## Workshop #4

##

## Topic: Election turnout

# 1. Setup ----------------------------------------------------------------

## Set working directory

setwd("~/Desktop/FIU\_R\_Workshop/data")

## Call to functionality

library(tidyverse)

## President vs midterm years

president\_years <- seq(1980, 2012, 4)

midterm\_years <- seq(1982, 2014, 4)

## Read in data

turnout\_data <- read.csv("voter\_turnout.txt") %>%

filter(state != "United States",

state != "United States (Excl. Louisiana)") %>%

mutate(election\_type = if\_else(year %in% president\_years, "Presidential", "Midterm")) %>%

dplyr::select(year, election\_type, state, votes, eligible\_voters) %>%

as\_tibble()

write.csv(turnout\_data, "voter\_turnout.csv", row.names = FALSE)

turnout\_data <- read.csv("voter\_turnout.csv")

# 2. Explore the dataset --------------------------------------------------

turnout\_data

turnout\_data %>% head()

## How many years back does this go?

unique(turnout\_data$year)

## Which states have data?

unique(turnout\_data$state)

# 3. Introducing group\_by() and summarise() -------------------------------

## So we have all this data on state by state voting each year, but that's a

## lot of information - possibly more than we want to tackle all at once.

## Say we just want to look at voting trends across the whole United States, irrespective

## of state. How could we modify this data in order to get the total number of votes and voters

## for every year??

## Conceptually, what we would want is to add together all values for these two columns

## across all the states for any given year. We can accomplish this using the very handy functions

## group\_by() and summarise()

## group\_by() tells R that you want to assign "groups" within your data. R then knows that all observations

## for a given column belong to that group. Let's try it out by using year as a grouping variable

**## Go through powerpoint presentation here!**

##

##

##

##

##

## So let's check out the original data again

## Original data

turnout\_data

## After grouping

turnout\_data %>%

group\_by(year)

## What changed?? Well, with the data, nothing!!! Nothing has been modified

## But notice at the top of the data frame, it tells you what column is being used as the group,

## as well as how many unique values there are.

## The real functionality of group\_by() comes in once we use another function to modify this data

## Let's try it by using the summarise() function

## Summarise does what we just talked about: it takes a data table, typically a grouped data table,

## and summarises a current column to create a new column based on a function that you provide

## Let's add up the total votes per year. We'll call the new column "US\_votes", which will represent

## the sum of all the individual states' votes.

turnout\_data %>%

group\_by(year) %>%

summarise(US\_votes = sum(votes, na.rm = TRUE))

## The na.rm = TRUE is important because it removes any missing values that would otherwise mess

## with R's math. Here's what happens when you take out that command: And that's because it tries

## adding a number to not a number, which doesn't work

turnout\_data %>%

group\_by(year) %>%

summarise(US\_votes = sum(votes))

## So total number of votes is one thing, but we also want to know the total number of eligible voters

## Calculate total voters in the US

## To summarise another column, we just add another argument separated by a comma

turnout\_data %>%

group\_by(year) %>%

summarise(US\_votes = sum(votes, na.rm = TRUE), # This line sums the total number of votes

US\_eligible = sum(eligible\_voters, na.rm = TRUE)) # This line sums the total number of eligible voters

## Notice that the summarise() function gets rid of extra columns. That is because it creates a new value for every

## unique value specified in the group\_by() function. Notice that there were 18 unique years in the whole dataset.

## Now, there are 18 rows in this new data table because we grouped by year. We wouldn't be able to retain

## any of the state name values, since the values associated with state all got summed together to create the new columns.

## So what if we want to group by two columns? For instance, what if we wanted to calculate the mean number of votes for each

## state across all years in the dataset, but also by each election type?

turnout\_data %>%

group\_by(election\_type, state)

## Notice that there were 51 unique states in the dataset (including DC) and 2 unique election types

## Now there are 102 unique groups, since there are 51\*2 unique state\*election\_type combinations!

turnout\_data %>%

group\_by(election\_type, state) %>%

summarise(mean\_votes = mean(votes, na.rm = TRUE))

## Now there are 102 rows! A value for every unique combination

## So what if we want to keep the election\_type column in with the first summarise() we did?

## Each year has only one election type - there are never multiple election types in a given year

## So if there are 18 unique years and 2 unique election types, will group\_by() create any new combinations?

## No - it still groups by year, but the election\_type doesn't add any new combo

US\_voter\_data <- turnout\_data %>%

group\_by(year, election\_type) %>%

summarise(US\_votes = sum(votes, na.rm = TRUE), # This line sums the total number of votes

US\_eligible = sum(eligible\_voters, na.rm = TRUE)) # This line sums the total number of eligible voters

## Notice, still only 18 rows. Does this make sense?

## And if you remove the election\_type grouping, nothing changes

turnout\_data %>%

group\_by(year) %>%

summarise(US\_votes = sum(votes, na.rm = TRUE), # This line sums the total number of votes

US\_eligible = sum(eligible\_voters, na.rm = TRUE)) # This line sums the total number of eligible voters

# 4. Calculate percentage --------------------------------------------------

## How has the number of votes in the US changed over the years?

US\_voter\_data %>%

ggplot(aes(year, US\_votes)) +

geom\_col()

## How has the number of eligible voters in the US changed over the years?

US\_voter\_data %>%

ggplot(aes(year, US\_eligible)) +

geom\_col()

## Challenge: Use mutate() to calculate the percentage of people who voted

US\_voter\_percent <- US\_voter\_data %>%

mutate(US\_perc\_vote = US\_votes/US\_eligible\*100)

## Plot percentage by year

US\_voter\_percent %>%

ggplot(aes(year, US\_perc\_vote)) +

geom\_col()

## Why does every other election vary so much??

## Fill by election type to find out

US\_voter\_percent %>%

ggplot(aes(year, US\_perc\_vote, fill = election\_type)) +

geom\_col()

## Midterm elections are much less voted in!

## Sidenote about adding proper limits

US\_voter\_percent %>%

ggplot(aes(year, US\_perc\_vote, fill = election\_type)) +

geom\_col() +

ylim(0,100)

# 5. Calculate voting record for Florida ----------------------------------

## Challenge time: Starting over, calculate the percent of voters per year/election type

## in Florida

## Then use group\_by and summarise to calculate the average percentage of eligible voters

## who actually voted per election type over the course of the data

## Two Products:

## 1) a graph of percentage of eligible voters who voted over time for only Florida

## 2) a summarised data table of the average percentage of people who voted over all the data

## in the dataset for florida

## Filter

FL\_votes <- turnout\_data %>%

# Filter the dataset to only include Florida

filter(state == "Florida") %>%

# Calculate percentage of voters

mutate(perc\_vote = votes/eligible\_voters\*100)

## Plot percentage of votes per year

FL\_votes %>%

ggplot(aes(year, perc\_vote, fill = election\_type)) +

geom\_col() +

ylim(0,100) +

ggtitle("Voter turnout for the state of Florida",

subtitle = "From 1992 to 2014") +

labs(x = "Year",

y = "Percentage of eligible voters who voted",

fill = "Election type")

## Calculate mean turnout

FL\_votes %>%

# Note: Since there is only one unique value for state in this new dataframe,

# you technically do not need to group by state, since it adds no extra unique combinations

# But I do, because it's helpful to see to keep context

group\_by(state, election\_type) %>%

summarise(avg\_turnout = mean(perc\_vote, na.rm = TRUE))

# 5. Fancy graph exploration ----------------------------------------------

# This is a necessary package for the graphs below

library(ggrepel)

# \* 5.1 Presidential vs Midterm for all states combined -------------------

elec.type.sum.stats <- turnout\_data %>%

mutate(perc\_vote = votes/eligible\_voters\*100) %>%

group\_by(election\_type) %>%

summarise(perc\_vote = mean(perc\_vote, na.rm = TRUE))

## Presidential vs Midterm for 2012 and 2014

turnout\_data %>%

# Calculate percentage of voters

mutate(perc\_vote = votes/eligible\_voters\*100) %>%

# Filter for the two most recent elections (2012 is presidential, 2014 is a midterm)

filter(year == 2012 | year == 2014) %>%

# Group by election type and create a new column that denotes, for each election type/year,

# which state had either the highest percent turnout (max(perc\_vote)) or the

# lowest percent turnout (min(perc\_vote)). Used for conditionally adding text labels

group\_by(election\_type) %>%

mutate(max\_or\_min = if\_else(

(perc\_vote == max(perc\_vote, na.rm = TRUE) | perc\_vote == min(perc\_vote, na.rm = TRUE)), 1, 0

)) %>%

ggplot(aes(x = election\_type, y = perc\_vote, fill = perc\_vote)) +

# Because of the order I want the layers, add the text first, only labeling points which represent

# states that have either the most or the least turnout (with the new column I created earlier using mutate)

geom\_text\_repel(aes(label = if\_else(max\_or\_min == 1,

paste(state, ": ", round(perc\_vote, 1), "%", sep = c("")),

'')), # This if\_else() statement only applies a label if a condition is met

# the position\_jitter argument makes the points divert a little bit from their coordinates to minimize

# overlap and create the effect of a point cloud

position = position\_jitter(seed = 1L,

width = .15),

# Size, segment, etc all change label appearance

size = 4,

segment.color = "black",

segment.size = .4,

segment.alpha = .5,

force = 3,

box.padding = 1.5,

direction = "both",

min.segment.length = 0) +

# This chunk adds the segment that leads from the 0% to the mean (white dot) of each group

geom\_segment(data = elec.type.sum.stats, aes(x=election\_type,

xend=election\_type,

y=0,

yend=perc\_vote),

color = "gray30") +

# This geom\_point() call adds points for each state represented in the dataset. The position\_jitter does the same thing as above

geom\_point(position = position\_jitter(seed = 1L,

width = .15),

size = 3,

shape = 21,

alpha = .9) +

# Set graph limits from 0 to 100%

ylim(0,100) +

# This creates a point for the mean of each group, displayed as a white point

# Notice, I overrid the data argument here. Normally, any calls to functions within the ggplot inherits the data that you

# feed in at the beginning. I created a new datatable with just the means (elec.type.sum.stats) because I wanted to display

# new summarised data that isn't present in the primary datatable, since it includes all the states together

geom\_point(data = elec.type.sum.stats,

aes(x = election\_type, y = perc\_vote),

size = 7,

shape = 21,

fill = "white") +

# coord\_flip() flips the plot's x and y axes!! Makes viewing the data more pretty

coord\_flip() +

# When I assigned the aesthetics to the graph in the ggplot(aes()) function, I told ggplot to assign "perc\_vote" to both

# the fill color AND the y-axis. This specifies custom colors for that gradient

scale\_fill\_gradient2(low = "red4", mid = "white", high = "blue3", midpoint = 50) +

# Create a title, subtitle, and axis labels

ggtitle("Voting percentages by election type", subtitle = "Data is from the 2012 presidential and 2014 midterm elections\nWhite dots are the averages of all states") +

labs(y = "Percent of eligible voters who voted", x = "Election Type", fill = "Percent voted") +

# Assign a theme by using some defaults of theme\_minimal, as well as adding on some of my own

theme\_minimal() +

theme(panel.grid.minor.x = element\_blank(),

panel.grid.major.y = element\_blank(),

plot.title = element\_text(size = 16,

margin = margin(b = 0.25, unit = "cm")),

plot.margin = margin(c(0.75, 0.75, 0.75, 0.75), unit = "cm"),

axis.text.x = element\_text(size = 12),

axis.text.y = element\_text(size = 12),

#axis.title.y = element\_text(size = 12),

axis.title.y = element\_blank(),

axis.title.x = element\_text(size = 12, margin = margin(t = 0.75, unit = "cm")),

legend.position = "none")