Intro to EPI 590R

Why this class?

About this class

Goal: Learn some best practices to make your life in R easier and your research more reproducible

- Quick! Intense!
  - It will require practice afterward, and time to sink in
  - The goal is to set you up for success and give you resources to learn more
- You don't have to use everything you learn here!
  - Some of these tools I use for *every* project, some just occasionally
  - Experiment with what works for you, a little at a time

12:26 PM Sun Nov 3
cran.r-project.org
89%
#> # origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#> # minute <dbl>, time_hour <dttm>
Add new columns with mutate()
Besides selecting sets of existing columns, it's often useful to add new columns that are functions of existing columns. This is the job of mutate():
mutate(flights,
gain = arr_delay - dep_delay,
speed = distance / air_time * 60
)
#> # A tibble: 336,776 x 21
#> year month day dep_time sched_dep_time dep_delay arr_time
#> <int> <int> <int> <int> <int> <dbl> <int>
#> 2013 1 1 517 515 830
#> 2013 1 1 533 529 850
#> 2013 1 1 542 540 923
#> 2013 1 1 544 545 1004
#> # with 336,772 more rows, and 14 more variables: sched_arr_time <int>,
#> # arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
#> # origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#> # hour <dbl>, minute <dbl>, time_hour <dttm>, gain <dbl>, speed <dbl>
dplyr::mutate() is similar to the base transform(), but allows you to refer to
mutate(flights,
gain = arr_delay - dep_delay,
gain_per_hour = gain / (air_time / 60)
)
#> # A tibble: 336,776 x 21
i would rather not behold...
BEHOLD! AN AMAZING FUNCTION!

About this class

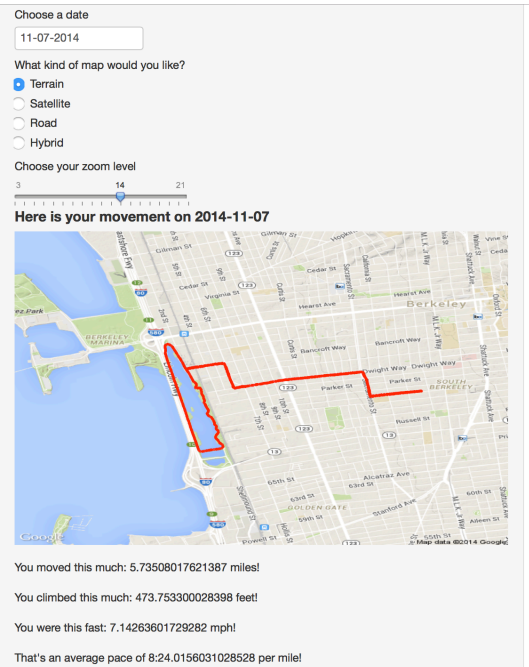
- Everything you need is at http://epi590r.louisahsmith.com
  - Canvas will link you there, but good to bookmark as well
  - The website will be up indefinitely
- General format:
  - Some slides (introduce new content and overview of exercises)
  - I'll demonstrate while you watch
  - Practice on your own/with your classmates

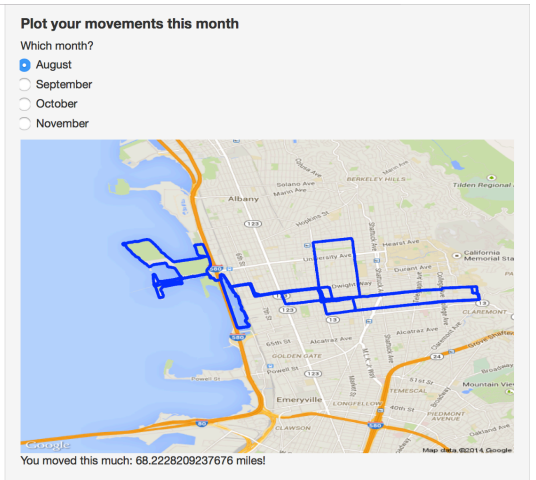
Balance

essential skill
COURSE CONTENT
ANOTHER IMPORTANT THING
ABSOLUTELY CRITICAL
CORE
@allison_horst

About Louisa

- Assistant professor at Northeastern University
  - Department of Health Sciences and the Roux Institute (Portland, Maine)
- Started using R during my master's (so just about 10 years of experience)
  - I learned mostly by doing!
  - Twitter, blogs, RStudio::conf videos, meetups
- Basically everything I do is in R!
- Actual epi research in causal inference, pregnancy, lots of other stuff





About Ollie

Why this class?

Economics | Genes | Orangutans

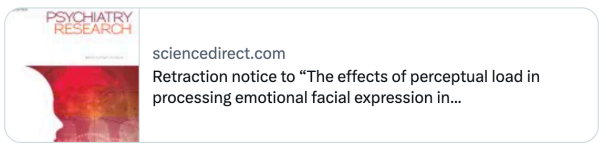
3.2 Spreadsheet coding error

In addition to these deliberate data exclusions by RR, a coding error in the RR working spreadsheet also unintentionally excludes five countries entirely (Australia, Austria, Belgium, Canada and Denmark) from all parts of the analysis.[9] The error appears in the calculations of both mean and median GDP growth with the 1946–2009 sample as well as with the mean and median GDP growth for the sample over the 220-year period 1790–2009. The omitted countries are selected alphabetically. It is clear from the spreadsheet itself that these are random exclusions. RR have since acknowledged this to be the case (RR, 2013A, 2013B, 2013C).

Errors are everywhere

Joana Grave
@joanafqg

A few months ago, I discovered an error in the database of my first paper. And today, the retraction notice is finally out!
Please don't be afraid to talk about your errors and to correct them. It's hard, but errors do happen!



sciencedirect.com
Retraction notice to “The effects of perceptual load in processing emotional facial expression in...

3:51 PM · Jul 11, 2021

24 Retweets **6** Quotes **187** Likes **2** Bookmarks

Julia Strand
@juliafstrand

I recently found a massive error in one of my published papers. The main finding was the result of a programming bug and was, in fact, completely untrue. THREAD with the very short version below, essay with the full description here:

elemental.medium.com
Scientists Make Mistakes. I Made a Big One.
A researcher learns the right thing to do when the wrong thing happens

2:16 PM · Mar 24, 2020

Leigh Senderowicz
@LSenderowicz

Hi all, today I'm writing a post that no researcher ever wants to write:

We've discovered a coding error in the analysis of an article that's already been published.

A short 📘

onlinelibrary.wiley.com
Supply-Side Versus Demand-Side Unmet Need: Implicati...
Despite its central importance to global family planning, the “unmet need for contraception” metric is frequently ...

1:20 PM · Aug 2, 2023 · **62.7K** Views

No one and no field is immune from errors in data analysis. Our goal is to make them as unlikely as possible (and report them when we find them!)

Not only improves science but also your life!

- It’s really boring to copy lots of numbers into a table
  - And then change a tiny thing in the analysis and do it all over again
- It’s really frustrating to lose work when your computer crashes, or completely change an analysis before your advisor forgets what they told you last time and has you change it back
- It’s fun when things just work! And you get more time for the fun parts of epidemiology

BEEP BEEP BEEP!!
oh just add SALT!?!!
RECIPE
@allison_horst

bless this
workflow
TOOLS
CODE
REPORT
FILES
@allison_horst

Questions?

Exercises: Connecting to GitHub

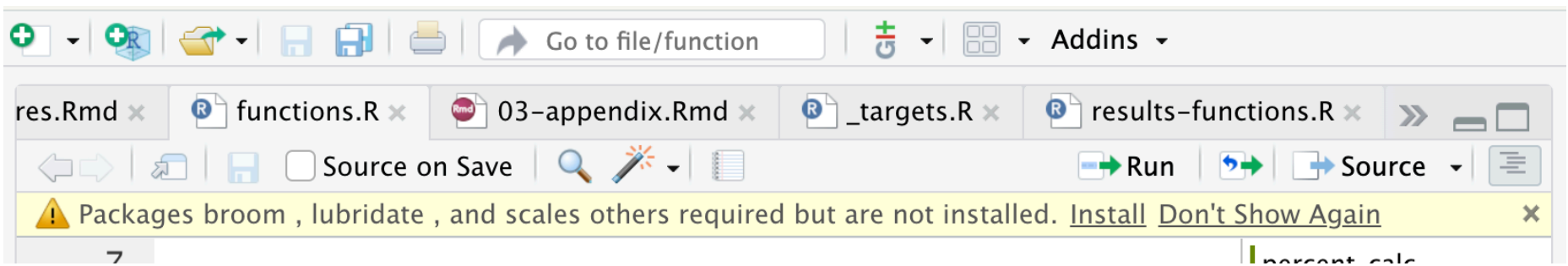
1. Install the `{usethis}` package:

```
install.packages("usethis")
```

Installing packages

If you just updated R to a new “major” version, you will need to reinstall packages

- I tend to do this as I need them rather than try to reinstall them all at once
  - RStudio tries to help!



Possible errors

Spelling the package or function's name wrong, or not installing or loading the package

```
> install.packages("blah")
Warning in install.packages :
  package ‘blah’ is not available for this version of R

A version of this package for your version of R might be available elsewhere,
see the ideas at
https://cran.r-project.org/doc/manuals/r-patched/R-admin.html#Installing-packages
> library(blah)
Error in library(blah) : there is no package called ‘blah’
> blah::blah()
Error in loadNamespace(x) : there is no package called ‘blah’
> blah()
Error in blah() : could not find function "blah"
```

Using packages: `library` vs. `::`

If you are writing a script you will save, and will use several functions from this package

```
1 library(usethis)
2 use_git_config(user.na
3                  user.em
```

If you are just running some quick code in the console or only need to use the package a few times in a script

```
1 usethis::use_git_confi
```

I try to only run `library(package)` from a script (not the console) so that there's a "record" of me loading the package, or else I might accidentally write code that doesn't work later

Set up git

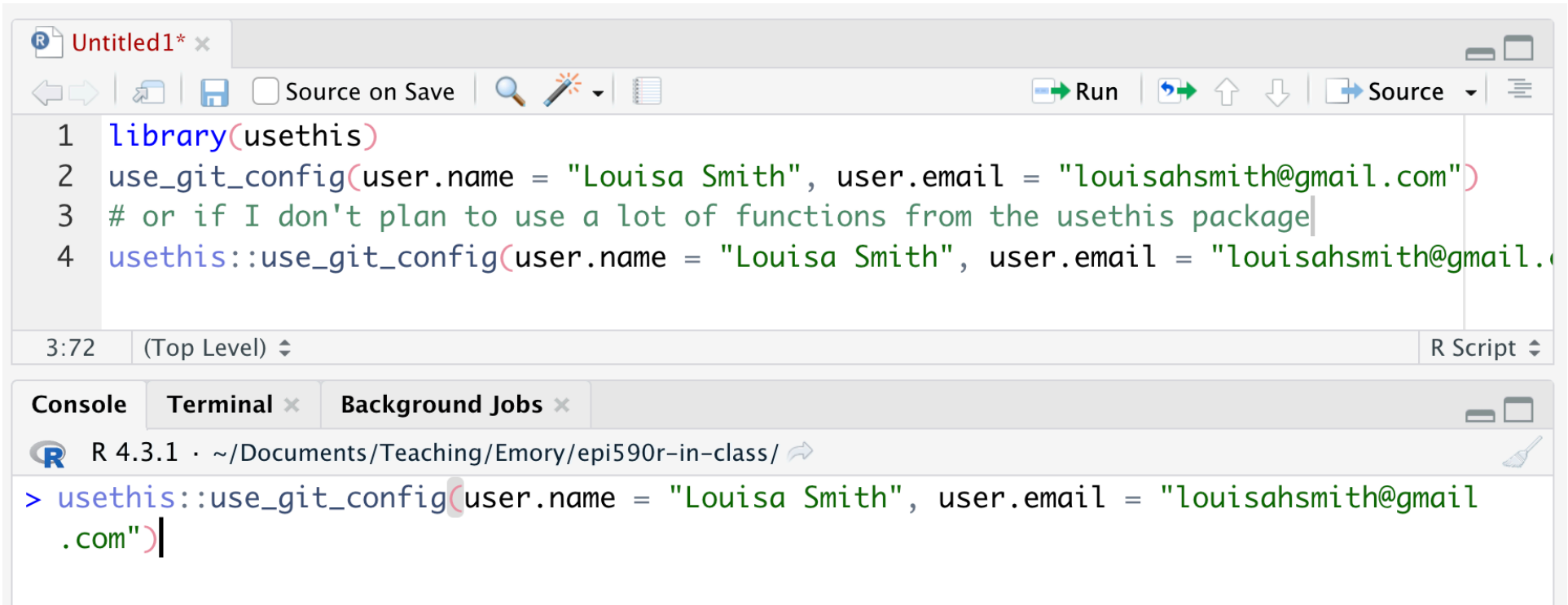
2. Introduce yourself:

```
usethis::use_git_config(user.name = "Louisa Smith",
user.email = "louisahsmith@gmail.com")
```

When you make changes to your code, they will be associated with this name and email address (this doesn't really matter for our purposes)

- You only need to do this once
- We'll explain all this in a little bit!

Since I only need to run this once, I would probably run this from the console (bottom) rather than a script (top)



Running from the console is great for `install.packages()`, quick calculations, fiddling with code until you get it right, or scenarios like this – otherwise save your code in a script!

Connect to GitHub

3. Create a github token:

```
usethis::create_github_token()
```

Instead of entering your password every time, this is a secure way to connect to GitHub

- If you are ever asked for your GitHub password in RStudio, you *have* to give this instead

Connect to GitHub

4. Copy the token
5. Back in R, run this code and paste your token in the console where it says “Enter password”:

```
gitcreds::gitcreds_set()
```

You can do this again whenever your token expires or you are using a different device

Exercises

- Refer back to the slides as needed
- Ask a classmate if you're stuck
- Raise your hand for the teaching team
- Done early? Help a friend! Read the resources section! Play around in R! Check your email!

15:00

Git and GitHub

A brief introduction

Git

- version control system
- works offline (*repositories* exist on your computer)
- tracks changes via *commits*
- has a command-line interface and integrations with GUIs (like RStudio)

GitHub

- web-based platform built around Git
- provides a remote location for hosting Git repositories
- enables collaboration
- offers other features for project management (pull requests, issue tracking)

Our Git/GitHub goals

- For **you**: Keep track of progress on projects
  - Go back when you need to
  - Don’t lose old work
  - Easily search the history of a project
- For **others**: Share your work
  - Have a place to store and link to code
  - Read and interact with others’ code

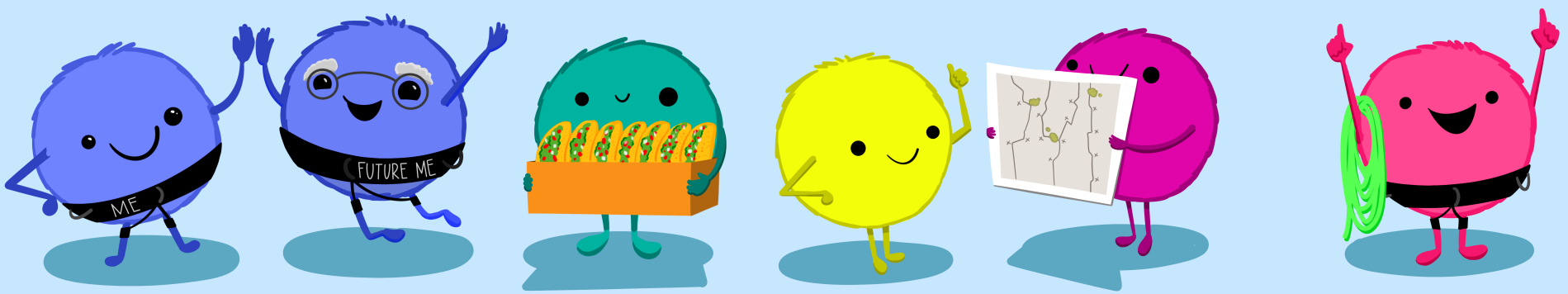
There is a ***lot*** to learn about this topic and I am ***not*** an expert on everything!

What we won't cover

- Collaboration
  - When multiple people are working on the same GitHub project, things get a little more complex
  - I went though almost my whole PhD without working on shared GitHub projects and only now do I feel semi-confident collaborating!
  - I think it's best to figure things out in your own projects first
- Git on the command line
  - There are a lot of functions you might hear about (`git fetch`, `git merge`, etc.)
  - RStudio and GitHub will have everything we need!

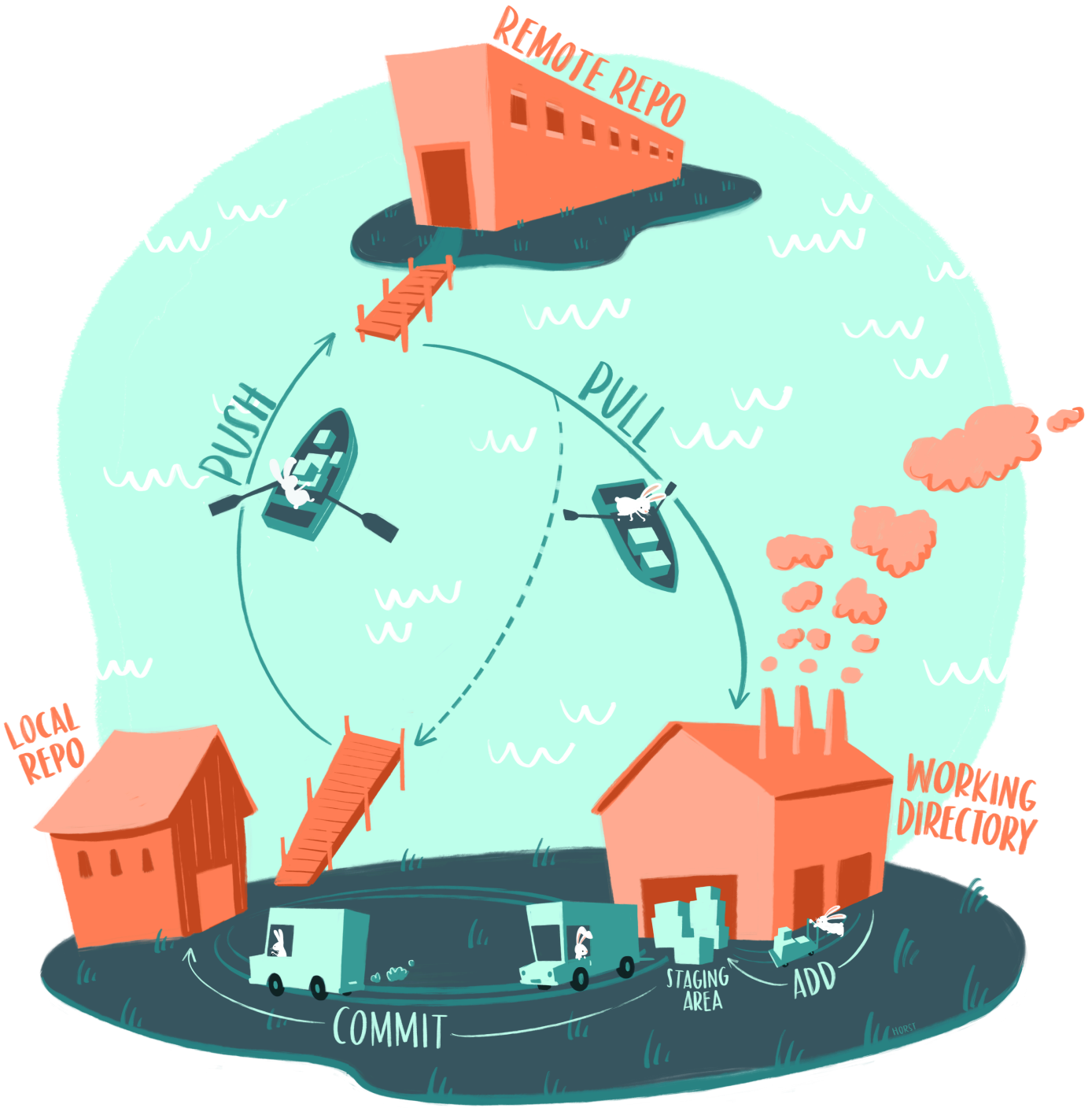
"**Collaboration is the most compelling reason to manage a project with Git and GitHub**. My definition of collaboration includes hands-on participation by multiple people, including your past and future self, as well as an asymmetric model, in which some people are active makers and others only read or review."

-JENNY BRYAN



Bryan, J. 2017. Excuse me, do you have a moment to talk about version control? PeerJ Preprints. 5:e3159v2. DOI: 10.7287/peerj.preprints.3159v2

Illustrations from the Openscapes blog **GitHub for supporting, contributing, and**


REMOTE REPO
PUSH
PULL
LOCAL REPO
WORKING DIRECTORY
STAGING AREA
ADD
COMMIT

Git

- version control system
- works offline (*repositories* exist on your computer)
- tracks changes via *commits*
- has a command-line interface and integrations with GUIs (like RStudio)

GitHub

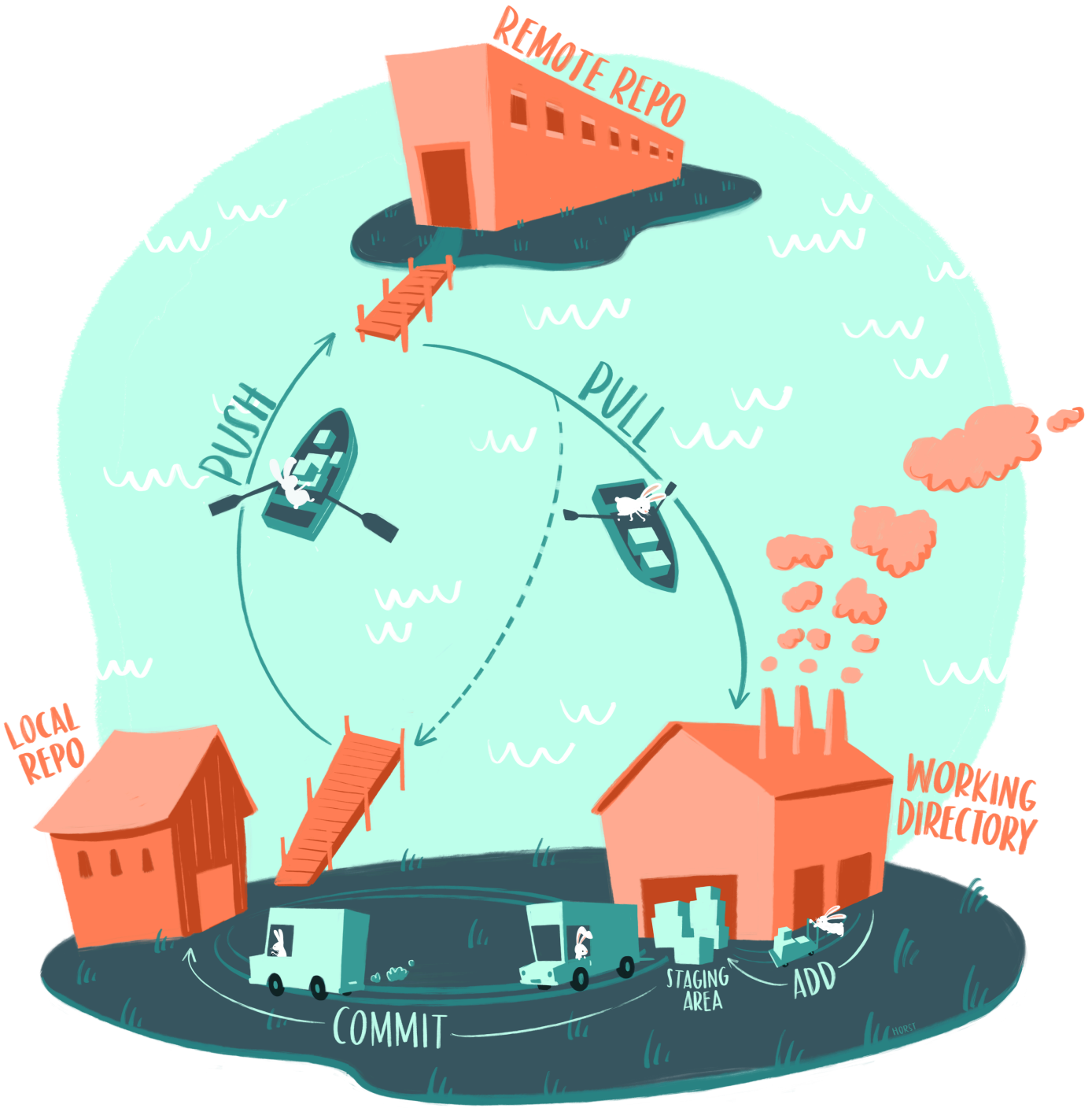
- web-based platform built around Git
- provides a remote location for hosting Git repositories
- enables collaboration
- offers other features for project management (pull requests, issue tracking)

Workflow

Create a repository (clone from GitHub, or create on your computer and connect to GitHub)

1. Write some code!
2. When you complete “something”, **add** it to the **staging** area
3. Write a brief description of what you did (“added linear model”; “created table 1”) and **commit**
4. **Push** to GitHub
5. Repeat!

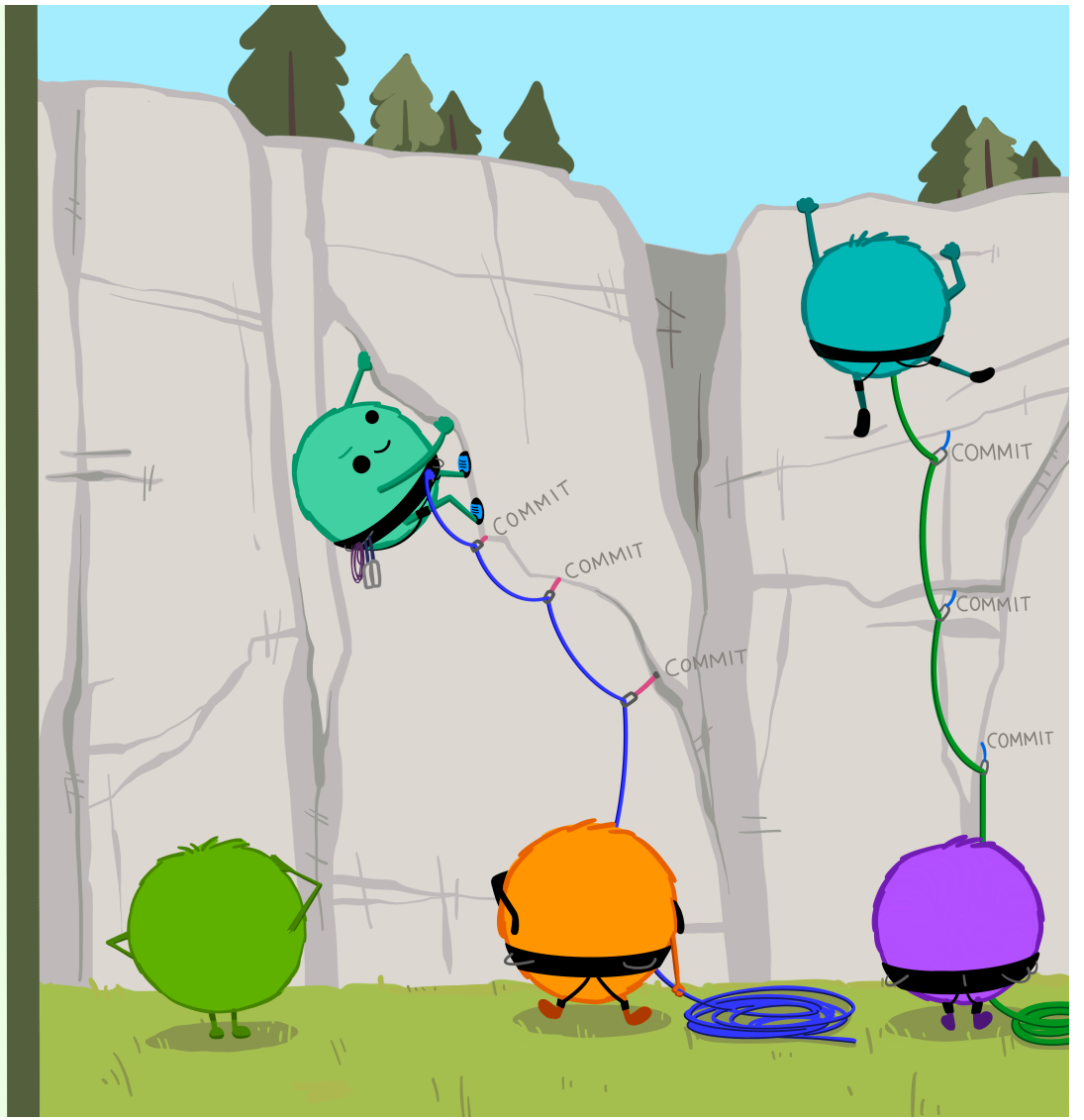
As long as you are working on your own, all on the same computer, you don’t need to worry about pulling


REMOTE REPO
PUSH
PULL
LOCAL REPO
WORKING DIRECTORY
STAGING AREA
ADD
COMMIT

What is a commit?

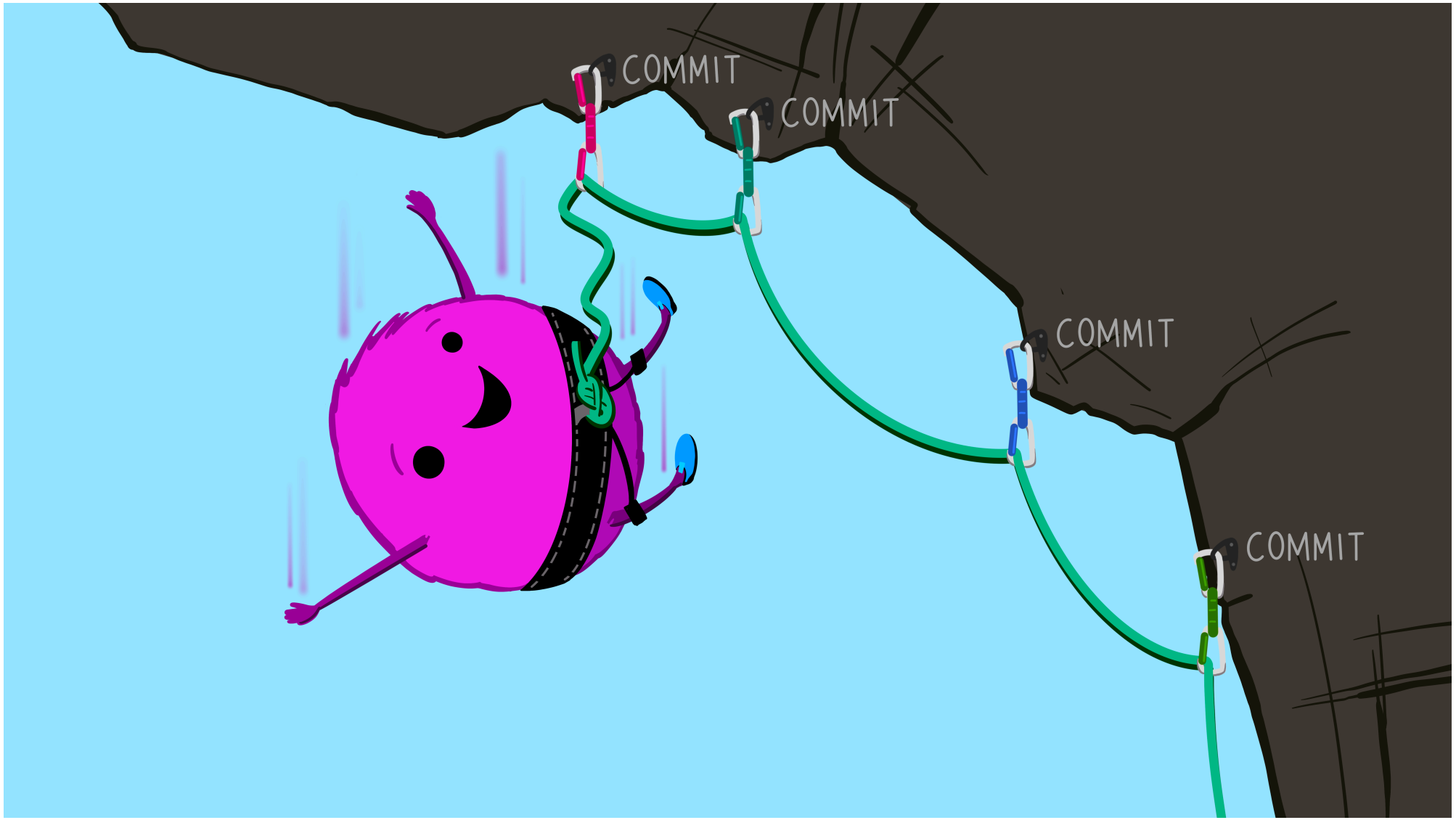
> Using a Git commit is like using anchors and other protection when climbing…**if you make a mistake, you can't fall past the previous commit**.
>
> Commits are also helpful to others, because **they show your journey, not just the destination**.
>
> – HADLEY WICKHAM & JENNY BRYAN

Wickham & Bryan, RPackages (https://r-packages.org/preface.html)



Illustrations from the Openscapes blog **GitHub for supporting, contributing, and**

What should you commit? Whatever you don’t want to lose!



Illustrations from the Openscapes blog **GitHub for supporting, contributing, and**

additional sensitivity analyses, re-render
louisahsmith committed on Mar 11, 2021
3a9a03f

figure out mice problem
louisahsmith committed on Mar 11, 2021
4a75681

tried and failed imputation
louisahsmith committed on Mar 11, 2021
21b38eb

add sensitivity analyses
louisahsmith committed on Mar 11, 2021
945ca2c

rerender text
louisahsmith committed on Mar 11, 2021
efb9476

updates to text
louisahsmith committed on Mar 11, 2021
3d6da19

render doc
louisahsmith committed on Mar 11, 2021
91ecf49

update pic
louisahsmith committed on Mar 11, 2021
16a8b21

add vertical line to pic
louisahsmith committed on Mar 11, 2021
609e76a

If you know that your code worked at 10am on October 21, 2015, and now it doesn't, you can return!

Exercises

1. *Fork* the repo (repository) at https://github.com/louisahsmith/epi590r-in-class
2. On **your** fork of the repository, click the green "Code" button. We are going to *clone* the repository to your computer using *HTTPS*.

Forking

- **Purpose:** Used to create a personal copy of another user's repository on your GitHub account.
- **Ownership:** The forked repository is still on the original owner's account, and you get your own copy to work with.
- **Collaboration:** Allows you to make changes without affecting the original repository. You can make changes, commit them to your fork, and then propose these changes to the original repository through pull requests.
- **Relationship:** The forked repository remains connected to the original, but changes aren't automatically synced.
- **Use Case:** Commonly used when you want to contribute to a project that you don't have direct write access to.

Cloning

- **Purpose:** Used to make a local copy of a GitHub repository on your computer.
- **Ownership:** You have a read-write copy on your local machine, but it's not automatically linked to your GitHub account (you can do so through RStudio).
- **Collaboration:** Allows you to work on the project locally and make changes, but these changes aren't automatically visible to others.
- **Relationship:** The cloned repository is a standalone copy, and changes won't automatically affect the original or other clones.
- **Use Case:** Useful when you want to work on a project locally and have full control over commits and pushes.

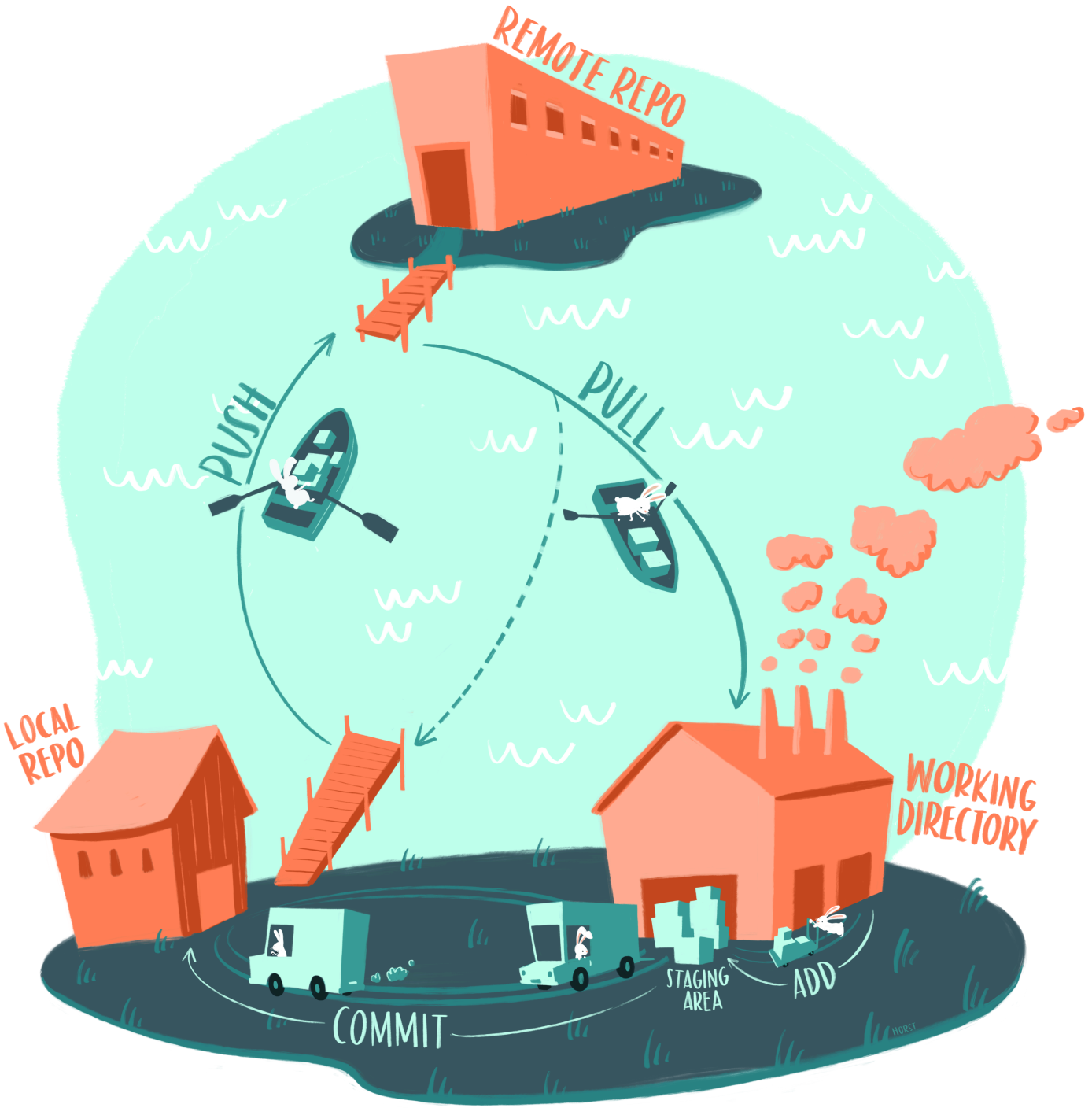
You have a forked repo on GitHub, now you are cloning that forked repo on your own computer

3. Open up RStudio.

- File > New Project > “Version Control” > “Git”

4. Paste the URL to your fork

- Name the project directory (easiest if it has the same name as the repo)
- Choose where you want to store the project (remember this spot!)
- Create Project!


REMOTE REPO
PUSH
PULL
LOCAL REPO
WORKING DIRECTORY
STAGING AREA
ADD
COMMIT

Practice making a change, staging, committing, pushing

5. From the filepane in RStudio, open `README.md`

- Change the file and save your changes

6. In your Git pane, click on the checkbox to stage the file

- Then click “Commit”

Exercise: edit, commit, push readme file

15:00

File management and projects in R

or, How to keep your computer safe from fire

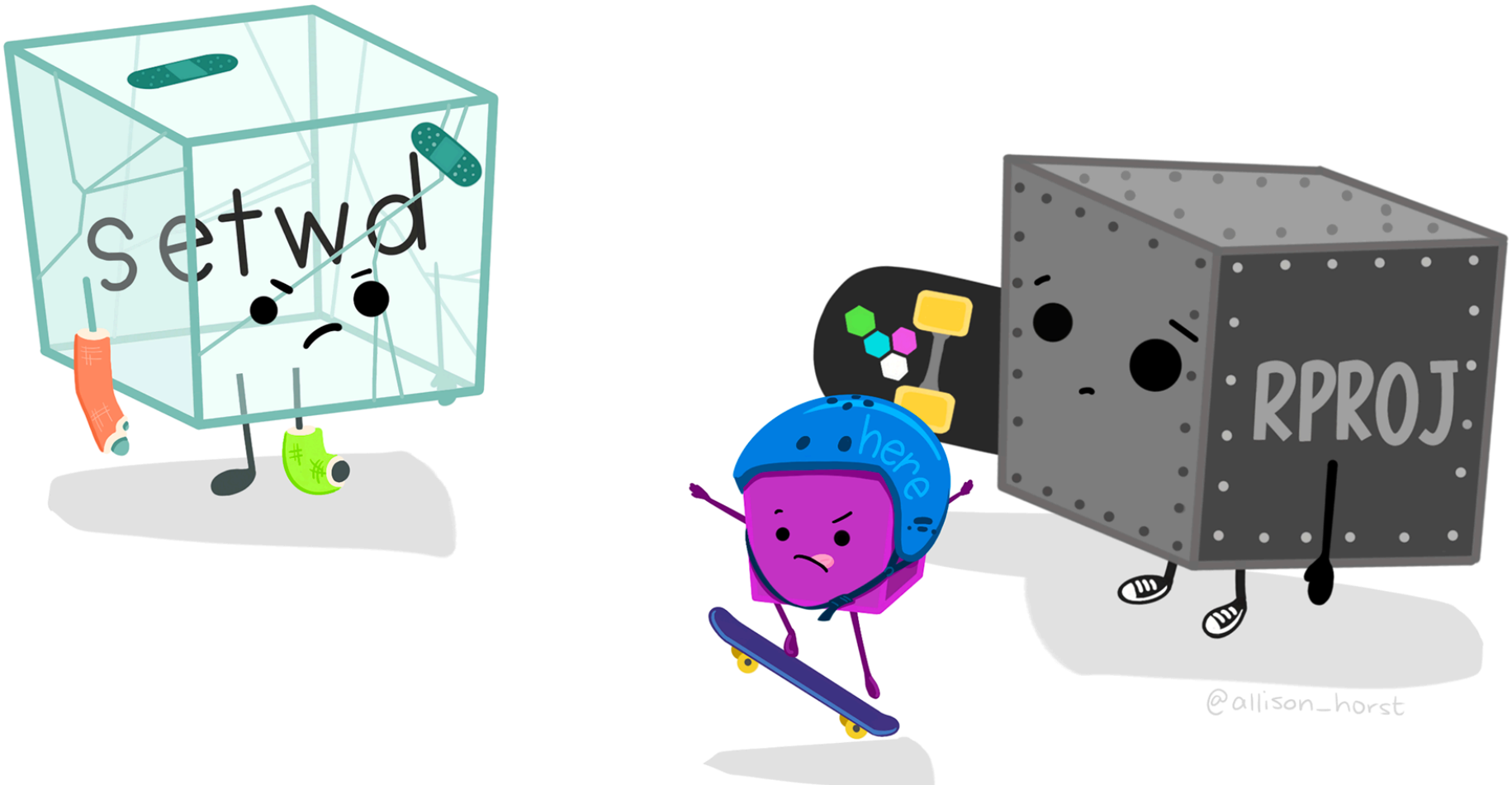
There's a famous blog post about workflows in R[1] about a talk Jenny Bryan gave that included this slide:

> If the first line of your R script is
>
> ```
> 1  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
>
> ```
> 1  rm(list = ls())
> ```
>
> I will come into your office and SET YOUR COMPUTER ON FIRE 🔥.

Instead: project-oriented workflow

- R projects provide a structured and organized way to work on projects in R
- R projects encapsulate all project-related files and settings into a single directory
- RStudio makes it easy to work with R projects

R Projects (and related tools) can prevent a lot of accidents!


setwd
here
RPROJ
@allison_horst

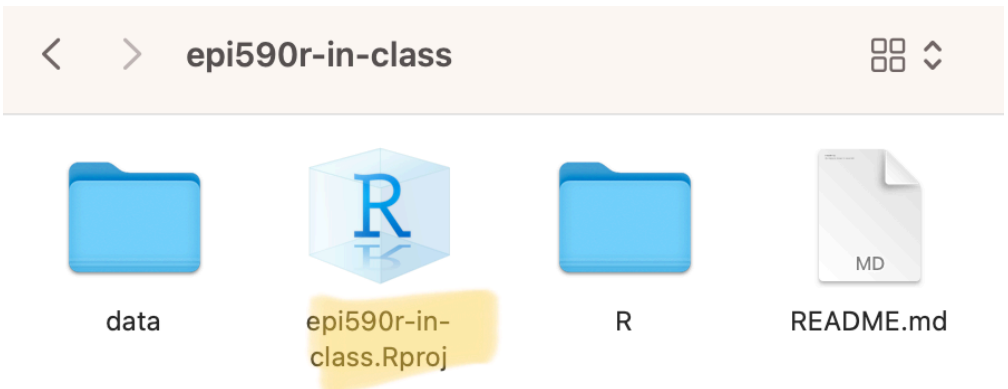
R Projects

Benefits of R Projects

1. **Isolation**: Each project has its own workspace, separate from other projects
2. **Reproducibility**: Projects ensure that code and data are self-contained and portable
3. **Collaboration**: Projects facilitate collaboration by sharing the entire project directory

Always open a project by opening the `.Rproj` file

Mac | Windows



You can have multiple projects open at once in different RStudio sessions!

You can also switch between R projects from RStudio

Creating an R Project

1. Open RStudio and go to **File > New Project**, or click on the projects button in the upper-right corner of RStudio.
2. Choose a project location (New Directory, Version Control, Existing Directory).
3. Specify the project directory (where on your computer you are storing the folder with the project) and create the project.
4. Choose the project type (e.g., regular project, R package, Shiny app, Quarto website, Bookdown book)

You already have an R project!

In the exercises, we are going to make some more changes to the repo you *forked* and *cloned*

1. Download an `.R` script and a `.csv` file from the website

- We'll be using some data from the 1979 National Longitudinal Survey of Youth

2. Find your `epi590r-in-class` repo in your file browser

- Create an `R` folder and a `data` folder
- Within the `data` folder add a `raw` and a `clean` folder.
- Put the `.csv` file in the `data/raw` folder and the script in `R` folder.

File structure goal

```
epi590r-in-class/
├─ epi590r-in-class.Rproj
├─ README.md
├─ R/
│   └─ clean-data-bad.R
├─ data/
│   ├─ raw/
│   │   └─ nlsy.csv
│   └─ clean/
```

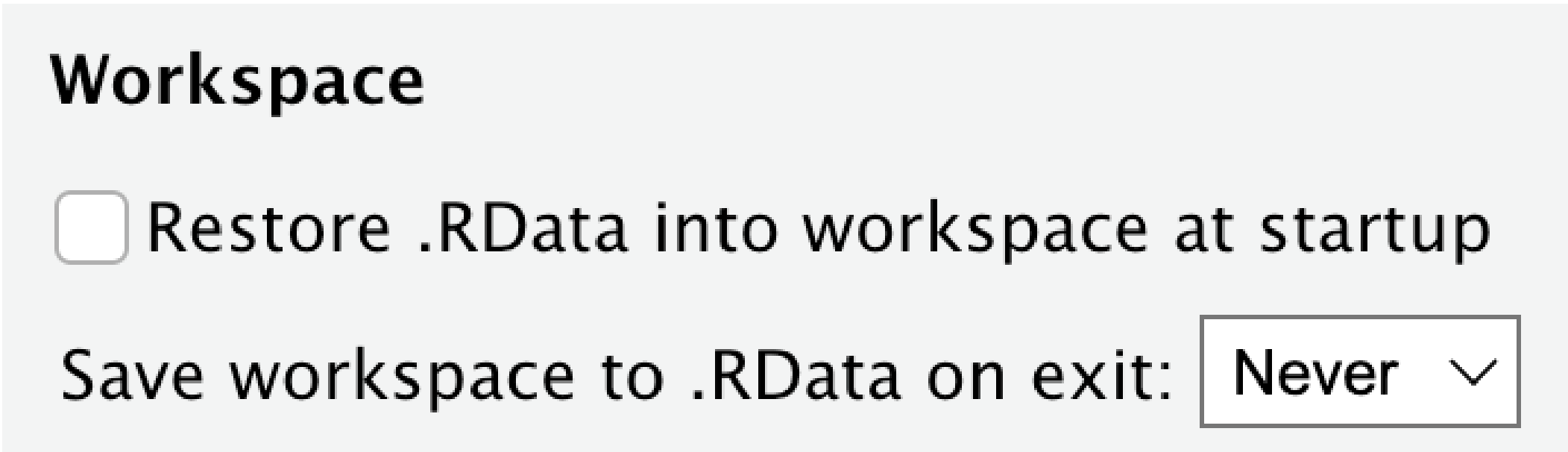
Exercises, cont.

3. Return to RStudio. If you closed RStudio, make sure you re-open this project. Look to the filepane to confirm the files are there.
4. Stage, commit, and push the changes you've made.
5. Try to run the code, line-by-line, in `clean-data-bad.R`.

- As you're running it, try to think of changes you might make

Stop for a settings change!

6. Tell RStudio to start fresh whenever you start a new session



7. Close RStudio, then open it up again by opening the `epi590r-in-class.Rproj` file in your file browser

Exercise: work with files in your R project

15:00

The {here} package

Fire safety, continued

The problem with `setwd()`

- `setwd()` changes the working directory, leading to potential issues in collaboration and reproducibility
  - You and I don't have the same file structure!
  - For example, my current working directory is

```
1  getwd()
```

```
[1] "/Users/l.smith/Documents/Teaching/Emory/epi590r-2024"
```

- It's also really annoying to change your working directory when you move around files and folders, even if it's just you using them

Do you think this code from 2015 still works?

Source on Save

```
1  #Create results table for poster
2  rm(list=ls())
3  setwd("~/Box Sync/Behavior Outcomes/")
4
5  #Load data
6  load("./Data/behavior_datasets.Rdata")
7
```

Referring to files with the here package

```
source(here::here("R", "functions.R"))

dat <- read_csv(here::here("data", "raw", "data.csv"))

p <- ggplot(dat) + geom_point(aes(x, y))

ggsave(plot = p,
       filename = here::here("results", "figures", "fig.pd
```

You can also separate the file paths with `/`:

```
dat <- read_csv(here::here("data/raw/data.csv"))
```

How it works

- Construct file paths with reference to the top directory holding your `.Rproj` file.
- `here::here("data", "raw", "data.csv")` for me, here, becomes `"/Users/l.smith/Documents/Teaching/Emory/epi590r-2024/data/raw/data.csv"`
  - If I change my working directory to somewhere else within my project, it will still give me that path
- And if I send you my code to run, it will become whatever file path *you* need it to be, as long as you're running it within the R Project.

Referring to the here package

```
here::here()
```

is equivalent to

```
library(here)
here()
```

I just prefer to write out the package name whenever I need it, but you can load the package for your entire session if you want.

Exercises

1. Install the `{here}` package:

   `install.packages("here")`

2. Make sure you're in your in-class R project! In the console, run:

   `here::here()` to print your project directory
   `getwd()` to print your working directory

   What do you notice?

3. In the console, run:

   `setwd("data")` to set your working directory

   Then run the same lines as above. What do you notice?

Exercises, cont.

4. Download the next `.R` script from the website and use your file browser to put it in the R folder in your project.

- Run through the code line-by-line.
- Compare it with the code from the last section.

15:00

Creating new projects

Starting from scratch

EPI 590R final project

Your goal will be to create an analysis that

- I can reproduce on my own computer
- Is easy to rerun if I tell you, for example, to remove the 12th row of your dataset

We'll start this in class!

New projects

We cloned our first project from GitHub; now we are going to start a new project from scratch

1. File > New Project > New Directory > New Project

- If you ever want to convert an existing folder that holds an analysis into an R project, you can choose “Existing Directory”
- You’ll also see other options besides “New Project” – an R package, a Shiny app, etc.
  - These will get you set up with some initial files for these types of projects
  - You can also make a template of your own!

New projects

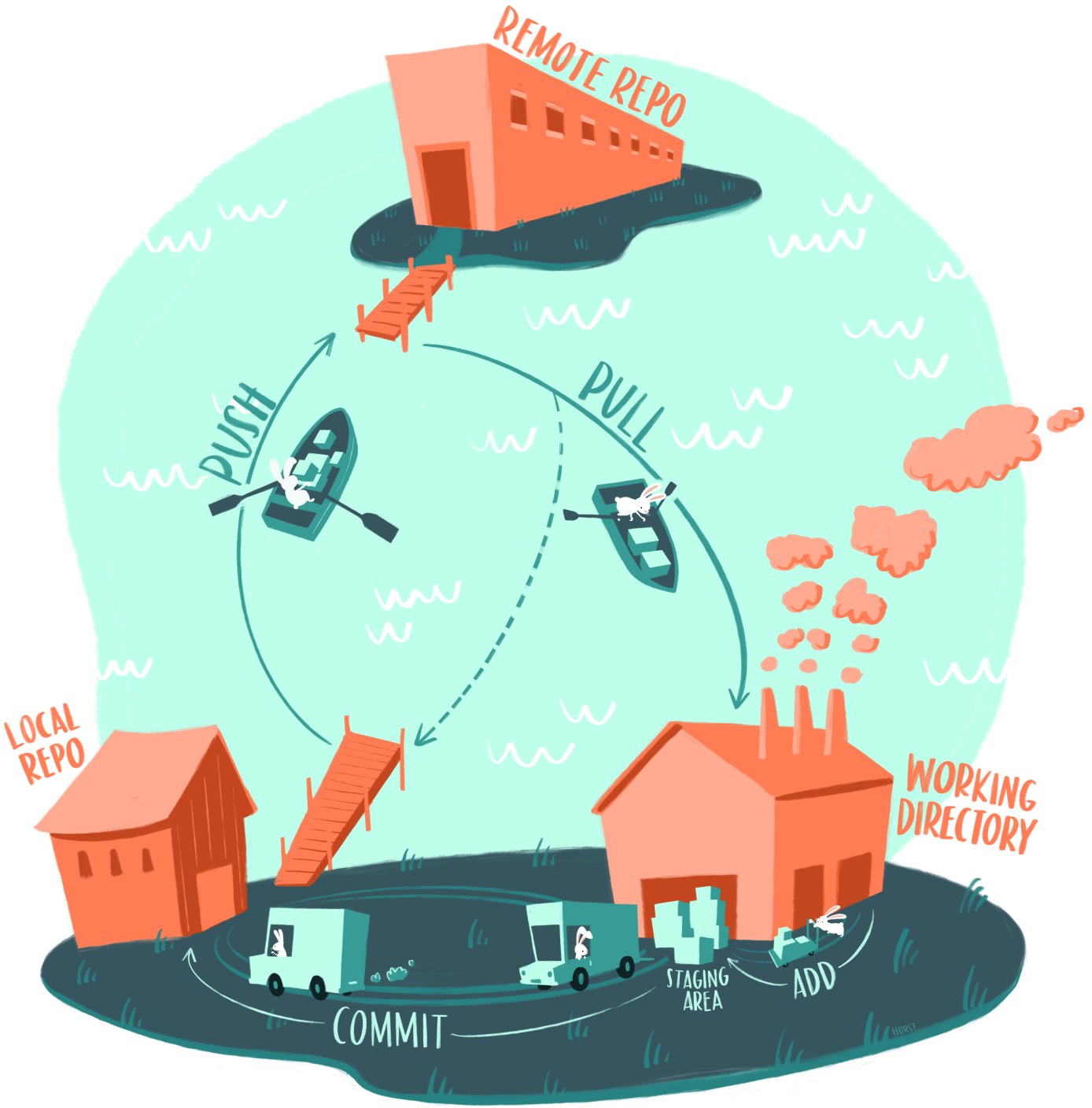
2. Choose a name for your new project and where to store it on your computer

- Check “Create a git repository”
  - This gets you all set to connect to GitHub and creates a `.gitignore` file
- You can leave “Use renv with this project” unchecked (we’ll be introducing the `{renv}` package later!)

Initial Git commit

3. Stage and commit the files

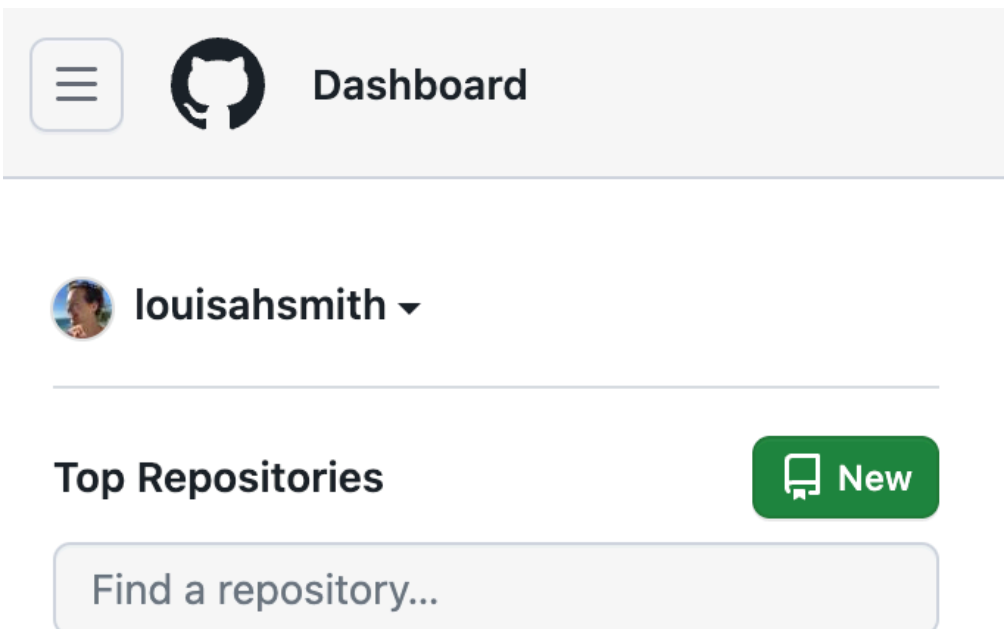
- I usually use “initial commit” as my first commit message since I haven’t don’t anything yet!
- We can’t yet push because we haven’t connected to a remote repository


REMOTE REPO
PUSH
PULL
LOCAL REPO
WORKING DIRECTORY
STAGING AREA
ADD
COMMIT

Creating a new repo on GitHub

4. Open up your web browser to GitHub and make a new repository

From github.com | From github.com/username



Repository options

5. Choose a name (preferably one that matches the name you gave your R project).

- You can choose to make it private, if you wish
  - Private repos have fewer features *unless* you have GitHub Pro (which you can get for free as a student with the GitHub student developer pack!)
- You don't need to click anything else

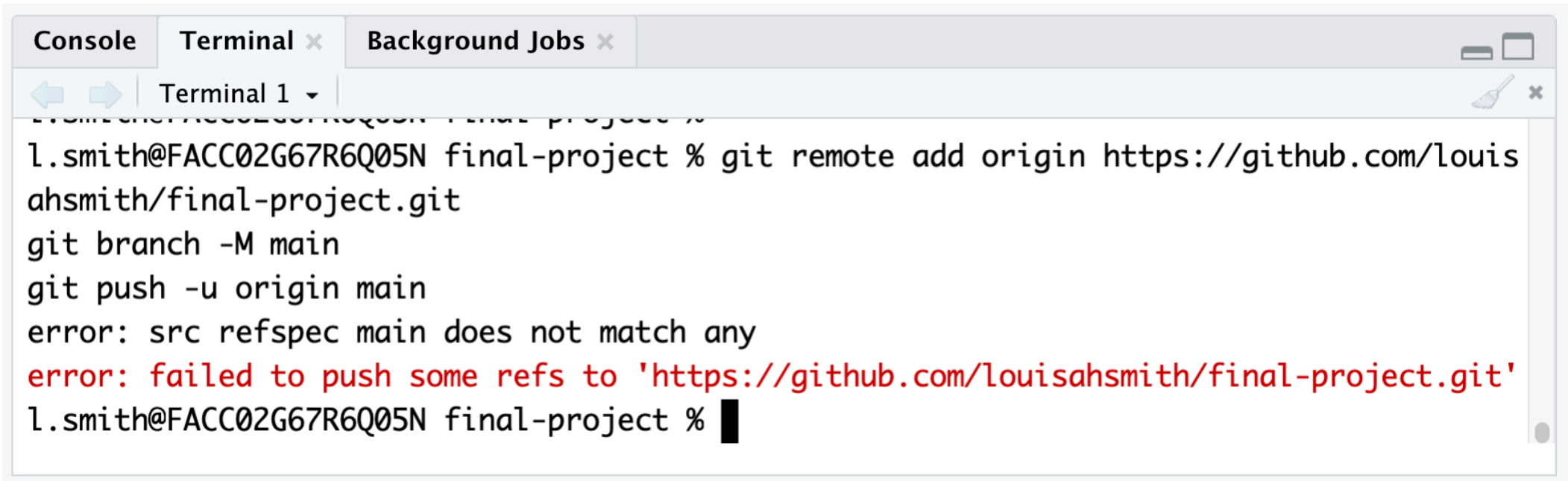
Connect the local to the remote

- You created your local repo with RStudio in a directory you chose
- Now you need to connect it to the remote repo on GitHub

6. Copy the code from the second section: “push an existing repository from the command line” in the *terminal* within RStudio.

Connect the local to the remote

7. Run the three lines of code *one at a time*, then refresh your GitHub page!



`.gitignore`

You likely don't want to push everything to GitHub, even if you have a private repository

- Be especially careful about data and passwords
- You also can't push very large files (>100 mb)

A `.gitignore` is a special text file that tells Git not to track certain files

- RStudio starts you off with a few entries, including `.Rhistory` since no one needs to see everything you've run in R!

`.gitignore` exercises

8. Create a new file called `secrets.txt` within this new repo

- Write down your deepest, darkest secrets and save (e.g., I am scared of spiders)

9. Open `.gitignore` via the RStudio filepane

- Add “secrets.txt” below the files that RStudio helpfully ignored for you
- Save

Keep your eye on the Git pane!

Starting the final project

10. Set up your folders how you'd like in your repo (you can always change this)

- Find some data, download it, and store it in your repo
- Commit and push to GitHub!

For your final project, your data must be something that can be stored online and accessed by me.

Some fun options for data are:

- https://data.fivethirtyeight.com/
- https://github.com/rfordatascience/tidytuesday#datasets
- https://github.com/higgi13425/medicaldata/tree/master/data (descriptions: https://higgi13425.github.io/medicaldata/#list-of-datasets)

Exercises

Get started making a new project and GitHub repo for your final project, editing the `.gitignore` file, and finding some fun data

You can always change anything you want later, and even delete the whole thing and start fresh!

15:00

Descriptive tables with {gtsummary}

Make an easy Table 1

What is `{gtsummary}`?

- Create tables that are publication-ready
- Highly customizable
- Descriptive tables, regression tables, etc.

gtsummary::tbl_summary()

```
1 library(gtsummary)
2
3 tbl_summary(
4   nlsy,
5   by = sex_cat,
6   include = c(sex_cat,
7               eyesight
```

Characteristic	**Male** N = 6,403 [1]	**Female** N = 6,283 [1]
race_eth_cat		
Hispanic	1,000 (16%)	1,002 (16%)
Black	1,613 (25%)	1,561 (25%)
Non-Black, Non-Hispanic	3,790 (59%)	3,720 (59%)
region_cat		
Northeast	1,296 (21%)	1,254 (20%)
North Central	1,488 (24%)	1,446 (23%)
South	2,251 (36%)	2,317 (38%)
West	1,253 (20%)	1,142 (19%)
Unknown	115	124
eyesight_cat		
Excellent	1,582 (38%)	1,334 (31%)
Very good	1,470 (35%)	1,500 (35%)
Good	792 (19%)	1,002 (23%)

[1] n (%); Median (Q1, Q3)

Characteristic	Male N = 6,403 [1]	Female N = 6,283 [1]
Fair	267 (6.4%)	365 (8.5%)
Poor	47 (1.1%)	85 (2.0%)
Unknown	2,245	1,997
glasses	1,566 (38%)	2,328 (54%)
Unknown	2,241	1,995
age_bir	25 (21, 29)	22 (19, 27)
Unknown	3,652	3,091

[1] n (%); Median (Q1, Q3)

You can also refer to variables using helper functions

```
1 library(gtsummary)
2
3 tbl_summary(
4   nlsy,
5   by = sex_cat,
6   include = c(ends_wit
```

Characteristic	Male N = 6,403 [1]	Female N = 6,283 [1]
region_cat		
Northeast	1,296 (21%)	1,254 (20%)
North Central	1,488 (24%)	1,446 (23%)
South	2,251 (36%)	2,317 (38%)
West	1,253 (20%)	1,142 (19%)
Unknown	115	124
race_eth_cat		
Hispanic	1,000 (16%)	1,002 (16%)
Black	1,613 (25%)	1,561 (25%)
Non-Black, Non-Hispanic	3,790 (59%)	3,720 (59%)
eyesight_cat		
Excellent	1,582 (38%)	1,334 (31%)

[1] n (%); Median (Q1, Q3)

Characteristic	Male N = 6,403 [1]	Female N = 6,283 [1]
Very good	1,470 (35%)	1,500 (35%)
Good	792 (19%)	1,002 (23%)
Fair	267 (6.4%)	365 (8.5%)
Poor	47 (1.1%)	85 (2.0%)
Unknown	2,245	1,997
glasses	1,566 (38%)	2,328 (54%)
Unknown	2,241	1,995
age_bir	25 (21, 29)	22 (19, 27)
Unknown	3,652	3,091

[1] n (%); Median (Q1, Q3)

We probably want to name the variables

```r
tbl_summary(
  nlsy,
  by = sex_cat,
  include = c(sex_cat, race_eth_cat,
              eyesight_cat, glasses,
  label = list(
    race_eth_cat ~ "Race/ethnicity",
    region_cat ~ "Region",
    eyesight_cat ~ "Eyesight",
    glasses ~ "Wears glasses",
    age_bir ~ "Age at first birth"
  ),
  missing_text = "Missing")
```

Characteristic	**Male** N = 6,403 [1]	**Female** N = 6,283 [1]
Race/ethnicity		
Hispanic	1,000 (16%)	1,002 (16%)
Black	1,613 (25%)	1,561 (25%)
Non-Black, Non-Hispanic	3,790 (59%)	3,720 (59%)
Region		
Northeast	1,296 (21%)	1,254 (20%)
North Central	1,488 (24%)	1,446 (23%)
South	2,251 (36%)	2,317 (38%)
West	1,253 (20%)	1,142 (19%)
Missing	115	124
Eyesight		
Excellent	1,582 (38%)	1,334 (31%)
Very good	1,470 (35%)	1,500 (35%)
Good	792 (19%)	1,002 (23%)

[1] n (%); Median (Q1, Q3)

Characteristic	Male N = 6,403 [1]	Female N = 6,283 [1]
Fair	267 (6.4%)	365 (8.5%)
Poor	47 (1.1%)	85 (2.0%)
Missing	2,245	1,997
Wears glasses	1,566 (38%)	2,328 (54%)
Missing	2,241	1,995
Age at first birth	25 (21, 29)	22 (19, 27)
Missing	3,652	3,091

[1] n (%); Median (Q1, Q3)

And do a million other things

```
tbl_summary(
  nlsy,
  by = sex_cat,
  include = c(sex_cat, race_eth_cat,
              eyesight_cat, glasses,
  label = list(
    race_eth_cat ~ "Race/ethnicity",
    eyesight_cat ~ "Eyesight",
    glasses ~ "Wears glasses",
    age_bir ~ "Age at first birth"
  ),
  missing_text = "Missing") |>
  add_p(test = list(all_continuous()
                    all_categorical(
  add_overall(col_label = "**Total**
  bold_labels() |>
  modify_footnote(update = everythin
  modify_header(label = "**Variable*
```

Variable	Total	Male N = 6,403	Female N = 6,283	P
Race/ethnicity				
Hispanic	2,002 (16%)	1,000 (16%)	1,002 (16%)	
Black	3,174 (25%)	1,613 (25%)	1,561 (25%)	
Non-Black, Non-Hispanic	7,510 (59%)	3,790 (59%)	3,720 (59%)	
Eyesight				
Excellent	2,916 (35%)	1,582 (38%)	1,334 (31%)	
Very good	2,970 (35%)	1,470 (35%)	1,500 (35%)	
Good	1,794 (21%)	792 (19%)	1,002 (23%)	
Fair	632 (7.5%)	267 (6.4%)	365 (8.5%)	
Poor	132 (1.6%)	47 (1.1%)	85 (2.0%)	
Missing	4,242	2,245	1,997	
Wears glasses	3,894 (46%)	1,566 (38%)	2,328 (54%)	
Missing	4,236	2,241	1,995	

Additional arguments

We saw `include =`, `by =`, `label =`, `missing_text =` in the example

`statistic =`:

- The default is `list(all_continuous() ~ "{median} ({p25}, {p75})", all_categorical() ~ "{n} ({p}%)")`
- For categorical variables, you can use `{n}` (frequency), `{N}` (denominator), `{p}` formatted percentage
- For continuous variables, you can use `{median}`, `{mean}`, `{sd}`, `{var}`, `{min}`, `{max}`, `{sum}`, `{p##}` (any percentile), or any function `{foo}`
- You can refer to individual variables with their names: `list(age ~ "min = {min}; max = {max}")`

Additional arguments

`digits =`:

- It will do its best to guess the appropriate number of digits
- Otherwise, you can pass a function:
  - `digits = everything() ~ style_sigfig`
- Or a value for each statistic shown
  - `statistic = list(age ~ "min = {min}; max = {max}", year_of_birth = "{median} ({p25}, {p75})")`:
  - `digits = list(age ~ c(1, 1) year_of_birth ~ c(0, 0, 0))`

Additional arguments

`type =`:

- One of “continuous”, “continuous2”, “categorical”, “dichotomous”
  - If a variable only has 0/1, TRUE/FALSE, or yes/no values, it will be treated as dichotomous
    - You can override this with `type = list(``varname``~ "categorical")`
    - Dichotomous variables only show one row (i.e., the percentage of 1’s) unless you change to categorical
      - You can change which level to show with `value = list(varname ~ "level to show")`
  - “continuous2” variables can have multiple rows of statistics

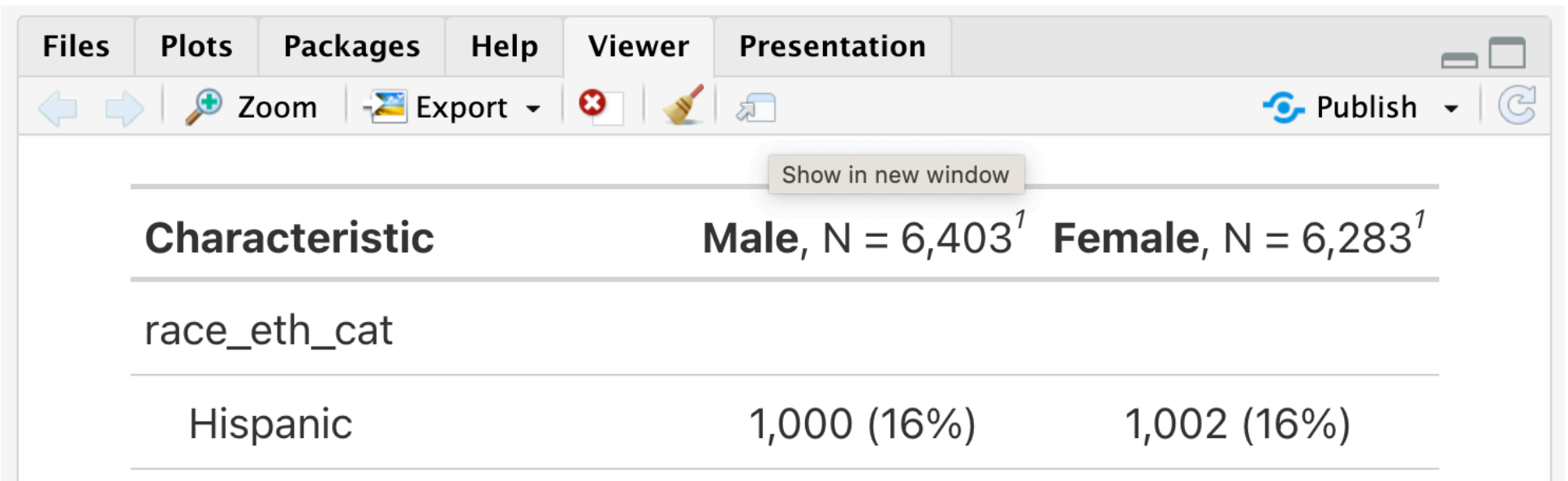
`missing =`:

Additional functions

- `add_overall()`: In a stratified table, add a column for all strata combined
- `bold_labels()`: Bold the variable names (also `bold_levels()`)
- `add_p()`: Add a p-value (required by some journals 🤷‍♀️)
- `modify_footnote(update = everything() ~ NA)`: Remove the footnotes (can also add footnotes!)
- `modify_header()`: Change the header column

```
tbl_summary()
```

- Incredibly customizeable
  - So many options can be overwhelming
  - The FAQ/gallery is an incredible resource
- To save, I often just view in the web browser and copy and paste into a Word document
  - Can also be used within quarto/R Markdown



Exercises

1. Open the script with some examples.
2. Install `{gtsummary}` and run the examples.

3-7. You're on your own! Work with your neighbors, and we'll come back together to go over these.

Extra time? Start a table using the data you downloaded for your final project! Make sure you switch to that R project!

15:00

Regression tables with {gtsummary}

On to Table 2!

Univariate regressions

Fit a series of univariate regressions of income on other variables.

```
1 tbl_uvregression(
2   nlsy,
3   y = income,
4   include = c(sex_cat,
5               eyesight
6   method = lm)
```

Characteristic	N	Beta	95% CI [1]	p-value
sex_cat	10,195			
Male		—	—	
Female		-358	-844, 128	0.15
race_eth_cat	10,195			
Hispanic		—	—	
Black		-1,747	-2,507, -988	<0.001
Non-Black, Non-Hispanic		3,863	3,195, 4,530	<0.001
eyesight_cat	6,789			
Excellent		—	—	
Very good		-578	-1,319, 162	0.13

[1] CI = Confidence Interval

Characteristic	N	Beta	95% CI [1]	p-value
Good		-1,863	-2,719, -1,006	<0.001
Fair		-4,674	-5,910, -3,439	<0.001
Poor		-6,647	-9,154, -4,140	<0.001
age_bir	4,773	595	538, 652	<0.001

[1] CI = Confidence Interval

Can also do logistic regression

```
tbl_uvregression(
  nlsy,
  y = glasses,
  include = c(sex_cat,
              eyesight
  method = glm,
  method.args = list(f
  exponentiate = TRUE)
```

Characteristic	N	OR [1]	95% CI [1]	p-value
sex_cat	8,450			
Male		—	—	
Female		1.97	1.81, 2.15	<0.001
race_eth_cat	8,450			
Hispanic		—	—	
Black		0.76	0.67, 0.86	<0.001
Non-Black, Non-Hispanic		1.34	1.19, 1.50	<0.001
eyesight_cat	8,444			
Excellent		—	—	
Very good		0.93	0.84, 1.03	0.2
Good		0.95	0.84, 1.07	0.4
Fair		0.81	0.68, 0.96	0.016
Poor		1.15	0.81, 1.63	0.4
age_bir	5,813	1.02	1.01, 1.03	<0.001

[1] OR = Odds Ratio, CI = Confidence Interval

We probably want to do some multivariable regressions

```
linear_model <- lm(income ~ sex_cat + age_bir + race_eth_c
                   data = nlsy)
```

```
linear_model_int <- lm(income ~ sex_cat*age_bir + race_eth
                   data = nlsy)
```

```
logistic_model <- glm(glasses ~ eyesight_cat + sex_cat + i
                      data = nlsy, family = binomial())
```

gtsummary::tbl_regression()

```
tbl_regression(
  linear_model,
  intercept = TRUE,
  label = list(
    sex_cat ~ "Sex",
    race_eth_cat ~ "Race/ethnicity",
    age_bir ~ "Age at first birth"
  ))
```

Characteristic	Beta	95% CI [1]	p-value
(Intercept)	2,147	493, 3,802	0.011
Sex			
Male	—	—	
Female	25	-654, 705	>0.9
Age at first birth	438	381, 495	<0.001
Race/ethnicity			
Hispanic	—	—	
Black	-772	-1,714, 171	0.11
Non-Black, Non-Hispanic	7,559	6,676, 8,442	<0.001

[1] CI = Confidence Interval

gtsummary::tbl_regression()

```
tbl_regression(
  logistic_model,
  exponentiate = TRUE,
  label = list(
    sex_cat ~ "Sex",
    eyesight_cat ~ "Eyesight",
    income ~ "Income"
  ))
```

Characteristic	OR [1]	95% CI [1]	p-value
Eyesight			
Excellent	—	—	
Very good	0.92	0.82, 1.03	0.2
Good	0.92	0.80, 1.05	0.2
Fair	0.80	0.66, 0.98	0.028
Poor	1.03	0.69, 1.53	0.9
Sex			
Male	—	—	
Female	2.04	1.85, 2.25	<0.001
Income	1.00	1.00, 1.00	<0.001

[1] OR = Odds Ratio, CI = Confidence Interval

Arguments

Argument	Description
`label=`	modify variable labels in table
`exponentiate=`	exponentiate model coefficients
`include=`	names of variables to include in output. Default is all variables
`show_single_row=`	By default, categorical variables are printed on multiple rows. If a variable is dichotomous and you wish to print the regression coefficient on a single row, include the variable name(s) here.
`conf.level=`	confidence level of confidence interval
`intercept=`	indicates whether to include the intercept
`estimate_fun=`	function to round and format coefficient estimates
`pvalue_fun=`	function to round and format p-values
`tidy_fun=`	function to specify/customize tidier function

You could put several together

```r
tbl_no_int <- tbl_regression(
  linear_model,
  intercept = TRUE,
  label = list(
    sex_cat ~ "Sex",
    race_eth_cat ~ "Race/ethnicity",
    age_bir ~ "Age at first birth"
  ))

tbl_int <- tbl_regression(
  linear_model_int,
  intercept = TRUE,
  label = list(
    sex_cat ~ "Sex"
```

You could put several together

```
tbl_merge(list(tbl_no_int, tbl_int),
          tab_spanner = c("**Model 1**", "**Model 2**"))
```

Characteristic	Model 1			Model 2		
	Beta	95% CI [1]	p-value	Beta	95% CI [1]	p-value
(Intercept)	2,147	493, 3,802	0.011	4,064	1,884, 6,245	<0.001
Sex						
Male	—	—		—	—	
Female	25	-654, 705	>0.9	-3,635	-6,432, -838	0.011
Age at first birth	438	381, 495	<0.001	364	285, 443	<0.001
Race/ethnicity						
Hispanic	—	—		—	—	
Black	-772	-1,714, 171	0.11	-759	-1,701, 183	0.11
Non-Black, Non-Hispanic	7,559	6,676, 8,442	<0.001	7,550	6,668, 8,433	<0.001
Sex/age interaction						

[1] CI = Confidence Interval

Characteristic	Model 1			Model 2		
	Beta	95% CI [1]	p-value	Beta	95% CI [1]	p-value
Female * Age at first birth				149	39, 260	0.008

[1] CI = Confidence Interval

Exercises

1. Open the script with some examples.
2. Run the examples.

3-6. You're on your own again!

Extra time? Start a table using the data you downloaded for your final project! Make sure you switch to that R project!

15:00

Finer control over statistics

We fit a series of univariate regressions

```r
income_table <- tbl_uv
  nlsy,
  y = income,
  include = c(
    sex_cat, race_eth_
    eyesight_cat, inco
  ),
  method = lm
)
income_table
```

Characteristic	N	Beta	95% CI [1]	p-value
sex_cat	10,195			
Male		—	—	
Female		-358	-844, 128	0.15
race_eth_cat	10,195			
Hispanic		—	—	
Black		-1,747	-2,507, -988	<0.001
Non-Black, Non-Hispanic		3,863	3,195, 4,530	<0.001
eyesight_cat	6,789			
Excellent		—	—	
Very good		-578	-1,319, 162	0.13
Good		-1,863	-2,719, -1,006	<0.001
Fair		-4,674	-5,910, -3,439	<0.001
Poor		-6,647	-9,154,	<0.001

[1] CI = Confidence Interval

Characteristic	N	Beta	95% CI [1]	p-value
			-4,140	
age_bir	4,773	595	538, 652	<0.001

[1] CI = Confidence Interval

But a table is a limited form of output

We might want to dig in a little more to those regressions

- One helpful option from `{gtsummary}` is to extract data from the table directly
- This can be reported in a manuscript (rather than copying and pasting from the table)

```
1  inline_text(income_table, variable = "age_bir")
```

```
[1] "595 (95% CI 538, 652; p<0.001)"
```

We'll look at this again later!

What if we want *all* the numbers, say to create a figure?

- Under the hood, `{gtsummary}` is using the `{broom}` package to extract the statistics from the various models
- We can also use that package directly!

Statistical models in R can be messy

```r
mod_sex_cat <- lm(income ~ sex_cat, data = nlsy)
```

We could look at the model summary:

```r
summary(mod_sex_cat)
```

```
Call:
lm(formula = income ~ sex_cat, data = nlsy)

Residuals:
   Min     1Q Median     3Q    Max
-14880  -8880  -3943   5477  60478

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    14880.3      172.6  86.237   <2e-16 ***
sex_catFemale   -357.8      247.8  -1.444    0.149
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12510 on 10193 degrees of freedom
  (2491 observations deleted due to missingness)
```

Statistical models in R can be messy

If we want to do something with the various values, we could extract each statistic individually:

```
coef(mod_sex_cat)
```

```
  (Intercept) sex_catFemale
   14880.3152     -357.8029
```

```
confint(mod_sex_cat)
```

```
                 2.5 %     97.5 %
(Intercept)   14542.079 15218.5512
sex_catFemale  -843.608   128.0022
```

```
summary(mod_sex_cat)$r.squared
```

```
[1] 0.0002044429
```

```
summary(mod_sex_cat)$coefficients
```

```
               Estimate Std. Error   t value  Pr(>|t|)
(Intercept)   14880.3152   172.5521 86.236672 0.0000000
sex_catFemale  -357.8029   247.8349 -1.443715 0.1488499
```

{broom} has three main functions: augment(), glance(), tidy()

augment() adds fitted values, residuals, and other statistics to the original data

```
1 library(broom)
2 augment(mod_sex_cat)
```

```
# A tibble: 10,195 × 9
   .rownames income sex_cat .fitted  .resid     .hat .sigma   .cooksd .std.resid
   <chr>      <dbl> <fct>     <dbl>   <dbl>    <dbl>  <dbl>     <dbl>      <dbl>
 1 1          30000 Female   14523.  15477. 0.000202 12506.   1.55e-4      1.24
 2 2          20000 Female   14523.   5477. 0.000202 12507.   1.94e-5      0.438
 3 3          22390 Female   14523.   7867. 0.000202 12507.   4.01e-5      0.629
 4 4          22390 Female   14523.   7867. 0.000202 12507.   4.01e-5      0.629
 5 5          36000 Male     14880.  21120. 0.000190 12505.   2.72e-4      1.69
 6 6          35000 Male     14880.  20120. 0.000190 12505.   2.46e-4      1.61
 7 7           8502 Male     14880.  -6378. 0.000190 12507.   2.48e-5     -0.510
 8 8           7227 Female   14523.  -7296. 0.000202 12507.   3.44e-5     -0.583
 9 9          17000 Male     14880.   2120. 0.000190 12507.   2.74e-6      0.170
10 10          3548 Female   14523. -10975. 0.000202 12506.   7.79e-5     -0.878
# ℹ 10,185 more rows
```

{broom} has three main functions: augment(), glance(), tidy()

glance() creates a table of statistics that pertain to the entire model

```
1  glance(mod_sex_cat)
```

```
# A tibble: 1 × 12
  r.squared adj.r.squared  sigma statistic p.value    df   logLik     AIC    BIC
      <dbl>         <dbl>  <dbl>     <dbl>   <dbl> <dbl>    <dbl>   <dbl>  <dbl>
1  0.000204      0.000106 12506.      2.08   0.149     1 -110644. 221295. 2.21e5
# ℹ 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

`{broom}` has three main functions: `augment()`, `glance()`, `tidy()`

`tidy()` is the most useful to me and probably you!

It extracts coefficients and confidence intervals from models

```
1  tidy(mod_sex_cat, conf.int = TRUE)
```

```
# A tibble: 2 × 7
  term          estimate std.error statistic p.value conf.low conf.high
  <chr>            <dbl>     <dbl>     <dbl>   <dbl>    <dbl>     <dbl>
1 (Intercept)     14880.      173.      86.2    0       14542.    15219.
2 sex_catFemale    -358.      248.      -1.44   0.149    -844.      128.
```

`tidy()` works on over 100 statistical methods in R!

Anova, ARIMA, Cox, factor analysis, fixed effects, GAM, GEE, IV, kappa, kmeans, multinomial, proportional odds, principal components, survey methods, ...

- See the full list here
- All the output shares column names
- This makes it really easy to work with the output and reuse code across analyses

Some models have additional arguments

For example, we might want exponentiated coefficients:

```
logistic_model <- glm(glasses ~ eyesight_cat + sex_cat + i
                      data = nlsy, family = binomial())
tidy(logistic_model, conf.int = TRUE, exponentiate = TRUE)
```

```
# A tibble: 7 × 7
  term                 estimate std.error statistic  p.value conf.low conf.high
  <chr>                   <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
1 (Intercept)             0.499   5.96e-2   -11.7   1.74e-31    0.444     0.560
2 eyesight_catVery good   0.920   5.96e-2    -1.39  1.64e- 1    0.819     1.03
3 eyesight_catGood        0.916   6.91e-2    -1.27  2.04e- 1    0.800     1.05
4 eyesight_catFair        0.802   1.00e-1    -2.20  2.77e- 2    0.658     0.976
5 eyesight_catPoor        1.03    2.01e-1     0.147 8.83e- 1    0.694     1.53
6 sex_catFemale           2.04    5.00e-2    14.2   5.46e-46    1.85      2.25
7 income                  1.00    1.93e-6     7.49  6.95e-14    1.00      1.00
```

We can also combine the results of lots of regressions

```
# we already made mod_sex_cat
mod_race_eth_cat <- lm(income ~ race_eth_cat, data = nlsy)
mod_eyesight_cat <- lm(income ~ eyesight_cat, data = nlsy)
mod_age_bir <- lm(income ~ age_bir, data = nlsy)

tidy_sex_cat <- tidy(mod_sex_cat, conf.int = TRUE)
tidy_race_eth_cat <- tidy(mod_race_eth_cat, conf.int = TRU
tidy_eyesight_cat <- tidy(mod_eyesight_cat, conf.int = TRU
tidy_age_bir <- tidy(mod_age_bir, conf.int = TRUE)
```

There are of course more efficient ways to do this instead of copy/pasting 4 times...

With a little finagling, we have the same data as in the original univartiate regression table...

```
dplyr::bind_rows(
  sex_cat = tidy_sex_cat,
  race_eth_cat = tidy_race_eth_cat,
  eyesight_cat = tidy_eyesight_cat,
  age_bir = tidy_age_bir, .id = "model") |>
  dplyr::mutate(
    term = stringr::str_remove(term, model),
    term = ifelse(term == "", model, term))
```

With a little finagling, we have the same data as in the original univartiate regression table...

```
# A tibble: 12 × 8
   model        term    estimate std.error statistic  p.value conf.low conf.high
   <chr>        <chr>      <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
 1 sex_cat      (Inter…   14880.     173.      86.2  0          14542.    15219.
 2 sex_cat      Female     -358.     248.      -1.44 1.49e- 1    -844.      128.
 3 race_eth_cat (Inter…   12867.     302.      42.7  0          12276.    13459.
 4 race_eth_cat Black     -1747.     387.      -4.51 6.58e- 6   -2507.     -988.
 5 race_eth_cat Non-Bl…    3863.     341.      11.3  1.20e-29    3195.     4530.
 6 eyesight_cat (Inter…   17683.     270.      65.6  0          17155.    18212.
 7 eyesight_cat Very g…    -578.     378.      -1.53 1.26e- 1   -1319.      162.
 8 eyesight_cat Good      -1863.     437.      -4.26 2.05e- 5   -2719.    -1006.
 9 eyesight_cat Fair      -4674.     630.      -7.42 1.35e-13   -5910.    -3439.
10 eyesight_cat Poor      -6647.    1279.      -5.20 2.07e- 7   -9154.    -4140.
11 age_bir      (Inter…    1707.     733.       2.33 1.99e- 2     270.     3143.
12 age_bir      age_bir     595.      29.1     20.4  3.71e-89     538.      652.
```

Even easier cleanup!

We could instead clean up the names and add reference rows with the `{tidycat}` package:

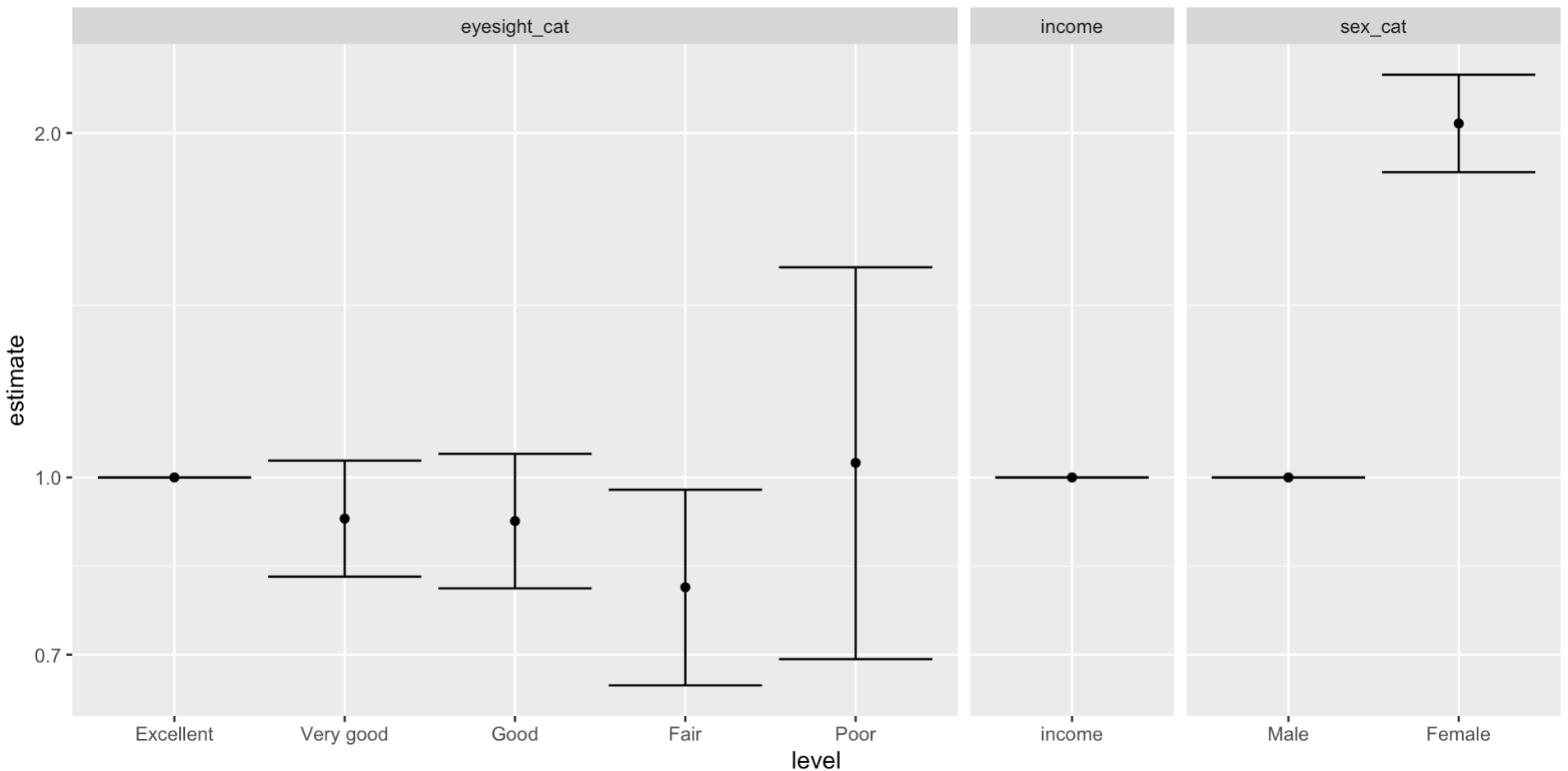
```
tidy(logistic_model, conf.int = TRUE, exponentiate = TRUE)
  tidycat::tidy_categorical(logistic_model, exponentiate =
  dplyr::select(-c(3:5))
```

```
# A tibble: 9 × 8
  term                estimate conf.low conf.high variable level effect reference
  <chr>                  <dbl>    <dbl>     <dbl> <chr>    <fct> <chr>  <chr>
1 (Intercept)            0.499    0.444     0.560 (Interc… (Int… main   Non-Base…
2 <NA>                   1        1         1     eyesigh… Exce… main   Baseline…
3 eyesight_catVery …     0.920    0.819     1.03  eyesigh… Very… main   Non-Base…
4 eyesight_catGood       0.916    0.800     1.05  eyesigh… Good  main   Non-Base…
5 eyesight_catFair       0.802    0.658     0.976 eyesigh… Fair  main   Non-Base…
6 eyesight_catPoor       1.03     0.694     1.53  eyesigh… Poor  main   Non-Base…
7 <NA>                   1        1         1     sex_cat  Male  main   Baseline…
8 sex_catFemale          2.04     1.85      2.25  sex_cat  Fema… main   Non-Base…
9 income                 1.00     1.00      1.00  income   inco… main   Non-Base…
```

This makes it easy to make forest plots, for example

```
library(ggplot2)
tidy(logistic_model, conf.int = TRUE, exponentiate = TRUE)
  tidycat::tidy_categorical(logistic_model, exponentiate =
  dplyr::slice(-1) |> # remove intercept
  ggplot(mapping = aes(x = level, y = estimate,
                       ymin = conf.low, ymax = conf.high))
  geom_point() +
  geom_errorbar() +
  facet_grid(cols = vars(variable), scales = "free", space
  scale_y_log10()
```

This makes it easy to make forest plots, for example



Exercises

1. Open the script with these examples.
2. Run it.
3. Teach yourself to use `broom::tidy()` to extract the results of the Poisson regression with robust standard errors and combine them with the results of the log-binomial regression.
4. Start creating some tables for your final project!

15:00

Intro to EPI 590R

Why this class?

About this class

Goal: Learn some best practices to make your life in R easier and your research more reproducible

- Quick! Intense!
  - It will require practice afterward, and time to sink in
  - The goal is to set you up for success and give you resources to learn more
- You don't have to use everything you learn here!
  - Some of these tools I use for *every* project, some just occasionally
  - Experiment with what works for you, a little at a time

12:26 PM Sun Nov 3
cran.r-project.org
89%
#> # origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#> # minute <dbl>, time_hour <dttm>
Add new columns with mutate()
Besides selecting sets of existing columns, it's often useful to add new columns that are functions of existing columns. This is the job of mutate():
mutate(flights,
gain = arr_delay - dep_delay,
speed = distance / air_time * 60
)
#> # A tibble: 336,776 x 21
#> year month day dep_time sched_dep_time dep_delay arr_time
#> <int> <int> <int> <int> <int> <dbl> <int>
#> 2013 1 1 517 515 830
#> 2013 1 1 533 529 850
#> 2013 1 1 542 540 923
#> 2013 1 1 544 545 1004
#> # with 336,772 more rows, and 14 more variables: sched_arr_time <int>,
#> # arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
#> # origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#> # hour <dbl>, minute <dbl>, time_hour <dttm>, gain <dbl>, speed <dbl>
dplyr::mutate() is similar to the base transform(), but allows you to refer to
mutate(flights,
gain = arr_delay - dep_delay,
gain_per_hour = gain / (air_time / 60)
)
#> # A tibble: 336,776 x 21
i would rather not behold...
BEHOLD! AN AMAZING FUNCTION!

About this class

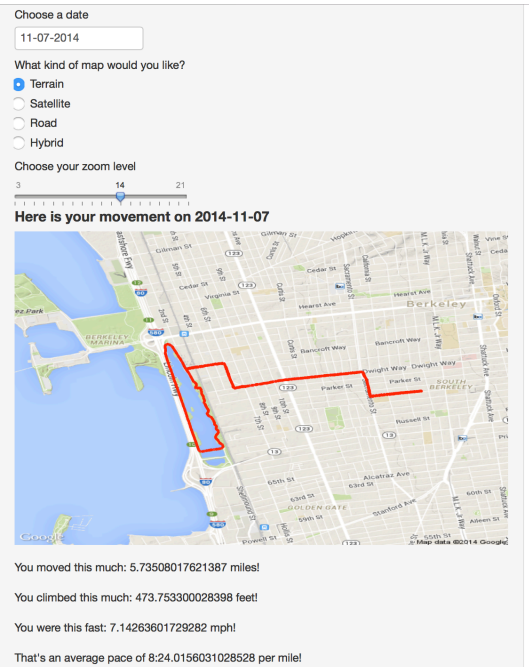
- Everything you need is at http://epi590r.louisahsmith.com
  - Canvas will link you there, but good to bookmark as well
  - The website will be up indefinitely
- General format:
  - Some slides (introduce new content and overview of exercises)
  - I'll demonstrate while you watch
  - Practice on your own/with your classmates

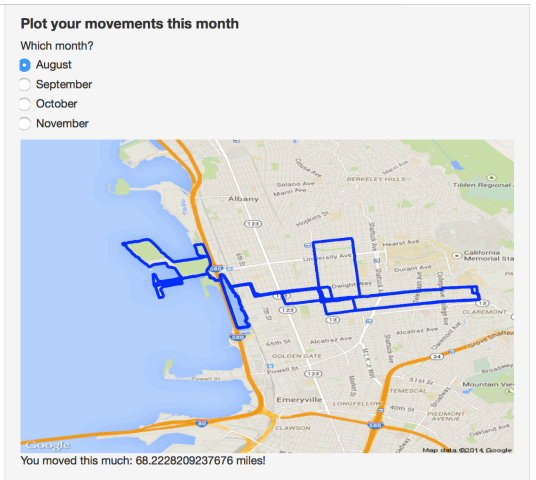
Balance

essential skill
COURSE CONTENT
ANOTHER IMPORTANT THING
ABSOLUTELY CRITICAL
CORE
@allison_horst

About Louisa

- Assistant professor at Northeastern University
  - Department of Health Sciences and the Roux Institute (Portland, Maine)
- Started using R during my master's (so just about 10 years of experience)
  - I learned mostly by doing!
  - Twitter, blogs, RStudio::conf videos, meetups
- Basically everything I do is in R!
- Actual epi research in causal inference, pregnancy, lots of other stuff





About Ollie

Why this class?

Economics | Genes | Orangutans

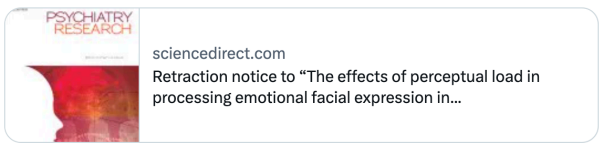
3.2 Spreadsheet coding error

In addition to these deliberate data exclusions by RR, a coding error in the RR working spreadsheet also unintentionally excludes five countries entirely (Australia, Austria, Belgium, Canada and Denmark) from all parts of the analysis.[9] The error appears in the calculations of both mean and median GDP growth with the 1946–2009 sample as well as with the mean and median GDP growth for the sample over the 220-year period 1790–2009. The omitted countries are selected alphabetically. It is clear from the spreadsheet itself that these are random exclusions. RR have since acknowledged this to be the case (RR, 2013A, 2013B, 2013C).

Errors are everywhere

Joana Grave
@joanafqg

A few months ago, I discovered an error in the database of my first paper. And today, the retraction notice is finally out!
Please don't be afraid to talk about your errors and to correct them. It's hard, but errors do happen!



sciencedirect.com
Retraction notice to “The effects of perceptual load in processing emotional facial expression in...

3:51 PM · Jul 11, 2021

24 Retweets **6** Quotes **187** Likes **2** Bookmarks

Julia Strand
@juliafstrand

I recently found a massive error in one of my published papers. The main finding was the result of a programming bug and was, in fact, completely untrue. THREAD with the very short version below, essay with the full description here:

elemental.medium.com
Scientists Make Mistakes. I Made a Big One.
A researcher learns the right thing to do when the wrong thing happens

2:16 PM · Mar 24, 2020

Leigh Senderowicz
@LSenderowicz

Hi all, today I'm writing a post that no researcher ever wants to write:

We've discovered a coding error in the analysis of an article that's already been published.

A short 📘

onlinelibrary.wiley.com
Supply-Side Versus Demand-Side Unmet Need: Implicati...
Despite its central importance to global family planning, the “unmet need for contraception” metric is frequently ...

1:20 PM · Aug 2, 2023 · **62.7K** Views

No one and no field is immune from errors in data analysis. Our goal is to make them as unlikely as possible (and report them when we find them!)

Not only improves science but also your life!

- It’s really boring to copy lots of numbers into a table
  - And then change a tiny thing in the analysis and do it all over again
- It’s really frustrating to lose work when your computer crashes, or completely change an analysis before your advisor forgets what they told you last time and has you change it back
- It’s fun when things just work! And you get more time for the fun parts of epidemiology

BEEP BEEP BEEP!!
oh just add SALT!?!!
RECIPE
@allison_horst

bless this
workflow
TOOLS
CODE
REPORT
FILES
@allison_horst

Questions?

Exercises: Connecting to GitHub

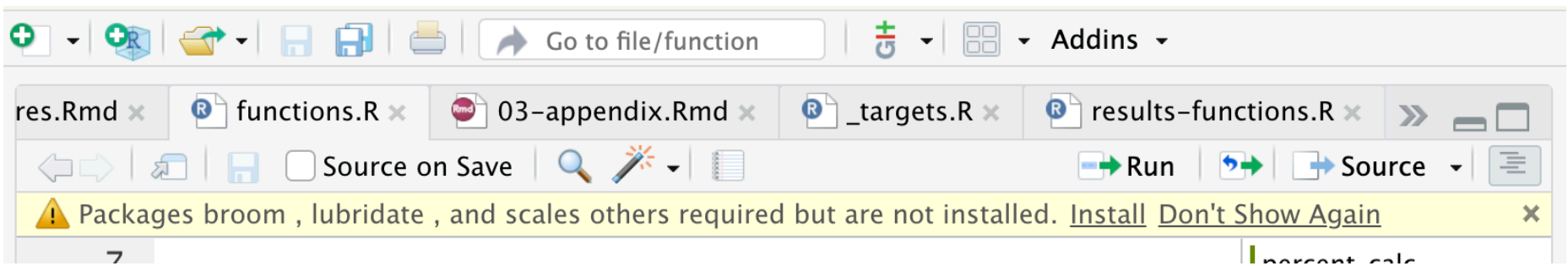
1. Install the `{usethis}` package:

```
install.packages("usethis")
```

Installing packages

If you just updated R to a new “major” version, you will need to reinstall packages

- I tend to do this as I need them rather than try to reinstall them all at once
  - RStudio tries to help!



Possible errors

Spelling the package or function's name wrong, or not installing or loading the package

```
> install.packages("blah")
Warning in install.packages :
  package ‘blah’ is not available for this version of R

A version of this package for your version of R might be available elsewhere,
see the ideas at
https://cran.r-project.org/doc/manuals/r-patched/R-admin.html#Installing-packages
> library(blah)
Error in library(blah) : there is no package called ‘blah’
> blah::blah()
Error in loadNamespace(x) : there is no package called ‘blah’
> blah()
Error in blah() : could not find function "blah"
```

Using packages: `library` vs. `::`

If you are writing a script you will save, and will use several functions from this package

```
1 library(usethis)
2 use_git_config(user.na
3                  user.em
```

If you are just running some quick code in the console or only need to use the package a few times in a script

```
1 usethis::use_git_confi
```

I try to only run `library(package)` from a script (not the console) so that there's a "record" of me loading the package, or else I might accidentally write code that doesn't work later

Set up git

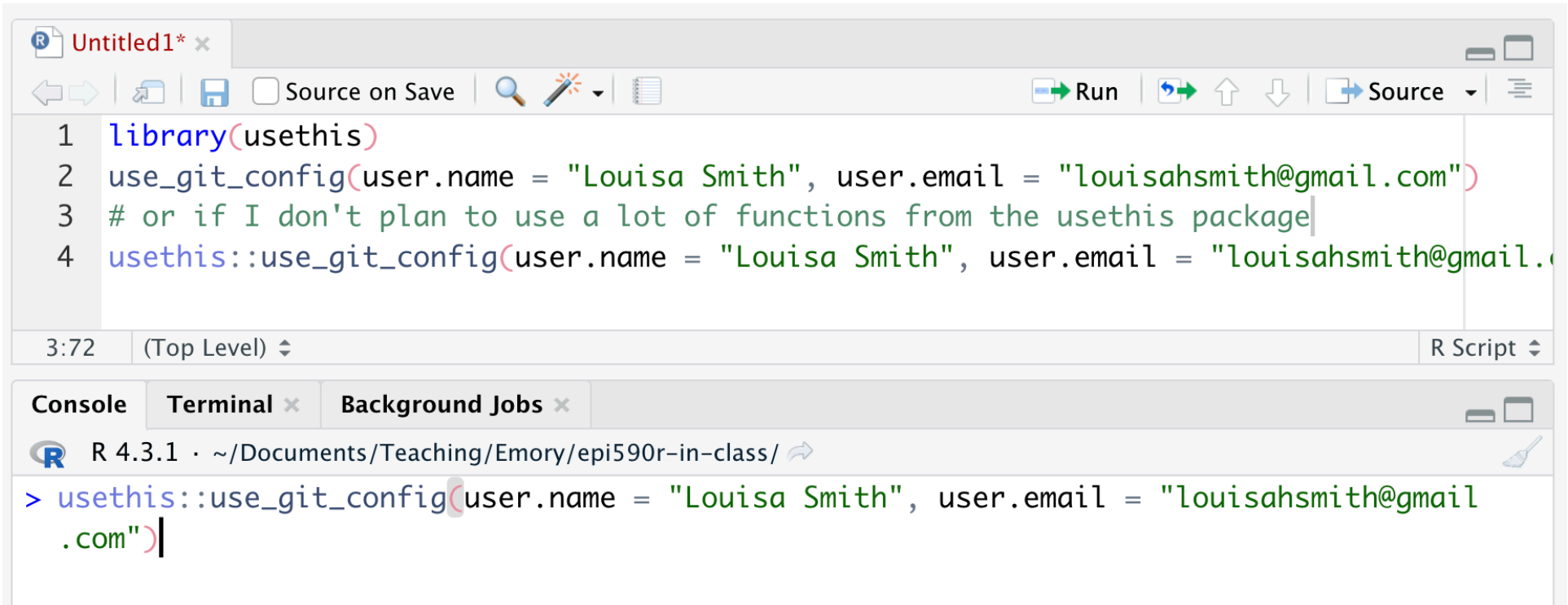
2. Introduce yourself:

```
usethis::use_git_config(user.name = "Louisa Smith",
user.email = "louisahsmith@gmail.com")
```

When you make changes to your code, they will be associated with this name and email address (this doesn't really matter for our purposes)

- You only need to do this once
- We'll explain all this in a little bit!

Since I only need to run this once, I would probably run this from the console (bottom) rather than a script (top)



Running from the console is great for `install.packages()`, quick calculations, fiddling with code until you get it right, or scenarios like this – otherwise save your code in a script!

Connect to GitHub

3. Create a github token:

```
usethis::create_github_token()
```

Instead of entering your password every time, this is a secure way to connect to GitHub

- If you are ever asked for your GitHub password in RStudio, you *have* to give this instead

Connect to GitHub

4. Copy the token
5. Back in R, run this code and paste your token in the console where it says “Enter password”:

```
gitcreds::gitcreds_set()
```

You can do this again whenever your token expires or you are using a different device

Exercises

- Refer back to the slides as needed
- Ask a classmate if you're stuck
- Raise your hand for the teaching team
- Done early? Help a friend! Read the resources section! Play around in R! Check your email!

15:00

Git and GitHub

A brief introduction

Git

- version control system
- works offline (*repositories* exist on your computer)
- tracks changes via *commits*
- has a command-line interface and integrations with GUIs (like RStudio)

GitHub

- web-based platform built around Git
- provides a remote location for hosting Git repositories
- enables collaboration
- offers other features for project management (pull requests, issue tracking)

Our Git/GitHub goals

- For **you**: Keep track of progress on projects
  - Go back when you need to
  - Don’t lose old work
  - Easily search the history of a project
- For **others**: Share your work
  - Have a place to store and link to code
  - Read and interact with others’ code

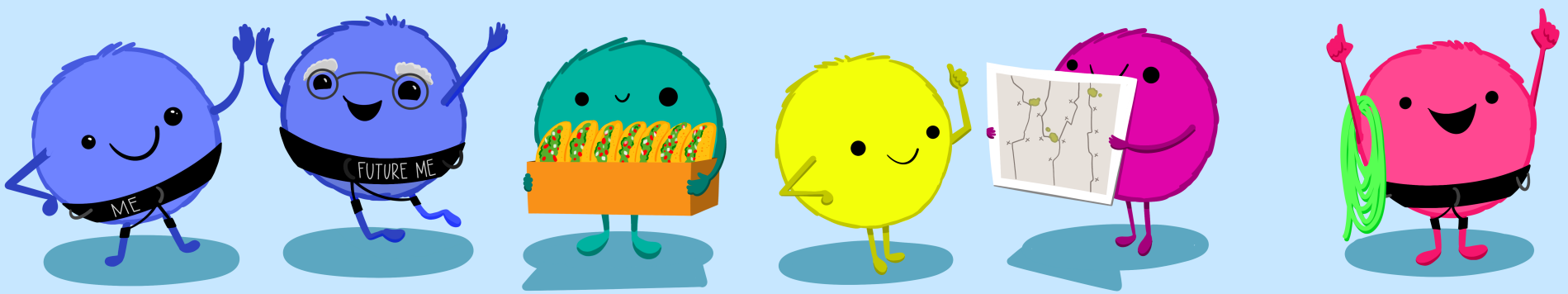
There is a ***lot*** to learn about this topic and I am ***not*** an expert on everything!

What we won't cover

- Collaboration
  - When multiple people are working on the same GitHub project, things get a little more complex
  - I went though almost my whole PhD without working on shared GitHub projects and only now do I feel semi-confident collaborating!
  - I think it's best to figure things out in your own projects first
- Git on the command line
  - There are a lot of functions you might hear about (`git fetch`, `git merge`, etc.)
  - RStudio and GitHub will have everything we need!

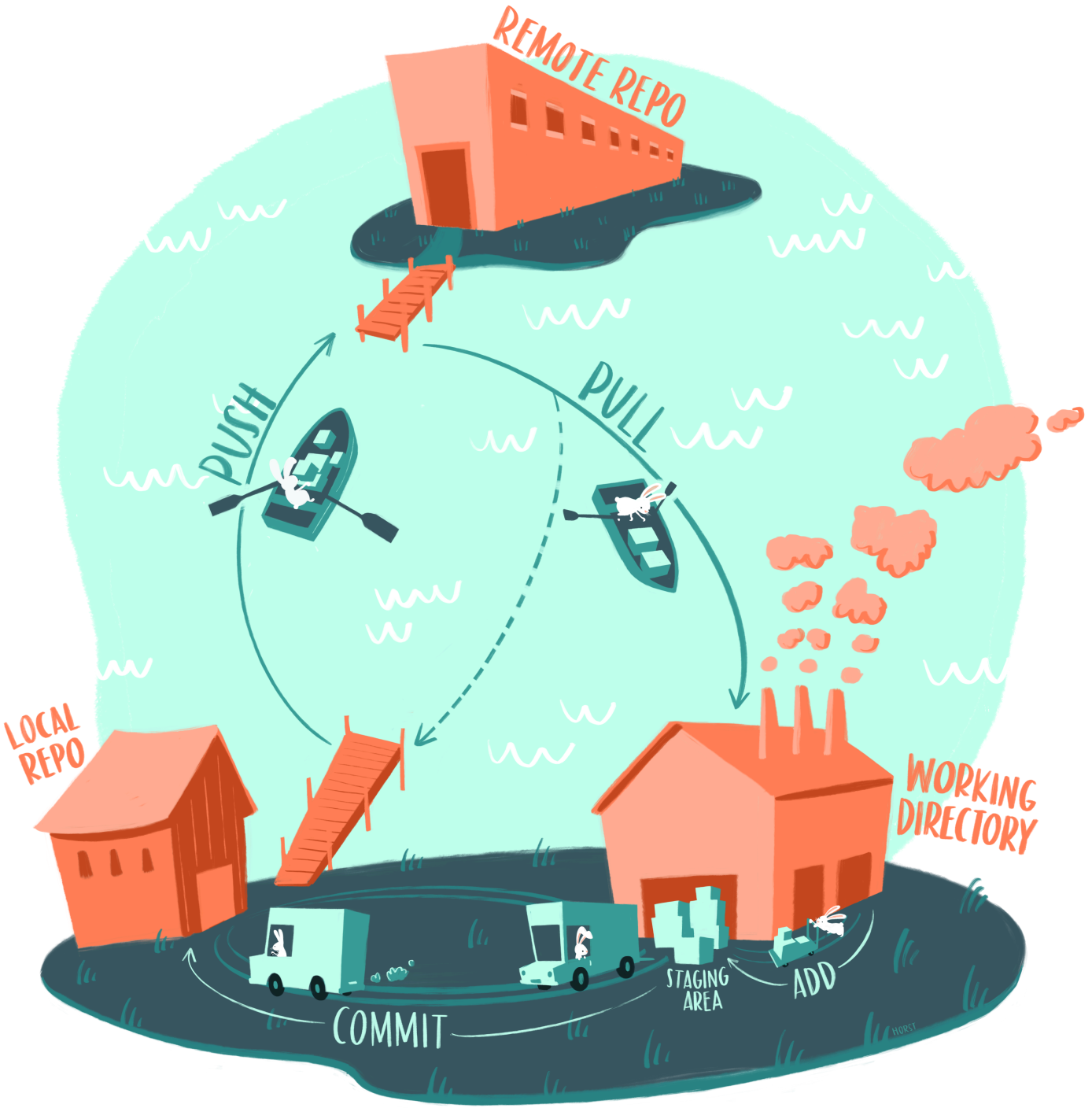
"**Collaboration is the most compelling reason to manage a project with Git and GitHub**. My definition of collaboration includes hands-on participation by multiple people, including your past and future self, as well as an asymmetric model, in which some people are active makers and others only read or review."

-JENNY BRYAN



Bryan, J. 2017. Excuse me, do you have a moment to talk about version control? PeerJ Preprints. 5:e3159v2. DOI: 10.7287/peerj.preprints.3159v2

Illustrations from the Openscapes blog **GitHub for supporting, contributing, and**


REMOTE REPO
PUSH
PULL
LOCAL REPO
WORKING DIRECTORY
STAGING AREA
ADD
COMMIT

Git

- version control system
- works offline (*repositories* exist on your computer)
- tracks changes via *commits*
- has a command-line interface and integrations with GUIs (like RStudio)

GitHub

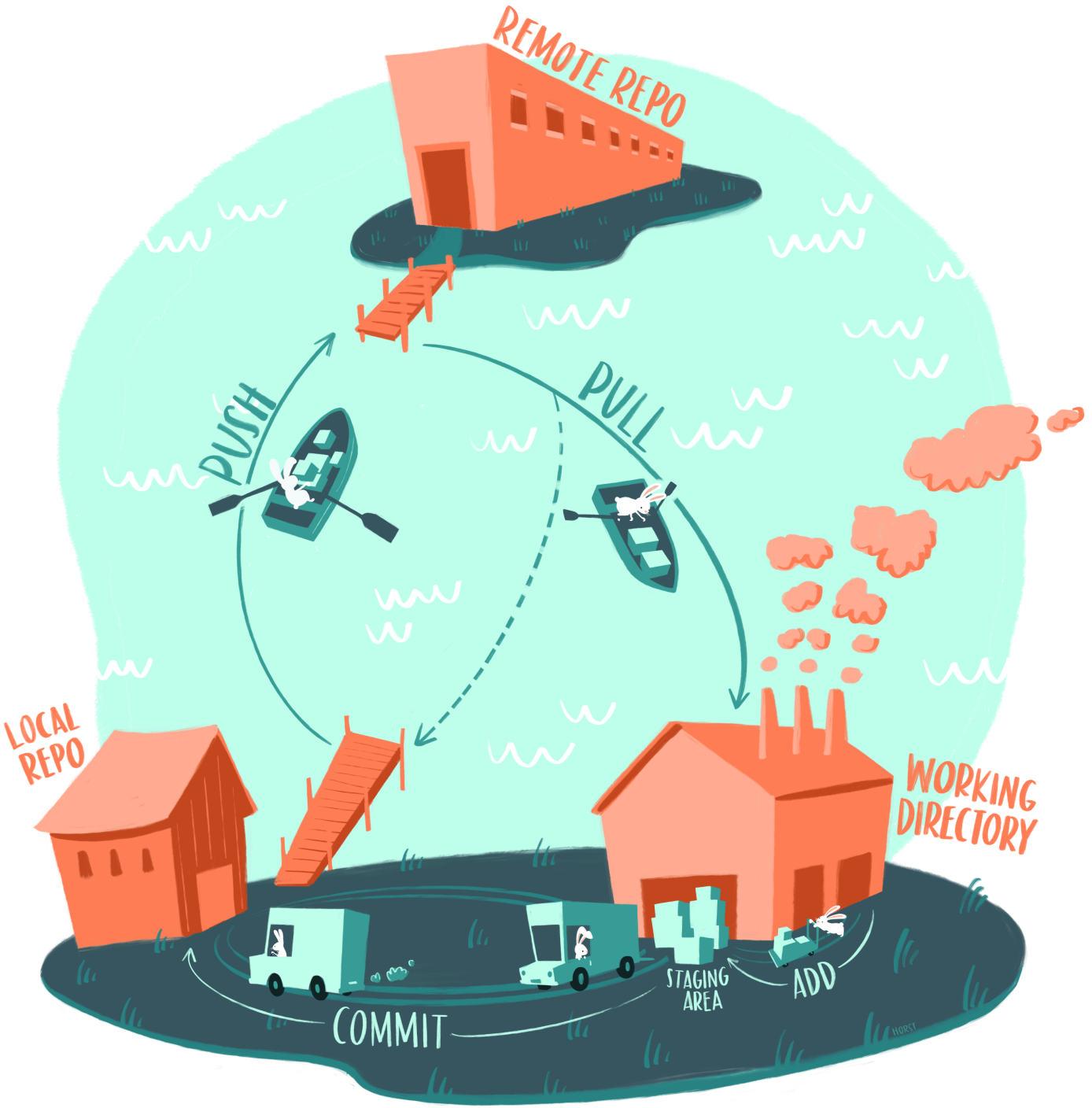
- web-based platform built around Git
- provides a remote location for hosting Git repositories
- enables collaboration
- offers other features for project management (pull requests, issue tracking)

Workflow

Create a repository (clone from GitHub, or create on your computer and connect to GitHub)

1. Write some code!
2. When you complete “something”, **add** it to the **staging** area
3. Write a brief description of what you did (“added linear model”; “created table 1”) and **commit**
4. **Push** to GitHub
5. Repeat!

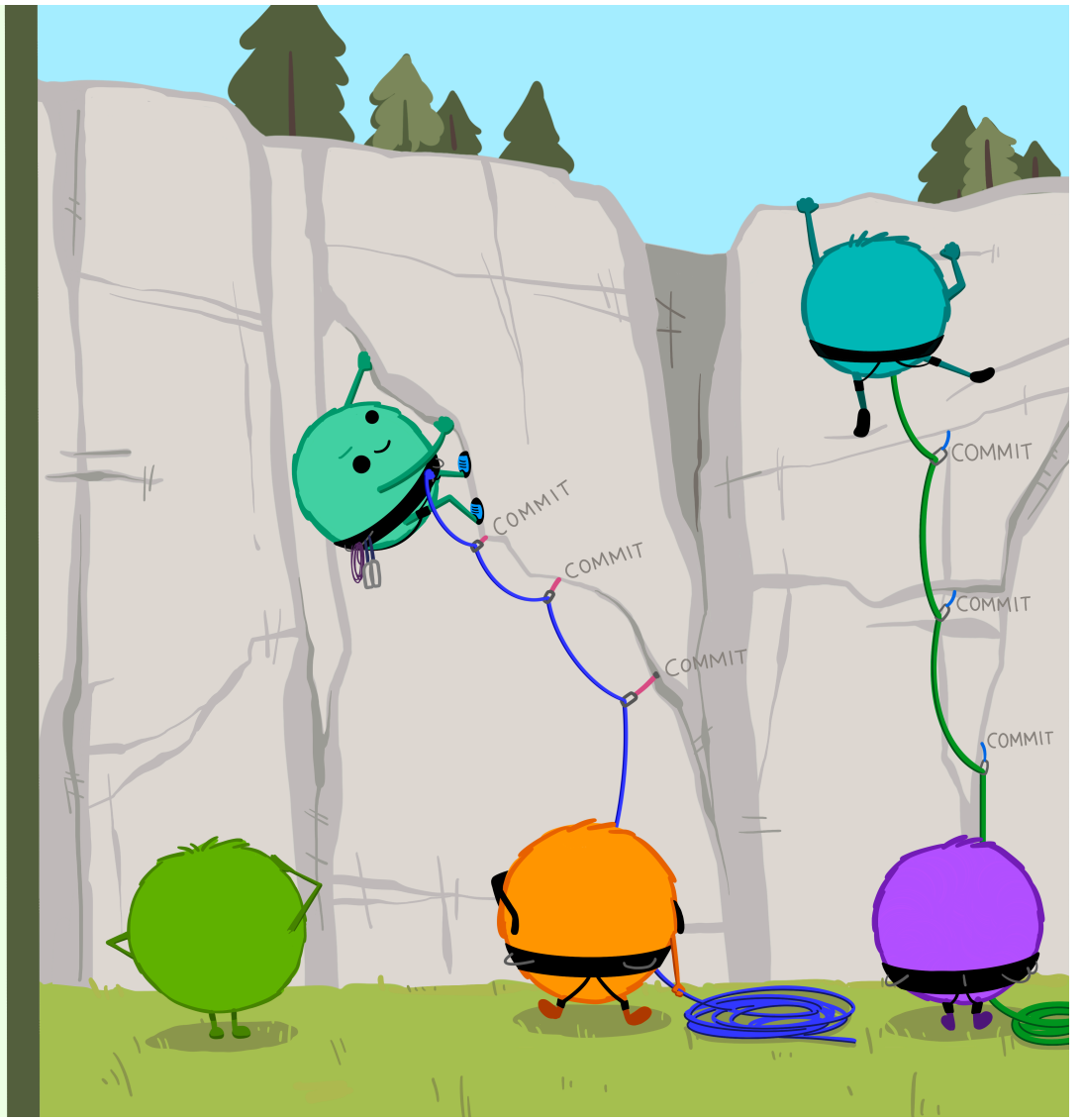
As long as you are working on your own, all on the same computer, you don’t need to worry about pulling


REMOTE REPO
PUSH
PULL
LOCAL REPO
WORKING DIRECTORY
STAGING AREA
ADD
COMMIT

What is a commit?

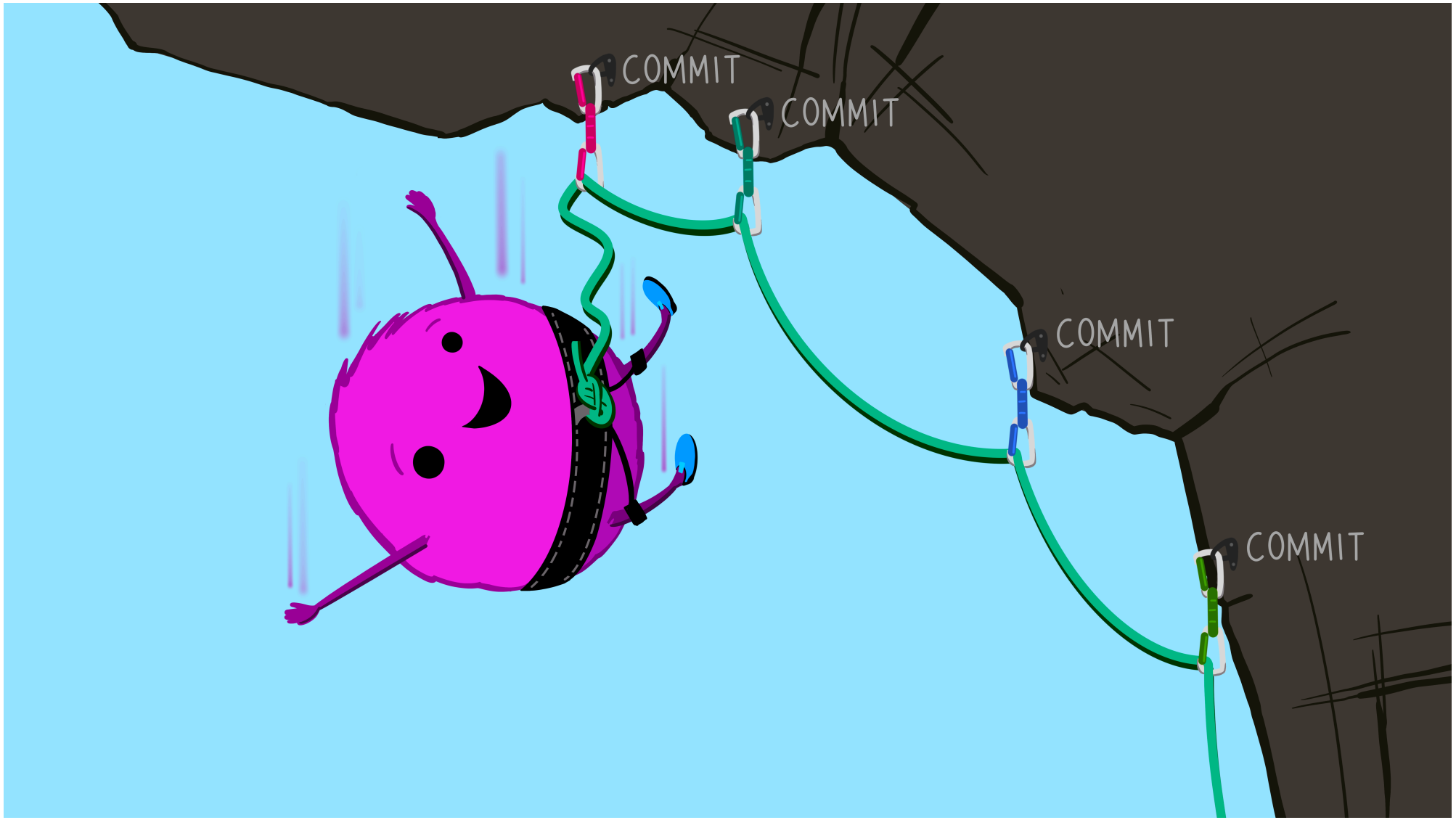
> Using a Git commit is like using anchors and other protection when climbing…**if you make a mistake, you can't fall past the previous commit**.
>
> Commits are also helpful to others, because **they show your journey, not just the destination**.
>
> – HADLEY WICKHAM & JENNY BRYAN

Wickham & Bryan, RPackages (https://r-packages.org/preface.html)



Illustrations from the Openscapes blog **GitHub for supporting, contributing, and**

What should you commit? Whatever you don’t want to lose!



Illustrations from the Openscapes blog **GitHub for supporting, contributing, and**

additional sensitivity analyses, re-render
louisahsmith committed on Mar 11, 2021
3a9a03f

figure out mice problem
louisahsmith committed on Mar 11, 2021
4a75681

tried and failed imputation
louisahsmith committed on Mar 11, 2021
21b38eb

add sensitivity analyses
louisahsmith committed on Mar 11, 2021
945ca2c

rerender text
louisahsmith committed on Mar 11, 2021
efb9476

updates to text
louisahsmith committed on Mar 11, 2021
3d6da19

render doc
louisahsmith committed on Mar 11, 2021
91ecf49

update pic
louisahsmith committed on Mar 11, 2021
16a8b21

add vertical line to pic
louisahsmith committed on Mar 11, 2021
609e76a

If you know that your code worked at 10am on October 21, 2015, and now it doesn't, you can return!

Exercises

1. *Fork* the repo (repository) at https://github.com/louisahsmith/epi590r-in-class
2. On **your** fork of the repository, click the green "Code" button. We are going to *clone* the repository to your computer using *HTTPS*.

Forking

- **Purpose:** Used to create a personal copy of another user's repository on your GitHub account.
- **Ownership:** The forked repository is still on the original owner's account, and you get your own copy to work with.
- **Collaboration:** Allows you to make changes without affecting the original repository. You can make changes, commit them to your fork, and then propose these changes to the original repository through pull requests.
- **Relationship:** The forked repository remains connected to the original, but changes aren't automatically synced.
- **Use Case:** Commonly used when you want to contribute to a project that you don't have direct write access to.

Cloning

- **Purpose:** Used to make a local copy of a GitHub repository on your computer.
- **Ownership:** You have a read-write copy on your local machine, but it's not automatically linked to your GitHub account (you can do so through RStudio).
- **Collaboration:** Allows you to work on the project locally and make changes, but these changes aren't automatically visible to others.
- **Relationship:** The cloned repository is a standalone copy, and changes won't automatically affect the original or other clones.
- **Use Case:** Useful when you want to work on a project locally and have full control over commits and pushes.

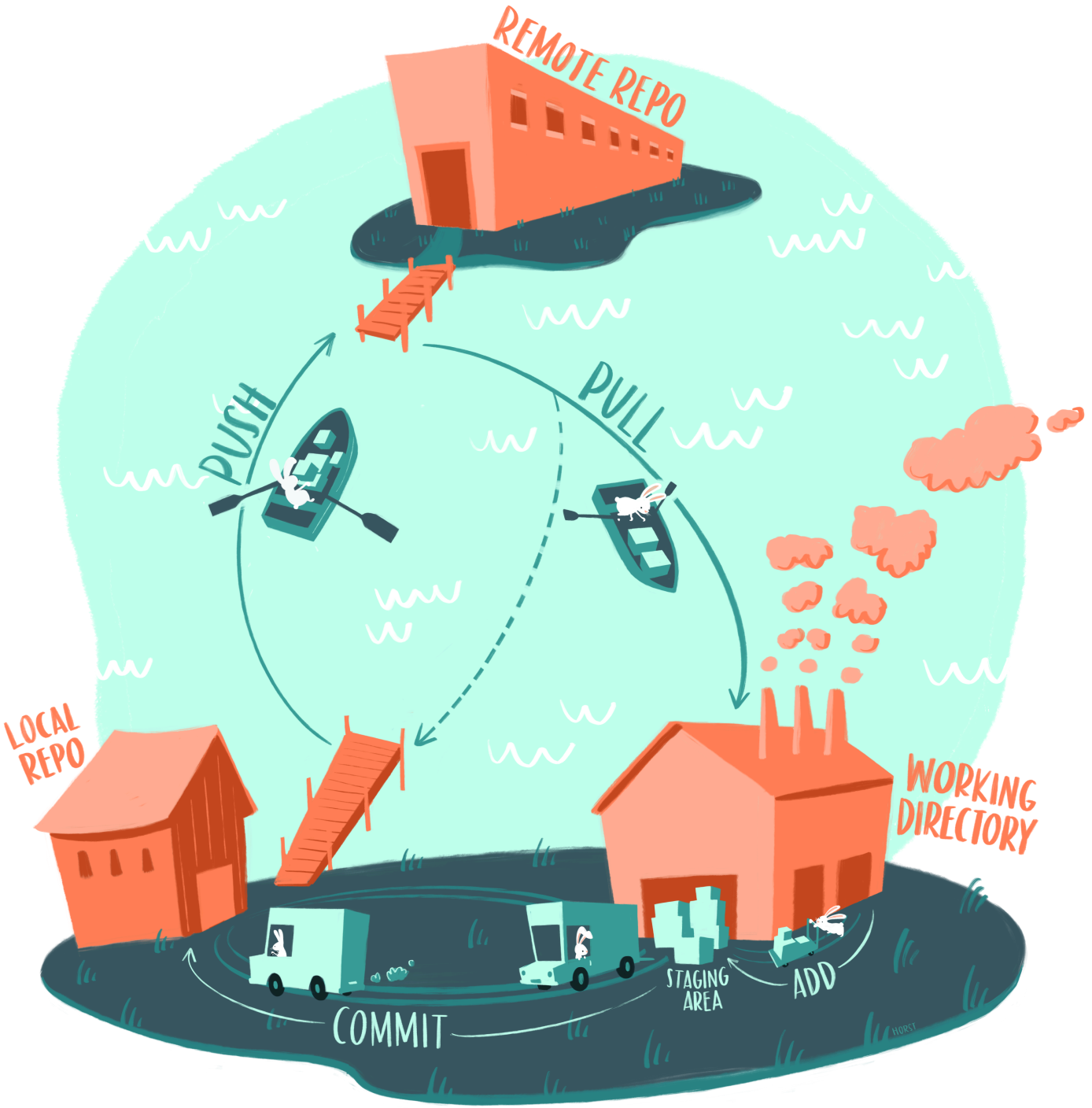
You have a forked repo on GitHub, now you are cloning that forked repo on your own computer

3. Open up RStudio.

- File > New Project > “Version Control” > “Git”

4. Paste the URL to your fork

- Name the project directory (easiest if it has the same name as the repo)
- Choose where you want to store the project (remember this spot!)
- Create Project!


REMOTE REPO
PUSH
PULL
LOCAL REPO
WORKING DIRECTORY
STAGING AREA
ADD
COMMIT

Practice making a change, staging, committing, pushing

5. From the filepane in RStudio, open `README.md`

- Change the file and save your changes

6. In your Git pane, click on the checkbox to stage the file

- Then click “Commit”

Exercise: edit, commit, push readme file

15:00

File management and projects in R

or, How to keep your computer safe from fire

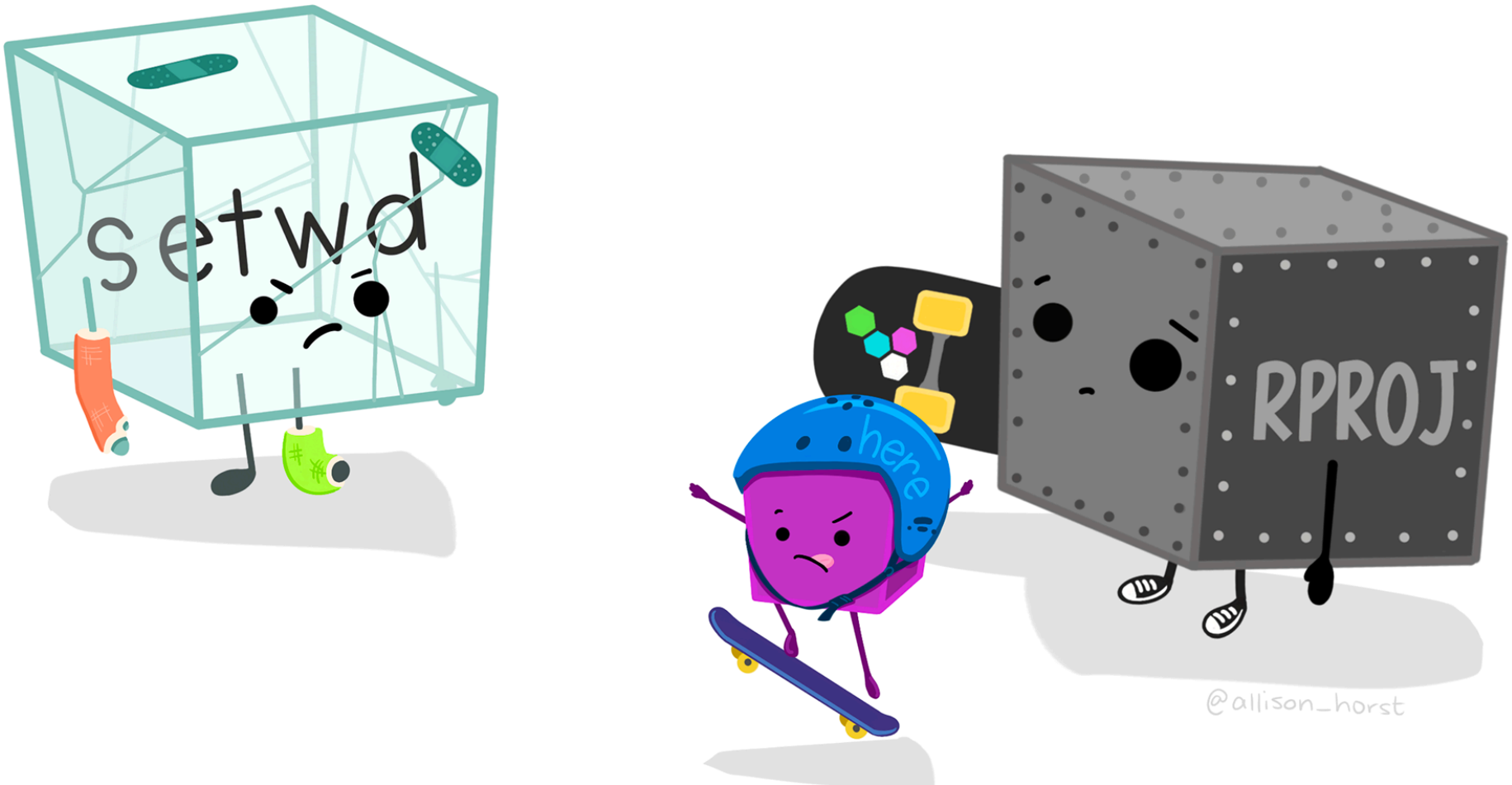
There's a famous blog post about workflows in R[1] about a talk Jenny Bryan gave that included this slide:

> If the first line of your R script is
>
> ```
> 1  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
>
> ```
> 1  rm(list = ls())
> ```
>
> I will come into your office and SET YOUR COMPUTER ON FIRE 🔥.

Instead: project-oriented workflow

- R projects provide a structured and organized way to work on projects in R
- R projects encapsulate all project-related files and settings into a single directory
- RStudio makes it easy to work with R projects

R Projects (and related tools) can prevent a lot of accidents!


setwd
here
RPROJ
@allison_horst

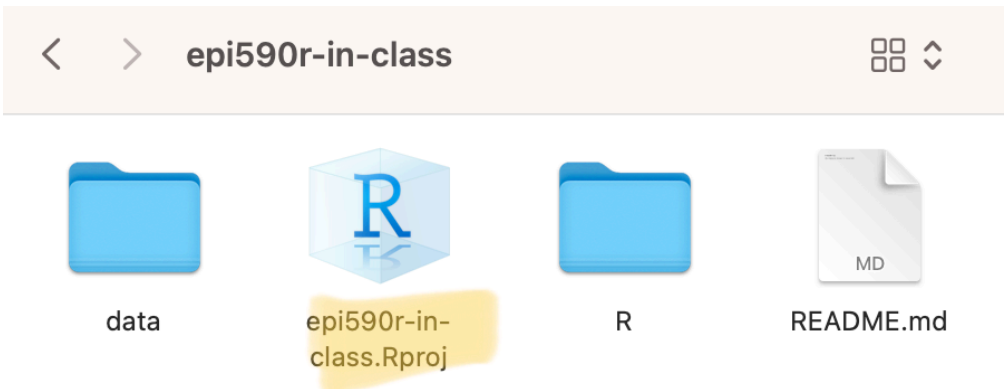
R Projects

Benefits of R Projects

1. **Isolation**: Each project has its own workspace, separate from other projects
2. **Reproducibility**: Projects ensure that code and data are self-contained and portable
3. **Collaboration**: Projects facilitate collaboration by sharing the entire project directory

Always open a project by opening the `.Rproj` file

Mac | Windows



You can have multiple projects open at once in different RStudio sessions!

You can also switch between R projects from RStudio

Creating an R Project

1. Open RStudio and go to **File > New Project**, or click on the projects button in the upper-right corner of RStudio.
2. Choose a project location (New Directory, Version Control, Existing Directory).
3. Specify the project directory (where on your computer you are storing the folder with the project) and create the project.
4. Choose the project type (e.g., regular project, R package, Shiny app, Quarto website, Bookdown book)

You already have an R project!

In the exercises, we are going to make some more changes to the repo you *forked* and *cloned*

1. Download an `.R` script and a `.csv` file from the website

- We'll be using some data from the 1979 National Longitudinal Survey of Youth

2. Find your `epi590r-in-class` repo in your file browser

- Create an `R` folder and a `data` folder
- Within the `data` folder add a `raw` and a `clean` folder.
- Put the `.csv` file in the `data/raw` folder and the script in `R` folder.

File structure goal

```
epi590r-in-class/
├─ epi590r-in-class.Rproj
├─ README.md
├─ R/
│   └─ clean-data-bad.R
├─ data/
│   ├─ raw/
│   │   └─ nlsy.csv
│   └─ clean/
```

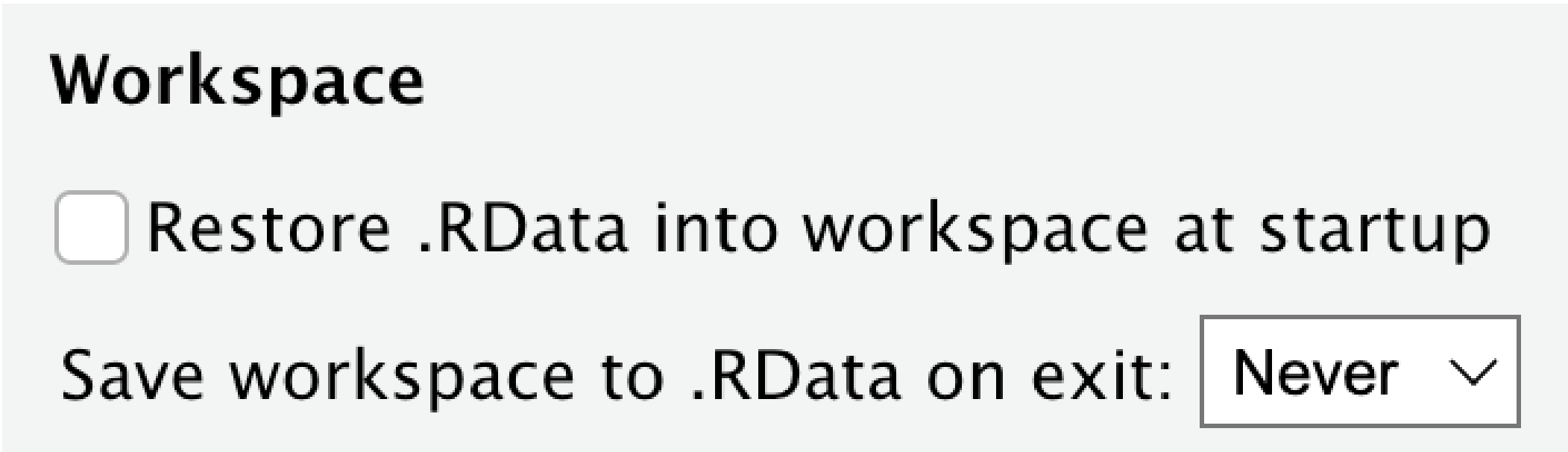
Exercises, cont.

3. Return to RStudio. If you closed RStudio, make sure you re-open this project. Look to the filepane to confirm the files are there.
4. Stage, commit, and push the changes you've made.
5. Try to run the code, line-by-line, in `clean-data-bad.R`.

- As you're running it, try to think of changes you might make

Stop for a settings change!

6. Tell RStudio to start fresh whenever you start a new session



7. Close RStudio, then open it up again by opening the `epi590r-in-class.Rproj` file in your file browser

Exercise: work with files in your R project

15:00

The {here} package

Fire safety, continued

The problem with `setwd()`

- `setwd()` changes the working directory, leading to potential issues in collaboration and reproducibility
  - You and I don't have the same file structure!
  - For example, my current working directory is

```
1  getwd()
```

```
[1] "/Users/l.smith/Documents/Teaching/Emory/epi590r-2024"
```

- It's also really annoying to change your working directory when you move around files and folders, even if it's just you using them

Do you think this code from 2015 still works?

Source on Save

```
1  #Create results table for poster
2  rm(list=ls())
3  setwd("~/Box Sync/Behavior Outcomes/")
4
5  #Load data
6  load("./Data/behavior_datasets.Rdata")
7
```

Referring to files with the here package

```
source(here::here("R", "functions.R"))

dat <- read_csv(here::here("data", "raw", "data.csv"))

p <- ggplot(dat) + geom_point(aes(x, y))

ggsave(plot = p,
       filename = here::here("results", "figures", "fig.pd
```

You can also separate the file paths with `/`:

```
dat <- read_csv(here::here("data/raw/data.csv"))
```

How it works

- Construct file paths with reference to the top directory holding your `.Rproj` file.
- `here::here("data", "raw", "data.csv")` for me, here, becomes `"/Users/l.smith/Documents/Teaching/Emory/epi590r-2024/data/raw/data.csv"`
  - If I change my working directory to somewhere else within my project, it will still give me that path
- And if I send you my code to run, it will become whatever file path *you* need it to be, as long as you're running it within the R Project.

Referring to the here package

```
here::here()
```

is equivalent to

```
library(here)
here()
```

I just prefer to write out the package name whenever I need it, but you can load the package for your entire session if you want.

Exercises

1. Install the `{here}` package:

   `install.packages("here")`

2. Make sure you're in your in-class R project! In the console, run:

   `here::here()` to print your project directory
   `getwd()` to print your working directory

   What do you notice?

3. In the console, run:

   `setwd("data")` to set your working directory

   Then run the same lines as above. What do you notice?

Exercises, cont.

4. Download the next `.R` script from the website and use your file browser to put it in the R folder in your project.

- Run through the code line-by-line.
- Compare it with the code from the last section.

15:00

Creating new projects

Starting from scratch

EPI 590R final project

Your goal will be to create an analysis that

- I can reproduce on my own computer
- Is easy to rerun if I tell you, for example, to remove the 12th row of your dataset

We'll start this in class!

New projects

We cloned our first project from GitHub; now we are going to start a new project from scratch

1. File > New Project > New Directory > New Project

- If you ever want to convert an existing folder that holds an analysis into an R project, you can choose “Existing Directory”
- You’ll also see other options besides “New Project” – an R package, a Shiny app, etc.
  - These will get you set up with some initial files for these types of projects
  - You can also make a template of your own!

New projects

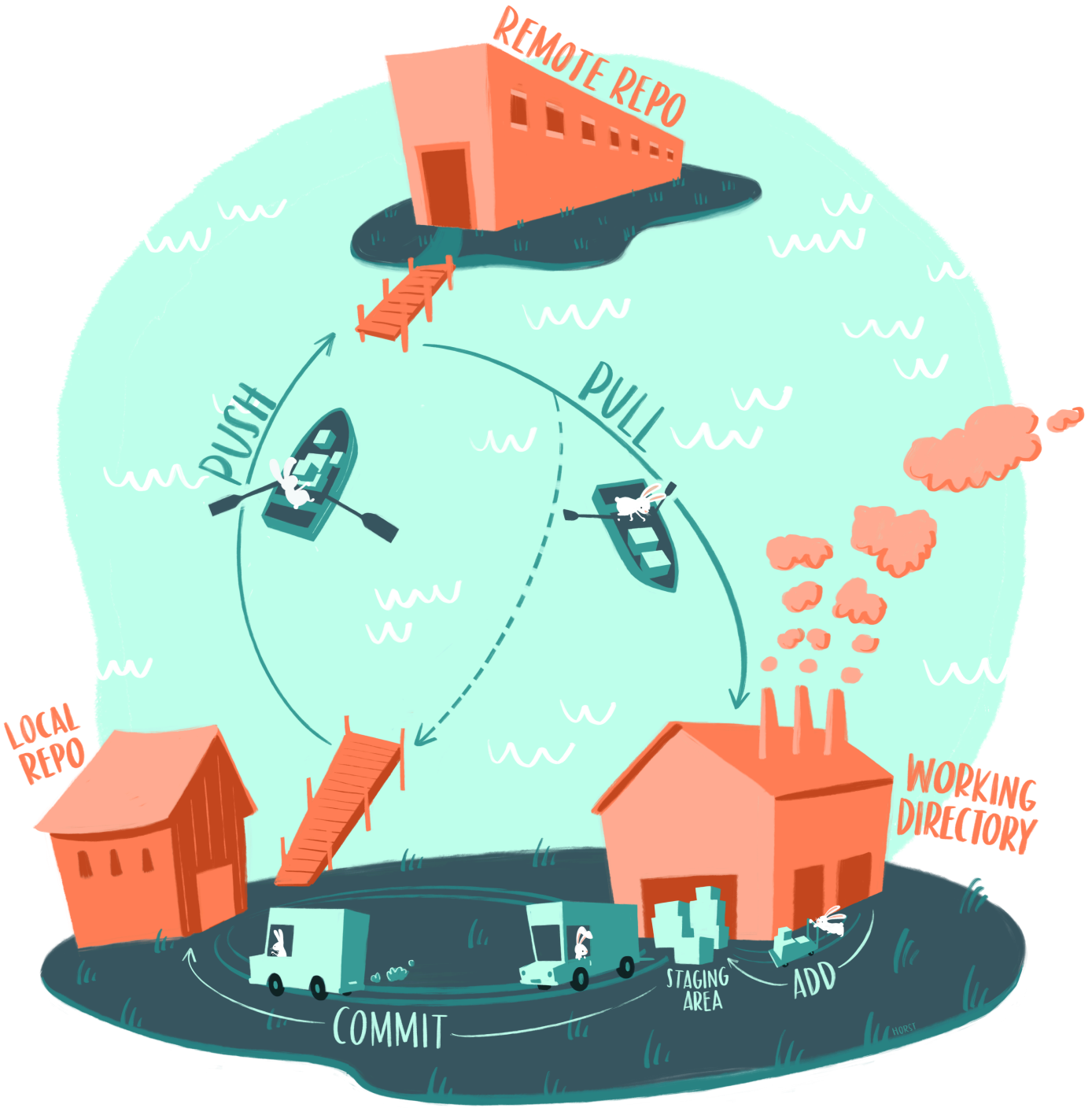
2. Choose a name for your new project and where to store it on your computer

- Check “Create a git repository”
  - This gets you all set to connect to GitHub and creates a `.gitignore` file
- You can leave “Use renv with this project” unchecked (we’ll be introducing the `{renv}` package later!)

Initial Git commit

3. Stage and commit the files

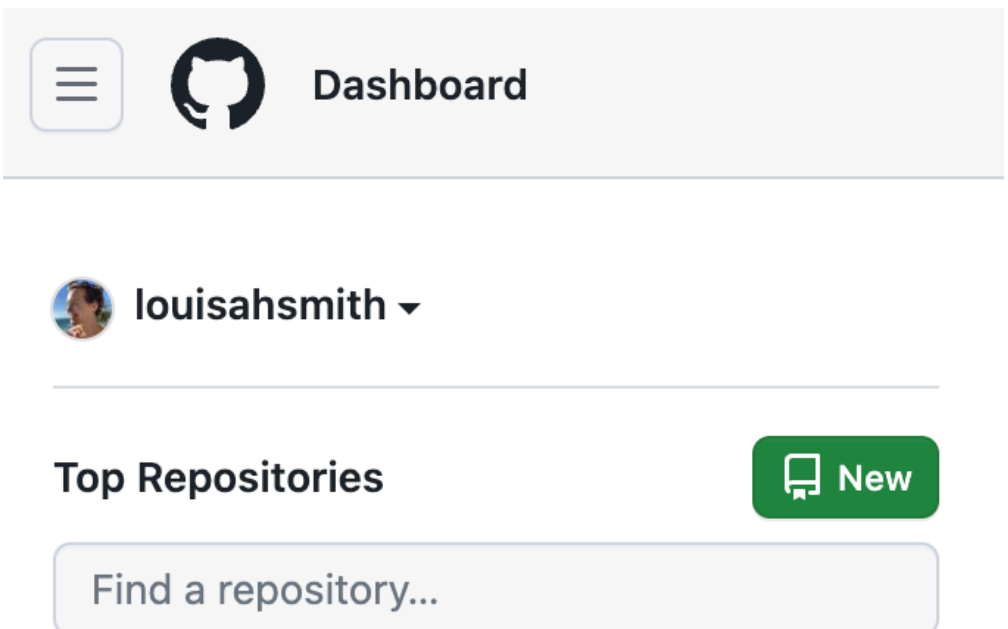
- I usually use “initial commit” as my first commit message since I haven’t don’t anything yet!
- We can’t yet push because we haven’t connected to a remote repository


REMOTE REPO
PUSH
PULL
LOCAL REPO
WORKING DIRECTORY
STAGING AREA
ADD
COMMIT

Creating a new repo on GitHub

4. Open up your web browser to GitHub and make a new repository

From github.com | From github.com/username



Repository options

5. Choose a name (preferably one that matches the name you gave your R project).

- You can choose to make it private, if you wish
  - Private repos have fewer features *unless* you have GitHub Pro (which you can get for free as a student with the GitHub student developer pack!)
- You don't need to click anything else

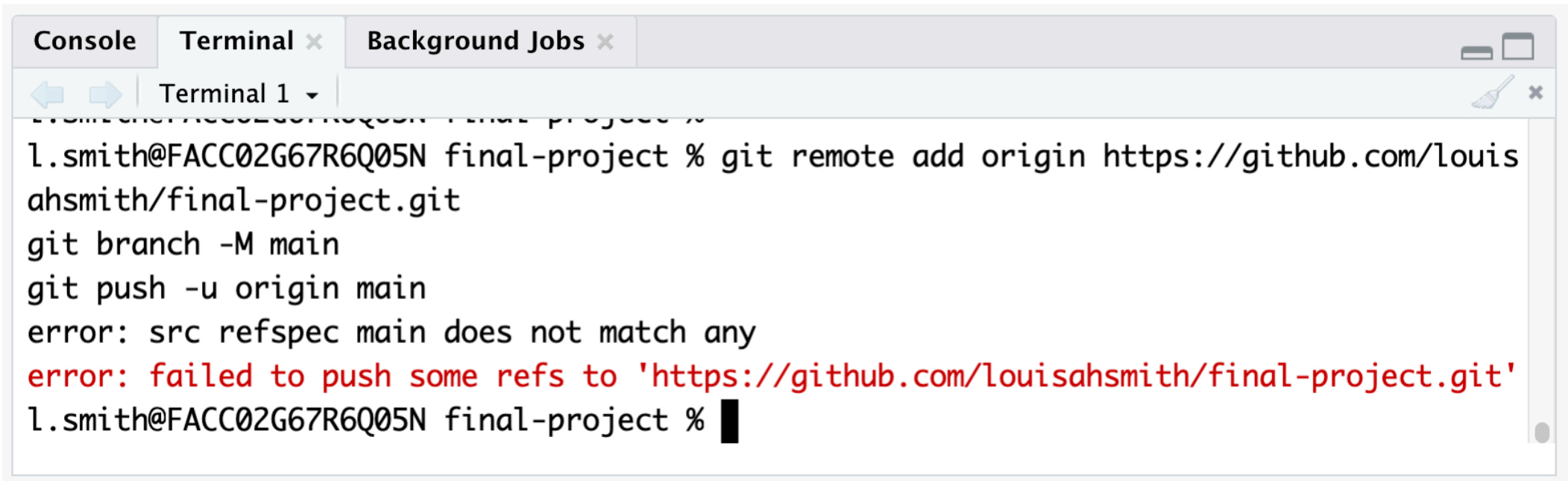
Connect the local to the remote

- You created your local repo with RStudio in a directory you chose
- Now you need to connect it to the remote repo on GitHub

6. Copy the code from the second section: “push an existing repository from the command line” in the *terminal* within RStudio.

Connect the local to the remote

7. Run the three lines of code *one at a time*, then refresh your GitHub page!



`.gitignore`

You likely don't want to push everything to GitHub, even if you have a private repository

- Be especially careful about data and passwords
- You also can't push very large files (>100 mb)

A `.gitignore` is a special text file that tells Git not to track certain files

- RStudio starts you off with a few entries, including `.Rhistory` since no one needs to see everything you've run in R!

`.gitignore` exercises

8. Create a new file called `secrets.txt` within this new repo

- Write down your deepest, darkest secrets and save (e.g., I am scared of spiders)

9. Open `.gitignore` via the RStudio filepane

- Add “secrets.txt” below the files that RStudio helpfully ignored for you
- Save

Keep your eye on the Git pane!

Starting the final project

10. Set up your folders how you'd like in your repo (you can always change this)

- Find some data, download it, and store it in your repo
- Commit and push to GitHub!

For your final project, your data must be something that can be stored online and accessed by me.

Some fun options for data are:

- https://data.fivethirtyeight.com/
- https://github.com/rfordatascience/tidytuesday#datasets
- https://github.com/higgi13425/medicaldata/tree/master/data (descriptions: https://higgi13425.github.io/medicaldata/#list-of-datasets)

Exercises

Get started making a new project and GitHub repo for your final project, editing the `.gitignore` file, and finding some fun data

You can always change anything you want later, and even delete the whole thing and start fresh!

15:00

Descriptive tables with {gtsummary}

Make an easy Table 1

What is `{gtsummary}`?

- Create tables that are publication-ready
- Highly customizable
- Descriptive tables, regression tables, etc.

gtsummary::tbl_summary()

```
1 library(gtsummary)
2
3 tbl_summary(
4   nlsy,
5   by = sex_cat,
6   include = c(sex_cat,
7               eyesight
```

Characteristic	**Male** N = 6,403 [1]	**Female** N = 6,283 [1]
race_eth_cat		
Hispanic	1,000 (16%)	1,002 (16%)
Black	1,613 (25%)	1,561 (25%)
Non-Black, Non-Hispanic	3,790 (59%)	3,720 (59%)
region_cat		
Northeast	1,296 (21%)	1,254 (20%)
North Central	1,488 (24%)	1,446 (23%)
South	2,251 (36%)	2,317 (38%)
West	1,253 (20%)	1,142 (19%)
Unknown	115	124
eyesight_cat		
Excellent	1,582 (38%)	1,334 (31%)
Very good	1,470 (35%)	1,500 (35%)
Good	792 (19%)	1,002 (23%)

[1] n (%); Median (Q1, Q3)

Characteristic	Male N = 6,403 [1]	Female N = 6,283 [1]
Fair	267 (6.4%)	365 (8.5%)
Poor	47 (1.1%)	85 (2.0%)
Unknown	2,245	1,997
glasses	1,566 (38%)	2,328 (54%)
Unknown	2,241	1,995
age_bir	25 (21, 29)	22 (19, 27)
Unknown	3,652	3,091

[1] n (%); Median (Q1, Q3)

You can also refer to variables using helper functions

```
1 library(gtsummary)
2
3 tbl_summary(
4   nlsy,
5   by = sex_cat,
6   include = c(ends_wit
```

Characteristic	Male N = 6,403 [1]	Female N = 6,283 [1]
region_cat		
Northeast	1,296 (21%)	1,254 (20%)
North Central	1,488 (24%)	1,446 (23%)
South	2,251 (36%)	2,317 (38%)
West	1,253 (20%)	1,142 (19%)
Unknown	115	124
race_eth_cat		
Hispanic	1,000 (16%)	1,002 (16%)
Black	1,613 (25%)	1,561 (25%)
Non-Black, Non-Hispanic	3,790 (59%)	3,720 (59%)
eyesight_cat		
Excellent	1,582 (38%)	1,334 (31%)

[1] n (%); Median (Q1, Q3)

Characteristic	Male N = 6,403 [1]	Female N = 6,283 [1]
Very good	1,470 (35%)	1,500 (35%)
Good	792 (19%)	1,002 (23%)
Fair	267 (6.4%)	365 (8.5%)
Poor	47 (1.1%)	85 (2.0%)
Unknown	2,245	1,997
glasses	1,566 (38%)	2,328 (54%)
Unknown	2,241	1,995
age_bir	25 (21, 29)	22 (19, 27)
Unknown	3,652	3,091

[1] n (%); Median (Q1, Q3)

We probably want to name the variables

```r
tbl_summary(
  nlsy,
  by = sex_cat,
  include = c(sex_cat, race_eth_cat,
              eyesight_cat, glasses,
  label = list(
    race_eth_cat ~ "Race/ethnicity",
    region_cat ~ "Region",
    eyesight_cat ~ "Eyesight",
    glasses ~ "Wears glasses",
    age_bir ~ "Age at first birth"
  ),
  missing_text = "Missing")
```

Characteristic	**Male** N = 6,403 [1]	**Female** N = 6,283 [1]
Race/ethnicity		
Hispanic	1,000 (16%)	1,002 (16%)
Black	1,613 (25%)	1,561 (25%)
Non-Black, Non-Hispanic	3,790 (59%)	3,720 (59%)
Region		
Northeast	1,296 (21%)	1,254 (20%)
North Central	1,488 (24%)	1,446 (23%)
South	2,251 (36%)	2,317 (38%)
West	1,253 (20%)	1,142 (19%)
Missing	115	124
Eyesight		
Excellent	1,582 (38%)	1,334 (31%)
Very good	1,470 (35%)	1,500 (35%)
Good	792 (19%)	1,002 (23%)

[1] n (%); Median (Q1, Q3)

Characteristic	Male N = 6,403 [1]	Female N = 6,283 [1]
Fair	267 (6.4%)	365 (8.5%)
Poor	47 (1.1%)	85 (2.0%)
Missing	2,245	1,997
Wears glasses	1,566 (38%)	2,328 (54%)
Missing	2,241	1,995
Age at first birth	25 (21, 29)	22 (19, 27)
Missing	3,652	3,091

[1] n (%); Median (Q1, Q3)

And do a million other things

```
tbl_summary(
  nlsy,
  by = sex_cat,
  include = c(sex_cat, race_eth_cat,
              eyesight_cat, glasses,
  label = list(
    race_eth_cat ~ "Race/ethnicity",
    eyesight_cat ~ "Eyesight",
    glasses ~ "Wears glasses",
    age_bir ~ "Age at first birth"
  ),
  missing_text = "Missing") |>
  add_p(test = list(all_continuous()
                    all_categorical(
  add_overall(col_label = "**Total**
  bold_labels() |>
  modify_footnote(update = everythin
  modify_header(label = "**Variable*
```

Variable	Total	Male N = 6,403	Female N = 6,283	P
Race/ethnicity				
Hispanic	2,002 (16%)	1,000 (16%)	1,002 (16%)	
Black	3,174 (25%)	1,613 (25%)	1,561 (25%)	
Non-Black, Non-Hispanic	7,510 (59%)	3,790 (59%)	3,720 (59%)	
Eyesight				
Excellent	2,916 (35%)	1,582 (38%)	1,334 (31%)	
Very good	2,970 (35%)	1,470 (35%)	1,500 (35%)	
Good	1,794 (21%)	792 (19%)	1,002 (23%)	
Fair	632 (7.5%)	267 (6.4%)	365 (8.5%)	
Poor	132 (1.6%)	47 (1.1%)	85 (2.0%)	
Missing	4,242	2,245	1,997	
Wears glasses	3,894 (46%)	1,566 (38%)	2,328 (54%)	
Missing	4,236	2,241	1,995	

Additional arguments

We saw `include =`, `by =`, `label =`, `missing_text =` in the example

`statistic =`:

- The default is `list(all_continuous() ~ "{median} ({p25}, {p75})", all_categorical() ~ "{n} ({p}%)")`
- For categorical variables, you can use `{n}` (frequency), `{N}` (denominator), `{p}` formatted percentage
- For continuous variables, you can use `{median}`, `{mean}`, `{sd}`, `{var}`, `{min}`, `{max}`, `{sum}`, `{p##}` (any percentile), or any function `{foo}`
- You can refer to individual variables with their names: `list(age ~ "min = {min}; max = {max}")`

Additional arguments

`digits =`:

- It will do its best to guess the appropriate number of digits
- Otherwise, you can pass a function:
  - `digits = everything() ~ style_sigfig`
- Or a value for each statistic shown
  - `statistic = list(age ~ "min = {min}; max = {max}", year_of_birth = "{median} ({p25}, {p75})")`:
  - `digits = list(age ~ c(1, 1) year_of_birth ~ c(0, 0, 0))`

Additional arguments

`type =`:

- One of “continuous”, “continuous2”, “categorical”, “dichotomous”
  - If a variable only has 0/1, TRUE/FALSE, or yes/no values, it will be treated as dichotomous
    - You can override this with `type = list(``varname``~ "categorical")`
    - Dichotomous variables only show one row (i.e., the percentage of 1’s) unless you change to categorical
      - You can change which level to show with `value = list(varname ~ "level to show")`
  - “continuous2” variables can have multiple rows of statistics

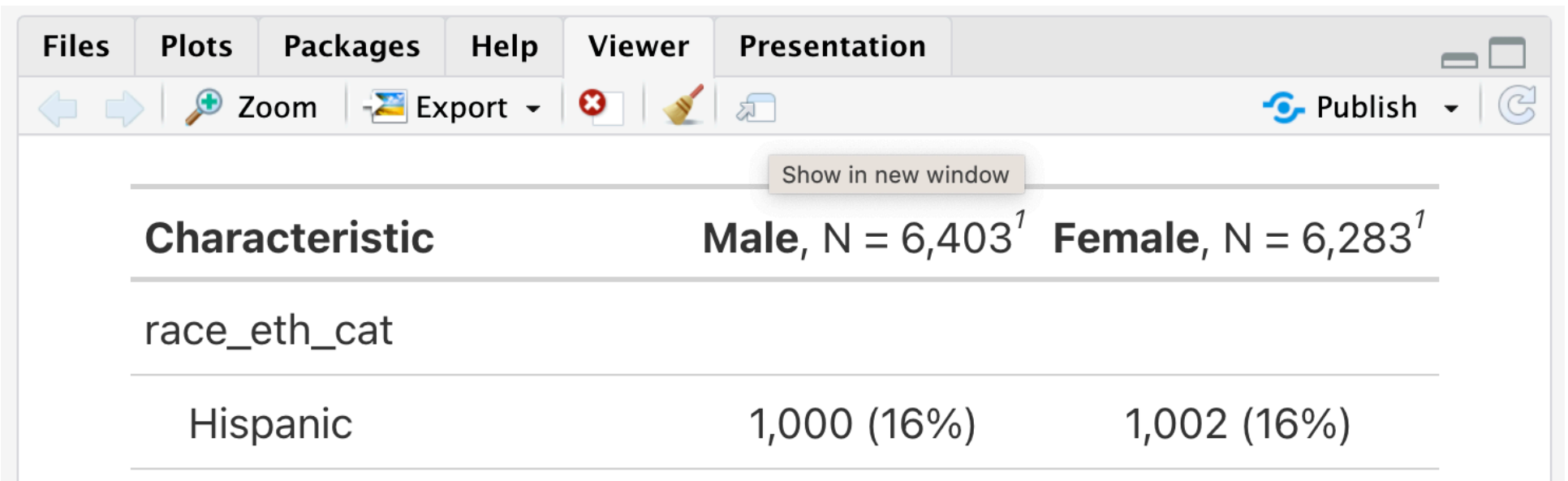
`missing =`:

Additional functions

- `add_overall()`: In a stratified table, add a column for all strata combined
- `bold_labels()`: Bold the variable names (also `bold_levels()`)
- `add_p()`: Add a p-value (required by some journals 🤷‍♀️)
- `modify_footnote(update = everything() ~ NA)`: Remove the footnotes (can also add footnotes!)
- `modify_header()`: Change the header column

```
tbl_summary()
```

- Incredibly customizeable
  - So many options can be overwhelming
  - The FAQ/gallery is an incredible resource
- To save, I often just view in the web browser and copy and paste into a Word document
  - Can also be used within quarto/R Markdown



Exercises

1. Open the script with some examples.
2. Install `{gtsummary}` and run the examples.

3-7. You're on your own! Work with your neighbors, and we'll come back together to go over these.

Extra time? Start a table using the data you downloaded for your final project! Make sure you switch to that R project!

15:00

Regression tables with {gtsummary}

On to Table 2!

Univariate regressions

Fit a series of univariate regressions of income on other variables.

```
1 tbl_uvregression(
2   nlsy,
3   y = income,
4   include = c(sex_cat,
5               eyesight
6   method = lm)
```

Characteristic	N	Beta	95% CI [1]	p-value
sex_cat	10,195			
Male		—	—	
Female		-358	-844, 128	0.15
race_eth_cat	10,195			
Hispanic		—	—	
Black		-1,747	-2,507, -988	<0.001
Non-Black, Non-Hispanic		3,863	3,195, 4,530	<0.001
eyesight_cat	6,789			
Excellent		—	—	
Very good		-578	-1,319, 162	0.13

[1] CI = Confidence Interval

Characteristic	N	Beta	95% CI [1]	p-value
Good		-1,863	-2,719, -1,006	<0.001
Fair		-4,674	-5,910, -3,439	<0.001
Poor		-6,647	-9,154, -4,140	<0.001
age_bir	4,773	595	538, 652	<0.001

[1] CI = Confidence Interval

Can also do logistic regression

```
tbl_uvregression(
  nlsy,
  y = glasses,
  include = c(sex_cat,
              eyesight
  method = glm,
  method.args = list(f
  exponentiate = TRUE)
```

Characteristic	N	OR [1]	95% CI [1]	p-value
sex_cat	8,450			
Male		—	—	
Female		1.97	1.81, 2.15	<0.001
race_eth_cat	8,450			
Hispanic		—	—	
Black		0.76	0.67, 0.86	<0.001
Non-Black, Non-Hispanic		1.34	1.19, 1.50	<0.001
eyesight_cat	8,444			
Excellent		—	—	
Very good		0.93	0.84, 1.03	0.2
Good		0.95	0.84, 1.07	0.4
Fair		0.81	0.68, 0.96	0.016
Poor		1.15	0.81, 1.63	0.4
age_bir	5,813	1.02	1.01, 1.03	<0.001

[1] OR = Odds Ratio, CI = Confidence Interval

We probably want to do some multivariable regressions

```
linear_model <- lm(income ~ sex_cat + age_bir + race_eth_c
                   data = nlsy)
```

```
linear_model_int <- lm(income ~ sex_cat*age_bir + race_eth
                   data = nlsy)
```

```
logistic_model <- glm(glasses ~ eyesight_cat + sex_cat + i
                      data = nlsy, family = binomial())
```

gtsummary::tbl_regression()

```
tbl_regression(
  linear_model,
  intercept = TRUE,
  label = list(
    sex_cat ~ "Sex",
    race_eth_cat ~ "Race/ethnicity",
    age_bir ~ "Age at first birth"
  ))
```

Characteristic	Beta	95% CI [1]	p-value
(Intercept)	2,147	493, 3,802	0.011
Sex			
Male	—	—	
Female	25	-654, 705	>0.9
Age at first birth	438	381, 495	<0.001
Race/ethnicity			
Hispanic	—	—	
Black	-772	-1,714, 171	0.11
Non-Black, Non-Hispanic	7,559	6,676, 8,442	<0.001

[1] CI = Confidence Interval

gtsummary::tbl_regression()

```
tbl_regression(
  logistic_model,
  exponentiate = TRUE,
  label = list(
    sex_cat ~ "Sex",
    eyesight_cat ~ "Eyesight",
    income ~ "Income"
  ))
```

Characteristic	OR [1]	95% CI [1]	p-value
Eyesight			
Excellent	—	—	
Very good	0.92	0.82, 1.03	0.2
Good	0.92	0.80, 1.05	0.2
Fair	0.80	0.66, 0.98	0.028
Poor	1.03	0.69, 1.53	0.9
Sex			
Male	—	—	
Female	2.04	1.85, 2.25	<0.001
Income	1.00	1.00, 1.00	<0.001

[1] OR = Odds Ratio, CI = Confidence Interval

Arguments

Argument	Description
`label=`	modify variable labels in table
`exponentiate=`	exponentiate model coefficients
`include=`	names of variables to include in output. Default is all variables
`show_single_row=`	By default, categorical variables are printed on multiple rows. If a variable is dichotomous and you wish to print the regression coefficient on a single row, include the variable name(s) here.
`conf.level=`	confidence level of confidence interval
`intercept=`	indicates whether to include the intercept
`estimate_fun=`	function to round and format coefficient estimates
`pvalue_fun=`	function to round and format p-values
`tidy_fun=`	function to specify/customize tidier function

You could put several together

```r
tbl_no_int <- tbl_regression(
  linear_model,
  intercept = TRUE,
  label = list(
    sex_cat ~ "Sex",
    race_eth_cat ~ "Race/ethnicity",
    age_bir ~ "Age at first birth"
  ))

tbl_int <- tbl_regression(
  linear_model_int,
  intercept = TRUE,
  label = list(
    sex_cat ~ "Sex"
```

You could put several together

```
tbl_merge(list(tbl_no_int, tbl_int),
          tab_spanner = c("**Model 1**", "**Model 2**"))
```

Characteristic	Model 1			Model 2		
	Beta	95% CI [1]	p-value	Beta	95% CI [1]	p-value
(Intercept)	2,147	493, 3,802	0.011	4,064	1,884, 6,245	<0.001
Sex						
Male	—	—		—	—	
Female	25	-654, 705	>0.9	-3,635	-6,432, -838	0.011
Age at first birth	438	381, 495	<0.001	364	285, 443	<0.001
Race/ethnicity						
Hispanic	—	—		—	—	
Black	-772	-1,714, 171	0.11	-759	-1,701, 183	0.11
Non-Black, Non-Hispanic	7,559	6,676, 8,442	<0.001	7,550	6,668, 8,433	<0.001
Sex/age interaction						

[1] CI = Confidence Interval

Characteristic	Model 1			Model 2		
	Beta	95% CI [1]	p-value	Beta	95% CI [1]	p-value
Female * Age at first birth				149	39, 260	0.008

[1] CI = Confidence Interval

Exercises

1. Open the script with some examples.
2. Run the examples.

3-6. You're on your own again!

Extra time? Start a table using the data you downloaded for your final project! Make sure you switch to that R project!

15:00

Finer control over statistics

We fit a series of univariate regressions

```r
income_table <- tbl_uv
  nlsy,
  y = income,
  include = c(
    sex_cat, race_eth_
    eyesight_cat, inco
  ),
  method = lm
)
income_table
```

Characteristic	N	Beta	95% CI [1]	p-value
sex_cat	10,195			
Male		—	—	
Female		-358	-844, 128	0.15
race_eth_cat	10,195			
Hispanic		—	—	
Black		-1,747	-2,507, -988	<0.001
Non-Black, Non-Hispanic		3,863	3,195, 4,530	<0.001
eyesight_cat	6,789			
Excellent		—	—	
Very good		-578	-1,319, 162	0.13
Good		-1,863	-2,719, -1,006	<0.001
Fair		-4,674	-5,910, -3,439	<0.001
Poor		-6,647	-9,154,	<0.001

[1] CI = Confidence Interval

Characteristic	N	Beta	95% CI [1]	p-value
			-4,140	
age_bir	4,773	595	538, 652	<0.001

[1] CI = Confidence Interval

But a table is a limited form of output

We might want to dig in a little more to those regressions

- One helpful option from `{gtsummary}` is to extract data from the table directly
- This can be reported in a manuscript (rather than copying and pasting from the table)

```
1  inline_text(income_table, variable = "age_bir")
```

```
[1] "595 (95% CI 538, 652; p<0.001)"
```

We'll look at this again later!

What if we want *all* the numbers, say to create a figure?

- Under the hood, `{gtsummary}` is using the `{broom}` package to extract the statistics from the various models
- We can also use that package directly!

Statistical models in R can be messy

```r
mod_sex_cat <- lm(income ~ sex_cat, data = nlsy)
```

We could look at the model summary:

```r
summary(mod_sex_cat)
```

```
Call:
lm(formula = income ~ sex_cat, data = nlsy)

Residuals:
   Min     1Q Median     3Q    Max
-14880  -8880  -3943   5477  60478

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    14880.3      172.6  86.237   <2e-16 ***
sex_catFemale   -357.8      247.8  -1.444    0.149
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12510 on 10193 degrees of freedom
  (2491 observations deleted due to missingness)
```

Statistical models in R can be messy

If we want to do something with the various values, we could extract each statistic individually:

```
coef(mod_sex_cat)
```

```
  (Intercept) sex_catFemale
   14880.3152     -357.8029
```

```
confint(mod_sex_cat)
```

```
                 2.5 %     97.5 %
(Intercept)   14542.079 15218.5512
sex_catFemale  -843.608   128.0022
```

```
summary(mod_sex_cat)$r.squared
```

```
[1] 0.0002044429
```

```
summary(mod_sex_cat)$coefficients
```

```
               Estimate Std. Error   t value  Pr(>|t|)
(Intercept)   14880.3152   172.5521 86.236672 0.0000000
sex_catFemale  -357.8029   247.8349 -1.443715 0.1488499
```

{broom} has three main functions: augment(), glance(), tidy()

augment() adds fitted values, residuals, and other statistics to the original data

```
1 library(broom)
2 augment(mod_sex_cat)
```

```
# A tibble: 10,195 × 9
   .rownames income sex_cat .fitted  .resid     .hat .sigma   .cooksd .std.resid
   <chr>      <dbl> <fct>     <dbl>   <dbl>    <dbl>  <dbl>     <dbl>      <dbl>
 1 1          30000 Female   14523.  15477. 0.000202 12506.   1.55e-4      1.24
 2 2          20000 Female   14523.   5477. 0.000202 12507.   1.94e-5      0.438
 3 3          22390 Female   14523.   7867. 0.000202 12507.   4.01e-5      0.629
 4 4          22390 Female   14523.   7867. 0.000202 12507.   4.01e-5      0.629
 5 5          36000 Male     14880.  21120. 0.000190 12505.   2.72e-4      1.69
 6 6          35000 Male     14880.  20120. 0.000190 12505.   2.46e-4      1.61
 7 7           8502 Male     14880.  -6378. 0.000190 12507.   2.48e-5     -0.510
 8 8           7227 Female   14523.  -7296. 0.000202 12507.   3.44e-5     -0.583
 9 9          17000 Male     14880.   2120. 0.000190 12507.   2.74e-6      0.170
10 10          3548 Female   14523. -10975. 0.000202 12506.   7.79e-5     -0.878
# ℹ 10,185 more rows
```

{broom} has three main functions: augment(), glance(), tidy()

glance() creates a table of statistics that pertain to the entire model

```
1  glance(mod_sex_cat)
```

```
# A tibble: 1 × 12
  r.squared adj.r.squared  sigma statistic p.value    df   logLik     AIC    BIC
      <dbl>         <dbl>  <dbl>     <dbl>   <dbl> <dbl>    <dbl>   <dbl>  <dbl>
1  0.000204      0.000106 12506.      2.08   0.149     1 -110644. 221295. 2.21e5
# ℹ 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

`{broom}` has three main functions: `augment()`, `glance()`, `tidy()`

`tidy()` is the most useful to me and probably you!

It extracts coefficients and confidence intervals from models

```
1  tidy(mod_sex_cat, conf.int = TRUE)
```

```
# A tibble: 2 × 7
  term          estimate std.error statistic p.value conf.low conf.high
  <chr>            <dbl>     <dbl>     <dbl>   <dbl>    <dbl>     <dbl>
1 (Intercept)     14880.      173.      86.2    0       14542.    15219.
2 sex_catFemale    -358.      248.      -1.44   0.149    -844.      128.
```

`tidy()` works on over 100 statistical methods in R!

Anova, ARIMA, Cox, factor analysis, fixed effects, GAM, GEE, IV, kappa, kmeans, multinomial, proportional odds, principal components, survey methods, ...

- See the full list here
- All the output shares column names
- This makes it really easy to work with the output and reuse code across analyses

Some models have additional arguments

For example, we might want exponentiated coefficients:

```
logistic_model <- glm(glasses ~ eyesight_cat + sex_cat + i
                      data = nlsy, family = binomial())
tidy(logistic_model, conf.int = TRUE, exponentiate = TRUE)
```

```
# A tibble: 7 × 7
  term                 estimate std.error statistic  p.value conf.low conf.high
  <chr>                   <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
1 (Intercept)             0.499   5.96e-2   -11.7   1.74e-31    0.444     0.560
2 eyesight_catVery good   0.920   5.96e-2    -1.39  1.64e- 1    0.819     1.03
3 eyesight_catGood        0.916   6.91e-2    -1.27  2.04e- 1    0.800     1.05
4 eyesight_catFair        0.802   1.00e-1    -2.20  2.77e- 2    0.658     0.976
5 eyesight_catPoor        1.03    2.01e-1     0.147 8.83e- 1    0.694     1.53
6 sex_catFemale           2.04    5.00e-2    14.2   5.46e-46    1.85      2.25
7 income                  1.00    1.93e-6     7.49  6.95e-14    1.00      1.00
```

We can also combine the results of lots of regressions

```
# we already made mod_sex_cat
mod_race_eth_cat <- lm(income ~ race_eth_cat, data = nlsy)
mod_eyesight_cat <- lm(income ~ eyesight_cat, data = nlsy)
mod_age_bir <- lm(income ~ age_bir, data = nlsy)

tidy_sex_cat <- tidy(mod_sex_cat, conf.int = TRUE)
tidy_race_eth_cat <- tidy(mod_race_eth_cat, conf.int = TRU
tidy_eyesight_cat <- tidy(mod_eyesight_cat, conf.int = TRU
tidy_age_bir <- tidy(mod_age_bir, conf.int = TRUE)
```

There are of course more efficient ways to do this instead of copy/pasting 4 times...

With a little finagling, we have the same data as in the original univartiate regression table...

```
dplyr::bind_rows(
  sex_cat = tidy_sex_cat,
  race_eth_cat = tidy_race_eth_cat,
  eyesight_cat = tidy_eyesight_cat,
  age_bir = tidy_age_bir, .id = "model") |>
  dplyr::mutate(
    term = stringr::str_remove(term, model),
    term = ifelse(term == "", model, term))
```

With a little finagling, we have the same data as in the original univartiate regression table...

```
# A tibble: 12 × 8
   model        term    estimate std.error statistic  p.value conf.low conf.high
   <chr>        <chr>      <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
 1 sex_cat      (Inter…   14880.     173.      86.2  0          14542.    15219.
 2 sex_cat      Female     -358.     248.      -1.44 1.49e- 1    -844.      128.
 3 race_eth_cat (Inter…   12867.     302.      42.7  0          12276.    13459.
 4 race_eth_cat Black     -1747.     387.      -4.51 6.58e- 6   -2507.     -988.
 5 race_eth_cat Non-Bl…    3863.     341.      11.3  1.20e-29    3195.     4530.
 6 eyesight_cat (Inter…   17683.     270.      65.6  0          17155.    18212.
 7 eyesight_cat Very g…    -578.     378.      -1.53 1.26e- 1   -1319.      162.
 8 eyesight_cat Good      -1863.     437.      -4.26 2.05e- 5   -2719.    -1006.
 9 eyesight_cat Fair      -4674.     630.      -7.42 1.35e-13   -5910.    -3439.
10 eyesight_cat Poor      -6647.    1279.      -5.20 2.07e- 7   -9154.    -4140.
11 age_bir      (Inter…    1707.     733.       2.33 1.99e- 2     270.     3143.
12 age_bir      age_bir     595.      29.1     20.4  3.71e-89     538.      652.
```

Even easier cleanup!

We could instead clean up the names and add reference rows with the `{tidycat}` package:

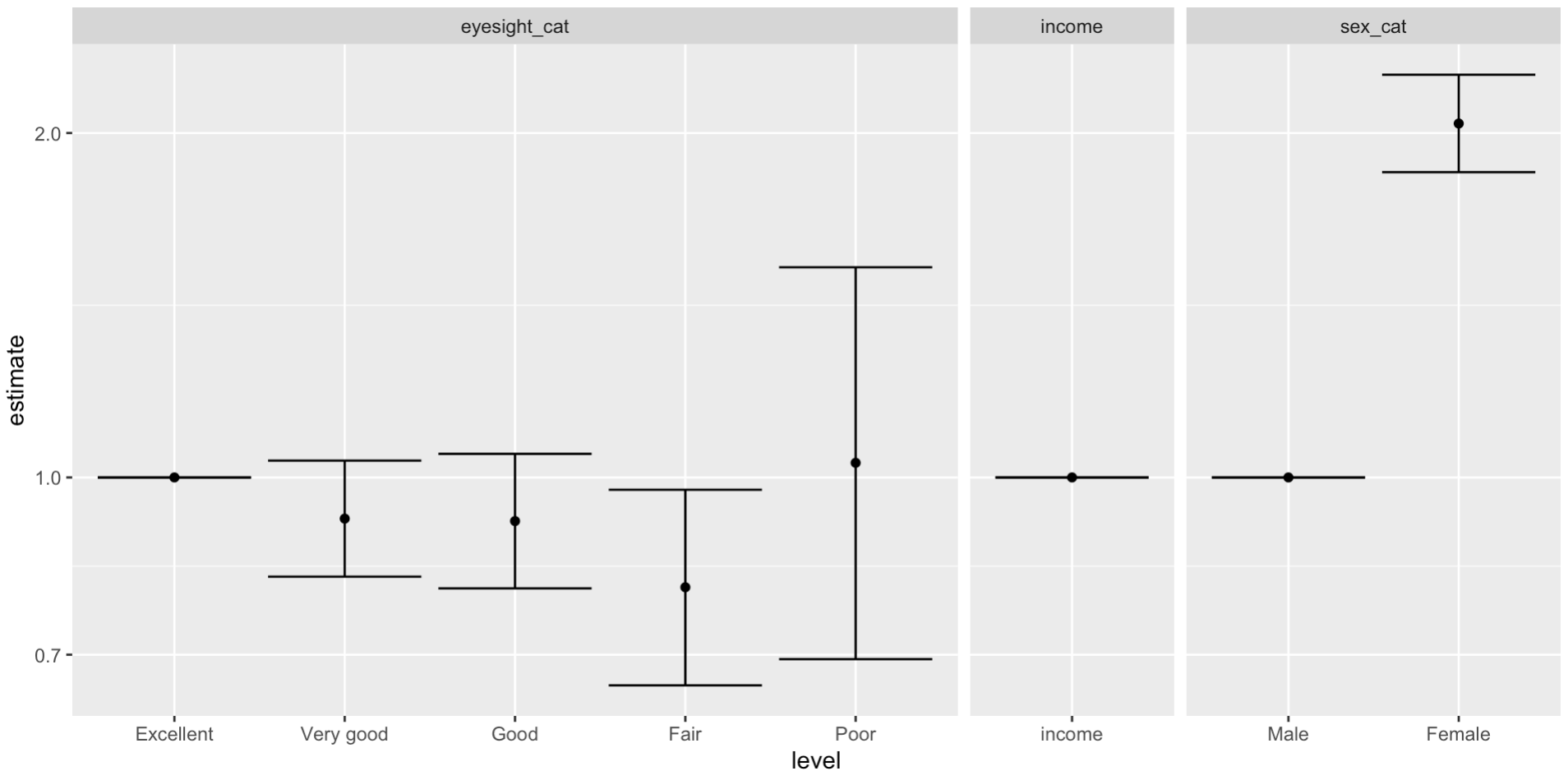
```
tidy(logistic_model, conf.int = TRUE, exponentiate = TRUE)
  tidycat::tidy_categorical(logistic_model, exponentiate =
  dplyr::select(-c(3:5))
```

```
# A tibble: 9 × 8
  term                estimate conf.low conf.high variable level effect reference
  <chr>                  <dbl>    <dbl>     <dbl> <chr>    <fct> <chr>  <chr>
1 (Intercept)            0.499    0.444     0.560 (Interc… (Int… main   Non-Base…
2 <NA>                   1        1         1     eyesigh… Exce… main   Baseline…
3 eyesight_catVery …     0.920    0.819     1.03  eyesigh… Very… main   Non-Base…
4 eyesight_catGood       0.916    0.800     1.05  eyesigh… Good  main   Non-Base…
5 eyesight_catFair       0.802    0.658     0.976 eyesigh… Fair  main   Non-Base…
6 eyesight_catPoor       1.03     0.694     1.53  eyesigh… Poor  main   Non-Base…
7 <NA>                   1        1         1     sex_cat  Male  main   Baseline…
8 sex_catFemale          2.04     1.85      2.25  sex_cat  Fema… main   Non-Base…
9 income                 1.00     1.00      1.00  income   inco… main   Non-Base…
```

This makes it easy to make forest plots, for example

```
library(ggplot2)
tidy(logistic_model, conf.int = TRUE, exponentiate = TRUE)
  tidycat::tidy_categorical(logistic_model, exponentiate =
  dplyr::slice(-1) |> # remove intercept
  ggplot(mapping = aes(x = level, y = estimate,
                       ymin = conf.low, ymax = conf.high))
  geom_point() +
  geom_errorbar() +
  facet_grid(cols = vars(variable), scales = "free", space
  scale_y_log10()
```

This makes it easy to make forest plots, for example



Exercises

1. Open the script with these examples.
2. Run it.
3. Teach yourself to use `broom::tidy()` to extract the results of the Poisson regression with robust standard errors and combine them with the results of the log-binomial regression.
4. Start creating some tables for your final project!

15:00