Exam 3

Connor_Hanna

7/9/2020

Question 1

"Clear the environment"

Done using GUI

Question 2

"Use the tidycensus package to"

```r
library(tidycensus)
library(tidyverse)
```

```
## -- Attaching packages ----------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   1.0.0
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts -------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(dplyr)
```

A)

"find the inequality Gini index variable explained on the last exam"

```r
vlist <- load_variables(2015, "acs5", cache = TRUE)
View(vlist)
```

B)

"import in the state-level inequality Gini estimates for 2010 and 2015 in the five-year American Community Survey as a single panel dataset"

```r
census_api_key("3ad99665ba1cb9142566342541aa52e1cd7ac642", overwrite = FALSE, install = FALSE)
```

```
## To install your API key for use in future sessions, run this function with `install = TRUE`.
```

```r
inequality_panel_2010 <- get_acs(geography = "state", variables = "B19083_001", year = 2010)
```

```
## Getting data from the 2006-2010 5-year ACS
```

```r
inequality_panel_2015 <- get_acs(geography = "state", variables = "B19083_001", year = 2015)
```

```
## Getting data from the 2011-2015 5-year ACS
```

```
inequality_panel_2010$year <- 2010

inequality_panel_2015$year <- 2015

inequality_panel <- bind_rows(inequality_panel_2010, inequality_panel_2015)
```

C)

"rename estimate as gini in your final data frame, which you should call inequality_panel"

Dataframe was named appropriately when called, leaving the variable renaming.

```
inequality_panel <- inequality_panel %>% rename(gini = estimate)
```

D)

"rename NAME to state as well"

```
inequality_panel <- inequality_panel %>% rename(state = NAME)
```

E)

"ensure that inequality_panel has a year variable so we can distinguish between the 2010 and 2015 gini index data"

```
View(inequality_panel)
```

F)

"as a final step, run the head() command so we can get a quick peak at inequality_panel"

```
head(inequality_panel)
```

```
## # A tibble: 6 x 6
##   GEOID state      variable    gini   moe  year
##   <chr> <chr>      <chr>      <dbl> <dbl> <dbl>
## 1 01    Alabama    B19083_001 0.47  0.003  2010
## 2 02    Alaska     B19083_001 0.412 0.006  2010
## 3 04    Arizona    B19083_001 0.453 0.002  2010
## 4 05    Arkansas   B19083_001 0.459 0.003  2010
## 5 06    California B19083_001 0.469 0.001  2010
## 6 08    Colorado   B19083_001 0.455 0.003  2010
```

3)

"Reshape the inequality_panel wide, such that the gini values for 2010 and 2015 have their own columns. Also, please keep both the state and GEOID variables. Call the resulting data frame inequality_wide. After you are done with the reshape, run the head() command so we can get a quick peak at the data."

```
library(tidyr)
inequality_wide <- spread(inequality_panel, year, gini)
head(inequality_wide)
```

```
## # A tibble: 6 x 6
##   GEOID state    variable      moe `2010` `2015`
##   <chr> <chr>    <chr>       <dbl>  <dbl>  <dbl>
## 1 01    Alabama B19083_001 0.0023 NA      0.475
## 2 01    Alabama B19083_001 0.003   0.47   NA
## 3 02    Alaska  B19083_001 0.006   0.412 NA
## 4 02    Alaska  B19083_001 0.0062 NA      0.418
## 5 04    Arizona B19083_001 0.0016 NA      0.465
## 6 04    Arizona B19083_001 0.002   0.453 NA
```

4)

“Reshape inequality_wide to long format. Once you are done, run the head() command so we can get a quick peak at the data.”

```
inequality_long <- gather(inequality_wide, key = "year", value = "gini", c(5, 6), na.rm = TRUE)
head(inequality_long)
```

```
## # A tibble: 6 x 6
##   GEOID state      variable     moe year   gini
##   <chr> <chr>      <chr>      <dbl> <chr> <dbl>
## 1 01    Alabama    B19083_001 0.003 2010  0.47
## 2 02    Alaska     B19083_001 0.006 2010  0.412
## 3 04    Arizona    B19083_001 0.002 2010  0.453
## 4 05    Arkansas   B19083_001 0.003 2010  0.459
## 5 06    California B19083_001 0.001 2010  0.469
## 6 08    Colorado   B19083_001 0.003 2010  0.455
```

5)

“Show with some R code that inequality_panel and inequality_long have the same number of observations”

```
count_panel <- count(inequality_panel)
count_long <- count(inequality_long)
count_long == count_panel
```

```
##         n
## [1,] TRUE
```

6)

“Collapse the inequality_long data frame by state, such that you obtain a single mean gini score for each state for the years 2010 and 2015. When collapsing, also keep both the GEOID and state variables. Call your resulting data frame inequality_collapsed”

```
inequality_collapsed <-
  inequality_long %>%
    group_by(state, GEOID) %>%
    summarize(mean_gini = mean(gini))
```

```
## `summarise()` regrouping output by 'state' (override with `.groups` argument)
```

```
head(inequality_collapsed)
```

```
## # A tibble: 6 x 3
## # Groups:   state [6]
##   state      GEOID mean_gini
##   <chr>      <chr>     <dbl>
## 1 Alabama    01        0.473
## 2 Alaska     02        0.415
## 3 Arizona    04         0.459
## 4 Arkansas   05         0.465
## 5 California 06         0.477
## 6 Colorado   08         0.457
```

7)

"Produce a map of the United States that colors in the state polygons by their mean gini scores from inequality_collapsed, using the WGS84 coordinate system. When doing so, use the viridis color scheme."

```r
library(maps)
```

```
##
## Attaching package: 'maps'

## The following object is masked from 'package:purrr':
##
##     map
```

```r
library(mapdata)
library(stringr)
library(ggmap)
```

```
## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.

## Please cite ggmap if you use it! See citation("ggmap") for details.
```

```r
library(ggplot2)
library(stringr)

states <- map_data("state")
head(states)
```

```
##        long      lat group order  region subregion
## 1 -87.46201 30.38968     1     1 alabama      <NA>
## 2 -87.48493 30.37249     1     2 alabama      <NA>
## 3 -87.52503 30.37249     1     3 alabama      <NA>
## 4 -87.53076 30.33239     1     4 alabama      <NA>
## 5 -87.57087 30.32665     1     5 alabama      <NA>
## 6 -87.58806 30.32665     1     6 alabama      <NA>
```

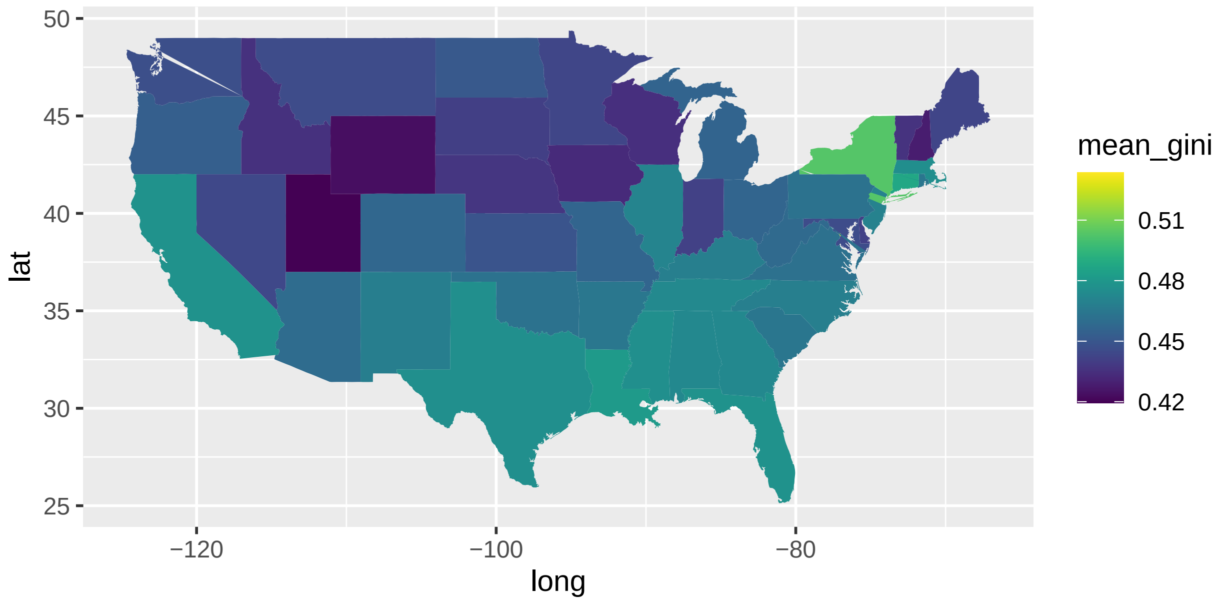
```r
states <- rename(states, "state" = "region")

inequality_collapsed_mapping <- inequality_collapsed
inequality_collapsed_mapping$state = tolower(inequality_collapsed_mapping$state)

states_gini <- left_join(states, inequality_collapsed_mapping, by = "state", copy = TRUE)
head(states_gini)
```

```
##        long      lat group order   state subregion GEOID mean_gini
## 1 -87.46201 30.38968     1     1 alabama      <NA>    01   0.47265
## 2 -87.48493 30.37249     1     2 alabama      <NA>    01   0.47265
## 3 -87.52503 30.37249     1     3 alabama      <NA>    01   0.47265
## 4 -87.53076 30.33239     1     4 alabama      <NA>    01   0.47265
## 5 -87.57087 30.32665     1     5 alabama      <NA>    01   0.47265
## 6 -87.58806 30.32665     1     6 alabama      <NA>    01   0.47265
```

```r
library(viridis)
```

```
## Loading required package: viridisLite
```

```
ggplot(data = states_gini) +
    scale_fill_gradientn(colors = viridis(n = 15537)) +
    geom_polygon(aes(x = long, y = lat, group = state, fill = mean_gini)) +
  coord_fixed(1.3)
```



8)

"Use the WDI package to import in data on Gross Domestic Product (GDP) in current US dollars. When doing so, include all countries and only the years 2006 and 2007. Rename your GDP variable to gdp_current."

```
library(WDI)
library(dplyr)

WDI_import <- WDI(country = "all", indicator = "6.0.GDP_current", start = 2006, end = 2007)
WDI_import <- rename(WDI_import, "gdp_current" = "6.0.GDP_current")
```

9)

"Deflate gdp_current to constant 2010 or 2015 US dollars, and call the new variable gdp_deflated. In words, also tell us the base year that you picked and why. At the end, run a head() command to prove that everything works"

```
library(priceR)

WDI_import$gdp_2010 <- NA
```

```
WDI_import$matching <- FALSE

for (row in 1:nrow(WDI_import)){
  country <- WDI_import$iso2c
  if (country_input_type(country, WDI_import) == "")
  WDI_import$matching <- TRUE
  }

WDI_import <- subset(WDI_import, matching == TRUE)

for (row in 1:nrow(WDI_import)){

  country <- WDI_import$iso2c
  inflation_dataframe <- retrieve_inflation_data(country)
  countries_dataframe <- show_countries()

  WDI_import$gdp_2010 <- adjust_for_inflation(WDI_import$gdp_current, 2020, country, to_date = 2010, in

  }
```

After spending an ungodly long time trying and failing to build a loop get the country codes to align using various methods and then apply inflation indices independently to each country, I gave up. The WDI data is labeled ISO2C but appears to feature ISO3C codes, and the inflation adjuster has refused to parse the country names.

10)

"In a Shiny app, what are the three main components and their subcomponents?"

A Shiny app is composed of a User Interface, a Server, and a Shinyapp function that synthesizes them into a finished application.

11)

"Pull this .pdf file from Mike Denly's webpage. It is a report on governance in Armenia that Mike Denly and Mike Findley prepared for the US Agency for International Development (USAID)."

```
library(tidyverse)
library(rvest)
```

```
## Loading required package: xml2

##
## Attaching package: 'rvest'

## The following object is masked from 'package:purrr':
##
##     pluck

## The following object is masked from 'package:readr':
##
##     guess_encoding
```

```
library(stringr)

url <- "https://pdf.usaid.gov/pdf_docs/PA00TNMG.pdf"
destfile <- "C:/Users/Connor/Documents/School/2020 SS/Data Science/Exam 3/usaid_pdf"
```

```
USAID_pdf <- download.file(url, destfile)
```

12

"Convert the text pulled from this .pdf file to a data frame, using the stringsAsFactors=FALSE option. Call the data frame armeniatext"

```
library(purrr)
library(pdftools)

USAID_text <- pdf_text("C:/Users/Connor/Documents/School/2020 SS/Data Science/Exam 3/usaid_pdf")

armeniatext <- data.frame((USAID_text),
                stringsAsFactors = FALSE)
```

I can't seem to figure out what's broken with the package. The code should work and pdftools is clearly finding the PDF, but it looks like it's unable to process it. I tried getting other parts of the pdftools package to parse it to no avail.

13)

"Tokenize the data by word and then remove stop words"

```
library(tokenizers)
library(stopwords)

armeniatext <- tokenize_words(armeniatext, stopwords = stopwords::("en"))
```

14)

"Figure out the top 5 most used word in the report"

```
library(tidytext)

armeniawords <- unnest_tokens(word, armeniatext)

armeniawords %>%
  count(word, sort = TRUE)

armeniatext %>%
  count(word, sort = TRUE)
```

I can't really see what I'm doing because I can't debug my code, but this looks about right. I included both datasets in case the previous function already unnested the words in addition to tokenizing them.

15)

"Load the Billboard Hot 100 webpage, which we explored in the course modules. Name the list object: hot100exam"

```
library(rvest)

hot100exam <- read_html("https://www.billboard.com/charts/hot-100")
```

read_html fails to read most of the text and I'm not certain why. readLines reads the entire HTML text of the webpage, but the output returns an error when parsed by html_nodes. I'm leaving read_html in the code because I couldn't get html_nodes to parse useful information either way.

16)

"Use rvest to obtain identify all of the nodes in the webpage"

```r
library(rvest)

html_nodes(hot100exam)
```

17)

"Use Google Chrome developer to identify the necessary tags and pull the data on Rank, Artist, Title, and Last Week"

Without the ability to debug the code in #16, I'm not sure if I can actually complete this.