Exam3

Mizuki Kurata

7/10/2020

1. Clear the environment. [5 points]

```
#clear environment
rm(list=ls(all=TRUE))
```

2. Use the tidycensus package to

(a) find the inequality Gini index variable explained on the last exam

```
library(tidycensus)
```

```
## Warning: package 'tidycensus' was built under R version 4.0.2
```

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------- tidyverse 1.3.
```

```
## v ggplot2 3.3.1     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

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

```
#getting access to census and acs
census_api_key("b0f75887f4369253f295f4c59d8527a5d420cf32",
               install = TRUE, overwrite = TRUE)
```

```
## Your original .Renviron will be backed up and stored in your R HOME directory if needed.
```

```
## Your API key has been stored in your .Renviron and can be accessed by Sys.getenv("CENSUS_API_KEY").
## To use now, restart R or run `readRenviron("~/.Renviron")`
```

```
## [1] "b0f75887f4369253f295f4c59d8527a5d420cf32"
```

```
#loading 2010 and 2015 variables
acs2010 = load_variables(year = 2010,"acs5")
acs2015 <- load_variables(year = 2015,"acs5")
```

(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;

```
#getting gini from each year
import_gini_2010 <- get_acs(geography = "state",
                  variables = c(gini = c("B19083_001")),
                  year = 2010)
```

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

```
import_gini_2010$year = 2010

import_gini_2015 <- get_acs(geography = "state",
                  variables = c(gini = c("B19083_001")),
                  year = 2015)
```

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

```
import_gini_2015$year = 2015

#append data
import_gini = bind_rows(import_gini_2010,import_gini_2015)
```

(c) rename estimate as gini in your final data frame, which you should call inequality_panel;

```
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

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

```
inequality_panel = setnames(import_gini,"estimate","gini")
```

(d) rename NAME to state as well;

```
#rename
setnames(inequality_panel,"NAME","state")
```

(e) ensure that inequality_panel has a year variable so we can distinguish between the 2010 and 2015 gini index data;

– Done in step b.

(f) as a final step, run the head() command so we can get a quick peak at inequality_panel (Hint: you may need to import each year separately and then append the two data frames together.) [15 points]

```
head(inequality_panel)
```

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

```
#pivoting wider
inequality_wide =
  inequality_panel %>%
  pivot_wider(id_cols = c("state","GEOID"),
              names_from = "year",
              values_from = "gini",
              names_prefix = "year_")
#head
head(inequality_panel)
```

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

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. [5 points]

```r
#pivoting longer
inequality_long =
inequality_wide %>%
  pivot_longer(cols = starts_with("year"),
               names_to = "year",
               names_prefix = "year_",
               values_to = "gini",
               values_drop_na = FALSE)
#head
head(inequality_long)
```

```
## # A tibble: 6 x 4
##   state   GEOID year   gini
##   <chr>   <chr> <chr> <dbl>
## 1 Alabama 01    2010  0.47
## 2 Alabama 01    2015  0.475
## 3 Alaska  02    2010  0.412
## 4 Alaska  02    2015  0.418
## 5 Arizona 04    2010  0.453
## 6 Arizona 04    2015  0.465
```

5. Show with some R code that inequality_panel and inequality_long have the same number of observations. [5 points]

```r
#number of observations for long
str(inequality_long)
```

```
## tibble [104 x 4] (S3: tbl_df/tbl/data.frame)
##  $ state: chr [1:104] "Alabama" "Alabama" "Alaska" "Alaska" ...
##  $ GEOID: chr [1:104] "01" "01" "02" "02" ...
##  $ year : chr [1:104] "2010" "2015" "2010" "2015" ...
##  $ gini : num [1:104] 0.47 0.475 0.412 0.418 0.453 ...
```

```r
dim(inequality_long)
```

```
## [1] 104   4
```

```r
#number of observations for panel
str(inequality_panel)
```

```
## tibble [104 x 6] (S3: tbl_df/tbl/data.frame)
##  $ GEOID   : chr [1:104] "01" "02" "04" "05" ...
##  $ state   : chr [1:104] "Alabama" "Alaska" "Arizona" "Arkansas" ...
##  $ variable: chr [1:104] "gini" "gini" "gini" "gini" ...
##  $ gini    : num [1:104] 0.47 0.412 0.453 0.459 0.469 0.455 0.482 0.436 0.535 0.471 ...
##  $ moe     : num [1:104] 0.003 0.006 0.002 0.003 0.001 0.003 0.003 0.005 0.005 0.002 ...
##  $ year    : num [1:104] 2010 2010 2010 2010 2010 2010 2010 2010 2010 2010 ...
```

```r
dim(inequality_panel)
```

```
## [1] 104   6
```

They both have 104 observations, but different variable numbers.

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. [5 points]

```
#collapsing, summarizing. summarizing numeric in unique state
inequality_collapsed =
 inequality_long %>%
  group_by(state,GEOID) %>%
  summarize(across(where(is.numeric),sum))
```

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

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. (Note: there are a few different ways to produce the map. We will accept any one of these ways, and the answer key will detail 3 ways. If you want to choose the method with the shape file, you can get the state-level shape file on the Census page). [10 points]

```
library(rio)
library(tidyverse)
library(googlesheets4)
```

```
## Warning: package 'googlesheets4' was built under R version 4.0.2
```

```
library(labelled)
```

```
## Warning: package 'labelled' was built under R version 4.0.2
```

```
library(data.table)
library(varhandle)
library(ggrepel)
```

```
## Warning: package 'ggrepel' was built under R version 4.0.2
```

```
library(geosphere)
```

```
## Warning: package 'geosphere' was built under R version 4.0.2
```

```
library(rgeos)
```

```
## Warning: package 'rgeos' was built under R version 4.0.2
```

```
## Loading required package: sp
```

```
## Warning: package 'sp' was built under R version 4.0.2

## rgeos version: 0.5-3, (SVN revision 634)
##  GEOS runtime version: 3.8.0-CAPI-1.13.1
##  Linking to sp version: 1.4-2
##  Polygon checking: TRUE
```

```
library(viridis)
```

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

```
#library(mapview)
library(rnaturalearth)
```

```
## Warning: package 'rnaturalearth' was built under R version 4.0.2
```

```
library(devtools)
```

```
## Warning: package 'devtools' was built under R version 4.0.2

## Loading required package: usethis

## Warning: package 'usethis' was built under R version 4.0.2
```

```
library(rnaturalearthhires)
library(sp)
library(sf)
```

```
## Warning: package 'sf' was built under R version 4.0.2

## Linking to GEOS 3.8.0, GDAL 3.0.4, PROJ 6.3.1
```

```
library(ggplot2)
#get map of US
#america <- ne_countries(country = 'United States',
 # scale = "medium",
  #returnclass = "sf")

#usa_map = ggplot()+
 # geom_sf(data = america)+
  #geom_sf(data = inequality_collapsed, aes(fill = `gini`))+
  #scale_fill_viridis(option = "viridis")+
  #ggtitle("")+
  #theme(plot.title = element_text(hjust=0.5))+
  #theme_void()

#print(usa_map)
```

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. [5 points]

```
library(WDI)
#import gdp data
import_gdp = WDI(country = "all",
                     indicator = "NY.GDP.DEFL.ZS",
                     start = 2006, end = 2007,
                     extra = FALSE,
                     cache = NULL)

#rename
setnames(import_gdp,"NY.GDP.DEFL.ZS","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. [5 points]

```
#subset to get a data frame only for US dollars
usd_deflator = subset(import_gdp,country == "United States")

#drop unnecessary variable
usd_deflator$iso2c = NULL
usd_deflator$country = NULL

#merge
#deflated_data = left_join(x = inequality_long,
                          # y = usd_deflator,
                          # by = "year")

#deflation
#inequality_long$deflated_amount = deflated_data$current_amount/
                                  #(deflated_data$deflator/100)
```

10. In a Shiny app, what are the three main components and their subcomponents? [5 points]

```
#library(shiny)
#component 1: User Interface (UI) [an HTML page]
  #subcomponent: assigning input and outputs
#ui <- fluidPage("Hello, World")
#component 2: server function
  #subcomponent: rendering functions, assigning it to output variables
#server <- function(input,output) {
#}
#component 3: execution, telling shiny what the ui and server is.
#shinyApp(ui = ui,server = server)
```

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). [5 points]

```
library(pdftools)
```

```
## Warning: package 'pdftools' was built under R version 4.0.2
```

```
## Using poppler version 0.73.0
```

```
library(tidyr)
library(tidytext)
```

```
## Warning: package 'tidytext' was built under R version 4.0.2
```

```
library(dplyr)
library(stringr)
#importing pdf
md_import = pdf_text(pdf = "https://pdf.usaid.gov/pdf_docs/PA00TNMG.pdf")
```

12. Convert the text pulled from this .pdf file to a data frame, using the , stringsAsFactors=FALSE option. Call the data frame armeniatext. [5 points]

```
#convert to data frame
armeniatext = data.frame(md_import,stringAsFactors=FALSE)
```

13. Tokenize the data by word and then remove stop words. [5 points]

```
data("stop_words")
#tokenized and removed stop words
armeniatext = armeniatext %>%
  unnest_tokens(word,md_import) %>%
  anti_join(stop_words)
```

```
## Joining, by = "word"
```

14. Figure out the top 5 most used word in the report. [5 points]

```
#reordering by count
armeniatext %>%
  count(word, sort = TRUE) %>%
  group_by(n) %>%
  top_n(n=5)
```

```
## Selecting by n
```

```
## # A tibble: 4,228 x 2
## # Groups:   n [74]
##    word           n
##    <chr>      <int>
##  1 armenia      252
##  2 political    207
##  3 corruption   186
##  4 governance   185
##  5 democracy    132
##  6 evidence     126
##  7 review       121
##  8 service      119
##  9 2017         106
## 10 public       105
## # ... with 4,218 more rows
```

15. Load the Billboard Hot 100 webpage, which we explored in the course modules. Name the list object: hot100exam [5 points]

```
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
```

```
#read html
hot100 = "https://www.billboard.com/charts/hot-100"
hot100exam = read_html(hot100)
```

16. Use rvest to obtain identify all of the nodes in the webpage. [5 points]

```
body_nodes <- hot100exam %>%
  html_node("body") %>%
  html_children()
```

17. Use Google Chrome developer to identify the necessary tags and pull the data on Rank, Artist, Title, and Last Week. HINT 1: In class we showed you how to get the first three of these. You simply need to add the Last Week ranking. HINT 2: You can navigate two ways. Hovering to find what you need or by doing Cmd+F / Ctrl+F and using actual data to find the location. HINT 3: You're looking to update the code based on the way the information is in referenced. Try out some different options and see what shows up in the environment. Keep trying until you see that you have a chr [1:100] with values that correspond to what is in the web page. [5 points]

```
rank = hot100exam %>%
  rvest::html_nodes('body') %>%
  xml2::xml_find_all("//span[contains(@class,
                     'chart-element__rank__number')]") %>%
  rvest::html_text()

artist= hot100exam %>%
  rvest::html_nodes('body') %>%
  xml2::xml_find_all("//span[contains(@class,
                     'chart-element__information__artist')]") %>%
  rvest::html_text()

title = hot100exam %>%
  rvest::html_nodes('body') %>%
  xml2::xml_find_all("//span[contains(@class,
```

```
                    'chart-element__information__song')]") %>%
  rvest::html_text()

lastweek = hot100exam %>%
  rvest::html_nodes('body') %>%
  xml2::xml_find_all("//div[contains(@class,
                    'chart-element__meta')]") %>%
  rvest::html_text()
```

Final question. Save all of the files (i.e. .Rmd, .dta, .pdf/Word Doc), push them to your GitHub repo, and provide us with the link to that repo. [no points; 15-point penalty for lack of submission (see above)]

exam3 github