# Launch Into The Tidyverse! Reproducible Data Analysis in R 🚀

Adam Stone

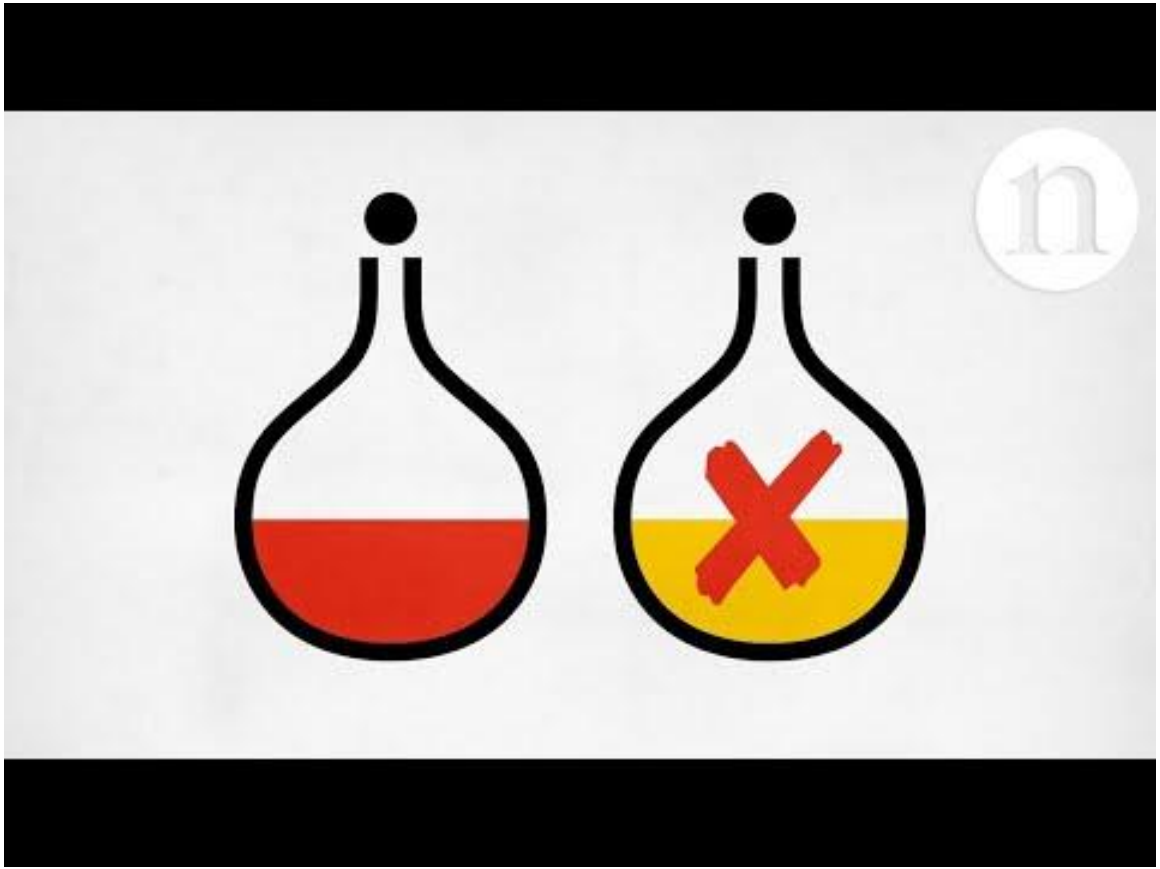
# Monday, March 25 (6-8p)

1. Introduction to reproducible data analysis and why scientists need to do it
2. Rstudio
3. Data visualization with ggplot2


WHALE HELLO THERE


Come on, ask questions.
This is so awkward.


# Introduction to reproducible data analysis and why scientists need to do it



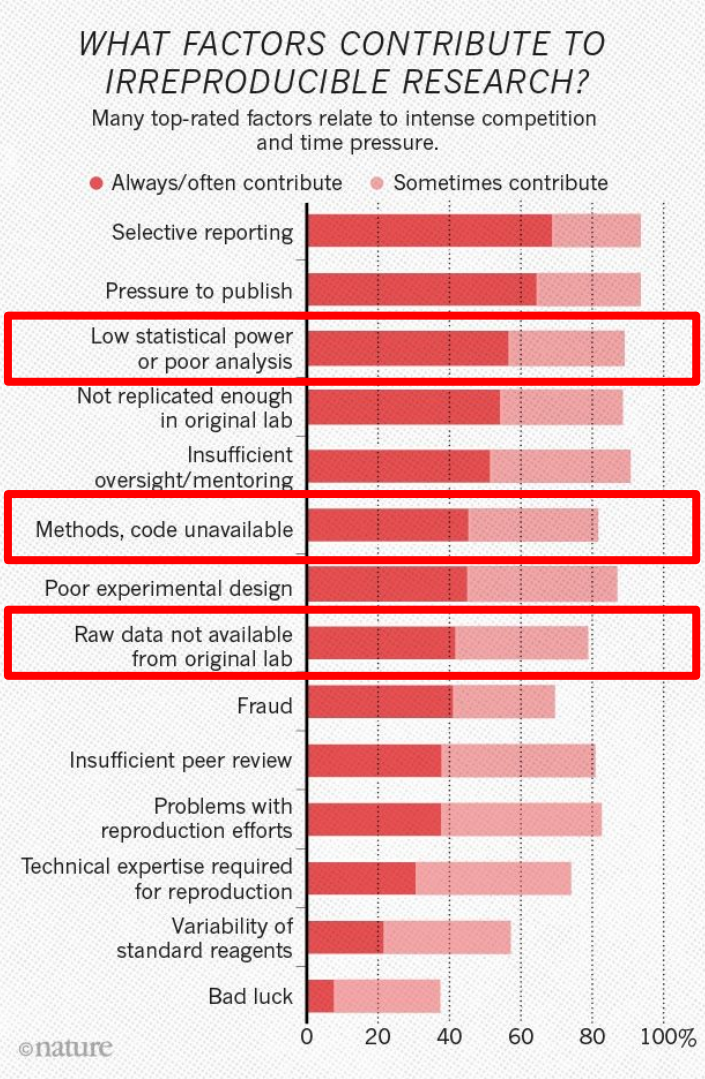
Baker, M. (2016). 1,500 scientists lift the lid on reproducibility. *Nature*, *533*(7604), 452–454.

# What’s Reproducible Science?

# The “R” Word


- **Methods reproducibility**: Can I do this study again?
- **Results reproducibility**: Can I get the same results?
- **Inferential reproducibility**: Can I get the same conclusion from a re-analysis or a new study?

Plesser, H. E. (2017). Reproducibility vs. Replicability: A Brief History of a Confused Terminology. *Frontiers in Neuroinformatics*, *11*, 76.




# Reproducible science is the opposite of all of these

*(what we'll learn about this week.)*

Baker (2016) - https://www.nature.com/news/1-500-scientists-lift-the-lid-on-reproducibility-1.19970

# Methods Reproducibility

**By using reproducible workflows, you:**

**Show evidence of the correctness of your results**

**Enable others to make use of your methods and results**


https://ropensci.github.io/reproducibility-guide/sections/introduction/

# Methods R

**Show ev**
**correct**
**r**

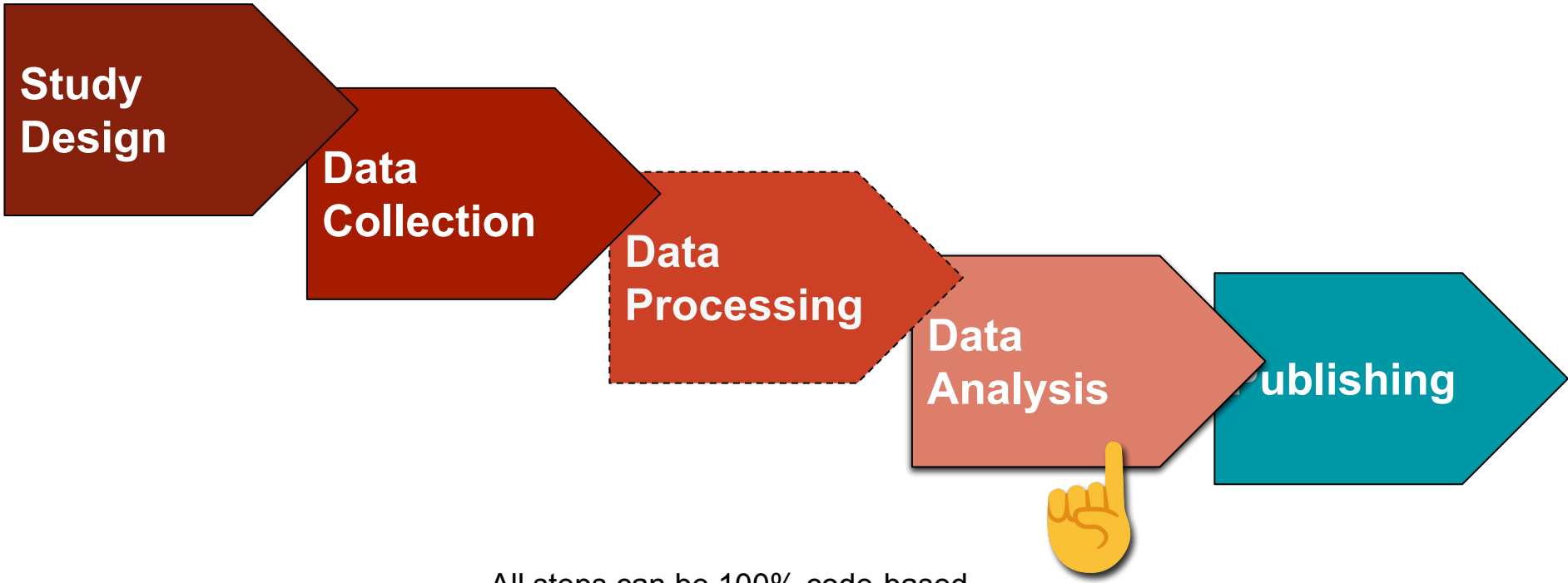

**ers to make**
**r methods**
**esults**

ROpenSci

https://ropensci.github.io/reproducibility-guide/sections/introduction/

# Code!

# Wait, our code goes where?

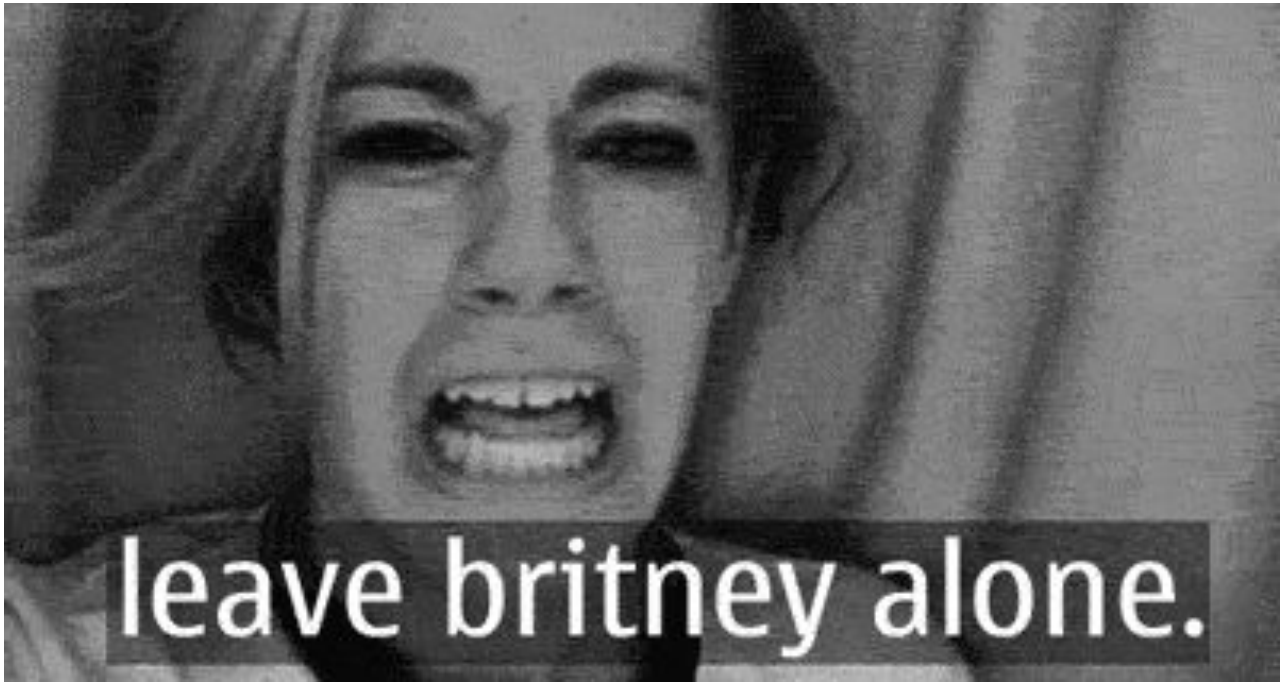



All steps can be 100% code-based.
If you use computers to design and collect data, that already code-based (more or less)
You can also publish using code! (But that's for another workshop...)

# Advantages of Coding (and doing it in R)

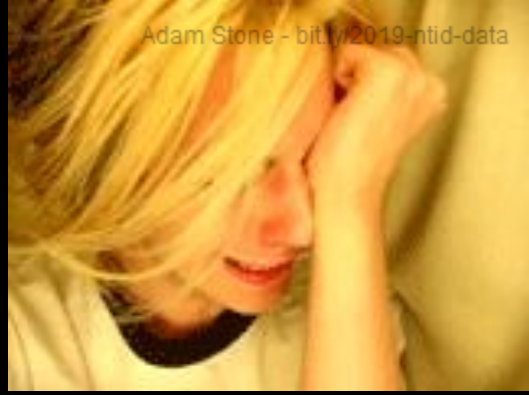
1. **Code is independent of data**
2. **Code is documentation**
3. Code enhances and outlives your research team
4. Code handles large or in-progress datasets
5. Code generates the visualizations that you actually want
6. R has an amazing community
7. You become a coder

1. Code is independent of data



1. Code is independent of data

# 1. Code is independent of data




**Your data is sacrosanct.** You shouldn't ever mess with it.

Code lets you analyze data without *touching* it (accidentally or intentionally).

It follows that we shouldn't be analyzing data using spreadsheets...

# JPMorgan Discloses $2 Billion in Trading Losses

BY JESSICA SILVER-GREENBERG AND PETER EAVIS MAY 10, 2012 10:11 PM 421

https://dealbook.nytimes.com/2012/05/10/jpmorgan-discloses-significant-losses-in-trading-group/

“Operated through a series of Excel spreadsheets...by a process of **copying and pasting** data from one spreadsheet to another"

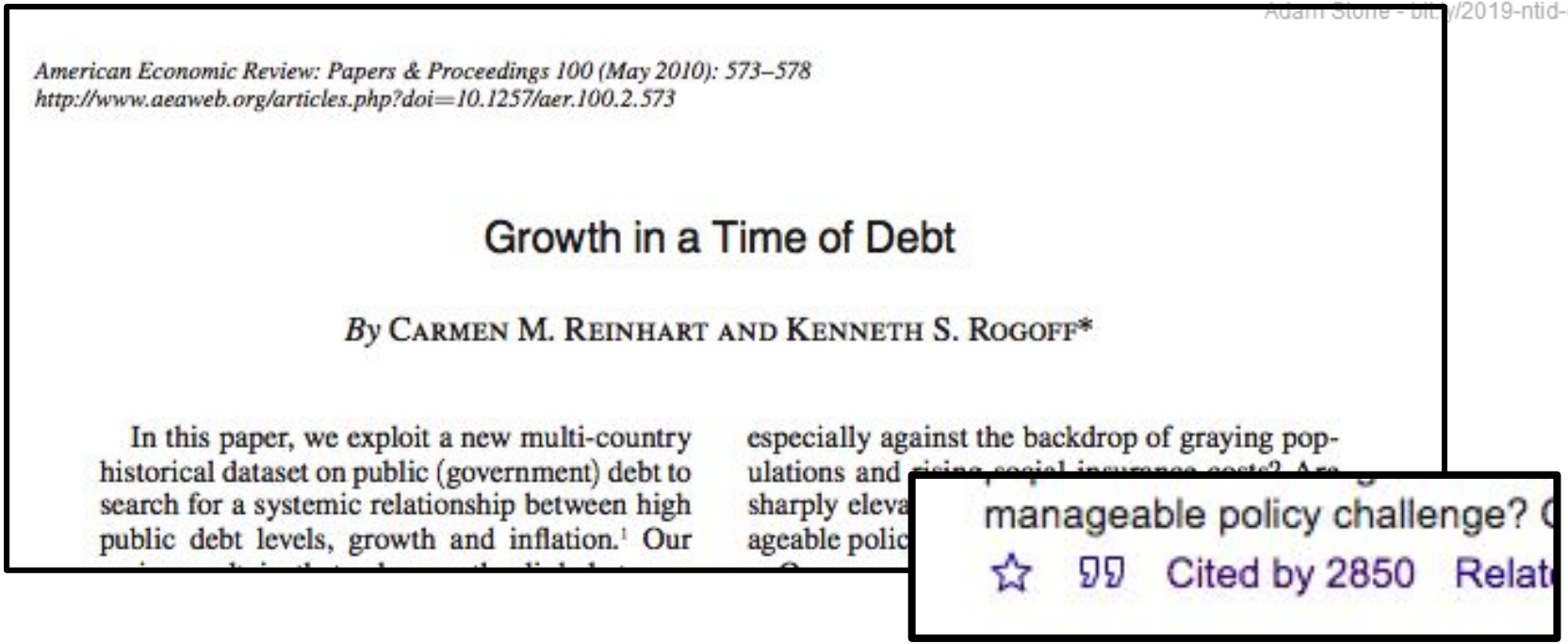
“After subtracting the old rate from the new rate, the spreadsheet divided by their sum **instead of their average**, as the modeler had intended...”

https://blog.revolutionanalytics.com/2013/02/did-an-excel-error-bring-down-the-london-whale.html

# What's the formula?

| x | y | result |
|---|---|---|
| 2 | 5 | 21 |
| 3 | 12 | 20 |

Is the formula the same on both rows?



*American Economic Review: Papers & Proceedings 100 (May 2010): 573–578*
*http://www.aeaweb.org/articles.php?doi=10.1257/aer.100.2.573*

## Growth in a Time of Debt

*By* CARMEN M. REINHART AND KENNETH S. ROGOFF*

In this paper, we exploit a new multi-country historical dataset on public (government) debt to search for a systemic relationship between high public debt levels, growth and inflation.[1] Our

especially against the backdrop of graying populations and

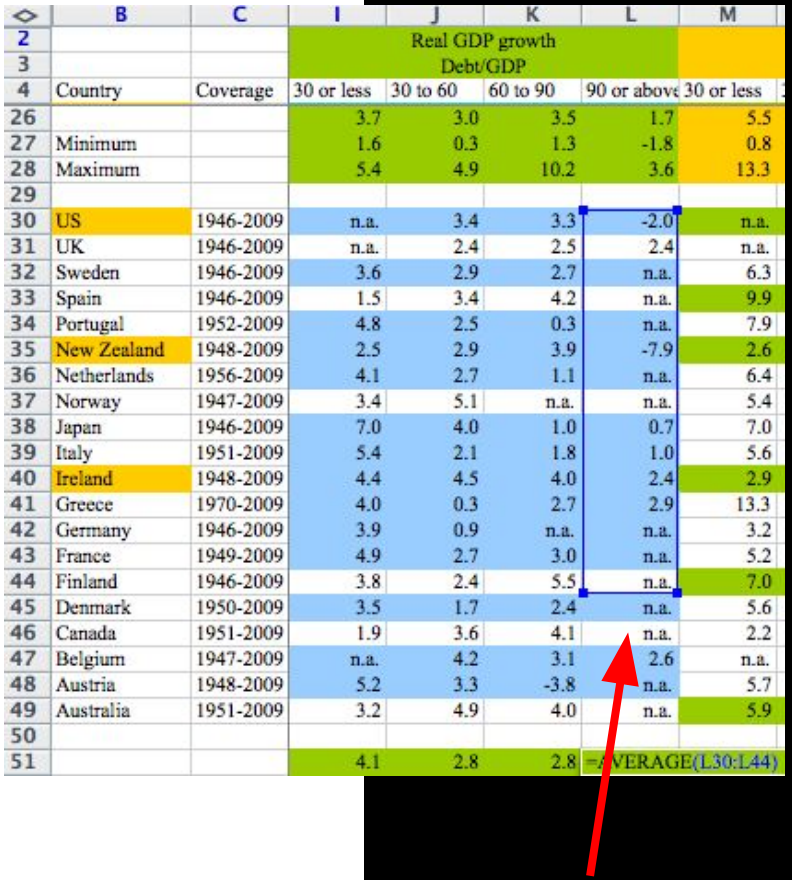
sharply

ageable

manageable policy challenge?

Cited by 2850

*"Our main finding is that across both advanced countries and emerging markets, high debt/GDP levels (>90%) are associated with notably lower growth outcomes."*



| | B | C | I | J | K | L | M |
|---|---|---|---|---|---|---|---|
| 2 | | | Real GDP growth | | | | |
| 3 | | | Debt/GDP | | | | |
| 4 | Country | Coverage | 30 or less | 30 to 60 | 60 to 90 | 90 or above | 30 or less |
| 26 | | | 3.7 | 3.0 | 3.5 | 1.7 | 5.5 |
| 27 | Minimum | | 1.6 | 0.3 | 1.3 | -1.8 | 0.8 |
| 28 | Maximum | | 5.4 | 4.9 | 10.2 | 3.6 | 13.3 |
| 29 | | | | | | | |
| 30 | US | 1946-2009 | n.a. | 3.4 | 3.3 | -2.0 | n.a. |
| 31 | UK | 1946-2009 | n.a. | 2.4 | 2.5 | 2.4 | n.a. |
| 32 | Sweden | 1946-2009 | 3.6 | 2.9 | 2.7 | n.a. | 6.3 |
| 33 | Spain | 1946-2009 | 1.5 | 3.4 | 4.2 | n.a. | 9.9 |
| 34 | Portugal | 1952-2009 | 4.8 | 2.5 | 0.3 | n.a. | 7.9 |
| 35 | New Zealand | 1948-2009 | 2.5 | 2.9 | 3.9 | -7.9 | 2.6 |
| 36 | Netherlands | 1956-2009 | 4.1 | 2.7 | 1.1 | n.a. | 6.4 |
| 37 | Norway | 1947-2009 | 3.4 | 5.1 | n.a. | n.a. | 5.4 |
| 38 | Japan | 1946-2009 | 7.0 | 4.0 | 1.0 | 0.7 | 7.0 |
| 39 | Italy | 1951-2009 | 5.4 | 2.1 | 1.8 | 1.0 | 5.6 |
| 40 | Ireland | 1948-2009 | 4.4 | 4.5 | 4.0 | 2.4 | 2.9 |
| 41 | Greece | 1970-2009 | 4.0 | 0.3 | 2.7 | 2.9 | 13.3 |
| 42 | Germany | 1946-2009 | 3.9 | 0.9 | n.a. | n.a. | 3.2 |
| 43 | France | 1949-2009 | 4.9 | 2.7 | 3.0 | n.a. | 5.2 |
| 44 | Finland | 1946-2009 | 3.8 | 2.4 | 5.5 | n.a. | 7.0 |
| 45 | Denmark | 1950-2009 | 3.5 | 1.7 | 2.4 | n.a. | 5.6 |
| 46 | Canada | 1951-2009 | 1.9 | 3.6 | 4.1 | n.a. | 2.2 |
| 47 | Belgium | 1947-2009 | n.a. | 4.2 | 3.1 | 2.6 | n.a. |
| 48 | Austria | 1948-2009 | 5.2 | 3.3 | -3.8 | n.a. | 5.7 |
| 49 | Australia | 1951-2009 | 3.2 | 4.9 | 4.0 | n.a. | 5.9 |
| 50 | | | | | | | |
| 51 | | | 4.1 | 2.8 | 2.8 | =AVERAGE(L30:L44) | |

“Reinhard and Rogoff's paper contained a simple calculation error in the Excel file...the formula calculating the average only included rows 30-44, rather than rows 30-49.”

http://www.mondaq.com/australia/x/235822/Forensic+Accounting/Spotting+the+mistake+how+a+simple+spreadsheet+error+undermined+a+political+movement

Not saying you *can't* use Excel. Lots of data live in spreadsheets, and that's fine. Just don't analyze data in the spreadsheet.

**It's too easy to change data.** Even if you're being really smart and careful, saving different copies...

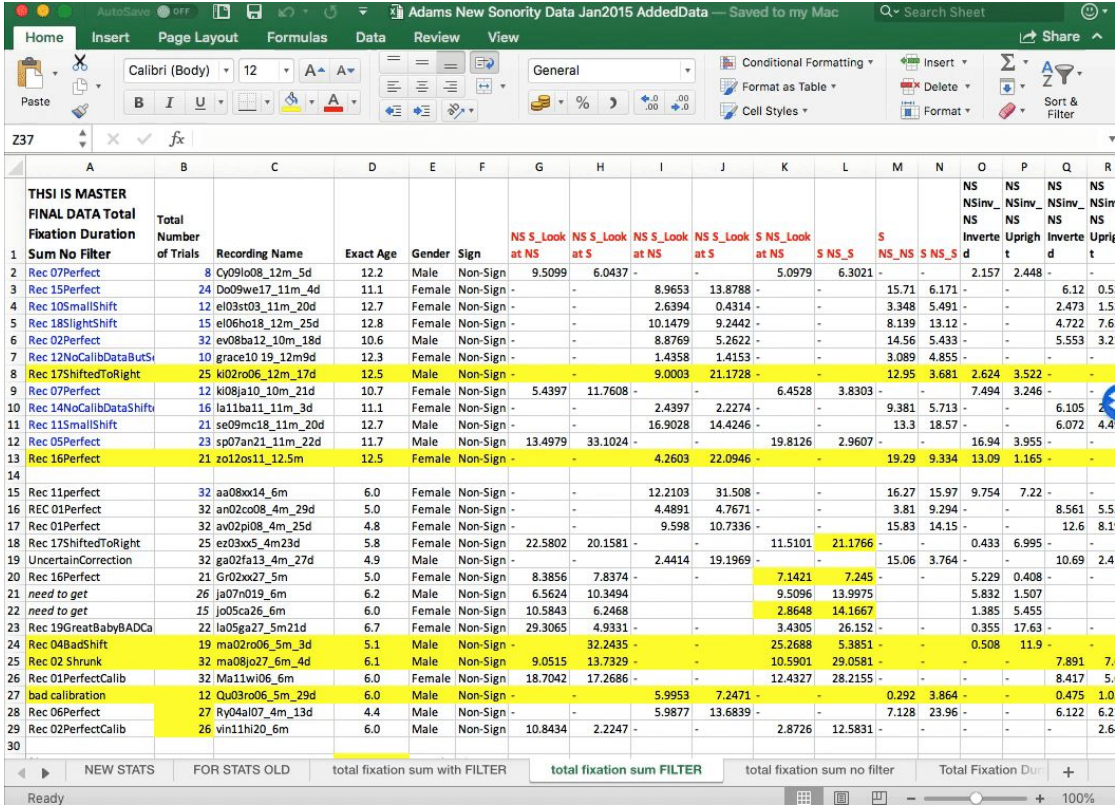
# 2. Code is documentation

This follows from **leaving data alone**:

> *“But what if I need to change some values, drop rows, or clean up missing/incorrect data?”*

Do it using code! Now it’s documented. Even better if you commented on **why** you did it that way.

# Which step is more reproducible?



Adams New Sonority Data Jan2015 AddedData — Saved to my Mac

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | THSI IS MASTER FINAL DATA Total Fixation Duration Sum No Filter | Total Number of Trials | Recording Name | Exact Age | Gender | Sign | NS S_Look at NS | NS S_Look at S | NS S_Look at NS | NS S_Look at S | S NS_Look at NS | S NS_S | S NS_NS | S NS_S | NS NSinv_ NS Inverted | NS NSinv_ NS Upright | NS NSinv_ NS Inverted | NS NSinv NS Uprig |
| 2 | Rec 07Perfect | 8 | Cy09lo08_12m_5d | 12.2 | Male | Non-Sign | 9.5099 | 6.0437 | - | - | 5.0979 | 6.3021 | - | - | 2.157 | 2.448 | - | - |
| 3 | Rec 15Perfect | 24 | Do09we17_11m_4d | 11.1 | Female | Non-Sign | - | - | 8.9653 | 13.8788 | - | - | 15.71 | 6.171 | - | - | 6.12 | 0.5 |
| 4 | Rec 10SmallShift | 12 | el03st03_11m_20d | 12.7 | Female | Non-Sign | - | - | 2.6394 | 0.4314 | - | - | 3.348 | 5.491 | - | - | 2.473 | 1.5 |
| 5 | Rec 18SlightShift | 15 | el06ho18_12m_25d | 12.8 | Female | Non-Sign | - | - | 10.1479 | 9.2442 | - | - | 8.139 | 13.12 | - | - | 4.722 | 7.6 |
| 6 | Rec 02Perfect | 32 | ev08ba12_10m_18d | 10.6 | Male | Non-Sign | - | - | 8.8769 | 5.2622 | - | - | 14.56 | 5.433 | - | - | 5.553 | 3.2 |
| 7 | Rec 12NoCalibDataButS | 10 | grace10 19_12m9d | 12.3 | Female | Non-Sign | - | - | 1.4358 | 1.4153 | - | - | 3.089 | 4.855 | - | - | - | - |
| 8 | Rec 17ShiftedToRight | 25 | ki02ro06_12m_17d | 12.5 | Male | Non-Sign | - | - | 9.0003 | 21.1728 | - | - | 12.95 | 3.681 | 2.624 | 3.522 | - | - |
| 9 | Rec 07Perfect | 12 | ki08ja10_10m_21d | 10.7 | Female | Non-Sign | 5.4397 | 11.7608 | - | - | 6.4528 | 3.8303 | - | - | 7.494 | 3.246 | - | - |
| 10 | Rec 14NoCalibDataShift | 16 | la11ba11_11m_3d | 11.1 | Female | Non-Sign | - | - | 2.4397 | 2.2274 | - | - | 9.381 | 5.713 | - | - | 6.105 | 2 |
| 11 | Rec 11SmallShift | 21 | se09mc18_11m_20d | 12.7 | Male | Non-Sign | - | - | 16.9028 | 14.4246 | - | - | 13.3 | 18.57 | - | - | 6.072 | 4.4 |
| 12 | Rec 05Perfect | 23 | sp07an21_11m_22d | 11.7 | Male | Non-Sign | 13.4979 | 33.1024 | - | - | 19.8126 | 2.9607 | - | - | 16.94 | 3.955 | - | - |
| 13 | Rec 16Perfect | 21 | zo12os11_12.5m | 12.5 | Female | Non-Sign | - | - | 4.2603 | 22.0946 | - | - | 19.29 | 9.334 | 13.09 | 1.165 | - | - |
| 14 | | | | | | | | | | | | | | | | | | |
| 15 | Rec 11perfect | 32 | aa08xx14_6m | 6.0 | Female | Non-Sign | - | - | 12.2103 | 31.508 | - | - | 16.27 | 15.97 | 9.754 | 7.22 | - | - |
| 16 | REC 01Perfect | 32 | an02co08_4m_29d | 5.0 | Female | Non-Sign | - | - | 4.4891 | 4.7671 | - | - | 3.81 | 9.294 | - | - | 8.561 | 5.5 |
| 17 | Rec 01Perfect | 32 | av02pi08_4m_25d | 4.8 | Female | Non-Sign | - | - | 9.598 | 10.7336 | - | - | 15.83 | 14.15 | - | - | 12.6 | 8.1 |
| 18 | Rec 17ShiftedToRight | 25 | ez03xx5_4m23d | 5.8 | Female | Non-Sign | 22.5802 | 20.1581 | - | - | 11.5101 | 21.1766 | - | - | 0.433 | 6.995 | - | - |
| 19 | UncertainCorrection | 32 | ga02fa13_4m_27d | 4.9 | Male | Non-Sign | - | - | 2.4414 | 19.1969 | - | - | 15.06 | 3.764 | - | - | 10.69 | 2.4 |
| 20 | Rec 16Perfect | 21 | Gr02xx27_5m | 5.0 | Female | Non-Sign | 8.3856 | 7.8374 | - | - | 7.1421 | 7.245 | - | - | 5.229 | 0.408 | - | - |
| 21 | *need to get* | *26* | ja07n019_6m | 6.2 | Male | Non-Sign | 6.5624 | 10.3494 | | | 9.5096 | 13.9975 | | | 5.832 | 1.507 | | |
| 22 | *need to get* | *15* | jo05ca26_6m | 6.0 | Female | Non-Sign | 10.5843 | 6.2468 | | | 2.8648 | 14.1667 | | | 1.385 | 5.455 | | |
| 23 | Rec 19GreatBabyBADCa | 22 | la05ga27_5m21d | 6.7 | Female | Non-Sign | 29.3065 | 4.9331 | - | - | 3.4305 | 26.152 | - | - | 0.355 | 17.63 | - | - |
| 24 | Rec 04BadShift | 19 | ma02ro06_5m_3d | 5.1 | Male | Non-Sign | - | 32.2435 | - | - | 25.2688 | 5.3851 | - | - | 0.508 | 11.9 | - | - |
| 25 | Rec 02 Shrunk | 32 | ma08jo27_6m_4d | 6.1 | Male | Non-Sign | 9.0515 | 13.7329 | - | - | 10.5901 | 29.0581 | - | - | - | - | 7.891 | 7. |
| 26 | Rec 01PerfectCalib | 32 | Ma11wi06_6m | 6.0 | Female | Non-Sign | 18.7042 | 17.2686 | - | - | 12.4327 | 28.2155 | - | - | - | - | 8.417 | 5. |
| 27 | bad calibration | 12 | Qu03ro06_5m_29d | 6.0 | Male | Non-Sign | - | - | 5.9953 | 7.2471 | - | - | 0.292 | 3.864 | - | - | 0.475 | 1.0 |
| 28 | Rec 06Perfect | 27 | Ry04al07_4m_13d | 4.4 | Male | Non-Sign | - | - | 5.9877 | 13.6839 | - | - | 7.128 | 23.96 | - | - | 6.122 | 6.2 |
| 29 | Rec 02PerfectCalib | 26 | vin11hi20_6m | 6.0 | Male | Non-Sign | 10.8434 | 2.2247 | - | - | 2.8726 | 12.5831 | - | - | - | - | - | 2.6 |
| 30 | | | | | | | | | | | | | | | | | | |

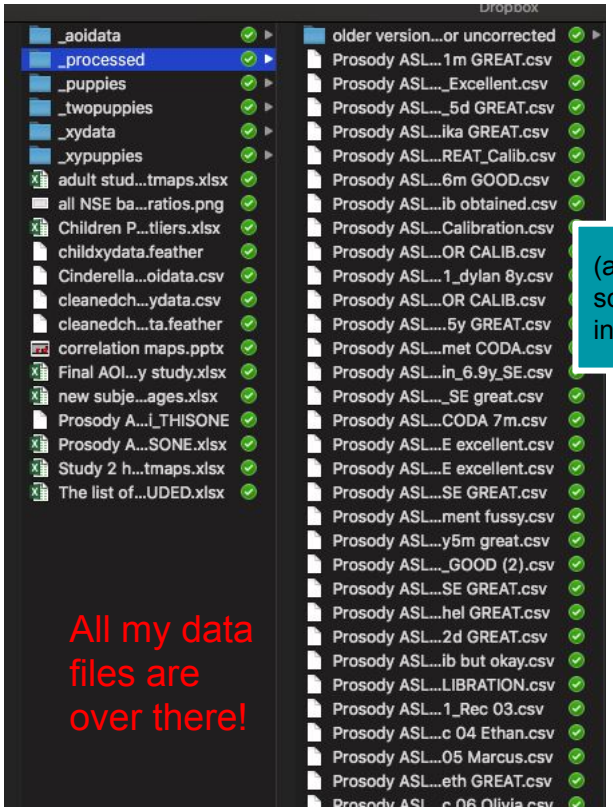
NEW STATS | FOR STATS OLD | total fixation sum with FILTER | total fixation sum FILTER | total fixation sum no filter | Total Fixation Dur

Ready 100%

```
# zo12os11 removed due to fussiness
data %>%
     filter(Recording_Name !=
     “zo12os11_12.5m”)
```

Journals can ask for **all** data collected, not just what you used in your analysis. Can you restore deleted data?

# Now:




All my data files are over there!

(a small script to read in all files)

```
First, get rid of all older than two years old. Then we're going to remove specific babies for various
reasons (usually bad calibration or weird-looking data), here's a list of who we removed.

```{r message=FALSE}
# Libraries
library(tidyverse)
library(feather)
library(stringr)
library(RColorBrewer)
#library(cowplot)

# Import data. If you're adding any new participants you need to run the topmost code block in
# 01importclean.Rmd which will add those to the .feather file. And add the age to childrenages.csv.
data <- read_feather("childrawdata.feather")

# Get ages
ages <- read_csv("childrenages.csv")
data <- data %>% left_join(ages, by = "participant")
#data %>% select(participant,language,age) %>% distinct() # print data table

# # Histogram of ages
# data %>% select(participant,language,age) %>%
#   distinct() %>%
#   ggplot(aes(x = age)) + geom_histogram(fill = "royalblue") + ggtitle("Ages in Full Dataset")

data <- data %>%
  filter(age < 2.0) %>%
  filter(participant != "AsherCalibOnly") %>%
  filter(participant != "do09ne07_6m_4d") %>%
  filter(participant != "Em12ad10_14m_24d") %>%
  filter(participant != "ka11es12_7m_MomTerp") %>%
  filter(participant != "boy 6 m SHIFTED") %>%
  filter(participant != "ma01wa22_10m") %>%
  filter(participant != "NI12GO05_5M_4D") %>%
  filter(participant != "Sara8monthsDeafCODA")  %>%
  filter(participant != "Gianna_CODA_18m")
```

My documentation is here!

I'm filtering out by age and fussiness here!
```

# You will have to revisit your data!


You publish the protocol of studies you've previously published, so others can learn how you did it.

***I had to look up how many participants we tested...***

# Me to my 2016 self:


A method

e previously

I had to look up how many participants we tested...

# 2. Code is documentation

Code will contain all your past decisions. So it's easier to remember how you did it last time **-- and why**.

Your code contains the steps in your analysis. Now it's really easy to write that up in a paper!

**Well-documented code is a gift to your future self.**

# Garbage in, garbage out

# Advantages of Coding (and doing it in R)

1. **Code is independent of data ✓**
2. **Code is documentation ✓**
3. Code enhances and outlives your research team
4. Code handles in-progress datasets
5. Code generates the visualizations that you actually want
6. R has an amazing community
7. You become a coder

# 3. Code enhances and outlives your research team

Researchers and RAs always will be doing *some* degree of manual work. **Are you sure they're doing it right, every single time? Did they understand the protocol?**

More code = simpler and safer. RAs can get up to speed more quickly. Turnover isn't as much of a problem.

# 4. Code handles in-progress datasets


**In-Progress:** You can write the code before you've collected all your participants. Then you just add files to a folder, and let the code find the new data and update everything. No copy/pasting rows. That's great, isn't it?

# 5. Get the visualizations you actually want


How many hours have you spent fighting Excel or Google or SPSS charts, just to get a simple bar or line chart? What about adding error bars? Forget trying anything more complex than that.

In R code, once you understand the “grammar of graphics” (ggplot2), it’s amazing!

# 6. R has an amazing community.


## How to find the community

Everyone seemed to be in agreement that (1) the community is one of R's biggest strengths and (2) a lot within the R community happens on twitter. During discussion, Julia Lowndes mentioned she joined twitter because she heard that people asked and answered questions about R there, and others echoed this sentiment. Simply, the R community is *not* just for 'power users' or developers. It's a place for users and people interested in learning more about R. So, if you want to get involved in

Julia Silge and 7 others liked

**Jasmine Daly** @jasdumas · 24 Jun 2017

I learned Matlab, C++, & **Python** before R, but never felt like I belonged until the #rstats **community** embraced me! 🙌☺️🎉

Malcolm Barrett 🦁🐻 (AFK) and 2 others liked

**Miles McBain** @MilesMcBain · 27 Nov 2018

A great look at all the axes of inclusion in #rstats vs **Python**. There's still much to do, but I am really proud of how diverse this **community** is, and it is something I frequently cite when hyping R to potential tidyverts.

## Why Women Are Flourishing In R Community But Lagging In Python

18 minute read

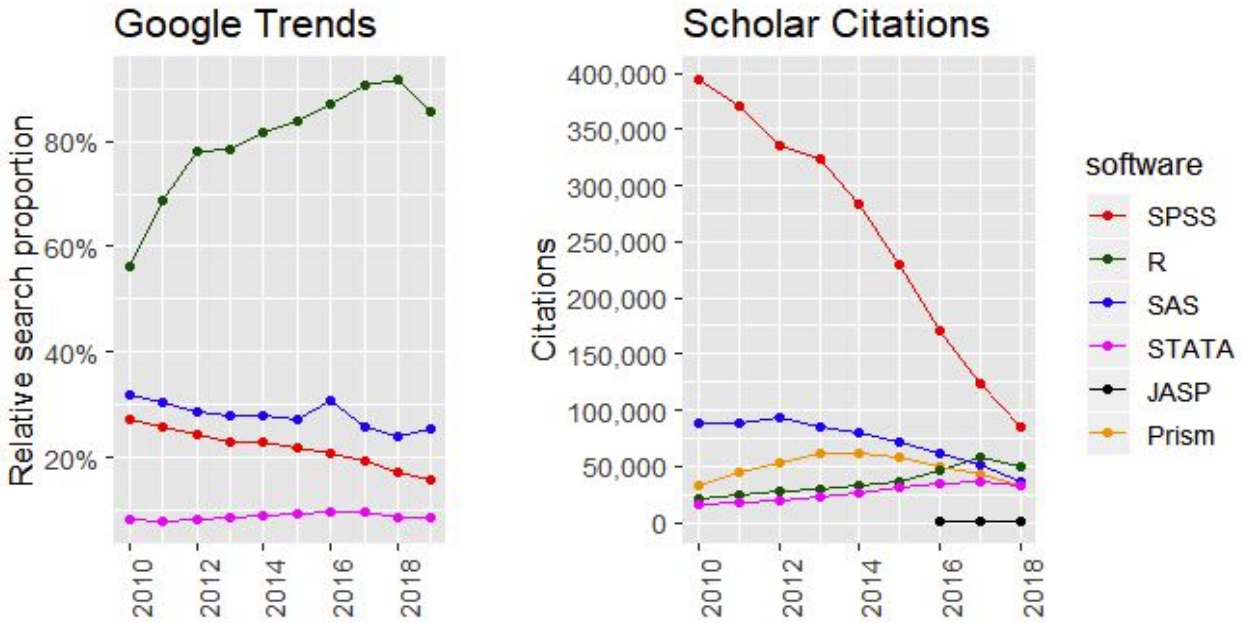
## The R community is one of R's best features

R is incredible software for statistics and data science. But while the bits and bytes of software are an essential component of its usefulness, software needs a **community** to be successful. And that's an area where R really shines, as Shannon Ellis explains in this lovely ROpenSci

https://reshamas.github.io/why-women-are-flourishing-in-r-community-but-lagging-in-python/

https://stackoverflow.blog/2017/10/10/impressive-growth-r/

# Active communities have more resources

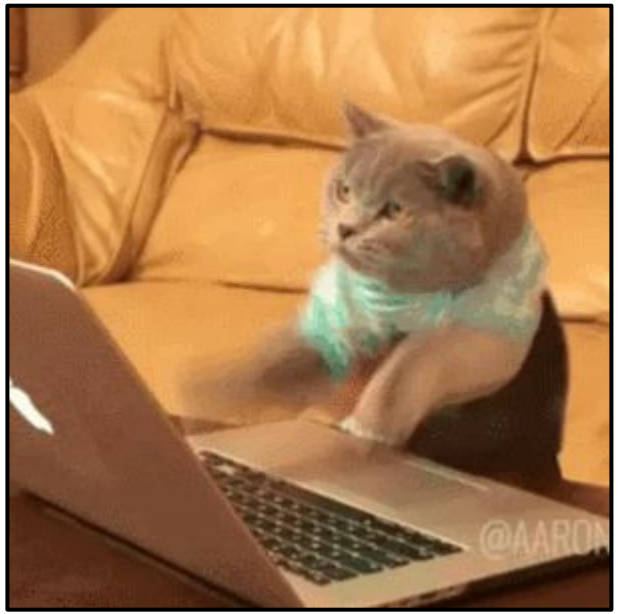



"SPSS is dying. It's time to change" (2019, 13 March) http://lindeloev.net/spss-is-dying/


@allison_horst

# 7. You become a coder

All it takes is one language -- and boom. You’re a coder (or programmer).




# Advantages of Coding (and doing it in R)

1. **Code is independent of data ✓**
2. **Code is documentation ✓**
3. Code enhances and outlives your research team ✓
4. Code handles large or in-progress datasets ✓
5. Code generates the visualizations that you actually want ✓
6. R has an amazing community ✓
7. You become a coder ✓

**And...now you can do reproducible research!**

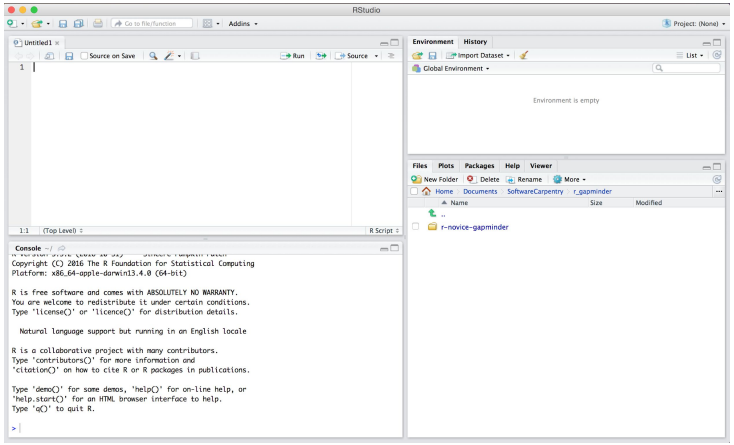
# Let’s code.


at first I was like...
...but now it's like...
Horst '18

# R v. RStudio v. Tidyverse

```
# My first program in R
myString <- "Hello, World!"

print (myString)
```





```
data %>%
     filter(age > 2)
```

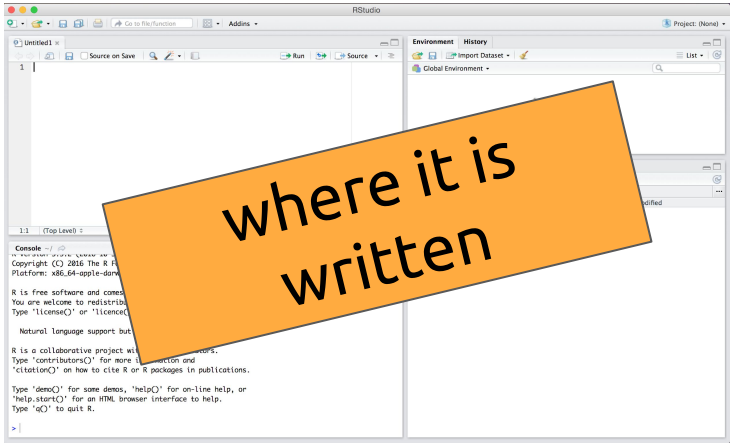
r-project.org

rstudio.com


tidyverse.org

# R v. RStudio v. Tidyverse

the language




where it is written

one way to write it

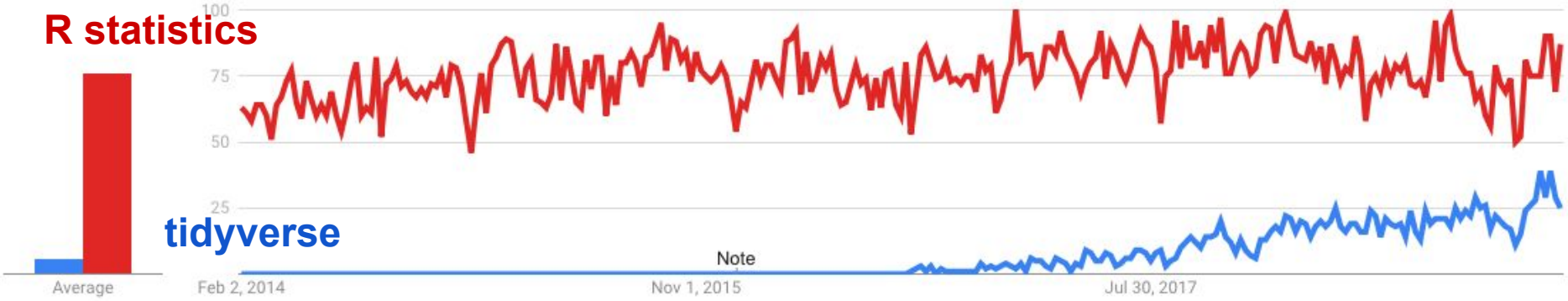
```
data %>%
    filter(age > 2)
```

r-project.org

rstudio.com

tidyverse.org

# The R world has changed a lot!




https://trends.google.com/trends/explore?date=2014-02-02%202019-03-02&q=tidyverse,R%20language

# Packages

Themed collection of functions and datasets for doing something

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






`install.packages("tidyverse")`
(do this only once on a machine)

# `library(tidyverse)`
(do this every time you start R)

https://www.tidyverse.org/packages

**Let’s go into RStudio!**

rstudio.cloud

# Data Visualization with ggplot2


ggplot2:
Build a data
MASTERPIECE
THEMES
gEOMS
aes
fill
color
ggANIMATE
plotly
HORST '18
@allison_horst

In RStudio, open 02_ggplot2_data_visualization.R

We'll do the first Visualization together...

Lesson ideas: Mine Çentinkaya-Rundel, bit.ly/let-eat-cake

# ggplot2 basics

```
internet_usage %>%
  filter(area %in% c('United States of America', 'Canada')) %>%
  ggplot(aes(x = year, y = value, color = area)) +
  geom_line() +
  geom_point()
  scale_y_conti                                              ) +
  labs(x = "yea
       y = "per
       title =
       subtitle
       color = "Country") +
  theme_gray() +
  theme(legend.position = 'right')
```

## Start with the data

*What do you think would happen if I took out the `filter()` argument? (don't do it!)*

# ggplot2 basics

```
internet_usage %>%
  filter(area %in% c('United States of America', 'Canada')) %>%
  ggplot(aes(x = year, y = value, color = area)) +
  geom_line() +
```

In the `ggplot()` function, define the *aesthetics*

Which column goes on the x-axis?
Which column goes on the y-axis?
Which column gets turned into colors?

```
  theme(legend.position = 'right')
```

# ggplot2 basics

```
internet_usage %>%
  filter(area %in% c(
  ggplot(aes(x = year,
  geom_line() +
  geom_point() +
  scale_y_continuous(
  labs(x = "year",
       y = "percentage
       title = "Interr
       subtitle = "200
       color = "Counti
  theme_gray() +
  theme(legend.position = "right")
```

Then add *geom_* layers.

They obey the aesthetics. Each one works a little differently.

https://ggplot2.tidyverse.org/reference/index.html

# ggplot2 basics

```
internet_usage %>%
  filter(area %in% c('United States of America', 'Canada')) %>%
  ggplot(aes(x = year, y = value, color = area)) +
  geom_line() +
  geom_point() +
  scale_y_continuous(limits = c(0,1), labels = scales::percent) +
  labs(x = "year",
       y = "percentage
       title = "Interne
       subtitle = "2000
       color = "Country
  theme_gray() +
  theme(legend.position = 'right')
```

Fix scales as needed...

# ggplot2 basics

```
internet_usage %>%
  filter(area %in% c(
  ggplot(aes(x = year,
  geom_line() +
  geom_point() +
  scale_y_continuous(limits = c(0,1), labels = scales::percent) +
  labs(x = "year",
       y = "percentage of total population",
       title = "Internet Users by Country",
       subtitle = "2000 to 2016",
       color = "Country") +
  theme_gray() +
  theme(legend.position = 'right')
```

Write labels...

# ggplot2 basics

```
internet_usage %>%
  filter(area %in% c('United States of America', 'Canada')) %>%
  ggplot(aes(x = year, y = value, color = area)) +
  geom_lin
  geom_poi
  scale_y_                                                    ent) +
  labs(x =
       y = "percentage of total population",
       title = "Internet Users by Country",
       subtitle = "2000 to 2016",
       color = "Country") +
  theme_gray() +
  theme(legend.position = 'right')
```

And change themes.

# ggplot2 basics

```
internet_usage %>%
  filter(area %in% c('United States of America', 'Canada')) %>%
  ggplot(aes(x = year, y = value, color = area)) +
  geom_line() +
  geom_point() +
  scale_y_continuous(limits = c(0,1), labels = scales::percent) +
  labs(x = "year",
       y = "percentage
       title = "Intern
       subtitle = "200
       color = "Country") +
  theme_gray() +
  theme(legend.position = 'right')
```

No + at the end!

# ggplot_flipbook

```
ggplot(data = dta)
```





https://evamaerey.github.io/ggplot_flipbook/

# ggplot_flipbook


https://evamaerey.github.io/ggplot_flipbook/

# Data Visualization with ggplot2 : : CHEAT SHEET

## Basics

**ggplot2** is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and geoms—visual marks that represent data points.


To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.


## Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

### GRAPHICAL PRIMITIVES

```
a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))
```

**a + geom_blank()**
(Useful for expanding limits)

**b + geom_curve(**aes(yend = lat + 1, xend=long+1,curvature=z)**)** - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

**a + geom_path(**lineend="butt", linejoin="round', linemitre=1**)**
x, y, alpha, color, group, linetype, size

**a + geom_polygon(**aes(group = group)**)**
x, y, alpha, color, fill, group, linetype, size

**b + geom_rect(**aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)**)** - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

**a + geom_ribbon(**aes(ymin=unemploy - 900, ymax=unemploy + 900)**)** - x, ymax, ymin, alpha, color, fill, group, linetype, size

### TWO VARIABLES

**continuous x , continuous y**

```
e <- ggplot(mpg, aes(cty, hwy))
```

**e + geom_label(**aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE**)** x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

**e + geom_jitter(**height = 2, width = 2**)**
x, y, alpha, color, fill, shape, size

**e + geom_point()**, x, y, alpha, color, fill, shape, size, stroke

**e + geom_quantile()**, x, y, alpha, color, group, linetype, size, weight

**e + geom_rug(**sides = "bl"**)**, x, y, alpha, color, linetype, size

**e + geom_smooth(**

**continuous bivariate distribution**

```
h <- ggplot(diamonds, aes(carat, price))
```

**h + geom_bin2d(**binwidth = c(0.25, 500)**)**
x, y, alpha, color, fill, linetype, size, weight

**h + geom_density2d()**
x, y, alpha, colour, group, linetype, size

**h + geom_hex()**
x, y, alpha, colour, fill, size

**continuous function**

```
i <- ggplot(economics, aes(date, unemploy))
```

**i + geom_area()**

# Secret Weapon!

https://www.rstudio.com/resources/cheatsheets/

# Let’s work with some marvelous data

# Main Takeaways

With ggplot2 you build charts piece by piece:

1. Pick the data
2. Map the data to aesthetics
3. Plot geometric objects
4. Define the look (scales, themes, labels)

R Community: loads of themes, geom_ objects, animations, in-app helpers. Google and Twitter are your friends! (e.g., http://www.ggplot2-exts.org/gallery/, https://bbc.github.io/rcookbook/)

If you accidentally make something weirdly beautiful: @accidental_aRt

https://twitter.com/foundinblank/status/1012331346558160896 (ggplot2 charts)

# Launch Into The Tidyverse! Reproducible Data Analysis in R 🚀

Adam Stone

## Day Two!

# Tuesday, March 26 (6-8p)

1. Recap about Reproducible Research (3 mins)

   https://github.com/rdpeng/courses/tree/master/05_ReproducibleResearch/Checklist

2. Manipulating and filtering data with dplyr (1h 10m)
3. Tidying and reshaping data with tidyr (45m)

# Advantages of Coding (and doing it in R)

1. **Code is independent of data ✓**
2. **Code is documentation ✓**
3. Code enhances and outlives your research team ✓
4. Code handles large or in-progress datasets ✓
5. Code generates the visualizations that you actually want ✓
6. R has an amazing community ✓
7. You become a coder ✓

**And...now you can do reproducible research!**

# Approaching R: Having the right attitude

> **"I have been writing R code for years, and every day I still write code that doesn't work!"**

**Wickham & Grolemund (2017: 7)**

Wickham, H. & Grolemund, G (2017). *R for Data Science*. Sebastopol, CA: O'Reilly.

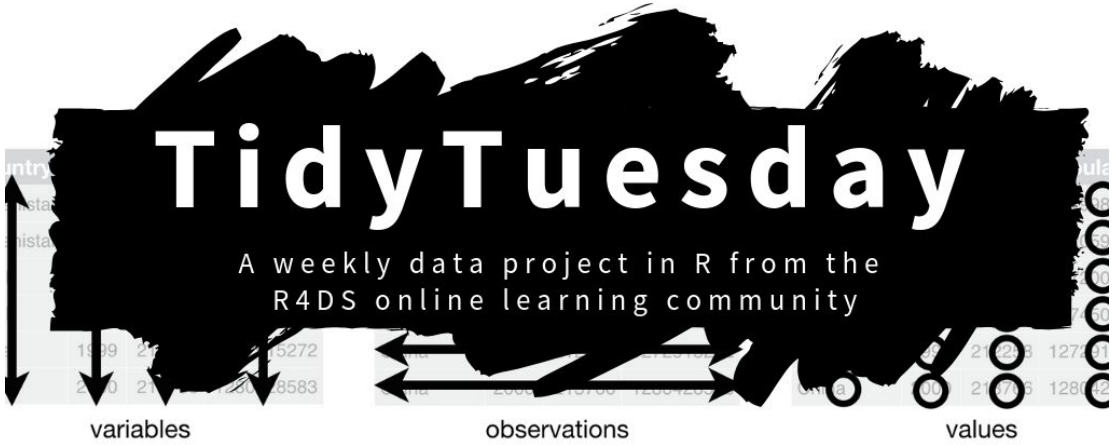
**Neo**
@NeoIghodaro

Follow

I think young developers (at least the ones I've met) get lost in how much time it'll take them to learn. I'll say this once; it will take time. Stop wasting your time worrying about that, nothing you do or say or think will change that. It will take time.

2:42 PM - 5 Feb 2019

**309** Retweets **1,026** Likes




https://github.com/rfordatascience/tidytuesday

# Manipulating with dplyr

# dplyr : go wrangling

“It is often said that 80% of data analysis is spent on the cleaning and preparing data. And it’s not just a first step, but it must be repeated many times over the course of analysis as new problems come to light or new data is collected.”

https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html

# *We're going to be working with clean data.*

Happy families are all alike; every unhappy family is unhappy in its own way

---

Leo Tolstoy

"Like families, tidy datasets are all alike but every messy dataset is messy in its own way." (p. 2)

Wickham, H. (2014). Tidy Data. *Journal of Statistical Software, Articles*, *59*(10), 1–23.

# Data Manipulation

Visualizations are great! But most of the work goes into *manipulating* the data so you can report things like this:

**Table 1** Stimuli descriptive statistics

| | Min | Max | Median | ASL-LEX (median) | *p* |
|---|---|---|---|---|---|
| Minimum neighborhood density | 517 | 923 | 783 | 780 | .826 |
| Maximum neighborhood density | 0 | 82 | 27 | 27 | .294 |
| Parameter-based neighborhood density | 0 | 22 | 3 | 3 | .599 |
| ASL frequency | 2.5 | 6.56 | 4.66 | 4.36 | .384 |
| English frequency | 1.98 | 5.00 | 3.54 | 3.23 | .123 |

Williams, J. T., Stone, A., & Newman, S. D. (2017). Operationalization of Sign Language Phonological Similarity and its Effects on Lexical Access. *Journal of Deaf Studies and Deaf Education*, *22*(3), 303–315.

# Data Manipulation

**Table 1**
Distribution of participants according to six social categories.

| Site | Gender | | Age | | Language background | | Social class | | Ethnicity | |
|---|---|---|---|---|---|---|---|---|---|---|
| | F | M | 16–50 | 51+ | Deaf | Hearing | Working | Middle | White | Other |
| Belfast ($n$ = 30) | 17 | 13 | 14 | 16 | 7 | 23 | 26 | 4 | 30 | - |
| Birmingham ($n$ = 30) | 13 | 17 | 18 | 12 | 12 | 18 | 16 | 14 | 27 | 3 |
| Bristol ($n$ = 32) | 17 | 15 | 15 | 17 | 17 | 15 | 16 | 16 | 30 | 2 |
| Cardiff ($n$ = 29) | 17 | 12 | 14 | 15 | 5 | 24 | 21 | 8 | 27 | 2 |
| Glasgow ($n$ = 30) | 15 | 15 | 16 | 14 | 6 | 24 | 17 | 13 | 27 | 3 |
| London ($n$ = 30) | 15 | 15 | 16 | 14 | 12 | 18 | 11 | 19 | 25 | 5 |
| Manchester ($n$ = 30) | 16 | 14 | 14 | 16 | 9 | 21 | 23 | 7 | 27 | 3 |
| Total | 109 | 102 | 107 | 104 | 68 | 143 | 130 | 81 | 193 | 18 |

FYI....a lot of “data science” is just counting and summarizing!

Fenlon, J., Schembri, A., Rentelis, R., & Cormier, K. (2013). Variation in handshape and orientation in British Sign Language: The case of the “1” hand configuration. *Language & Communication*, *33*, 69–91.

# Data Manipulation

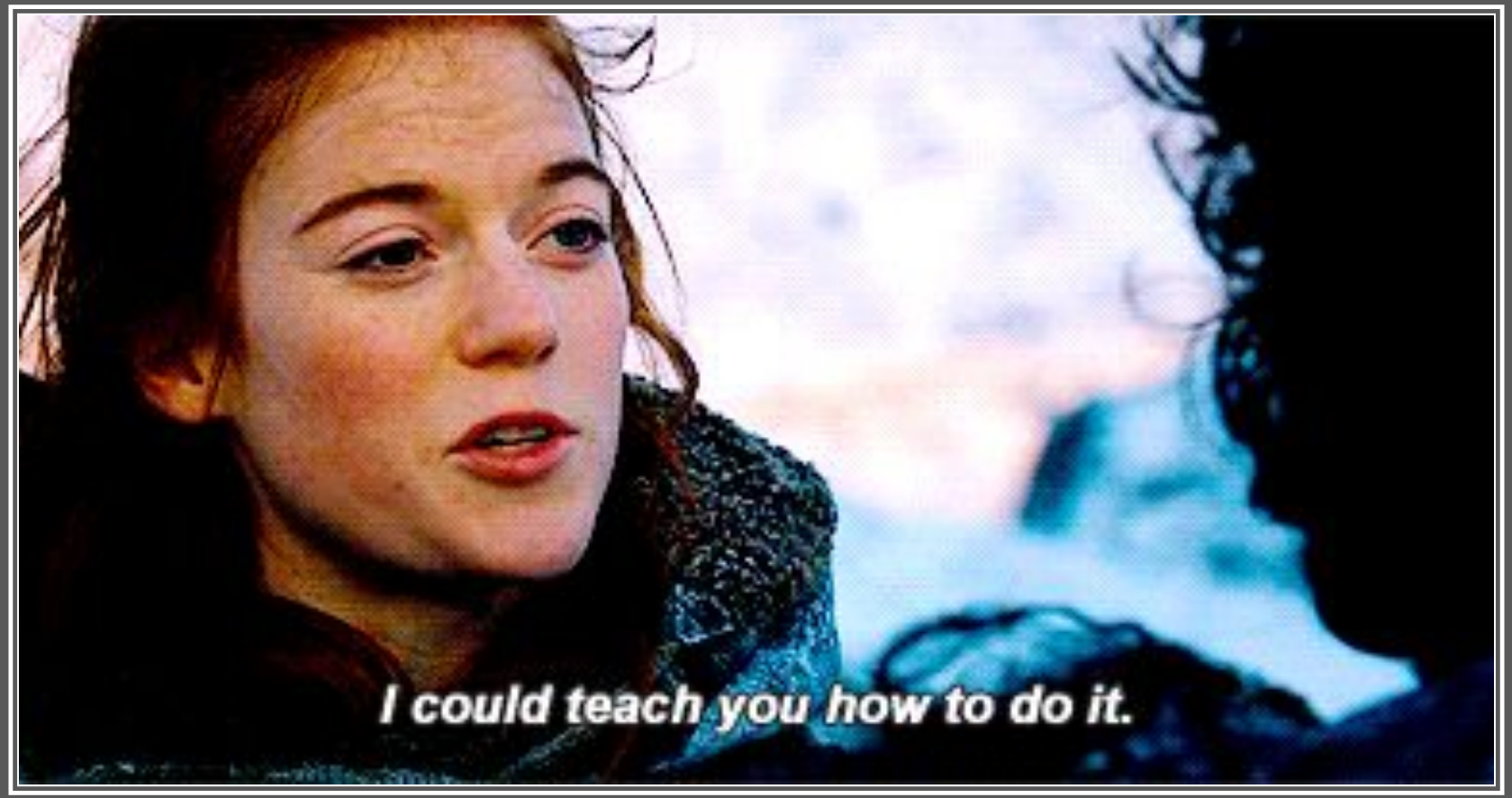
Common approach: use spreadsheets.

But when you add new rows, now you're not sure if the formulas updated or not, right? (Remember the Excel snafus from yesterday!)

Or if you drop a row. Then you go back and think, "why did I drop that row? What was it?" - you've lost data AND your documentation/reasoning.

Code does away with all that! Let's dive in:


FILTER()
SELECT()
ARRANGE()
MUTATE()
GROUP_BY() %>% SUMMARISE()


I could teach you how to do it.

In RStudio, open 03_dplyr_data_manipulation.R

https://injepijournal.biomedcentral.com/articles/10.1186/s40621-018-0174-7

# summarise()

**“What’s the average for each group?”**

**Some things you may want:**

- the mean of all ages
- the median IQ
- the standard deviation of reaction times
- min score
- max score
- the count (number of rows/cases)


You can use summarize() but that conflicts with some packages so summarise() is safer

```
got %>%
    group_by(sex) %>%
    summarise(characters = n())
```

“With the got dataset, group the data into sexes, then return the count of characters in each group.”

# filter()

***“But we want to look only at deceased characters!”***

Include or exclude cases based on values. How?

# Logical Statements

You have to write a **logical** statement.

| | |
|---|---|
| `age > 18` | “all cases with age greater than 18” |
| `iq <= 100` | “all cases with IQ less than or equal to 100” |
| `id == 10` | “all cases where ID is 10” |
| `group != “deaf”` | “all cases where group is not ‘deaf’” |
| `is.na(reaction_time)` | “all cases where reaction_time is blank/NA” |
| `!is.na(country)` | “all cases where country is not blank/NA” |

```
got %>%
    filter(featured_episode_count > 1)
```

“With the got dataset, keep all rows with featured_episode_count greater than 1.”

# The pipe operator

Pop quiz. You've seen **%>%** several times now. What do you think it does?

It's what makes the tidyverse work so well and its code so readable!

# Which do you prefer to read?

```
summarise(filter(filter(got, house == "Targaryen"), dth_flag ==
    "Dead"), avg = mean(exp_episode)))
```

```
got <- filter(got, house == "Targaryen")
got <- filter(got, dth_flag == "Dead")
summarise(got, avg = mean(exp_episode))
```

```
got %>%
    filter(house == "Targaryen") %>%
    filter(dth_flag == "Dead") %>%
    summarise(avg = mean(exp_episode))
```

# Use %>% everywhere...except after ggplot()

```
internet_usage %>%
  filter(area %in% c('United States of America', 'Canada')) %>%
  ggplot(aes(x = year, y = value, color = area)) +
  geom_line() +
  geom_point() +
  scale_y_continuous(limits = c(0,1), labels = scales::percent) +
  labs(x = "year",
       y = "percentage of total population",
       title = "Internet Users by Country",
       subtitle = "2000 to 2016",
       color = "Country") +
  theme_gray() +
  theme(legend.position = 'right')
```

# Creating data objects with `<-`

Pipes can output transformations to new data objects. This is useful and you will probably do this all the time! (We did this yesterday, btw.)

A good workflow:

1. Load all the data into a data object named `raw_data` or `all_data`
2. Filter out cases/participants/trials and output to `final_data`
3. Make summary tables and output to `summary_data`
4. Plot with `final_data` or `summary_data`

Btw, `<-` is called the “assignment operator.” It assigns whatever’s on the right side to the name on the left side.

https://foundinblank.github.io/study2children/03importcleanbabies.nb.html

# select()

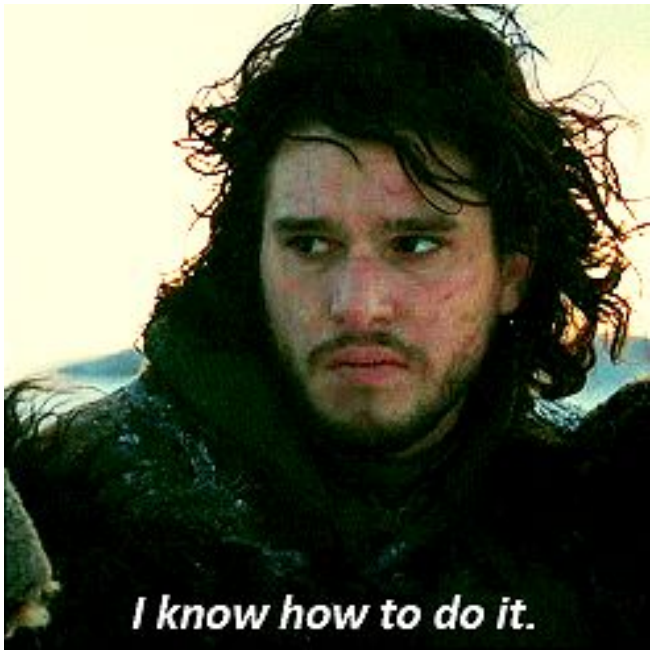
***"I have too many columns!"***

Include, exclude, rearrange columns.

Simpler datasets also reduce cognitive load. That's important.

```
got %>%
    select(name, sex, house)
```

"With the got dataset, keep the name, sex, and house columns (and get rid of everything else)."



# mutate()


***"I need to do some math/functions on every row!"***

It works the same as summarise, except:

- `summarise()` throws away all other columns
- `mutate()` keeps all columns

*Calculate ratios. Convert numbers to log(). Etc.*

"With the got dataset, for every row, calculate a new column called "lifespan_mins" by dividing lifespan_secs by 60."

```
got %>%
    mutate(lifespan_mins = lifespan_secs/60)
```

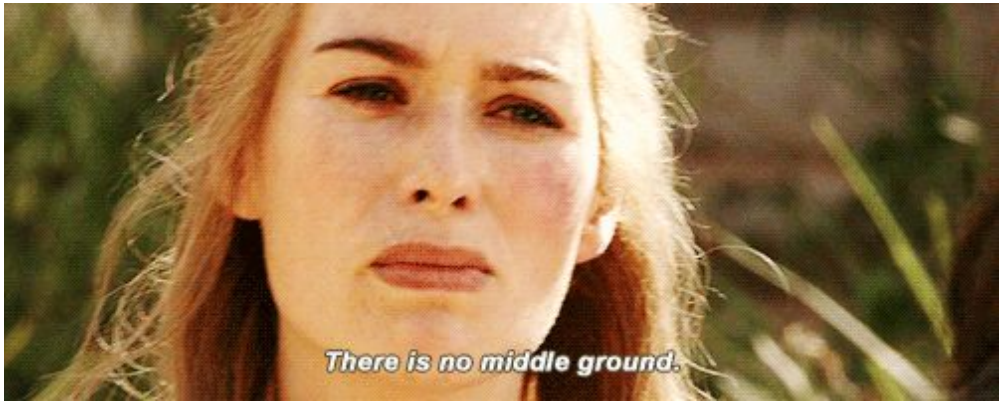
@allison_horst

# arrange()

***"I need rows to be in order!"***

Simple. You may not need it much, but it's one of the core verbs.

*(Usually used after summarise to make tables more presentable. Or with slice() to keep first or last rows in a series)*




```
got %>%
    arrange(intro_episode)
```

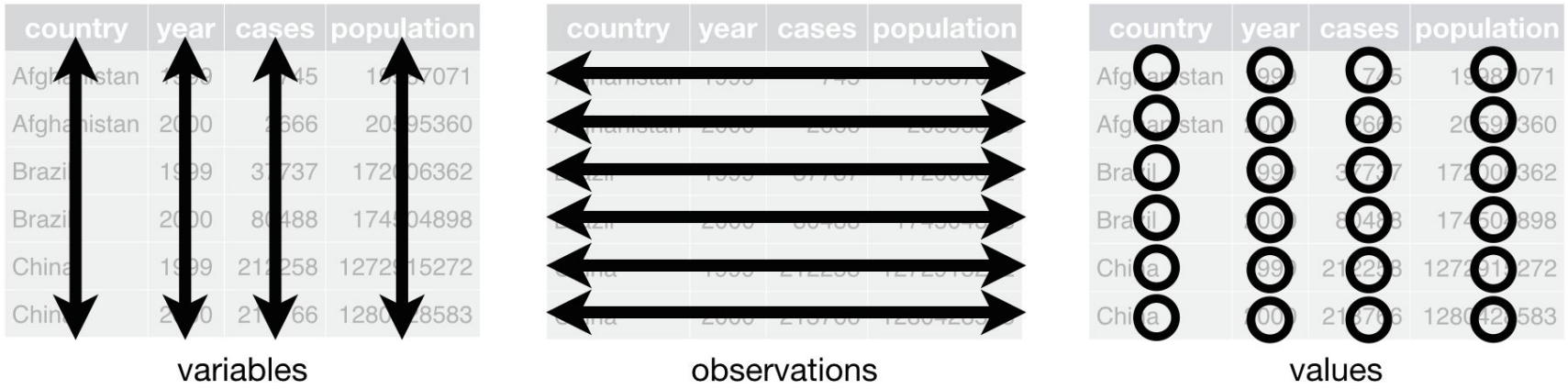
"With the got dataset, order rows by intro_episode."

# Tidying and reshaping with tidyr


tidyr
GATHER
SPREAD

# What is “tidy” data?

- Each row is an observation
- Each column is a variable
- Each value is in its own cell




https://r4ds.had.co.nz/tidy-data.html

# What is “t

- Each row
- Each colu
- Each valu

| country | year | ca | cases | population |
|---|---|---|---|---|
| Afghanistan | 1999 | | 745 | 19987071 |
| Afghanistan | 2000 | | 2666 | 20595360 |
| Brazil | 1999 | 3 | 37737 | 172006362 |
| Brazil | 2000 | 8 | 80488 | 174504898 |
| China | 1999 | 21 | 212258 | 1272915272 |
| China | 2000 | 21 | 213766 | 1280428583 |

variab alues

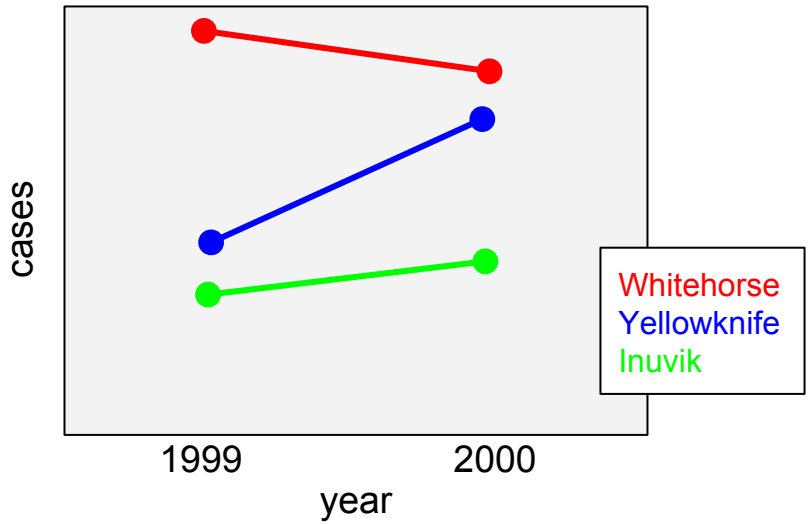
https://r4ds.had.co.nz/tidy-data.html

# What is “tidy” data?

- That wasn’t very clear, right?
- Tidy data tends to be “tall” instead of “wide”
- Think about columns you want to assign to aesthetics in ggplot

| site | 1999 | 2000 |
|---|---|---|
| Whitehorse | 745 | 2666 |
| Yellowknife | 37737 | 80488 |
| Inuvik | 212258 | 213766 |

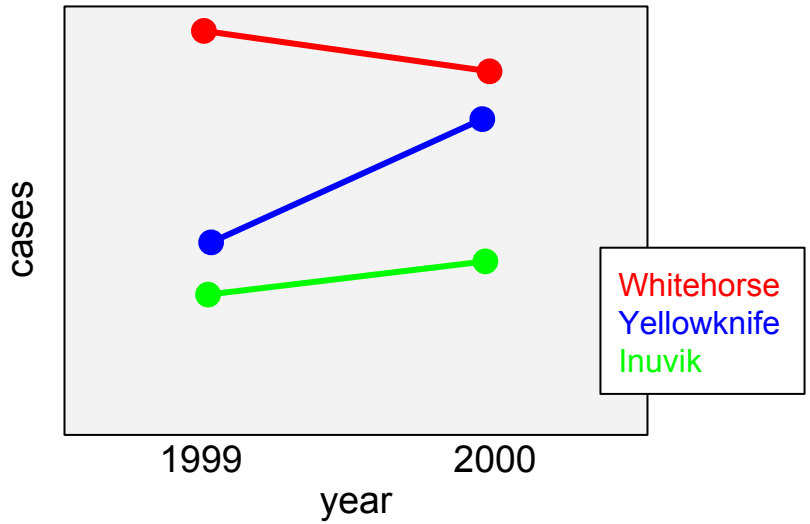
```
aes(x = _____, y = _______, color = _______)
```



# What is “tidy” data?

- That wasn’t very clear, right?
- Tidy data tends to be “tall” instead of “wide”
- Think about columns you want to assign to ggplot aesthetics

| site | year | cases |
|---|---|---|
| Whitehorse | 1999 | 745 |
| Whitehorse | 2000 | 2666 |
| Yellowknife | 1999 | 37737 |
| Yellowknife | 2000 | 80488 |
| Inuvik | 1999 | 212258 |
| Inuvik | 2000 | 213766 |



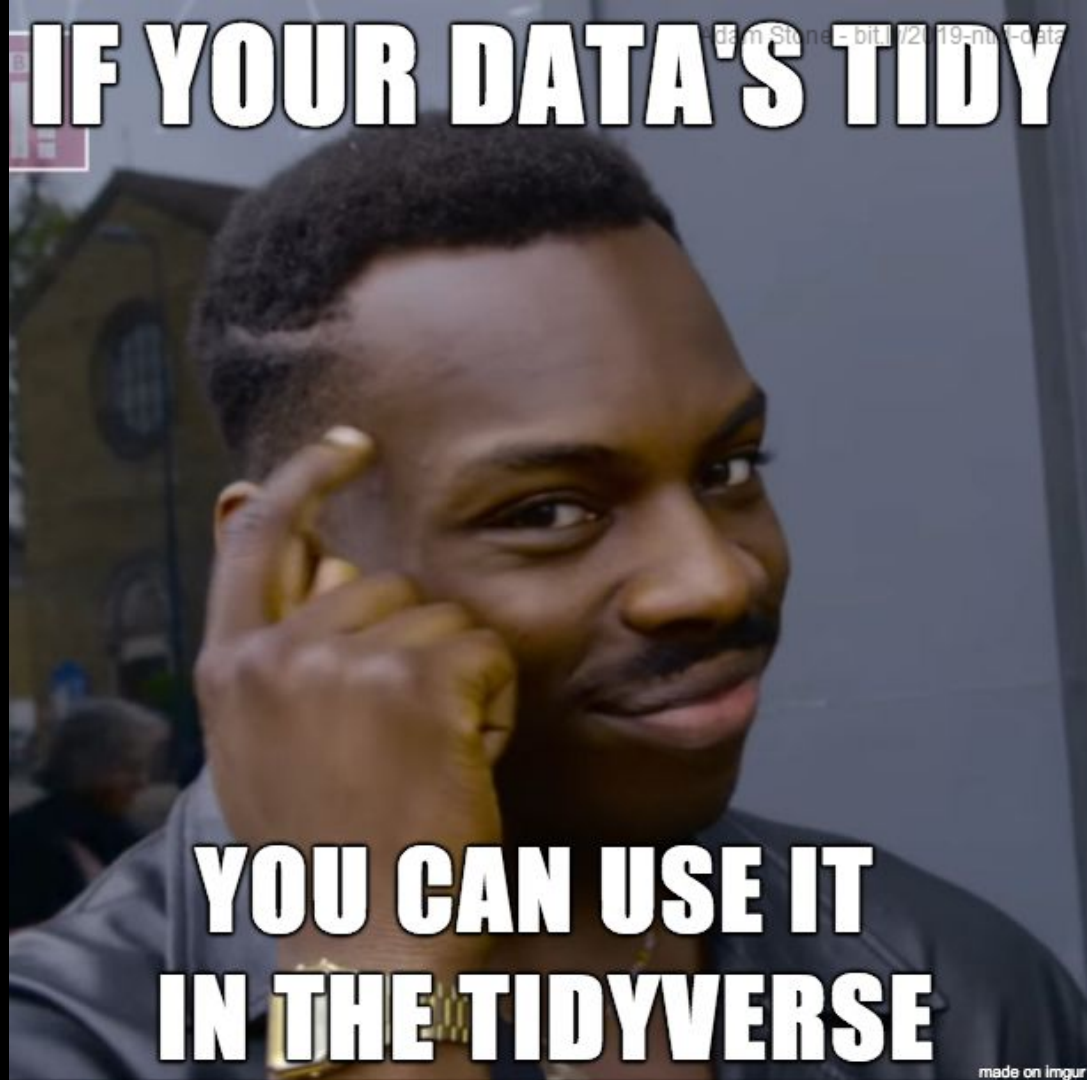

```
aes(x = _____, y = _______, color = _______)
```

Lots of R/tidyverse functions

...including statistical testing...

depend on tidy data!

```
lm(reaction_time ~ age + sex)
ggplot(aes(x = age, y = rt, color = sex))
correlate(age, reaction_time)
```



# gather()

When you've got data scattered across columns, you **gather** them up! Let's do a couple examples.

In RStudio, open 04_tidyr_data_tidying.R

Inspiration from Mike Frank's Tidyverse Tutorial:
https://github.com/mcfrank/tidyverse-tutorial/blob/master/tidyverse_tutorial_clean.Rmd

# spread()

But tidy data isn't always the "right" format. That's just a data format that tidyverse packages expect.

Humans aren't R packages. Which is easier to compare/understand?

| | year | sex | characters |
|---|---|---|---|
| 1 | 1939 | F | 10 |
| 2 | 1939 | Other | 59 |
| 3 | 1940 | F | 33 |
| 4 | 1940 | Other | 188 |
| 5 | 1941 | F | 15 |
| 6 | 1941 | Other | 192 |
| 7 | 1942 | F | 14 |
| 8 | 1942 | Other | 230 |
| 9 | 1943 | F | 13 |
| 10 | 1943 | Other | 185 |

| | year | F | Other |
|---|---|---|---|
| 1 | 1939 | 10 | 59 |
| 2 | 1940 | 33 | 188 |
| 3 | 1941 | 15 | 192 |
| 4 | 1942 | 14 | 230 |
| 5 | 1943 | 13 | 185 |

# spread()

But tidy data isn't always the "right" format. That's just a data format that tidyverse packages expect.

Humans aren't R packages. Which is easier to compare/understand?

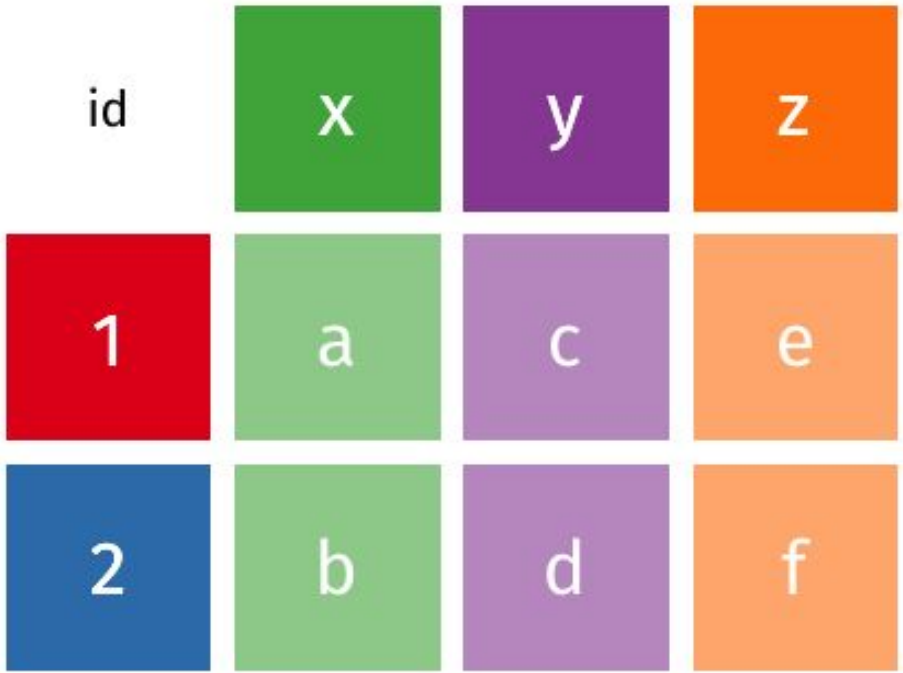
| | year | sex | characters |
|---|---|---|---|
| 1 | 1939 | F | 10 |
| 2 | 1939 | Other | 59 |
| 3 | 1940 | F | 33 |
| 4 | 1940 | Other | 188 |
| 5 | 1941 | F | 15 |
| 6 | 1941 | Other | 192 |
| 7 | 1942 | F | 14 |
| 8 | 1942 | Other | 230 |
| 9 | 1943 | F | 13 |
| 10 | 1943 | Other | 185 |

| | year | F | Other |
|---|---|---|---|
| 1 | 1939 | 10 | 59 |
| 2 | 1940 | 33 | 188 |
| 3 | 1941 | 15 | 192 |
| 4 | 1942 | 14 | 230 |
| 5 | 1943 | 13 | 185 |

```
data %>%
     spread(sex, characters)
```

*Also easier to use mutate().*

# wide



# How do I load data?

You've used `read_csv()` for CSVs. There are many parameters you can add to this, and there are many `readr::` vignettes and tutorials. (Also, `read_delim()` for plain text CSV-like files)

This blog explains how to load a folder of CSVs:
https://www.gerkelab.com/blog/2018/09/import-directory-csv-purrr-readr/

For other file types, there's:

- `readxl::` Excel files
- `haven::` SPSS, SAS, and Stata files
- `googlesheets::` Yep, you can pull files off Google Drive...or write to it!

# Tips

`skimr::skim()` means that I'm using the `skim()` function from the `skimr` package. It also means I don't need to use `library()` because I'm directly referring to the package here. (It took me a while to figure that out, so putting it here.)

In R, `' '` and `" "` are the same thing! I use both. (I'm very inconsistent there).

It's fine to have long-named variables. Text is cheap, brain capacity is not. Be as descriptive as possible. `average_trial_fMRI_data` is always better than `df3`.

Likewise for file names: use descriptive, predictable names! Jenny Bryan has a great slideshow about this: https://speakerdeck.com/jennybc/how-to-name-files/

# Resources for Further Learning

**R for Data Science (R4DS)**
*(the definitive work on tidyverse)*
Book, or free website: http://r4ds.had.co.nz

**R4DS Community & Slack**
*(they have mentors with office hours on Slack)*
https://twitter.com/r4dscommunity

**Data Camp** www.datacamp.com

**#RStats on Twitter**
https://twitter.com/hashtag/rstats

**(and Tidy Tuesdays!)**

**RLadies Twitter & Meetups** https://rladies.org/

**R Live-Coding/Screencasts**
https://twitter.com/drob/status/1106606012658794496

**RIT DataFest** https://www.rit.edu/science/datafest

**Google is your best friend!**