

Week 3: Tidy Data and Data Wrangling

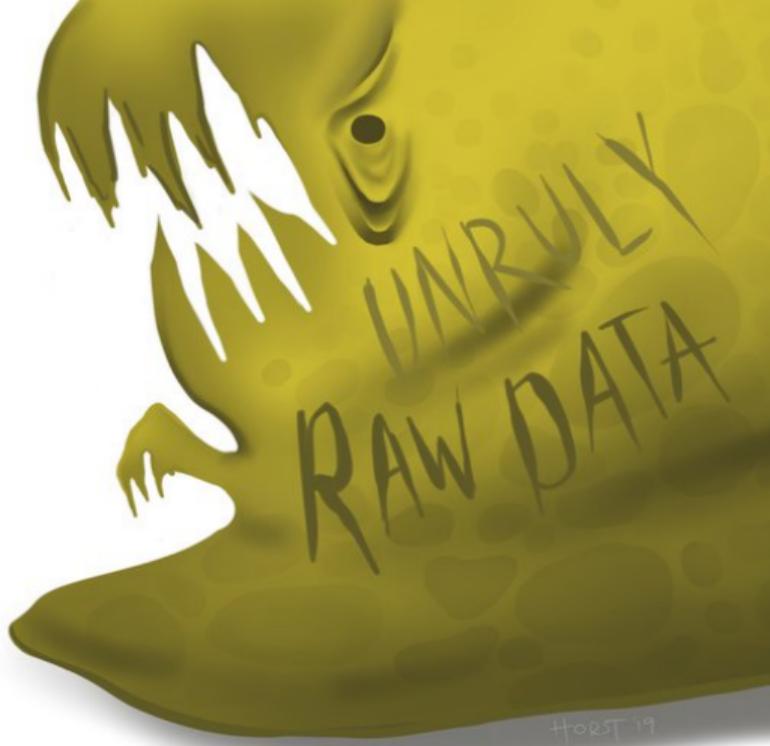
with dplyr (and a small mention of tidyr)

Scott Schwartz

May 18, 2021



^ Us / the plan 4 2-day



@Allison_Horst

Class Check Round 1

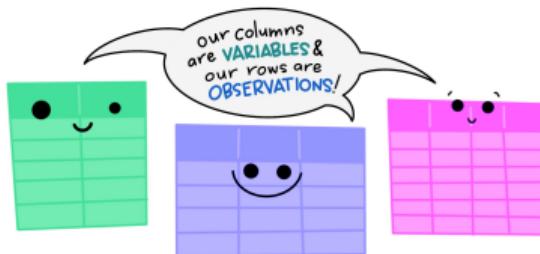
<https://pollev.com/sta> (3 questions)

Tidy Data

"Happy families are all alike; every unhappy family is unhappy in its own way." – Tolstoy

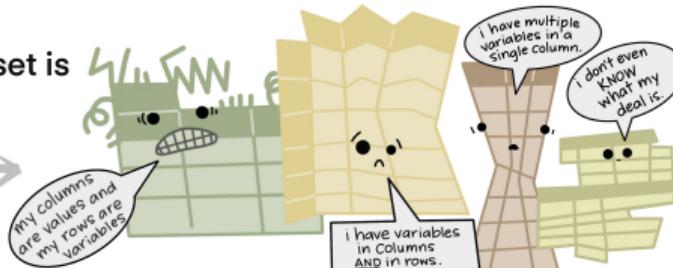
Standardized "Tidy Data" means much less work...

The standard structure of
tidy data means that
"tidy datasets are all alike..."



...but every messy dataset is
messy in its own way."

—HADLEY WICKHAM



@Allison_Horst

*"Write code for humans,
but data for computers"*

– Vince Buffalo, author of
Bioinformatics Data Skills

*"It's often said that 80%
of data analysis is spent
on the process of cleaning
and preparing the data"*

– Hadley Wickham, author
of the [R4DS](#) and [dplyr](#)

Tidy Data

“Tidy Data” really just means “‘Easy for a Computer to Use’ Data”

country	year	cases	population
Afghanistan	1989	745	1537071
Afghanistan	2000	1666	2059360
Brazil	1989	3737	17206362
Brazil	2000	84488	174504898
China	1989	210258	127235272
China	2000	21666	128042583

variables

country	year	cases	population
Afghanistan	1989	745	1537071
Afghanistan	2000	1666	2059360
Brazil	1989	3737	17206362
Brazil	2000	84488	174504898
China	1989	210258	127235272
China	2000	21666	128042583

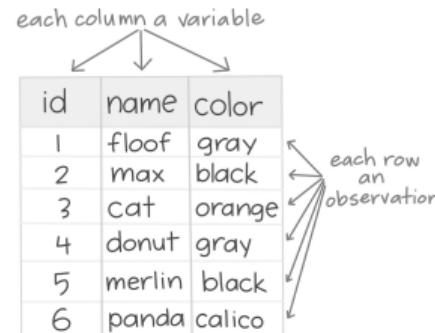
observations

country	year	cases	population
Afghanistan	1989	745	1537071
Afghanistan	2000	1666	2059360
Brazil	1989	3737	17206362
Brazil	2000	84488	174504898
China	1989	210258	127235272
China	2000	21666	128042583

values

In tidy data:

- each variable forms a column
- each observation forms a row
- each cell is a single measurement



@R4DS and @Allison_Horst

Quiz 1 on Tidy Data

Which of these tables is organized in the "Tidy Data" format?

Which most easily plots the distribution of grades for all of the students?

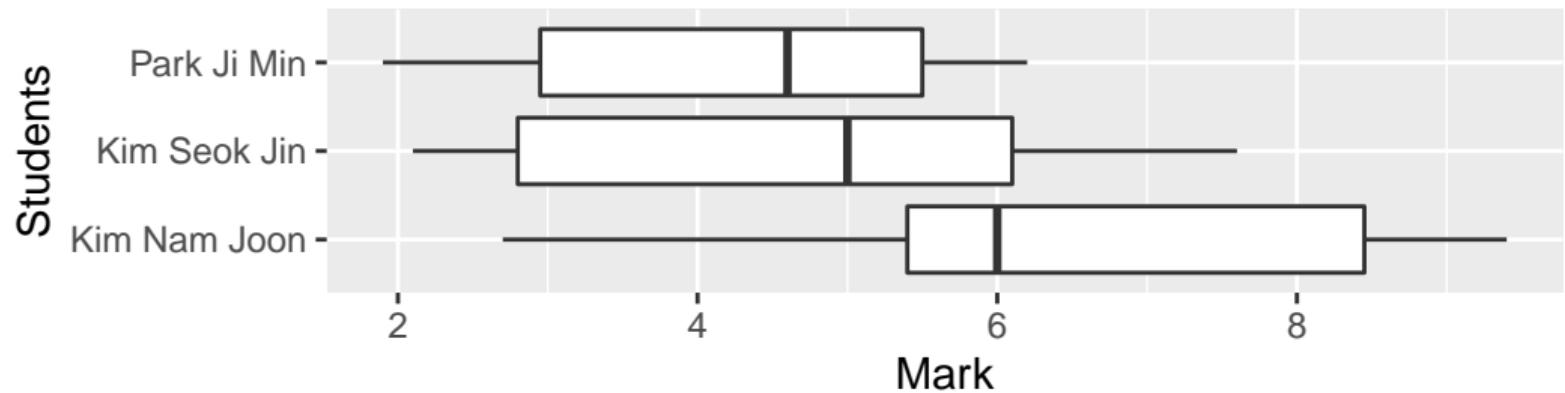
```
## # A tibble: 3 x 8
##   Students    Mark1  Mark2  Mark3  Mark4  Mark5  Mark6  Mark7
##   <chr>     <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 Kim Nam Joon  9.2    9.4     6     2.7    7.7    5.7    5.1
## 2 Kim Seok Jin  5.1     5     2.1    7.1    2.7    7.6    2.9
## 3 Park Ji Min  1.9     6     4.6    6.2    5     2.9    3

## # A tibble: 7 x 4
##   `Kim Nam Joon` `Kim Seok Jin` `Park Ji Min` Marks
##   <dbl>          <dbl>          <dbl> <chr>
## 1 9.2            5.1            1.9  Mark1
## 2 9.4            5              6    Mark2
## 3 6              2.1            4.6  Mark3
## 4 2.7            7.1            6.2  Mark4
## 5 7.7            2.7            5    Mark5
## 6 5.7            7.6            2.9  Mark6
## 7 5.1            2.9            3    Mark7

## # A tibble: 21 x 3
##   Marks Students    Mark
##   <chr> <chr>     <dbl>
## 1 Mark1 Kim Nam Joon  9.2
## 2 Mark1 Kim Seok Jin  5.1
## 3 Mark1 Park Ji Min  1.9
## 4 Mark2 Kim Nam Joon  9.4
## 5 Mark2 Kim Seok Jin  5
## 6 Mark2 Park Ji Min  6
## 7 Mark3 Kim Nam Joon  6
## 8 Mark3 Kim Seok Jin  2.1
## 9 Mark3 Park Ji Min  4.6
## 10 Mark4 Kim Nam Joon 2.7
## ... with 11 more rows
```

This should be easy with Tidy Data

```
# {r, fig.height=1.5, fig.width=5.5}  
tidy_marks %>% ggplot(aes(x=Mark,y=Students)) + geom_boxplot()
```

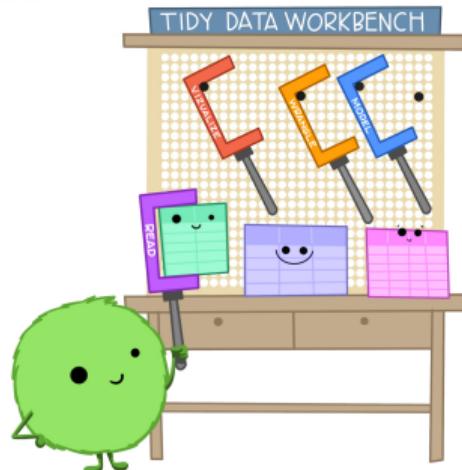


- Comparisons within columns should always be sensible
- Comparisons across columns should be less meaningful

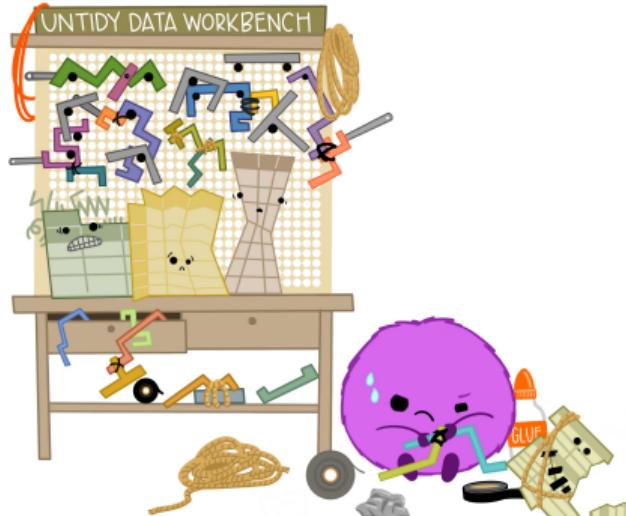
“One Size Fits All” when you have Tidy Data

Efficiency Increases when you have a Standard Reliable Tool Set

When working with tidy data,
we can use the same tools in
similar ways for different datasets...



...but working with untidy data often means
reinventing the wheel with one-time
approaches that are hard to iterate or reuse.



@Allison_Horst

Quiz 2 on Tidy Data

Which of these is “Tidy”?

```
## # A tibble: 12 x 4
##   country     year   type     count
##   <chr>       <int> <chr>    <int>
## 1 Afghanistan 1999  cases      745
## 2 Afghanistan 1999  population 19987071
## 3 Afghanistan 2000  cases      2666
## 4 Afghanistan 2000  population 20595360
## 5 Brazil       1999  cases      37737
## 6 Brazil       1999  population 172006362
## 7 Brazil       2000  cases      80488
## 8 Brazil       2000  population 174504898
## 9 China        1999  cases      212258
## 10 China       1999  population 1272915272
## 11 China       2000  cases      213766
## 12 China       2000  population 1280428583
```

```
## # A tibble: 6 x 4
##   country     year   cases population
##   <chr>       <int> <int>      <int>
## 1 Afghanistan 1999      745  19987071
## 2 Afghanistan 2000     2666  20595360
## 3 Brazil       1999    37737 172006362
## 4 Brazil       2000    80488 174504898
## 5 China        1999   212258 1272915272
## 6 China        2000   213766 1280428583

## # A tibble: 6 x 3
##   country     year   rate
##   <chr>       <int> <chr>
## 1 Afghanistan 1999 745/19987071
## 2 Afghanistan 2000 2666/20595360
## 3 Brazil       1999 37737/172006362
## 4 Brazil       2000 80488/174504898
## 5 China        1999 212258/1272915272
## 6 China        2000 213766/1280428583
```

Tidying and making data useful by Data Wrangling

- “80% of data analysis is spent on the process of cleaning and preparing data”
- “Write code for humans, but data for computers”



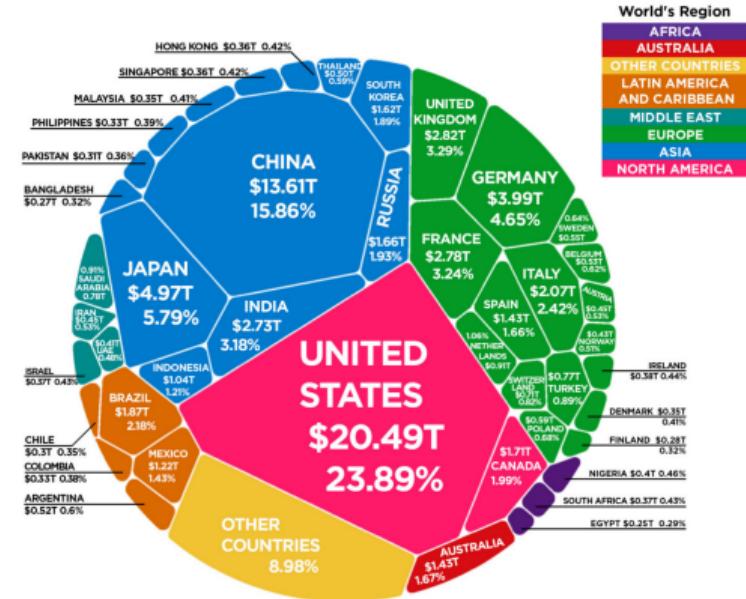
- `dplyr` is designed on a structured and very general grammar (like `ggplot2`)

The `dplyr` and `tidyr` libraries are part of the `tidyverse`. Introductions to these libraries are respectively given in the [Transformation](#) and [Tidy Data](#) chapters of the [R4DS](#) textbook, in the official `dplyr` and `tidyr` cheat sheets, and in the UoFT [DoSS Toolkit](#).

Some Data for Data Wrangling



+



Article & Sources:

<https://howmuch.net/articles/the-world-economy-2018>
<https://databank.worldbank.org>

Some Data for Data Wrangling

Created by previous STA130 profs. based on [VGAMdata](#)

```
#install.packages("VGAMdata")
library(VGAMdata)
help(oly12)
```

by adding in ISO, GDP, and pop

Variable	Description
Country	Name of the country
athletes_f	# of female athletes
athletes_m	# of male athletes
athletes_total	# of athletes
G/S/B	# of G/S/B medals won

Variable	Description
ISO	International Standard Code of 2 or 3 characters uniquely identifying each country
GDP.2011	Gross domestic product in 2011
pop.2010	Population in 2010

- A similar data set exists on [kaggle](#)

G=gold; S=silver; B=bronze

Some Data for Data Wrangling

```
#install.packages("tidyverse") # preinstalled on jupyterhub so not required
library(tidyverse) # autoloads `readr`, `dplyr`, `ggplot2`, `tidyrr`, etc.
olympics <- read_csv("oly12countries.csv")
glimpse(olympics)
```

```
## Rows: 204
## Columns: 10
## $ Country           <chr> "Afghanistan", "Albania", "Algeria", "American Samoa", ~
## $ ISO                <chr> "AFG", "ALB", "DZA", "ASM", "AND", "AGO", "ATG", "ARG", ~
## $ GDP.2011            <dbl> 2.034346e+10, 1.295956e+10, 1.886810e+11, 5.370000e+08, ~
## $ pop.2010             <dbl> 34385000, 3205000, 35468000, 68420, 84864, 19082000, 88~
## $ athletes_f            <dbl> 1, 4, 18, 1, 2, 30, 2, 43, 4, 1, 188, 31, 14, 11, 8, 1, ~
## $ athletes_m            <dbl> 5, 7, 21, 4, 4, 5, 3, 99, 21, 3, 225, 39, 39, 15, 4, 4, ~
## $ athletes_total          <dbl> 6, 11, 39, 5, 6, 35, 5, 142, 25, 4, 413, 70, 53, 26, 12~
## $ gold                 <dbl> 0, 0, 1, 0, 0, 0, 1, 0, 0, 7, 0, 2, 1, 0, 0, 0, 2, 0~
## $ silver                <dbl> 0, 0, 0, 0, 0, 0, 1, 1, 0, 16, 0, 2, 0, 0, 0, 0, 5, ~
## $ bronze                <dbl> 1, 0, 0, 0, 0, 0, 2, 2, 0, 12, 0, 6, 0, 1, 0, 0, 5, ~
```

select() Data Wrangling

```
dplyr::select()
```

Extract a subset of VARIABLES

```
olympics %>% select(Country, athletes_total, gold, silver, bronze) %>%
  print(n=6) # What is `print(n=6)` compared to `head()`?
```

```
## # A tibble: 204 x 5
##   Country      athletes_total   gold   silver   bronze
##   <chr>          <dbl>     <dbl>    <dbl>    <dbl>
## 1 Afghanistan       6         0        0        1
## 2 Albania           11        0        0        0
## 3 Algeria            39        1        0        0
## 4 American Samoa     5         0        0        0
## 5 Andorra            6         0        0        0
## 6 Angola             35        0        0        0
## # ... with 198 more rows
```

select() Data Wrangling

```
dplyr::select()
```

Extract a subset of VARIABLES / Remove specific VARIABLES

```
olympics %>% select(-ISO,-GDP.2011,-pop.2010,-athletes_f,-athletes_m) %>%
  print(n=6) # https://tibble.tidyverse.org/reference/formatting.html
```

```
## # A tibble: 204 x 5
##   Country     athletes_total   gold   silver bronze
##   <chr>          <dbl>    <dbl>    <dbl>    <dbl>
## 1 Afghanistan      6        0        0        1
## 2 Albania           11       0        0        0
## 3 Algeria            39       1        0        0
## 4 American Samoa     5        0        0        0
## 5 Andorra            6        0        0        0
## 6 Angola             35       0        0        0
## # ... with 198 more rows
```

select() Data Wrangling

dplyr::select() [and see also dplyr::rename()]

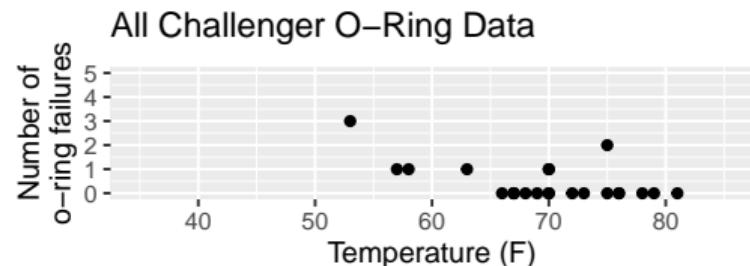
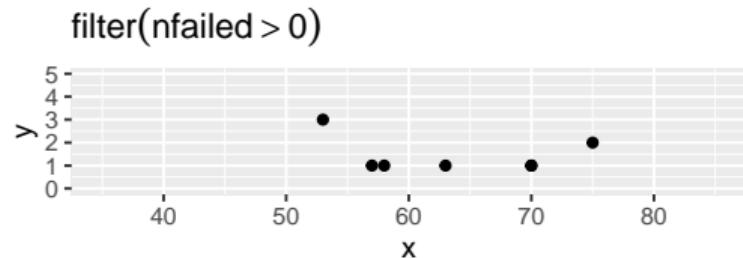
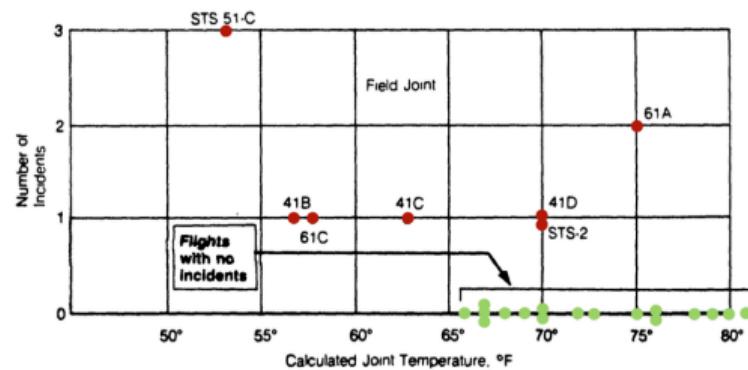
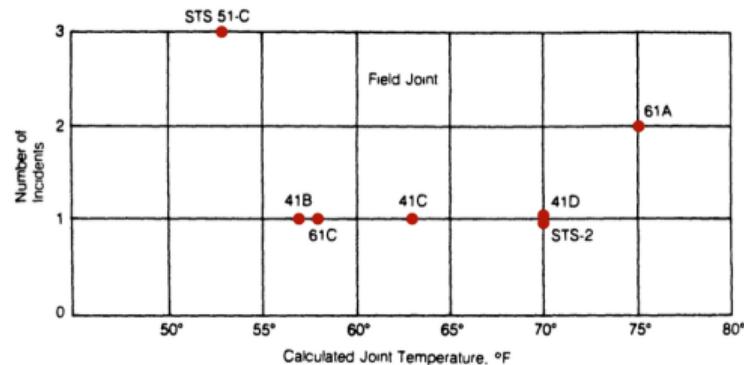
Extract a subset of VARIABLES / Remove specific VARIABLES / Rename VARIABLES

```
olympics %>% select(Country, `# of athletes`=athletes_total,
                       `#G`=gold, Num_S=silver, NumB=bronze) %>% print(n=6)

## # A tibble: 204 x 5
##   Country      `# of athletes` `#G`  Num_S  NumB
##   <chr>          <dbl> <dbl> <dbl> <dbl>
## 1 Afghanistan     6     0     0     1
## 2 Albania         11    0     0     0
## 3 Algeria         39     1     0     0
## 4 American Samoa  5     0     0     0
## 5 Andorra          6     0     0     0
## 6 Angola           35    0     0     0
## # ... with 198 more rows
```

With great power comes great responsibility

The Challenger Disaster was caused by a bad filter() command...



filter() Data Wrangling

```
dplyr::filter()
```

Extract OBSERVATION subsets based on conditions on values in one or more columns

```
olympics %>% filter(athletes_total > 300 | gold > 5) %>% print(n=7)
```

```
## # A tibble: 15 x 10
##   Country   ISO    GDP.2011 pop.2010 athletes_f athletes_m athletes_total gold
##   <chr>     <chr>    <dbl>      <dbl>      <dbl>      <dbl>      <dbl> <dbl>
## 1 Australia AUS    1.37e12   2.23e7     188       225       413      7
## 2 China     CHN    7.30e12   1.34e9     208       163       371      38
## 3 France    FRA    2.77e12   6.49e7     148       187       335      11
## 4 Germany   DEU    3.57e12   8.18e7     176       219       395      11
## 5 Hungary   HUN    1.40e11   1e+07      63        95       158      8
## 6 Italy     ITA    2.19e12   6.05e7     122       159       281      8
## 7 Japan     JPN    5.87e12   1.27e8     162       141       303      7
## # ... with 8 more rows, and 2 more variables: silver <dbl>, bronze <dbl>
```

select() + filter() Data Wrangling

dplyr::filter() and dplyr::select()

Extract DATA subsets of interest / Remove DATA of no interest

```
olympics %>% filter(athletes_total > 200 & gold < 10) %>%  
  select(Country, athletes_total, gold, silver, bronze) %>% print(n=6)  
  
## # A tibble: 8 x 5  
##   Country    athletes_total   gold   silver   bronze  
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>  
## 1 Australia      413        7       16       12  
## 2 Brazil         266        3        5        9  
## 3 Canada         279        1        5       12  
## 4 Italy          281        8        9       11  
## 5 Japan          303        7       14       17  
## 6 Poland         217        2        2        6  
## # ... with 2 more rows
```

Quiz on `select()` + `filter()` Data Wrangling

```
olympics %>% filter(athletes_total < 100 & gold > 1) %>%  
  select(Country, athletes_total, gold, silver, bronze)
```

Options

- A. All countries in the original tibble with fewer than 100 athletes and all variables from the original tibble
- B. All countries in the original tibble and five variables: Country, # of athletes, and # of gold/silver/bronze medals
- C. All countries in the original tibble which won more than one gold medal and had fewer than 100 athletes and five variables: Country, # of athletes, and # of gold/silver/bronze medals
- D. All countries in the original tibble with fewer than 100 athletes or who won more than one gold medal and five variables: Country, # of athletes, and # of gold/silver/bronze medals

arrange() + select() + filter() Data Wrangling

```
dplyr::arrange() and dplyr::arrange(dplyr::desc())
```

Sort rows/observations based on the values in one or more columns/variables

```
olympics %>%
  arrange(desc(athletes_total), gold) %>%
  filter(athletes_total<100 & gold>1) %>%
  select(Country, athletes_total, gold)

## # A tibble: 8 x 3
##   Country    athletes_total  gold
##   <chr>          <dbl>   <dbl>
## 1 Norway           65      2
## 2 Lithuania         62      2
## 3 North Korea       55      4
## 4 Azerbaijan        53      2
## 5 Iran              53      4
## 6 Kenya             50      2
## 7 Jamaica            50      4
## 8 Ethiopia           35      3
```

- What does 'desc()' do?
- Is the following "*order of operations*" more efficient?

- ① select()
- ② filter()
- ③ arrange()

```
olympics %>%
  select(Country,athletes_total,gold) %>%
  filter(athletes_total<100 & gold>1) %>%
  arrange(desc(athletes_total), gold)
```

Review Quiz on Data Wrangling

Do the outputs match the code?

Check the `select()`, `arrange()`, `desc()`, and `head()/print()` functions & input

```
olympics %>% arrange(tot_medals) %>%  
  select(tot_medals, avg_medals,  
         Country, ISO) %>% head()  
  
## # A tibble: 204 x 3  
##   tot_medals avg_medals Country  
##       <dbl>      <dbl> <chr>  
## 1        104      0.196 US  
## 2        88       0.237 China  
## 3        82       0.189 Russia  
## 4        65       0.117 UK  
## 5        44       0.111 Germany  
## # ... with 199 more rows
```

```
olynew %>% arrange(desc(avg_medals)) %>%  
  select(tot_medals, avg_medals,  
         Country, ISO) %>% head()  
  
## # A tibble: 6 x 4  
##   tot_medals avg_medals Country ISO  
##       <dbl>      <dbl> <chr> <chr>  
## 1        1       0.25 Botswana BWA  
## 2        12      0.24 Jamaica JAM  
## 3        88      0.237 China CHN  
## 4        12      0.226 Iran IRN  
## 5        11      0.22 Kenya KEN  
## 6         7      0.2   Ethiopia ETH
```

Some errors during Data Wrangling

```
olympics %>% filter(tot_medals > 1 & avg_medals > 0.1)

## Error in `filter()`:
## ! Problem while computing `...1 = tot_medals > 1 & avg_medals > 0.1` .
## Caused by error:
## ! object 'tot_medals' not found
```

Errors messages in RStudio are often (but not always) informative about what's gone wrong; *but, you have to read error output text to find the explanation.*

```
olympics %>% select(tot_medals, avg_medals)

## Error in `select()`:
## ! Can't subset columns that don't exist.
## x Column `tot_medals` doesn't exist.
```

mutate() Data Wrangling

```
dplyr::mutate()
```

Defining and creating new useful and interesting variables

```
olympics %>% select(Country, athletes_total, gold, silver, bronze) %>%
  mutate(tot_medals = gold+silver+bronze,
        avg_medals = tot_medals/athletes_total) -> olynew
olynew %>% print(n=4)

## # A tibble: 204 x 8
##   Country      ISO  athletes_total  gold  silver bronze tot_medals avg_medals
##   <chr>       <chr>        <dbl> <dbl>  <dbl>  <dbl>      <dbl>      <dbl>
## 1 Afghanistan AFG          6     0     0     1         1     0.167
## 2 Albania      ALB         11     0     0     0         0     0
## 3 Algeria      DZA         39     1     0     0         1     0.0256
## 4 American Samoa ASM         5     0     0     0         0     0
## # ... with 200 more rows
```

case_when() + mutate() Data Wrangling

```
dplyr::mutate() and dplyr::case_when()
```

```
olympics %>% mutate(majority = case_when(
    athletes_f > athletes_m ~ "Female",
    athletes_f < athletes_m ~ "Male",
    athletes_f == athletes_m ~ "Balanced"),
    total_medals = gold+silver+bronze) -> olynew
olynew %>% select(Country, athletes_total, athletes_f,
    athletes_m, majority, total_medals) %>% print(n=4)

## # A tibble: 204 x 6
##   Country      athletes_total athletes_f athletes_m majority total_medals
##   <chr>          <dbl>        <dbl>        <dbl> <chr>        <dbl>
## 1 Afghanistan     6            1            5 Male           1
## 2 Albania         11           4            7 Male           0
## 3 Algeria         39           18           21 Male          1
## 4 American Samoa  5            1            4 Male           0
## # ... with 200 more rows
```

summarise() for Summary Tables

dplyr::summarise() and Aggregation Functions

```
olympics %>% summarise(n=n(), sum_gold=sum(gold), median_gold=median(gold),
                           mean_gold=mean(gold), var_gold=var(gold), sd_gold=sd(gold))

## # A tibble: 1 x 6
##       n   sum_gold median_gold  mean_gold  var_gold  sd_gold
##   <int>     <dbl>        <dbl>      <dbl>     <dbl>     <dbl>
## 1    204      302          0       1.48     27.2      5.21

olympics %>% summarize(n=n(), sumGold=sum(gold), minGold=min(gold), maxGold=max(gold),
                        `25th%tile` = quantile(gold, 0.25), IQRgold=IQR(gold),
                        `75th%tile` = quantile(gold, 0.75), `90th%tile` = quantile(gold, 0.90))

## # A tibble: 1 x 8
##       n   sumGold minGold maxGold `25th%tile` IQRgold `75th%tile` `90th%tile`
##   <int>     <dbl>    <dbl>    <dbl>      <dbl>     <dbl>      <dbl>     <dbl>
## 1    204      302      0       46         0        1         1        3
```

group_by() + summarise() for Summary Tables

dplyr::summarise() and Aggregation Functions and dplyr::group_by()

Creating more complex summary tables with multiple rows

```
olympics %>% mutate(teamsize = case_when(
  athletes_total >= 100 ~ "big",
  (athletes_total < 100) & (athletes_total >= 20) ~ "medium",
  athletes_total < 20 ~ "small")) %>%
  group_by(teamsize) %>%
  summarize(n=n(), `Total Gold`=sum(gold), Average_Gold=mean(gold),
            MedianGold=median(gold), maxGold=max(gold))

## # A tibble: 3 x 6
##   teamsize     n `Total Gold` Average_Gold MedianGold maxGold
##   <chr>     <int>       <dbl>        <dbl>      <dbl>    <dbl>
## 1 big         36        266        7.39        3        46
## 2 medium      51        34        0.667       0        4
## 3 small      117        2        0.0171       0        1
```

group_by() + summarise() for Summary Tables

dplyr::summarise() and Aggregation Functions and dplyr::group_by()

Creating more complex Aggregation Functions for multiple row summary tables

```
olynew %>% group_by(majority) %>%  
  summarize(n=n(), `Average Medals`=mean(gold+silver+bronze),  
            minMedals=min(gold+silver+bronze), max_Medals=max(gold+silver+bronze))  
  
## # A tibble: 4 x 5  
##   majority     n `Average Medals`  minMedals max_Medals  
##   <chr>     <int>          <dbl>        <dbl>        <dbl>  
## 1 Balanced     25           0.6         0          12  
## 2 Female       34          10.8        0         104  
## 3 Male        144          4.02        0          65  
## 4 <NA>         1            0          0          0
```

Question: Why are there 4 groups above? What's going on with the <NA> thing?

is.na() for Handling NA Missing Data in R

```
vector <- c(1,2,NA,4,NA); is.na(vector)
## [1] FALSE FALSE  TRUE FALSE  TRUE
olynew %>% filter(is.na(majority))

## # A tibble: 1 x 8
##   Country athletes_f athletes_m athletes_total majority gold silver bronze
##   <chr>      <dbl>      <dbl>        <dbl> <chr>    <dbl>  <dbl>  <dbl>
## 1 Barbados       NA          6           6 <NA>      0     0     0
olynew %>% filter(!is.na(majority)) %>% group_by(majority) %>%
  summarize(n=n(), `Average Medals`=mean(gold+silver+bronze),
            minMedals=min(gold+silver+bronze), max_Medals=max(gold+silver+bronze))

## # A tibble: 3 x 5
##   majority      n `Average Medals` minMedals max_Medals
##   <chr>      <int>        <dbl>      <dbl>        <dbl>
## 1 Balanced      25         0.6        0        12
## 2 Female        34        10.8        0       104
## 3 Male         144        4.02       0        65
```

na.rm() for Handling NA Missing Data in R

You can't calculate with NA *but you can remove and ignore NA with na.rm()*

```
vector <- c(1,2,NA,4,NA); mean(vector)
## [1] NA
mean(vector, na.rm=TRUE)
## [1] 2.333333
```

Be careful with the n() function

- It doesn't know about missing data NA so it can misrepresent data...

```
olynew %>% summarize(n_NumberOfRows=n(), n_NumberOfMeasurements=sum(!is.na(majority)),
                      mean=mean(athletes_f), mean_na_rm=mean(athletes_f, na.rm=TRUE))

## # A tibble: 1 x 4
##   n_NumberOfRows n_NumberOfMeasurements   mean mean_na_rm
##             <int>                  <int>  <dbl>        <dbl>
## 1              204                      203     NA       23.7
```

Class Check Round 2

<https://pollev.com/sta> (8 questions)

Self Quiz

- Can you explain the benefit of having tidy data?
- Can you explain why a data set is not tidy?
- Can you describe the new tibble produced from a sequence of dplyr functions applied to a given tibble?
- Can you use dplyr filter() and select() functions to extract a subset of a larger data set?
- Can you list 10 *aggregation functions* that are used with dplyr summarise() and group_by functions?
- Can you use dplyr summarise() and group_by along with *aggregation functions* to answer questions with summary tables?

Class Check Round 3

<https://pollev.com/sta> (2 questions)

Rstudio Demo

- ① Click this [jupyterhub](#) repo launcher link