Course overview, Introduction and R Basics

TIRTHANKAR DASGUPTA

DATA WRANGLING AND HUSBANDRY

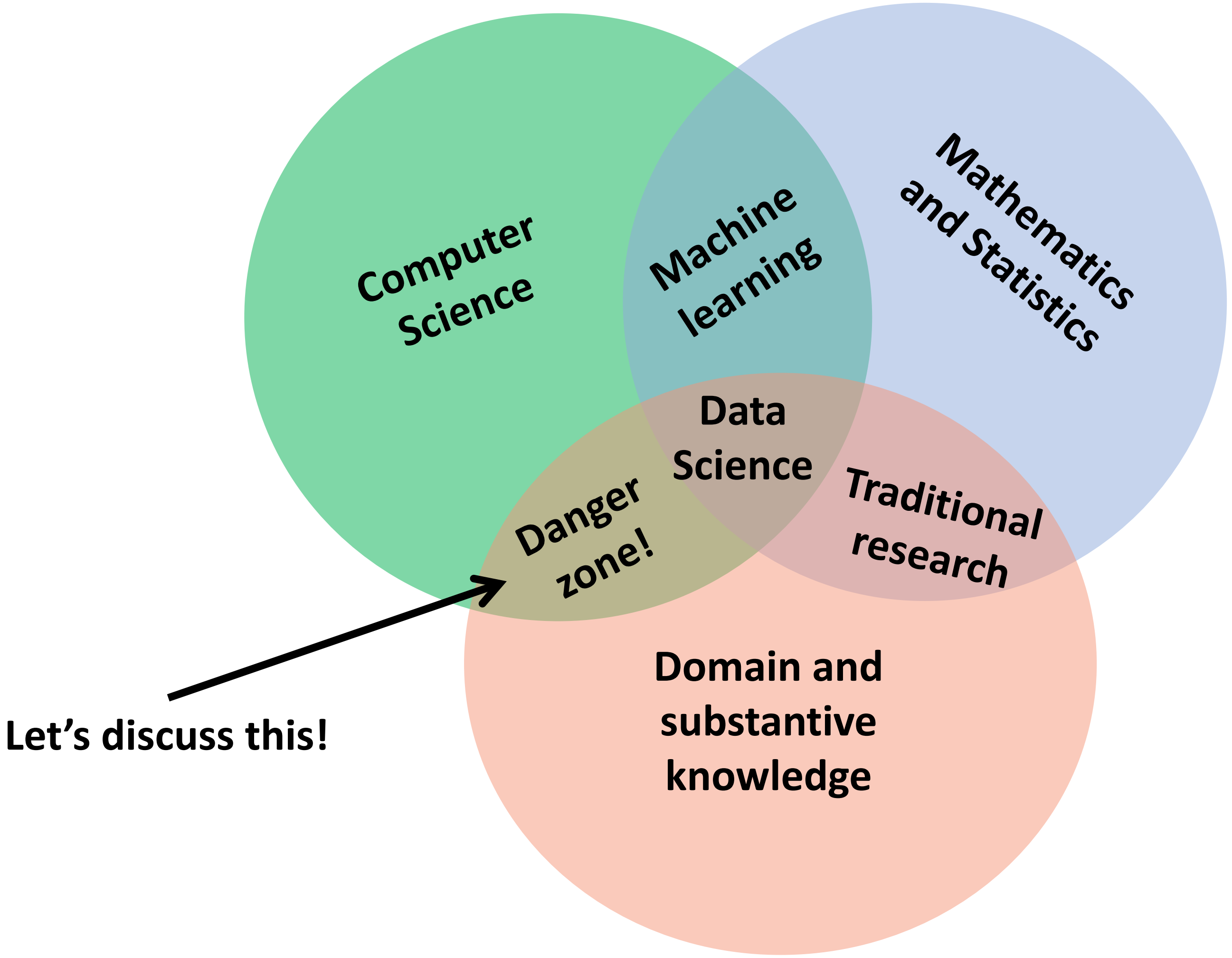
LECTURE 1 (January 25, 2021)

Data Science and the Role of Statistics

What is Data Science?

- "A data scientist is a statistician who lives in San Francisco"
- "Data Science is statistics on a Mac."
- "A data scientist is someone who is better at statistics than any software engineer and better at software engineering than any statistician."

A Better Answer



Using data to answer a “causal” COVID-related question

- 13 critically ill COVID-19 infected patients were reported to have taken an experimental drug, and 12 of them survived.
- Does this data suggest that the new drug is effective for treating COVID?

Does “Big Data” help?

- 13,000 critically ill COVID-19 infected patients were reported to have taken an experimental drug, and 12,000 of them survived.
- Does this data suggest that the new drug is effective for treating COVID?

The “missing” data: WHAT IF

- 13,000 critically ill COVID-19 infected patients were reported to have taken an experimental drug, and 12,000 of them survived (factual).

- What would have happened to these patients if they did not take the drug? (counterfactual)

Adding a “control” group

Treatment	Outcome		Total
	Survived	Died	
Took new drug (Treatment)	12	1	13
Just kept under observation(Control)	11	2	13
Total	23	3	26

Conclusions?

Stronger evidence?

Treatment	Outcome		Total
	Survived	Died	
Took new drug (Treatment)	12	1	13
Just kept under observation(Control)	7	6	13
Total	19	7	26

Even stronger evidence?

Treatment	Outcome		Total
	Survived	Died	
Took new drug (Treatment)	12	1	13
Just kept under observation(Control)	2	11	13
Total	14	12	26

But what if .....

- 13 individuals exposed to the new drug were:

- 13 individuals kept only under observation were:

Images: www.vectorstock.com, www.dreamstime.com

Confounding of effects

- 13 individuals receiving the new drug were young and healthy females

- 13 individuals kept under observation were old males with pre-existing conditions

- Age and underlying medical conditions are **CONFOUNDERS**!

Images: www.vectorstock.com, www.dreamstime.com

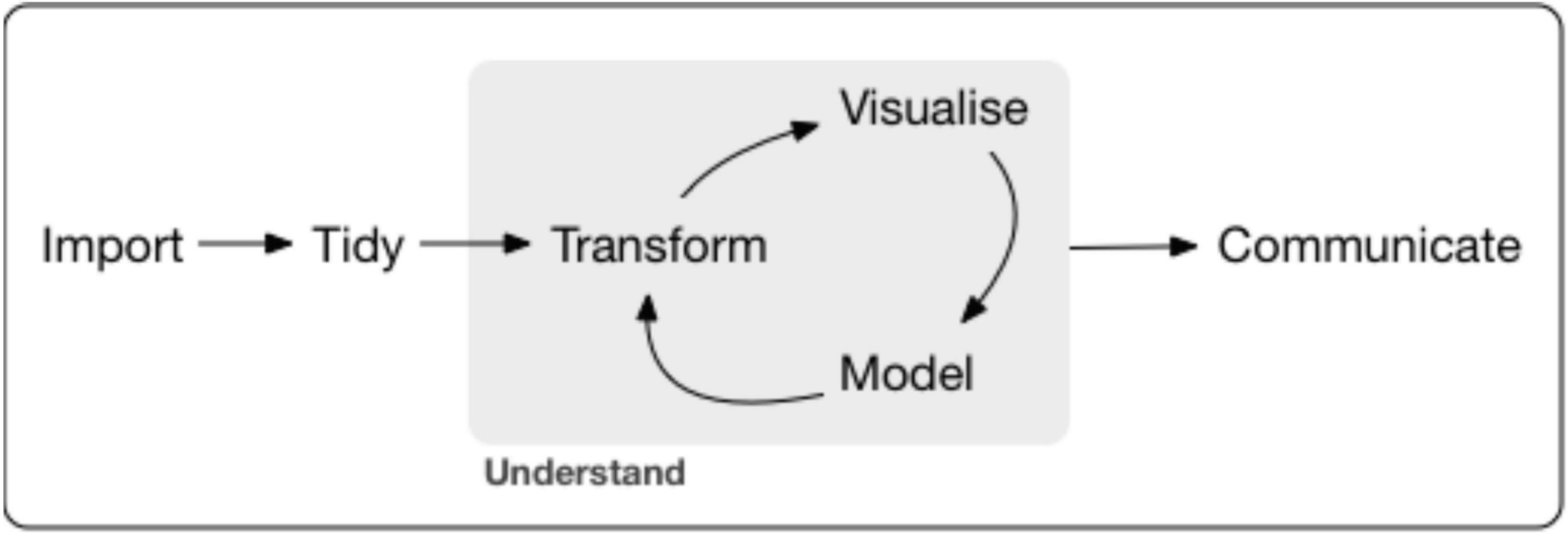
Association and Causation

Treatment	Outcome		Total
	Survived	Died	
Took new drug (Treatment)	12	1	13
Just kept under observation(Control)	2	11	13
Total	14	12	26

Outcome is *associated* with treatment; but not necessarily *caused* by treatment

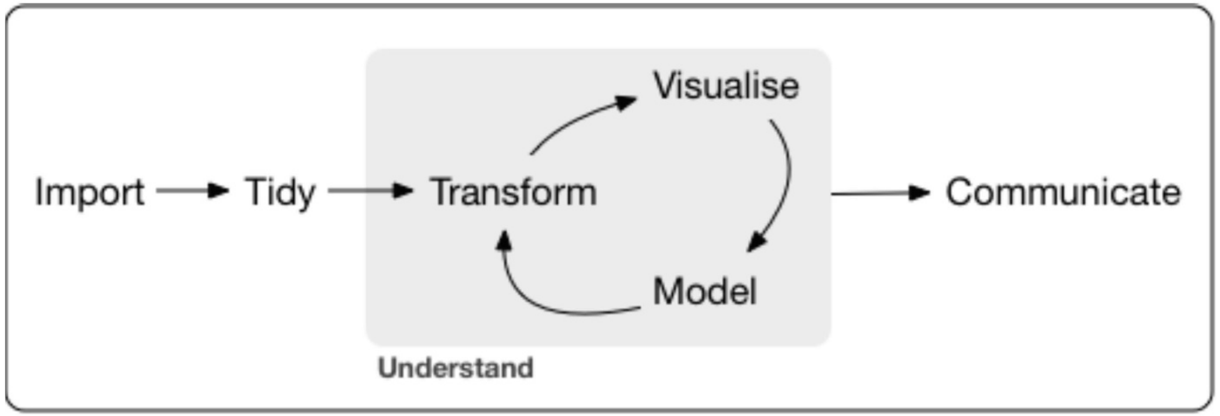
THIS COURSE: OVERVIEW AND LOGISTICS

Essential Steps in Data Science



From R for Data Science, Garrett Grolemund & Hadley Wickham, O'Reilly

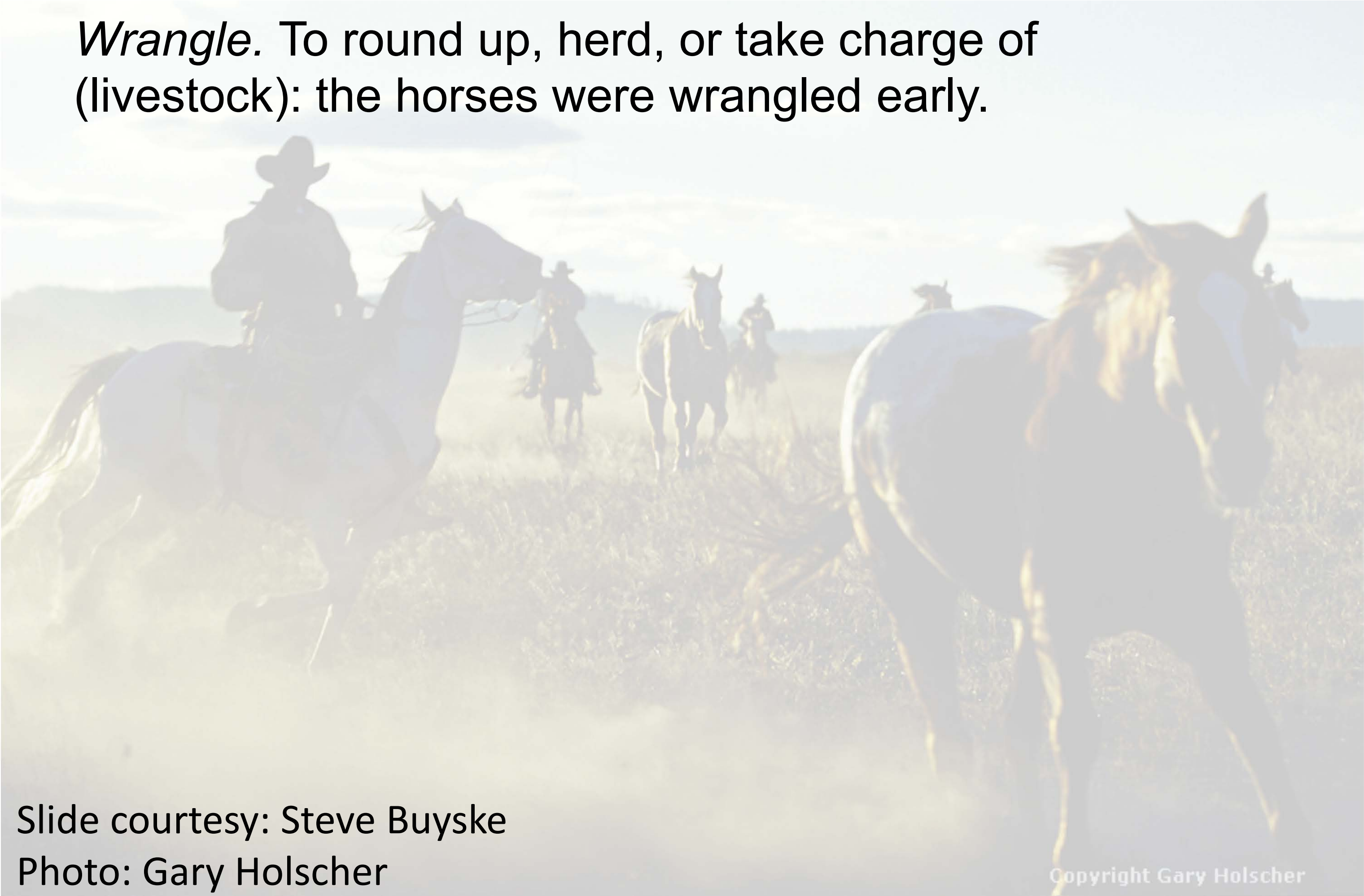
This course



- Statistics courses mostly focus on the central part, and mostly on the model part
- The steps on the left can easily take 80% of the time
- This course is focused on tools for that 80%
- The course is mostly about tools, somewhat about principles, and maybe a little on statistical insights

Meaning of “Wrangling”

Wrangle. To round up, herd, or take charge of (livestock): the horses were wrangled early.



Slide courtesy: Steve Buyske
Photo: Gary Holscher

Meaning of Husbandry

Husbandry. 1. the care, cultivation, and breeding of crops and animals: crop husbandry. 2. management and conservation of resources.

Slide courtesy: Steve Buyske

Textbooks

- Text 1: R for Data Science, Garrett Grolemund & Hadley Wickham, O'Reilly, http://r4ds.had.co.nz/
- Text 2: Data Wrangling with R, Bradley C. Boehmke, Springer,

https://catalog-libraries-rutgers-edu.proxy.libraries.rutgers.edu/vufind/Record/5725290

- Note: the first text is available for free on line, while the second is available online to Rutgers students from the Rutgers library.

Course Work

- Weekly graded assignments (posted every Monday and due the following Monday) and a final project. The homework will collectively count towards 70% of the grade and the final project the remaining 30%. There will be no exams.

Today's goal

- Basics of R and Rstudio
- Getting warmed up with the R Console
- Be able to
  - Install and load packages
  - Understand the structure of rectangular databases [row by column]
  - Perform basic operations like creating subsets (filtering and selecting), sorting/arranging
  - Create graphs using ggplot
  - Tweak a template R markdown file to submit assignments

BASICS OF R

Advantages of R

- Implemented in 1990’s by Ihaka and Gentleman at the University of New Zeland, Auckland
- Chapter 2 of Boehmke
- Open Source
  - Blurs distinction between developed and user
- Flexibility
  - Anybody can access, modify, improve code
- Community
  - Diverse and engages
  - Section 1.6 of Wickham & Grolemund (“Getting help and learning more”)

What we'll need to start

- Download R
    - Section 1.4.1 of Wickham & Grolemund
    - Section 3.1 of Boehmke

- Understanding and working with **Rstudio** [We'll do a live demo]
    - Section 1.4.2 of Wickham & Grolemund
    - Section 3.2-3.3 of Boehmke

- Installing and loading some packages
    - An R **package** is a collection of functions, data, and documentation that extends the capabilities of base R. Using packages is key to the successful use of R.
    - Section 3.4 of Boehmke
    - Install tidyverse

- Running basic R codes

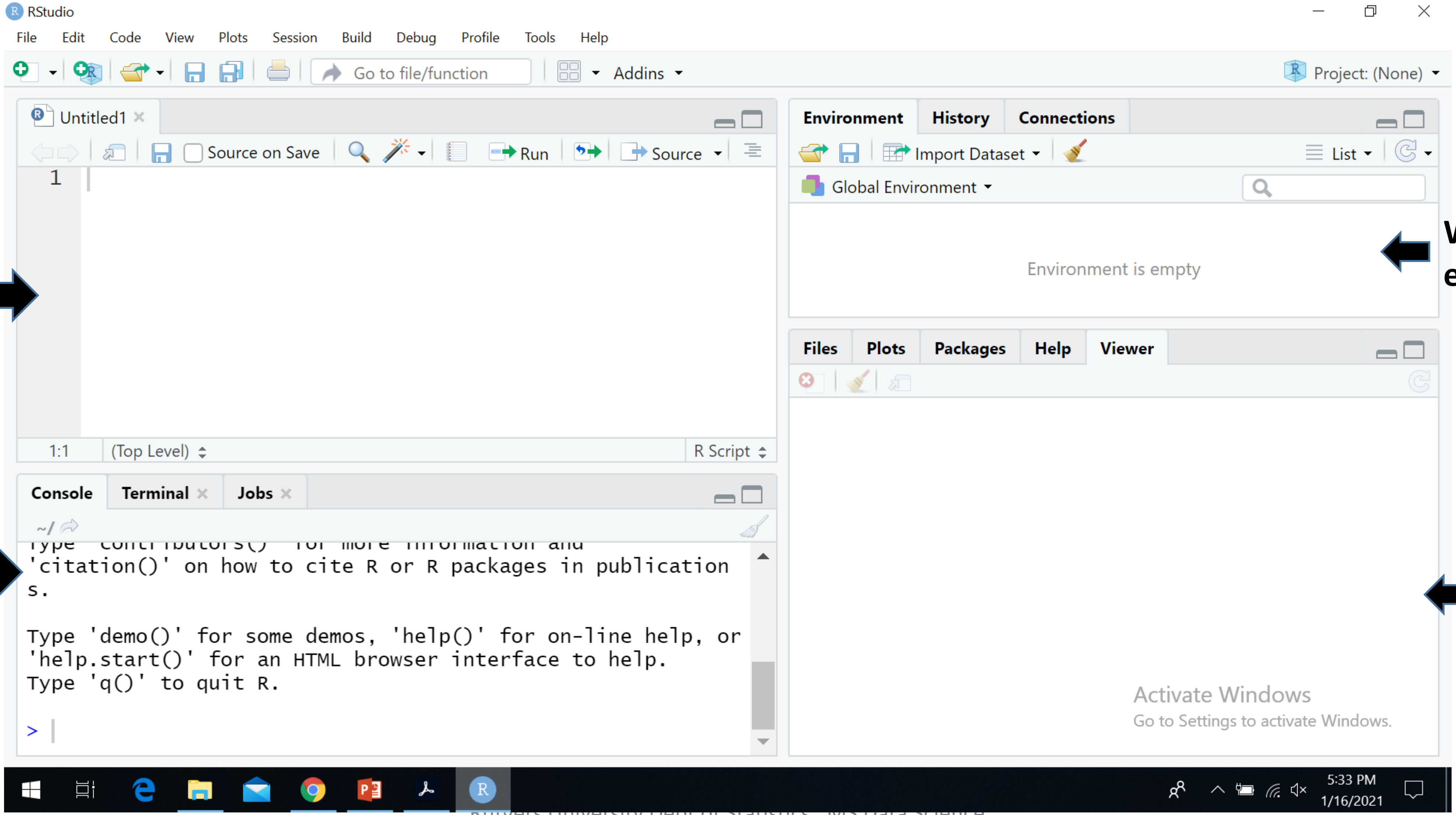
The “tidyverse” package

- Section 1.4.3 of Wickham and Grolemund
- Starting 13 years ago, there has been an effort led by Hadley Wickham to improve the data handling and visualization aspects of R (once known as the "Hadleyverse" but now known as the "tidyverse")
- Old-timers tend to use the older, though less convenient, base R commands.
- The tidyverse approach is rapidly winning out

Packages within tidyverse

```
library(tidyverse)
#> ── Attaching packages
───────────────────────────────────────
tidyverse 1.3.0 ──
#> ✓ ggplot2 3.3.2     ✓ purrr   0.3.4
#> ✓ tibble  3.0.3     ✓ dplyr   1.0.2
#> ✓ tidyr   1.1.2     ✓ stringr 1.4.0
#> ✓ readr   1.4.0     ✓ forcats 0.5.0
#> ── Conflicts
──────────────────────────────────────────
tidyverse_conflicts() ──
#> ✖ dplyr::filter() masks stats::filter()
#> ✖ dplyr::lag()    masks stats::lag()
```

R Studio



Script editor

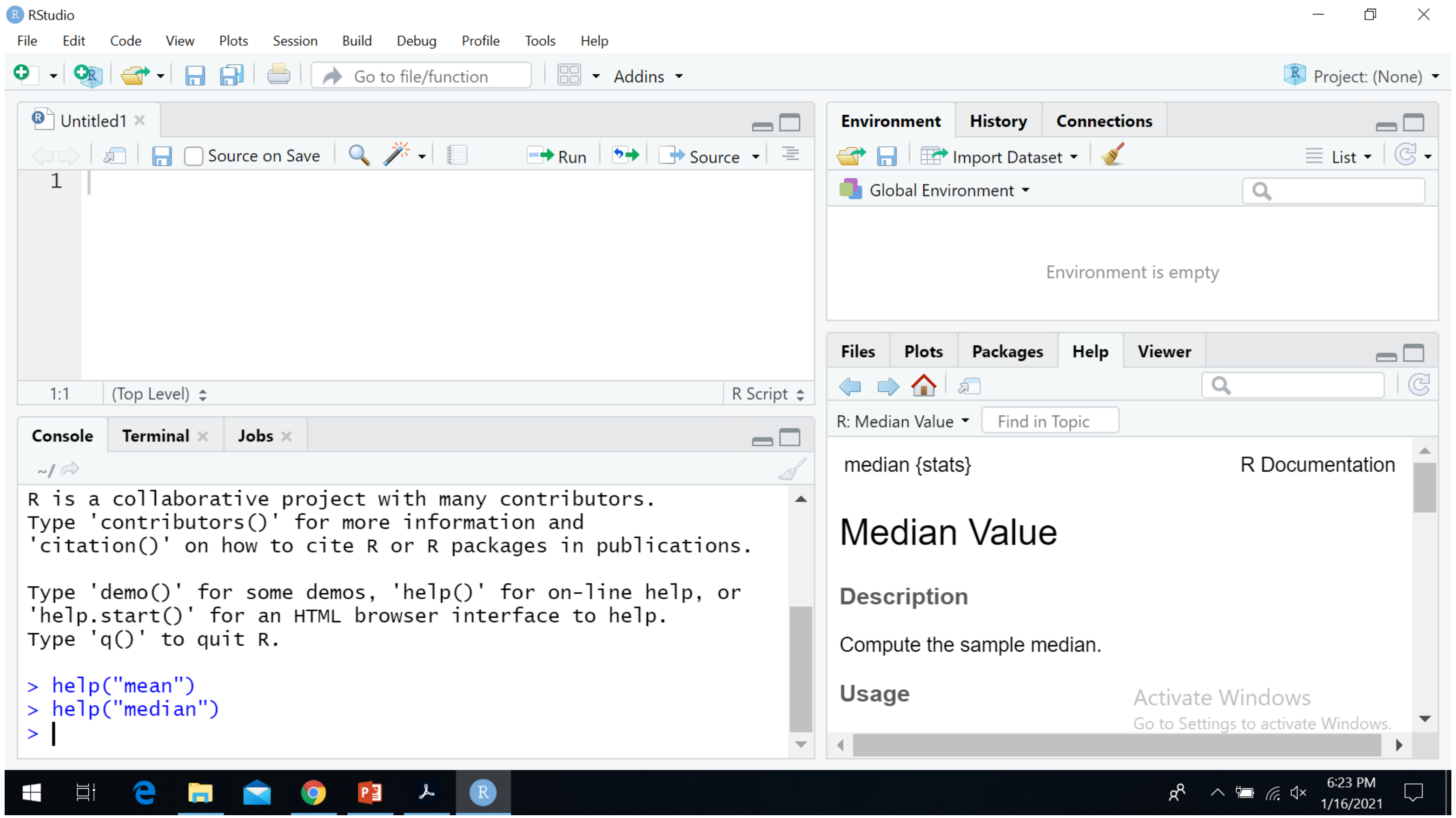
Console

Workspace environment

Misc. displays

Getting help in RStudio

- Section 3.3 of Boehmke, Section 1.6 of Wickham & Grolemund
- Type help(“topic”), help(functionname), example(functionname) in console



Other important workflow basics

- Chapter 4 of Wickham & Grolemund

- Assignment (Boehmke 3.4)
  - X<-5 or X=5

- Calculations (Boehmke 3.5)
  - Operators
  - Vectorization

- Code Styling guide (Boehmke 3.6)
  - Notation and naming
  - Organization
  - Syntax

Different Types of Data in R

- Integer
- Numeric (Double Precision floating point numbers https://en.wikipedia.org/wiki/Double-precision_floating-point_format)
  - Convert between integer and numeric/double using as.integer( ) and as.numeric( ).
- Character
- Complex
- Logical

Generating sequences of non-random numbers

- Boehmke 4.2
- The colon : operator , e.g., x <- c(1:8)
- The seq( ) operator, e.g., x1<- seq(from=0, to=2, by=.5)

```
> x1<- seq(from=0, to=2, by=.5)
> x1
[1] 0.0 0.5 1.0 1.5 2.0
```

- The rep( ) operator, e.g., x2<-rep(x1,2)

```
> x2<-rep(x1,2)
> x2
[1] 0.0 0.5 1.0 1.5 2.0 0.0 0.5 1.0 1.5 2.0
```

Comparison Operators

- Boehmke 4.5
- Binary operators x < y, x > y, x >= y, x<=y, x==y, x!=y – compare two scalars or vectors and provide output as logical forms
- sum(x==y) counts number of elements of x and y that are equal
- which(x==y) identifies elements of x and y that are equal
- identical(x==y) tests if two objects x and y are exactly equal

Rounding numbers

- Boehmke 4.6

```
> x<-c(1.35,2.56,3.39,4.23)

> round(x)
[1] 1 3 3 4
> ceiling(x)
[1] 2 3 4 5
> floor(x)
[1] 1 2 3 4
> round(x,digits=1)
[1] 1.4 2.6 3.4 4.2
```

Character strings

- **Chapter 5** of Boehmke
- Create
  - a <- “This lecture is boring”, b<- “we are feeling sleepy”
- Paste two strings or strings and numbers
  - paste(a,b):
  - paste(“Value of pi is”,pi)

```
> paste(a,b)
[1] "This lecture is boring, we are feeling sleepy"
>paste("The value of pi is:", pi)
[1] "The value of pi is: 3.14159265358979"
```

- Converting to strings
- Printing strings
- Counting words and characters

DATA VISUALIZATION BASICS

ggplot2: The data visualization package

```
library(tidyverse)
#> ── Attaching packages ───────────────────────────────────────
tidyverse 1.3.0 ──
#> ✓ ggplot2 3.3.2     ✓ purrr   0.3.4
#> ✓ tibble  3.0.3     ✓ dplyr   1.0.2
#> ✓ tidyr   1.1.2     ✓ stringr 1.4.0
#> ✓ readr   1.4.0     ✓ forcats 0.5.0
#> ── Conflicts ───────────────────────────────────────────────
tidyverse_conflicts() ──
#> ✖ dplyr::filter() masks stats::filter()
#> ✖ dplyr::lag()    masks stats::lag()
```

Car mileage data

Description
This dataset contains a subset of the fuel economy data that the EPA makes available on http://fueleconomy.gov. It contains only models which had a new release every year between 1999 and 2008 - this was used as a proxy for the popularity of the car.
Usage
mpg
Format
A data frame with 234 rows and 11 variables:
1. manufacturer: manufacturer name
2. model: model name
3. displ: engine displacement, in litres
4. year: year of manufacture
5. cyl: number of cylinders
6. trans: type of transmission
7. drv: the type of drive train, where f = front-wheel drive, r = rear wheel drive, 4 = 4wd
8. cty:city miles per gallon
9. hwy: highway miles per gallon
10. fl: fuel type
11. class: "type" of car

Structure of data

- Rectangular (data frames / tibbles)
- Rows indicate entities
- Columns indicate variables
- In “mpg” dataset, rows are cars
- Each of eleven columns denote a variable
- ?mpg described the data (as in the previous slide)
- 234x11 matrix

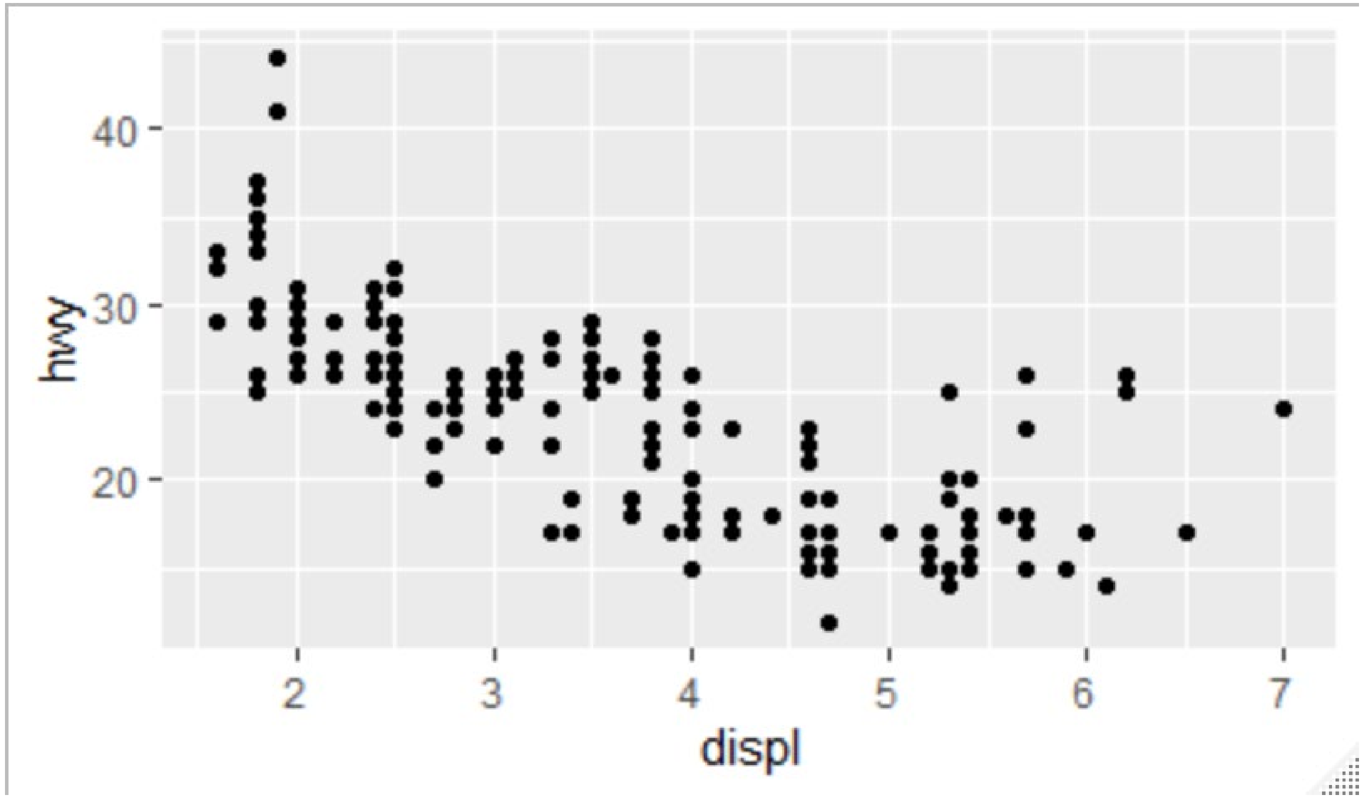
Question:

Wickham & Grolemund 3.2:

Do cars with big engines use more fuel than cars with small engines? You probably already have an answer, but try to make your answer precise. What does the relationship between engine size and fuel efficiency look like? Is it positive? Negative? Linear? Nonlinear?

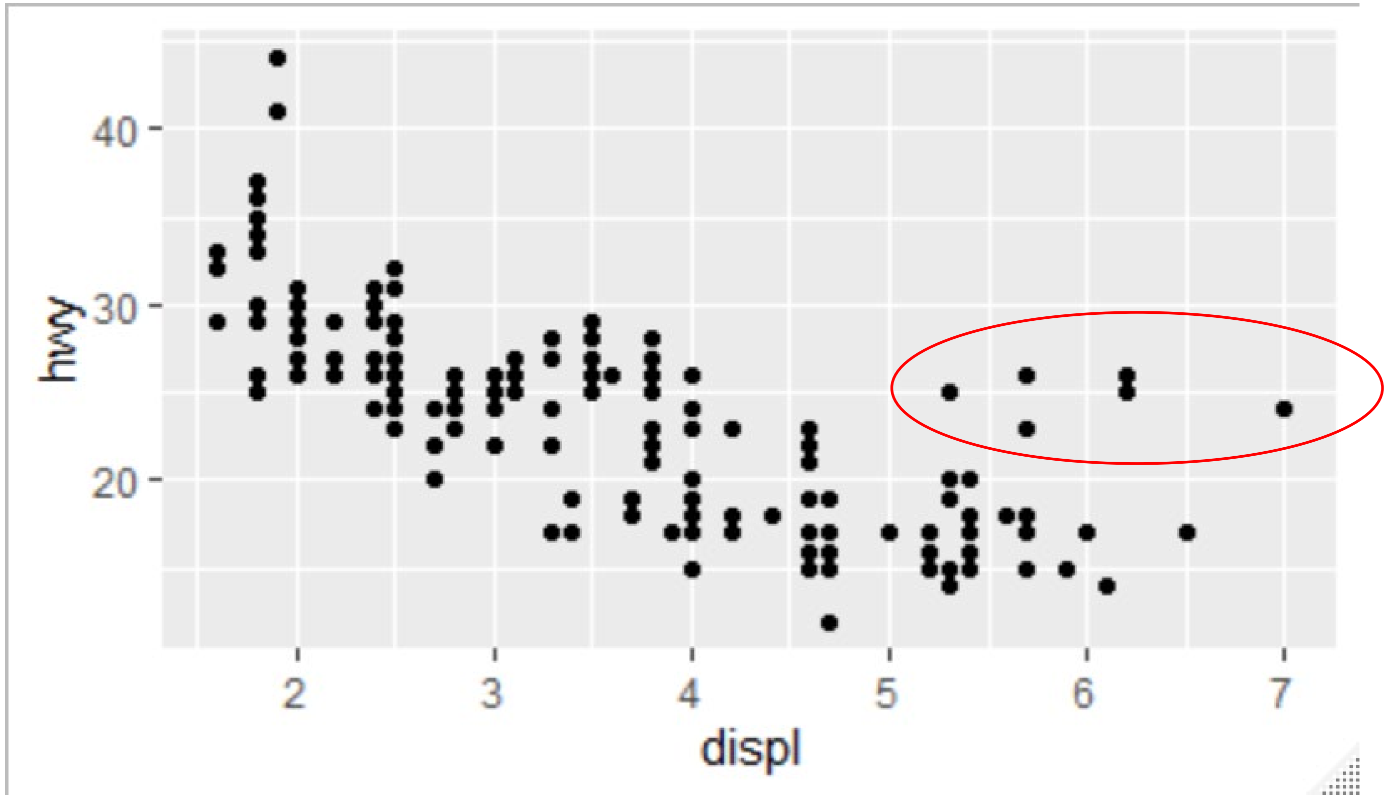
Plotting Highway miles against engine size

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```



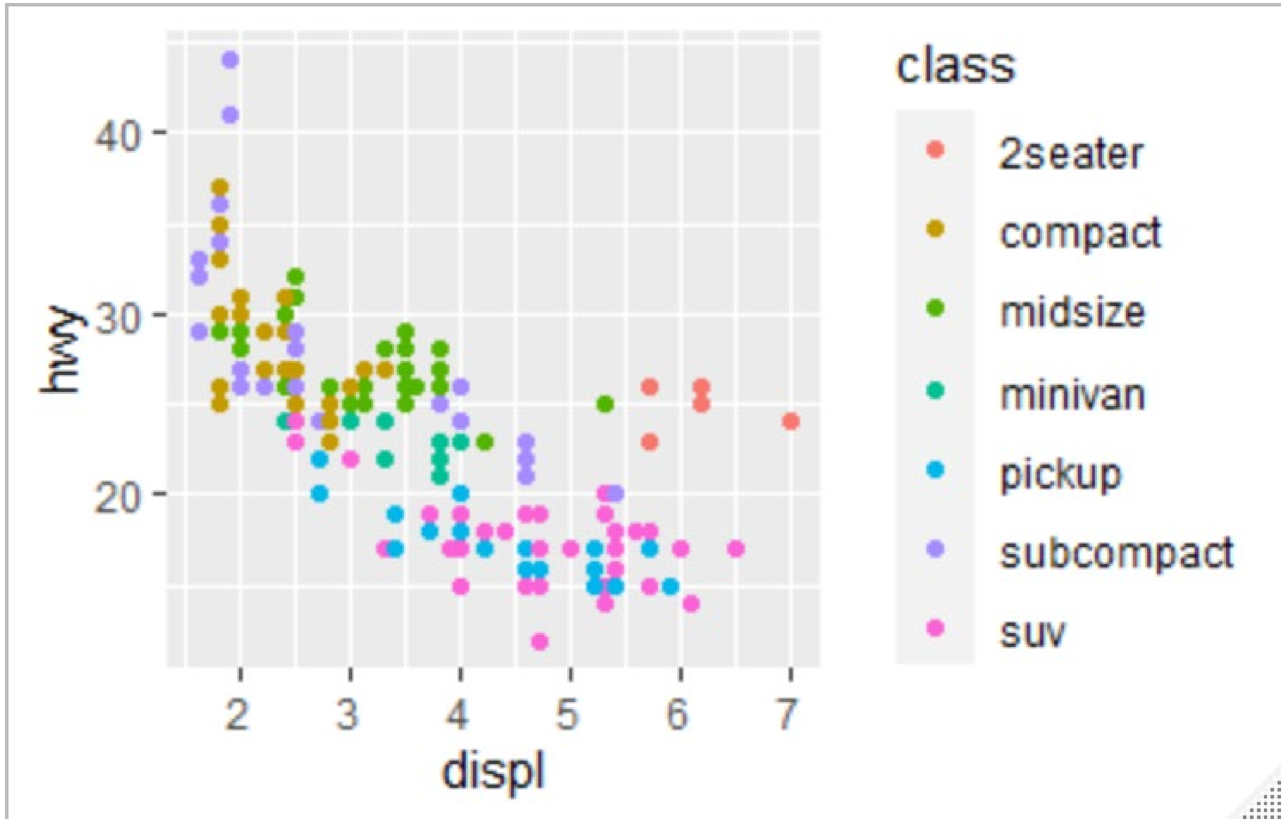
Looking at outliers

```
Why does the circled cluster of points depict a
different behavior?
```



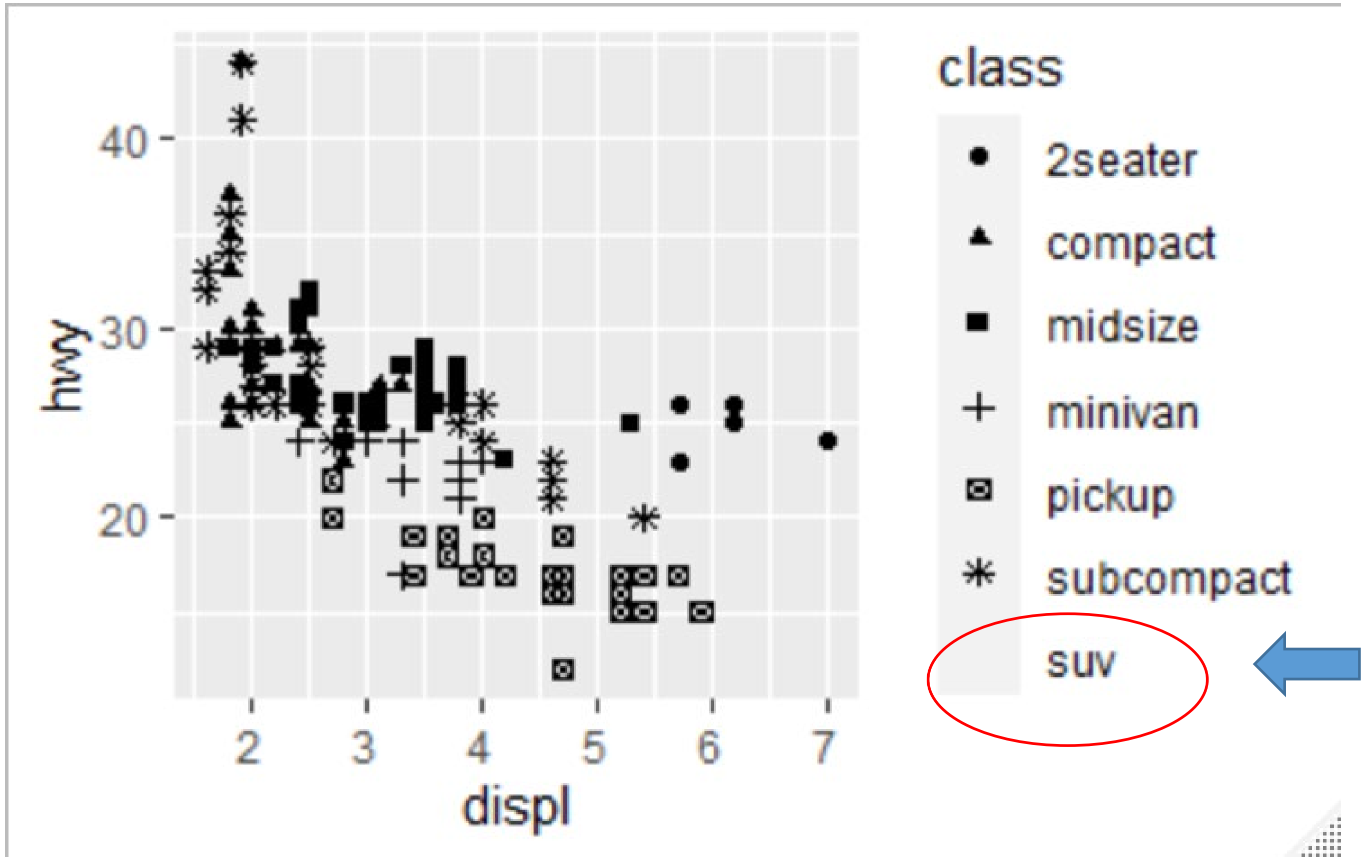
Adding aesthetics (stratifying by color)

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```



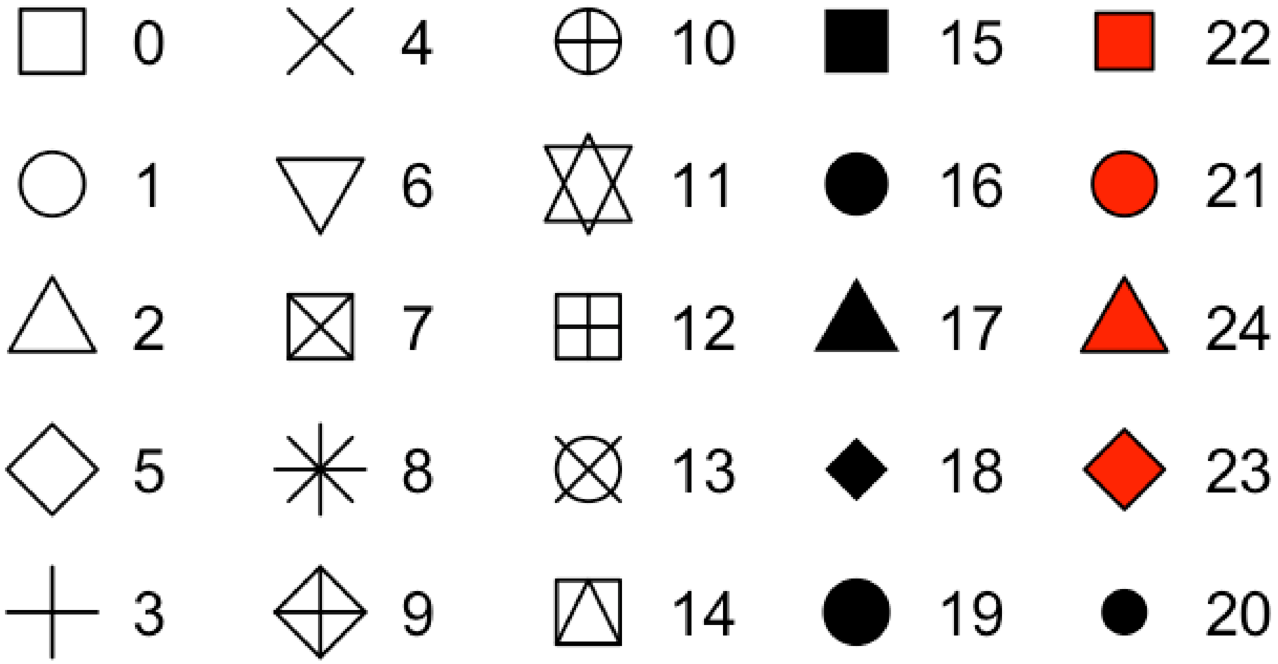
Adding aesthetics (stratifying by shape of points)

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, shape = class))
```



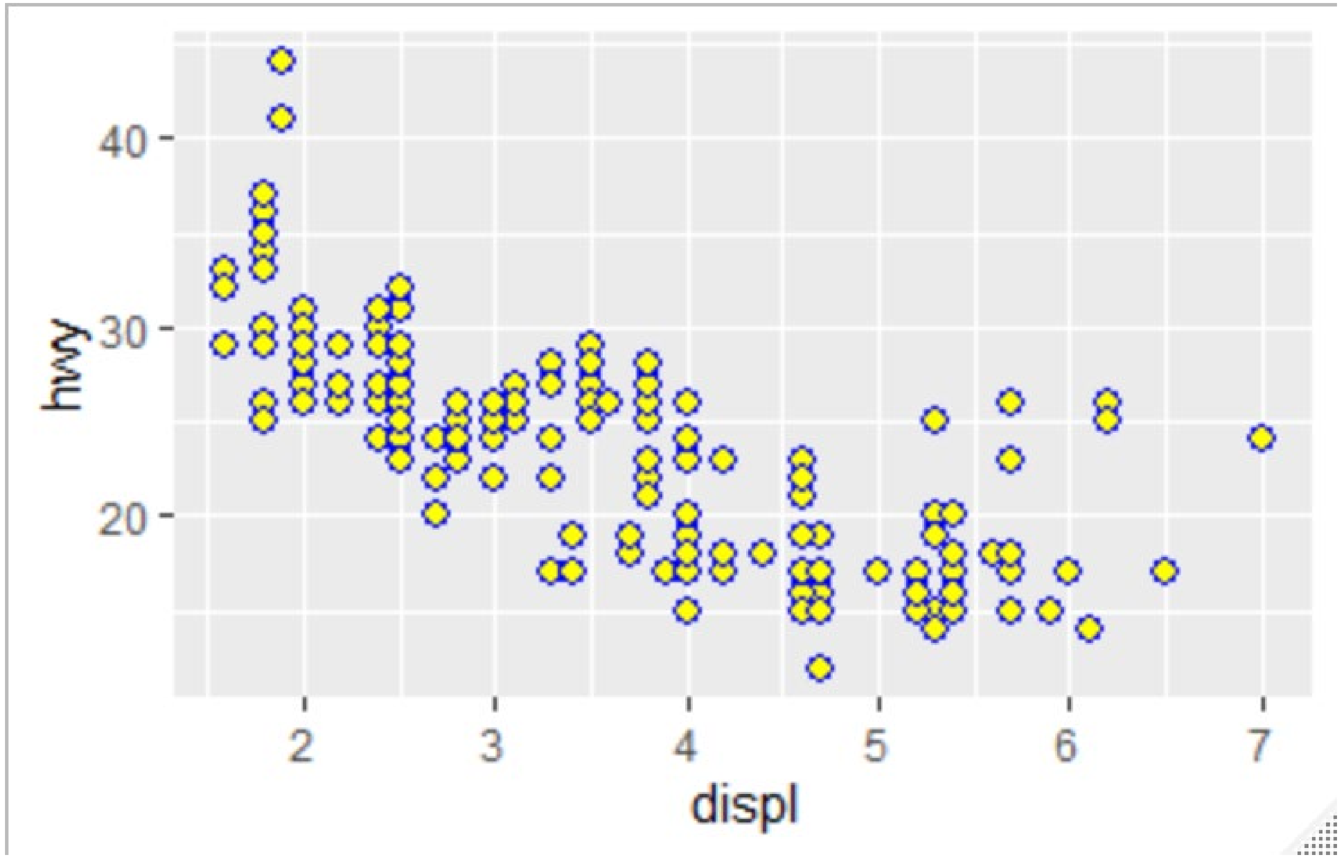
No Shape?

Shapes in ggplot



Adding features to the entire plot

```
ggplot(data=mpg) +
  geom_point(mapping = aes(x=displ, y=hwy),shape = 21, colour = "blue", fill = "yellow", size = 2, stroke=1)
```



Common Problems and Syntax Errors

```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy))
```

```
ggplot(data = mpg)
+ geom_point(mapping = aes(x = displ, y = hwy))
```

Read Section 3.4 (Common problems) of Wickham & Grolemund

DATA FRAMES AND BASIC OPERATIONS

dplyr: The data transformation package

```
library(tidyverse)
#> ── Attaching packages ───────────────────────────────────────
tidyverse 1.3.0 ──
#> ✓ ggplot2 3.3.2     ✓ purrr   0.3.4
#> ✓ tibble  3.0.3     ✓ dplyr   1.0.2
#> ✓ tidyr   1.1.2     ✓ stringr 1.4.0
#> ✓ readr   1.4.0     ✓ forcats 0.5.0
#> ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
#> ✖ dplyr::filter() masks stats::filter()
#> ✖ dplyr::lag()    masks stats::lag()
```

Data frames and basic operations

- Recall that we will work with rectangular data (data frames / tibbles)
- Rows indicate entities
- Columns indicate variables
- Let us understand some basic operations using flight data
- Install package “nycflights13”
- On-time data for all flights that departed NYC (i.e. JFK, LGA or EWR) in 2013.
- 336776 rows (flights), 19 columns (variables)

Data frame (Tibble) “flights”

```
> library(tidyverse)
> library(nycflights13)
> # Now we will work with flights data
> flights
# A tibble: 336,776 x 19
year month day dep_time sched_dep_time dep_delay arr_time
int> <int> <int> <int>     <int>           <dbl>     <int>
1  2013 1     1     517       515               2       830
2  2013 1     1     533       529               4       850
3  2013 1     1     542       540               2       923
4  2013 1     1     544       545              -1      1004
5  2013 1     1     554       600              -6       812
6  2013 1     1     554       558              -4       740
7  2013 1     1     555       600              -5       913
8  2013 1     1     557       600              -3       709
9  2013 1     1     557       600              -3       838
10  2013 1    1     558       600              -2       753
# ... with 336,766 more rows, and 12 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>
```

Variables in flight data

year, month, day: Date of departure.
dep_time, arr_time: Actual departure and arrival times (format HHMM or HMM), local tz.
sched_dep_time, sched_arr_time: Scheduled departure and arrival times (format HHMM or HMM), local tz.
dep_delay, arr_delay: Departure and arrival delays, in minutes. Negative times represent early departures/arrivals.
Carrier: Two letter carrier abbreviation. See airlines to get name.
Flight: Flight number.
Tailnum: Plane tail number. See planes for additional metadata.
origin, dest: Origin and destination. See airports for additional metadata.
air_time: Amount of time spent in the air, in minutes.
Distance: Distance between airports, in miles.
hour, minute: Time of scheduled departure broken into hour and minutes.
time_hour: Scheduled date and hour of the flight as a POSIXctdate. Along with origin, can be used to join flights data to weather data.

Five basic dplyr functions (Wickham & Grolemund 5.1.3)

- Pick observations by their values (filter()).
- Reorder the rows (arrange()).
- Pick variables by their names (select()).
- Create new variables with functions of existing variables (mutate()).
- Collapse many values down to a single summary (summarise()).

Finding a subset of the data frame using "filter"

All flights on Feb 1

```
flightsFeb1 = filter(flights, month==2, day==1)
> flightsFeb1
```

```
# A tibble: 926 x 19
year month day dep_time sched_dep_time dep_delay
arr_time
<int> <int> <int> <int>          <int>       <dbl>     <int>
1 2013     2     1     456            500          -4       652
2 2013     2     1     520            525          -5       816
3 2013     2     1     527            530          -3       837
```

Understanding AND and OR conditions

All flights in January, February (Actually Jan OR Feb)

```
(flightsJanFeb = filter(flights, month==1 | month==2))
```

```
# A tibble: 51,955 x 19
```

Understanding AND and OR conditions (Contd).

Flights that were NOT delayed by less than two hours (either at arrival or departure)

A and B

```
filter(flights, arr_delay <= 120, dep_delay <= 120)
```

Not A or Not B

```
filter(flights, !(arr_delay > 120 | dep_delay > 120))
```

De-Morgan's laws in set theory

Arranging (sorting) data

Arrange by descending order of delay departure

```
arrange(flights, desc(dep_delay))
#> # A tibble: 336,776 x 19
#>    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
#>   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
#> 1  2013     1     9      641            900      1301     1242           1530
#> 2  2013     6    15     1432           1935      1137     1607           2120
#> 3  2013     1    10     1121           1635      1126     1239           1810
#> 4  2013     9    20     1139           1845      1014     1457           2210
#> 5  2013     7    22      845           1600      1005     1044           1815
#> 6  2013     4    10     1100           1900       960     1342           2211
#> # … with 336,770 more rows, and 11 more variables: 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>
```

Selecting Columns

```
# Select columns by name
select(flights, year, month, day)
#> # A tibble: 336,776 x 3
#>    year month   day
#>   <int> <int> <int>
#> 1  2013     1     1
#> 2  2013     1     1
#> 3  2013     1     1
#> 4  2013     1     1
#> 5  2013     1     1
#> 6  2013     1     1
#> # … with 336,770 more rows
```

Another way to select consecutive columns

```
# Select all columns between year and day (inclusive)
select(flights, year:day)
#> # A tibble: 336,776 x 3
#>    year month   day
#>   <int> <int> <int>
#> 1  2013     1     1
#> 2  2013     1     1
#> 3  2013     1     1
#> 4  2013     1     1
#> 5  2013     1     1
#> 6  2013     1     1
#> # … with 336,770 more rows
```

Excluding columns

```
# Select all columns except those from year to day (inclusive)
select(flights, -(year:day))
#> # A tibble: 336,776 x 16
#>   dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay
carrier
#>      <int>          <int>     <dbl>    <int>          <int>     <dbl> <chr>
#> 1      517            515         2      830            819        11 UA
#> 2      533            529         4      850            830        20 UA
#> 3      542            540         2      923            850        33 AA
#> 4      544            545        -1     1004           1022       -18 B6
#> 5      554            600        -6      812            837       -25 DL
#> 6      554            558        -4      740            728        12 UA
#> # … with 336,770 more rows, and 9 more variables: flight <int>, tailnum
<chr>,
#> #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#> #   minute <dbl>, time_hour <dttm>
```

Exercises from Wickham & Grolemund

- Do Exercises 5.2.4 (Filter)
- Do Exercises 5.3.1 (Arrange)
- Do Exercises 5.4.1 (Select)

BRIEF OVERVIEW OF R MARKDOWN

R Markdown (Quoted from Wickham & Grolemund 27)

- Provides a unified authoring framework for data science, combining your code, its results, and your prose commentary.
- Fully reproducible and support several output formats, like PDFs, html documents, and more.
- R Markdown files are designed to be used in three ways:
  - For communicating to decision makers, who want to focus on the conclusions, not the code behind the analysis.
  - For collaborating with other data scientists (including future you!), who are interested in both your conclusions, and how you reached them (i.e. the code).
  - As an environment in which to *do* data science, as a modern day lab notebook where you can capture not only what you did, but also what you were thinking.

At this stage

- Two examples of R markdown files are provided.
- We will discuss intricacies later.
- At this stage we will simply use it as a tool to create assignment outputs for submission.
- You can simply tweak the example codes, play with them and incorporate your solutions to assignments.

ASSIGNMENT 1
(DUE FEB 1)

Two Parts

- Reading from the two texts
- Exercises from Wickham & Grolemund
- Creating some visualizations from the “babynames” package guided by specific queries
- Examples follow

Babynames

The Social Security Administration provides a nice dataset of first names at birth by sex and year.

```
# A tibble: 1,924,665 x
5 year sex name          n   prop
<dbl> <chr> <chr>     <int> <dbl>
1   1880 F  Mary       7065 0.0724
2   1880 F  Anna       2604 0.0267
3   1880 F  Emma       2003 0.0205
4   1880 F  Elizabeth  1939 0.0199
5   1880 F  Minnie     1746 0.0179
6   1880 F  Margaret   1578 0.0162
7   1880 F  Ida        1472 0.0151
8   1880 F  Alice      1414 0.0145
9   1880 F  Bertha     1320 0.0135
10  1880 F  Sarah      1288 0.0132
# ... with 1,924,655 more rows
```

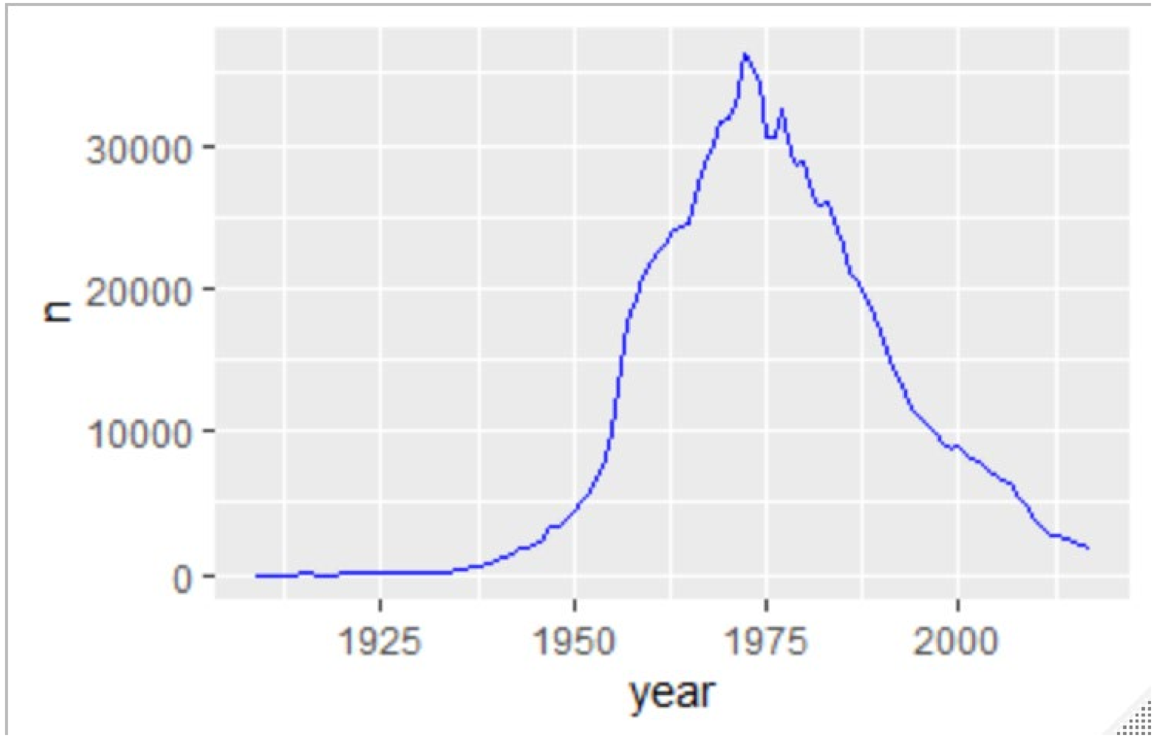
Filter entries that are male and have name "Brian"

```
babynames_Brian = filter(babynames, name=="Brian“, sex==“M”)
babynames_Brian
```

```
➢ babynames_Brian
➢ # A tibble: 179 x 5
➢ year sex name n prop
➢ <dbl> <chr> <chr> <int> <dbl>
➢ 1 1909 M Brian 5 0.0000283
➢ 2 1911 M Brian 7 0.000029
➢ 3 1912 M Brian 6 0.0000133
➢ 4 1913 M Brian 14 0.0000261
➢ 5 1914 M Brian 11 0.0000161
➢ 6 1915 M Brian 21 0.0000238
➢ 7 1916 M Brian 17 0.0000184
➢ 8 1917 M Brian 14 0.0000146
➢ 9 1918 M Brian 12 0.0000114
➢ 10 1919 M Brian 14 0.0000138 # ...
  with 169 more rows
```

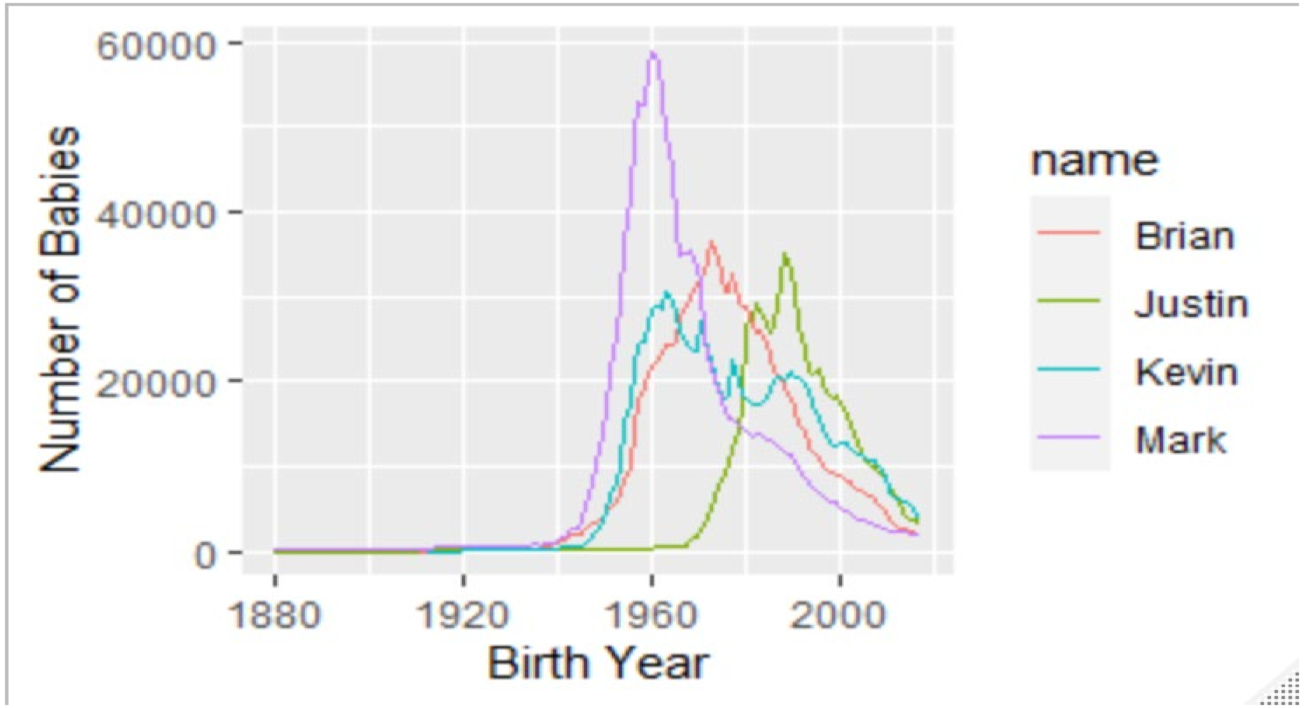
Plot the data

```
ggplot(data=babynames_Brian) +
  geom_line(mapping=aes(x=year,y=n),color="blue")
```



Another example

```
babynames4=filter(babynames, name %in% c("Brian", "Justin", "Kevin", "Mark"), sex == "M")
  ggplot(babynames4) +
  geom_line(mapping=aes(year, n, colour = name)) +
  ylab("Number of Babies") +
  xlab("Birth Year")
```



PART 2 (TO HAND IN)

- Install the package "babynames"
- Plot the number of male and female babies named Taylor *by year*
- Answer the following questions, showing plots to substantiate your answers:
    - Is a 16 year old named Quinn more likely to be a boy or a girl?
    - Is a 2 year old named Quinn more likely to be a boy or a girl?
    - What is your best guess as to how old a woman named Susan is?

Submit your answers (via Canvas) as a single RMarkdown file that can be run on anyone's machine (i.e., that doesn't refer to your files or directories).