-10-11	EAD010b

Commits 1-100 of 584

09-reproduciblerearch2.Rmd

View file @ 2293e8f8

@@ -399,7 +399,7 @@ In this part of the group exercise, you will set up an R Project to use for the

399 399 - The "data-row" directory will ultimately have your raw data as well as some R scripts with code for cleaning up the raw data. The homework requires you to pull FARS data from several years. Create a subdirectory in 'data-row' that will just have that data. In the "Files" pane in RStudio, navigate into the 'data-row' subdirectory. Use the "New Folder" button to create a new subdirectory within the 'data-row' subdirectory. Name it 'yearly\_person\_data'.

400 400 - Download FARS data from the years 1999 to 2010. From each year, pull out the "person" file. Save these yearly "person" files in the 'yearly\_person\_data' subdirectory you created. As a file name, use "person\_." and then the year. For example, if you are saving this file for 1999 in the form of a csv, you would name the file "person\_1999.csv".

401 401 - The 'writing' subdirectory will have your R Markdown file and its output. Create a new R Markdown file ("File" -> "New File" -> "R Markdown") and save it to this subdirectory. You can change the name and date for the file if you'd like. Delete all the text that comes as a default. Write a piece of code that lists the files you saved in 'data-row/yearly\_person\_data'. Remember that the working directory for an R Markdown file is the directory in which it's saved, so you'll need to use a relative pathname that goes up one directory ('..') and then goes into 'data-row' and 'yearly\_person\_data'.

402 402 - If you have time, go to the [FARS documentation](https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812316) and find out more about which variables are included in this data set and which values they can have.

402 403 - If you have time, go to the FARS documentation that you found in an earlier in-course exercise.

403 404 ## Linking your project with a GitHub repository

405 405

```
irma_week_accs <- fl_accidents %>%  
  group_by(fips) %>%  
  summarize(fatals = sum(fatals))
```

# [Live coding example]

```
irma_accs <- fl_counties %>%  
  full_join(irma_week_accs, by = c("GEOID" %>% 'fips')) %>%  
  mutate(fatals = ifelse(is.na(fatals), 0, fatals))  
  
fl_accidents <- fl_accidents %>%  
  st_as_sf(coords = c("longitud", "latitude")) %>%  
  st_set_crs(st_crs(st_read(dsn, layer, ...)))
```

```
irma_track <- st_read("data/al112017_best_track",  
  layer = "al112017_lin") %>%  
  st_transform(crs = st_crs(irma_accs))
```

# Using GitHub to collaborate



<https://github.com/ropenscilabs/miner>

# Hosting content with GitHub Pages



## R for Environmental Health Research

*Workshop for Climate and Health students at Columbia Mailman School of Public Health*

*Brooke Anderson*

*April 9, 2019*

### Chapter 1 Prerequisites

#### 1.0.1 Overview

BASED ON REQUESTS FROM some of the students for this workshop, I've focused here on a few topics relevant to environmental health research: organizing projects and tracking them with version control, creating your own packages, and collecting and processing large datasets relevant to environmental health research. You can download the slides from the workshop by [clicking here](#).

# 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

Folders	Documents	Developer
weathermetrics ►	cran-comments.md NEWS.md README.md	data.R heat_index.R moisture_conversions.R rainmeasure_conversion.R temperature_conversions.R weathermetrics.R wind_conversions.R
PDF Documents		
weathermetrics.pdf		
Other		
weathermetrics_1.2.0.tar.gz		
weathermetrics_1.2.2.tar.gz		

# R package template

The screenshot shows the RStudio interface with a project titled "convertr" open. The left pane displays the "hello.R" script, which contains a series of R comments and code snippets demonstrating keyboard shortcuts for package authoring. The right pane shows the "Files" view, listing the contents of the package directory: ".Rbuildignore", "convertr.Rproj", "DESCRIPTION", "man", "NAMESPACE", and "R". The "Console" tab at the bottom shows the R environment setup and basic help documentation.

```
~/Documents/r_workshops/convertr - RStudio
hello.R
1 # You can learn more about package authoring with RStudio at:
2 #
3 #   http://r-pkgs.had.co.nz/
4 #
5 # Some useful keyboard shortcuts for package authoring:
6 #
7 # Build and Reload Package: 'Cmd + Shift + B'
8 # Check Package:           'Cmd + Shift + E'
9 # Test Package:            'Cmd + Shift + T'
10
11
12 hello <- function() {
13   print("Hello, world!")
14 }
15
16
17
18
19
1:1 (Top Level) ▾
```

Environment is empty

Files Plots Packages Help Viewer

Name	Size
.Rbuildignore	28 B
convertr.Rproj	356 B
DESCRIPTION	369 B
man	
NAMESPACE	31 B
R	

# 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

```
irma_week_accs <- fl_accidents %>%  
  group_by(fips) %>%  
  summarize(fatals = sum(fatals))
```

# [Live coding example]

```
irma_accs <- fl_counties %>%  
  full_join(irma_week_accs, by = c("GEOID" ~ "fips")) %>%  
  mutate(fatals = ifelse(is.na(fatals), 0, fatals))  
  
fl_accidents <- fl_accidents %>%  
  st_as_sf(coords = c("longitud", "latitude")) %>%  
  st_set_crs(st_crs(st_read(dsn, layer, ...)))
```

```
irma_track <- st_read("data/al112017_best_track",  
  layer = "al112017_lin") %>%  
  st_transform(crs = st_crs(irma_accs))
```

# Resources



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

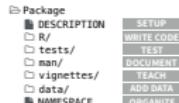
# Resources

## Package Development: : CHEAT SHEET

### Package Structure

A package is a convention for organizing files into directories.

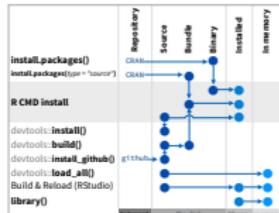
This sheet shows how to work with the 7 most common parts of an R package:



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.



### 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()`
- Add a package to the Imports or Suggests field

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

### Write Code (tests/)

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)

Package: mypackage  
Title: Title of Package  
Version: 0.1.0  
Author/R: person("Hadley", "Wickham", email = "hadley.wickham@gmail.com", role = "aut")  
Description: What this package does (one paragraph)  
Depends: R (>= 3.1.0)  
License: MIT  
LazyData: true  
Imports:  
Suggests:  
Linker: (>= 0.4.0),  
ggvis: (>= 0.2),  
Suggests:  
knitr: (>= 1.1.0)



### 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. Edit your code.
2. Load your code with one of `devtools::load_all()`  
Re-load 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-pkg.had.co.nz/r.html#style](http://r-pkg.had.co.nz/r.html#style)
- Click on a function and press F2 to open its definition
- Search for a function with Ctrl + F



### Example Test

```
context("Arithmetic")
test_that("math works", {
  expect_equal(1 + 1, 2)
  expect_equal(1 + 2, 3)
  expect_equal(1 + 3, 4)
})
```

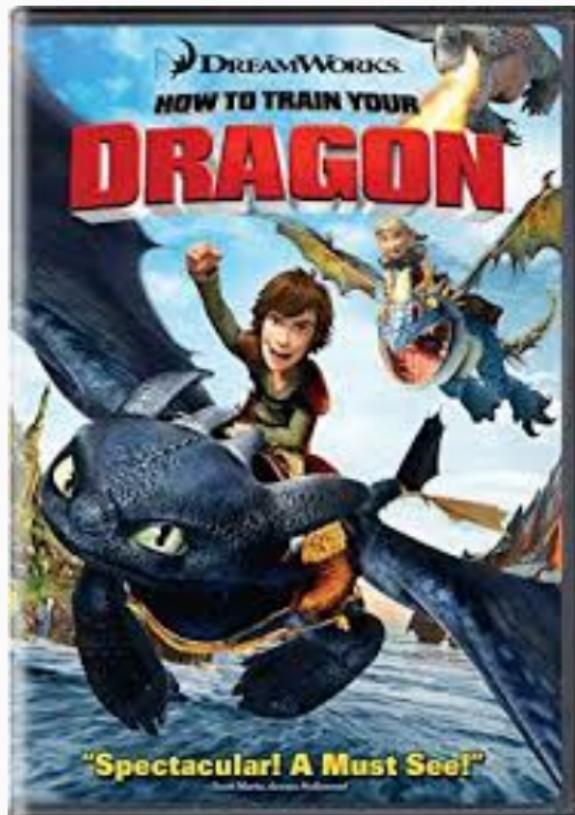
### 3. Repeat until all tests pass

Expected statement	Tests
<code>expect_equal()</code>	is equal within small numerical tolerance?
<code>expect_identical()</code>	is exactly equal?
<code>expect_match()</code>	matches specified string or regular expression?
<code>expect_subset()</code>	prints specified output?
<code>expect_message()</code>	displays specified message?
<code>expect_warning()</code>	displays specified warning?
<code>expect_error()</code>	throws specified error?
<code>expect_no_error()</code>	outputs inherits from certain class?
<code>expect_file()</code>	returns FALSE?
<code>expect_true()</code>	returns TRUE?

RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio - info@rstudio.com • 844-440-1232 - rstudio.com • Learn more at <http://r-pkg.had.co.nz/> • devtools 1.5.1 • Updated: 2015-02-01

<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
```

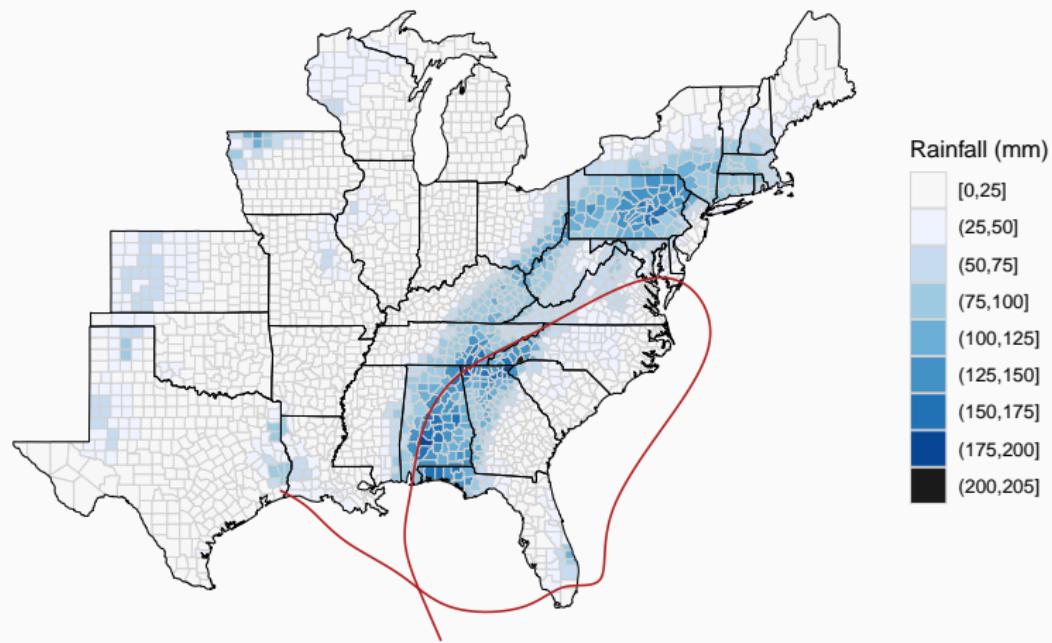
# Data packages

```
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
```

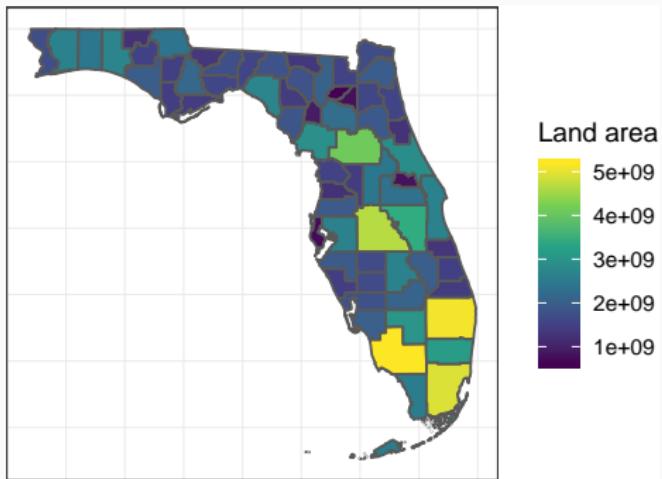
# Data packages

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



# Open Data APIs

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



```
irma_week_accs <- fl_accidents %>%  
  group_by(fips) %>%  
  summarize(fatals = sum(fatals))
```

# [Live coding example]

```
irma_accs <- fl_counties %>%  
  full_join(irma_week_accs, by = c("GEOID" ~ "fips")) %>%  
  mutate(fatals = ifelse(is.na(fatals), 0, fatals))  
  
fl_accidents <- fl_accidents %>%  
  st_as_sf(coords = c("longitud", "latitude")) %>%  
  st_set_crs(st_crs(st_read(dsn, layer, ...)))  
  
irma_track <- st_read("data/al112017_best_track",  
  layer = "al112017_lin") %>%  
  st_transform(crs = st_crs(irma_accs))
```

# Resources

# #rstats



**Dirk Eddelbuettel** @eddelbuettel · 27 Jan 2017

Big congratulations to @gbwanderson whose new package 'hurricaneexposure' just became package 10,000 on CRAN !!

**CRAN Package Updates** @CRANberriesFeed

9999 packages on CRAN right now, so imagine dozens of R nerds hanging in suspense waiting for the package to make it 10k ...

2

35

93



## Resources

ROpenSci

# Homework!!

<https://bit.ly/2WQV6XT>