

**Meet the toolkit:
version control and collaboration**

IS 407

JooYoung Seo

Learning goals

By the end of the course, you will be able to...

- gain insight from data
- gain insight from data, **reproducibly**
- gain insight from data, reproducibly, **using modern programming tools and techniques**
- gain insight from data, reproducibly **and collaboratively**, using modern programming tools and techniques
- gain insight from data, reproducibly **(with literate programming and version control)** and collaboratively, using modern programming tools and techniques

Reproducible data analysis

Reproducibility checklist

What does it mean for a data analysis to be "reproducible"?

Near-term goals:

- Are the tables and figures reproducible from the code and data?
- Does the code actually do what you think it does?
- In addition to what was done, is it clear *why* it was done?

Long-term goals:

- Can the code be used for other data?
- Can you extend the code to do other things?

Toolkit for reproducibility

- Scriptability → R
- Literate programming (code, narrative, output in one place) → R Markdown
- Version control → Git / GitHub

Course toolkit

Doing data science

- Programming:
 - R
 - RStudio
 - tidyverse
 - R Markdown
- Version control and collaboration:
 - Git
 - GitHub

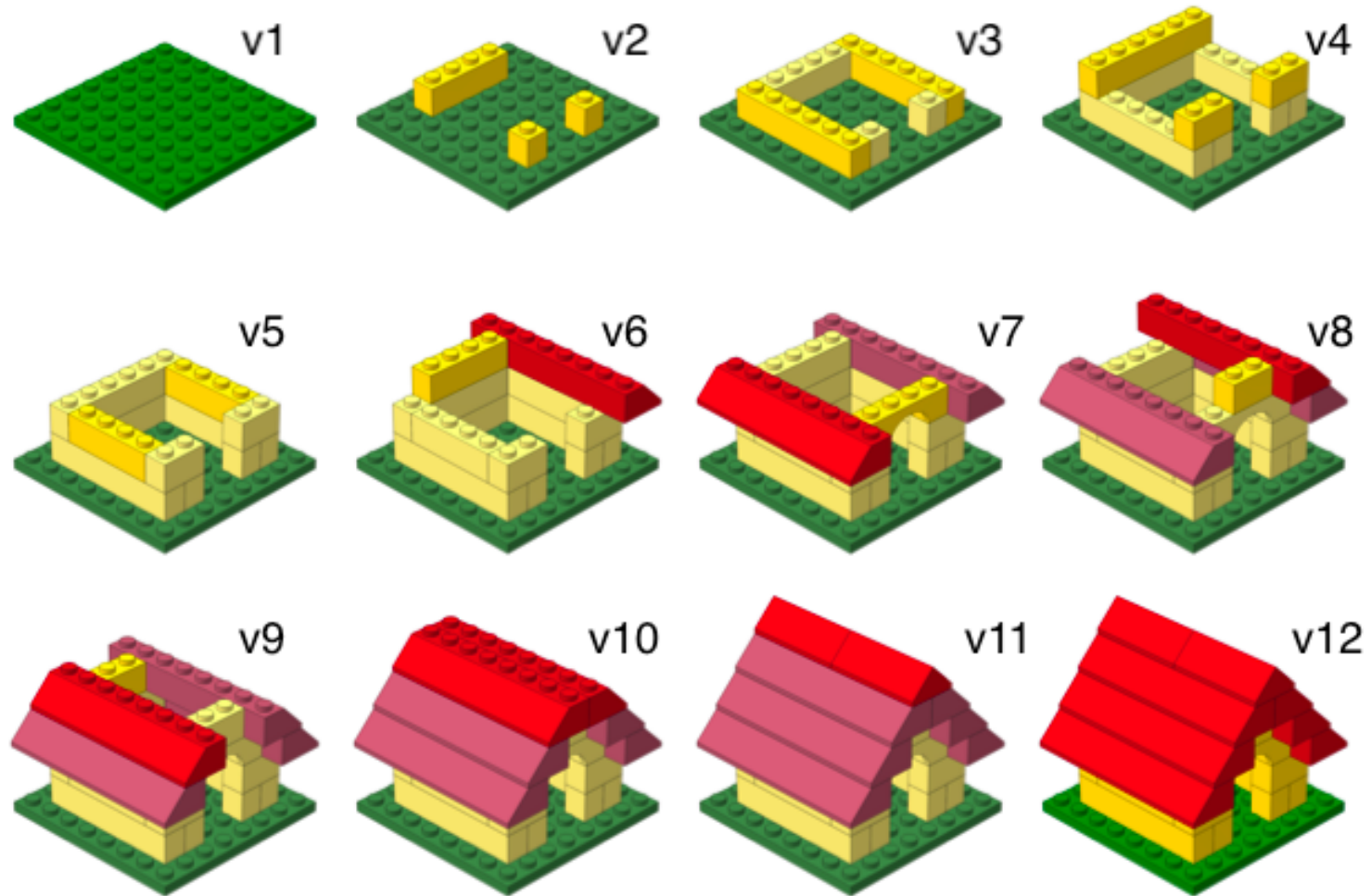
Git and GitHub

Git and GitHub

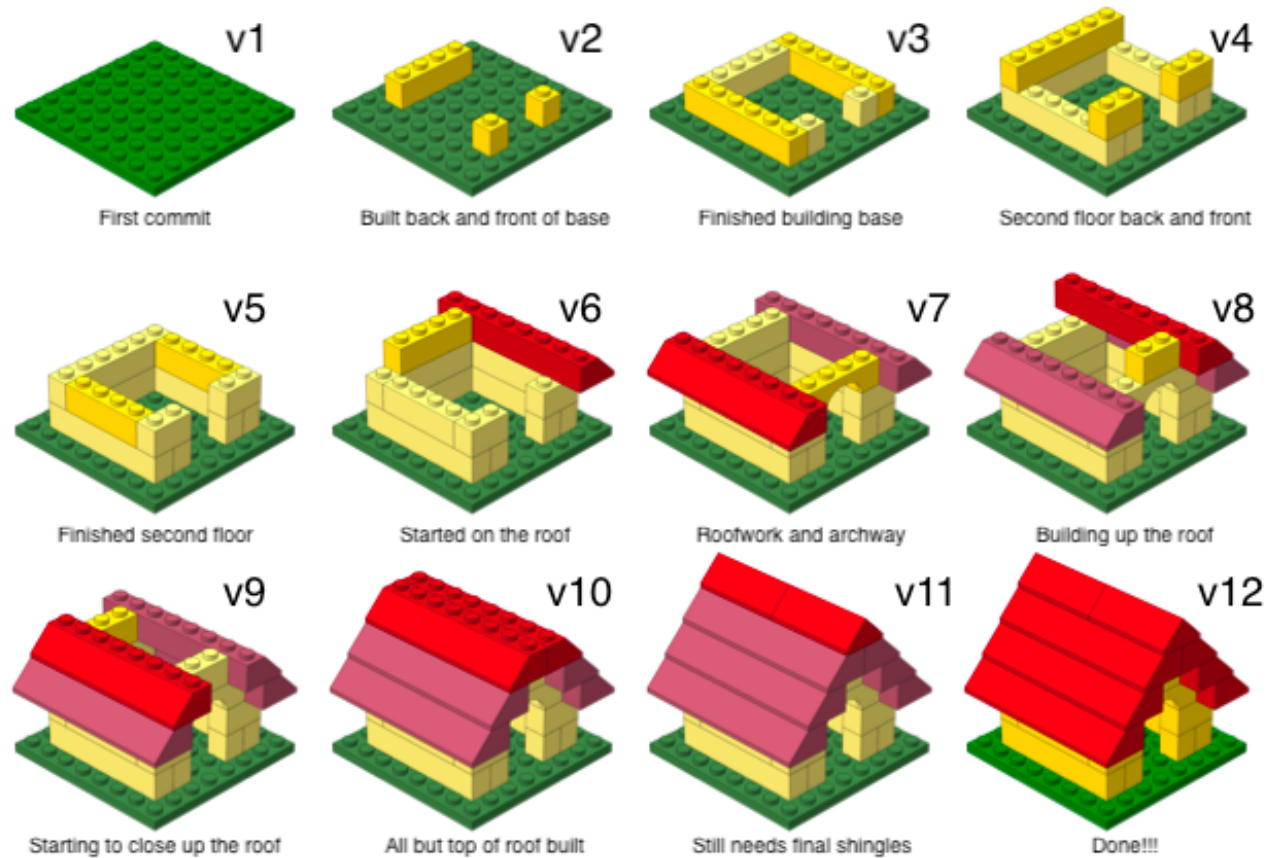


- Git is a version control system -- like “Track Changes” features from Microsoft Word, on steroids
- It's not the only version control system, but it's a very popular one
- GitHub is the home for your Git-based projects on the internet -- like DropBox but much, much better
- We will use GitHub as a platform for web hosting and collaboration (and as our course management system!)

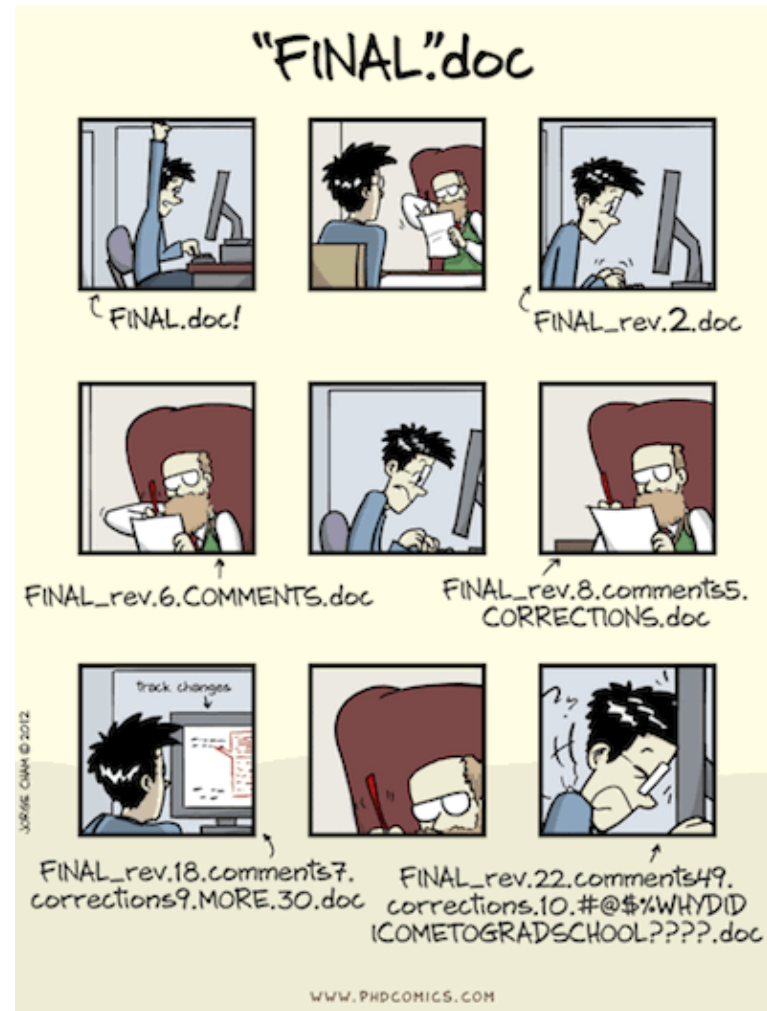
Versioning



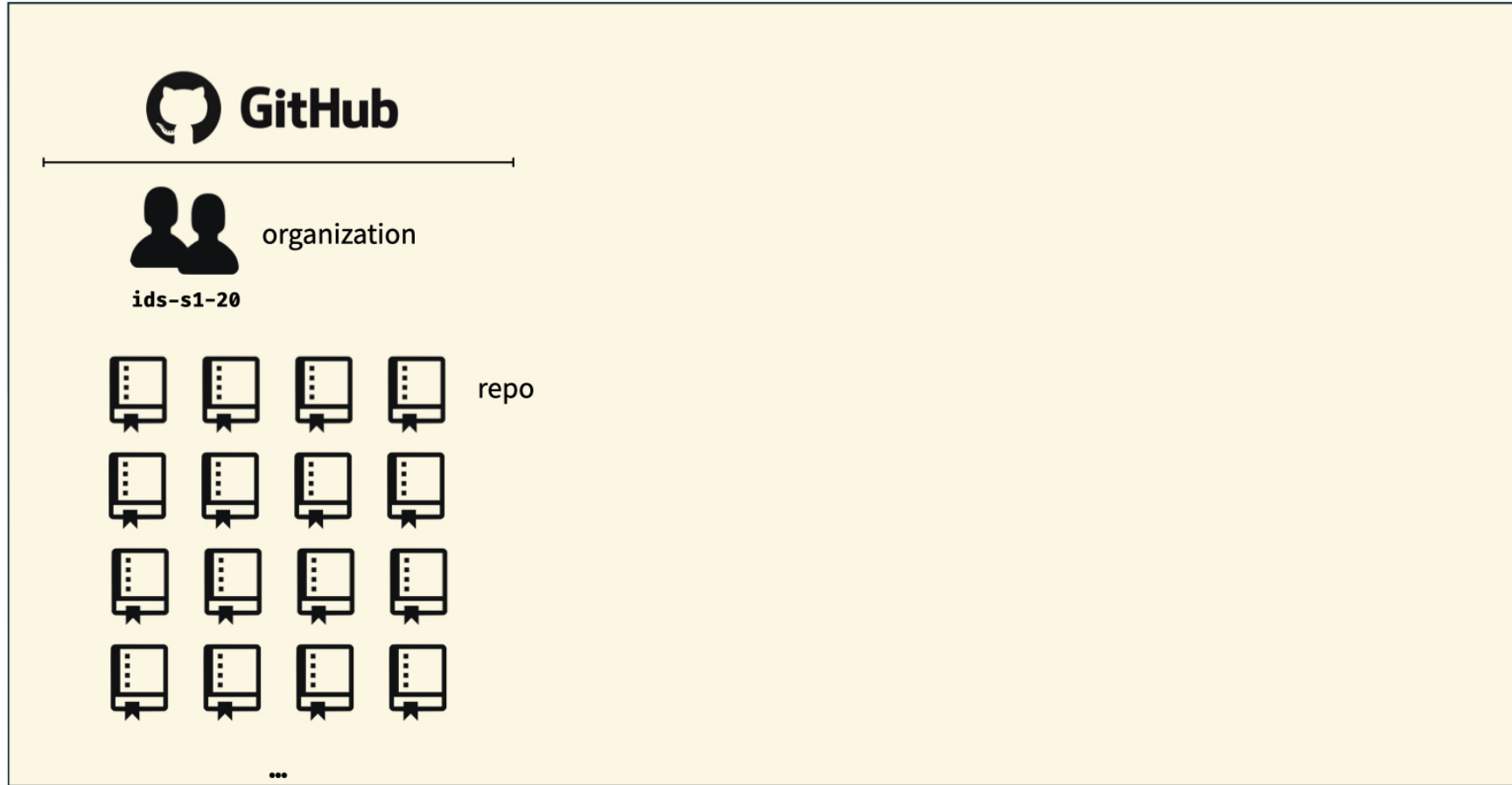
Versioning
with human readable messages



Why do we need version control?



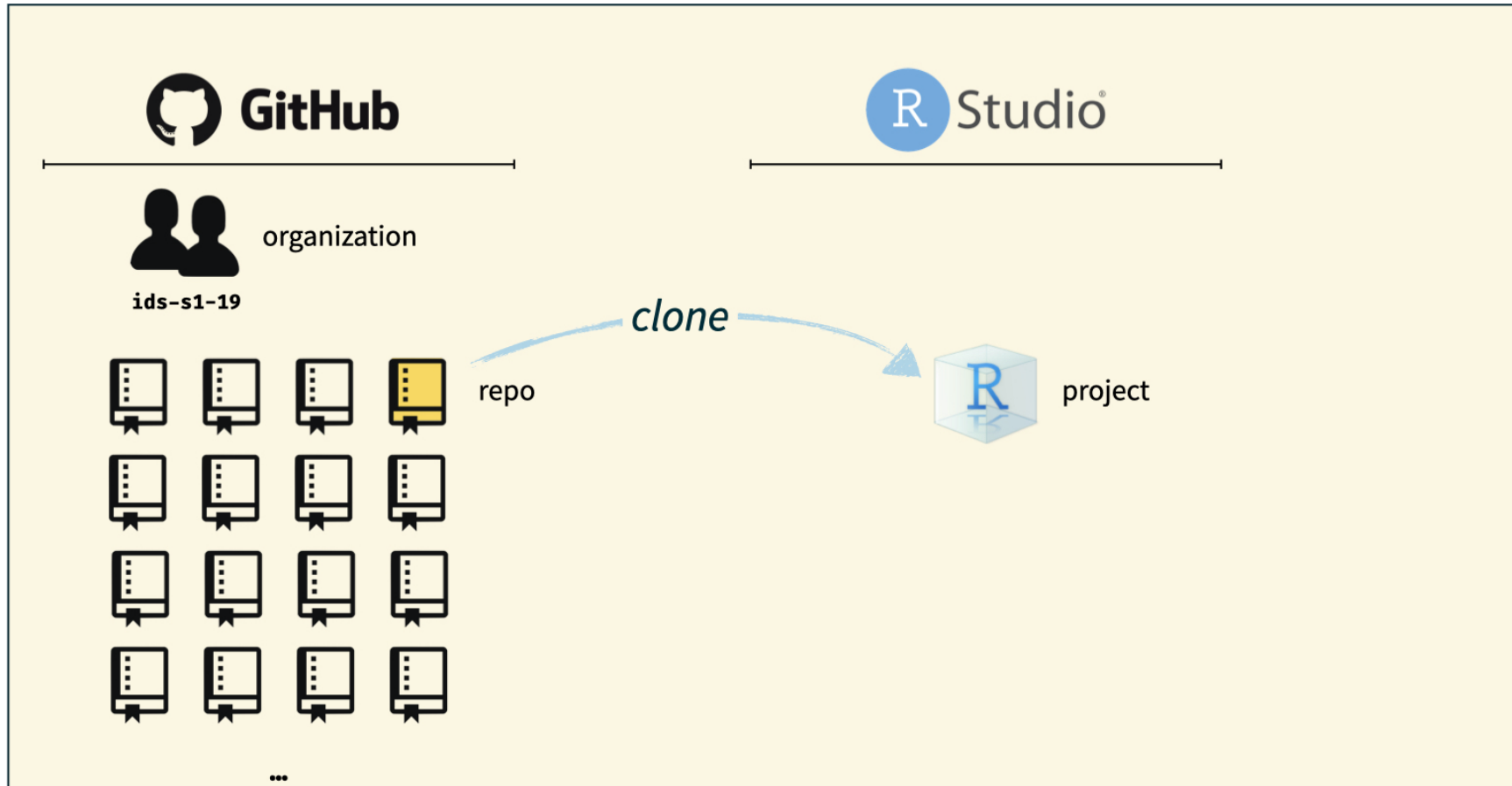
How will we use Git and GitHub?



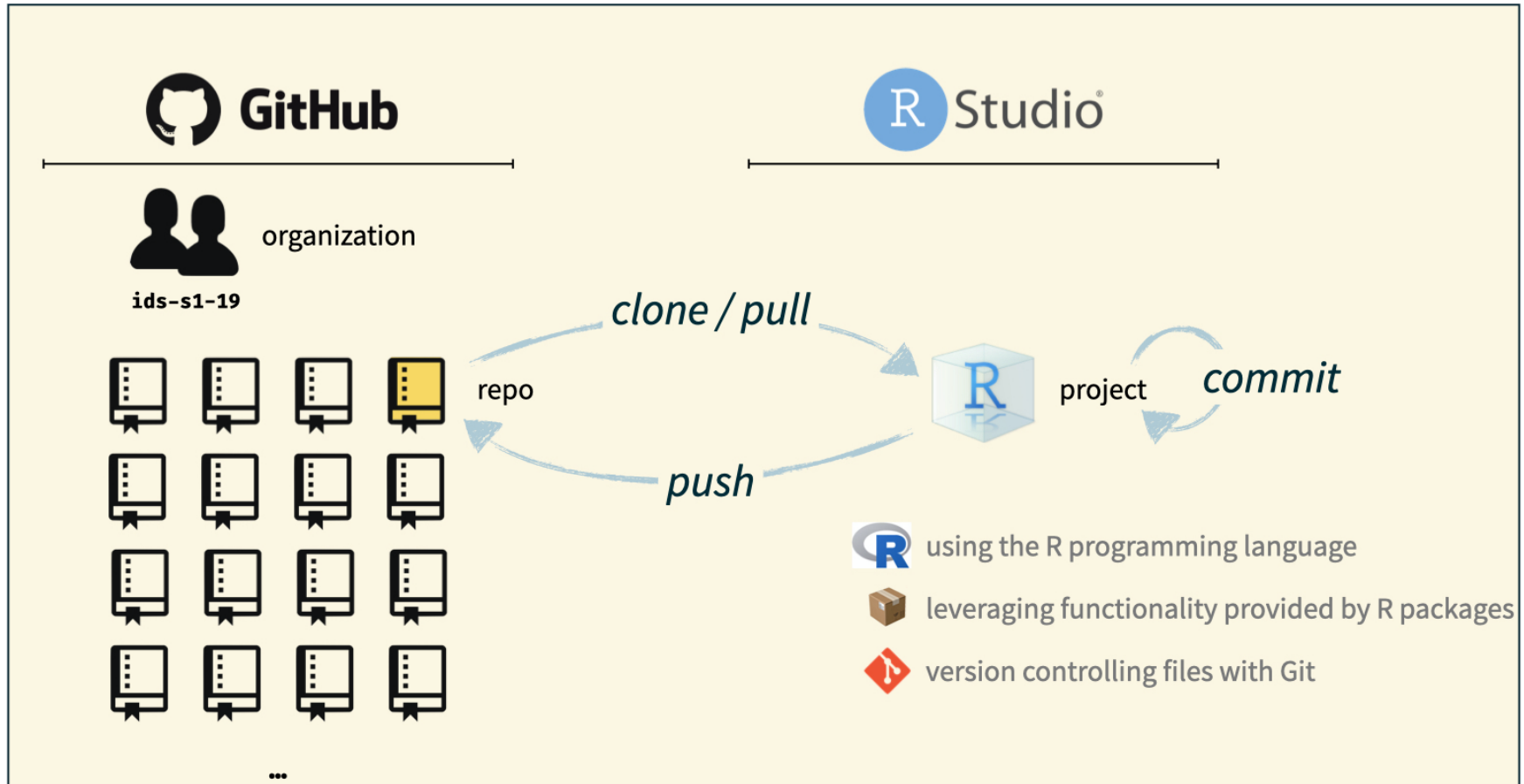
How will we use Git and GitHub?



How will we use Git and GitHub?



How will we use Git and GitHub?



Git and GitHub tips

- There are millions of git commands -- ok, that's an exaggeration, but there are a lot of them -- and very few people know them all. 99% of the time you will use git to add, commit, push, and pull.
- We will be doing Git things and interfacing with GitHub through RStudio, but if you google for help you might come across methods for doing these things in the command line -- skip that and move on to the next resource unless you feel comfortable trying it out.
- There is a great resource for working with git and R: happygitwithr.com. Some of the content in there is beyond the scope of this course, but it's a good place to look for help.

Tour: Git and GitHub

- Create a GitHub account
- Verify your GitHub email
- Adjust your GitHub settings for a more pleasant GitHub experience

Next week...

Work with R, RStudio, Git, and GitHub together!+

⁺Just like a real data scientist!

Course toolkit

Doing data science

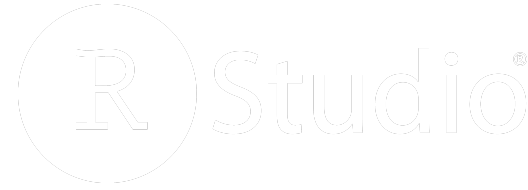
- Programming:
 - R
 - RStudio
 - tidyverse
 - R Markdown
- Version control and collaboration:
 - Git
 - GitHub

R and RStudio

R and RStudio



- R is an open-source statistical **programming language**
- R is also an environment for statistical computing and graphics
- It's easily extensible with *packages*



- RStudio is a convenient interface for R called an **IDE** (integrated development environment), e.g. *"I write R code in the RStudio IDE"*
- RStudio is not a requirement for programming with R, but it's very commonly used by R programmers and data scientists

R packages

- **Packages** are the fundamental units of reproducible R code. They include reusable R functions, the documentation that describes how to use them, and sample data¹

There are over 16,000 R packages available on **CRAN** (the Comprehensive R Archive Network)²

- We're going to work with a small (but important) subset of these!

¹ Wickham and Bryan, R Packages.

² CRAN contributed packages.

Tour: R and RStudio

data viewer

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
1	Adelie	Torgersen	39.1	18.7	181	
2	Adelie	Torgersen	39.5	17.4	186	
3	Adelie	Torgersen	40.3	18.0	195	
4	Adelie	Torgersen	NA	NA	NA	
5	Adelie	Torgersen	36.7	19.3	193	
6	Adelie	Torgersen	39.3	20.6	190	
7	Adelie	Torgersen	38.9	17.8	181	
8	Adelie	Torgersen	39.2	19.6	195	
9	Adelie	Torgersen	34.1	18.1	193	
10	Adelie	Torgersen	42.0	20.2	190	
11	Adelie	Torgersen	37.8	17.1	186	

Showing 1 to 11 of 344 entries, 7 total columns

environment

help

Arithmetic Mean

Description

Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)
```

Default S3 method:
mean(x, trim = 0, na.rm = FALSE, ...)

Arguments

x: An R object. Currently there are methods for numeric/logical vectors and

Examples

```
x <- c(0:10, 50)
xm <- mean(x)
c(xm, mean(x, trim = 0.10))
```

[Package base version 4.0.2 [Index](#)]

load package

view data

get help

Object assignment

access variable

use function

```
> 2 + 2
[1] 4
> x <- 2
> x * 3
[1] 6
> library(palmerpenguins)
> View(penguins)
> penguins$flipper_length_mm
[1] 181 186 195 NA 193 190 181 195 193 190 186 180 182 191
[337] 206 189 195 207 202 193 210 198
> mean(penguins$flipper_length_mm)
[1] NA
> ?mean
> mean(penguins$flipper_length_mm, na.rm = TRUE)
[1] 200.9152
```

A short list (for now) of R essentials

- Functions are (most often) verbs, followed by what they will be applied to in parentheses:

```
verb(object)
eat(food)
do_that(to_this, to_that, with_those)
```

- Packages are installed with the `install.packages` function and loaded with the `library` function, once per session:

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

R essentials (continued)

- Columns (variables) in data frames are accessed with `$`:

```
dataframe$var_name
```

- Object documentation can be accessed with `?`

```
?mean
```




- The **tidyverse** is an opinionated collection of R packages designed for data science
- All packages share an underlying philosophy and a common grammar

rmarkdown.rstudio.com

- **rmarkdown** and the various packages that support it enable R users to write their code and prose in reproducible computational documents
- We will generally refer to R Markdown documents (with `.Rmd` extension), e.g. *"Do this in your R Markdown document"* and rarely discuss loading the rmarkdown package



R Markdown

R Markdown

- Fully reproducible reports -- each time you knit the analysis is ran from the beginning
- Simple markdown syntax for text
- Code goes in chunks, defined by three backticks, narrative goes outside of chunks

Tour: R Markdown

The image shows the RStudio interface with an R Markdown document open. The document is titled "Bechdel.Rmd" and contains the following content:

```
1 ---
2 title: "Bechdel"
3 author: "Mine Çetinkaya-Rundel"
4 output:
5   html_document:
6     fig_height: 4
7     fig_width: 9
8 ---
9
10 In this mini analysis we work with the data used
11 in the FiveThirtyEight story titled ["The
12 Dollar-And-Cents Case Against Hollywood's
13 Exclusion of Women"](https://fivethirtyeight.com/f
14 eatures/the-dollar-and-cents-case-against-hollywo
15 ds-exclusion-of-women/). Your task is to fill in
16 the blanks denoted by `___`.
17
18 ## Data and packages
19
20 We start with loading the packages we'll use.
21
22 ```{r load-packages, message=FALSE}
23 library(fivethirtyeight)
24 library(tidyverse)
25 ```
```

Handwritten annotations on the left side of the R Markdown editor:

- knit**: A yellow arrow points to the Knit button in the top toolbar.
- link**: A green arrow points to the link in the text of the first code chunk.
- code chunk**: A pink bracket highlights the code chunk starting at line 16.
- yaml**: A red bracket highlights the YAML header (lines 1-8).

The rendered HTML output on the right shows the following content:

Bechdel

Mine Çetinkaya-Rundel

In this mini analysis we work with the data used in the FiveThirtyEight story titled [“The Dollar-And-Cents Case Against Hollywood’s Exclusion of Women”](https://fivethirtyeight.com/features/the-dollar-and-cents-case-against-hollywoods-exclusion-of-women/). Your task is to fill in the blanks denoted by `___`.

Data and packages

We start with loading the packages we’ll use.

```
library(fivethirtyeight)
library(tidyverse)
```

The dataset contains information on 1794 movies released between 1970 and 2013. However we’ll focus our analysis on movies released between 1990 and 2013.

```
bechdel190_13 <- bechdel %>%
  filter(between(year, 1990, 2013))
```

There are `___` such movies.

The financial variables we’ll focus on are the following:

- `budget_2013`: Budget in 2013 inflation adjusted dollars
- `domgross_2013`: Domestic gross (US) in 2013 inflation adjusted dollars
- `intgross_2013`: Total International (i.e., worldwide) gross in 2013 inflation adjusted dollars

R Markdown help

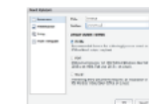
R Markdown Cheat Sheet
Help -> Cheatsheets

R Markdown : : CHEAT SHEET

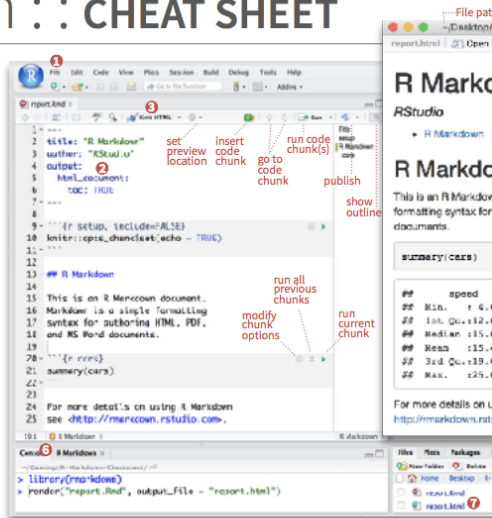
What is R Markdown?

- .Rmd files** - An R Markdown (.Rmd) file is a record of your research. It contains the code that a scientist needs to reproduce your work along with the narration that a reader needs to understand your work.
- Reproducible Research** - At the click of a button, or the type of a command, you can rerun the code in an R Markdown file to reproduce your work and export the results as a finished report.
- Dynamic Documents** - You can choose to export the finished report in a variety of formats, including html, pdf, MS Word, or RTF documents; html or pdf based slides, Notebooks, and more.

Workflow



- 1 Open a new .Rmd file at File > New File > R Markdown. Use the wizard that opens to pre-populate the file with a template
- 2 Write document by editing template
- 3 Knit document to create report; use knit button or `render()` to knit
- 4 Preview Output in IDE window
- 5 Publish (optional) to web server
- 6 Examine build log in R Markdown console
- 7 Use output file that is saved along side .Rmd



render

Use `rmarkdown::render()` to render/knit at cmd line. Important args:

Input - file to render	output_options - List of render options (as in YAML)	output_file output_dir	params - list of params to use
------------------------	--	------------------------	--------------------------------

Embed code with knitr syntax

INLINE CODE

Insert with `<code>`. Results appear as text without code.

Built with `r.getRversion()` Built with 3.2.3

CODE CHUNKS

One or more lines surrounded with ````{r}` and `````. Place chunk options within curly braces, after `r`. Insert with `<code>`.

````{r echo=TRUE}`  
`getRversion()`

#### GLOBAL OPT

Set with knitr: ````{r include=FALSE knitr:opts_chunk=}`

Markdown Quick Reference  
Help -> Markdown Quick Reference

FilesPlotsPackagesHelpViewer

Markdown Quick ReferenceFind in Topic

Markdown Quick Reference

R Markdown is an easy-to-write plain text format for creating dynamic documents and reports. See [Using R Markdown](#) to learn more.

**Emphasis**  
`*italic*` `**bold**`  
`_italic_` `__bold__`

**Headers**  
`# Header 1`  
`## Header 2`  
`### Header 3`

**Lists**  
**Unordered List**  
`* Item 1`  
`* Item 2`  
`+ Item 2a`  
`+ Item 2b`  
**Ordered List**  
`1. Item 1`  
`2. Item 2`  
`3. Item 3`  
`+ Item 3a`  
`+ Item 3b`

**Manual Line Breaks**  
End a line with two or more spaces:  
Roses are red,  
Violets are blue.

**Links**  
Use a plain http address or add a link to a phrase:  
`<a href="http://example.com">example.com</a>`



## How will we use R Markdown?

- Every assignment / report / project / etc. is an R Markdown document
- You'll always have a template R Markdown document to start with
- The amount of scaffolding in the template will decrease over the semester

Your turn: Cumulative deaths from COVID-19

covid.Rmd