Data 607 - Final Project

Project Proposal: Understanding the Link Between Socioeconomic Status and Educational Attainment

Motivation:

The motivation for this project is rooted in the critical examination of how socioeconomic status (SES) shapes educational opportunities and outcomes. One significant source of inspiration is the article Education and Socioeconomic Status (https://www.apa.org/pi/ses/resources/publications/education) published by the American Psychological Association. This article highlights the profound influence of SES on access to quality education, academic achievement, and long-term economic security.

The disparities in educational attainment associated with SES are not just numbers—they represent systemic barriers faced by millions of individuals, often perpetuated across generations. Children from lower-income households frequently experience fewer opportunities for academic success due to factors such as under-resourced schools, limited access to advanced coursework, and a lack of support for extracurricular enrichment. These inequities translate into significant disparities in higher education attainment, including bachelor's degree completion, a key driver of upward mobility.

In addition to the APA article, the project draws insights from the findings in the article Socioeconomic Status and Student Learning: Insights from an Umbrella Review (https://link.springer.com/article/10.1007/s10648-024-09929-3). This review sheds light on the cumulative effects of SES, emphasizing its role in shaping early learning experiences and perpetuating disparities in education outcomes. By integrating these perspectives, the project aims to address the pressing need to better understand and quantify these inequities.

Through this analysis, I aim to go beyond merely documenting disparities—I seek to uncover actionable insights that can inform policies and interventions aimed at closing the equity gap in education. By analyzing the relationship between per capita income and bachelor's degree attainment across U.S. counties, this project aspires to contribute to the broader effort to promote educational equality and economic opportunity for all.

Goal of the Project:

The primary objective of this project is to evaluate the relationship between per capita income and educational attainment, with a specific focus on bachelor's degree completion rates across U.S. counties. Educational attainment is a key measure of economic mobility and social progress, and understanding the factors that influence it is critical for addressing disparities and fostering equitable opportunities.

This project seeks to quantify the impact of socioeconomic status, as measured by per capita income, on access to and achievement in higher education. By examining this relationship, the analysis aims to reveal patterns and correlations that highlight how economic factors influence educational success. This understanding is essential for identifying areas where targeted interventions or policy changes may have the greatest effect in reducing inequities.

In addition to measuring the impact of income levels on educational outcomes, the findings will provide insights into the broader societal implications of economic disparities. By addressing this question through a data-driven approach, the project aims to contribute to the development of strategies that promote equitable educational attainment, empowering individuals and communities to overcome systemic barriers and achieve long-term success.

How to Achieve the Goal:

- **Collect the Required Data:** Gather relevant datasets from reliable sources, including socioeconomic indicators and educational attainment data. This will involve web scraping, API calls, and importing CSV files to ensure diverse and comprehensive data coverage.
- **Perform Exploratory Data Analysis (EDA):** Conduct an in-depth analysis of the data structure, distribution, and trends to identify key features and outliers. Use visualizations and summary statistics to understand the relationships between variables and guide the subsequent analysis steps.
- **Execute the Extract, Transform, and Load (ETL) Process:** Prepare the data by cleaning and standardizing formats, merging datasets from different sources, and handling missing or inconsistent values. Transform the raw data into a unified, analysis-ready format that aligns with the project's objectives.
- **Apply a Linear Regression Model:** Implement a linear regression analysis to examine the relationship between per capita income and bachelor's degree attainment. This will help quantify the extent to which income levels predict educational success.
- **Estimate the Correlation:** Calculate and interpret the correlation coefficient to evaluate the strength and direction of the relationship between per capita income and bachelor's degree attainment. This step will provide critical insights into the data-driven patterns.
- **Interpret the Results:** Analyze the outcomes of the regression model and correlation estimates to derive meaningful insights. Summarize findings in the context of socioeconomic disparities and their influence on educational opportunities, offering actionable recommendations for addressing these issues.

Data Sources:

- **FIPS Codes for U.S. Counties:** To identify counties across the United States, FIPS codes are collected through web scraping from the following resource: Wikipedia: List of United States FIPS Codes by County (https://en.wikipedia.org/wiki/List_of_United_States_FIPS_codes_by_county). These codes are essential for integrating datasets from multiple sources accurately.
- **Census Data via API:** Demographic and economic data for all U.S. counties are retrieved using the U.S. Census Bureau's 2019 ACS API. This dataset provides comprehensive insights into population and income levels: U.S. Census Bureau API (https://api.census.gov/data/2019/acs/acs1?get=NAME,B01001_001E&for=county:*).
- **Economic and Demographic Data:** Additional socioeconomic indicators, such as employment rates, poverty levels, and economic activity, are sourced from the County-Level Data Sets provided by the Economic Research Service of the U.S. Department of Agriculture. The CSV file can be downloaded from: ERS County-Level Data Sets (https://www.ers.usda.gov/data-products/county-level-data-sets/download-data/).

Importing Libraries

```
library(tidycensus)
```

```
## Warning: package 'tidycensus' was built under R version 4.1.3
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.1.3
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## Warning: package 'ggplot2' was built under R version 4.1.2
```

```
## Warning: package 'stringr' was built under R version 4.1.2
```

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

```
library(dplyr)
library(plotly)
```

```
## Warning: package 'plotly' was built under R version 4.1.3
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
library(tidyr)
library(stringr)
library(rvest)
```

```
## Warning: package 'rvest' was built under R version 4.1.3
```

```
##
## Attaching package: 'rvest'
```

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

```
library(ggpubr)
```

```
## Warning: package 'ggpubr' was built under R version 4.1.3
```

```
library(ggiraph)
```

```
## Warning: package 'ggiraph' was built under R version 4.1.3
```

```
library(ggiraphExtra)
```

```
## Warning: package 'ggiraphExtra' was built under R version 4.1.3
```

```
library(plyr)
```

```
## Warning: package 'plyr' was built under R version 4.1.3
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following object is masked from 'package:ggpubr':
##
##     mutate
```

```
## The following objects are masked from 'package:plotly':
##
##     arrange, mutate, rename, summarise
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

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

Web Scraping Counties fips code

Generate the html source code by calling the read_html function

```
webpage <- read_html("https://en.wikipedia.org/wiki/List_of_United_States_FIPS_codes_by_county")

tbls <- html_nodes(webpage, "table")

head(tbls)
```

```
## {xml_nodeset (5)}
## [1] <table class="box-Very_long plainlinks metadata ambox ambox-style ambox-v ...
## [2] <table class="wikitable sortable">\n<caption>\n</caption>\n<tbody>\n<tr>\ ...
## [3] <table class="nowraplinks mw-collapsible expanded navbox-inner" style="bo ...
## [4] <table class="nowraplinks hlist mw-collapsible autocollapse navbox-inner" ...
## [5] <table class="nowraplinks mw-collapsible autocollapse navbox-inner" style ...
```

Calling html_nodes function with CSS Selector and generating the table using html_table

```
fips_df <- webpage %>%
        html_nodes("#mw-content-text") %>% html_table() %>% .[[1]]

head(fips_df)
```

```
## # A tibble: 6 x 3
##   X1      X2                                                    X3
##   <chr>   <chr>                                                 <chr>
## 1 ""      This article may be too long to read and navigate comf~ <NA>
## 2 "FIPS"  County or equivalent                                  State or equi~
## 3 "01001" Autauga County                                        Alabama
## 4 "01003" Baldwin County                                        Alabama
## 5 "01005" Barbour County                                        Alabama
## 6 "01007" Bibb County                                           Alabama
```

Data Cleaning for fips_df:

Renaming the columns

```
fips_df <- fips_df %>%
  dplyr::rename(
    fips_code = X1,
    county = X2,
    state = X3
    )
head(fips_df)
```

```
## # A tibble: 6 x 3
##   fips_code county                                              state
##   <chr>     <chr>                                               <chr>
## 1 ""        This article may be too long to read and navigate co~ <NA>
## 2 "FIPS"    County or equivalent                                State or equi~
## 3 "01001"   Autauga County                                      Alabama
## 4 "01003"   Baldwin County                                      Alabama
## 5 "01005"   Barbour County                                      Alabama
## 6 "01007"   Bibb County                                         Alabama
```

Removing the unwanted rows

```
fips_df <- fips_df[-c(1, 2), ]

head(fips_df)
```

```
## # A tibble: 6 x 3
##   fips_code county         state
##   <chr>     <chr>          <chr>
## 1 01001     Autauga County Alabama
## 2 01003     Baldwin County Alabama
## 3 01005     Barbour County Alabama
## 4 01007     Bibb County    Alabama
## 5 01009     Blount County  Alabama
## 6 01011     Bullock County Alabama
```

```
colnames(fips_df)
```

```
## [1] "fips_code" "county"    "state"
```

```
# Drop the columns of the dataframe
#select (fips_df,-c('.'))
```

```
fips_df <- na.omit(fips_df) # Method 1 - Remove NA
head(fips_df)
```

```
## # A tibble: 6 x 3
##   fips_code county         state
##   <chr>     <chr>          <chr>
## 1 01001     Autauga County Alabama
## 2 01003     Baldwin County Alabama
## 3 01005     Barbour County Alabama
## 4 01007     Bibb County    Alabama
## 5 01009     Blount County  Alabama
## 6 01011     Bullock County Alabama
```

Removing string ‘county’

```
#fips_df$county<-gsub(" County","",as.character(fips_df$county))

#head(fips_df)
```

```
names(fips_df) <- tolower(names(fips_df))

head(fips_df)
```

```
## # A tibble: 6 x 3
##   fips_code county         state
##   <chr>     <chr>          <chr>
## 1 01001     Autauga County Alabama
## 2 01003     Baldwin County Alabama
## 3 01005     Barbour County Alabama
## 4 01007     Bibb County    Alabama
## 5 01009     Blount County  Alabama
## 6 01011     Bullock County Alabama
```

Checking the working directory path and exporting data frame into

a CSV file

```
# Get working directory path
path <- getwd()

path
```

```
## [1] "C:/Users/CUNY_SPS_PROJECTS/Data_607_Final_Project"
```

Export file as csv to working directory.

```
write.csv(fips_df, file.path(path, "/posgresql/fips_data.csv"))
```

```
fips_df$NAME = paste(fips_df$county, fips_df$state, sep=", ")
names(fips_df) <- tolower(names(fips_df))
head(fips_df)
```

```
## # A tibble: 6 x 4
##   fips_code county         state   name
##   <chr>     <chr>          <chr>   <chr>
## 1 01001     Autauga County Alabama Autauga County, Alabama
## 2 01003     Baldwin County Alabama Baldwin County, Alabama
## 3 01005     Barbour County Alabama Barbour County, Alabama
## 4 01007     Bibb County    Alabama Bibb County, Alabama
## 5 01009     Blount County  Alabama Blount County, Alabama
## 6 01011     Bullock County Alabama Bullock County, Alabama
```

```
#str(census_df)
str(fips_df)
```

```
## tibble [3,249 x 4] (S3: tbl_df/tbl/data.frame)
##  $ fips_code: chr [1:3249] "01001" "01003" "01005" "01007" ...
##  $ county   : chr [1:3249] "Autauga County" "Baldwin County" "Barbour County" "Bibb County"
...
##  $ state    : chr [1:3249] "Alabama" "Alabama" "Alabama" "Alabama" ...
##  $ name     : chr [1:3249] "Autauga County, Alabama" "Baldwin County, Alabama" "Barbour Count
y, Alabama" "Bibb County, Alabama" ...
##  - attr(*, "na.action")= 'omit' Named int [1:13] 3243 3244 3245 3246 3247 3248 3249 3250 3251
3252 ...
##   ..- attr(*, "names")= chr [1:13] "3243" "3244" "3245" "3246" ...
```

Preparing for API Call Census 2010 Data

```
Sys.setenv(census_api_key = "f7ce5a76ddf2088c73fda9c0a87410995cebbffa")
Sys.getenv("census_api_key")
```

```
## [1] "f7ce5a76ddf2088c73fda9c0a87410995cebbffa"
```

Access the API key

Save the Census API key in the environmental variable.

```
#census_api_key("YOUR KEY GOES HERE", install = TRUE)
```

```
Sys.getenv("census_api_key")
```

```
## [1] "f7ce5a76ddf2088c73fda9c0a87410995cebbffa"
```

Fetching Census Data for the year 2020

```
# https://api.census.gov/data/2019/acs/acs1?get=NAME,B01001_001E&for=county:*

census_df <- get_acs(
  geography = "county",
  variables = c(population = "B01003_001E",
                median_age = "B01002_001E",
                household_income = "B19013_001E",
                per_capita_income = "B19301_001E",
                poverty_count = "B17001_002E",
                unemployment_count = "B23025_005E"
                ),
  output = "wide",
  year = 2020
)
```

```
## Getting data from the 2016-2020 5-year ACS
```

```
head(census_df)
```

```
## # A tibble: 6 x 14
##   GEOID NAME      population B01003_001M median_age B01002_001M household_income
##   <chr> <chr>          <dbl>       <dbl>      <dbl>       <dbl>            <dbl>
## 1 01001 Autauga ~      55639          NA       38.6         0.6            57982
## 2 01003 Baldwin ~     218289          NA       43.2         0.4            61756
## 3 01005 Barbour ~      25026          NA       40.1         0.6            34990
## 4 01007 Bibb Cou~      22374          NA       39.9         1.2            51721
## 5 01009 Blount C~      57755          NA       41           0.5            48922
## 6 01011 Bullock ~      10173          NA       39.7         1.9            33866
## # ... with 7 more variables: B19013_001M <dbl>, per_capita_income <dbl>,
## #   B19301_001M <dbl>, poverty_count <dbl>, B17001_002M <dbl>,
## #   unemployment_count <dbl>, B23025_005M <dbl>
```

Data Cleaning for census_df:

Convert colnames to lowercase

```
names(census_df) <- tolower(names(census_df))

head(census_df)
```

```
## # A tibble: 6 x 14
##   geoid name      population b01003_001m median_age b01002_001m household_income
##   <chr> <chr>          <dbl>       <dbl>      <dbl>       <dbl>            <dbl>
## 1 01001 Autauga ~      55639          NA       38.6         0.6            57982
## 2 01003 Baldwin ~     218289          NA       43.2         0.4            61756
## 3 01005 Barbour ~      25026          NA       40.1         0.6            34990
## 4 01007 Bibb Cou~      22374          NA       39.9         1.2            51721
## 5 01009 Blount C~      57755          NA       41           0.5            48922
## 6 01011 Bullock ~      10173          NA       39.7         1.9            33866
## # ... with 7 more variables: b19013_001m <dbl>, per_capita_income <dbl>,
## #   b19301_001m <dbl>, poverty_count <dbl>, b17001_002m <dbl>,
## #   unemployment_count <dbl>, b23025_005m <dbl>
```

Dropping null rows

```
census_df <- na.omit(census_df) #  Remove NA
head(census_df)
```

```
## # A tibble: 6 x 14
##   geoid name      population b01003_001m median_age b01002_001m household_income
##   <chr> <chr>          <dbl>       <dbl>      <dbl>       <dbl>            <dbl>
## 1 49009 Daggett ~        590         157       38.9         6.1            74911
## 2 49031 Piute Co~       1870         157       53.4         7.7            29125
## 3 50009 Essex Co~       6179           3       51.4         0.7            47035
## 4 31005 Arthur C~        439          78       48           5.6            48500
## 5 31017 Brown Co~       2887         143       48.6         2.4            41979
## 6 31029 Chase Co~       3707         189       44.2         3.3            56135
## # ... with 7 more variables: b19013_001m <dbl>, per_capita_income <dbl>,
## #   b19301_001m <dbl>, poverty_count <dbl>, b17001_002m <dbl>,
## #   unemployment_count <dbl>, b23025_005m <dbl>
```

Selecting the column of interest

```
census_df <- select(census_df, name, population, median_age, household_income, per_capita_incom
e, poverty_count, unemployment_count)

str(census_df)
```

```
## tibble [157 x 7] (S3: tbl_df/tbl/data.frame)
##  $ name              : chr [1:157] "Daggett County, Utah" "Piute County, Utah" "Essex County,
Vermont" "Arthur County, Nebraska" ...
##  $ population        : num [1:157] 590 1870 6179 439 2887 ...
##  $ median_age        : num [1:157] 38.9 53.4 51.4 48 48.6 44.2 42.9 44.5 49.6 37.9 ...
##  $ household_income  : num [1:157] 74911 29125 47035 48500 41979 ...
##  $ per_capita_income : num [1:157] 27568 18148 27583 25277 29420 ...
##  $ poverty_count     : num [1:157] 17 351 870 55 275 325 209 237 161 115 ...
##  $ unemployment_count: num [1:157] 3 18 153 1 11 66 11 66 17 3 ...
##  - attr(*, "na.action")= 'omit' Named int [1:3064] 1 2 3 4 5 6 7 8 9 10 ...
##   ..- attr(*, "names")= chr [1:3064] "1" "2" "3" "4" ...
```

```
#sort descending
census_df[order(-census_df$household_income), ]
```

```
## # A tibble: 157 x 7
##    name    population median_age household_income per_capita_inco~ poverty_count
##    <chr>        <dbl>      <dbl>            <dbl>            <dbl>         <dbl>
##  1 Nassau~    1355683       41.9           120036            53363         72203
##  2 Suffol~    1481364       41.7           105362            46466         93999
##  3 Alpine~       1159       47.6            85750            37690           139
##  4 Maui C~     166657       41.8            84363            36872         14836
##  5 Wake C~    1091662       36.4            83567            42721         91083
##  6 Borden~        653       37.2            83281            33412            24
##  7 Arapah~     649980       36.8            80291            42184         49932
##  8 Bristo~        739       44.8            79808            46950            36
##  9 Shelby~     216350       39.5            78889            39711         14619
## 10 Steele~       1817       45.9            77167            38907           174
## # ... with 147 more rows, and 1 more variable: unemployment_count <dbl>
```

Now joining fips_df to census dataset

```
census_df <- inner_join(census_df, fips_df, by = "name") # Applying inner_join() function

head(census_df)
```

```
## # A tibble: 6 x 10
##   name     population median_age household_income per_capita_inco~ poverty_count
##   <chr>         <dbl>      <dbl>            <dbl>            <dbl>         <dbl>
## 1 Daggett~        590       38.9            74911            27568            17
## 2 Piute C~       1870       53.4            29125            18148           351
## 3 Essex C~       6179       51.4            47035            27583           870
## 4 Arthur ~        439       48              48500            25277            55
## 5 Brown C~       2887       48.6            41979            29420           275
## 6 Chase C~       3707       44.2            56135            30850           325
## # ... with 4 more variables: unemployment_count <dbl>, fips_code <chr>,
## #   county <chr>, state <chr>
```

Split the name into two

```
#census_df %>% separate(name, c("county","state"), sep = " County,")
census_df <- select(census_df,-c(name))
```

Removing name column

```
colnames(census_df)<-gsub(".x","",colnames(census_df))
head(census_df)
```

```
## # A tibble: 6 x 9
##   population median_age household_income per_capita_income poverty_count
##        <dbl>      <dbl>            <dbl>              <dbl>         <dbl>
## 1        590       38.9            74911              27568            17
## 2       1870       53.4            29125              18148           351
## 3       6179       51.4            47035              27583           870
## 4        439       48              48500              25277            55
## 5       2887       48.6            41979              29420           275
## 6       3707       44.2            56135              30850           325
## # ... with 4 more variables: unemployment_count <dbl>, fips_code <chr>,
## #   county <chr>, state <chr>
```

Removing string ‘county’ from column values

```
census_df$county<-gsub(" County","",as.character(census_df$county))
head(census_df)
```

```
## # A tibble: 6 x 9
##   population median_age household_income per_capita_income poverty_count
##        <dbl>      <dbl>            <dbl>              <dbl>         <dbl>
## 1        590       38.9            74911              27568            17
## 2       1870       53.4            29125              18148           351
## 3       6179       51.4            47035              27583           870
## 4        439       48              48500              25277            55
## 5       2887       48.6            41979              29420           275
## 6       3707       44.2            56135              30850           325
## # ... with 4 more variables: unemployment_count <dbl>, fips_code <chr>,
## #   county <chr>, state <chr>
```

```
# converting character type
# column to numeric
census_df <- transform(census_df,
                       fips_code = as.numeric(fips_code))
```

Checking the working directory path and exporting data frame into

a CSV file

```
# Get working directory path
path <- getwd()

path
```

```
## [1] "C:/Users/CUNY_SPS_PROJECTS/Data_607_Final_Project"
```

Export file as csv to working directory.

```
write.csv(census_df, file.path(path, "/posgresql/census_data.csv"))
```

Fetching the education data from GitHub

```
education_df <- read_csv("https://raw.githubusercontent.com/Shriyanshh/Data-607---Final-Project/
refs/heads/main/ers_usda_education.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double(),
##   State = col_character(),
##   `Area name` = col_character(),
##   `Less than a high school diploma, 1970` = col_number(),
##   `High school diploma only, 1970` = col_number(),
##   `Some college (1-3 years), 1970` = col_number(),
##   `Four years of college or higher, 1970` = col_number(),
##   `Less than a high school diploma, 1980` = col_number(),
##   `High school diploma only, 1980` = col_number(),
##   `Some college (1-3 years), 1980` = col_number(),
##   `Four years of college or higher, 1980` = col_number(),
##   `Less than a high school diploma, 1990` = col_number(),
##   `High school diploma only, 1990` = col_number(),
##   `Some college or associate's degree, 1990` = col_number(),
##   `Bachelor's degree or higher, 1990` = col_number(),
##   `Less than a high school diploma, 2000` = col_number(),
##   `High school diploma only, 2000` = col_number(),
##   `Some college or associate's degree, 2000` = col_number(),
##   `Bachelor's degree or higher, 2000` = col_number(),
##   `Less than a high school diploma, 2015-19` = col_number(),
##   `High school diploma only, 2015-19` = col_number()
##   # ... with 2 more columns
## )
## i Use `spec()` for the full column specifications.
```

```
#head(education_df)
```

```
#colnames(education_df)
```

```
education_df <- education_df[-c(2:40)]
education_df <- education_df[-c(1),]

#head(education_df)
```

```
education_df <- rename_with(education_df, ~ tolower(gsub(",", "", .x, fixed = TRUE)))
education_df <- rename_with(education_df, ~ tolower(gsub("'s", "", .x, fixed = TRUE)))
education_df <- rename_with(education_df, ~ tolower(gsub(" ", "_", .x, fixed = TRUE)))
education_df <- rename_with(education_df, ~ tolower(gsub("-", "_", .x, fixed = TRUE)))
head(education_df)
```

```
## # A tibble: 6 x 8
##   fips_code high_school_dipl~ some_college_or~ bachelor_degree~ percent_of_adul~
##       <dbl>             <dbl>            <dbl>            <dbl>            <dbl>
## 1      1000           1022839           993344           845772             13.8
## 2      1001             12551            10596             9929             11.5
## 3      1003             41797            47274            48148              9.2
## 4      1005              6396             4676             2080             26.8
## 5      1007              7256             3848             1678             20.9
## 6      1009             13299            13519             5210             19.5
## # ... with 3 more variables:
## #   percent_of_adults_with_a_high_school_diploma_only_2015_19 <dbl>,
## #   percent_of_adults_completing_some_college_or_associate_degree_2015_19 <dbl>,
## #   percent_of_adults_with_a_bachelor_degree_or_higher_2015_19 <dbl>
```

```
education_df <- na.omit(education_df) # Method 1 - Remove NA
#head(education_df)
```

```
###str(education_df)
```

Checking the working directory path and exporting data frame into a CSV file

```
# Get working directory path
path <- getwd()

path
```

```
## [1] "C:/Users/CUNY_SPS_PROJECTS/Data_607_Final_Project"
```

Export file as csv to working directory.

```
write.csv(education_df, file.path(path, "/posgresql/education_data.csv"))
```

```
census_df$fips_code <- as.integer(census_df$fips_code)
education_df$fips_code <- as.integer(education_df$fips_code)
```

```
str(census_df)
```

```
## 'data.frame':    147 obs. of  9 variables:
##  $ population        : num  590 1870 6179 439 2887 ...
##  $ median_age        : num  38.9 53.4 51.4 48 48.6 44.2 42.9 44.5 49.6 37.9 ...
##  $ household_income  : num  74911 29125 47035 48500 41979 ...
##  $ per_capita_income : num  27568 18148 27583 25277 29420 ...
##  $ poverty_count     : num  17 351 870 55 275 325 209 237 161 115 ...
##  $ unemployment_count: num  3 18 153 1 11 66 11 66 17 3 ...
##  $ fips_code         : int  49009 49031 50009 31005 31017 31029 31057 31063 31069 31075 ...
##  $ county            : chr  "Daggett" "Piute" "Essex" "Arthur" ...
##  $ state             : chr  "Utah" "Utah" "Vermont" "Nebraska" ...
```

```
#str(education_df)

#dim(census_df)
#dim(education_df)
```

```
new_df <- census_df %>% inner_join(education_df, by="fips_code")
       # Join by multiple columns

nrow(new_df)
```

```
## [1] 147
```

```
new_df <- new_df %>%
  relocate(fips_code, .before=county)
new_df <-new_df %>%
  relocate(state, .after=county)
head(new_df)
```

```
##   population median_age household_income per_capita_income poverty_count
## 1        590       38.9            74911              27568            17
## 2       1870       53.4            29125              18148           351
## 3       6179       51.4            47035              27583           870
## 4        439       48.0            48500              25277            55
## 5       2887       48.6            41979              29420           275
## 6       3707       44.2            56135              30850           325
##   unemployment_count fips_code  county    state
## 1                  3     49009 Daggett     Utah
## 2                 18     49031   Piute     Utah
## 3                153     50009   Essex  Vermont
## 4                  1     31005  Arthur Nebraska
## 5                 11     31017   Brown Nebraska
## 6                 66     31029   Chase Nebraska
##   high_school_diploma_only_2015_19 some_college_or_associate_degree_2015_19
## 1                              161                                      163
## 2                              527                                      393
## 3                             2175                                     1167
## 4                               74                                      119
## 5                              914                                      752
## 6                              689                                     1105
##   bachelor_degree_or_higher_2015_19
## 1                                54
## 2                               263
## 3                               766
## 4                                75
## 5                               463
## 6                               562
##   percent_of_adults_with_less_than_a_high_school_diploma_2015_19
## 1                                                            6.0
## 2                                                            9.0
## 3                                                           13.2
## 4                                                            6.3
## 5                                                            4.9
## 6                                                           11.9
##   percent_of_adults_with_a_high_school_diploma_only_2015_19
## 1                                                      40.0
## 2                                                      40.5
## 3                                                      46.0
## 4                                                      25.9
## 5                                                      40.8
## 6                                                      25.8
##   percent_of_adults_completing_some_college_or_associate_degree_2015_19
## 1                                                                  40.5
## 2                                                                  30.2
## 3                                                                  24.7
## 4                                                                  41.6
## 5                                                                  33.6
## 6                                                                  41.3
##   percent_of_adults_with_a_bachelor_degree_or_higher_2015_19
## 1                                                       13.4
## 2                                                       20.2
```

```
## 3                                                    16.2
## 4                                                    26.2
## 5                                                    20.7
## 6                                                    21.0
```

```
new_df$name = paste(new_df$county, new_df$state, sep=", ")
new_df <- new_df %>%
  relocate(name, .before=county)
#head(new_df)
```

To calculate the percentage of poverty_count and unemployment_count

```
new_df <- mutate(new_df, percent_poverty_rate = (poverty_count/population)*100)
new_df <- mutate(new_df, percent_unemployment_rate = (unemployment_count/population)*100)
#head(new_df)
```

Assumptions for performing linear regression:

Before applying the linear regression model, the following four assumption must be made. And it is important to visualize the data based on these assumption.

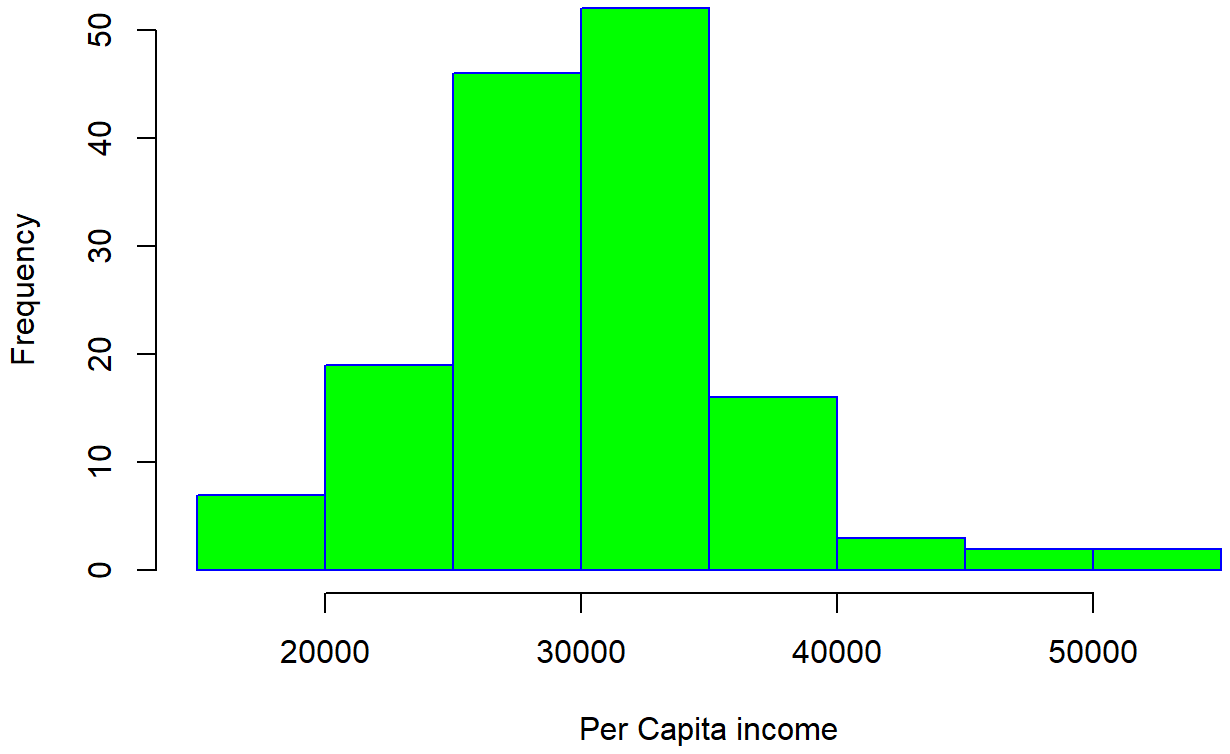
1.Independence of observations (aka no autocorrelation):

Since there is only one independent variable and one dependent variable, we don't need to test for any hidden relationships among variables.

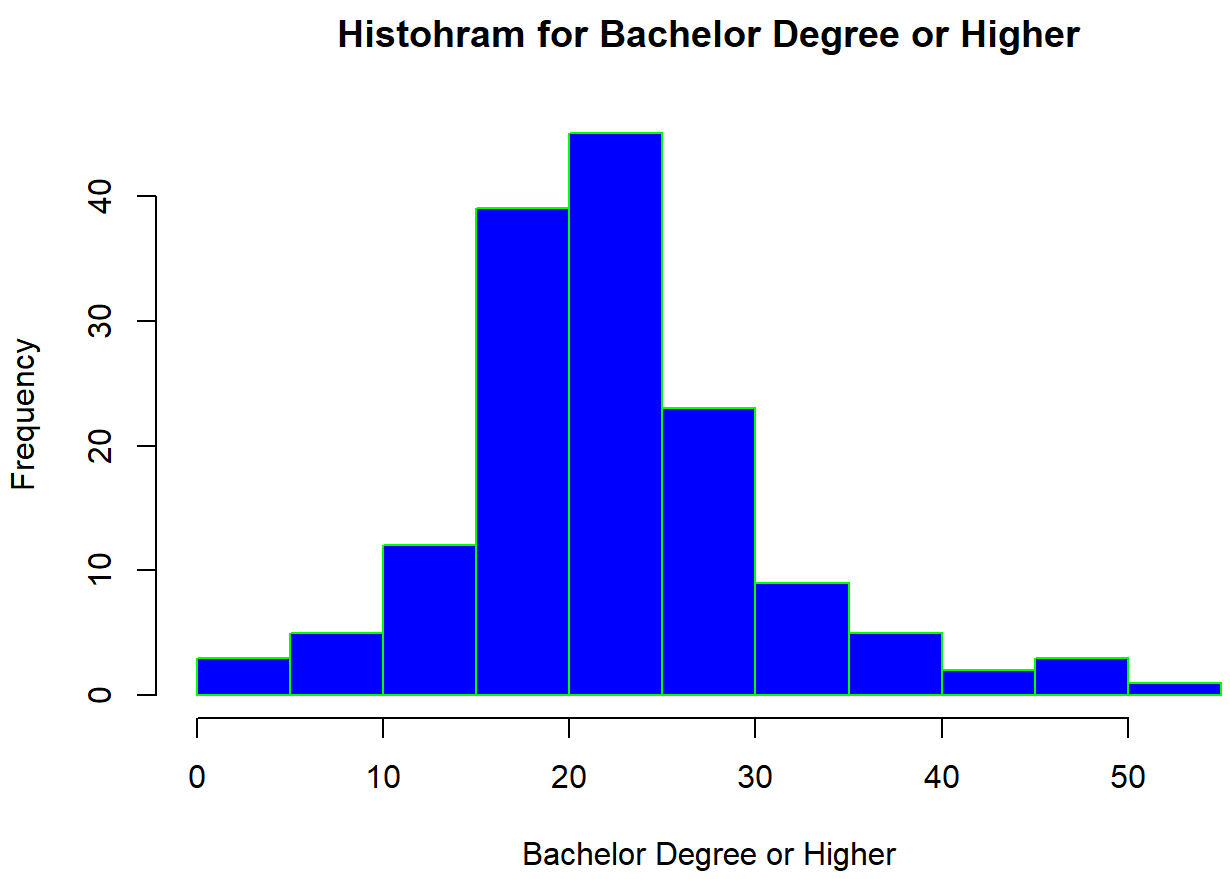
2. Normality:

The normality can be checked by using the hist() function, which tells whether the dependent variable follows a normal distribution or not.

```
hist(new_df$per_capita_income,
     main = "Histohram for Per Capita income",
     xlab = "Per Capita income",
     col = "green",
     border="blue"
     )
```




```
hist(new_df$percent_of_adults_with_a_bachelor_degree_or_higher_2015_19,
     main = "Histohram for Bachelor Degree or Higher",
     xlab = "Bachelor Degree or Higher",
     col = "blue",
     border="green")
```

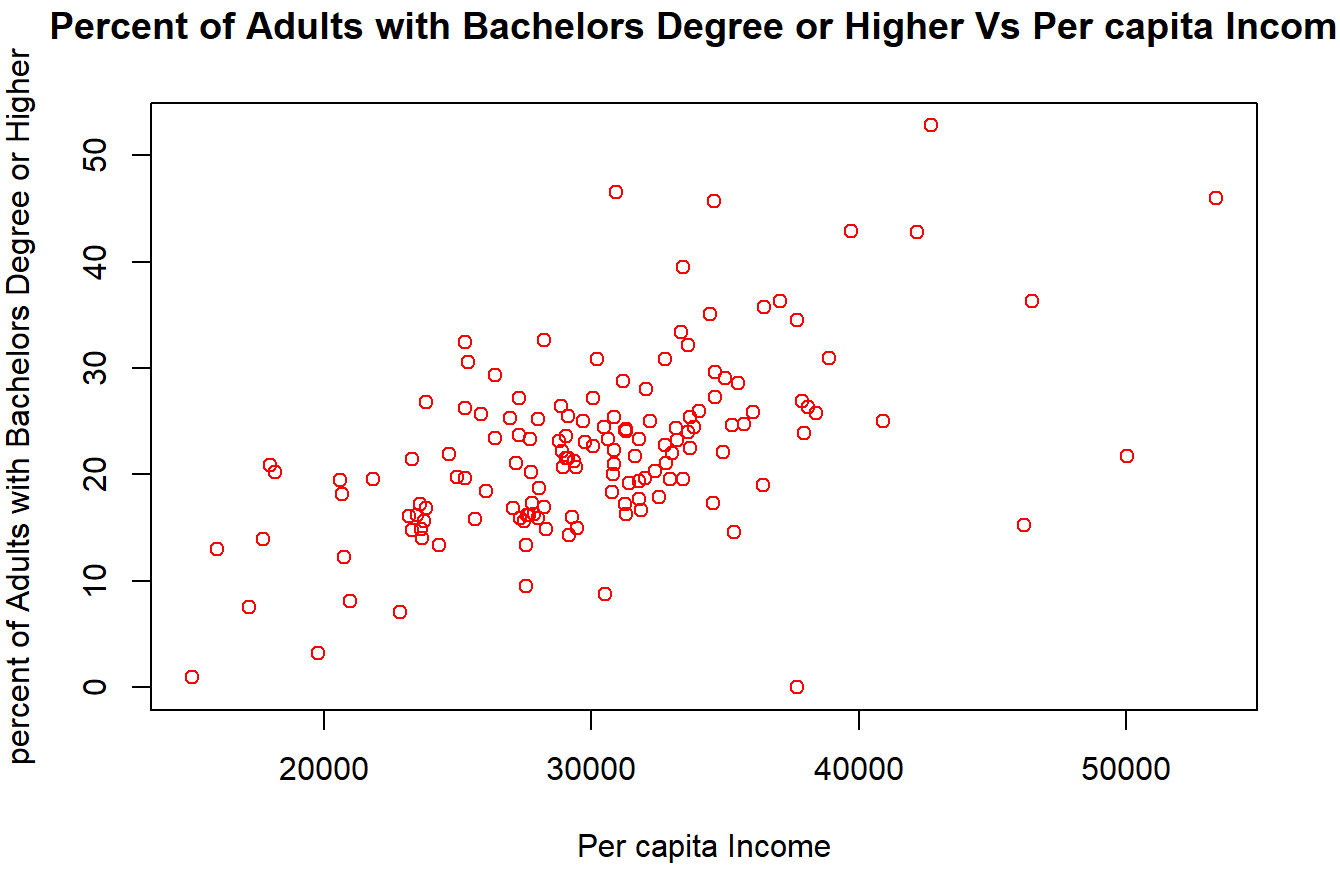


The histograms show that our data follows the normal distribution.

3- Linearity:

The relationship between the independent and dependent variable must be linear. We can test this visually with a scatter plot to see if the distribution of data points could be described with a straight line.

```
plot(percent_of_adults_with_a_bachelor_degree_or_higher_2015_19 ~ per_capita_income, data = new_
df, main = "Percent of Adults with Bachelors Degree or Higher Vs Per capita Income",
     xlab = "Per capita Income",
     ylab = "percent of Adults with Bachelors Degree or Higher",
     col = "red")
```



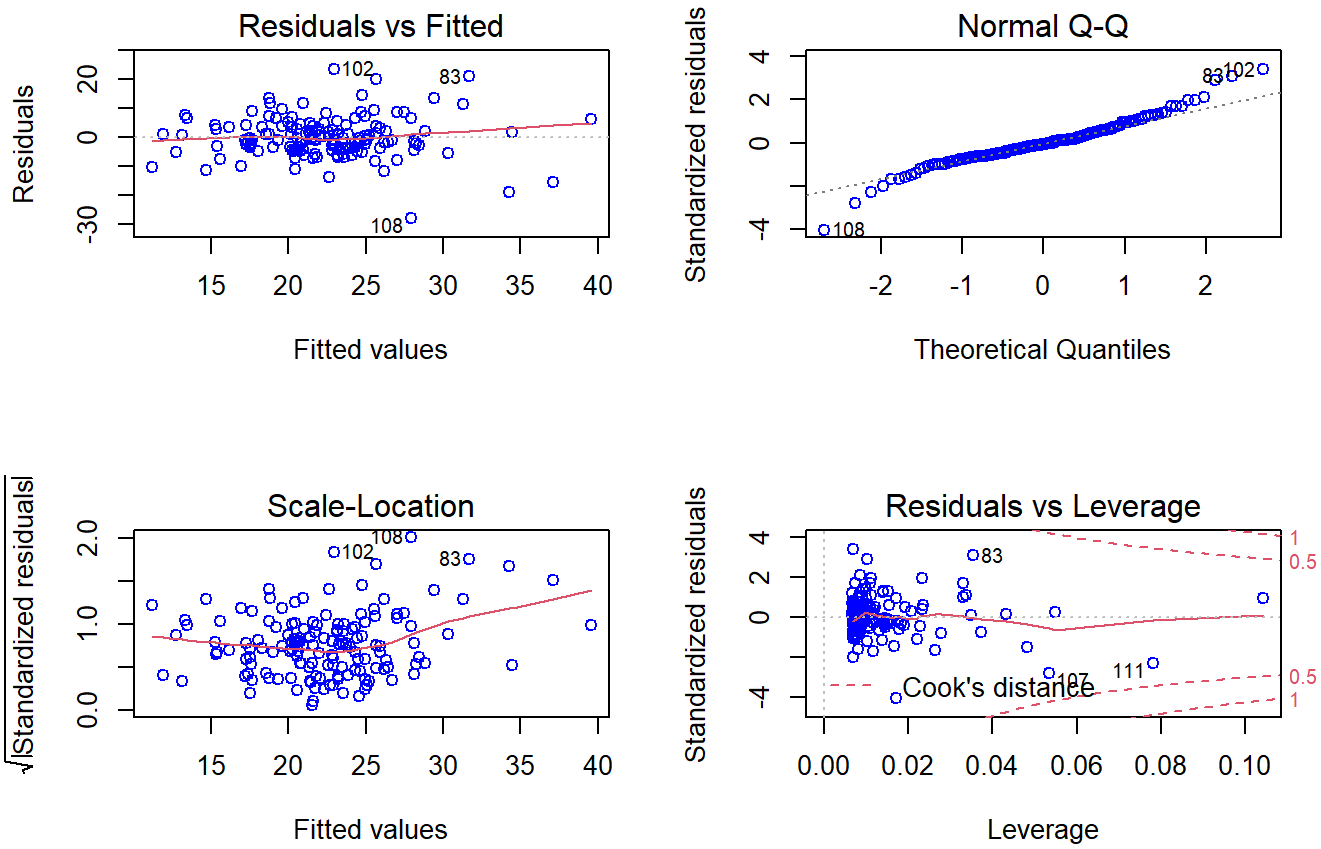
This scatter plot is showing a linear relationship between x and y variables.

4- Homoscedasticity (aka homogeneity of variance):

This means that the prediction error doesn't change significantly over the range of prediction of the model.

The residual plots:

```
model <- lm(percent_of_adults_with_a_bachelor_degree_or_higher_2015_19 ~ per_capita_income, data
= new_df)
par(mfrow=c(2,2))
plot(model, col = "blue")
```



The red lines representing the mean of the residuals are all basically horizontal and centered around zero. This means there are no outliers or biases in the data that would make a linear regression invalid.

In the Normal Q-Qplot in the top right, we can see that the real residuals from our model form an almost perfectly one-to-one line with the theoretical residuals from a perfect model.

Based on these residuals, we can say that our model meets the assumption of homoscedasticity.

and the appropriate visualizations helps us better understand the data.

Simple Linear Regression Model:

Regression is a supervised learning algorithm which helps in determining how does one variable influence another variable.

Simple linear regression is useful for modelling the relationship between a numeric or dependent variable (Y) and multiple explanatory or independent variable (X).

The dependent variable (Y) here is ‘percent_of_adults_with_a_bachelor_degree_or_higher_2015_19’ and the independent variable (X) is per_capita_income.

Simple Linear Regression Equation:

percent_of_adults_with_a_bachelor_degree_or_higher_2015_19 = b0 + b4*per_capita_income

Compute the summary of the model:

```
model <- lm(percent_of_adults_with_a_bachelor_degree_or_higher_2015_19 ~ per_capita_income, data
= new_df)
summary(model)
```

```
##
## Call:
## lm(formula = percent_of_adults_with_a_bachelor_degree_or_higher_2015_19 ~
##     per_capita_income, data = new_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -27.9469  -4.0576  -0.5654   3.5357  23.5879
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -2.294e-02  2.886e+00  -0.008    0.994
## per_capita_income  7.420e-04  9.374e-05   7.916 5.83e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.967 on 145 degrees of freedom
## Multiple R-squared:  0.3017, Adjusted R-squared:  0.2969
## F-statistic: 62.66 on 1 and 145 DF,  p-value: 5.829e-13
```

The F-Test of overall significance has the following two hypotheses:

Null hypothesis (H0) : The model with no predictor variables (also known as an intercept-only model) fits the data as well as the regression model defined here.

Alternative hypothesis (HA) : Your regression model fits the data better than the intercept-only model.

Interpretation:

A large F-statistic will correspond to a statistically significant p-value ($p < 0.05$). In our example, the F-statistic equal 62.7 producing a p-value: 5.83e-13, which is highly significant. This means that, the predictor variables is significantly related to the outcome variable.

And we accept the alternate hypothesis that our regression model fits the data better than the intercept-only model.

The coefficients table shows the estimate of regression beta coefficients and the associated t-statistics p-values:

```
summary(model)$coefficient
```

```
##                         Estimate   Std. Error      t value     Pr(>|t|)
## (Intercept)       -0.0229359887 2.886142e+00 -0.007946937 9.936703e-01
## per_capita_income  0.0007420436 9.374395e-05  7.915642488 5.829052e-13
```

For a given predictor, the t-statistic evaluates whether or not there is significant association between the predictor and the outcome variable, that is whether the beta coefficient of the predictor is significantly different from zero.

we can see, changing the per_capita_income variable is significantly associated to changes in percent_of_adults_with_a_bachelor_degree_or_higher_2015_19

For a given predictor variable, the coefficient (b) can be interpreted as the average effect on y of a one unit increase in predictor, holding all other predictors fixed.

One unit increase in per_capita_income will significantly increase percent_of_adults_with_a_bachelor_degree_or_higher_2015_19, holding all other predictors fixed.

The confidence interval of the model coefficient can be:

```
confint(model)
```

```
##                           2.5 %       97.5 %
## (Intercept)       -5.7272787106 5.6814067332
## per_capita_income  0.0005567625 0.0009273247
```

Model accuracy assessment:

R-squared:

The value of R will always be positive and will range from zero to one. An R2 value close to 1 indicates that the model explains a large portion of the variance in the outcome variable.

The R2 = 0.302, meaning that “30% of the variance in the measure of percent_of_adults_with_a_bachelor_degree_or_higher_2015_19 can be predicted by coefficient predictor.

Residual Standard Error (RSE), or sigma:

The RSE estimate gives a measure of error of prediction. The lower the RSE, the more accurate the model (on the data in hand).

The error rate can be estimated by dividing the RSE by the mean outcome variable:

```
sigma(model)/mean(new_df$percent_of_adults_with_a_bachelor_degree_or_higher_2015_19)
```

```
## [1] 0.3115227
```

The correlation coefficient between the two variables using the R

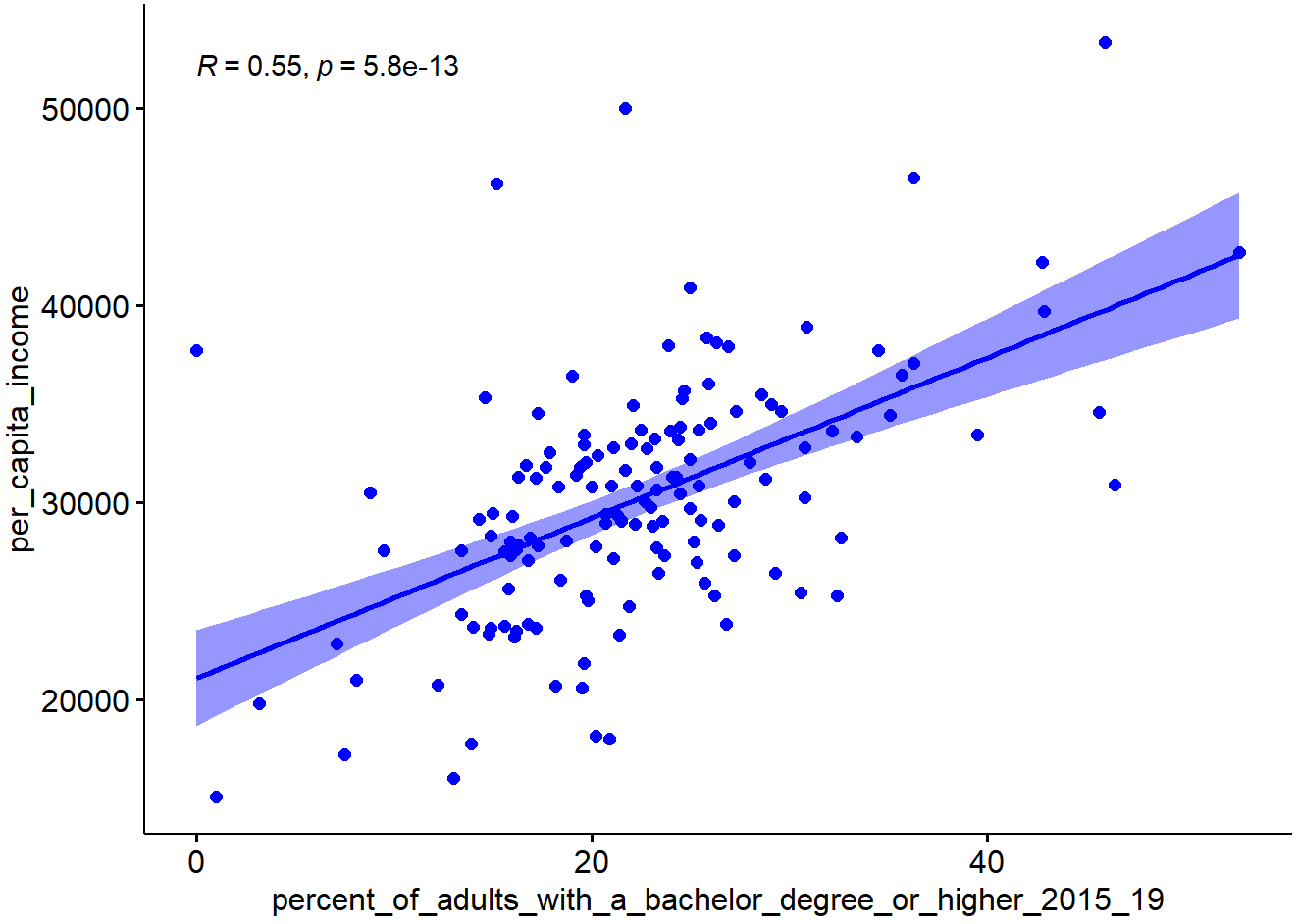
function cor():

```
cor(new_df$percent_of_adults_with_a_bachelor_degree_or_higher_2015_19, new_df$per_capita_income,
method="pearson")
```

```
## [1] 0.5493036
```

```
ggscatter(new_df, x = "percent_of_adults_with_a_bachelor_degree_or_higher_2015_19", y = "per_cap
ita_income",
          add = "reg.line", conf.int = TRUE,
          cor.coef = TRUE, cor.method = "pearson",
          xlab = "percent_of_adults_with_a_bachelor_degree_or_higher_2015_19", ylab = "per_capit
a_income", col = "blue")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Using ggPredict() - Visualize Regression Model

We can show this model with ggPredict() function and adjust the

number of regression lines with parameter colorn.

```
ggPredict(model,interactive=TRUE,colorn=10,jitter=FALSE)
```