~~~
---
title: "exam3"
author: "Maya Khan"
date: "7/9/2020"
output: pdf_document
---
library(rmarkdown)
#github link is https://github.com/mayazuberikhan/Data-Science-Class

1. Clearing the environment in R
```{r}
rm(list = ls(all=TRUE))
```



2. - Importing the tidycensus package using library(tidycensus), installing:
install.packages("tidycensus") was done first
The Gini index was then loaded in from the year 2010
Gini dateframe is then given
-
```{r tidycensus}
#loading the package
library(tidycensus)
index_2010 <- load_variables(year = 2010, "acs5")
index_2015 <- load_variables(year = 2015, "acs5")

#merging them
index <- c(index_2010, index_2015)
#creating data frame
inequality_panel <- get_acs(geography = "state", variables = index, year =
2015)
#calling inequality panel
inequality_panel
#renaming estimate
names(inequality_panel)[names(inequality_panel) == 'estimate'] <- 'gin'
#renaming NAME to state
names(inequality_panel)[names(inequality_panel) == 'NAME'] <- 'state'
head(inequality_panel)
```

3. Reshaping the panel to wide format
```{r}
#reshaping it to wide to 2010 and 2015 have own columns
inequality_wide <- reshape(inequality_panel, idvar = c(2010, 2015), timevar =
"gini", direction = "wide")

#taking a peak of the data
head(inequality_wide)
```
~~~

4. Reshaping the panel to long format

~~~
```{r}
#reshaping it to long for 2010 and 2015 to have its own columns
inequality_long <- reshape(inequality_panel, idvar = c(2010, 2015), timevar =
"gini", direction = "long")

#taking a peak of the data
head(inequality_wide)
```
~~~

5. R codes that shows if both inequality_panel and inequality_long have the same number of observations

~~~
```{r}
#this must return true if both dataframes are equal
#it will return false if they do not have the same number of observations.
all.equal(inequality_panel,inequality_long)

#compare is also another R code to check
compare(inequality_panel, inequality_long)
```
~~~

6. Collapsing the inequality_long data frame by state

~~~
```{r}
#loading the dplyr package for collasping data
library(dplyr)
#collapse inequality_long
inequality_long %>%
    filter(year == 2010, year = 2015) %>%
    group_by(state) %>%
    summarize(mean_gini = mean(gini))
#renaming the new collapsed data
inequality_collapsed <- inequality_long
```
~~~

7. Creating a map of the United States

~~~
```{r}
#loading package to extract coordinates
library(sf)
#loading in the map borders based on the coordinate system
map_borders <- st_transform(s.sf, "+proj=longlat +ellps=WGS84 +datum=WGS84")
#shape is no longer needed
rm(map_borders)

#creating the map with viridis color scheme
us_map = ggplot() +
~~~

~~~
geom_sf(data = inequality_collapsed, aes(fill= mean_gini)) +
scale_fill_viridis(option = "viridis") +
ggtitle("United States Gini Score") + theme(plot.title = element_text(hjust =
0.5)) +
theme_void()
```

8. importing WDI package and creating GDP
```{r}
#installing the WDI package
install.packages('WDI')
#Including all US countries
GDP <- WDI(country="all", start = 2006, end = 2007)
#checking for the column/variable names
head(GDP)
#renaming GDP variable (not my dataframe) to gdp_current
names(gdp)[names(gdp) == 'GDP'] <- 'gdp_current'
```

9. Deflating GDP current
I chose 2015 because it has the most recent GDP values of each country and the
values were immensely different compared to 2010
```{r}
gdp_deflated <- deflate(GDP$gdp_current, 2015)
#peeking new dataframe
head(gdp_deflated)
```

10. In the Shiny app, the three main components are:
1. Installing libaries with subcomponents of packages
2. User Interface with subcomponents of design and user experience
3. The server used with subcomponents of app plots and widgets

11.Pulling Armenian Report for US Agency
```{r}
#getting the file report
report <- pdf_text("PA00TNMG.pdf")
```

12. convert text in to a dataframe
```{r}
#convert to dataframe
armeniatext <- as.data.frame(lapply(report, trimws), stringsAsFactors = FALSE)
```

13. Tokenize by word
```{r}
#tokenize words here
token_words <- armeniatext %>%
~~~

~~~
  unnest_tokens(word, text)
#remove stop words here
token_words <- token_words %>% anti_join(stop_words)
```

14. R code for the top 5 words in report
```{r}
#R code for the top 5 words in report
token_words %>%
  count(word, sort = TRUE)
```

15. Loading in Billboard page
``` {r}
#billboard data frame of top 100
hot100exam <- "https://www.billboard.com/charts/hot-100"
hot100 <- read_html(hot100exam)
```

16. Using revest for all nodes
``` {r}
#obtaining all using html_notes and xml functions
all <- hot100 %>%
+     rvest::html_nodes('all') %>% xml2::xml_find_all("//td[contains(@data-th,
'CASE')]") %>% rvest::html_text()
```

17. Google developer
``` {r}
#gets rank
rank <- hot100 %>%
rvest::html_nodes('all') %>% xml2::xml_find_all("//span[contains(@class,
'chart-element__rank__number')]") %>%
rvest::html_text()

#gets the artist
artist <- hot100 %>% rvest::html_nodes('all') %>% xml2::xml_find_all("//
span[contains(@class,
'chart-element__information__artist')]") %>%
rvest::html_text()

#gets the title
title <- hot100 %>% rvest::html_nodes('all') %>% xml2::xml_find_all("//
span[contains(@class,
'chart-element__information__song')]") %>%
  rvest::html_text()

#gets last weeks
~~~

~~~
last_week <- hot100 %>% rvest::html_nodes('all') %>% xml2::xml_find_all("//
span[contains(@class,
'chart-element__information__last__week')]") %>%
rvest::html_text()

```


render("exam3.Rmd", "pdf_document")
~~~