HDAT9800 Health Data Visualisation & Communication

Chapter 2 Interactive Tutorial - blogging with `distill`

Tim Churches

UNSW Medicine

8th June 2022

Agenda for Chapter 2 interactive session

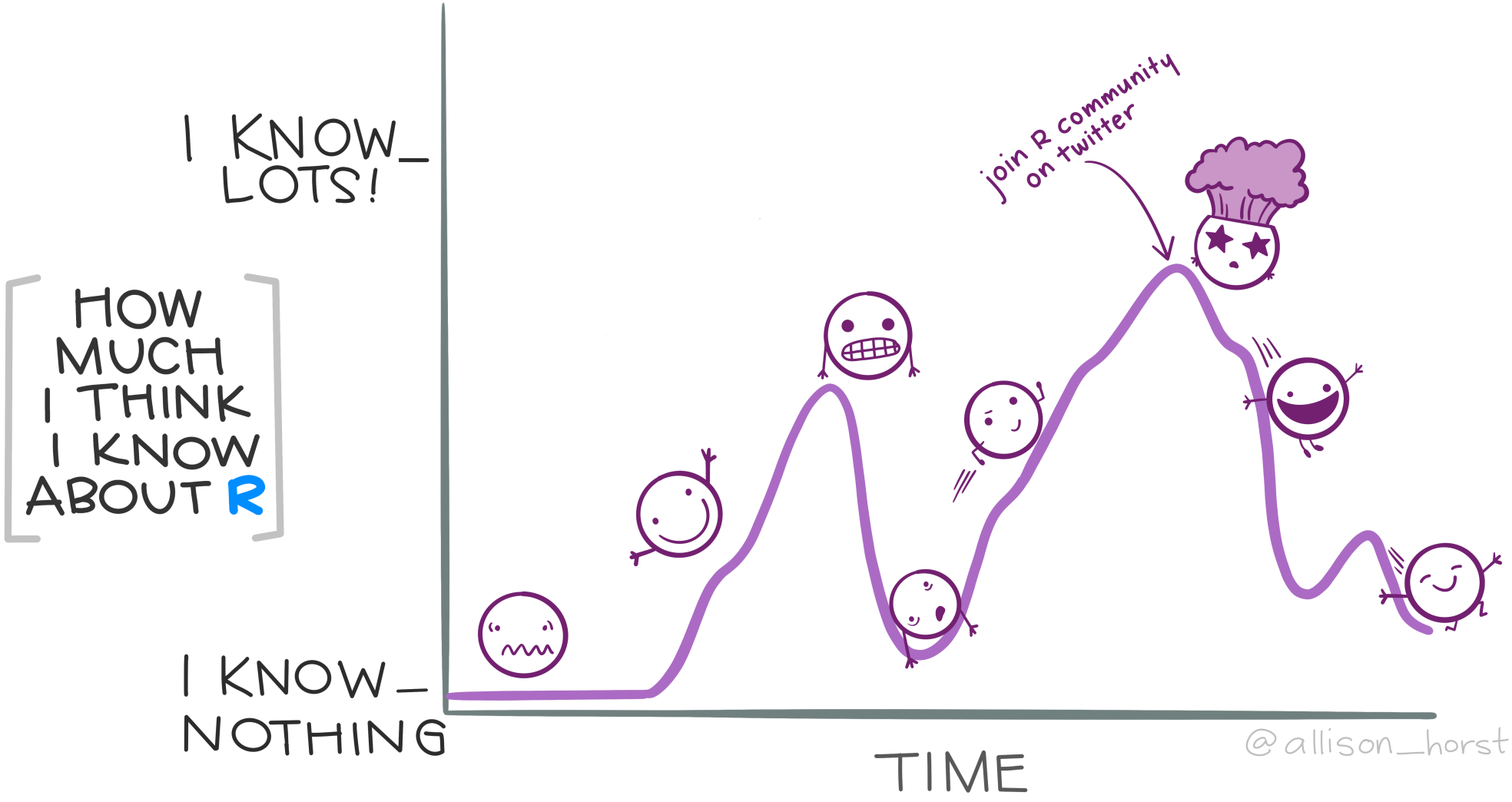
- Chapter 1
  - lots of content
  - learning SQL - a good idea but not essential
  - challenge solutions (to be posted in Teams)
  - Q & A
  - assessment and requesting early marking

Agenda for Chapter 2 interactive session, cont'd

- Chapter 2
  - deliberately less content this week
  - Q & A
  - core readings (Wilke Chapters 1 & 2 recap in preparation for Chapter 3)
  - hands-on blogging with `distill` for `R`
  - review of how the internet and Web work
  - types of web site architectures/platforms
  - the blogosphere as a means of technical communication
  - overview of `distill`
  - hands-on setting up your own blog site

Chapter 1 recap

- lots of content
  - especially if you are new to `R` or to programming
  - don't panic!
  - we are here to help
  - no-one expects to to be an instant *Top Gun* coder
  - it's a journey


I KNOW LOTS!
HOW MUCH I THINK I KNOW ABOUT R
I KNOW NOTHING
join R community on twitter
TIME
@allison_horst


maintain
mentors!
friends
teachers!
developers!
bloggers!
contrib
i'm doing a thing all on my own!
code hero
support
community
@allison_horst

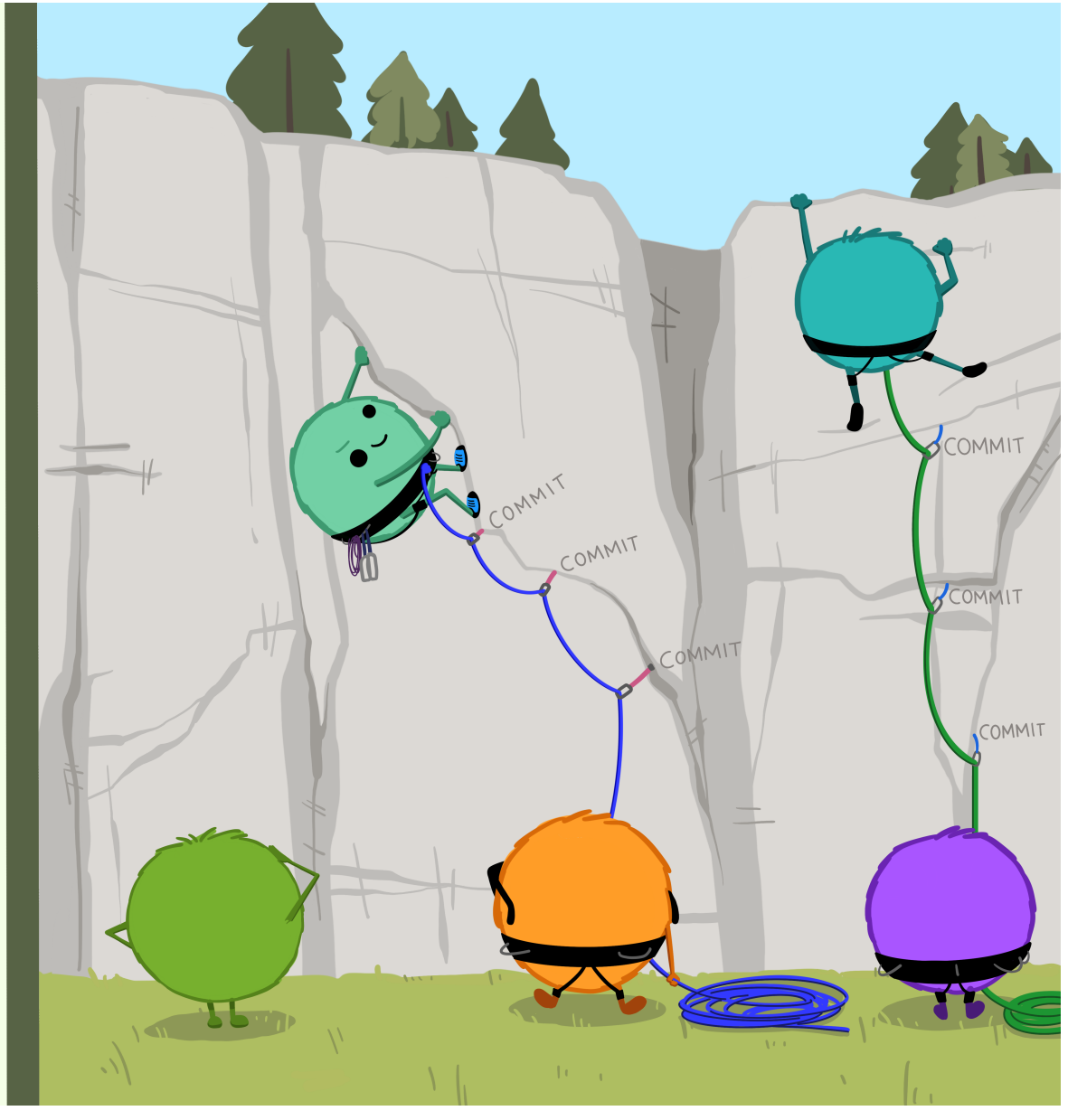
Chapter 1 recap

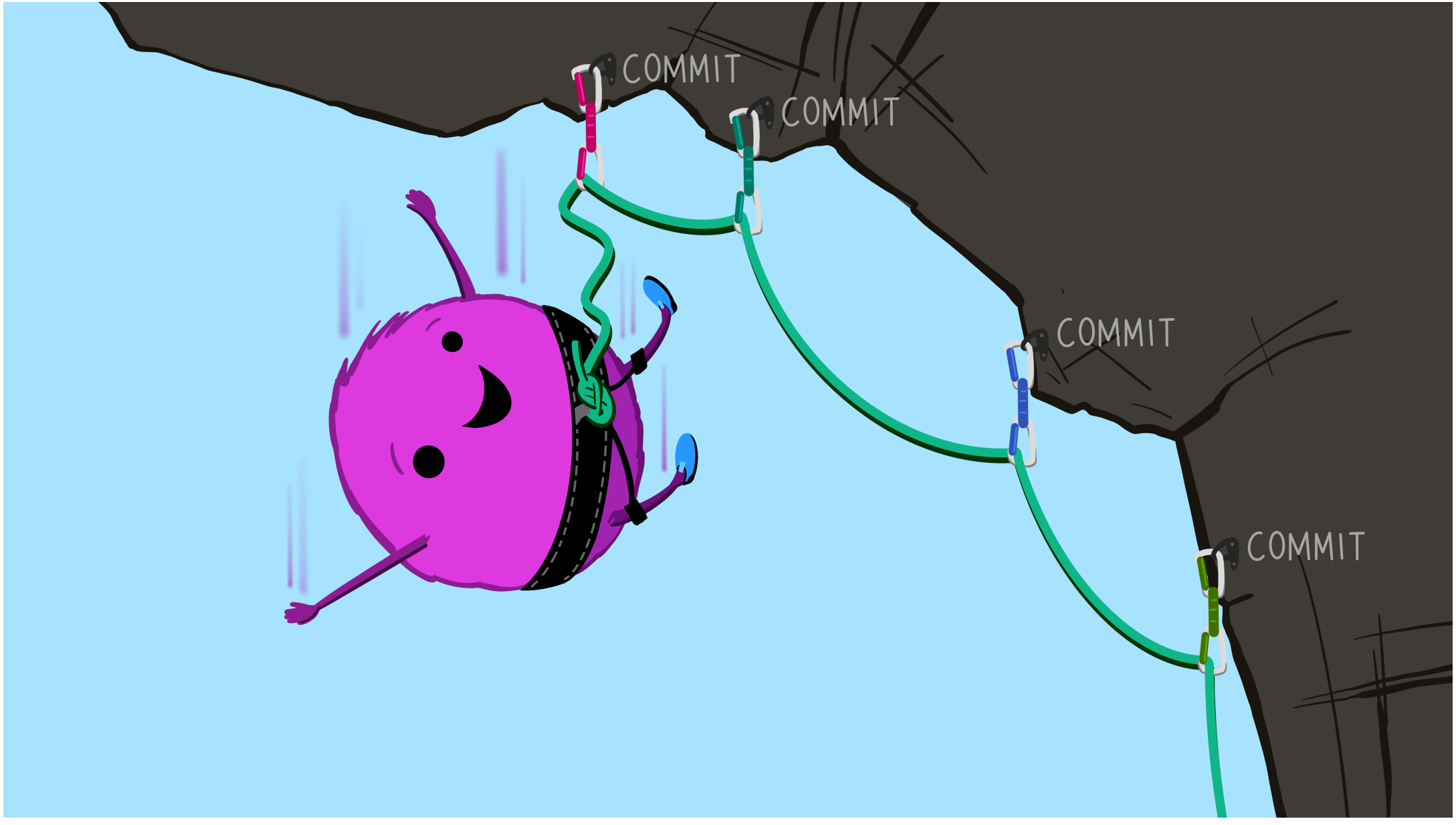
- git and GitHub confuse almost everyone at first
    - practice brings comfort

> 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

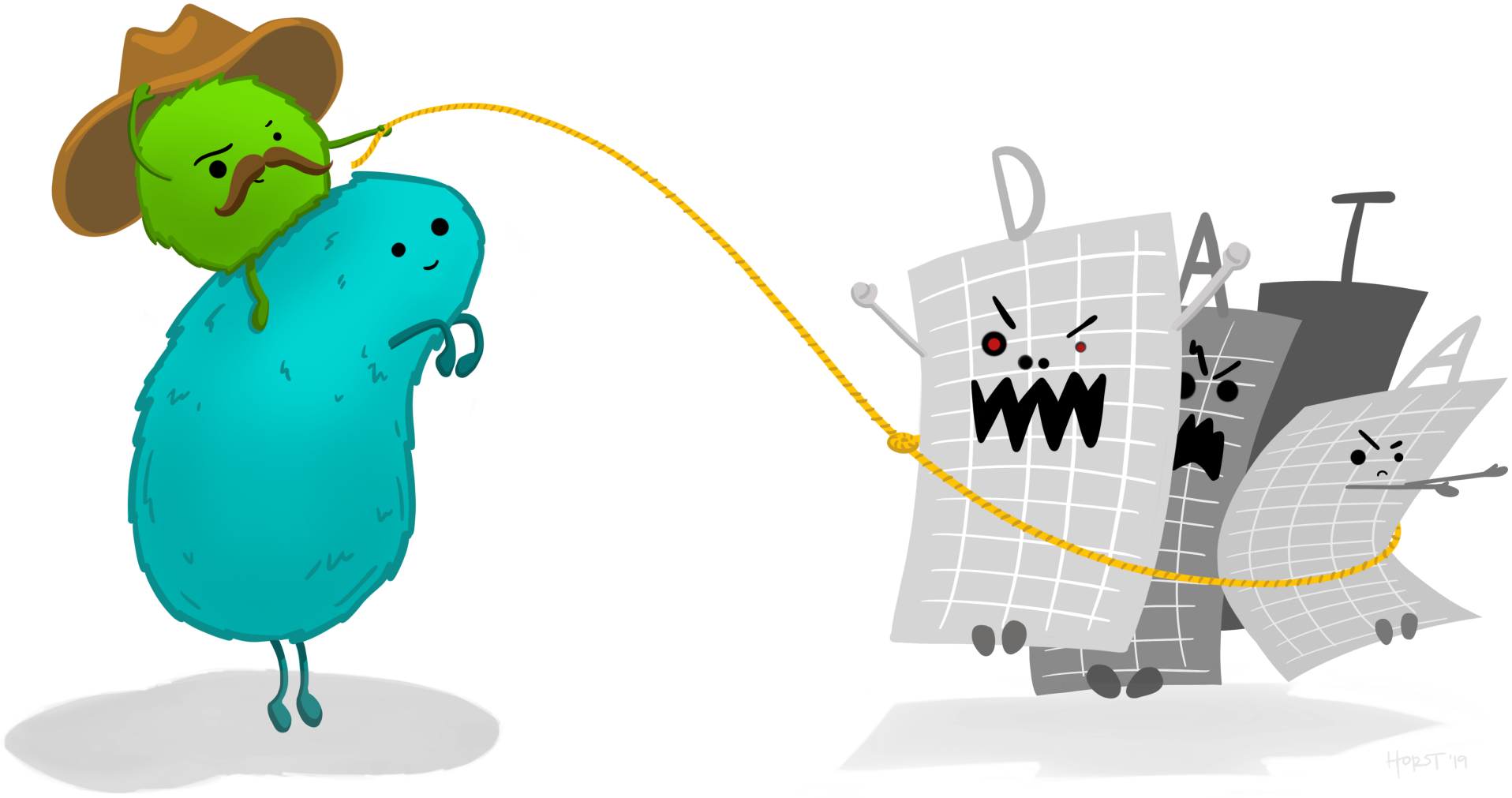
Wickham & Bryan, RPackages (https://r-packages.org/preface.html)




COMMIT
COMMIT
COMMIT
COMMIT

Chapter 1 recap, cont'd

- `dplyr` and related packages (`tidyr`, `purrr`) worth learning
  - we'll cover `tidyr` and `purrr` briefly in a later interactive session
  - learning SQL - a good idea, but not necessary for this course
- `dplyr`/`dbplyr` challenge solutions (to be posted in Teams)


D
A
A
HORST '19

Chapter 1 recap, cont'd

- assessment
  - very easy, just to get you used to using `git` and `GitHub`
- procedure for requesting early marking
- live demo of it (seeking a volunteer!)

Chapter 1 recap, cont'd

- Q & A

Chapter 2

- deliberately less content this week
- Q & A


Rmarkdown
TEXT. CODE. OUTPUT.
(GET IT TOGETHER, PEOPLE.)
#HEADERS
CODE
italics
TEXT
PANDOC
KNITR
FAIRY DUST
RECIPE
1. ADD TEXT
2. ADD CODE
3. KNIT
4. (magic)
5. CELEBRATE PERCEIVED WIZARDRY
YAML
rmarkdown
.Rmd
knitr
.md
pandoc
Horst '19

Wilke Chapter 1

- data visualisation is 50% art & 50% science
  - most important thing is to accurately communicate the data without distortion or bias (the science)
  - but do so in an aesthetically pleasing way (the art)
- scientists tend to be good at the science but are often aesthetically challenged
- artists and designers tend to be good at preparing beautiful or pleasing visualisations, but care less about the accuracy of what they are presenting
- three parts:
  - from data to visualisation
  - principles of figure/chart design
  - miscellaneous topics (not core reading for this course)

Wilke Chapter 1, cont'd

- good and bad examples, using the same data
  - ugly - a chart with aesthetics problems (a matter of taste) but otherwise clear, informative and accurate
  - bad - a chart with perceptual problems, making it unclear, confusing, too complicated or possibly deceiving
  - wrong - a chart with objective, mathematical errors, indisputably incorrect

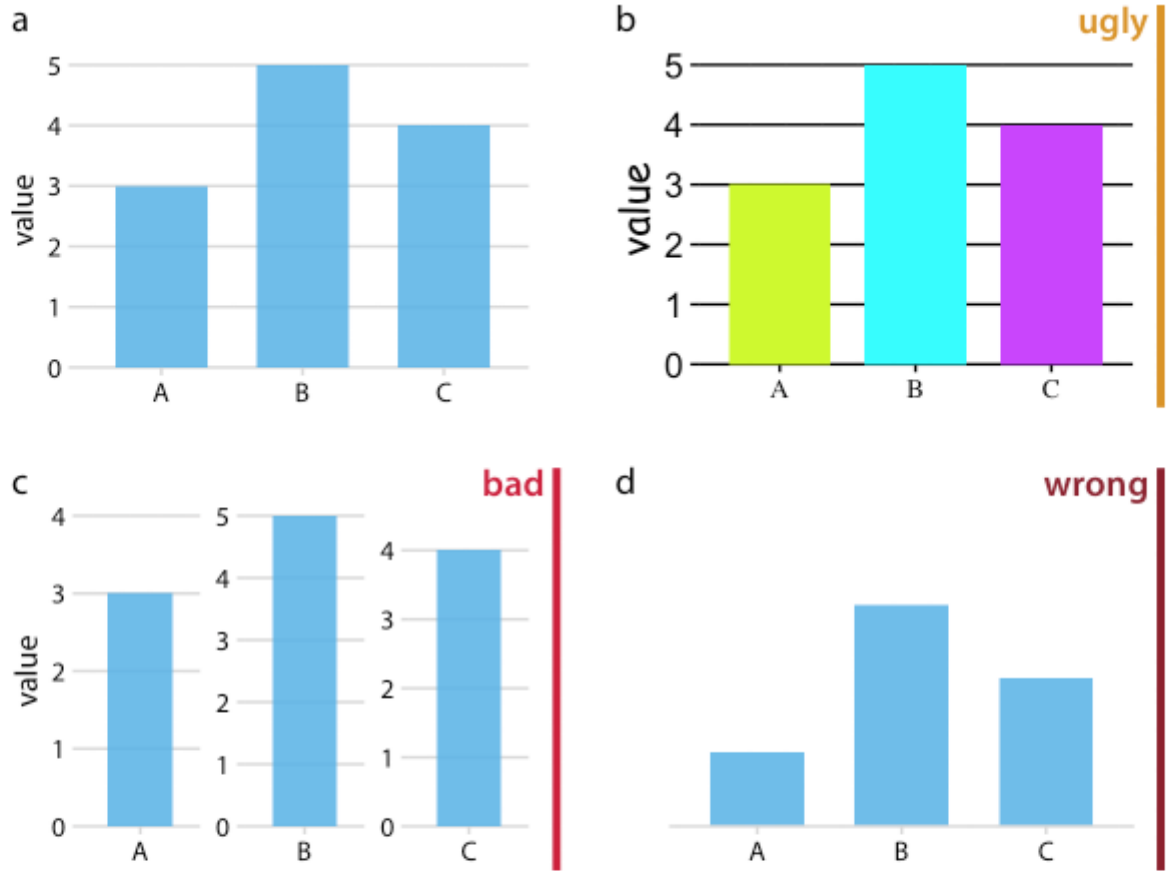


Figure 1.1: Examples of ugly, bad, and wrong figures. (a) A bar plot showing three values (A = 3, B = 5, and C = 4). This is a reasonable visualization with no major flaws. (b) An ugly version of part (a). While the plot is technically correct, it is not aesthetically pleasing. The colors are too bright and not useful. The background grid is too prominent. The text is displayed using three different fonts in three different sizes. (c) A bad version of part (a). Each bar is shown with its own *y*-axis scale. Because the scales don't align, this makes the figure misleading. One can easily get the impression that the three values are closer together than they actually are. (d) A wrong version of part (a). Without an explicit *y* axis scale, the numbers represented by the bars cannot be ascertained. The bars appear to be of lengths 1, 3, and 2, even though the values displayed are meant to be 3, 5, and 4.

wrong!

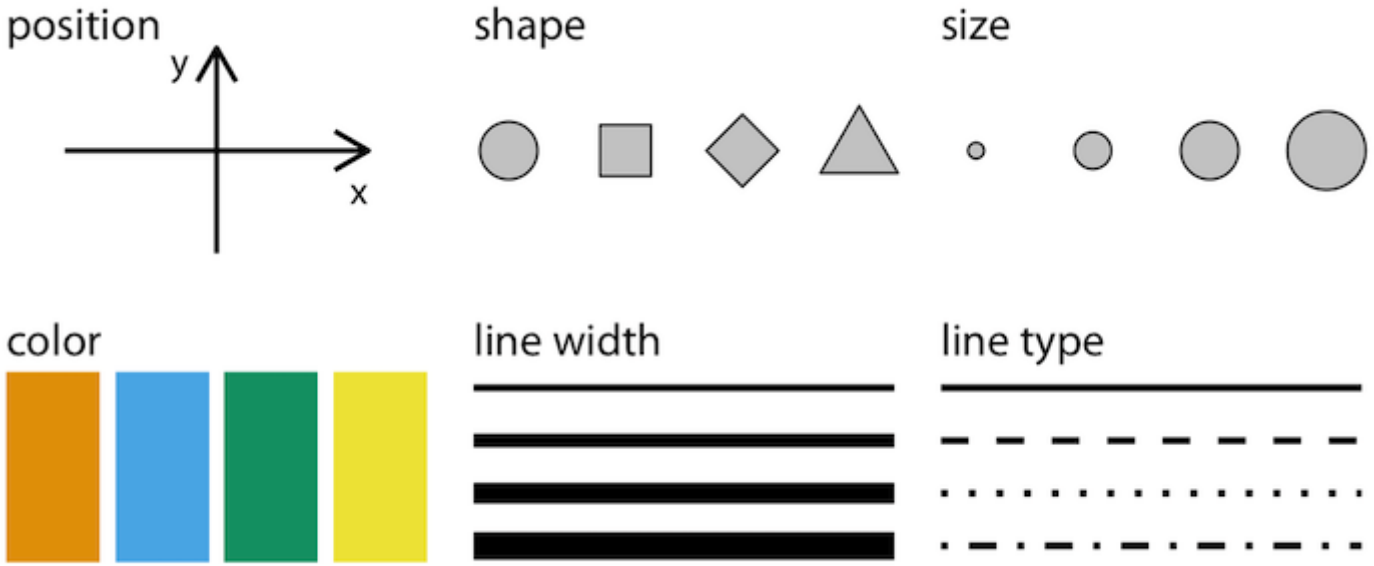
An incorrect meta-analysis detected due to an incorrect chart

Wilke Chapter 2

- restates the principles espoused by Leland Wilkinson, building on Bertin
- all data visualisations map data values onto quantifiable features of the chart or graphic
    - we call these feature *aesthetics* (in a particular, technical sense, different from the usual meaning of *aesthetics*)

Aesthetics and data types

- aesthetics describe every aspect of a particular graphical element
  - for example, the *position* of a point (described by *x* and *y* values in a 2D chart)
  - graphical elements have a *shape*, a *size*, a *colour*, a *line type*, a *line width* etc



Data types in charts and graphics

Table 2.1: Types of variables encountered in typical data visualization scenarios.

Type of variable	Examples	Appropriate scale	Description
quantitative/numerical continuous	1.3, 5.7, 83, $1.5x10^{-2}$	continuous	Arbitrary numerical values. These can be integers, rational numbers, or real numbers.
quantitative/numerical discrete	1, 2, 3, 4	discrete	Numbers in discrete units. These are most commonly but not necessarily integers. For example, the numbers 0.5, 1.0, 1.5 could also be treated as discrete if intermediate values cannot exist in the given dataset.
qualitative/categorical unordered	dog, cat, fish	discrete	Categories without order. These are discrete and unique categories that have no inherent order. These variables are also called *factors*.
qualitative/categorical ordered	good, fair, poor	discrete	Categories with order. These are discrete and unique categories with an order. For example, "fair" always lies between "good" and "poor". These variables are also called *ordered factors*.
date or time	Jan. 5 2018, 8:03am	continuous or discrete	Specific days and/or times. Also generic dates, such as July 4 or Dec. 25 (without year).
text	The quick brown fox jumps over the lazy dog.	none, or discrete	Free-form text. Can be treated as categorical if needed.

What are the data types in this table?

Table 2.2: First 12 rows of a dataset listing daily temperature normals for four weather stations. Data source: NOAA.

Month	Day	Location	Station ID	Temperature
Jan	1	Chicago	USW00014819	25.6
Jan	1	San Diego	USW00093107	55.2
Jan	1	Houston	USW00012918	53.9
Jan	1	Death Valley	USC00042319	51.0
Jan	2	Chicago	USW00014819	25.5
Jan	2	San Diego	USW00093107	55.3
Jan	2	Houston	USW00012918	53.8
Jan	2	Death Valley	USC00042319	51.2
Jan	3	Chicago	USW00014819	25.3
Jan	3	San Diego	USW00093107	55.3
Jan	3	Death Valley	USC00042319	51.3
Jan	3	Houston	USW00012918	53.8

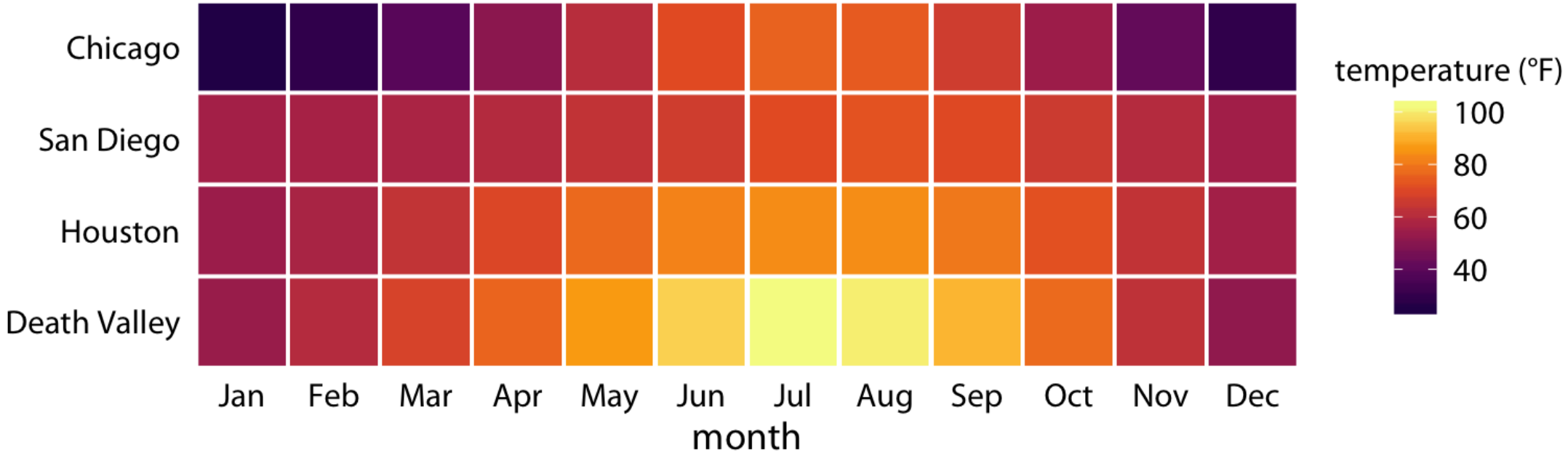
Scales map data values onto aesthetics

- to map data values to *aesthetics* we need to specify which data values correspond to which aesthetic values or characteristics
    - eg if our chart has an *x* axis, we need to specify which data values (eg which column of data in a table/data frame) specify where element fall along the *x* axis
    - this mapping is done via a *scale*
    - a *scale* defines a unique mapping from data values to aesthetics
    - must be a one-to-one mapping, otherwise the graphic will be ambiguous or uninterpretable

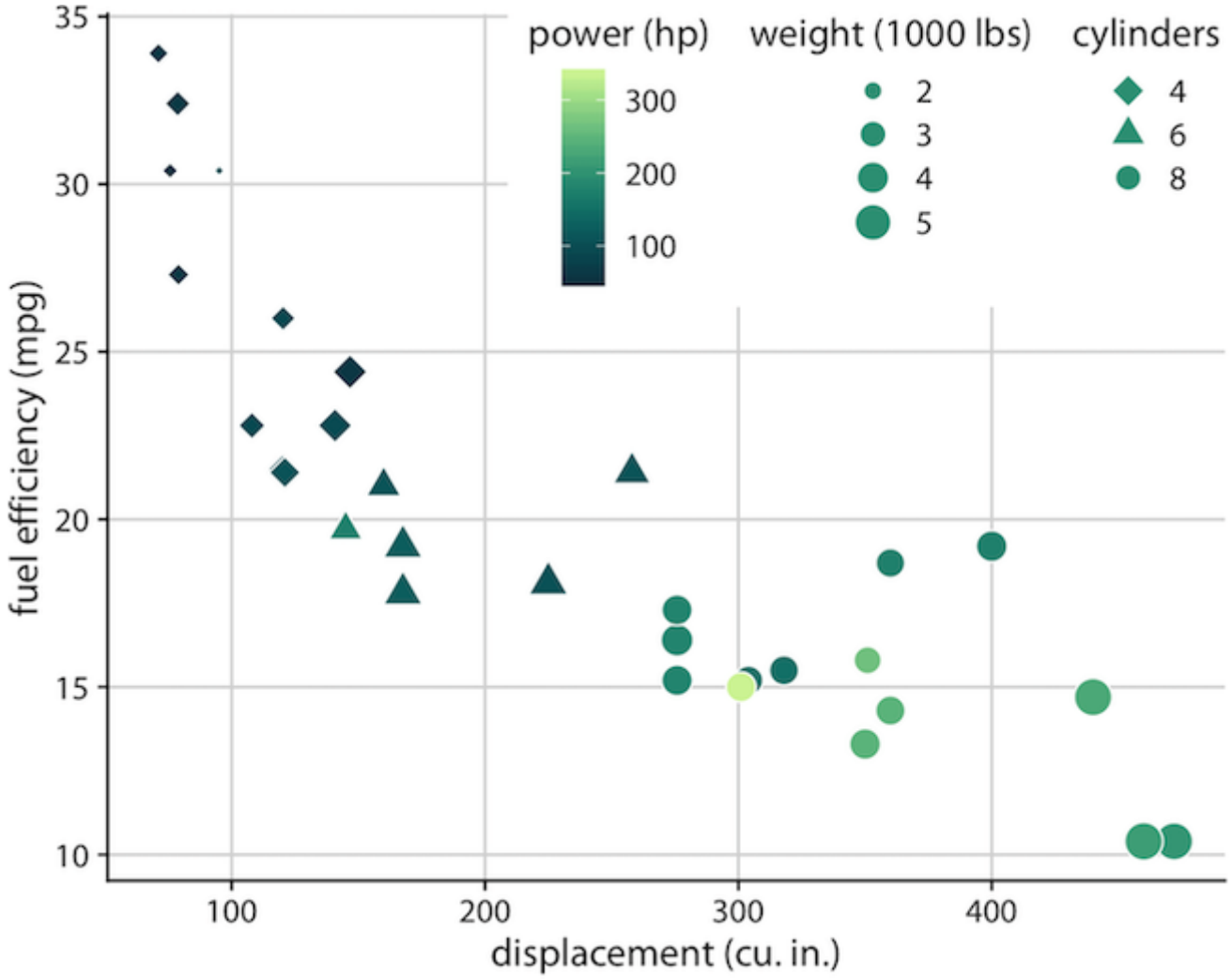
Mapping data to aesthetics in practice

- three columns from the table we saw earlier mapped to aesthetics

An alternative mapping of the same data to aesthetics



Five mappings of data to aesthetics



Blogging and the blogosphere

- blogs are a value means of communicating your work in health data science
  - as the primary means for presenting your results (but don't forget peer-reviewed scientific publishing)
  - to talk about how you got your results
  - data collection/sources, statistics and epidemiology, software
  - to discuss or comment on (constructively!) on other people's work
  - training material and guides for others

Data science blogosphere

- very rich, and growing
- a nice blog site is a great addition to your CV/resumé
- Top 40 R programming blogs to follow in 2022
- R Bloggers blog aggregation site
- Tim Churches' health data science blog

Let's build our own blog sites

- GitHub allows you to publish static web pages from a GitHub repository
  - see GitHub Pages for details
  - we will leverage that to host our personal blog sites
  - static in the sense that the blog site is a set of static files served up by the GitHub Pages web server
  - but we can modify and add to that set of files at any time, and the blog site content will be updated according
  - all managed via git

Distill

- has a sound theoretical underpinning -- see Hohman *et al*., 2020
  - a very nice framework for data science blogging (and websites)
- a Distil scientific website: https://cbdrh.github.io/covoidance/
  - see the Distill web sites for details
- intro to Distill
- re-intro to Distill
- reference documentation site

Overview of steps in setting up you own blog site

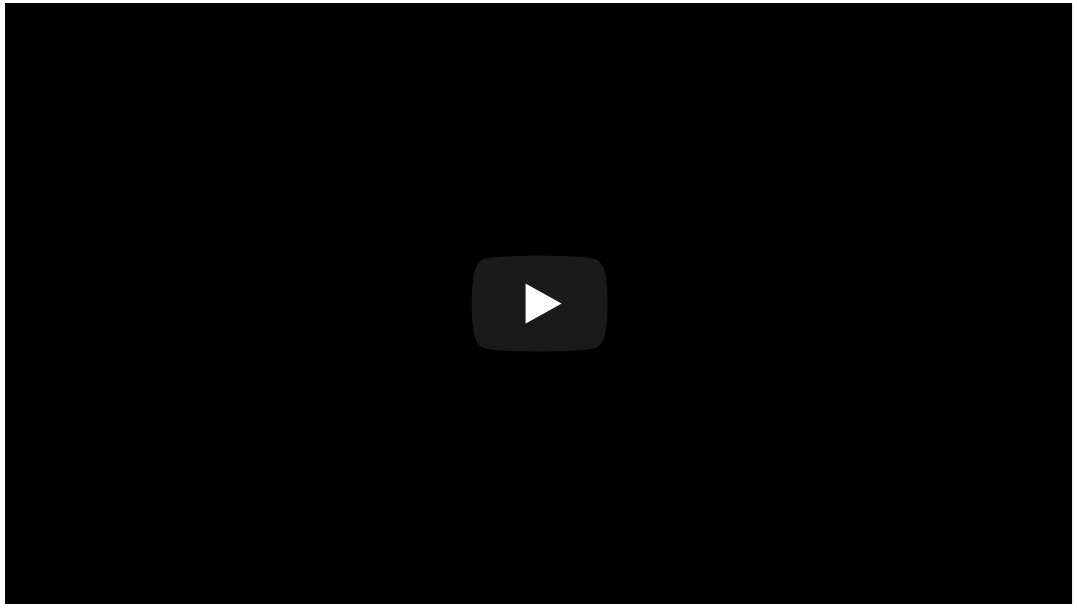
- create a repository (repo) for the blog under your personal GitHub account
    - should be a public repository
- set up a new RStudio project and clone the blog repo to your computer
- set up a new Distill blog site project in the same directory that you cloned the repo to
- do some initial customisation of the blog site
- rebuild the blog site
- commit to git and push back to the GitHub repo
- enable GitHub Pages in GitHub
- check if it worked

Maintaining your own blog site

- further customisation of your site
- modify an existing blog post
- add one or more new blog posts

A video illustrating these steps

- the steps are described in subsequent slides



Install packages

- you need to install the `distill` package
    - this will automatically install all the other packages it depends on

```
install.packages("distill", dependencies = TRUE)
```

Create a new GitHub repo for your blog

- call it whatever you wish
  - but put "blog" in the name
- go to your GitHub home page
  - create a new repository
  - make it a public repository

Clone that repo to you local computer using RStudio

Load `distill` and create a new web site

```
library(distill)
create_blog(dir = ".", title = "My Blog")
```

Exit RStudio then re-open your blog project in RStudio

- double click on the `.Rproj` file for yiur blog project to re-open the project in RStudio
    - this is necessary so that RStudio detects that you are using `distill` so that a Build tab appears in the upper-right pane in RStudio

Set up GitHub Pages

- change Pages branch and directory to:
  - branch main
  - directory docs/

Make a `.nojekyll` file in your blog repo

This tells Google Pages not to do any further transformation of web pages using something called `jekyll`

```
file.create(".nojekyll")
```

Adjust _site.yml to suit

- adjust the `output_dir` parameter to `docs`

Build the site

- click on the Build tab in the upper-right pane in RStudio, then click Build Website

Commit all modified files to git then push to GitHub

- then wait a bit
- check if published by accessing the URL for your GitHub Pages site provided in the Pages setup screen on GitHub

Create a new post

```
create_post("I love blogging")
```

- then knit that post so it is rendered
- rebuild the website (see above)
- commit all changes to git and push to GitHub

Customise!

https://rstudio.github.io/distill/

Enough!