Advanced Quantitative Methods

R Basics 2

Instructor: Gregory Eady
Office: 18.2.10
Office hours: Fridays 13-15

Today

- Introduction to `tidyverse` by Hadley Wickham
- Data processing with `tidyverse`

Recall installing and loading libraries in R

```
# Install the tidyverse package
install.packages("tidyverse")
library(tidyverse)
```

Creating vectors in R

```
# Create a variable of random words
var_words <- c("Apple", "Orange", "Pear", "Cherry")
var_words
class(var_words)

# Create a variable of random numbers
var_num <- c(123, 91, -50, 15200, -1, 0, 0)
var_num
class(var_num)
```

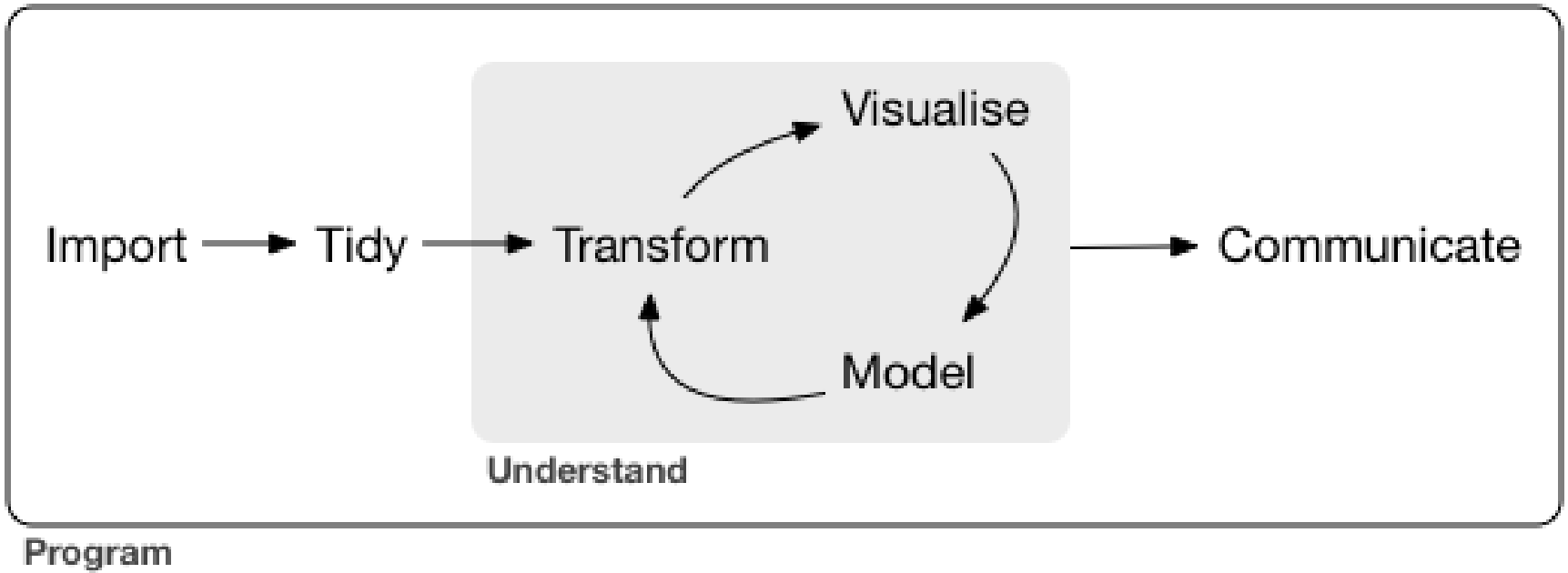
Creating vectors in R

```
# Create a variable of numbers and words?
var_num_words <- c(123, 91, -50, "Apple", 124)
var_num_words
class(var_num_words)
```

Creating a data.frame in R

```
# Create some variables
fruits <- c("Apple", "Peach", "Pear", "Lemon", "Pineapple")
prices <- c(123, 91, 50, 41, 124)

# Put the variables in a data.frame
D <- data.frame(fruits, prices)

# Look at the data.frame we created
D

# Look at one of the variables
D$fruits

# Add a new variable to our existing data.frame
D$store <- c("Netto", "Netto", "Tesco", "Aldi", "Tesco")

# The frequency of the store variable
table(D$store)
```

Tidyverse workflow



Messy data

Name	Publisher	Sales (mil. units)	Critic	User
Wii Sports	Nintendo	82.53	Score = 76, Count = 51	Score = 8, Count = 322
Mario Kart Wii	Nintendo	35.52	Score = 82, Count = 73	Score = 8.3, Count = 709
Wii Sports Resort	Nintendo	32.77	Score = 80, Count = 73	Score = 8, Count = 192
New Super Mario Bros.	Nintendo	29.80	Score = 89, Count = 65	Score = 8.5, Count = 431
Wii Play	Nintendo	28.92	Score = 58, Count = 41	Score = 6.6, Count = 129

Source: https://rss.onlinelibrary.wiley.com/doi/full/10.1111/j.1740-9713.2018.01169.x

Tidy data

name	publisher	sales	score	
Wii Sports	Nintendo	82.53	Critic	76
Wii Sports	Nintendo	82.53	User	80
Mario Kart Wii	Nintendo	35.52	Critic	82
Mario Kart Wii	Nintendo	35.52	User	83
Wii Sports Resort	Nintendo	32.77	Critic	80
Wii Sports Resort	Nintendo	32.77	User	80
New Super Mario Bros.	Nintendo	29.80	Critic	89
New Super Mario Bros.	Nintendo	29.80	User	85
Wii Play	Nintendo	28.92	Critic	58
Wii Play	Nintendo	28.92	User	66

How do we get from messy to tidy data?

Action	Function in `dplyr`
Filter	`filter()`
Aggregate	`group_by()` and `summarize()`
Sort	`arrange()`
Reshape	`pivot_longer()` and `pivot_wider()`
Recode	`mutate()` and `recode()`

Data Wrangling with dplyr and tidyr

Cheat Sheet

RStudio

Tidy Data - A foundation for wrangling in R

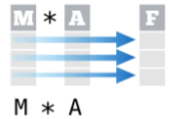
In a tidy data set:

Each **variable** is saved in its own **column**

&

Each **observation** is saved in its own **row**

Tidy data complements R's **vectorized operations**. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.



Syntax - Helpful conventions for wrangling

dplyr::**tbl_df(iris)**

Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen:

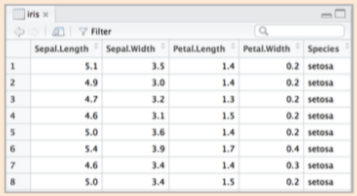
```
Source: local data frame [150 x 5]

   Sepal.Length Sepal.Width Petal.Length
1           5.1         3.5          1.4
2           4.9         3.0          1.4
3           4.7         3.2          1.3
4           4.6         3.1          1.5
5           5.0         3.6          1.4
..          ...         ...          ...
Variables not shown: Petal.Width (dbl),
  Species (fctr)
```

dplyr::**glimpse(iris)**

Information dense summary of tbl data.

utils::**View(iris)**

View data set in spreadsheet-like display (note capital V).



	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

dplyr::**%>%**

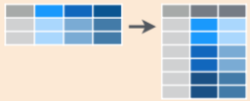
Passes object on left hand side as first argument (or . argument) of function on righthand side.

```
x %>% f(y)  is the same as  f(x, y)
y %>% f(x, .., z)  is the same as  f(x, y, z )
```

"Piping" with %>% makes code more readable, e.g.

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```

Reshaping Data - Change the layout of a data set



tidyr::**gather(cases, "year", "n", 2:4)**

Gather columns into rows.

tidyr::**spread(pollution, size, amount)**

Spread rows into columns.

tidyr::**separate(storms, date, c("y", "m", "d"))**

Separate one column into several.

tidyr::**unite(data, col, ..., sep)**

Unite several columns into one.

dplyr::**data_frame(a = 1:3, b = 4:6)**

Combine vectors into data frame (optimized).

dplyr::**arrange(mtcars, mpg)**

Order rows by values of a column (low to high).

dplyr::**arrange(mtcars, desc(mpg))**

Order rows by values of a column (high to low).

dplyr::**rename(tb, y = year)**

Rename the columns of a data frame.

Subset Observations (Rows)

dplyr::**filter(iris, Sepal.Length > 7)**

Extract rows that meet logical criteria.

dplyr::**distinct(iris)**

Remove duplicate rows.

dplyr::**sample_frac(iris, 0.5, replace = TRUE)**

Randomly select fraction of rows.

dplyr::**sample_n(iris, 10, replace = TRUE)**

Randomly select n rows.

dplyr::**slice(iris, 10:15)**

Select rows by position.

dplyr::**top_n(storms, 2, date)**

Select and order top n entries (by group if grouped data).

Logic in R - ?Comparison, ?base::Logic

<	Less than	!=	Not equal to
>	Greater than	%in%	Group membership
==	Equal to	is.na	Is NA
<=	Less than or equal to	!is.na	Is not NA
>=	Greater than or equal to	&, |, !, xor, any, all	Boolean operators

Subset Variables (Columns)

dplyr::**select(iris, Sepal.Width, Petal.Length, Species)**

Select columns by name or helper function.

Helper functions for select - ?select

select(iris, contains("."))
Select columns whose name contains a character string.

select(iris, ends_with("Length"))
Select columns whose name ends with a character string.

select(iris, everything())
Select every column.

select(iris, matches(".t."))
Select columns whose name matches a regular expression.

select(iris, num_range("x", 1:5))
Select columns named x1, x2, x3, x4, x5.

select(iris, one_of(c("Species", "Genus")))
Select columns whose names are in a group of names.

select(iris, starts_with("Sepal"))
Select columns whose name starts with a character string.

select(iris, Sepal.Length:Petal.Width)
Select all columns between Sepal.Length and Petal.Width (inclusive).

select(iris, -Species)
Select all columns except Species.

devtools::install_github("rstudio/EDAWR") for data sets

Learn more with browseVignettes(package = c("dplyr", "tidyr")) • dplyr 0.4.0• tidyr 0.2.0 • Updated: 1/15

Summarise Data

dplyr::**summarise(iris, avg = mean(Sepal.Length))**
Summarise data into single row of values.
dplyr::**summarise_each(iris, funs(mean))**
Apply summary function to each column.
dplyr::**count(iris, Species, wt = Sepal.Length)**
Count number of rows with each unique value of variable (with or without weights).

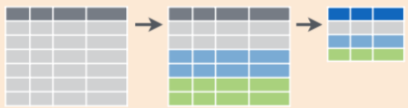
Summarise uses **summary functions**, functions that take a vector of values and return a single value, such as:

dplyr::**first**
First value of a vector.
dplyr::**last**
Last value of a vector.
dplyr::**nth**
Nth value of a vector.
dplyr::**n**
# of values in a vector.
dplyr::**n_distinct**
# of distinct values in a vector.
IQR
IQR of a vector.
min
Minimum value in a vector.
max
Maximum value in a vector.
mean
Mean value of a vector.
median
Median value of a vector.
var
Variance of a vector.
sd
Standard deviation of a vector.

Group Data

dplyr::**group_by(iris, Species)**
Group data into rows with the same value of Species.
dplyr::**ungroup(iris)**
Remove grouping information from data frame.

iris %>% group_by(Species) %>% summarise(...)
Compute separate summary row for each group.



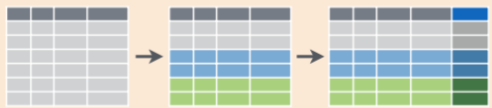
Make New Variables

dplyr::**mutate(iris, sepal = Sepal.Length + Sepal. Width)**
Compute and append one or more new columns.
dplyr::**mutate_each(iris, funs(min_rank))**
Apply window function to each column.
dplyr::**transmute(iris, sepal = Sepal.Length + Sepal. Width)**
Compute one or more new columns. Drop original columns.

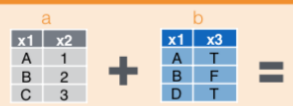
Mutate uses **window functions**, functions that take a vector of values and return another vector of values, such as:

dplyr::**lead**
Copy with values shifted by 1.
dplyr::**lag**
Copy with values lagged by 1.
dplyr::**dense_rank**
Ranks with no gaps.
dplyr::**min_rank**
Ranks. Ties get min rank.
dplyr::**percent_rank**
Ranks rescaled to [0, 1].
dplyr::**row_number**
Ranks. Ties got to first value.
dplyr::**ntile**
Bin vector into n buckets.
dplyr::**between**
Are values between a and b?
dplyr::**cume_dist**
Cumulative distribution.
dplyr::**cumall**
Cumulative `all`
dplyr::**cumany**
Cumulative `any`
dplyr::**cummean**
Cumulative `mean`
cumsum
Cumulative `sum`
cummax
Cumulative `max`
cummin
Cumulative `min`
cumprod
Cumulative `prod`
pmax
Element-wise `max`
pmin
Element-wise `min`

iris %>% group_by(Species) %>% mutate(...)
Compute new variables by group.



Combine Data Sets



Mutating Joins

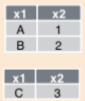
dplyr::**left_join(a, b, by = "x1")**
Join matching rows from b to a.

dplyr::**right_join(a, b, by = "x1")**
Join matching rows from a to b.

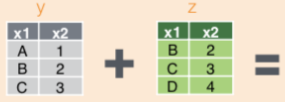
dplyr::**inner_join(a, b, by = "x1")**
Join data. Retain only rows in both sets.

dplyr::**full_join(a, b, by = "x1")**
Join data. Retain all values, all rows.

Filtering Joins



dplyr::**semi_join(a, b, by = "x1")**
All rows in a that have a match in b.
dplyr::**anti_join(a, b, by = "x1")**
All rows in a that do not have a match in b.



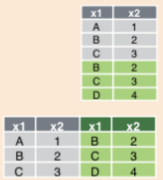
Set Operations

dplyr::**intersect(y, z)**
Rows that appear in both y and z.

dplyr::**union(y, z)**
Rows that appear in either or both y and z.

dplyr::**setdiff(y, z)**
Rows that appear in y but not z.

Binding



dplyr::**bind_rows(y, z)**
Append z to y as new rows.
dplyr::**bind_cols(y, z)**
Append z to y as new columns.
Caution: matches rows by position.

devtools::install_github("rstudio/EDAWR") for data sets
Learn more with browseVignettes(package = c("dplyr", "tidyr")) • dplyr 0.4.0 • tidyr 0.2.0 • Updated: 1/15

Download the following file:

`GM.csv` from the **2. R Basics II** page of the **Weekly Readings** on the course website.

Gapminder data

Loading data in R

```
# Set the working directory where the file is
setwd("~/Downloads/")

# Load GapMinder data
GM <- read_csv("GM.csv")
```

Gapminder variables

Variable	Description
`country`	Country
`continent`	Continent that country is located on
`year`	Year
`lifeExp`	Life expectancy at birth
`pop`	Total population size
`gdpPercap`	GDP (per capita)

```
> # Look at the loaded data
> GM
# A tibble: 1,704 x 6
   country     continent  year lifeExp      pop gdpPercap
   <chr>       <chr>     <dbl>   <dbl>    <dbl>     <dbl>
 1 Afghanistan Asia       1952    28.8  8425333       779.
 2 Afghanistan Asia       1957    30.3  9240934       821.
 3 Afghanistan Asia       1962    32.0 10267083       853.
 4 Afghanistan Asia       1967    34.0 11537966       836.
 5 Afghanistan Asia       1972    36.1 13079460       740.
 6 Afghanistan Asia       1977    38.4 14880372       786.
 7 Afghanistan Asia       1982    39.9 12881816       978.
 8 Afghanistan Asia       1987    40.8 13867957       852.
 9 Afghanistan Asia       1992    41.7 16317921       649.
10 Afghanistan Asia       1997    41.8 22227415       635.
# ... with 1,694 more rows
```

```
# Look at the first (6) rows
head(GM)

# Look at the last (6) rows
tail(GM)

# Show names of variables in the data
names(GM)

# Descriptive statistics of all variables
summary(GM)
```

Tidyverse

```
# Look at a single variable
GM$gdpPercap

# Look at specific values of a single variable
GM$gdpPercap[50:60]

# Look at values conditional on other values/variables
GM$gdpPercap[GM$country == "Denmark"]

# Look at all rows conditional on other values/variables
GM$gdpPercap[GM$country == "Denmark", ]
```

Let's say we want to do the following

1. Group the observations by country
2. Calculate average life expectancy and median population over each period for each country
3. Filter our data for countries with more than 10 mil. inhabitants
4. Sort reverse alphabetically by country

Solution 1

```
##### Solution 1 (solution: break task into pieces)

# Split data by country
By_country <- group_by(GM, country)

# Calculate avg. life expectancy and median pop. per country
Country_sum <- summarize(By_country,
                         avg_life_exp = mean(lifeExp),
                         median_pop = median(pop))

# Subset by countries with higher than 10 million inhabitants
Big_countries <- filter(Country_sum, median_pop > 10000000)

# Sort the data by country and life expectancy
Result <- arrange(Big_countries, desc(country))
```

Solution 2

```
##### Solution 2 (solution: nested functions)
Result_2 <- arrange(
              filter(
                summarize(group_by(GM, country),
                  avg_life_exp = mean(lifeExp),
                  median_pop = median(pop)),
                median_pop > 10000000),
              desc(country))
```

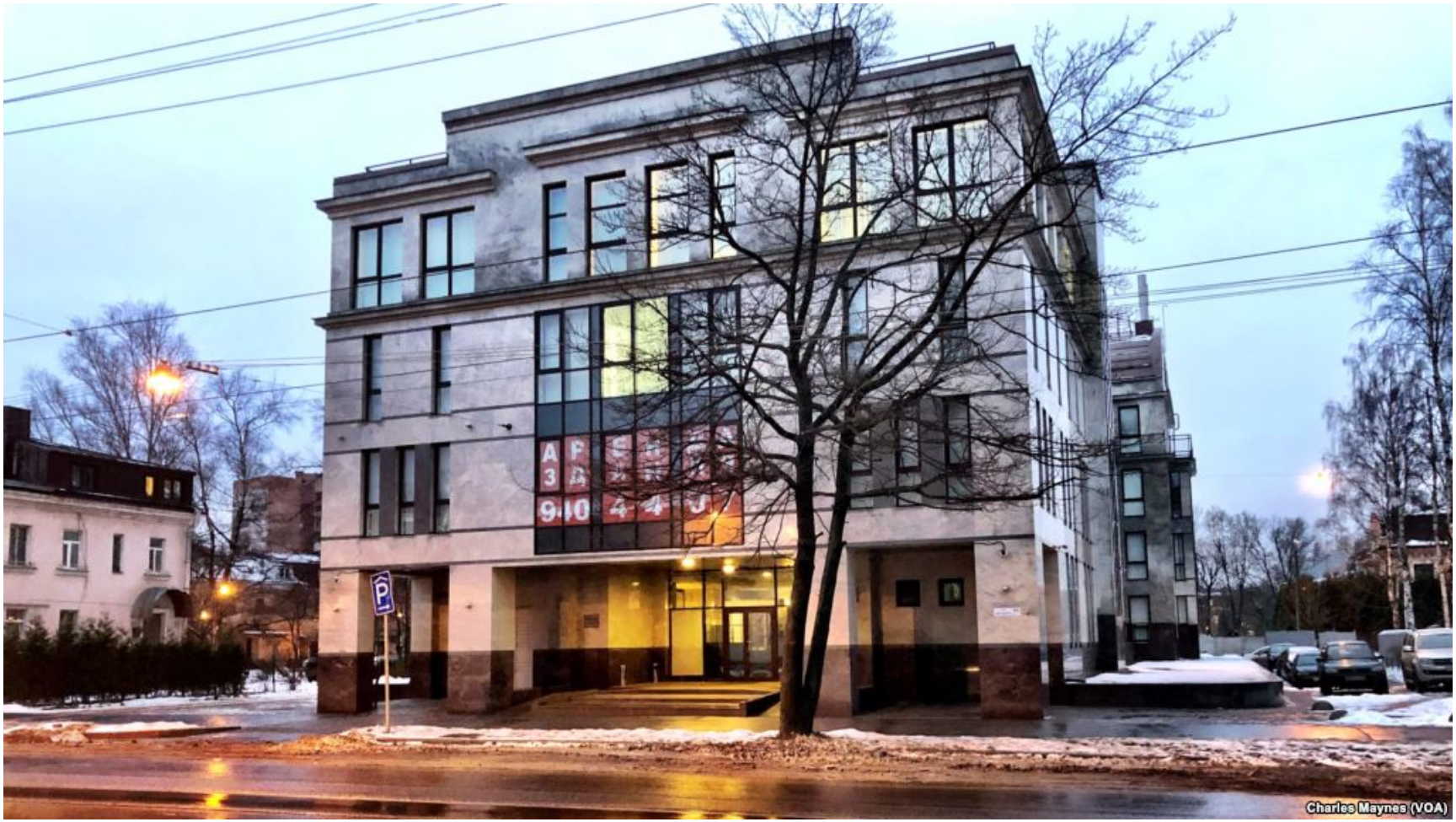
Solution 3

```
##### Solution 3 (solution: tidyverse)
Result_3 <- GM %>%
            group_by(country) %>%
            summarize(avg_life_exp = mean(lifeExp),
                      median_pop = median(pop)) %>%
            filter(median_pop > 10000000) %>%
            arrange(desc(country))
```

Change a variable within a data.frame

```
##### Create variable as the log of population
Result_3 <- Result_3 %>%
            mutate(log_median_pop = log(median_pop))

##### Or similarly without mutate()
Result_3$log_median_pop <- log(Result_3$median_pop)
```

Russia's Internet Research Agency



3 million Russian disinformation campaign tweets

This directory contains data on nearly 3 million tweets sent from Twitter accounts connected to the Internet Research Agency: the Russian "troll factory" that interfered in the 2016 US election. The tweets in this database were sent between February 2012 and May 2018, with the vast majority posted from 2015 through 2017.

Source: `https://github.com/fivethirtyeight/russian-troll-tweets`

Exercise

1. Download & load a Russian troll dataset from `https://github.com/fivethirtyeight/russian-troll-tweets`
2. What are the most frequent and second-most frequent languages?
    - `n()`
3. Which region's tweets were received by the most followers?
4. On average, how many followers did each tweet reach in each region?
5. How many tweets are retweets in each language?
6. How many tweets are *not* retweets in each language?
7. How often are "Trump" and "Clinton" mentioned in the tweets?
    - `str_count()`, `tolower()`

Download one of the datasets from:

https://gapminder.org/data/

```
library(tidyverse)
library(readxl) # If you download an (Excel) xlsx file

# Set the working directory
setwd(~/Downloads/)

# Load GapMinder data on AIDS prevalence
A <- read_excel("sh_dyn_aids_zs.xlsx")

# Look at the names of the variables in the dataset
names(A)

# Look at the first few rows of the data
head(A)
```

```
A <- A %>%
    pivot_longer("1990":"2018",
        names_to = "year",
        values_to = "hiv_prevalence") %>%
    arrange(country, year)

A <- A %>%
     mutate(year = as.numeric(year),
            hiv_prevalence = as.numeric(hiv_prevalence),
            # Or can recode a variable as follows
            country = recode(country, "Afghanistan" = "AFGH",
                                      "Denmark" = "DK"))

# Calculate the mean level of HIV prevalence per country
A_Mean <- A %>%
          group_by(country) %>%
          # na.rm = TRUE tells the function mean() to ignore missing values
          summarize(mean_hiv = mean(hiv_prevalence, na.rm = TRUE)) %>%
          arrange(desc(mean_hiv))

# Print the first 15 rows
print(A_Mean, n = 15)
```

Graph the data (for next week)

```
A_Mean_15 <- A_Mean[1:15, ]
ggplot(A_Mean_15, aes(x = mean_hiv,
                      y = fct_reorder(country, mean_hiv))) +
  coord_cartesian(xlim = c(0.7, max(A_Mean$mean_hiv))) +
  labs(x = "HIV Prevalence (avg.)", y = "") +
  geom_segment(aes(xend = 0, yend = country), size = 0.5) +
  geom_point(shape = 21, fill = "white", size = 2) +
  geom_vline(xintercept = 0, size = 0.5) +
  theme_minimal()
```

What's next?

- Refresher on Ordinary Least Squares (OLS) regression

Exercise solution 1

```
# EXERCISE 1
# 1. Download and load a Russian troll dataset from
# https://github.com/fivethirtyeight/russian-troll-tweets
# Note: The file and path will depend on what file you
#       download and where you saved it on your computer
library(tidyverse)

setwd("~/Downloads/")
IRA <- read_csv("IRAhandle_tweets_1.csv.bz2")
```

Exercise solution 2

```
# EXERCISE 2
# 2. What are the most frequent and second-most
# frequent languages?
IRA %>%
  group_by(language) %>%
  summarize(n = n()) %>%
  arrange(desc(n))
```

Exercise solution 3

```
# EXERCISE 3
# 3. Which region's tweets were received by
#    the most followers?
IRA %>%
group_by(region) %>%
summarize(followers = sum(followers)) %>%
arrange(desc(followers))
```

Exercise solution 4

```
# EXERCISE 4
# 4. On average, how many followers did each tweet
# reach in each region?
IRA %>%
group_by(region) %>%
summarize(followers = mean(followers)) %>%
arrange(desc(followers))
```

Exercise solution 5

```
# EXERCISE 5
# 5. How many tweets are retweets in each language?
IRA %>%
group_by(language) %>%
summarize(num_retweets = sum(retweet)) %>%
arrange(desc(num_retweets))
```

or

```
IRA %>%
filter(retweet == 1) %>%
group_by(language) %>%
summarize(num_retweets = n()) %>%
arrange(desc(num_retweets))
```

Exercise solution 6

```
# EXERCISE 6
# 6. How many tweets are _not_ retweets in each language?
# There are various ways to test if a tweet is a retweet
# E.g. sum(retweet != 1), sum(!retweet), sum(retweet == 0)
IRA %>%
group_by(language) %>%
summarize(num_retweets = sum(retweet == 0)) %>%
arrange(desc(num_retweets))
```

or

```
IRA %>%
filter(retweet == 0) %>%
group_by(language) %>%
summarize(num_retweets = n()) %>%
arrange(desc(num_retweets))
```

Exercise solution 7

```
# EXERCISE 7
# 7. How frequently are Trump and Clinton mentioned in the tweets?
# This exercise is more involved.
# str_count(): tells you the number of pattern matches in a string
# e.g. str_count("Trump Trumps Clinton", pattern = "Trump") is 2
# e.g. str_count("Trump Trumps Clinton", pattern = "Clinton") is 1
# Then we sum up all of these counts with sum()
# Notice, however, the use of tolower(), which makes a string lowercase.
# Why do this?
# Because str_count("Trump trumps Clinton", pattern = "trump") return 1 not 2
# If we turn all characters to lower case first, however, we don't need to
# worry about case matching
# i.e. tolower("Trump trumps Clinton") returns "trump trumps clinton"
# Thus str_count(tolower("Trump trumps Clinton"), "trump") returns 2
IRA %>%
summarize(trump_mentions = sum(str_count(tolower(content),
                                pattern = "trump")),
          clinton_mentions = sum(str_count(tolower(content),
                                    pattern = "clinton")))
```