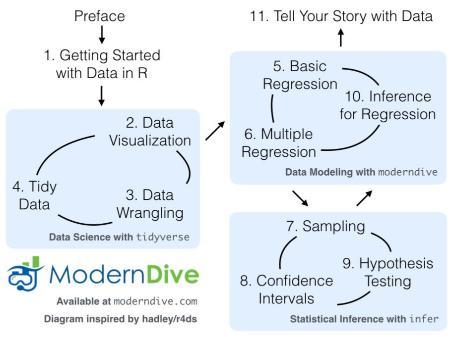
# DATA 202

Table of Contents

# ***Visualization 1***

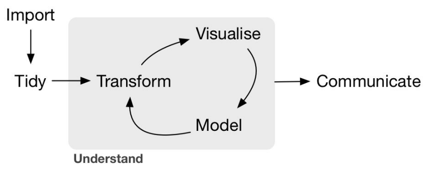
## ModernDive Preface and Chapter 1

### PREFACE



Preface
1. Getting Started with Data in R
2. Data Visualization
3. Data Wrangling
4. Tidy Data
Data Science with tidyverse
5. Basic Regression
6. Multiple Regression
10. Inference for Regression
Data Modeling with moderndive
11. Tell Your Story with Data
7. Sampling
8. Confidence Intervals
9. Hypothesis Testing
Statistical Inference with infer
ModernDive
Available at moderndive.com
Diagram inspired by hadley/r4ds

- Data/Science Pipeline:



- Reproducible Research:
    - **computational reproducibility**. This refers to being able to pass all of one's data analysis, datasets, and conclusions to someone else and have them get exactly the same results on their machine.

### CHAPTER 1: Getting Started w/ Data in R

- *Vectors*: a series of values. These are created using the `c()` function, where `c()` stands for "combine" or "concatenate." For example, `c(6, 11, 13, 31, 90, 92)` creates a six element series of positive integer values.
- Logical operators: `&` representing "and" as well as `|` representing "or."

Errors, warnings, & messages
- **Error:** something legitimately went wrong; prefaced with "Error in…" & will try to explain what went wrong. Generally, code will not run.
    - Something is wrong; figure out what's causing it.
- **Warnings**: code will generally still work, but with caveats.

    - Everything is working fine; but watch out / pay attention.
- **Messages:** just a friendly message. Helpful diagnostic messages.
    - Everything is working fine; keep going.

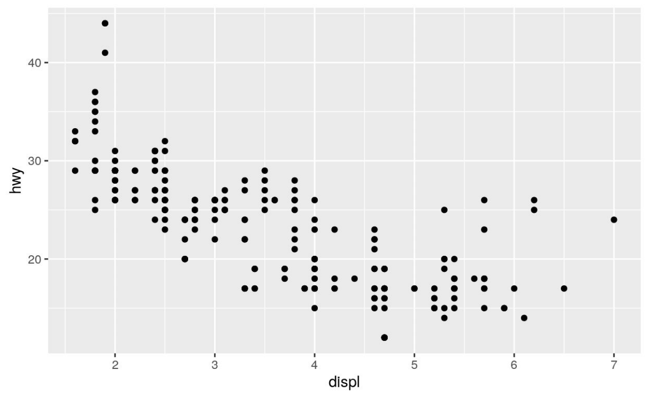
Packages:
- Always load them first. library(packagename)

Functions for viewing data:
- kable(dataframename): used to display data in a print-friendly format.
- $ operator: extracts only specified variable, and returns a vector of length (row)

Variable Types:
- **Identification variable**: uniquely identifies each observational unit (like airport name, GRR, LAX, SNA, MDW, or Gerald R Ford International Airport, Los Angeles International, John Wayne Airport, Chicago Midway International Airport, etc)
    - Good practice = ID variable on left side of dataframe
- **Measurement/Characteristic Variable**: describe the properties of each observational variable (like latitude, longitude, altitude, etc)

## Data Viz Basics Tutorial

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



**Mapping argument**: defines which variables are mapped to which axes in the graph.
- Mapping is always paired w/ aes()

**Graphing workflow:** common for making graphs w/ ggplot2
1. Start graph w/ ggplot()
2. Add elements to graph w/ geom_[] function
3. Select variables with the mapping = aes() argument

```
ggplot(data = <DATA>) +
   <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

Helpful hints:
"**+**" must be at the *end* of a line, not the start.

**Aesthetics**: an aesthetic is a visual property of the objects in your plot. Aesthetics include things like the size, the shape, or the color of your points.

- x, y, colour, size…
- Do this inside of mapping = aes (HERE)
- MAPPING tells us which variables to map *to which visual properties*
- The same variable can be mapped to multiple aesthetics.

If you want all your data to have the same color/size/anything, do this *outside* of the aes() function, like:
geom_point(mapping = aes(x=,y=), color = "blue") [need quotes for color names]

**Setting vs. Mapping**
Setting: set the aesthetic to a ***value*** in the visual space, set it outside of aes()
Mapping: map the aesthetic to a ***variable*** in the data space, map it inside of aes

| To ____ | the aesthetic to a ____ | in the ____ space, | do so ___ of aes(). |
|---|---|---|---|
| SET | value | visual | Outside |
| MAP | variable | data | Inside |

* if you need a *legend* to understand the color/shape/etc., then put it *inside* of aes().
* if there's no meaning, do it outside of aes()

**Geometric Objects**

- Geoms: the geometrical object that a plot uses to represent observations
- (like bar, line, boxplot, point, smooth [fitted line])
- See ggplot2 cheatsheet
- See any geom's help page (?geom_smooth)

Exercise 1:
What geom would you use to draw a:
Line chart: geom_line()
Boxplot: geom_boxplot()
Histogram: geom_histogram()
Area chart: geom_area()

Exercise 2:
\`se\` argument to geom_smooth(): add/remove a standard error ribbon around the smooth line

This is the ***G***rammar of ***G***raphics… or, ***gg***plot

# Lecture, Day 1: Intro to Data Science

What *is* Data Science??

- People —> computers —> people
- Collecting, analyzing, sharing/communicating, visualizing.
- "Systematic collection & study of data"
- A tool that helps us see

What does data science help you *see*??

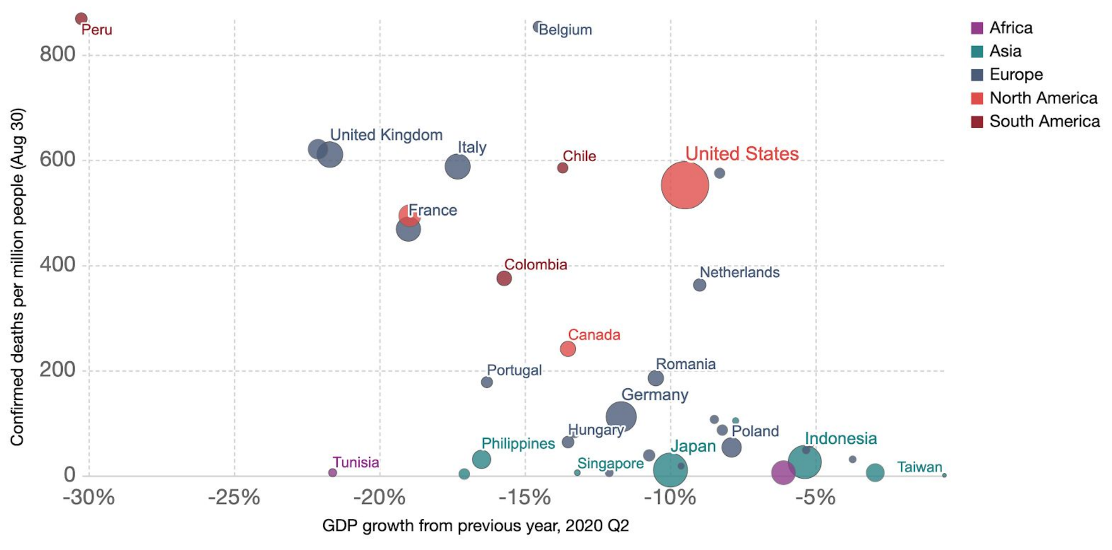
Viz, Inference, Prediction.

- See patterns on a large scale
- Show inequalities: every individual decision may seem fair, but *aggregated*, there's inequalities that appear. (trade off depth for breath)
- How countries are *doing* w/ COVID: able to use specific metrics, can geet overall trends.
    - U.S.: investing economically? Or, investing in health.
    - **VISUALIZATION:**
        - Econ:
        - Health: case/population, death/case,
        - Is there a correlation between econ & health?

Economic decline in the second quarter of 2020 vs rate of confirmed deaths due to COVID-19

Our World in Data

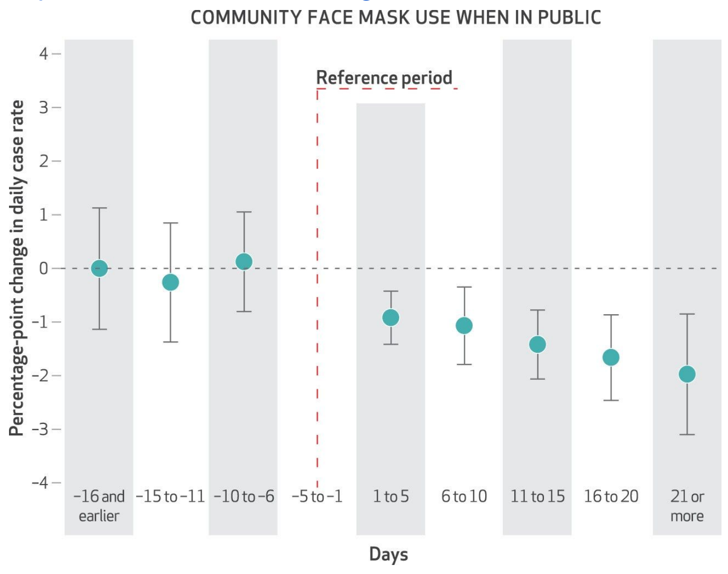
The vertical axis shows the number of COVID-19 deaths per million, as of August 30. The horizontal axis shows the percentage decline of GDP relative to the same quarter in 2019. It is adjusted for inflation.



Source: European CDC, Eurostat, OECD and individual national statistics agencies CC BY
Note: Limited testing and challenges in the attribution of the cause of death means that the number of confirmed deaths may not be an accurate count of the true number of deaths from COVID-19. Data for China is not shown given the earlier timing of its economic downturn. The country saw positive growth of 3.2% in Q2 preceded by a fall of 6.8% in Q1.

-
- Opposite of what you might expect… it doesn't just "trade off"
- Countries w/ better health = w/ better economy

- Just hit not as hard?
    - In the visualization with the relationship between COVID death rates and economic decline, I wondered if the relationship was due to how hard the countries were hit by COVID. For example, a county with a low death rate might simply be indicative of a low number of COVID cases, not necessarily of large health interventions put into place. This seemed to differ from the explanation you had provided in class, so my question (probably outside the scope of this class) would be what is really causing this relationship between death rate and GDP decline. Obviously, I would have to read the article to get more information on this, but I thought that was interesting.

- **INFERENCE**
    - Community face mask use when in public
    - https://www.healthaffairs.org/doi/10.1377/hlthaff.2020.00818
    - 

- **PREDICTION**
    - Focus of this class… *predictive analytics*
    - Used minivan price estimates
        - Based on past sales of used cars
        - The #s comse from data
    - Good model: Number tells you @ what price it'll be
    - Why is this useful? -- ultimately, the prediction from a model is used for the benefit of people.

This class…
- Covers:
    - wrangling

        - predictive modeling and validation
        - visualization and communication
    - but touches on all of the DS lifecycle.
    - Uses:
        - **<u>R</u>** (tidyverse, tidymodels, ggplot/plotly) the *<u>first</u>* time we see something
        - **<u>Python</u>** (Pandas, sklearn) the *<u>second</u>* time we see something
        - occasionally: SQL, other tools according to student interest -- there's some flexibility towards the end of the course
    - Goals:
        - *Skill*: how to do these things
        - *Knowledge*: understanding the underlying concepts
        - *Character*: wisdom in practicing these skills
          - humility (cite sources for data & processes, acknowledge limitations, transparent processes, validation of results) [RMarkdown -- the whole process is that code!]
          - integrity (resisting the temptation to manipulate data to get the answer you want) (evaluation claims, articulate analysis decisions and rationale, using exploratory analytics to validate data against assumptions)
          - hospitality (choose our tools to clarify, not obscuring) (good visualizations = hard to misunderstand, making your processes clear to others)
          - compassion & justice (don't cause harm; reveal it)
        - Bring up issues in Random channel, or email him.

## Day 2: Lab — Dino Data

Why R? -- it gives **names to concepts**
- Python: using syntax to do something
- R: use a named function to do that same thing)

Why git?
- reproducibility, hospitality, everybody uses it!

### Dino Data Lab -- Helpful Notes & Tips

**<u>pipe operator: %>%</u>**, takes what comes before it and sends it as the first argument to what comes after it. We can pronounce it "then".

**<u>Insert chunk</u>**: *CMD+OPT+i*

## Day 2.5: HW —

Getting started:
Open RStudio –> File –> New Proj –> Version Control –> Git –> Repository URL –> Tab –> Create!

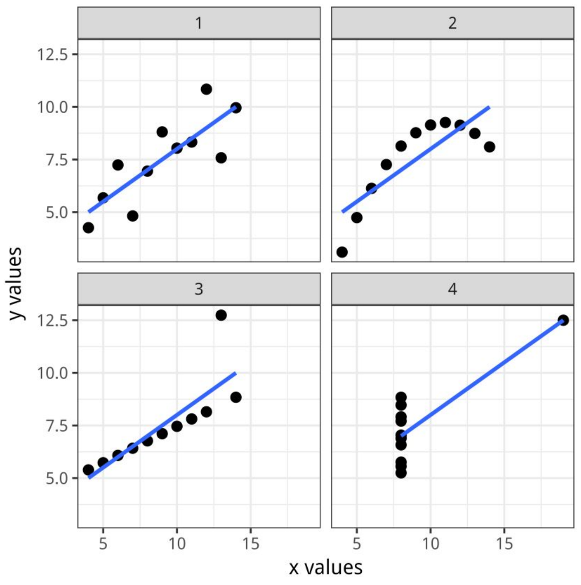
# ***Visualization 2***

## *WHY*: Healy "Look at Data" (from Data Visualization)

Look @ Data: *Look at the examples: can you explain to someone else what those examples show?*

**Anscombe's Quartet**: shows that standard measures of the association (correlation coeff, mean, med, mode, etc) aren't enough to tell you about data. This is why viz is important.

- summary statistics alone are not sufficient for data exploration



Types of problems with figures:

- Aesthetic -- ugly, inconsistent design choices
- Substantive -- due to data being presented
- Perceptual -- confusing/misleading because of how people perceive it

Tips to make things better:

- maximize the "data-to-ink" ratio
    - Delete backgrounds, gridlinees, superfluous axis marks, legends
    - YET: it's not memorable if it doesn't have all that extra stuff

Continue looking @ other plots on this website for good and bad plots. You know most of this intuitively or from DATA 101, though.

## *HOW*: ModernDive "Data Visualization"

Data Visualization: *Try to actually answer the "Learning Check" questions for yourself. Yes this takes longer than just skimming right past them. But they may show up on a quiz…*

5 Named Graphs:

1. scatterplots
2. linegraphs
3. histograms
4. boxplots
5. barplots

## *Application*: You did some visualization in Lab 1. How did that exercise relate to the "why" reading?

# Class, Week 2, Day 1: Meet the Toolkit

## Reproducible Data Analysis

- Someone can replicate your work & get the same results
- Share the steps (how) & the reasoning (why)
- "Near term goals:"
    - Can you ***re-make*** all tables and figures easily?
    - Does the code ***actually do*** what you think it does?
    - Is it clear ***why*** decisions were made? (e.g., how were parameter settings chosen?)
        - Rmarkdown lets you seamlessly embed a discussion about your code!
- "Long-term goals:"
    - Can the code be used for ***other data***?
        - Not great for excel, if you have references to rows 1:17… if you add a row 18, you have to change everything
    - Can you extend the code to ***do other things***?
- Ideally: document, explain, share everything (the code, data, & report)

## Toolkit

What we're using:

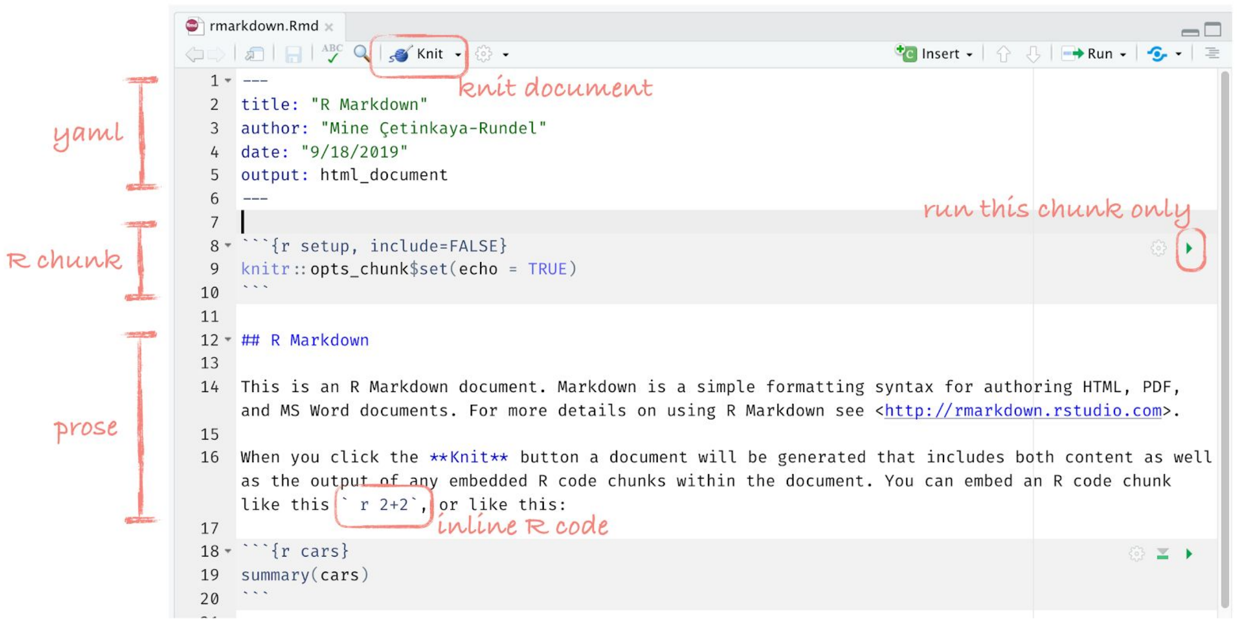
- Scriptability: R
- Literate programming (all in one place): R Markdown
- Version control: Git / GitHub

R

- Most common data type = "data frame"
        - Rows = same structure
        - Columns = same structure
    - Example: nursing: people, and their pulse, blood pressure, & other vitals
    - Example: `mtcars` in R
- The $ operator to access a variable w/in a data frame
- Functions = verbs (glimpse, view, make_plot)
    - You can arrange these into a pipeline
    - Something %>% another thing %>% etc.
- Package = basic unit shareable code
    - Tidyverse, ggplot2, dplyr…
    - Over 16k packages on CRAN (comprehensive R Archive Network)
        - Contain nice functions! Prepackaged for you! No need to program yourself!

        - How do you combine the pieces that exist to do what we want?
    - Using packages:
        - install.packages("x")
        - library(x) (or `require` if it's in a function…)
        - ?x (or, click their name in the page list in RStudio)

RMarkdown



~~~
---
title: "R Markdown"
author: "Mine Çetinkaya-Rundel"
date: "9/18/2019"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
~~~

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF,
and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well
as the output of any embedded R code chunks within the document. You can embed an R code chunk
like this `r 2+2`, or like this:

```{r cars}
summary(cars)
```
```

- *This is a yaml file?!?!?!* (metadata)
- R chunk
    - He names them?
    - Benefit of naming chunk
- Markdown
    - **bold**
    - *italics*
    - `in line r code`
- TIPS
    - Environment
        - 2 environments... R Markdown document env, vs. Console env
    - Always run a chunk after editing it

**GUAC**

CNTL+SHIFT+ALT: allows for clipboard functionality within guacamole

Vignettes = helpful text walkthrough

**TODAY'S QUESTIONS:**

Saving workspace?

- Always clear it, go into settings to set this as the default
```

Naming chunks?
- Shows up on bottom & is read

Logging out of guac?
- Log out as if you're a user logging out

Env
- Saverds

## Class, Week 2, Day 2: Fundamentals of Data Viz

Misc. Q & A:
- Assignment operators? <- and =

| | | |
|---|---|---|
| <- | assigns variables in current environment | data <- read_csv() |
| = | labels arguments to functions | ggplot(data = dino_data) |

- .Rmd vs. .md
    - html doesn't show up nicely in github, so if we make an md that works.
    - Knit (CMD+SHIFT+K)
- STEPS:
    1. Knit
    2. Commit (everything)
    3. Push

    !!!!!
- %>% takes the thing on the left and uses it as an argument for what's on the right.
    - glimpse(data)
    - data %>% glimpse()
- HOW TO ASK Qs:
    - Always include your code and the error
    - Create a minimum working example (we'll keep working on this throughout the semester)
    - Use code formatting

GAPMINDER: see 0909healthandwealth.Rmd

~~~
```{r}
library(tidyverse)
library(ggplot2)
library(dplyr)
gapminder <- read.csv('https://sldr.netlify.app/data/gapminder_clean.csv')
ggplot(data = gapminder) +
~~~

```
  geom_point(mapping = aes(x = income, y = life_expectancy, colour = four_regions,
                           size = population, alpha = .1))
```

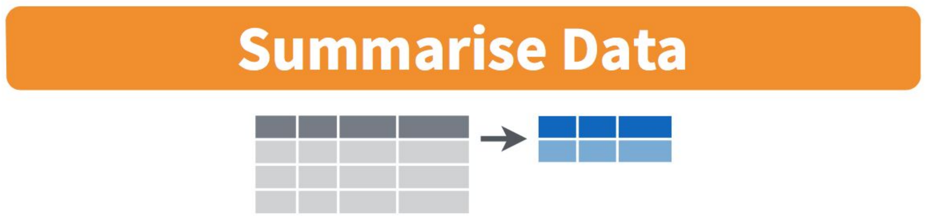
# ***Wrangling 1***

## Prep: ModernDive Ch3 & "Working w/ Data" Tutorial

***ModernDive Ch3***

- filter(): subset observations
- summarize():
- group_by():  assign different rows to be part of the same group.
- mutate(): mutate its existing columns/vars to create new ones
- arrange(): sort in ascending or descending order (use desc())
- join(): merge two data frames together using a "key" variable.

***3.3 `summarize` variables***

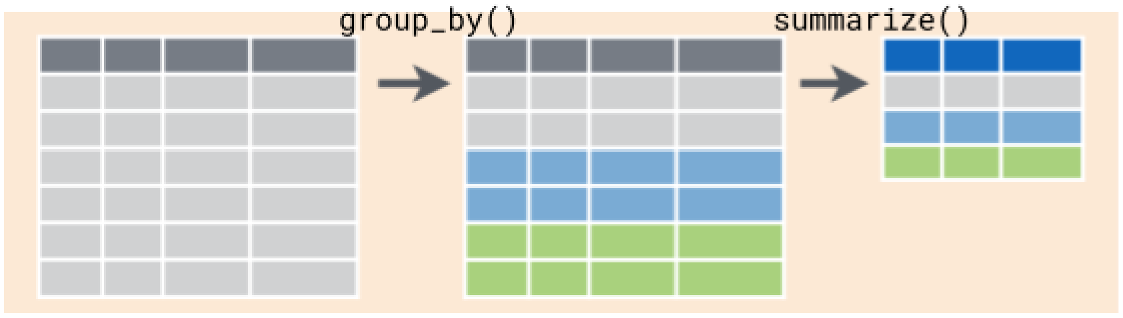


- Returns a data frame w/ only one row with summary statistics

-
  ```
  summary_temp <- weather %>%
      summarize(mean = mean(temp, na.rm = T),
                std_dev = sd(temp, na.rm = T))
  ```
- `na.rm`: remove NAs.

- *Functions that take many values & returns one. SUCH AS:*
- `mean()`: the average
- `sd()`: the standard deviation, which is a measure of spread
- `min()` and `max()`: the minimum and maximum values, respectively
- `IQR()`: interquartile range
- `sum()`: the total amount when adding multiple numbers
- `n()`: a count of the number of rows in each group. This particular summary function will make more sense when `group_by()` is covered in Section 3.4.

***3.4 `group_by` rows***



Uses:

- Computing 12 mean temperatures, one for each month separately
    - "Group" temp observations by the values of another variable (by month)
    - Use group_by —> summarise

*It is important to note that the `group_by()` function doesn't change data frames by itself. Rather it changes the meta-data, or data about the data, specifically the grouping structure. It is only after we apply the `summarize()` function that the data frame changes.*

To remove this grouping structure meta-data, pipe the resulting data frame into the `ungroup()` function

`sum()` vs. `n()`

- while `sum()` returns the sum of a numerical variable, `n()` returns a count of the number of rows/observations.

***Working w. Data***

*Animated Plots?:* Plotly, frame = date

## Class, Week 3, Day 1: Grammar of Data Transformation

ggplot2: concepts worn on their sleeve, obvious what is doing what
dplyr: similar…. But for data transformation instead of viz

- Functions = verbs
- Concepts show up again in Python w/ Pandas & SQL

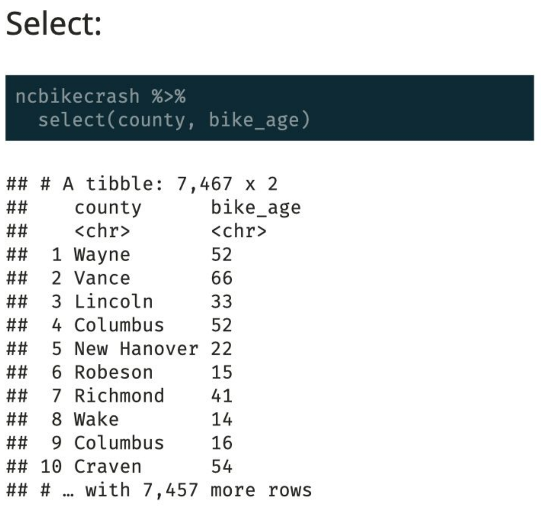
Grammar of Data Wrangling:

- `select`: pick columns by name

- `arrange`: reorder rows
- `slice`: pick rows by index(es)
- `slice_sample`: randomly sample rows
- `filter`: pick rows matching criteria
- `distinct`: filter for unique rows
- `mutate`: add new variables – maybe better renamed as derive
- `summarize`: reduce variables to values

- Rules for dplyr functions:
    - Don’t modify in place, it makes a new data frame
    - 1st argument: data frame
    - Subsequent args: what to do w/ that df
    - Returns: a data frame

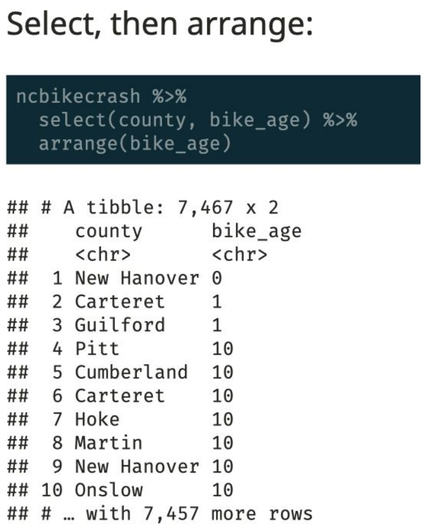
Looking @ Bike Crashes in NC 2007-2014 -- selecting only a few columns, and sorting by the values in a column



Select:

```
ncbikecrash %>%
  select(county, bike_age)
```

```
## # A tibble: 7,467 x 2
##    county      bike_age
##    <chr>       <chr>
##  1 Wayne       52
##  2 Vance       66
##  3 Lincoln     33
##  4 Columbus    52
##  5 New Hanover 22
##  6 Robeson     15
##  7 Richmond    41
##  8 Wake        14
##  9 Columbus    16
## 10 Craven      54
## # … with 7,457 more rows
```



Select, then arrange:

```
ncbikecrash %>%
  select(county, bike_age) %>%
  arrange(bike_age)
```

```
## # A tibble: 7,467 x 2
##    county      bike_age
##    <chr>       <chr>
##  1 New Hanover 0
##  2 Carteret    1
##  3 Guilford    1
##  4 Pitt        10
##  5 Cumberland  10
##  6 Carteret    10
##  7 Hoke        10
##  8 Martin      10
##  9 New Hanover 10
## 10 Onslow      10
## # … with 7,457 more rows
```

** be aware of ***data types***… if you try sorting bike_age in order, and if it’s of character type, it’s going to be 0, 1, 10, 2, etc… weird.

`%>%`: a pipe
- Normally: nested functions
    - `arrange(select(ncbikecrash, county, bike_age), bike_age)`
- w/ Pipes:  you "pipe" the output of the previous line of code as the first input of the next line of code

-
  ```
  ncbikecrash %>%
    select(county, bike_age) %>%
    arrange(bike_age)
  ```

Looking @ Hotel bookings

## Class, Week 3, Day 2: Practice with Data Transformation

- How to: sort in descending order
    - desc() around a variable w/ arrange()

- Class Structure Changes:
    - *enrichment activities*: Do one or two optional activities throughout the semester, report back to the class
        - ICPSR Data Fair next week, "Philosophy of Data Science" podcast, propose others!
    - *Data Science in the News:* share current events related to DS
        - Articles that use data science to tell a story
        - Articles about data science

# ***Wrangling 2***

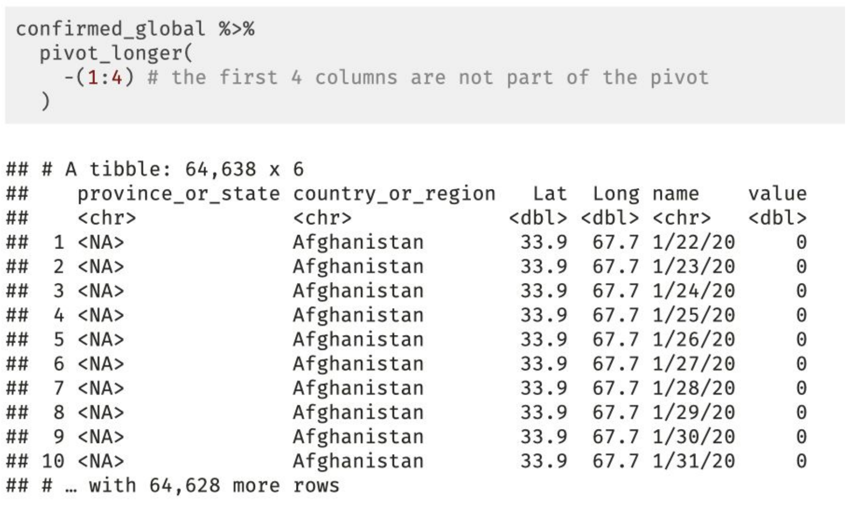
## Class, Week 4, Day 1:

To plot the number of covid cases over time, what should the table look like?
Each row is a case.

| DAY | STATE | COUNTY |
|---|---|---|
| 09/20/2020 | MI | Kent |

Or, what if each row is a state-day?

### Tidying Step 1: `pivot_longer`

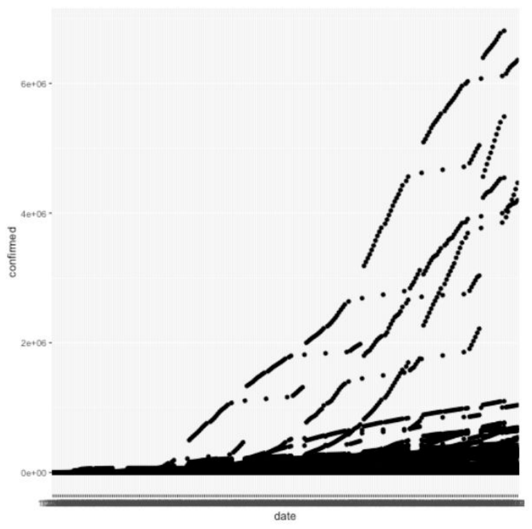


```
confirmed_global %>%
  pivot_longer(
    -(1:4) # the first 4 columns are not part of the pivot
  )
```

```
## # A tibble: 64,638 x 6
##    province_or_state country_or_region   Lat  Long name    value
##    <chr>             <chr>             <dbl> <dbl> <chr>   <dbl>
##  1 <NA>              Afghanistan        33.9  67.7 1/22/20     0
##  2 <NA>              Afghanistan        33.9  67.7 1/23/20     0
##  3 <NA>              Afghanistan        33.9  67.7 1/24/20     0
##  4 <NA>              Afghanistan        33.9  67.7 1/25/20     0
##  5 <NA>              Afghanistan        33.9  67.7 1/26/20     0
##  6 <NA>              Afghanistan        33.9  67.7 1/27/20     0
##  7 <NA>              Afghanistan        33.9  67.7 1/28/20     0
##  8 <NA>              Afghanistan        33.9  67.7 1/29/20     0
##  9 <NA>              Afghanistan        33.9  67.7 1/30/20     0
## 10 <NA>              Afghanistan        33.9  67.7 1/31/20     0
## # … with 64,628 more rows
```

```
confirmed_global_long <-
 confirmed_global %>%
 pivot_longer(
   -(1:4),
   names_to = "date",
   values_to = "confirmed"
 )
```

```
confirmed_global_long %>%
  ggplot(aes(x = date, y = confirmed)) +
  geom_point()
```

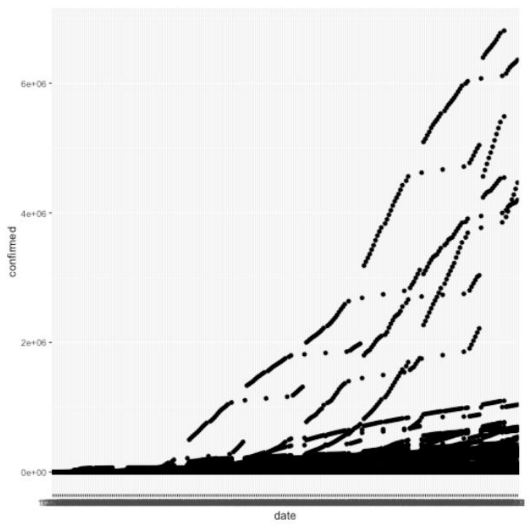


What is so terribly wrong with this graph?
Geom line would be even worse….?

## Class, Week 4, Day 2: Tidying & Joining Data

https://github.com/Calvin-DS202-FA20/lab04-E2

```
confirmed_global_long %>%
  ggplot(aes(x = date, y = confirmed)) +
  geom_point()
```



Date is kinda weird….

- Is it writing every single date value on top of each other?
- How is it sorting? (factor?)

Select! == heym I want this data

- select(-Lat, -Long)
    - Take out Lat & Long

<u>Homework 4: Python & Plotly</u>
Plot.ly and Seaborn.

```
%>% Code here!
```

# ***Wrangling 3***

## Class, Week 5, Day 1: Joining Data from Multiple Sources

**Discussion 3:** Collect example visualizations; post a critique;
- A visual you RESPECT, but DISAGREE with.
- Different POV, and reason WHY you disagree w/ it.
- From a different point on the spectrum

### `mutate()`
- Actually, poorly named.
- Should be: add computed column

### `count()` vs. `group_by %>% summarize()`
- Group by & summarize is better
- count() is mostly just shorthand for group_by(s) %>% summarize(n = n())

### `select` vs. `filter`
- Select: select_columns
- Filter: select_rows
- Filter IN rows. Keep these.

### `~ approach ~`
- Think about what you WANT
- And THEN write the code.
- Draw out in detail what you want the result to look like.
- If you don't know how to do the code from there, show your sketch to everyone else & ask on Q&A.

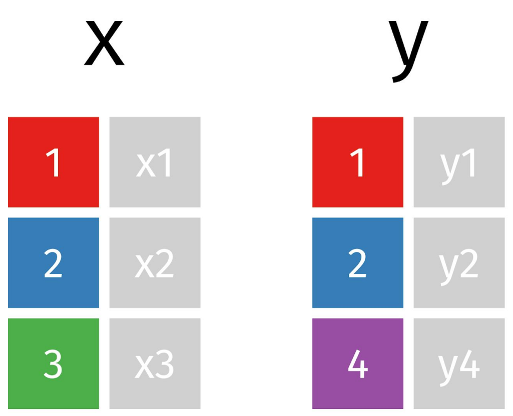
### `tests & assessments`
- Open-everything (except for people)
- Goal: be able to get conceptual foundation that can be used & applied in various other scenarios
- Not know how to do things very specifically from memory

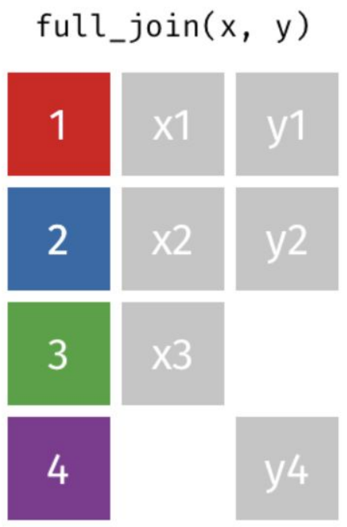
`joins!!!`

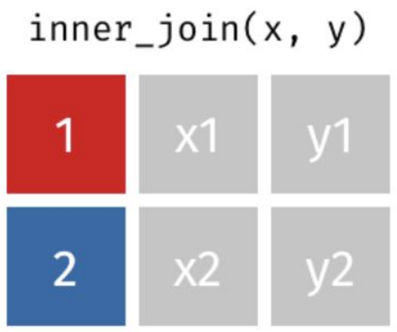
Data frame, x (COVID cases)

Extra information about the things in x, y (population of countries)

Needs a `key`: what has to match -- must match EXACTLY.







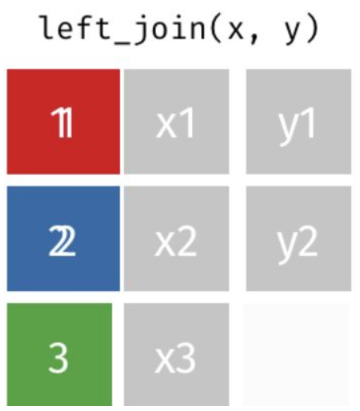
Types of Joins

*"What you do when things don't exactly line up."*

`FULL / OUTER`: leaves blanks (NA) for matches

`INNER`: drops rows w/ *any* mis-matchees

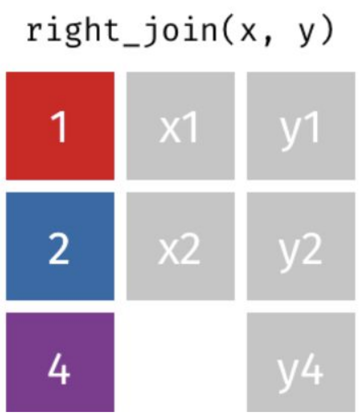
`LEFT / RIGHT`: drops rows where *one* of the sides has a mismatch



**<u>Going through Lab4:</u>**

How do you rename a column?

```
left_join(
    Confirmed_global_long %>%
        rename(country = country of region),
    Population,
    By = "country"
)
```



If you do `group_by` & `summarize`, you'll only ever get those columns you specified. So…

1. group_by(ADD IN YOUR EXTRA COLNAME!)

2. Add something to summarize??
3. Do join the opposite way?????

```
mutate(
    Country = case_when(
        TRUE ~ country_or_region  #default case! Reads:
                       #if true, use country_or_region
as metric
        ??
```

```
confirmed_global_long %>%
mutate(country = case_when(
  country_or_region == "Russia" ~ "Russian Federation",
  country_or_region == "US" ~ "United States",
  country_or_region == "Korea, South" ~ "Korea, Rep.",
  TRUE ~ country_or_region
))
```

## Class, Week 5, Day 2: Problem Solving

- Know your goal… have a very specific picture in mind of what you want

- **<u>Understand your data</u>**: at the input and at each step in a pipeline.
    - Read the first row out loud. What does it say?
    - How many rows are there? What does each row represent? Express what that row is telling you.
- **Think about your goal**
    - What should the first row of my output be?
    - If I had to work out each value in that row by hand, what parts of the input would I need?
- **Take small steps**
    - Look at the results after each step
    - Work out your computation for one datum first, then go to all of them.

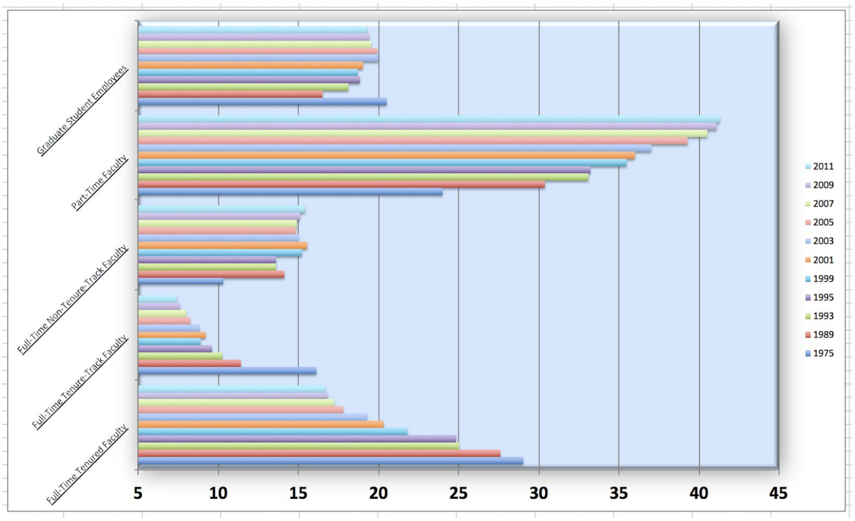
- Data science is **not magic**. Deliberate application of concepts and strategies will (eventually) bear fruit, no matter what language / library.

## Class, Week 5, Day 3: Data tidying and reshaping

### `tidyr`

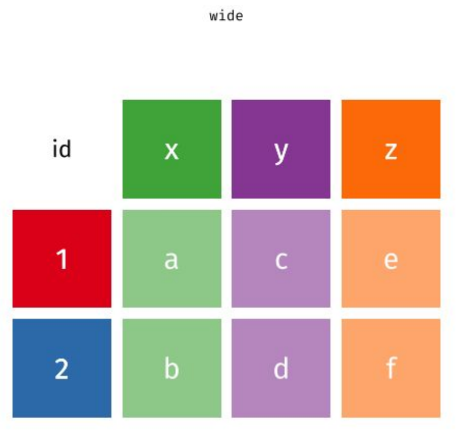
Data we're looking at:

- Trends in instructional staff employment
- Ugly graph :( -- it doesn't obviously show what it could so clearly & obviously show!

- 

| Faculty type | Year | Percentage of hires |
|---|---|---|
| Full time tenured | 1975 | 29 |
| Full time tenure track | 1975 | 16.1 |
| Full time non tenured | 1975 | 10.3 |
| Part time faculty | 1975 | 24 |
| Grad student | 1975 | 20.5 |
| [same] | 1989 | etc. |
|  |  |  |

We want: 5*11 = 55 total rows
Tidy dataframes are *longer.*



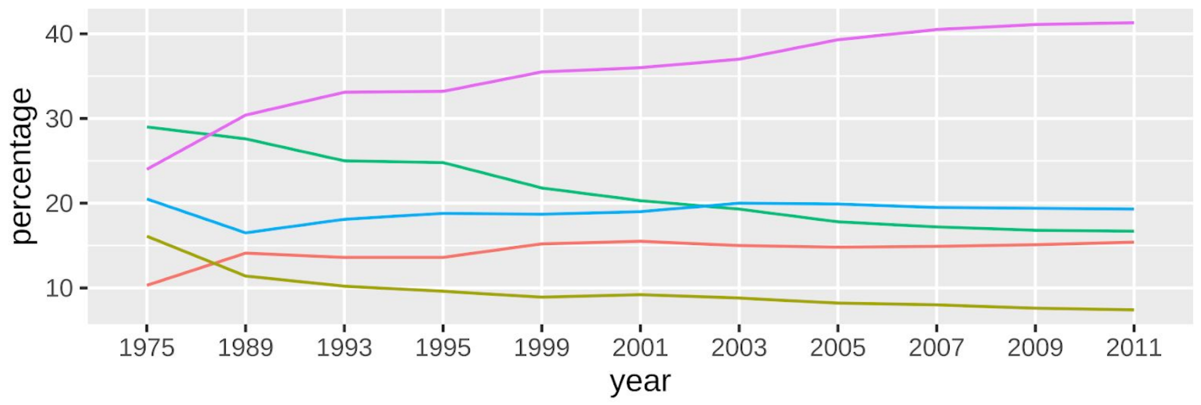
`pivot_longer(data, cols, names_to = "name", values_to = "value")`

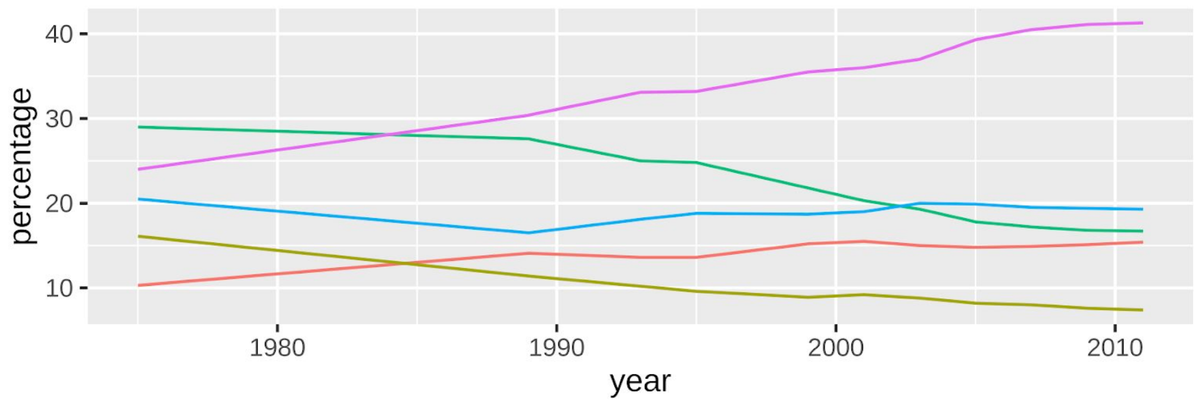
`cols` = which columns to pivot into longer format. Which do you want to change?
In our case, it's everything except for faculty_type.

`names_to = "name"` = name of the column we want it to put the column names ("year")

`values_to = "value"` = the name of the column to put the values ("percentage"

```
staff_long <- staff %>%
 pivot_longer(
   cols = -faculty_type,
   names_to = "year",
   values_to = "percentage"
   )
```





- Spacing of first one uses factor, or character
    - Doesn't maintain the distances, because it doesn't know what they MEAN...
- Second one uses the actually *date… this is a NUMBER*
    - `mutate(year = as.numeric(year))`

Voice Threads
Case_when
… it's like an if / elif / else in python.

```
case_when(
    age < 0  ~ “invalid”,
    age < 18 ~ “child”,
    TRUE     ~ “adult”
)
```

- First to True wins! (in both python and R)

- `TRUE` corresponds to else (the default)

Case_when can be used on vectors! Nice!!!
So, if…
Age <- c(-1, 0, 17, 18)
Case_when will be applied to *every element of the vector.*

### `Case_when vs. if_else`

*Which is preferable?*

```
if_else(
      age < 0, “invalid”,
            if_else(age < 18, “child”, “other”))
```

That’s gross -- case_when is just a lot more readable and understandable.

- Can be used within mutate!

Country == “United STates” ~ “USA”,
Iso3c == “GBR” ~ “UK”,
TRUE ~ country

NICEEEE

# Modeling 1

## Class, Week 6, Day 1: Predictive Modeling Intro

### Predictive Modeling: powerful tool to turn data into action

- Works because the universe is predictable (the world has actionable structure)
    - So, if we learn how to perceive that structure & act within it,
    - We can have better actions, be less surprised by what we see (predicting our perceptions)
- Need for wisdom -- great good & great harm....
- Harm due to...
    - Lack of Fairness: facial recognition, sentencing, lending, job applicant scoring
    - Lack of Transparency: how “Big Data” systems make conclusions
    - Lack of Privacy: as data is increasingly collected & aggregated
    - Amplification of extreme positions in social media, YouTube, etc.
    - Oversimplification of human experience
    - Hidden human labour
    - Illusion of objectivity

## Stating and refining the question

- *This is what data science tasks often start with!*
- 6 TYPES of questions
- Six types of questions
    - ***Descriptive***: summarize a characteristic of a set of data
        - severity of viral illnesses in a set of data collected from a group of individuals
    - ***Exploratory***: analyze to see if there are patterns, trends, or relationships between variables (hypothesis generating)
        - examine *relationships* between a range of dietary factors and viral illnesses
    - ***Inferential***: analyze patterns, trends, or relationships in representative data from a population
        - examine whether *any relationship* between dietary factors and viral illnesses found in the sample *hold for the population at large*
    - ***Predictive***: make predictions for individuals or groups of individuals
        - given a person's demographics and diet, *predict the severity* of illness
    - ***Causal***: whether changing one factor will change another factor, on average, in a population
        - whether people who were *randomly assigned* to eat a diet high in fresh fruits and vegetables or one that was low in fresh fruits and vegetables contract more severe viral illnesses
    - ***Mechanistic***: explore "how" as opposed to whether
        - *how* a diet high in fresh fruits and vegetables leads to a reduction in the severity of viral illnesses
- & of these, our focus: prediction….

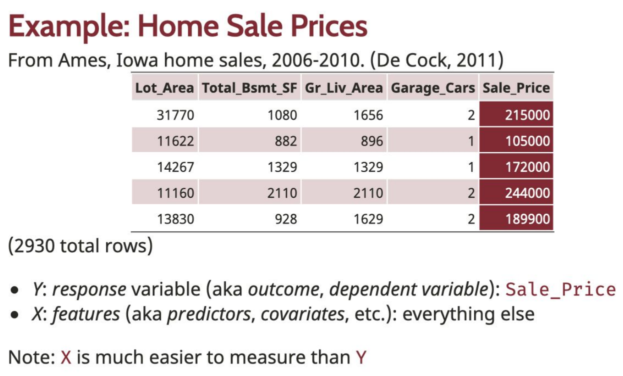
## Prediction

What we'll do:

- Predict something unknown from something known. Specifically: complete-the-table model

How we'll do it:

- methods that consider similar examples (Nearest Neighbors)
- methods that look at overall trends (linear/logistic regression)
- more advanced methods, time permitting



**Example: Home Sale Prices**

From Ames, Iowa home sales, 2006-2010. (De Cock, 2011)

| Lot_Area | Total_Bsmt_SF | Gr_Liv_Area | Garage_Cars | Sale_Price |
|---|---|---|---|---|
| 31770 | 1080 | 1656 | 2 | 215000 |
| 11622 | 882 | 896 | 1 | 105000 |
| 14267 | 1329 | 1329 | 1 | 172000 |
| 11160 | 2110 | 2110 | 2 | 244000 |
| 13830 | 928 | 1629 | 2 | 189900 |

(2930 total rows)

- *Y*: *response* variable (aka *outcome*, *dependent variable*): `Sale_Price`
- *X*: *features* (aka *predictors*, *covariates*, etc.): everything else

Note: X is much easier to measure than Y

-

## Class, Week 6, Day 2: Predictive Modeling Intro

**Types of Tasks**

**Regression:** predict a *number* ("continuous") -- number should be "close" to the correct number
**Classification**: predict a *category* -- 2 groups? 500000 groups? How likely is it to be in group *i*?

| **Regression** | **Classification** |
|---|---|
| How much rain in GR next year? | Inside / outside of a restaurant? |
| How much will this home sell for? | Is this person having a seizure? |
| How much time will this person spend watching this video? | Which word did this person mean to type? |
| How big a fruit will this plant produce? | Will this person "like" this post? |

Examples for Today:
CLASSIFICATION: The embrace2 -- a wrist-worn wearable that detects seizures. (seizure classication)
REGRESSION: Ames, IA home prices

### *~~ What makes a good prediction?: Regression ~~*

Measuring a good prediction… how do you know if it's a *good* prediction vs. a *bad* prediction?

***For one data point:***

**Residual:** actual – predicted (200 - 250 = –50k)
**absolute error:** |200-250 = –50k| = +50k
**squared error:** (50k)^2 = 2500k -- because being off my 50k might be more than twice as bad as 25k

***Across the entire dataset:***

**average error:** tend to predict too high? Too low?
**Max** absolute error
**Mean** absolute error
**Mean squared error** (MSE)
Normalized Squared Error: MSE / Variance
- & the confusingly named "R2" ($R^2$) = (1 – normalized squared error)

## *~~ What makes a good prediction?: Classification ~~*
Make a 2x2 matrix!

| | Seizure Happened | No Seizure Happened |
|---|---|---|
| Seizure Predicted | True + | *False* + (type 1 error) |
| No seizure predicted | *False* – (type 2 error) | True – |

**Accuracy** (% correct) **=** (TP + TN) / (# episodes)
**False negative** ("miss") **rate =** FN / (# actual seizures)
**False positive** ("false alarm") **rate =** FP / (# true non-seizures)

Trade Offs: for a seizure alert system, do you want sensitivity & specificity to be high or low?
- Ideally, you want both sensitivity and specificity to be high -- this will give you the most True Positives and True Negatives. For the seizure alert system specifically, it's probably better to have higher sensitivity (err on the side of too many false positives), because you could follow up and check if a seizure is actually occuring or not.

- If you prioritize sensitivity: a lot of false positives -- "boy who cried wolf" scenario
    - If it's important to catch every single event, you'll choose this.
    - If there's even a *slight* possibility that the event is occurring, prioritize sensitivity.
- If you prioritize specitivity: a lot of false negatives --

CLASS 10/09/2020

Mean Absolute Error:
- When we set incept to MEDIAN, we get lowest M.A.E.
- (Lowest M.S.E. when we set intercept to mean)

Training vs. Testing Dataset
- On test set, M.A.E. is higher (bad) :(
- Training (26.17)
- Testing (56.16)

ADDING Slope:
- Int: 121
- Slope: .0046 (lot area)
- MAE: 22.12
- MSE: 984.23
- OR…
- Int: 113
- Slope: .0046 (lot area)
- MAE: 23.59
- MSE: 932.41
- OR…
- Int = 141?
- Slope = .0002? Lot_Area
- 906.39?

Qs:
- When adding slope, i needed to adjust the intercept -- because
- Better error was achieved using slope & intercept, beecause more parameters = better fit.

Validation:
- You *must* validate your model on *unseen* data.
- No overfitting -- failure to generalize

## Class, Week 7, Day 2: P

RECAP: regression error measures
MAE: mean absolute error ($)
MSE: mean squared error ($^2)
RMSE: sqrt/(MSE) $ !! :)

If we're looking @ sale price in $s...

| Abbreviation | Error measure | Units |
|---|---|---|
| MAE | Mean *absolute* error | $ |
| MSE | mean *squared* error | $\$^2$ |
| RMSE | sqrt(MSE) | $ |

- Make sure to report it in units of understandable things.

*When to use what? (MSE, MAE…?)*

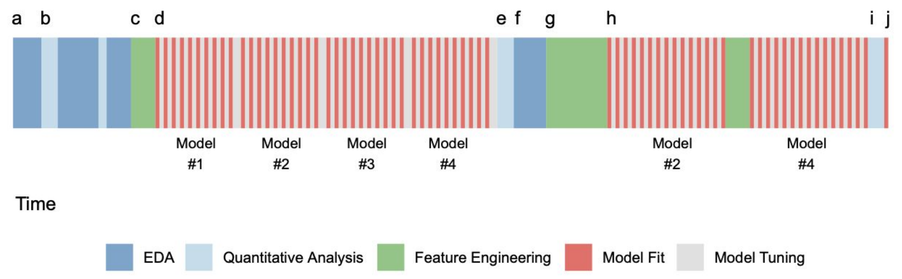
Think: use Mean or Median to summarize errors? (Mean: sensitive to outliers)

- (*) If the model is mostly good but makes a few large errors, is that *bad* (use MSE), or does it mean we should probably ignore those points as outliers (use MAE)
- Median minimizes MAE. Mean minimizes MSE.
  - What choice would minimize *max absolute* error?

Instead of finding coeffs by guessing, is there a better way?

- Practically: ML / stats. Just throw @ computer at it.
- Mathematically: Gradient descent
  - From your data, randomly pick a batch of a few observations.
  - Put those X's through your model, tracing the computations on the way.
  - Compute error (e.g., MSE) on that batch.
  - Compute what small change to each coefficient would have reduced error?
  - Make all of those small changes, repeat.

EDA: exploratory data analysis.



## Predictive Modeling Workflow

1. ***Define the problem***: predict *what*, based on *what*? What *metrics* will indicate success? (Measure success in multiple ways!)
2. ***Explore*** your data (EDA): understand its structure, make lots of plots

3. ***Pick a model***: Which type(s) of models are appropriate for task and data?
4. ***Transform the data*** as needed by the model ("feature engineering", preprocessing", "recipe")
5. ***Split the data*** to allow for validation.
6. ***Fit and evaluate the model***
7. ***Tune***: adjust model hyperparameters
8. ***Analyze model errors*** and refine all earlier steps

How we'll do this…

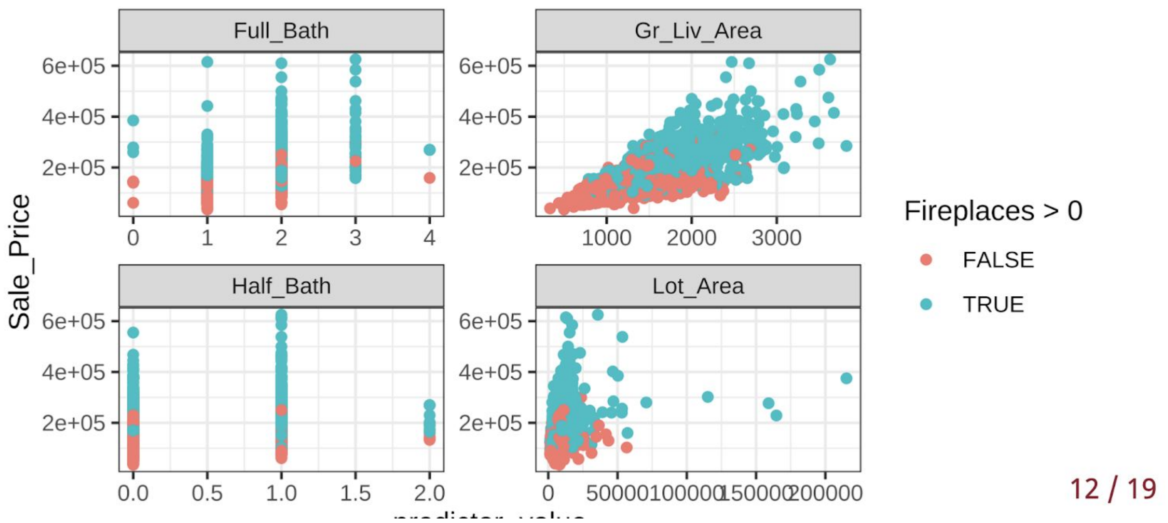
`library(tidymodels)`

- `parsnip`: ***Specify*** and ***train*** the model you want
- `recipes`: ***Prepare*** the data

- `rsample`: ***Split*** data into training and validation
- `yardstick`: Compute ***metrics*** for performance
- `tune`: Helps you set the dials.

Adsf

```
ames %>% select(Sale_Price, Gr_Liv_Area, Lot_Area, Full_Bath, Half_Bath, Fireplaces) %>%
  pivot_longer(-c(Sale_Price, Fireplaces), names_to = "predictor", values_to = "predictor_value") %
  ggplot(aes(x = predictor_value, y = Sale_Price, color = Fireplaces > 0)) + geom_point() +
  facet_wrap(vars(predictor), scales = "free") + theme_bw()
```



Validation

Hold out some data to use for validation:

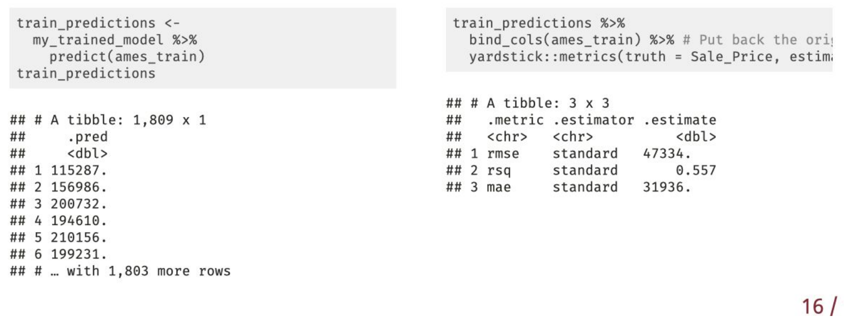
```
set.seed(10)
ames_split <- initial_split(ames, prop = 3/4)
ames_train <- training(ames_split)
ames_test <- testing(ames_split)
glue("Using {nrow(ames_train)} sales to train, {nrow(ames_test)} to test")
```

```
## Using 1809 sales to train, 603 to test
```

Specify the model to use.
Train the model on the training set

```
my_trained_model <- my_model_spec %>%
  fit(Sale_Price ~ Lot_Area + Gr_Liv_Area + Full_Bath, data = ames_train)
```

Make predictions on training set:



```
train_predictions <-
  my_trained_model %>%
    predict(ames_train)
train_predictions
```

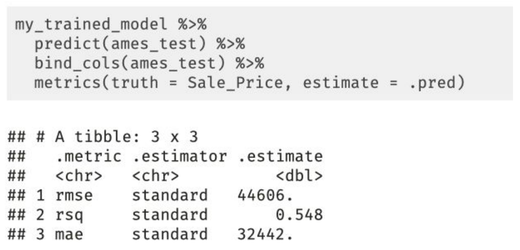
```
## # A tibble: 1,809 x 1
##     .pred
##     <dbl>
## 1 115287.
## 2 156986.
## 3 200732.
## 4 194610.
## 5 210156.
## 6 199231.
## # … with 1,803 more rows
```

```
train_predictions %>%
  bind_cols(ames_train) %>% # Put back the ori
  yardstick::metrics(truth = Sale_Price, estim
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard   47334.
## 2 rsq     standard       0.557
## 3 mae     standard   31936.
```

16 /

Evaluate on test set:



```
my_trained_model %>%
  predict(ames_test) %>%
  bind_cols(ames_test) %>%
  metrics(truth = Sale_Price, estimate = .pred)
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard   44606.
## 2 rsq     standard       0.548
## 3 mae     standard   32442.
```

## Class, Week 7, Day 3: Predictive Modeling

Formula interface

`y ~ x`

`y ~ x1 + x2 + x3`

* note: this doesn't include coefficients, right? Yeah. So it'd actually look more like:
y = c1*x1 + c2*x2 + c3*x3 + c4, yeah.

## Class, Week 8, Day 3: Feature Engineering

Project:

1. Data: source & assumptions
2. Vis design: retinal variables chosen for which data variables & why?

Midterm Quiz: Quiz 8 open for a week… similar structure to Quiz 7

RECIPES:

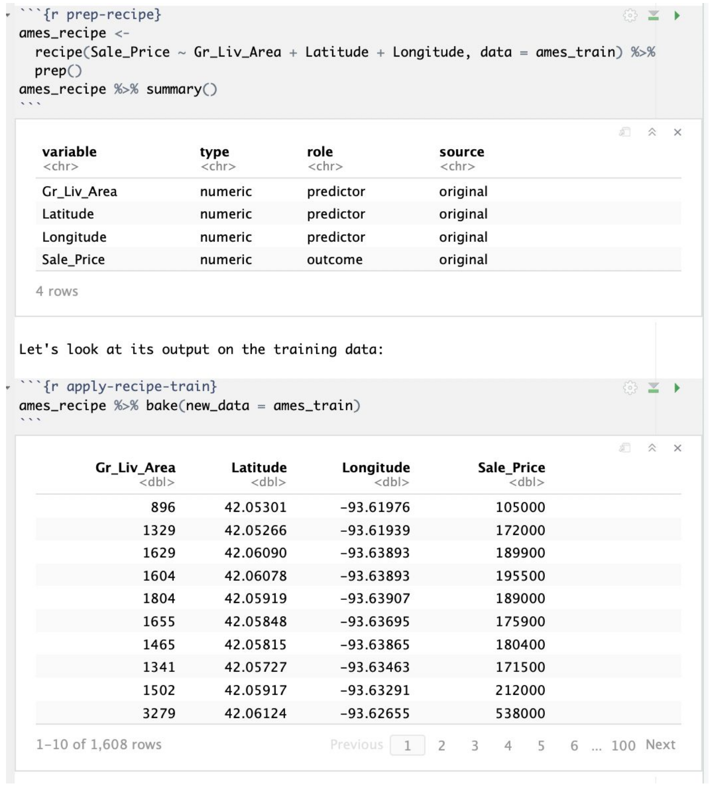
- A recipe is a data processing pipeline (like %>%) where the steps can be "smart".

Why Recipes:

- Add expressive power (like conditional logic) to simple modeels
- Make the model more (/less) understandable

```
ames_recipe <-
  recipe(Sale_Price ~ Gr_Liv_Area + Latitude + Longitude, data =
ames_train) %>%
  prep()
ames_recipe %>% summary()
```

Get the structure in place….



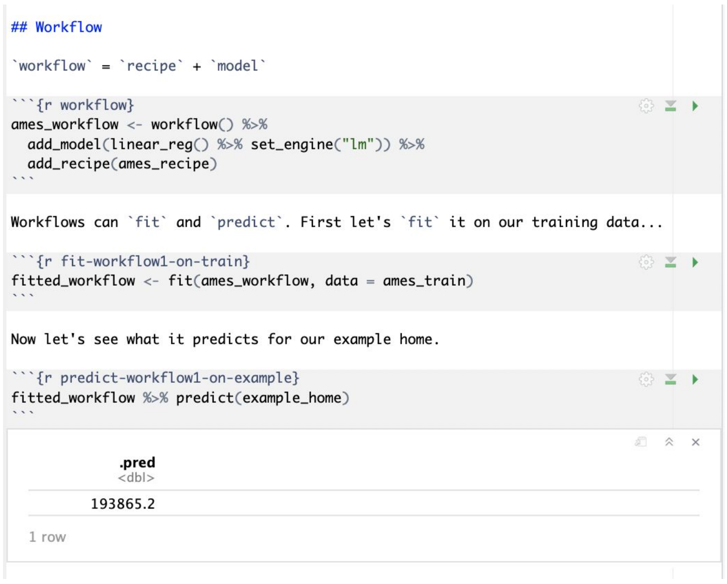
~~~
```{r prep-recipe}
ames_recipe <-
  recipe(Sale_Price ~ Gr_Liv_Area + Latitude + Longitude, data = ames_train) %>%
  prep()
ames_recipe %>% summary()
```
~~~

```
variable      type      role       source
<chr>         <chr>     <chr>      <chr>
Gr_Liv_Area   numeric   predictor  original
Latitude      numeric   predictor  original
Longitude     numeric   predictor  original
Sale_Price    numeric   outcome    original
4 rows
```

Let's look at its output on the training data:

~~~
```{r apply-recipe-train}
ames_recipe %>% bake(new_data = ames_train)
```
~~~

```
 Gr_Liv_Area   Latitude   Longitude   Sale_Price
       <dbl>      <dbl>       <dbl>        <dbl>
         896   42.05301   -93.61976       105000
        1329   42.05266   -93.61939       172000
        1629   42.06090   -93.63893       189900
        1604   42.06078   -93.63893       195500
        1804   42.05919   -93.63907       189000
        1655   42.05848   -93.63695       175900
        1465   42.05815   -93.63865       180400
        1341   42.05727   -93.63463       171500
        1502   42.05917   -93.63291       212000
        3279   42.06124   -93.62655       538000
1-10 of 1,608 rows          Previous 1 2 3 4 5 6 ... 100 Next
```



## Workflow

`workflow` = `recipe` + `model`

~~~
```{r workflow}
ames_workflow <- workflow() %>%
  add_model(linear_reg() %>% set_engine("lm")) %>%
  add_recipe(ames_recipe)
```
~~~

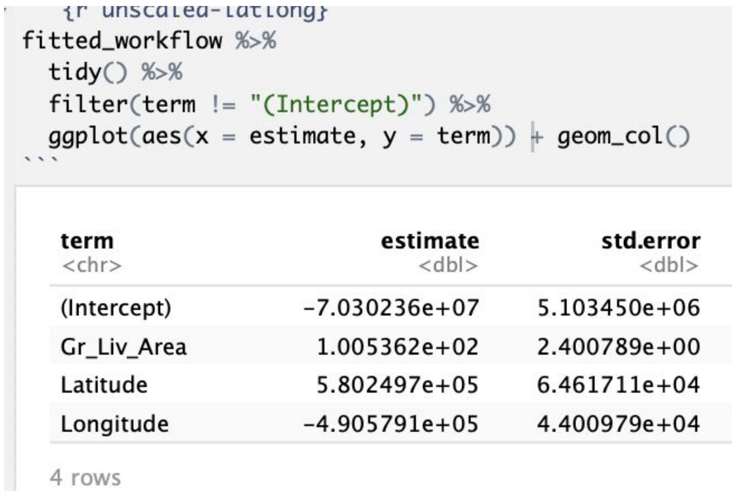
Workflows can `fit` and `predict`. First let's `fit` it on our training data...

~~~
```{r fit-workflow1-on-train}
fitted_workflow <- fit(ames_workflow, data = ames_train)
```
~~~

Now let's see what it predicts for our example home.

~~~
```{r predict-workflow1-on-example}
fitted_workflow %>% predict(example_home)
```
~~~

```
     .pred
     <dbl>
  193865.2
1 row
```



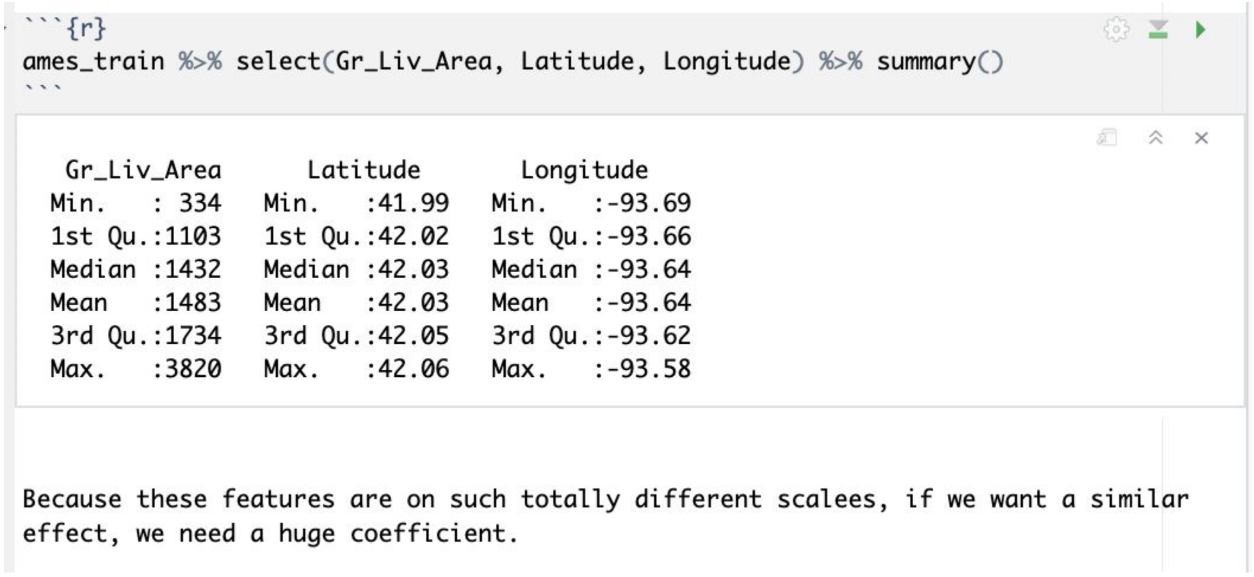
```
 {r unscaled-latlong}
fitted_workflow %>%
  tidy() %>%
  filter(term != "(Intercept)") %>%
  ggplot(aes(x = estimate, y = term)) + geom_col()
```

| term <chr> | estimate <dbl> | std.error <dbl> |
|---|---|---|
| (Intercept) | -7.030236e+07 | 5.103450e+06 |
| Gr_Liv_Area | 1.005362e+02 | 2.400789e+00 |
| Latitude | 5.802497e+05 | 6.461711e+04 |
| Longitude | -4.905791e+05 | 4.400979e+04 |

4 rows

We get huge coefficients...

Because these features are on such totally different scales, if we want a similar effect, we need a huge coefficient.



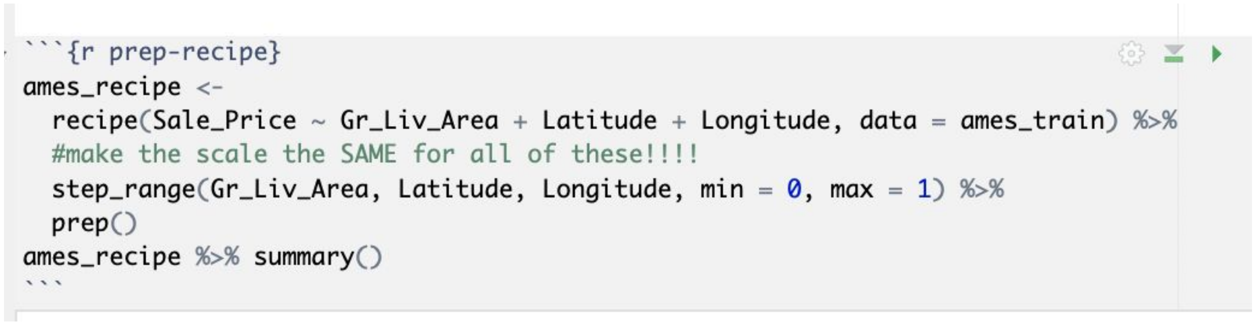
~~~
```{r}
ames_train %>% select(Gr_Liv_Area, Latitude, Longitude) %>% summary()
```
~~~

```
  Gr_Liv_Area       Latitude        Longitude     
 Min.   : 334   Min.   :41.99   Min.   :-93.69  
 1st Qu.:1103   1st Qu.:42.02   1st Qu.:-93.66  
 Median :1432   Median :42.03   Median :-93.64  
 Mean   :1483   Mean   :42.03   Mean   :-93.64  
 3rd Qu.:1734   3rd Qu.:42.05   3rd Qu.:-93.62  
 Max.   :3820   Max.   :42.06   Max.   :-93.58  
```

Because these features are on such totally different scalees, if we want a similar effect, we need a huge coefficient.

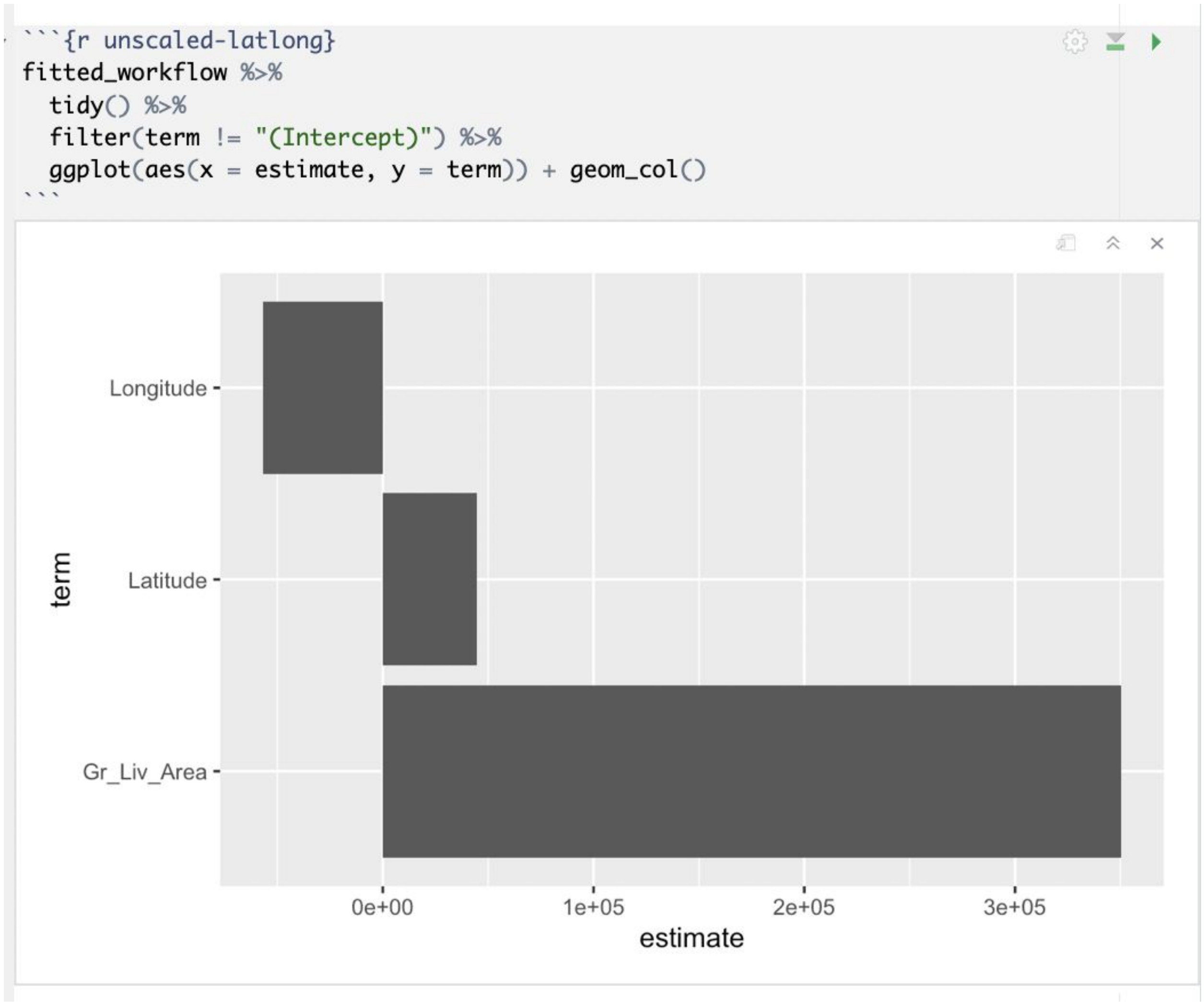
SO, ADD:

```
ames_recipe <-
  recipe(Sale_Price ~ Gr_Liv_Area + Latitude + Longitude, data =
ames_train) %>%
  #make the scale the SAME for all of these!!!!
  step_range(Gr_Liv_Area, Latitude, Longitude, min = 0, max = 1) %>%
  prep()
ames_recipe %>% summary()
```



~~~
```{r prep-recipe}
ames_recipe <-
  recipe(Sale_Price ~ Gr_Liv_Area + Latitude + Longitude, data = ames_train) %>%
  #make the scale the SAME for all of these!!!!
  step_range(Gr_Liv_Area, Latitude, Longitude, min = 0, max = 1) %>%
  prep()
ames_recipe %>% summary()
```
~~~

- Our units aren't helpful anymore… but it tells us about the relative weights of each one more easily.



~~~
```{r unscaled-latlong}
fitted_workflow %>%
  tidy() %>%
  filter(term != "(Intercept)") %>%
  ggplot(aes(x = estimate, y = term)) + geom_col()
```
~~~

-

## Class, Week 9, Day 1: Feature Engineering & Review

Q&A

Recipes vs. Data Wrangling Pipelines

- Recipe = a data wrangling pipeline
    - … that can be easily applied to new data (e.g., a test set)
    - … that can have learnable state (like ranges of data values)
    - (kinda like a… function??)
- Linear regression is linear, doesn't care what units the data is in… so the specific range didn't matter (0 to1), (-1 to 1), only the coeffs change

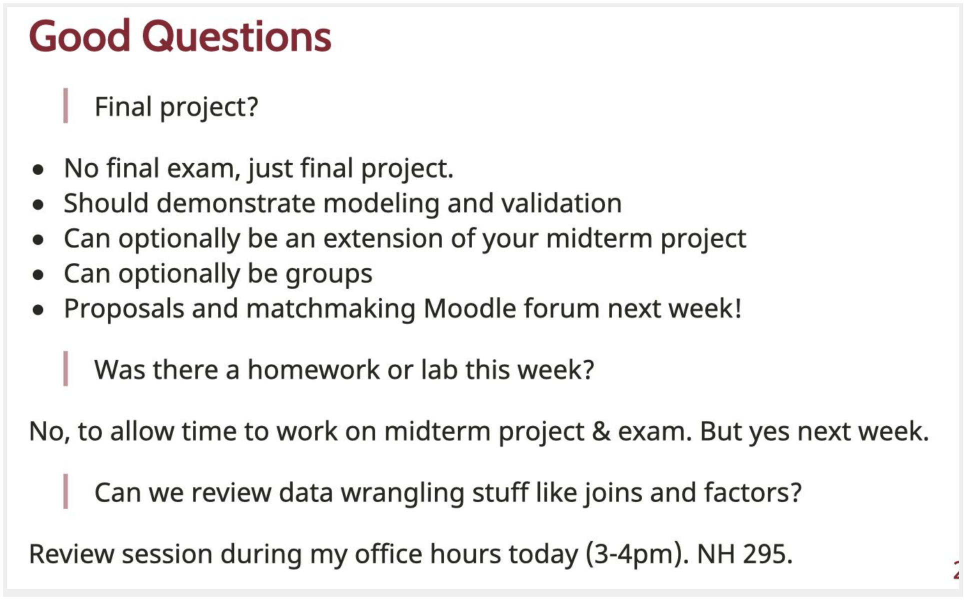
REVIEW

- Go over summarize(), group_by(), count()

```
rides %>%
     muate(weekendweekday = case_when(
               day_of_week == “Sat” ~ “weekend”,
               day_of_week == “Sun” ~ “weekend”,
               TRUE ~ “weekday”

Avg duration of ride by d.o.w.
Rides %>%
     group_by(day_of_week) %>%
     summarize()
```

## Class, Week 9, Day 2: Conditional Logic



**Good Questions**

> Final project?

- No final exam, just final project.
- Should demonstrate modeling and validation
- Can optionally be an extension of your midterm project
- Can optionally be groups
- Proposals and matchmaking Moodle forum next week!

> Was there a homework or lab this week?

No, to allow time to work on midterm project & exam. But yes next week.

> Can we review data wrangling stuff like joins and factors?

Review session during my office hours today (3-4pm). NH 295.

Today:

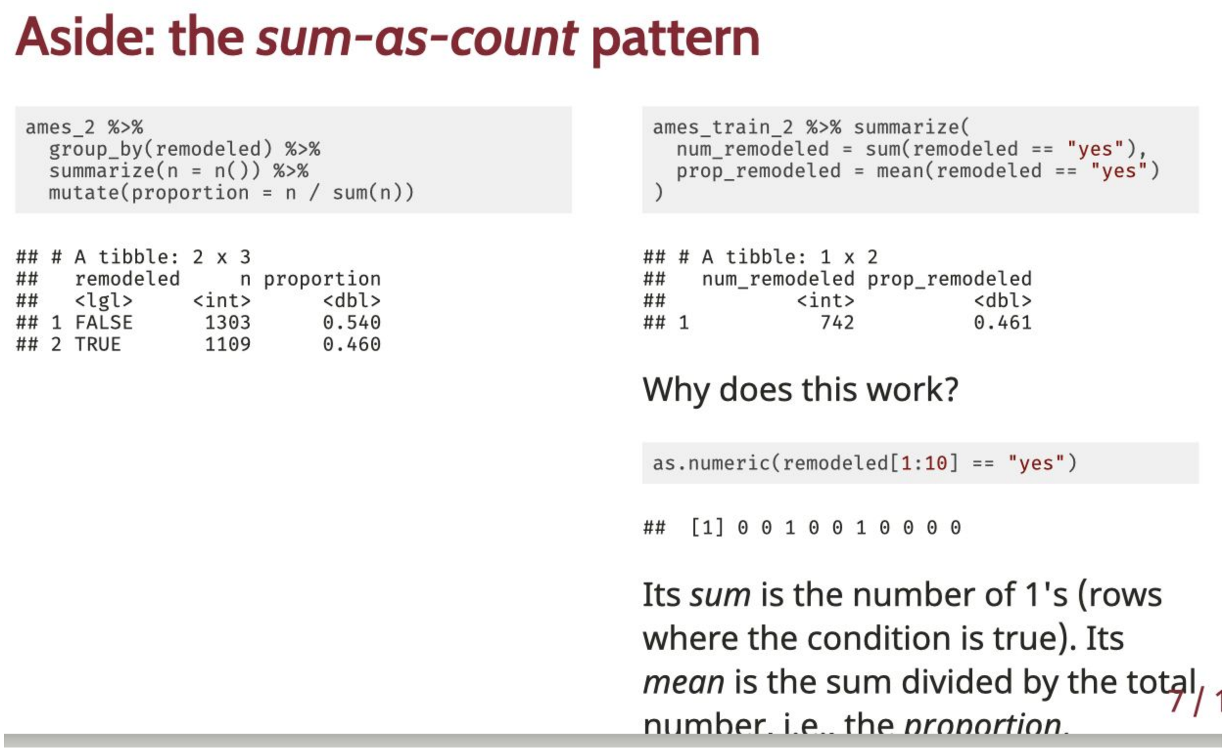
- Apply dummy encoding to add simple conditional logic to linear regression models
    - Explain how many columns get added in dummy encoding, and why
- Compare and contrast how linear regression and decision tree regression make predictions

Notes in ~/lab08-template

*What computations can a <u>linear model</u> do?*

- Add terms
- Multiple each term by a constant
- (that's it…)

He reviews papers -- interesting, i wonder which journal it's for…!



## Aside: the *sum-as-count* pattern

```
ames_2 %>%
  group_by(remodeled) %>%
  summarize(n = n()) %>%
  mutate(proportion = n / sum(n))
```

```
## # A tibble: 2 x 3
##   remodeled     n proportion
##   <lgl>     <int>      <dbl>
## 1 FALSE      1303      0.540
## 2 TRUE       1109      0.460
```

```
ames_train_2 %>% summarize(
  num_remodeled = sum(remodeled == "yes"),
  prop_remodeled = mean(remodeled == "yes")
)
```

```
## # A tibble: 1 x 2
##   num_remodeled prop_remodeled
##           <int>          <dbl>
## 1           742          0.461
```

Why does this work?

```
as.numeric(remodeled[1:10] == "yes")
```

```
##  [1] 0 0 1 0 0 1 0 0 0 0
```

Its *sum* is the number of 1's (rows where the condition is true). Its *mean* is the sum divided by the total number, i.e., the *proportion*.

7 / 1

```
Ames_2 %>%
  group_by (remodeled) %>%
  summarize(n = n() %>%
  mutate(proportion = n / sum(n))
```

~ 46% remodelled

```
Sum of a boolean, counting the truths
Ames_2 %>%
  summarize(num_remodelled = sum(remodelled == "yes"),
            Prop_remodelled = mean(remodelled == “yes”))
<--INSERT MONDAY’S NOTES HERE!!-->
```

## Class, Week 10, Day 2: Cross-Validation

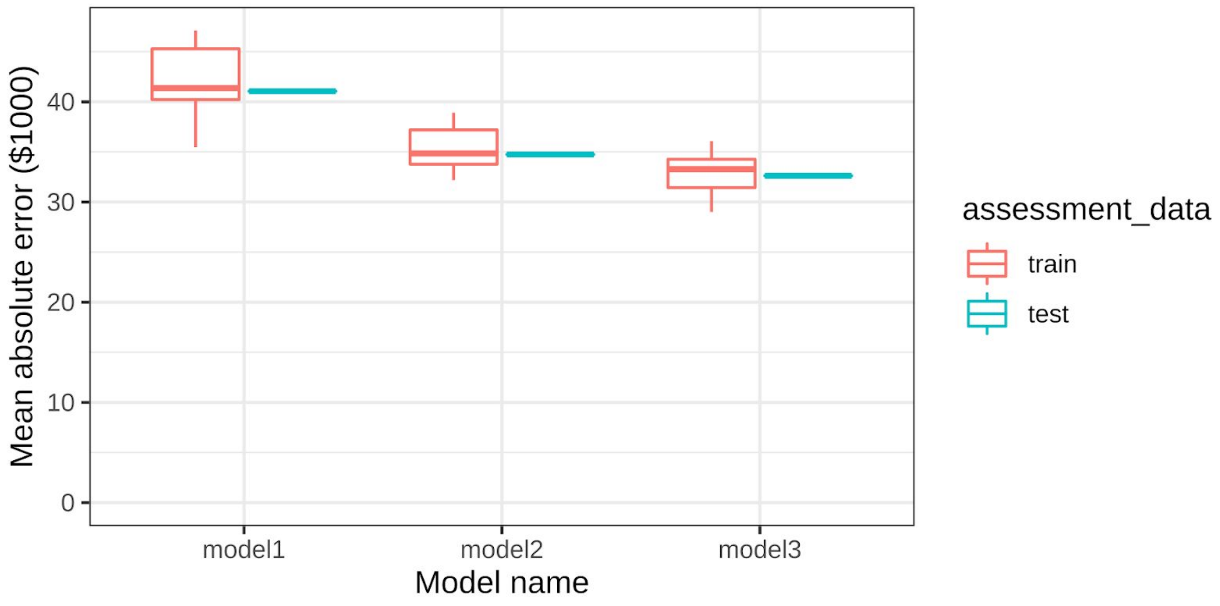
### Q&A on decision trees:

- Decision tree vs. lin regression?
    - Decision tree: crisp regions are easy, smooth variation is hard
    - Linear regression: smooth variation is easy; crisp regions are hard
- Linear regression can use different types of basis functions -- splines, sinusoids, rectifiers
    - Like more cyclic data (yearly), you don’t want lin.reg., you want something that’ll smooth the boundaries
- Limit to num of decisions?
    - Skldfj
- Can greedy algorithms be worse?
    - Yep… short-term gain gives long-term regret.
        - >> it’s a nice analogy too lol
    - Do outliers mess it up? Or why is this bad? -- not sure, not answered.

### Cross-Validation

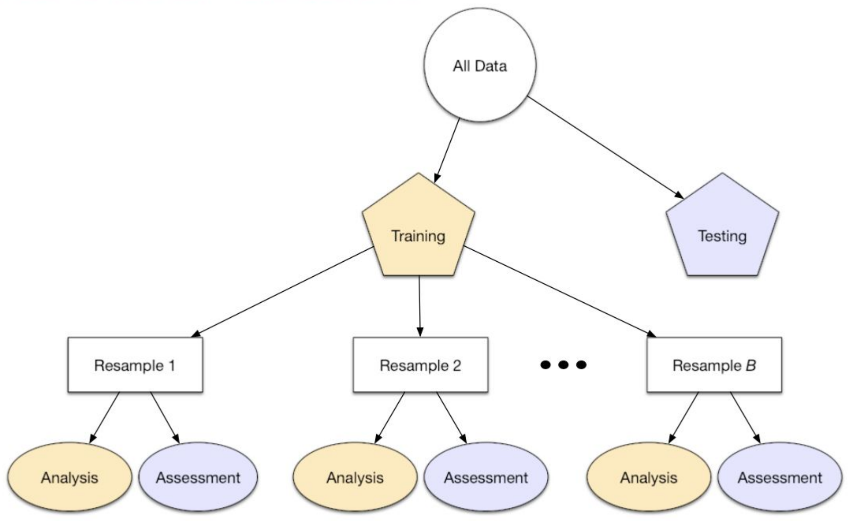
***Why?:*** measure accuracy on unseen data *without peek @ test set!!*

Lab 9: gives a much better assessment compared to

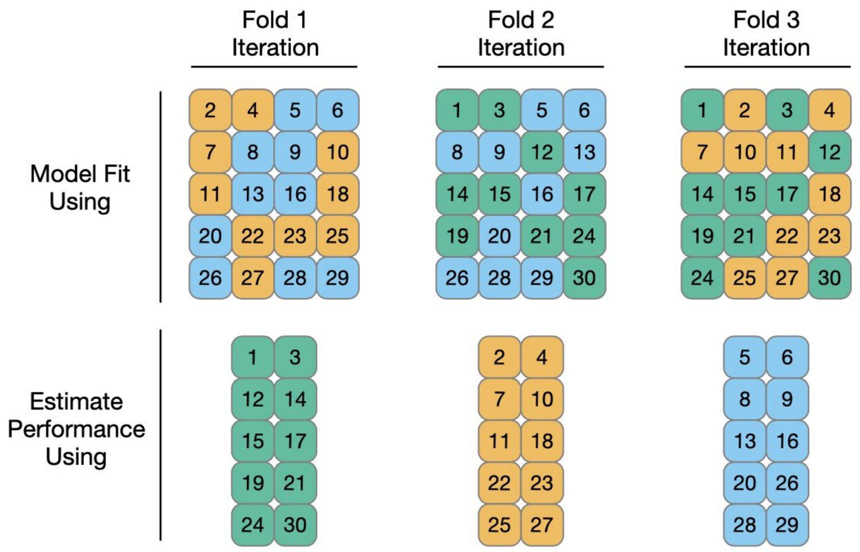


Estimates of model performance….

***What is it doing?***



- Resampling!
    - Draw with replacement?
    - Resample 1 MAY CONTAIN some of the same data points as Resample 2

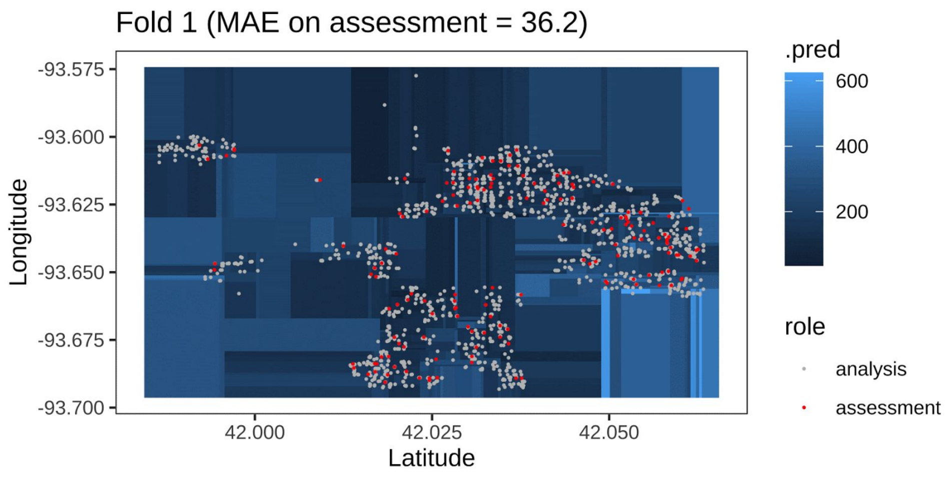
- 

    - Estimates are still somewhat related (not independent assessments) because they have some fitting overlap
    - Testing has no overlap
    - Good for moderate —> large datasets.
        - Nested cross-validation: reassign the folds multiple times

***How to do Cross-Validation?***

0. Ideally, have more data to (test/train) on???

1. Declare splitting strategy
    a. rsample::vfold_cv(v = 10)
    b. `Ames_resamples <- ames_train %>% vfold_cv(v = 10)`
2. Fit on each resample, evaluate using a set of metrics…
    a. Houses in the analysis set (grey, not red) tell us about the patterns the decision tree picks up on



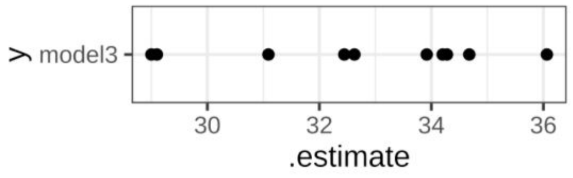
    b. FIT RESAMPLES!!

```
model3_samples <- model3_spec %>%
 fit_resamples(
   Sale_Price ~ Latitude + Longitude,
   resamples = ames_resamples,
   metrics = metric_set(mae))
model3_samples %>% collect_metrics(summarize = FALSE)
```

```
## # A tibble: 10 x 4
##   id     .metric .estimator .estimate
##   <chr>  <chr>   <chr>          <dbl>
## 1 Fold01 mae     standard        34.2
## 2 Fold02 mae     standard        31.1
## 3 Fold03 mae     standard        29.0
## 4 Fold04 mae     standard        33.9
## 5 Fold05 mae     standard        29.1
## 6 Fold06 mae     standard        36.1
## # … with 4 more rows
```

3. Plot and/or summarize the metrics
    a. PLOT:

```
model3_samples %>%
 collect_metrics(summarize = FALSE) %>%
 ggplot(aes(x = .estimate, y = "model3")) +
 geom_point()
```



b. SUMMARIZE:

```
model3_samples %>%
 collect_metrics(summarize = TRUE)
```

```
## # A tibble: 1 x 5
##   .metric .estimator  mean     n std_err
##   <chr>   <chr>      <dbl> <int>   <dbl>
## 1 mae     standard    32.7    10   0.751
```

TIDY WAY TO COMPARE MODELS:

1. Data frame of model specs

```
all_models <- tribble(
 ~model_name, ~spec,
 "model1",    decision_tree(mode = "regression", tree_depth
= 2),
 "model2",    decision_tree(mode = "regression", tree_depth
= 30),
 "model3",    decision_tree(mode = "regression",
cost_complexity = 1e-6, min_n = 2)
)
```

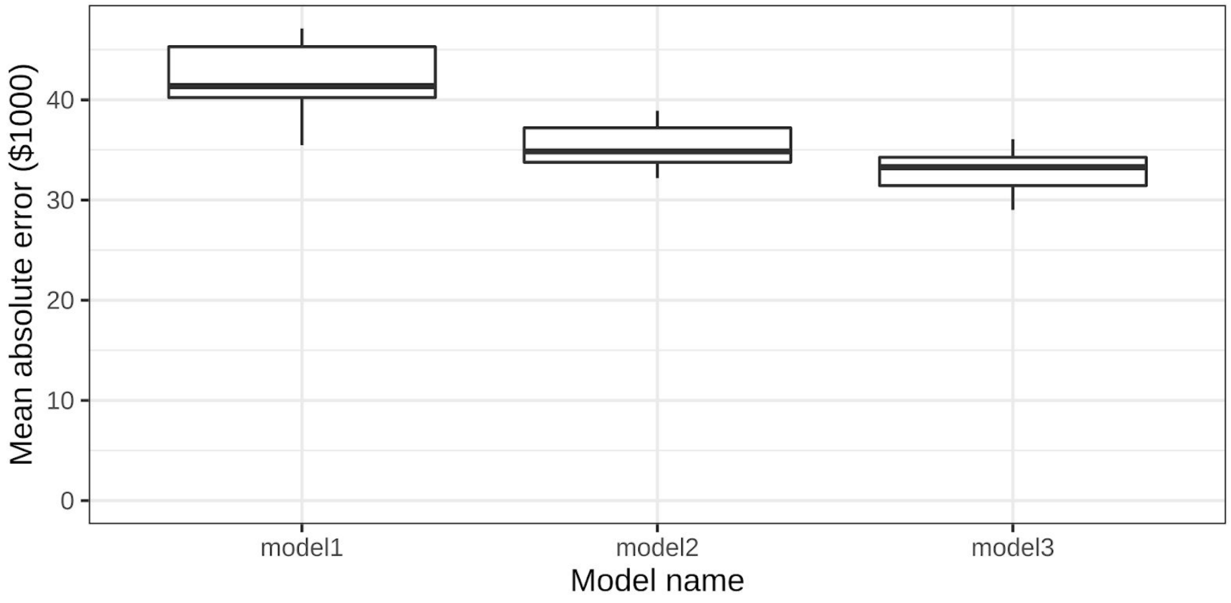
2. Sample each model (using dplyr::rowwise) -- essentially making a group for each row… a for loop?

```
models_with_samples <- all_models %>%
 rowwise() %>%
 mutate(samples = list(
   spec %>% fit_resamples(
     Sale_Price ~ Latitude + Longitude,
     resamples = ames_resamples,
     metrics = metric_set(mae))))
```

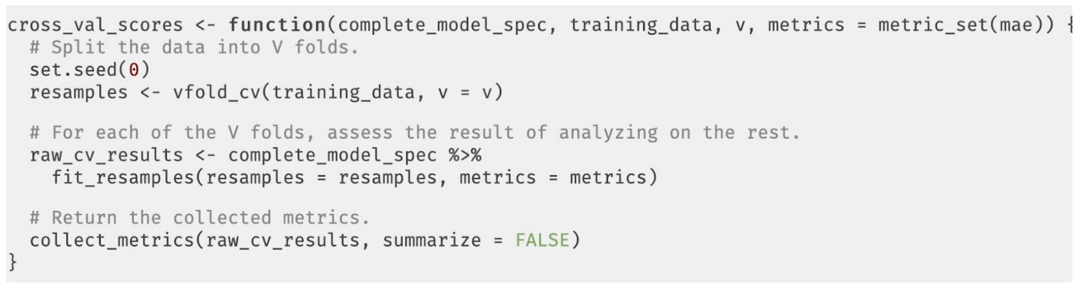
```
models_with_samples
```

```
## # A tibble: 3 x 3
## # Rowwise:
##   model_name spec       samples
##   <chr>      <list>     <list>
## 1 model1     <spec[+]> <tibble [10 × 4]>
## 2 model2     <spec[+]> <tibble [10 × 4]>
## 3 model3     <spec[+]> <tibble [10 × 4]>
```

```
models_with_samples %>%
 rowwise(model_name) %>%
 summarize(collect_metrics(samples, summarize = FALSE)) %>%
 ggplot(aes(x = model_name, y = .estimate)) +
   geom_boxplot() + labs(x = "Model name", y = "Mean
absolute error ($1000)") + coord_cartesian(ylim = c(0, NA))
```



Week 11 Day 1: Cross-Validation
Why?

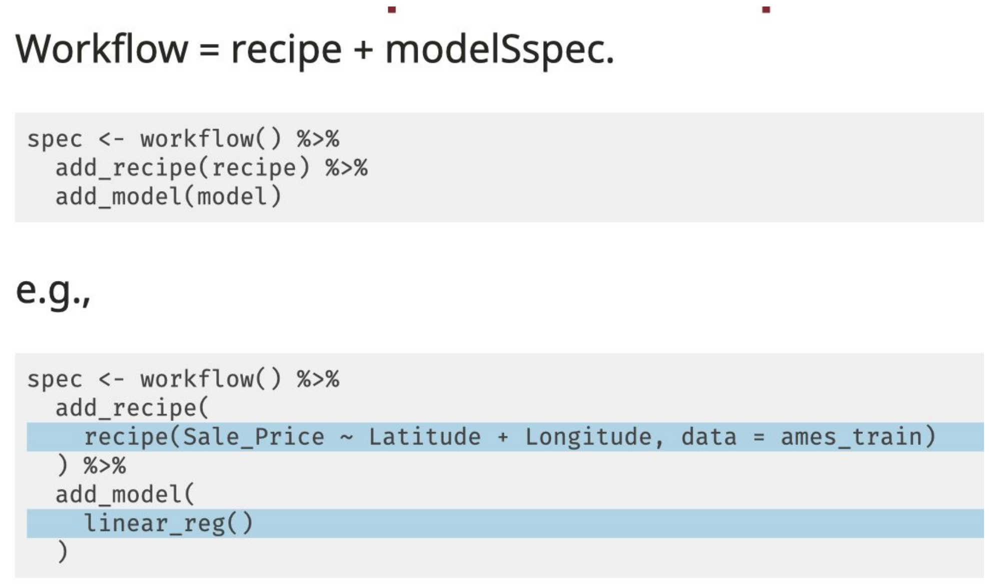
- Puzzle:
    - We want to pick the model that works best on unseen data.... but as soon as we try one model, we've peeked at the data!
- Solution:
    - Divide training data into V piles (e.g., 10)
    - Hide one pile from yourself.
        - train on ("analyze") the rest,
        - evaluate ("assess") on the one you held out.
    - Repeat for each of the V piles.
    - 

```
cross_val_scores <- function(complete_model_spec, training_data, v, metrics = metric_set(mae)) {
  # Split the data into V folds.
  set.seed(0)
  resamples <- vfold_cv(training_data, v = v)

  # For each of the V folds, assess the result of analyzing on the rest.
  raw_cv_results <- complete_model_spec %>%
    fit_resamples(resamples = resamples, metrics = metrics)

  # Return the collected metrics.
  collect_metrics(raw_cv_results, summarize = FALSE)
}
```

A model spec?

- A workflow: recipe e+ model_spec
- A model spec is like a class, and a model is like a specific instance of a class.
- 

Workflow = recipe + modelSspec.

```
spec <- workflow() %>%
  add_recipe(recipe) %>%
  add_model(model)
```

e.g.,

```
spec <- workflow() %>%
  add_recipe(
    recipe(Sale_Price ~ Latitude + Longitude, data = ames_train)
  ) %>%
  add_model(
    linear_reg()
  )
```

## Week 10, Day 2: Classification

~ Classification: Which of several categories? ~

- Problem intro: EDA, Data wrangling in R
- Classification workflow:
    - Models: decision tree, logistic regression
    - Model outputs: scores and decisions
    - Model metrics: accuracy, sensitivity, specificity
    - Validation

Logistics notes:

- Project should be:
- ***Interesting:*** should be interesting in *some* aspect… really thorny data wrangling, or really interesting modeling, or bringing it together with another dataset, (gene annotations / systems), or asking really interesting questions of it. >> *think: which parts are the interesting parts*
- ***Your own:*** ask a different question of the same data; here's two ways to do this, and here's what I think is better