

Exercise 7: Mapping the Presidential Election

ECON 256

Data Analysis and Visualization

Objective

Plot a choropleth map in R.

1 Set up Your R Workspace

In addition to TidyVerse, this exercise will use a package called `sf` (which stands for simple feature).

Use `install.packages("sf")` to install the `sf` package.

Then set up your R script with the normal preface, but remember to initialize TidyVerse as well as `sf`. eg:

```
setwd ("/Users/justintyndall/Desktop/myproject")
library(tidyverse)
library(sf)
```

2 Download the Data

We will use two data sets. Both data sets are available on Laulima. The first is the county level voting results from the 2020 Presidential election (`election2020.csv`). We used this same data set in Exercise 6. Add this file to your working directory

The second data set is a shapefile of all US counties. Shapefiles need to be saved in a special way. In order to use the shapefile you need to download a folder that contains all the components of the shapefile. Download the zipped folder (`Counties.zip`) from Laulima to your working directory and unzip it.

3 Load the data into R

Load the `.csv` file into R in the normal way. Use the `read_csv()` function to save it as an object. Call the object `results`.

We load the shapefile by using the `read_sf()` function, as shown here:

```
counties <- read_sf(dsn = "Counties", layer = "Counties")
```

The argument `dsn` refers to the name of the folder that holds the shapefile. The argument `layer` refers to the name of the shapefile. (In this case they are named the same thing). Note the capitalization. Note that for the `layer` argument we don't include any file extension, ie we put `Counties` and not `Counties.shp`.

Now R will have a map (a shapefile) in its memory called `counties`.

4 Clean up the Data

Let's make a couple changes to the data so we can make an election map.

First, lets create a new variable in the election results data that indicates who won that county. Like this:

```
results2<-mutate(results, winner=ifelse(Biden>Trump, "Biden", "Trump"))
```

We are using the `mutate()` function to create the variable. Recall we can use the `ifelse()` function to create a logical condition. In this case we are saying: make a new variable called `winner`, if the Biden votes were greater than the Trump votes make the new variable equal to the word “Biden”, and if Biden votes were not greater than Trump votes make the new variable equal to the word “Trump”.

Now lets make a couple changes to the `counties` object that holds the shapefile. First lets drop Hawaii and Alaska. Its hard to represent all 50 states on the same projection, because Hawaii and Alaska are so far from the other 48. Use a `filter()` function to create a new object that only includes data on the other 48 states. Note there is a variable called `STATE_NAME`. We can use that in a `filter()` function to drop Hawaii and Alaska. Hints: recall we can use the symbol `!=` to specify that something is NOT equal. Also recall that we can include two logical conditions in a `filter()` function if they are separated by a `&` symbol.

Finally, lets create a new variable in shapefile data. We are going to join the two data sets together, but to do this they need to have a variable in common. The variable needs to: contain the same information, have the same name, and be the same variable type (numeric, character, factor...). The election results data has a variable that contains county level FIPS codes, called `county_fips`, that is saved as a numerical variable. The shapefile data has a variable called `FIPS`, that contains county level FIPS codes, called `FIPS`, that is saved as a character variable. So lets create a variable in the shapefile data that is compatible with the election results data. Something like this:

```
county3<-mutate(county2, county_fips=as.numeric(FIPS))
```

Now the two data sets have a variable in common.

5 Join the Two Objects

Lets join the shapefile data and election results data into a single object.

We can use a `left_join` as we have done before. The first argument needs to be the object containing the shapefile, the second argument will be the object containing the results data, and the third argument will be the variable used to join the data, in quotes. Something like this (with the blanks filled in):

```
_____<-left_join(_____, _____, by="_____")
```


Note that the new object will take on the object type of the input object you listed first. In this case, we want to put the spatial object (shapefile) first in the `left_join()` function so that the new object is also a spatial object.

6 Plot a Choropleth Map

Lets make a [choropleth map](#).

Add the following command, filling in the blank with the name of the simple feature object you created in the previous step:

```
ggplot(data=_____, aes(fill=winner))+  
  geom_sf()
```

Run your entire code to make sure it properly produces a map. If it is difficult to see your map in the small window, click on the “zoom” icon above your map to see a larger version ( Zoom)

It's simple to create a basic map. Add a version with some more options:

```
ggplot(data=_____,aes(fill=winner))+  
  geom_sf()+  
  theme_void()+  
  scale_fill_manual(values=c("blue","red"))+  
  labs(fill="Winning Candidate")
```

7 Save Your Map

What a beautiful map you have made!

Add the following command to save your map to your working directory. Note we can adjust the dimensions and give it a white background.

```
ggsave("mymap.png",height=7,width=10,bg="white")
```

8 Send me Your Code

Save your R script. Name it with your last name, followed by the exercise number and submit it on Laulima.