Exam 3

Ricardo Corona

7/9/2020

#clearing the environment  
#rm(list=ls(all=TRUE))  
  
#loading tidycensus  
library(tidycensus)

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

#loading the API Key  
census\_api\_key("0426b3f1f3fcd947dc8359b771cb0392e2429626")

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

#loading the American Community Survey for 2010 and 2015  
v15 = load\_variables(year = 2015,  
 dataset = "acs5")  
v10 = load\_variables(year = 2010,  
 dataset = "acs5")  
  
#obtaining gini indexes for 2010 and 2015  
#2015 gini index  
gini\_data15 = get\_acs(geography = "state",  
 variables = c(gini\_score2015 = c("B19083\_001")),  
 year = 2015)

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

#2010 gini index  
gini\_data10 = get\_acs(geography = "state",  
 variables = c(gini\_score2010 = c("B19083\_001")),  
 year = 2010)

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

#appending gini data into inequality dataset  
library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.0.2

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

## v ggplot2 3.3.2 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

## Warning: package 'ggplot2' was built under R version 4.0.2

## -- Conflicts ------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

inequality\_panel = bind\_rows(gini\_data15, gini\_data10)  
inequality\_panel$year = ifelse(inequality\_panel$variable ==   
 "gini\_score2015","2015","2010")  
#renaming variable names in 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

setnames(inequality\_panel, "NAME","state")  
setnames(inequality\_panel, "estimate","gini")  
  
#running head on inequality panel to view data  
head(inequality\_panel)

## # A tibble: 6 x 6  
## GEOID state variable gini moe year   
## <chr> <chr> <chr> <dbl> <dbl> <chr>  
## 1 01 Alabama gini\_score2015 0.475 0.0023 2015   
## 2 02 Alaska gini\_score2015 0.418 0.0062 2015   
## 3 04 Arizona gini\_score2015 0.465 0.0016 2015   
## 4 05 Arkansas gini\_score2015 0.470 0.0025 2015   
## 5 06 California gini\_score2015 0.486 0.0008 2015   
## 6 08 Colorado gini\_score2015 0.459 0.0018 2015

#reshaping inequality data in wide format  
inequality\_wide=  
 inequality\_panel %>%   
 pivot\_wider(id\_cols = c("state","GEOID","year"),  
 names\_from = "year",  
 values\_from = "gini",  
 names\_prefix = "year\_")  
  
#head of inequality\_wide  
head(inequality\_wide)

## # A tibble: 6 x 4  
## state GEOID year\_2015 year\_2010  
## <chr> <chr> <dbl> <dbl>  
## 1 Alabama 01 0.475 0.47   
## 2 Alaska 02 0.418 0.412  
## 3 Arizona 04 0.465 0.453  
## 4 Arkansas 05 0.470 0.459  
## 5 California 06 0.486 0.469  
## 6 Colorado 08 0.459 0.455

#reshaping inequality\_wide to long format  
inequality\_long =  
 inequality\_wide %>%   
 pivot\_longer(cols = starts\_with("year"),  
 names\_to = "year",  
 names\_prefix = "year\_",  
 values\_to = "gini",  
 values\_drop\_na = FALSE) %>%   
 filter(!(gini == 0))  
  
#head of inequality\_long  
head(inequality\_long)

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

#comparing observations of inequality\_long and inequality\_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\_score2015" "gini\_score2015" "gini\_score2015" "gini\_score2015" ...  
## $ gini : num [1:104] 0.475 0.418 0.465 0.47 0.486 ...  
## $ moe : num [1:104] 0.0023 0.0062 0.0016 0.0025 0.0008 0.0018 0.0019 0.004 0.0042 0.0014 ...  
## $ year : chr [1:104] "2015" "2015" "2015" "2015" ...

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] "2015" "2010" "2015" "2010" ...  
## $ gini : num [1:104] 0.475 0.47 0.418 0.412 0.465 ...

#collpasing the inequality\_long data  
inequality\_collapsed=  
 inequality\_panel %>%   
 group\_by(state,gini,GEOID, year) %>%   
 summarize(across(where(is.numeric),mean))

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

#producing map of US with gini scores  
library(rnaturalearth)

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

library(viridis)

## Loading required package: viridisLite

#obatining just US  
#US = ne\_countries(state = 'united\_states',  
 #scale = "medium",  
 #returnclass = "sf")  
  
#plotting map  
#library(ggplot2)  
#US\_map = ggplot() +  
 #geom\_sf(data = US )+  
 #geom\_sf(data = inequality\_collapsed, aes(fill = gini))+  
 #scale\_fill\_viridis(option = "viridis")  
  
#print(US\_map)  
  
#using WDI package to load in GDP data  
library(WDI)  
deflator\_data = WDI(country = "all",  
 indicator = "NY.GDP.MKTP.CD",  
 start = 2006, end = 2017,  
 extra = FALSE, cache = NULL)  
  
#renaming GDP variable to gdp\_cuuren  
setnames(deflator\_data, "NY.GDP.MKTP.CD", "gdp\_current")  
  
#subset to get a data frame only for US dollars  
usd\_deflator = subset(deflator\_data, country == "United States")  
  
#subset(usd\_deflator, deflator==100)  
  
#deflated\_data = left\_join(x = inequality\_collapsed,  
 #y = usd\_deflator,  
 # by = "year")  
#deflating data  
#deflated\_data$deflated\_amount = deflated\_data$gdp\_amount/  
 #(deflated\_data$deflator)/100  
  
#head(deflated\_data)

The main three components of a shiny app are the User Interface, which includes inputs and outputs, the server, in which you must store, render, and refer the outputs, and then the app exectuion in which you must specify the user Interface and the server.

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)  
  
#pulling pdf text from online  
online\_text=pdf\_text(pdf = "https://pdf.usaid.gov/pdf\_docs/PA00TNMG.pdf")  
  
#converting text to data frame  
armeniatext=as.data.frame(online\_text, stringAsFactors = FALSE)  
  
#tokenizing the data and removing stop words  
data(stop\_words)  
#armeniatext = armeniatext %>%   
 #unnest\_tokens(word, text) %>%   
 #anti\_join(stop\_words)  
  
#looking for the top 5 most used words  
#armeniatext %>%   
 #count(word, sort = TRUE)  
  
#loading the hot100 bilboards page  
#hot100page = "https://www.billboard.com/charts/hot-100"  
#hot100exam = read\_html(hot100page)  
  
#using rvest fucntion to idntify all nodes  
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

#body\_nodes <-hot100exam%>%  
 #html\_node("body")%>%  
 #html\_children()  
  
#body\_nodes  
  
#pulling rank , artist, title ,and last week data  
#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()  
  
#last\_week = hot100exam %>%   
 #rvest::html\_nodes("body") %>%   
 #xml2::xml\_find\_all("//span[contains(@class,  
 #'chart-element\_\_information\_\_delta\_\_text')]") %>%   
 #rvest::html\_text()