Data Wrangling

Data Science
Week 3

Today's class

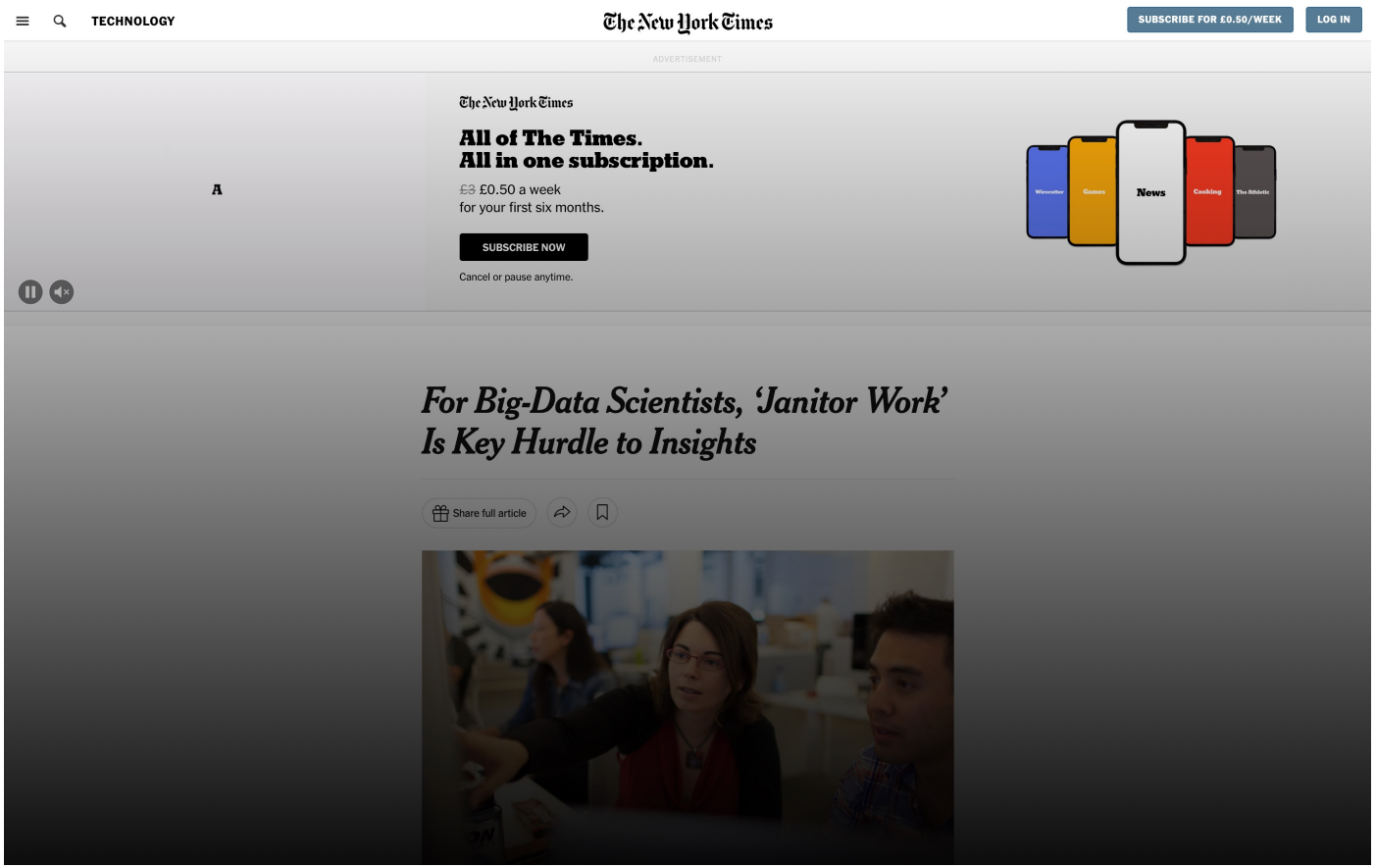
Plenary:

- What's data wrangling? Why?
  - "The tidyverse"
  - Best practices
- Any github issues?
- Assessment 1 details

In class activity

- Wrangle the LPI dataset!

What’s data wrangling?



50-80% of time spent on data wrangling ('janitor work' - rude)

The TidyVerse

Hadley Wickham

Twitter | GitHub | Email

Hi! I'm Hadley Wickham, Chief Scientist at RStudio, and an Adjunct Professor of Statistics at the University of Auckland, Stanford University, and Rice University. I build tools (computational and cognitive) that make data science easier, faster, and more fun. I'm from New Zealand but I currently live in Houston, TX with my husband and dogs.

If you'd like to learn more about what I do, and how to use R effectively, you might enjoy one of my books (which are all free to read on the web):

- R for Data Science, with Garrett Grolemund, is the place to start if you want to learn how to do data science with R.
- ggplot2: elegant graphics for data analysis shows you how to use ggplot2 to create graphics that help you understand your data.
- Advanced R helps you master R as a programming language, teaching you what makes R tick.
- R packages teaches good software engineering practices for R, using packages for bundling, documenting, and testing your code.

Outside of work, I love to bake and make cocktails. My sister Charlotte is a Assistant Professor of Statistics at Oregon State University.

https://hadley.nz

The TidyVerse

Think of Base R vs the TidyVerse as different dialects of the same language

TidyVerse

```
elong_subset <- filter(elongation_long, zone %in% c(2, 3), year %in% c("X2009", "X2010", "X2011"))
```

Base R

```
elong_subset <- elongation_long[elongation_long$zone %in% c(2,3) & elongation_long$year %in% c("X2009", "X2010", "X2011"), ]
```

The TidyVerse

Is my stack exchange answer using tidyverse code?

Some hints:

Does the code have pipes (%>%)

Are there any functions using a bad pun of ‘r’ instead of ‘er’ (eg tidyr rather than tidier)

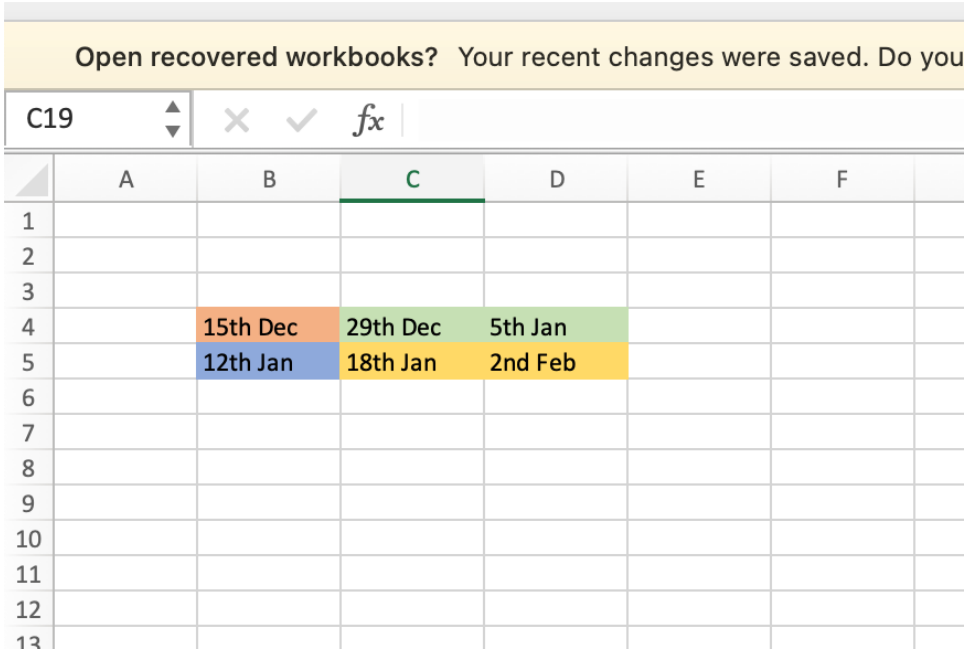
The TidyVerse

The tidy verse can’t do everything!! It’s great for data wrangling, but you will sometimes need to step outside its syntax.

The stages of good practice

1. Organise your input data!

What not to do…



The stages of good practice

1. Organise your input data! Good input data should be…
   - In .csv or .txt format
   - Rather than multiple ‘sheets’ in a spreadsheet, save as separate files (though if it’s already happened use package readxl)
   - Have no colours, special characters or merged cells

The stages of good practice

1. Organise your input data!
2. Tidy your data

The stages of good practice

1. Organise your input data!
2. Tidy your data
   - Convert to long format

Wide Format (good for visualization, papers, presentations)

	Zone	Indiv	X2007	X2008	X2009	X2010	X2011	X2012
1	2	373	5.1	5.1	4.8	8.7	6.3	3.2
2	2	379	8.1	13.3	8.6	4.9	5.9	6.3
3	2	383	9.3	8.5	11.7	7.9	8.0	6.3
4	2	389	15.0	10.3	6.8	6.9	5.9	7.6
5	2	390	3.5	6.2	4.7	3.8	3.5	3.0
6	2	395	6.1	5.6	4.4	4.5	4.5	7.6

Long Format (Necessary for analysis!)

Zone	Indiv	Year	Length
2	373	X2007	5.1
2	379	X2007	8.1
2	383	X2007	9.3
2	389	X2007	15.0
2	390	X2007	3.5
2	395	X2007	6.1
2	396	X2007	7.2
2	408	X2007	6.1
2	412	X2007	4.6
2	421	X2007	7.2
2	425	X2007	6.4
2	429	X2007	8.9
2	431	X2007	3.5

The stages of good practice

1. Organise your input data!
2. Tidy your data
   - Convert to long format
   - Get your names sorted

   A column named “year data was collected” is no good

The stages of good practice

1. Organise your input data!
2. Tidy your data
   - Convert to long format
   - Get your names sorted

   A column named “year data was collected” is no good
   Change to “year”

Flat Case: speciesnames

Camel Case: speciesNames

Snake Case: species_names

✨ Pick your naming style ✨

Pascal Case: SpeciesNames

Point Case: species.names

Kebab Case: species-names

The stages of good practice

1. Organise your input data!
2. Tidy your data
   - Convert to long format
   - Get your names sorted
   - Make sure columns are in the correct format (numeric, character, factor)

The stages of good practice

1. Organise your input data!
2. Tidy your data
   - Convert to long format
   - Get your names sorted
   - Make sure columns are in the correct format (numeric, character, factor)

Numeric = Numbers
Character = Words
Factor = *Ordered* words, where the levels are the order (e.g. Good, Better, Best)

The stages of good practice

1. Organise your input data!
2. Tidy your data
3. Figure out your workflow

The stages of good practice

1. Organise your input data!
2. Tidy your data
3. Figure out your workflow

E.g. I have a dataset of counts of birds in 3 forests through time. What's the average number of birds per species per year?

The stages of good practice

1. Organise your input data!
2. Tidy your data
3. Figure out your workflow

E.g. I have a dataset of counts of birds in 3 forests through time. What's the average number of birds per species per year?

Now to wrangle - dplyr

Set of many useful functions for wrangling

Look here: https://dplyr.tidyverse.org/reference/index.html

Cheat sheets here: https://rstudio.com/resources/cheatsheets/

Now to wrangle - Pipes

A pipe allows you to be more efficient with your code by stringing calls together

Means you are able to name objects less

How to trouble shoot? Break it down into sections

REMEMBER TO UNGROUP

Now to wrangle - Loops

We don’t really touch on these, but good to be aware of

Allows you to do something multiple times.

A good example: for every species, run a model correlating population size with year.

Tidyverse removes many cases where you would need loops

But if you end up needing one: for loops and apply functions are what to google

Any Github Issues?

Assessment 1:
Clean-That-Code

Clean That Code

1. We've given you some messy, inefficient code. Time to clean it!

2. Formative (i.e. won't count to your grade) but very important for development!

3. Don't use AI, for the same reason!

Instructions

You'll find instructions in the github link in Week 2 Issue.
Basically:

1. Clean that Code!
2. Be finished by end of play Tuesday 7th, leaving time for feedback...
3. Also by the end of that Tuesday, make sure you add your 'Providing Feedback' partner to your repo (See 'Feedback List' in repo)
4. From Wednesday 8th, go give feedback
5. Feedback and final edits due midday Friday 10th Oct, when the repos will freeze
6. After this we'll release the full list of corrections, a prize for anyone that discovers efficiencies we didn't

Re Feedback:

See the Assessment README for the feedback table:

Receiving	Providing
Hannah	James

e.g. in this case James is providing feedback to Hannah

Re Feedback:

Acceptable way to provide feedback: raise an issue in their repo, and write your comments

Re Feedback:

Acceptable way to provide feedback: raise an issue in their repo, and write your comments

Gold star way to provide feedback: make a pull request, comment code directly, push back

Re Feedback:

What is good feedback?

Re Feedback:

What is good feedback?

- Kind
- Specific
- Actionable

Re Feedback:

Start giving feedback from start of Weds 8th (means you need to have finished your own edits by Tuesday 7th)

Deadline: Next Friday (10th) 12pm NOON

As with future summative assignments, you don't "submit" anything. Just make sure it's all done by the deadline, because at the cutoff github will take a snapshot of whatever's been done! And lock you out.

And now! Activity

Open the data science project, pull the week 3 activity code

Complete your groups work

BUT FIRST – write a workflow, on the whiteboard or pen and paper!

Make sure code is clean, commented, efficient