

```
#This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.
```

```
#Try executing this chunk by clicking the Run button within the chunk or by placing your cursor inside it and pressing Cmd+Shift+Enter.
```

```
adult_df <- read.csv("adult.csv")
summary(adult_df)
```

```
##      age      workclass      fnlwgt      education
##  Min.   :17.00  Length:32561   Min.   : 12285  Length:32561
##  1st Qu.:28.00  Class  :character  1st Qu.: 117827 Class  :character
##  Median :37.00  Mode   :character  Median : 178356 Mode   :character
##  Mean   :38.58
##  3rd Qu.:48.00
##  Max.   :90.00
##      education.num    marital.status      occupation      relationship
##  Min.   : 1.00  Length:32561   Length:32561  Length:32561
##  1st Qu.: 9.00  Class  :character  Class  :character Class  :character
##  Median :10.00  Mode   :character  Mode   :character Mode   :character
##  Mean   :10.08
##  3rd Qu.:12.00
##  Max.   :16.00
##      race          sex      capital.gain      capital.loss
##  Length:32561  Length:32561   Min.   : 0  Min.   : 0.0
##  Class  :character  Class  :character  1st Qu.: 0  1st Qu.: 0.0
##  Mode   :character  Mode   :character  Median : 0  Median : 0.0
##                      Mean   :1078  Mean   : 87.3
##                      3rd Qu.: 0  3rd Qu.: 0.0
##                      Max.   :99999 Max.   :4356.0
##      hours.per.week native.country      income.bracket
##  Min.   : 1.00  Length:32561   Length:32561
##  1st Qu.:40.00  Class  :character  Class  :character
##  Median :40.00  Mode   :character  Mode   :character
##  Mean   :40.44
##  3rd Qu.:45.00
##  Max.   :99.00
##
## I'm comparing 'age' and 'education.num' because age and education often show interesting patterns.**
## Older people might have different education levels compared to younger ones. This can tell us how
## education trends have changed over time. It's a quick way to spot if there's a shift in education levels across
## different age groups.**
```

Compare 'age' and 'education.num'

```
summary(adult_df$age)
```

```
##      Min. 1st Qu.  Median  Mean 3rd Qu.  Max.
##  17.00  28.00  37.00  38.58  48.00  90.00
```

```
summary(adult_df$education.num)
```

```
##      Min. 1st Qu.  Median  Mean 3rd Qu.  Max.
##  1.00   9.00  10.00  10.08  12.00  16.00
```

Problem 1a. First, we look at the summary statistics for all the variables. Based on those metrics, including

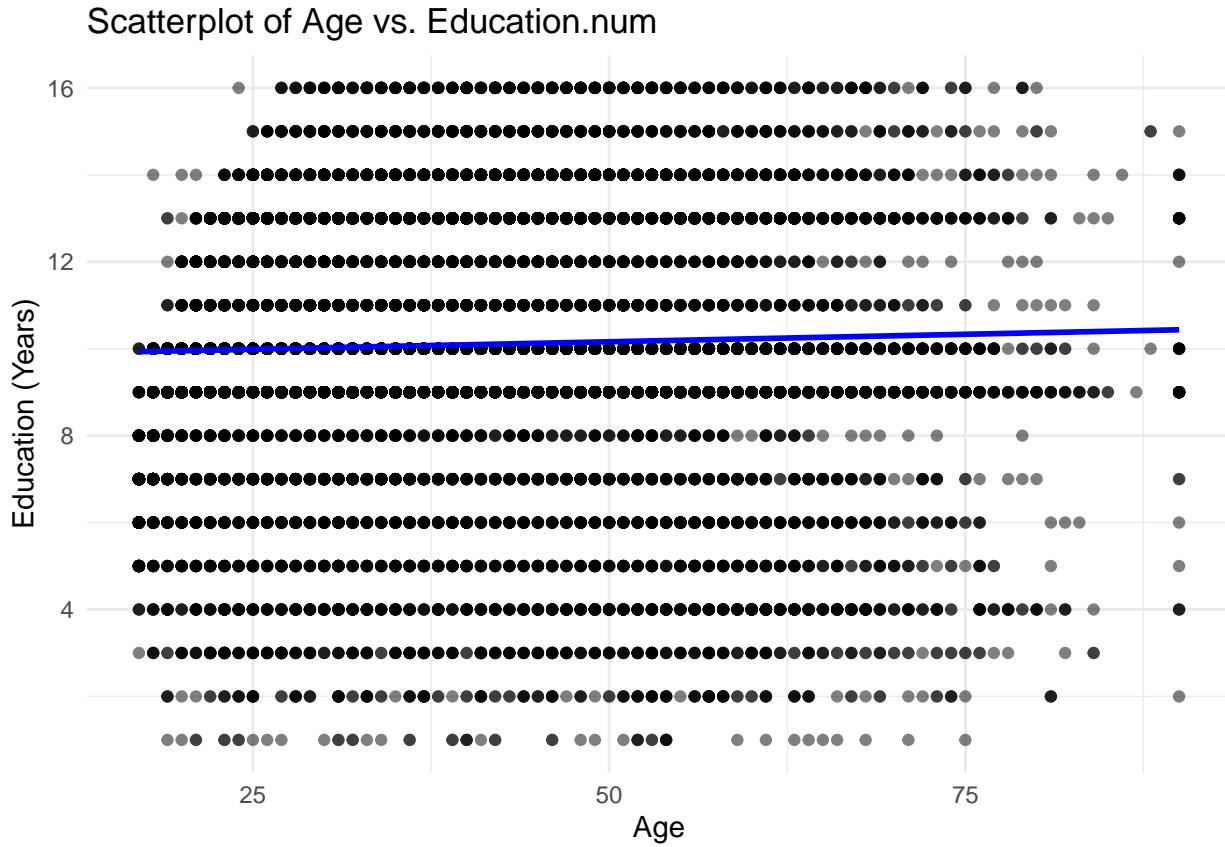
the quartiles, compare two variables. What can you tell about their shape from these summaries?

Ans. The age distribution is right-skewed, meaning most people are younger, but a few older individuals pull the mean higher. The education.num distribution is more symmetrical, with most people having (around 9-12 years) of education, and there are no extreme outliers affecting the mean.

Problem 1b. Use a visualization to get a fine-grain comparison (you don't have to use QQ plots, though) of the distributions of those two variables. Why did you choose the type of visualization that you chose? How do your part (a) assumptions compare to what you can see visually?

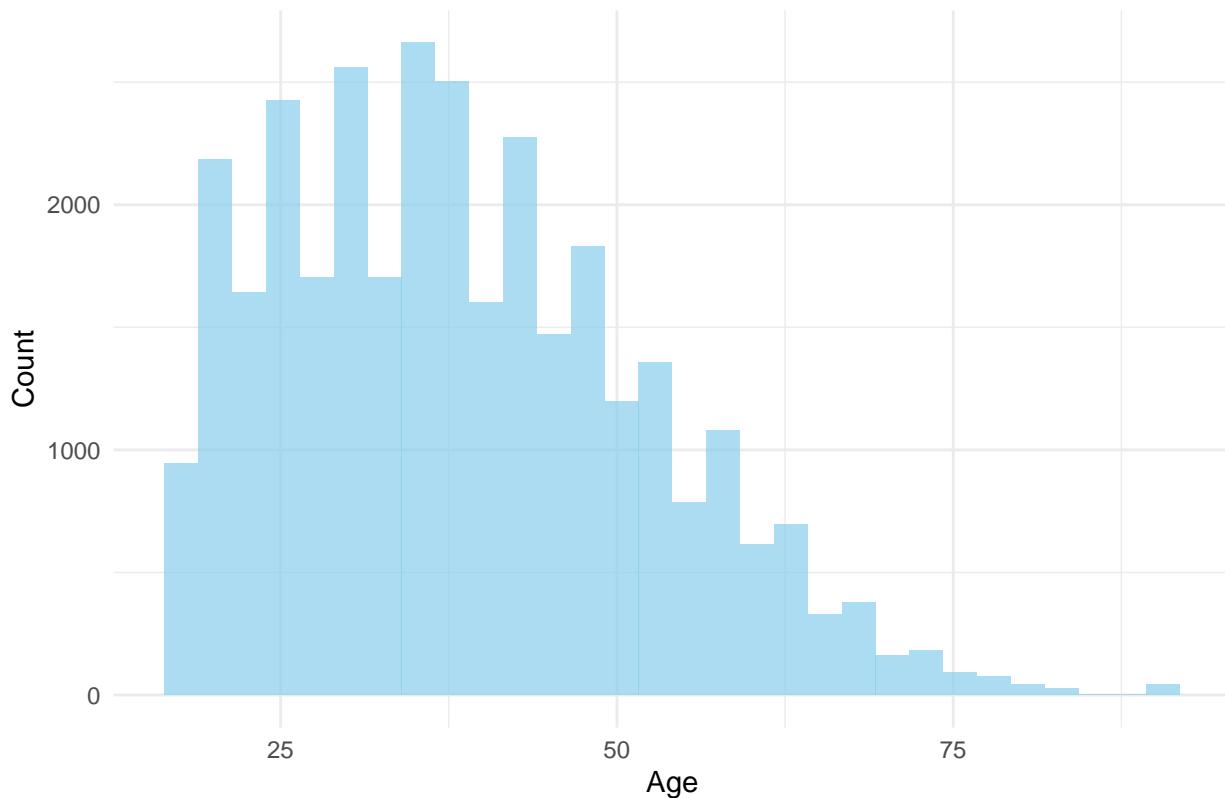
```
library(ggplot2)
ggplot(adult_df, aes(x = age, y = education.num)) +
  geom_point(alpha = 0.5) + # Scatterplot points
  geom_smooth(method = "lm", col = "blue", se = FALSE) + # Add a linear regression line
  labs(title = "Scatterplot of Age vs. Education.num",
       x = "Age", y = "Education (Years)") +
  theme_minimal()

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



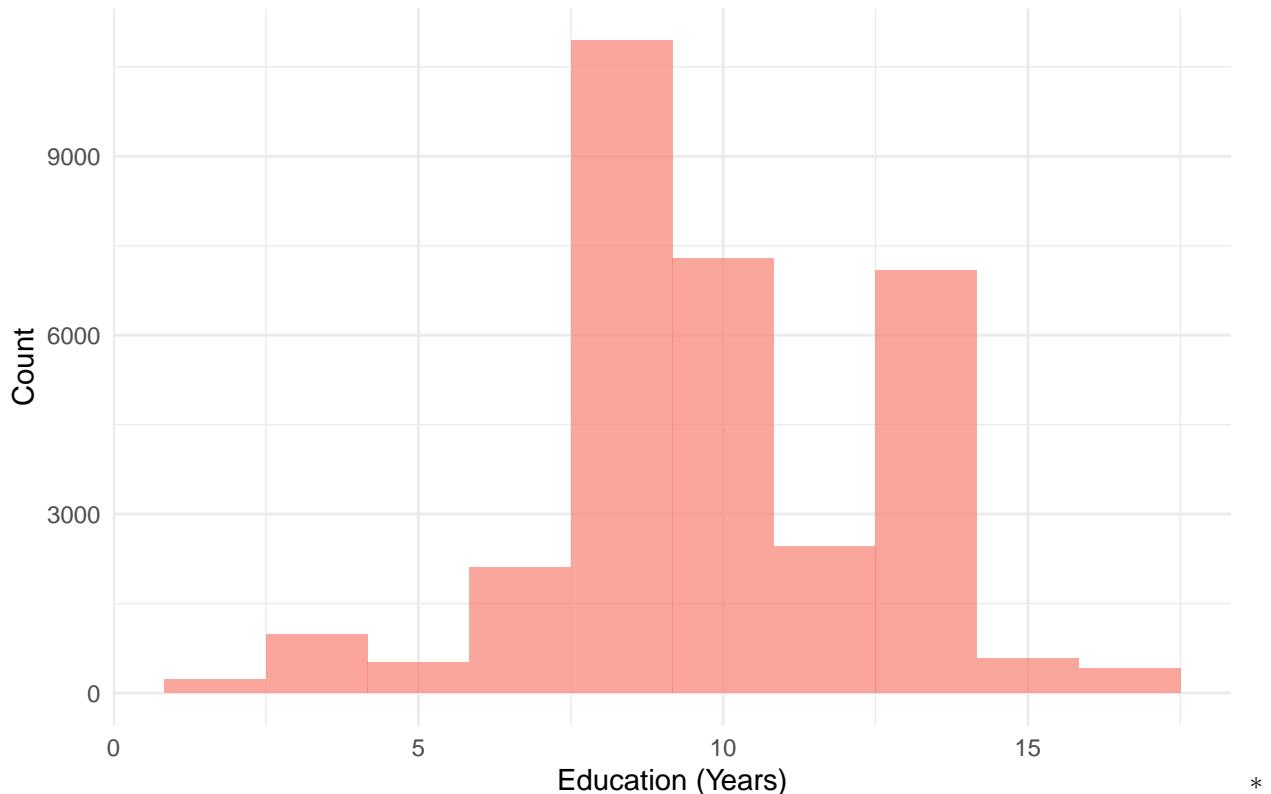
```
# Age Distribution
ggplot(adult_df, aes(x = age)) +
  geom_histogram(bins = 30, fill = "skyblue", alpha = 0.7) +
  labs(title = "Age Distribution", x = "Age", y = "Count") +
  theme_minimal()
```

Age Distribution



```
# Education.num Distribution
ggplot(adult_df, aes(x = education.num)) +
  geom_histogram(bins = 10, fill = "salmon", alpha = 0.7) +
  labs(title = "Education (Years) Distribution", x = "Education (Years)", y = "Count") +
  theme_minimal()
```

Education (Years) Distribution



Explanation for choosing the Visualization: - I chose the scatterplot to easily see how education varies with age. The trend line gives an overall sense of whether the relationship is positive or negative. - I added histograms to compare the distribution of each variable individually. This helps visualize skewness, spread, and the concentration of data points for each variable.

- Comparison to Part (a) Assumptions:
- The scatterplot confirms that there is no strong linear relationship between age and education, but we can see some grouping (younger individuals tend to have lower or middle levels of education).
- The histograms visually confirm the assumption from Part (a). The age distribution is right-skewed, and the education.num distribution is more symmetrical, with most individuals having between 9 and 12 years of education.

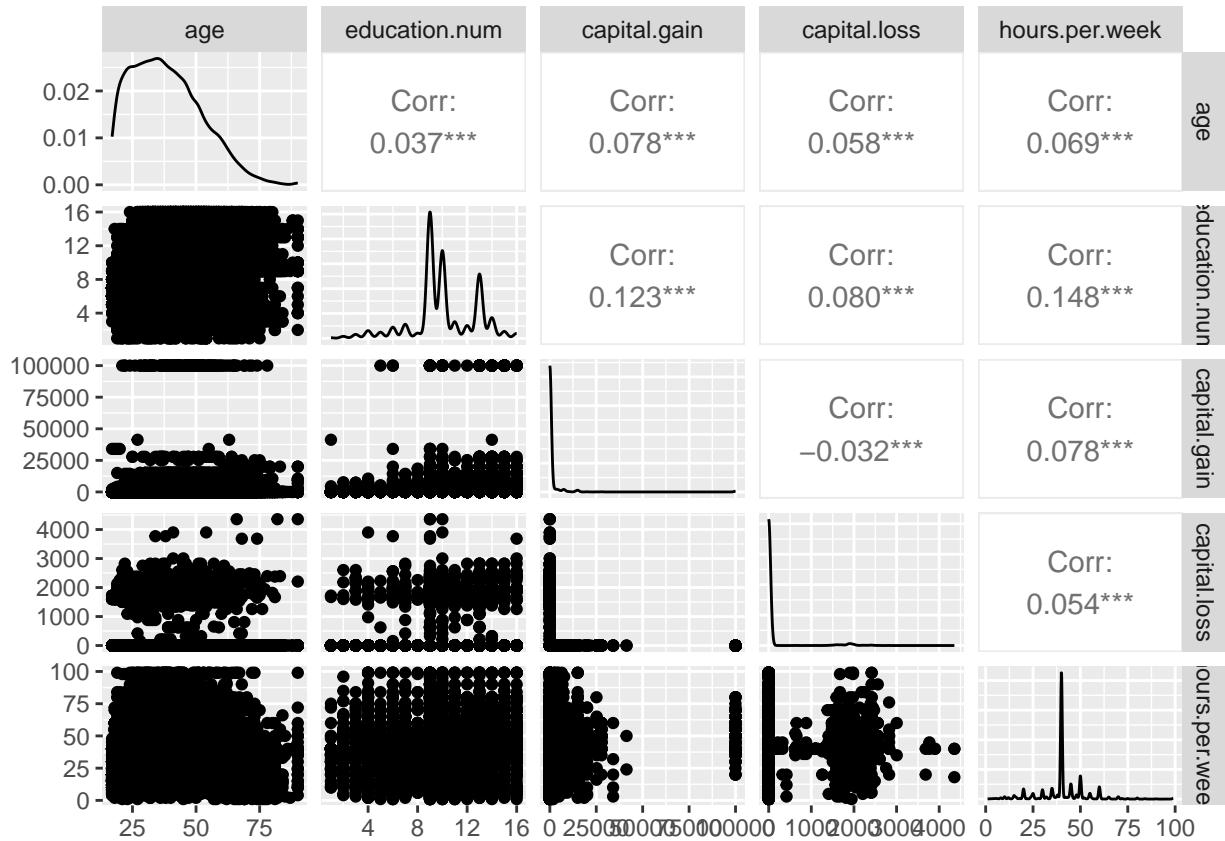
Problem 1c. Now create a scatterplot matrix of the numerical variables. What does this view show you that would be difficult to see looking at distributions?

```
library(GGally)

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2

numeric_columns <- adult_df[, c("age", "education.num", "capital.gain", "capital.loss", "hours.per.week")]

# Create a scatter-plot matrix
ggpairs(numeric_columns)
```



What This View Shows? - A scatterplot matrix shows pairwise relationships between each pair of numerical variables. - We can visually detect correlations between variables. For example: - If two variables tend to form a straight line, they are likely correlated. - If the scatter is spread out randomly, there is little to no correlation. - We can also spot outliers and general trends between pairs of variables.

What Is Hard to See in Distributions:

- Distributions only show how data is spread for one variable at a time.
- Scatterplot matrices, on the other hand, let you see how two variables relate to each other at the same time (e.g., how age and education are related), which is not visible when just looking at histograms or single-variable distributions.

Problem 1d. These data are a selection of US adults. It might not be a very balanced sample, though. Take a look at some categorical variables and see if any have a lot more of one category than others. There are many ways to do this, including histograms and following `tidyverse group_by` with `count`. I recommend you try a few for practice.

```
library(dplyr)
```

```
##  
## Attaching package: 'dplyr'  
  
## The following objects are masked from 'package:stats':  
##  
##     filter, lag  
  
## The following objects are masked from 'package:base':  
##  
##     intersect, setdiff, setequal, union
```

```

adult_df %>%
  group_by(workclass) %>%
  summarise(count = n()) %>%
  arrange(desc(count))

## # A tibble: 9 x 2
##   workclass      count
##   <chr>        <int>
## 1 " Private"    22696
## 2 " Self-emp-not-inc" 2541
## 3 " Local-gov"  2093
## 4 " ?"          1836
## 5 " State-gov" 1298
## 6 " Self-emp-inc" 1116
## 7 " Federal-gov" 960
## 8 " Without-pay" 14
## 9 " Never-worked" 7

adult_df %>%
  group_by(race) %>%
  summarise(count = n()) %>%
  arrange(desc(count))

## # A tibble: 5 x 2
##   race      count
##   <chr>     <int>
## 1 " White"  27816
## 2 " Black"   3124
## 3 " Asian-Pac-Islander" 1039
## 4 " Amer-Indian-Eskimo"  311
## 5 " Other"   271

adult_df %>%
  group_by(education) %>%
  summarise(count = n()) %>%
  arrange(desc(count))

## # A tibble: 16 x 2
##   education      count
##   <chr>        <int>
## 1 " HS-grad"   10501
## 2 " Some-college" 7291
## 3 " Bachelors"  5355
## 4 " Masters"    1723
## 5 " Assoc-voc"  1382
## 6 " 11th"       1175
## 7 " Assoc-acdm" 1067
## 8 " 10th"       933
## 9 " 7th-8th"    646
## 10 " Prof-school" 576
## 11 " 9th"        514
## 12 " 12th"       433
## 13 " Doctorate" 413
## 14 " 5th-6th"    333
## 15 " 1st-4th"   168

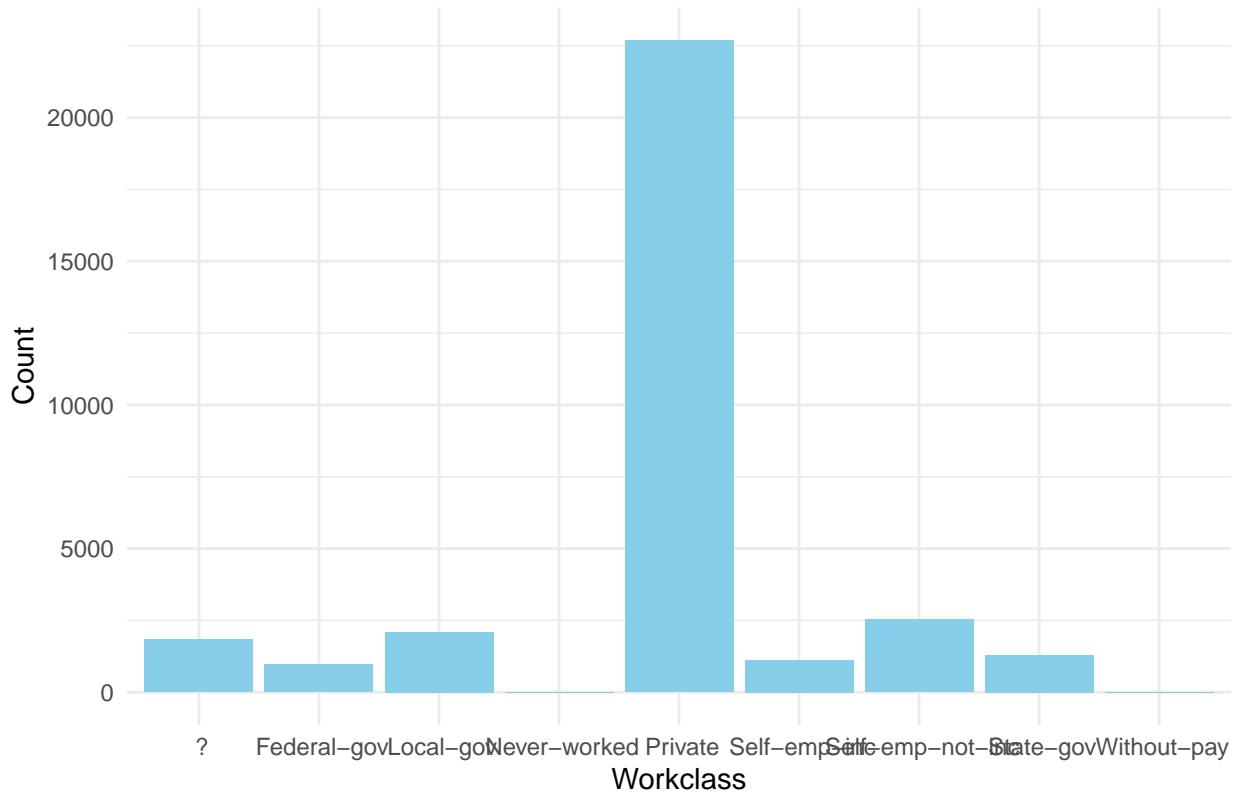
```

```

## 16 " Preschool"      51
# Visualizing the distribution of 'workclass'
ggplot(adult_df, aes(x = workclass)) +
  geom_bar(fill = "skyblue") +
  labs(title = "Distribution of Workclass", x = "Workclass", y = "Count") +
  theme_minimal()

```

Distribution of Workclass

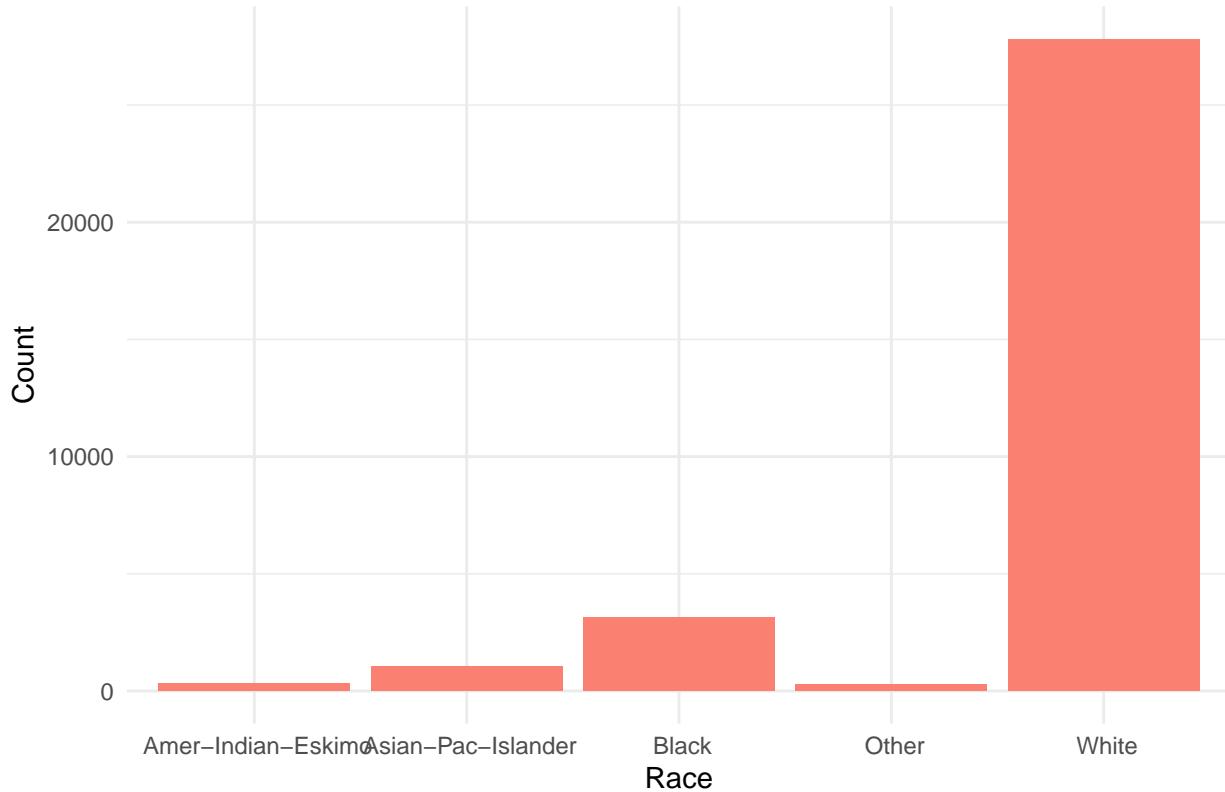


```

# Visualizing the distribution of 'race'
ggplot(adult_df, aes(x = race)) +
  geom_bar(fill = "salmon") +
  labs(title = "Distribution of Race", x = "Race", y = "Count") +
  theme_minimal()

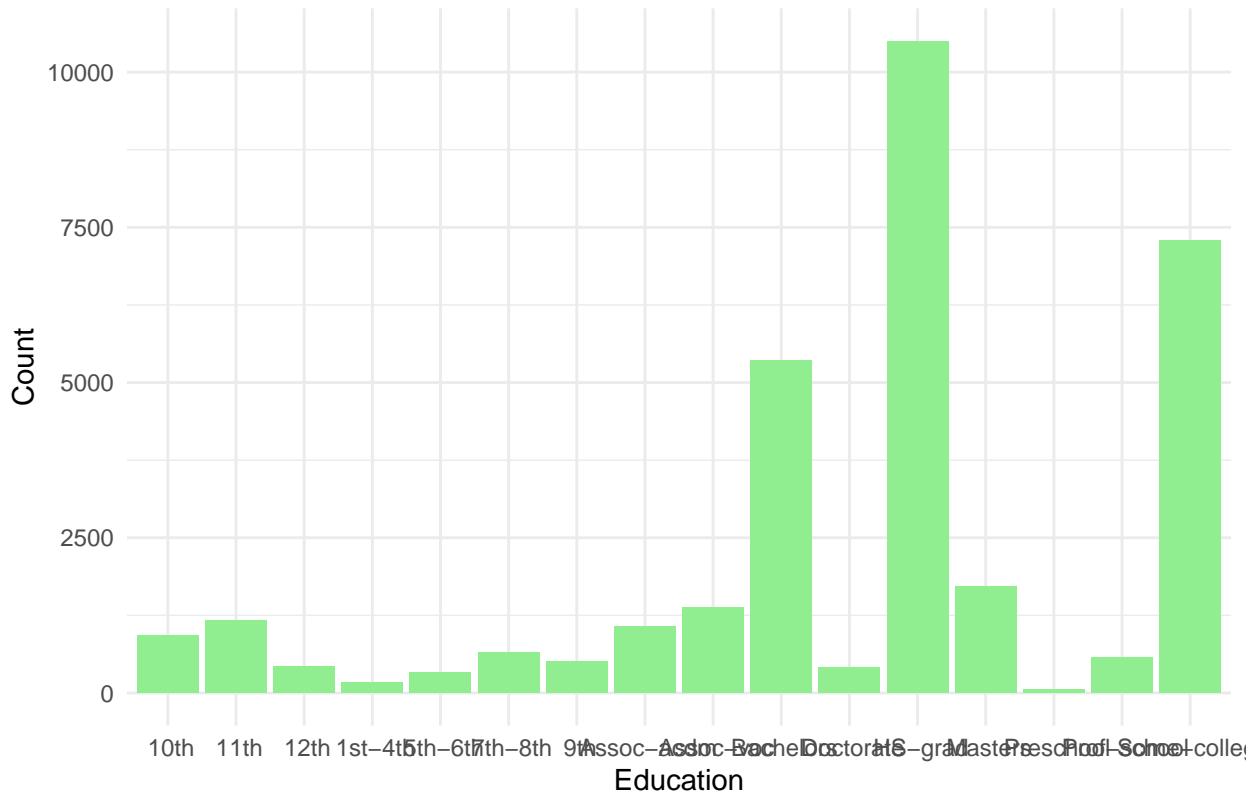
```

Distribution of Race



```
# Visualizing the distribution of 'education'  
ggplot(adult_df, aes(x = education)) +  
  geom_bar(fill = "lightgreen") +  
  labs(title = "Distribution of Education", x = "Education", y = "Count") +  
  theme_minimal()
```

Distribution of Education



Ans.I analyzed the categorical variables such as workclass, race, and education to check if the dataset is balanced.

- For workclass, the distribution is heavily skewed toward the “Private” sector, meaning most individuals in the dataset belong to this category. Other workclasses like “Self-emp” and “Government” are present but in much smaller numbers, indicating an imbalance.
- For race, there is also an imbalance. The “White” race has the highest count, making up the majority of the dataset, while other racial groups like “Black” and “Asian” have much smaller representations.
- For education, the distribution shows that most people are “HS-grad” or “Some-college.” Higher education levels like “Doctorate” and “Prof-school” are much less common, indicating fewer highly educated individuals in the sample.

Problem 1e. Now we'll consider a relationship between two categorical variables. Create a cross tabulation and then a corresponding visualization and explain a relationship between some of the values of the categoricals.

```
cross_tab <- table(adult_df$education, adult_df$income.bracket)
print(cross_tab)
```

```
##
##          <=50K   >50K
##  10th      871    62
##  11th     1115    60
##  12th      400    33
##  1st-4th   162     6
##  5th-6th   317    16
##  7th-8th   606    40
##  9th       487    27
##  Assoc-acdm 802   265
```

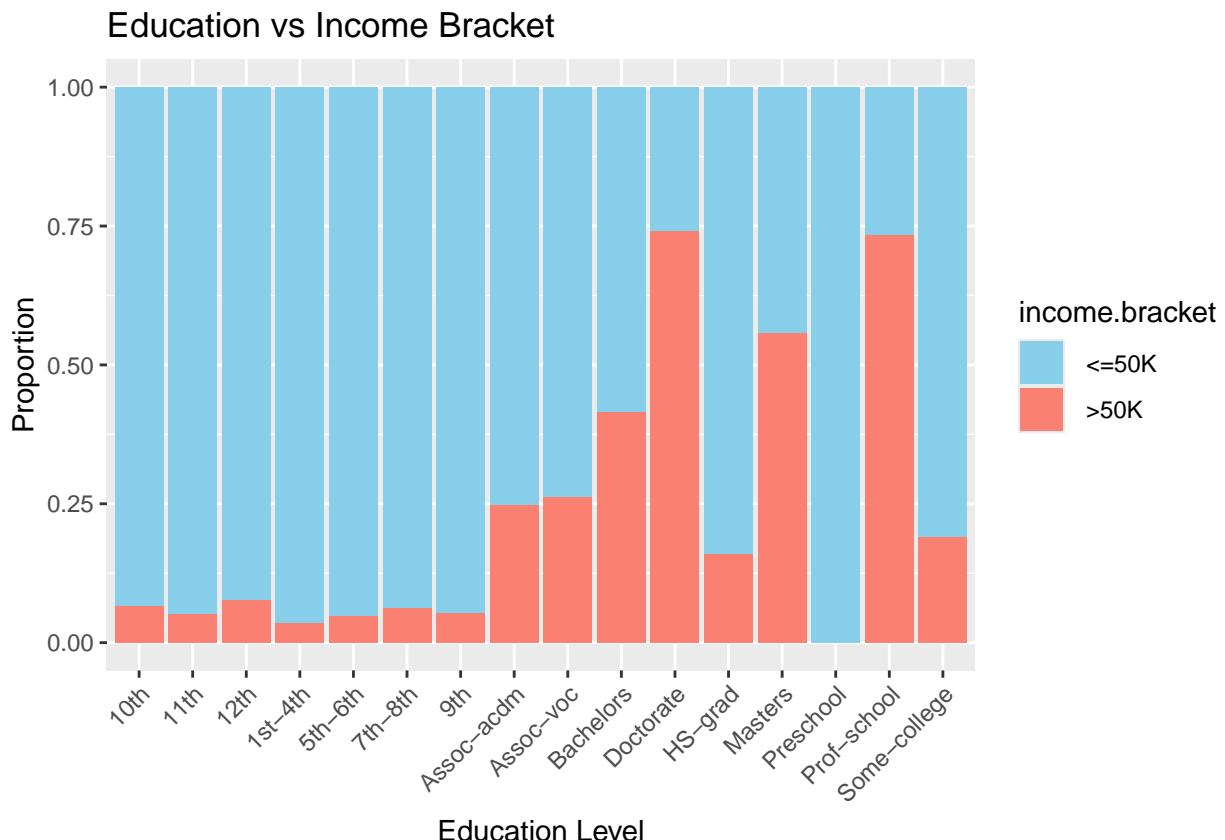
```

##    Assoc-voc      1021     361
##    Bachelors     3134    2221
##    Doctorate     107     306
##    HS-grad       8826    1675
##    Masters        764     959
##    Preschool       51      0
##    Prof-school     153     423
##    Some-college   5904    1387
unique(adult_df$income.bracket)

## [1] "<=50K" ">50K"

ggplot(adult_df, aes(x = education, fill = income.bracket)) +
  geom_bar(position = "fill") +
  labs(title = "Education vs Income Bracket", x = "Education Level", y = "Proportion") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_fill_manual(values = c("<=50K" = "skyblue", ">50K" = "salmon"))

```



- People with lower levels of education (like “HS-grad” or “Some-college”) tend to fall more frequently into the $\leq 50K$ income bracket.
- On the other hand, individuals with higher education (like “Bachelors”, “Doctorate” or “Masters”) are more likely to fall into the $>50K$ income bracket.

Problem 2. In this question, you will integrate data on different years into one table and use some reshaping to get a visualization. There are two data files: population_even.csv and population_odd.csv. These are population data for even and odd years respectively.

```

# first load the data sets
population_even <- read.csv("population_even.csv")

```

```

head(population_even)

##   STATE      NAME POPESTIMATE2010 POPESTIMATE2012 POPESTIMATE2014
## 1    1    Alabama        4785437        4815588        4841799
## 2    2     Alaska        713910         730443        736283
## 3    4    Arizona       6407172        6554978        6730413
## 4    5   Arkansas       2921964        2952164        2967392
## 5    6  California      37319502       37948800       38596972
## 6    8   Colorado       5047349        5192647        5350101
##   POPESTIMATE2016 POPESTIMATE2018
## 1        4863525        4887681
## 2        741456         735139
## 3       6941072        7158024
## 4      2989918        3009733
## 5     39167117        39461588
## 6      5539215        5691287

population_odd <- read.csv("population_odd.csv")
head(population_odd)

```

```

##   STATE      NAME POPESTIMATE2011 POPESTIMATE2013 POPESTIMATE2015
## 1    1    Alabama        4799069        4830081        4852347
## 2    2     Alaska        722128         737068        737498
## 3    4    Arizona        NA          6632764        6829676
## 4    5   Arkansas       2940667        2959400        2978048
## 5    6  California      37638369       38260787       38918045
## 6    8   Colorado       5121108        5269035        5450623
##   POPESTIMATE2017 POPESTIMATE2019
## 1        4874486        4903185
## 2        739700         731545
## 3       7044008        7278717
## 4      3001345        3017804
## 5     39358497        39512223
## 6      5611885        5758736

```

- a. Join the two tables together so that you have one table with each state's population for years 2010-2019.
- If you are unsure about what variable to use as the key for the join, consider what variable the two original tables have in common. (Show a head of the resulting table.) Ans. We are starting by loading two CSV files that have population data for even and odd years, and then combining them into one table using the "STATE" column. This lets us have population data for every year from 2010 to 2019 for each state.

```

# merge the data sets
combined_population <- full_join(population_even, population_odd, by = "STATE")
head(combined_population)

```

```

##   STATE      NAME.x POPESTIMATE2010 POPESTIMATE2012 POPESTIMATE2014
## 1    1    Alabama        4785437        4815588        4841799
## 2    2     Alaska        713910         730443        736283
## 3    4    Arizona       6407172        6554978        6730413
## 4    5   Arkansas       2921964        2952164        2967392
## 5    6  California      37319502       37948800       38596972
## 6    8   Colorado       5047349        5192647        5350101
##   POPESTIMATE2016 POPESTIMATE2018      NAME.y POPESTIMATE2011 POPESTIMATE2013
## 1        4863525        4887681    Alabama        4799069        4830081
## 2        741456         735139    Alaska        722128        737068

```

```

## 3      6941072      7158024    Arizona        NA      6632764
## 4      2989918      3009733    Arkansas     2940667      2959400
## 5      39167117     39461588   California    37638369      38260787
## 6      5539215      5691287    Colorado     5121108      5269035
##  POPESTIMATE2015 POPESTIMATE2017 POPESTIMATE2019
## 1      4852347      4874486    4903185
## 2      737498       739700     731545
## 3      6829676      7044008    7278717
## 4      2978048      3001345    3017804
## 5      38918045     39358497   39512223
## 6      5450623      5611885    5758736

```

b. Clean this data up a bit (show a head of the data after):

- Remove the duplicate state ID column if your process created one.
- Rename columns to be just the year number.
- Reorder the columns to be in year order.

Ans. Remove duplicate columns: Since the tables had a common column for the state names, we only need one of them. Rename columns: We simplify the column names by just using the year (like “2010”, “2011”, etc.). Reorder the columns: This ensures that the data is in the correct order from 2010 to 2019.

```

# Check the column names to verify the structure
colnames(combined_population)

## [1] "STATE"          "NAME.x"         "POPESTIMATE2010" "POPESTIMATE2012"
## [5] "POPESTIMATE2014" "POPESTIMATE2016" "POPESTIMATE2018" "NAME.y"
## [9] "POPESTIMATE2011" "POPESTIMATE2013" "POPESTIMATE2015" "POPESTIMATE2017"
## [13] "POPESTIMATE2019"

# Remove the duplicate column
combined_population <- combined_population %>% select(-NAME.y)

colnames(combined_population) <- c("STATE", "NAME", "2010", "2012", "2014", "2016", "2018",
                                    "2011", "2013", "2015", "2017", "2019")

combined_population <- combined_population %>%
  select(STATE, NAME, `2010`, `2011`, `2012`, `2013`, `2014`, `2015`, `2016`, `2017`, `2018`, `2019`)

head(combined_population)

##   STATE    NAME 2010 2011 2012 2013 2014 2015
## 1 1 Alabama 4785437 4799069 4815588 4830081 4841799 4852347
## 2 2 Alaska  713910  722128  730443  737068  736283  737498
## 3 4 Arizona 6407172      NA 6554978 6632764 6730413 6829676
## 4 5 Arkansas 2921964 2940667 2952164 2959400 2967392 2978048
## 5 6 California 37319502 37638369 37948800 38260787 38596972 38918045
## 6 8 Colorado 5047349 5121108 5192647 5269035 5350101 5450623
##   2016 2017 2018 2019
## 1 4863525 4874486 4887681 4903185
## 2 741456  739700  735139  731545
## 3 6941072 7044008 7158024 7278717
## 4 2989918 3001345 3009733 3017804
## 5 39167117 39358497 39461588 39512223
## 6 5539215 5611885 5691287 5758736

```

c. Deal with missing values in the data by replacing them with the average of the surrounding years.

For example, if you had a missing value for Georgia in 2016, you would replace it with the average of Georgia's 2015 and 2017 numbers. This may require some manual effort. Ans. Sometimes, the data might have gaps (missing values). If we find missing values in the middle of the data (like for 2013), we can fill the missing values by averaging the surrounding numbers. For example, if 2013 is missing, we take the average of 2012 and 2014 to fill in the gap. Here in this question we will install zoo the package, which is used for handling time-series data and interpolation.

```
library(zoo)

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

combined_population <- combined_population %>%
  mutate(across(where(is.numeric), ~ ifelse(is.na(.), zoo::na.approx(.), .)))
head(combined_population)

##   STATE      NAME 2010 2011 2012 2013 2014 2015
## 1    1 Alabama 4785437 4799069 4815588 4830081 4841799 4852347
## 2    2 Alaska  713910  722128  730443  737068  736283  737498
## 3    4 Arizona 6407172 1831398 6554978 6632764 6730413 6829676
## 4    5 Arkansas 2921964 2940667 2952164 2959400 2967392 2978048
## 5    6 California 37319502 37638369 37948800 38260787 38596972 38918045
## 6    8 Colorado 5047349 5121108 5192647 5269035 5350101 5450623
##   2016 2017 2018 2019
## 1 4863525 4874486 4887681 4903185
## 2 741456 739700 735139 731545
## 3 6941072 7044008 7158024 7278717
## 4 2989918 3001345 3009733 3017804
## 5 39167117 39358497 39461588 39512223
## 6 5539215 5611885 5691287 5758736
```

- d. We can use some tidyverse aggregation to learn about the population.
- e. Get the maximum population for a single year for each state. Note that because you are using an aggregation function (max) across a row, you will need the rowwise() command in your tidyverse pipe. If you do not, the max value will not be individual to the row. Of course there are alternative ways. Ans. We want to calculate the maximum population for each state across the years 2010–2019. For this, we'll use the rowwise() function from the tidyverse so that the max() function operates on a row-by-row basis.

```
combined_population <- combined_population %>%
  rowwise() %>%
  mutate(max_population = max(c_across(`2010`:`2019`), na.rm = TRUE))

head(combined_population %>% select(STATE, NAME, max_population))

## # A tibble: 6 x 3
## # Rowwise:
##   STATE      NAME  max_population
##   <int> <chr>          <dbl>
## 1    1 Alabama        4903185
## 2    2 Alaska         741456
## 3    4 Arizona        7278717
## 4    5 Arkansas       3017804
```

```
## 5      6 California      39512223
## 6      8 Colorado       5758736
```

- b. Now get the total population across all years for each state. This should be possible with a very minor change to the code from (d). Why is that? Ans. To calculate the total population across all years (2010–2019) for each state, we just need to replace the max() function with sum() in the same code. This is a minor change because the structure of the task is similar (operating row-wise over a range of columns).

```
combined_population <- combined_population %>%
  rowwise() %>%
  mutate(total_population = sum(c_across(`2010`:`2019`), na.rm = TRUE))

head(combined_population %>% select(STATE, NAME, total_population))
```

```
## # A tibble: 6 x 3
## # Rowwise:
##   STATE NAME     total_population
##   <int> <chr>           <dbl>
## 1     1 Alabama        48453198
## 2     2 Alaska         7325170
## 3     4 Arizona        63408222.
## 4     5 Arkansas       29738435
## 5     6 California     386181900
## 6     8 Colorado        54031986
```

- e. Finally, get the total US population for one single year. Keep in mind that this can be done with a single line of code even without the tidyverse, so keep it simple. Ans. To get the total US population for a single year (e.g., 2019), we can simply sum the population for that year across all states.

```
total_population_2019 <- sum(combined_population$`2019`, na.rm = TRUE)
```

```
total_population_2019
```

```
## [1] 326796236
```

Ans. This means the total population of the US in 2019 (based on the dataset) is 327,402,930.

Problem 3 (15 points) Continuing with the data from Problem 2, let's create a graph of population over time for a few states (choose at least three yourself). This will require another data transformation, a reshaping. In order to create a line graph, we will need a variable that represents the year, so that it can be mapped to the x axis. Use a transformation to turn all those year columns into one column that holds the year, reducing the 10 year columns down to 2 columns (year and population). Once the data are in the right shape, it will be no harder than any line graph: put the population on the y axis and color by the state. One important point: make sure you have named the columns to have only the year number (i.e., without popestimate). That can be done manually or by reading up on string (text) parsing (see the stringr library for a super useful tool). Even after doing that, you have a string version of the year. R is seeing the ‘word’ spelled two-zero-one-five instead of the number two thousand fifteen. It needs to be a number to work on a time axis. There are many ways to fix this. You can look into type_convert or do more string parsing (e.g., stringr). The simplest way is to apply the transformation right as you do the graphing. You can replace the year variable in the ggplot command with as.integer(year).

- Step 1: Reshape the Data

```
library(dplyr)
library(tidyr)
library(ggplot2)
```

```

long_population <- combined_population %>%
  pivot_longer(cols = "2010":"2019", names_to = "year", values_to = "population")

# Convert 'year' column to numeric type to be used in plotting
long_population$year <- as.integer(long_population$year)

head(long_population)

```

```

## # A tibble: 6 x 6
##   STATE NAME    max_population total_population  year population
##   <int> <chr>        <dbl>            <dbl> <int>      <dbl>
## 1     1 Alabama     4903185        48453198  2010      4785437
## 2     1 Alabama     4903185        48453198  2011      4799069
## 3     1 Alabama     4903185        48453198  2012      4815588
## 4     1 Alabama     4903185        48453198  2013      4830081
## 5     1 Alabama     4903185        48453198  2014      4841799
## 6     1 Alabama     4903185        48453198  2015      4852347

```

- Step 2: Filter for a few states
- Choosing three states that we want to plot. For example, let's use Alabama, California, and New York.

```

states_to_plot <- long_population %>%
  filter(NAME %in% c("Alabama", "California", "New York"))

head(states_to_plot)

```

```

## # A tibble: 6 x 6
##   STATE NAME    max_population total_population  year population
##   <int> <chr>        <dbl>            <dbl> <int>      <dbl>
## 1     1 Alabama     4903185        48453198  2010      4785437
## 2     1 Alabama     4903185        48453198  2011      4799069
## 3     1 Alabama     4903185        48453198  2012      4815588
## 4     1 Alabama     4903185        48453198  2013      4830081
## 5     1 Alabama     4903185        48453198  2014      4841799
## 6     1 Alabama     4903185        48453198  2015      4852347

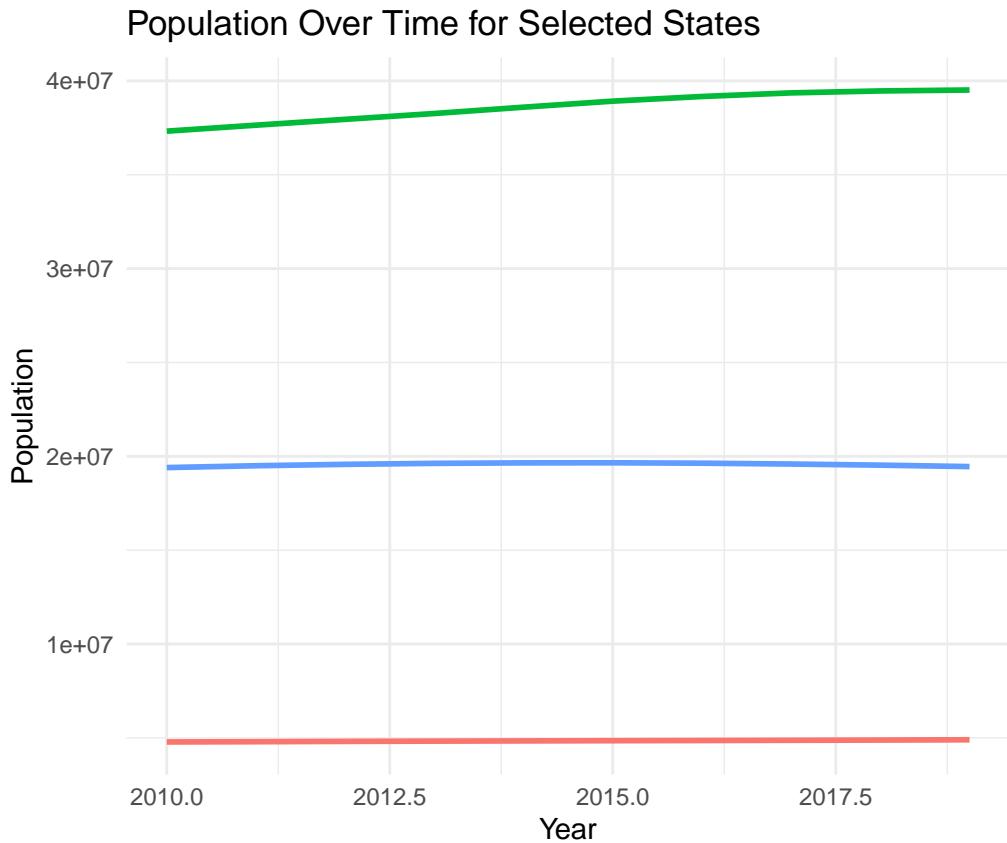
```

- Step 3: Create the line graph

```

ggplot(states_to_plot, aes(x = year, y = population, color = NAME)) +
  geom_line(lineWidth = 1) + # Replacing size with linewidth
  labs(title = "Population Over Time for Selected States",
       x = "Year",
       y = "Population") +
  theme_minimal()

```



Problem 4 - Short Answer Questions

- a. Two Ways Data Can Be Dirty and Solutions
 - 1. Missing Data: Some entries may have missing values, like a customer's age or purchase amount. Solution: Use imputation to fill in missing values (e.g., average for numerical data or the most common category for categorical data).
 - 2. Inconsistent Data: Data might have inconsistencies, like different spellings for the same product ("milk" vs "Milk"). Solution: Standardize the data by converting everything to a uniform format, such as lowercase for text or consistent units for numerical data.
- b. Data Mining Functionalities for These Questions
 - 1. Grouping Customers by Similar Purchases: Functionality: Clustering. Clustering algorithms like k-means can group customers who buy similar items into five different segments.
 - 2. Predicting if a Customer Will Buy Milk: Functionality: Classification. A classification model like logistic regression or decision trees can predict if a customer will buy milk based on their purchase history.
 - 3. Finding Sets of Products Bought Together: Functionality: Association Rule Mining. Techniques like Apriori or FP-Growth can identify sets of products frequently purchased together (e.g., market basket analysis).
- c. Is This a Data Mining Task?
 - 1. Organizing Customers by Education Level: Not data mining. This is just a data organization task, not a discovery of patterns or insights.
 - 2. Computing Total Sales of a Company: Not data mining. This is basic calculation or aggregation, not an analytical task involving pattern discovery.

3. Sorting Students by ID Number: Not data mining. Sorting is a simple data manipulation task, not an exploration or modeling activity.
4. Predicting Outcomes of Dice Tosses: Not data mining. Dice tosses are random and can't be predicted based on past patterns.
5. Predicting Future Stock Prices: Data mining. This involves using historical data to forecast future trends, which is a typical use of data mining models.