Homework1

Kavana Manvi KrishnaMurthy 2158

2024-04-22

Problem 1

1.a. 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?

```
adult <- read.csv("/Users/kavanamanvi/Desktop/FDS/HW1/adult.csv")
summary(adult)
```

```
##       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                      Mean   : 189778
##  3rd Qu.:48.00                      3rd Qu.: 237051
##  Max.   :90.00                      Max.   :1484705
##  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
```

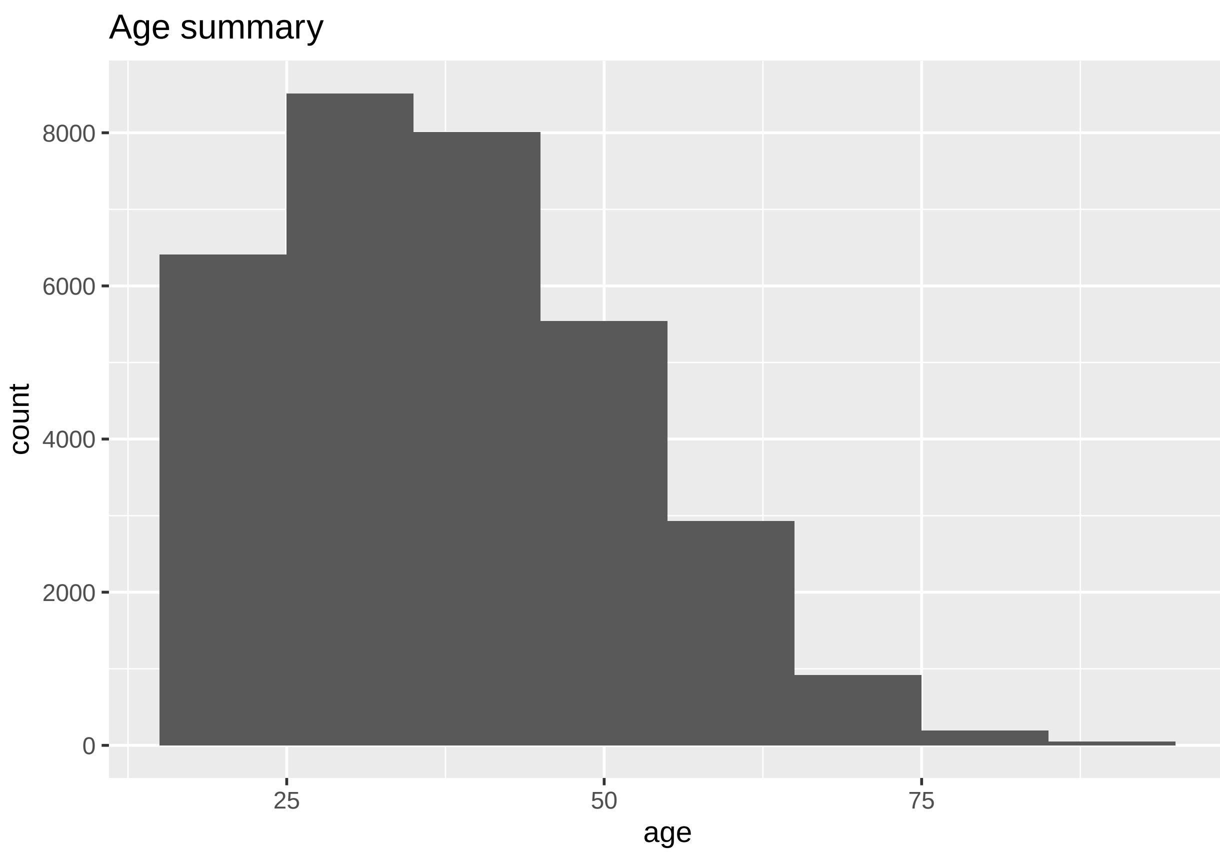
Age Summary

```r
summary(adult$age)
```

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

The histogram of age tells us that it's a bell shaped curve. It's right skewed. It is positive skewed as mean is greater than median.

```r
library("ggplot2")
ggplot(adult, aes(age)) + geom_histogram( binwidth = 10)+
labs(title = "Age summary",   x = "age",y = "count")
```



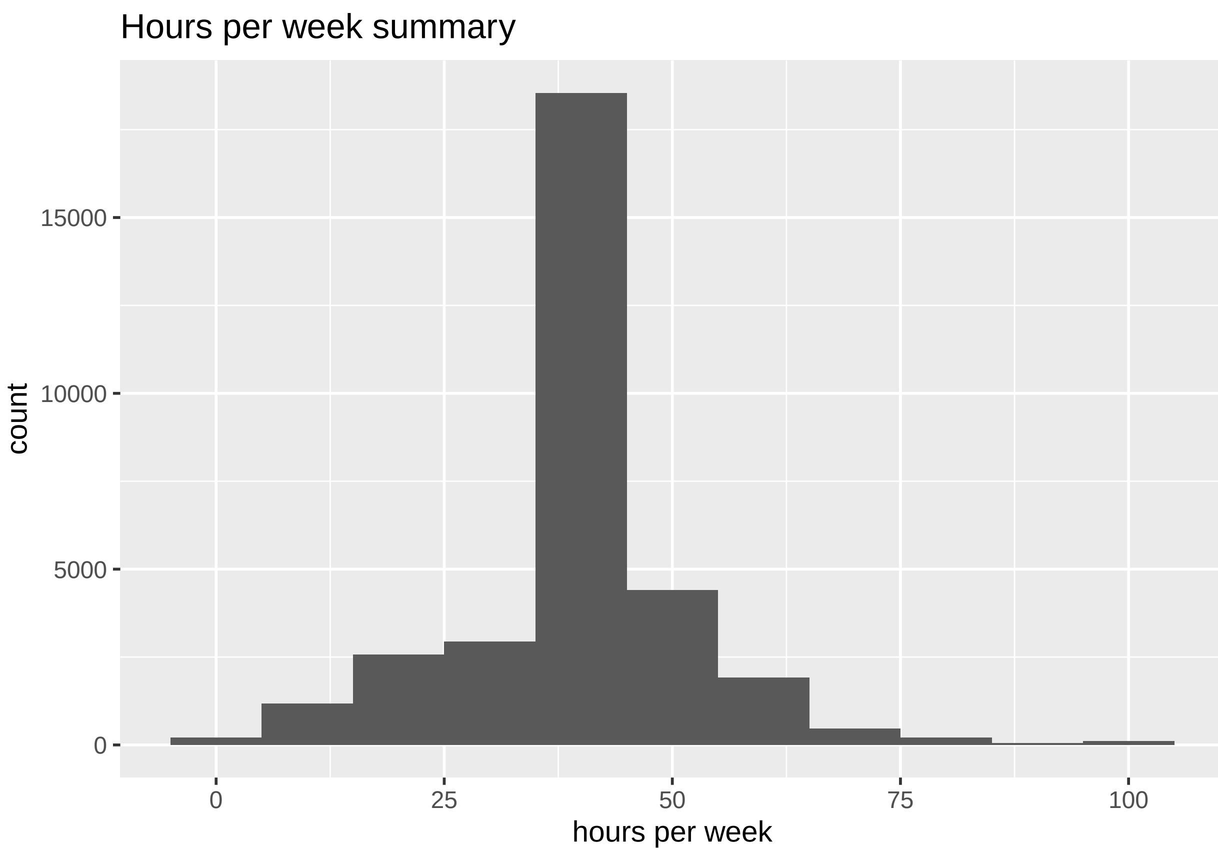
Hours per week summary

```r
summary(adult$hours.per.week)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   40.00   40.00   40.44   45.00   99.00
```

It is a bell-shaped histogram. This is a symmetrical shape that is often seen in data that is normally distributed. It is a symmetrical distribution as mean is almost equal to median.

```r
ggplot(adult, aes(hours.per.week)) + geom_histogram( binwidth = 10)+
  labs(title = "Hours per week summary",   x = "hours per week",y = "count")
```

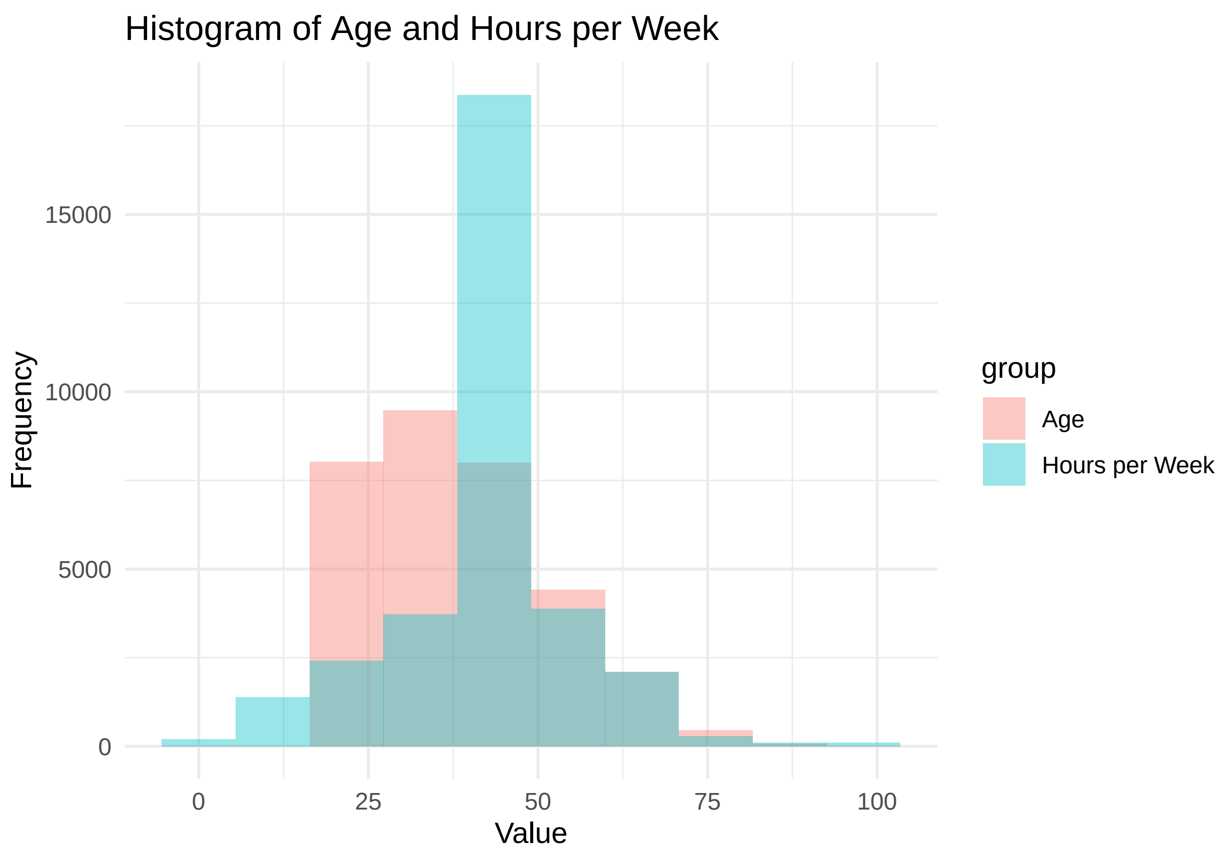


b. Use a visualisation to get a fine-grain comparison (you don't have to use QQ plots, though) of the distributions of those two variables. Why did you choose the type of visualisation that you chose? How do your part (a) assumptions compare to what you can see visually?

In order to get a detail comparison, emphasise shape by layering histograms. It does not generalise too many variables the way box plot does it. A layered histogram allows us to overlay the histograms of two variables in the same plot, making it easy to compare their distributions. We choose this type of visualization because it provides a clear visual comparison of the two variables while showing their individual distributions. We can closely examining the distributions of variables to identify any small differences or patterns that may not be immediately obvious.

```r
 data_subset <- data.frame(values = c(adult$age, adult$hours.per.week),
                          group = c(rep("Age", nrow(adult)),
                          rep("Hours per Week", nrow(adult))))
```

```r
ggplot(data_subset, aes(x = values, fill = group)) +
geom_histogram(position = "identity", alpha = 0.4, bins = 10) +
labs(title = "Histogram of Age and Hours per Week", x = "Value",
y = "Frequency") + theme_minimal()
```



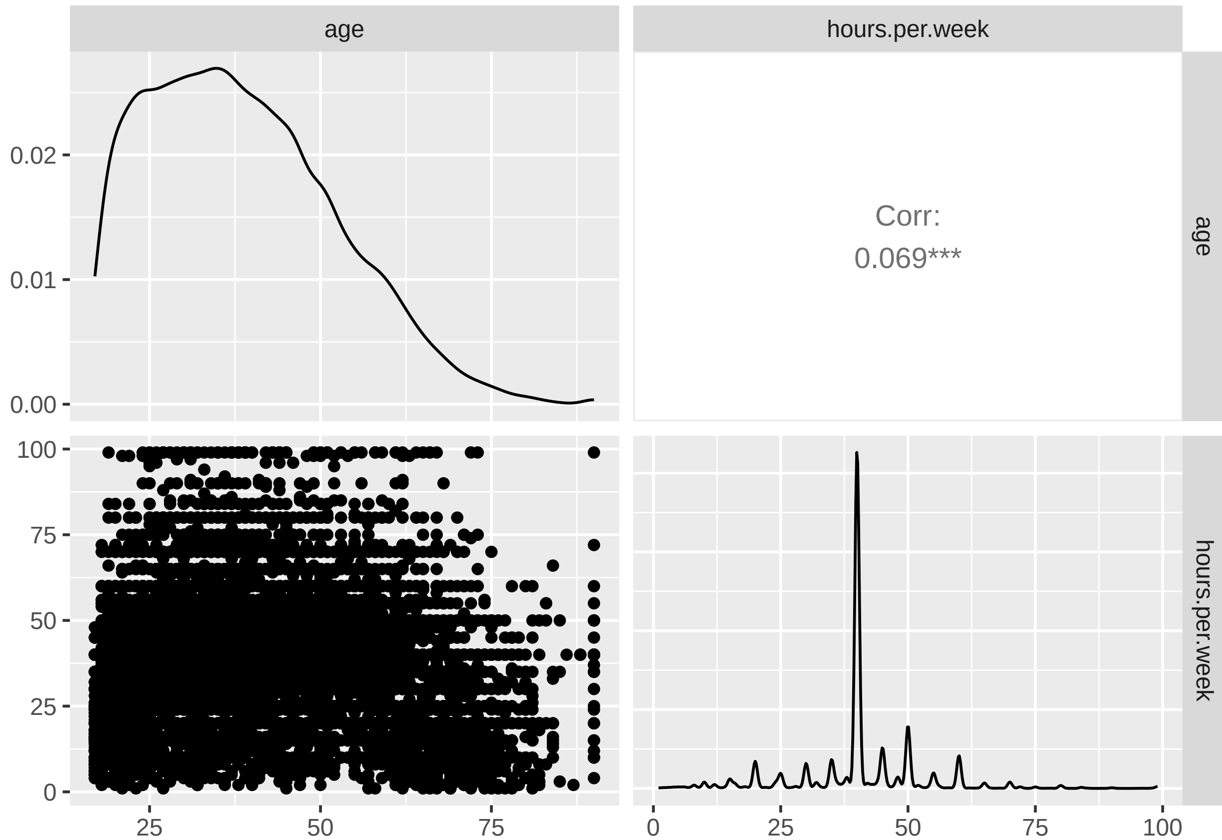
Initial Thoughts on the Distribution: • Age: The age distribution might lean towards younger age groups. This suggests a potentially higher concentration of individuals in those brackets compared to older ones. • Hours Worked per Week: The distribution of working hours might have a central peak, indicating a common range for most people. However, a tail extending towards the right suggests a portion of the population works significantly more hours than the average. Visually Confirming the Trends: The histogram seems to support these initial thoughts. The age distribution appears wider in the center, hinting at a possible concentration around a central age group. Conversely, the working hours distribution appears to have a broader spread, suggesting more variation in the number of hours worked compared to age.

c. 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
```

```
ggpairs(adult, columns = c("age", "hours.per.week"))
```



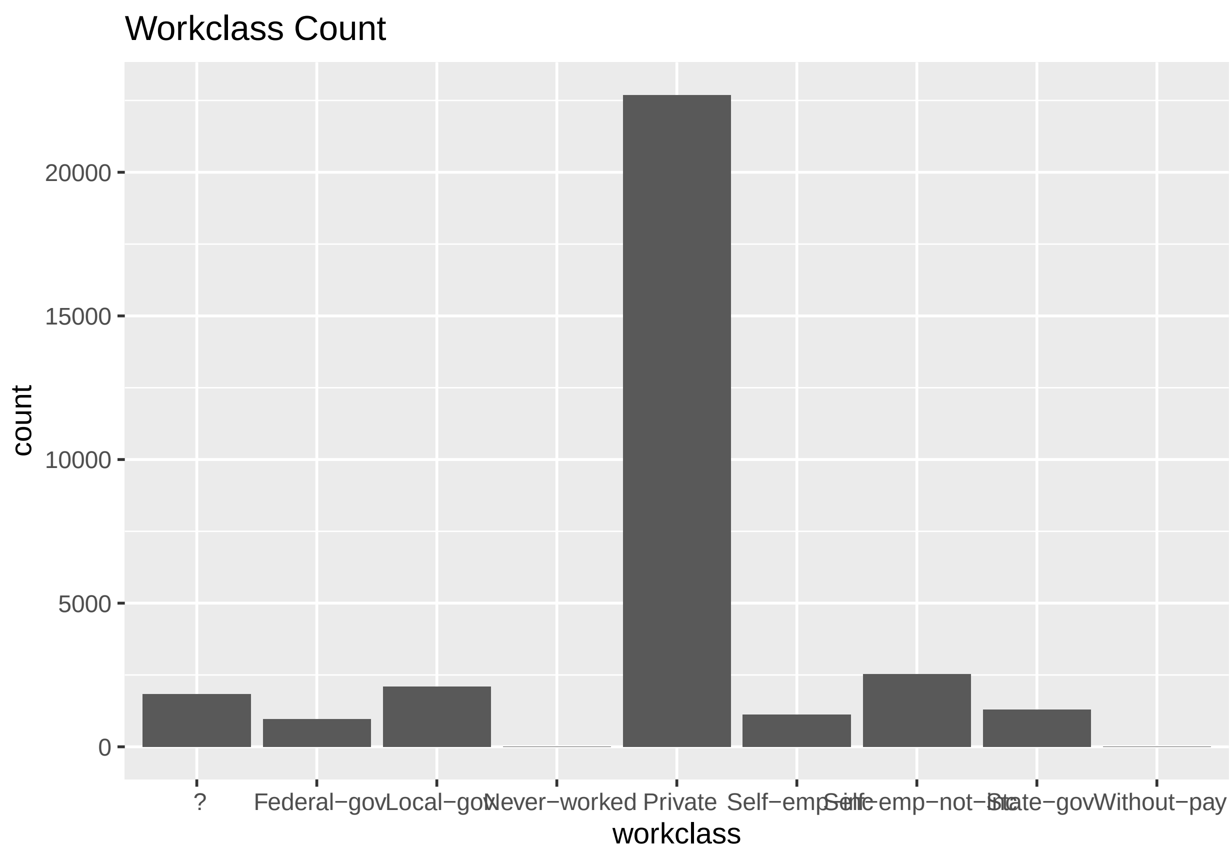
A scatterplot matrix of numerical variables, such as age and hours per week, presents a visual overview of the relationships between these variables and all other numerical variables in the dataset. By displaying pairwise scatterplots, this matrix enables a rapid evaluation of possible correlations or patterns. It can uncover connections between age and hours worked per week that might not be evident when examining their distributions independently. For instance, it can reveal if there's a tendency for older individuals to work longer hours or if there's no discernible link between age and weekly work hours.

d. 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 tidyererse group_by with count. I recommend you try a few for practice.

The categorical variables wrorkclass and education have a lot more in one category than in others.

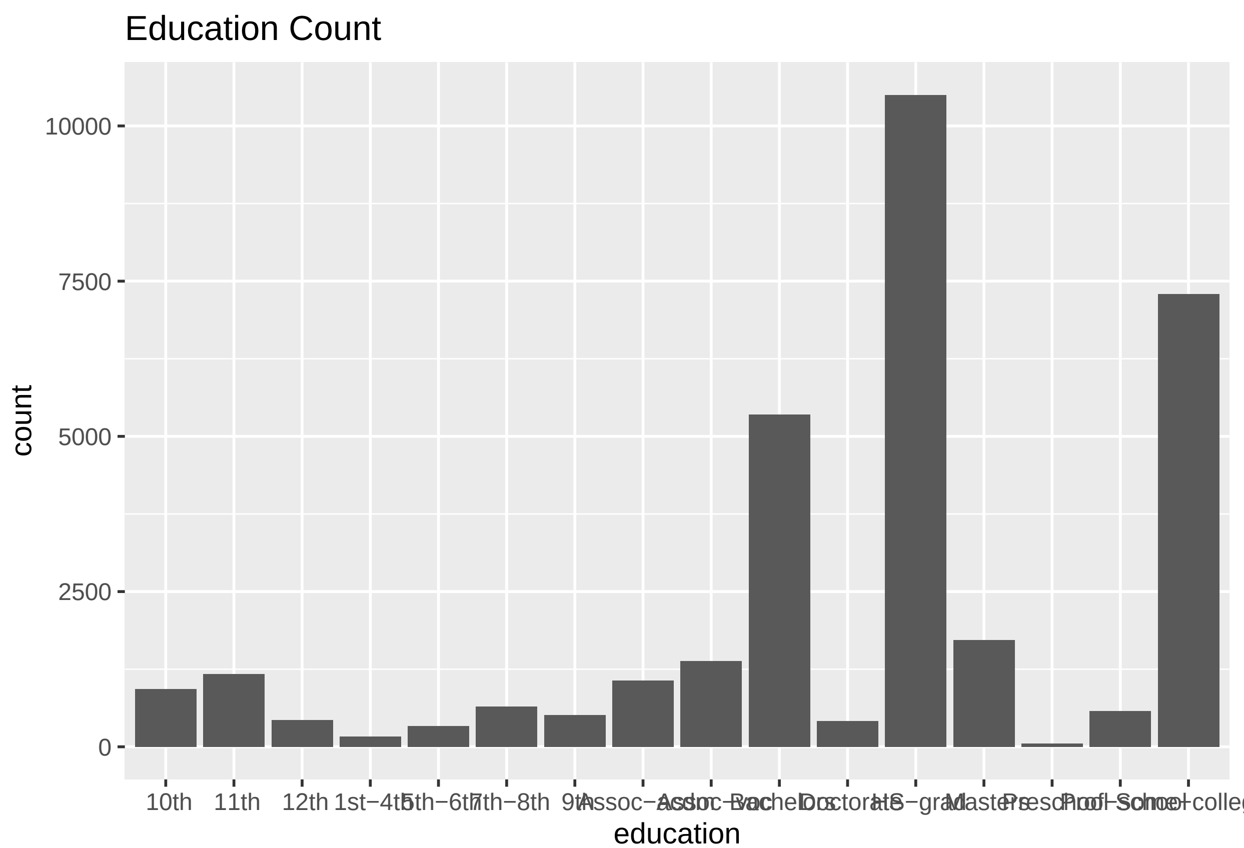
Categorical variable : workclass- has more values in private category.

```
ggplot(adult, aes(x=workclass)) + geom_bar()+
  labs(title = "Workclass Count",   x = "workclass",y = "count")
```



Categorical variable : education- has more values in HS-grad category

```
ggplot(adult, aes(x=education)) + geom_bar()+
  labs(title = "Education Count",   x = "education",y = "count")
```



Practicing with other technique on different categorical variable: Race

```
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
```

```
race_counts <- adult %>%
  group_by(race) %>%
  summarise(count = n())
print(race_counts)
```

```
## # A tibble: 5 x 2
##   race                  count
##   <chr>                 <int>
## 1 " Amer-Indian-Eskimo"   311
```

```
## 2 " Asian-Pac-Islander"  1039
## 3 " Black"               3124
## 4 " Other"                271
## 5 " White"              27816
```

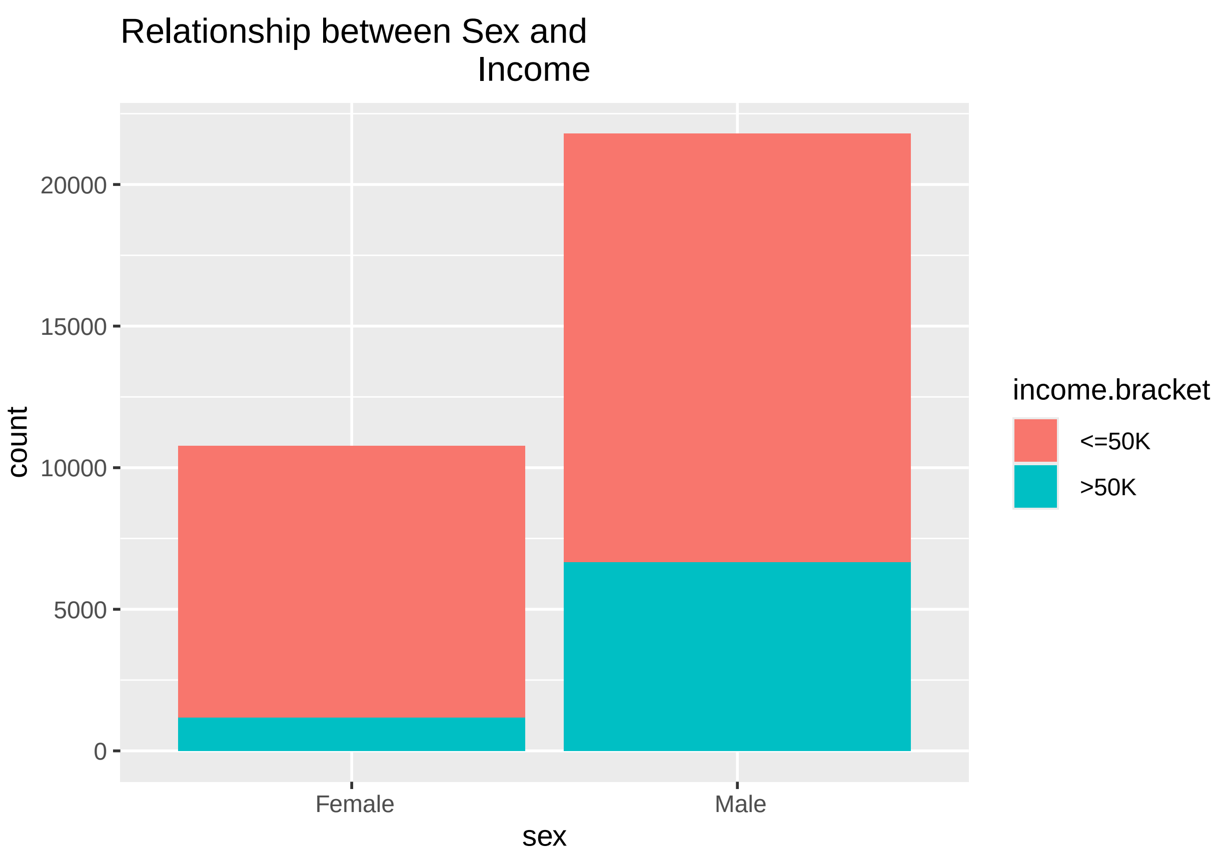
e. Now we'll consider a relationship between two categorical variables. Create a cross tabulation and then a corresponding visualization and explain a relationship between some of the values of the categorical variables.

```
cross_tab <- table(adult$sex,adult$income.bracket)
print(cross_tab)
```

```
##
##              <=50K  >50K
##     Female    9592  1179
##     Male     15128  6662
```

Visualization:

```
p <- ggplot(adult, aes(x=sex, fill=income.bracket))
p + geom_bar(position="stack") +labs(title = "Relationship between Sex and
                                     Income",
                                     x = "sex",y = "count")
```



#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.

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.)

Load the population_even.csv and population_odd.csv to variables pe and po respectively. Perform full outer join to combine the two tables using 'STATE' as key.

```
po <- read.csv("/Users/kavanamanvi/Desktop/FDS/HW1/population_odd.csv")
pe <- read.csv("/Users/kavanamanvi/Desktop/FDS/HW1/population_even.csv")
library("dplyr")
pop_join <- full_join(po, pe, by="STATE")
head(pop_join)
```

```
##   STATE     NAME.x 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     NAME.y POPESTIMATE2010 POPESTIMATE2012
## 1         4874486         4903185    Alabama         4785437         4815588
## 2          739700          731545     Alaska          713910          730443
## 3         7044008         7278717    Arizona         6407172         6554978
## 4         3001345         3017804   Arkansas         2921964         2952164
## 5        39358497        39512223 California        37319502        37948800
## 6         5611885         5758736   Colorado         5047349         5192647
##   POPESTIMATE2014 POPESTIMATE2016 POPESTIMATE2018
## 1         4841799         4863525         4887681
## 2          736283          741456          735139
## 3         6730413         6941072         7158024
## 4         2967392         2989918         3009733
## 5        38596972        39167117        39461588
## 6         5350101         5539215         5691287
```

b. Clean this data up a bit (show a head of the data after):
c. Remove the duplicate state ID column if your process created one.

Select only the necessary columns and remove columns ending with ".y"

Remove the column named 'NAME.y' that's redundant.

```
pop_final <- pop_join %>% select(-matches("\\.y$"))
head(pop_final)
```

```
##   STATE     NAME.x 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 POPESTIMATE2010 POPESTIMATE2012
## 1         4874486         4903185         4785437         4815588
## 2          739700          731545          713910          730443
## 3         7044008         7278717         6407172         6554978
## 4         3001345         3017804         2921964         2952164
## 5        39358497        39512223        37319502        37948800
## 6         5611885         5758736         5047349         5192647
##   POPESTIMATE2014 POPESTIMATE2016 POPESTIMATE2018
## 1         4841799         4863525         4887681
## 2          736283          741456          735139
## 3         6730413         6941072         7158024
## 4         2967392         2989918         3009733
## 5        38596972        39167117        39461588
## 6         5350101         5539215         5691287
```

rename name.x to name

```r
pop_final <- pop_final %>%rename(NAME = NAME.x)
head(pop_final)
```

```
##   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 POPESTIMATE2010 POPESTIMATE2012
## 1         4874486         4903185         4785437         4815588
## 2          739700          731545          713910          730443
## 3         7044008         7278717         6407172         6554978
## 4         3001345         3017804         2921964         2952164
## 5        39358497        39512223        37319502        37948800
## 6         5611885         5758736         5047349         5192647
##   POPESTIMATE2014 POPESTIMATE2016 POPESTIMATE2018
## 1         4841799         4863525         4887681
## 2          736283          741456          735139
## 3         6730413         6941072         7158024
## 4         2967392         2989918         3009733
## 5        38596972        39167117        39461588
## 6         5350101         5539215         5691287
```

b. Rename columns to be just the year number.

```r
colnames(pop_final)[2]<-"NAME"
colnames(pop_final)[3]<-"2011"
colnames(pop_final)[4]<-"2013"
colnames(pop_final)[5]<-"2015"
```

```r
colnames(pop_final)[6]<-"2017"
colnames(pop_final)[7]<-"2019"
colnames(pop_final)[8]<-"2010"
colnames(pop_final)[9]<-"2012"
colnames(pop_final)[10]<-"2014"
colnames(pop_final)[11]<-"2016"
colnames(pop_final)[12]<-"2018"
head(pop_final)
```

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

c. Reorder the columns to be in year order.

```r
pop_final <- pop_final %>% relocate("2012", .after = "2011")
pop_final <- pop_final %>% relocate("2014", .after = "2013")
pop_final <- pop_final %>% relocate("2016", .after = "2015")
pop_final <- pop_final %>% relocate("2018", .after = "2017")
pop_final <- pop_final %>% relocate("2010", .after = "NAME")
head(pop_final)
```

```
##   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.

Missing values are in the following columns-2011, 2013, 2015, 2017 and 2019

Calculate the missing values and update it in the table:

```
pop_final[3,"2011"] <- (pop_final[36,"2010"]+ pop_final[36,"2012"])/2
pop_final[36,"2013"] <- (pop_final[36,"2012"]+ pop_final[36,"2014"])/2
pop_final[13,"2015"] <- (pop_final[13,"2014"]+ pop_final[13,"2016"])/2
pop_final[27,"2017"] <- (pop_final[27,"2016"]+ pop_final[27,"2018"])/2
pop_final[50,"2019"] <- 5819346
summary(pop_final)
```

```
##      STATE           NAME                2010              2011
##  Min.   : 1.00   Length:52          Min.   :  564487   Min.   :  567299
##  1st Qu.:16.75   Class :character   1st Qu.: 1764843   1st Qu.: 1776482
##  Median :29.50   Mode  :character   Median : 4092836   Median : 4120928
##  Mean   :29.79                      Mean   : 6020061   Mean   : 6159752
##  3rd Qu.:42.50                      3rd Qu.: 6610438   3rd Qu.: 7145259
##  Max.   :72.00                      Max.   :37319502   Max.   :37638369
##       2012              2013              2014              2015
##  Min.   :  576305   Min.   :  582122   Min.   :  582531   Min.   :  585613
##  1st Qu.: 1788808   1st Qu.: 1793237   1st Qu.: 1794895   1st Qu.: 1795724
##  Median : 4142674   Median : 4163564   Median : 4188796   Median : 4220884
##  Mean   : 6105105   Mean   : 6145883   Mean   : 6189152   Mean   : 6232963
##  3rd Qu.: 6721518   3rd Qu.: 6775982   3rd Qu.: 6835611   3rd Qu.: 6913171
##  Max.   :37948800   Max.   :38260787   Max.   :38596972   Max.   :38918045
##       2016              2017              2018              2019
##  Min.   :  584215   Min.   :  578931   Min.   :  577601   Min.   :  578759
##  1st Qu.: 1793862   1st Qu.: 1792182   1st Qu.: 1790852   1st Qu.: 1790876
##  Median : 4264079   Median : 4297946   Median : 4321520   Median : 4342705
##  Mean   : 6275923   Mean   : 6313637   Mean   : 6343863   Mean   : 6373656
##  3rd Qu.: 7029497   3rd Qu.: 7138846   3rd Qu.: 7249485   3rd Qu.: 7362761
##  Max.   :39167117   Max.   :39358497   Max.   :39461588   Max.   :39512223
```

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

```
pop_final_max <- pop_final %>%
  rowwise() %>%
  mutate(MAX_population = max(`2010`, `2011`, `2012`, `2013`, `2014`,
                              `2015`, `2016`, `2017`, `2018`, `2019`,
                              na.rm = TRUE)) %>%
  select(NAME, MAX_population) %>%
  distinct()
print(pop_final_max)
```

```
## # A tibble: 52 x 2
## # Rowwise:
##    NAME                     MAX_population
##    <chr>                             <dbl>
##  1 Alabama                         4903185
```

```
##   2 Alaska                       741456
##   3 Arizona                    11544130.
##   4 Arkansas                    3017804
##   5 California                 39512223
##   6 Colorado                    5758736
##   7 Connecticut                 3594841
##   8 Delaware                     973764
##   9 District of Columbia         705749
##  10 Florida                    21477737
## # i 42 more rows
```

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?

Use the SUM aggregate function instead of max.

```r
pop_final_total <- pop_final %>%
  rowwise() %>%
  mutate(Total_population = sum(`2010`, `2011`, `2012`, `2013`, `2014`,
                                `2015`, `2016`, `2017`, `2018`, `2019`,
                                na.rm = TRUE)) %>%
  select(NAME, Total_population) %>%
  distinct()
print(pop_final_total)
```

```
## # A tibble: 52 x 2
## # Rowwise:
##    NAME                  Total_population
##    <chr>                            <dbl>
##  1 Alabama                       48453198
##  2 Alaska                         7325170
##  3 Arizona                       73120954.
##  4 Arkansas                      29738435
##  5 California                   386181900
##  6 Colorado                      54031986
##  7 Connecticut                   35826676
##  8 Delaware                       9364455
##  9 District of Columbia           6636276
## 10 Florida                      201096314
## # i 42 more rows
```

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.

```r
total_us_population_2012 <- sum(pop_final$`2012`, na.rm = TRUE)
print(total_us_population_2012)
```

```
## [1] 317465478
```

#Problem 3 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).

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

Reshape the data

```
pop_reshape <- pop_final %>%
  pivot_longer(cols = starts_with("20"), names_to = "year",
               values_to = "population") %>%
  mutate(year = as.integer(year))
print(pop_reshape)
```

```
## # A tibble: 520 x 4
##    STATE NAME     year population
##    <int> <chr>   <int>      <dbl>
##  1     1 Alabama  2010    4785437
##  2     1 Alabama  2011    4799069
##  3     1 Alabama  2012    4815588
##  4     1 Alabama  2013    4830081
##  5     1 Alabama  2014    4841799
##  6     1 Alabama  2015    4852347
##  7     1 Alabama  2016    4863525
##  8     1 Alabama  2017    4874486
##  9     1 Alabama  2018    4887681
## 10     1 Alabama  2019    4903185
## # i 510 more rows
```

Select a few states for plotting

```
states <- c("California", "Illinois", "New York", "Texas")
print(states)
```

```
## [1] "California" "Illinois"   "New York"   "Texas"
```

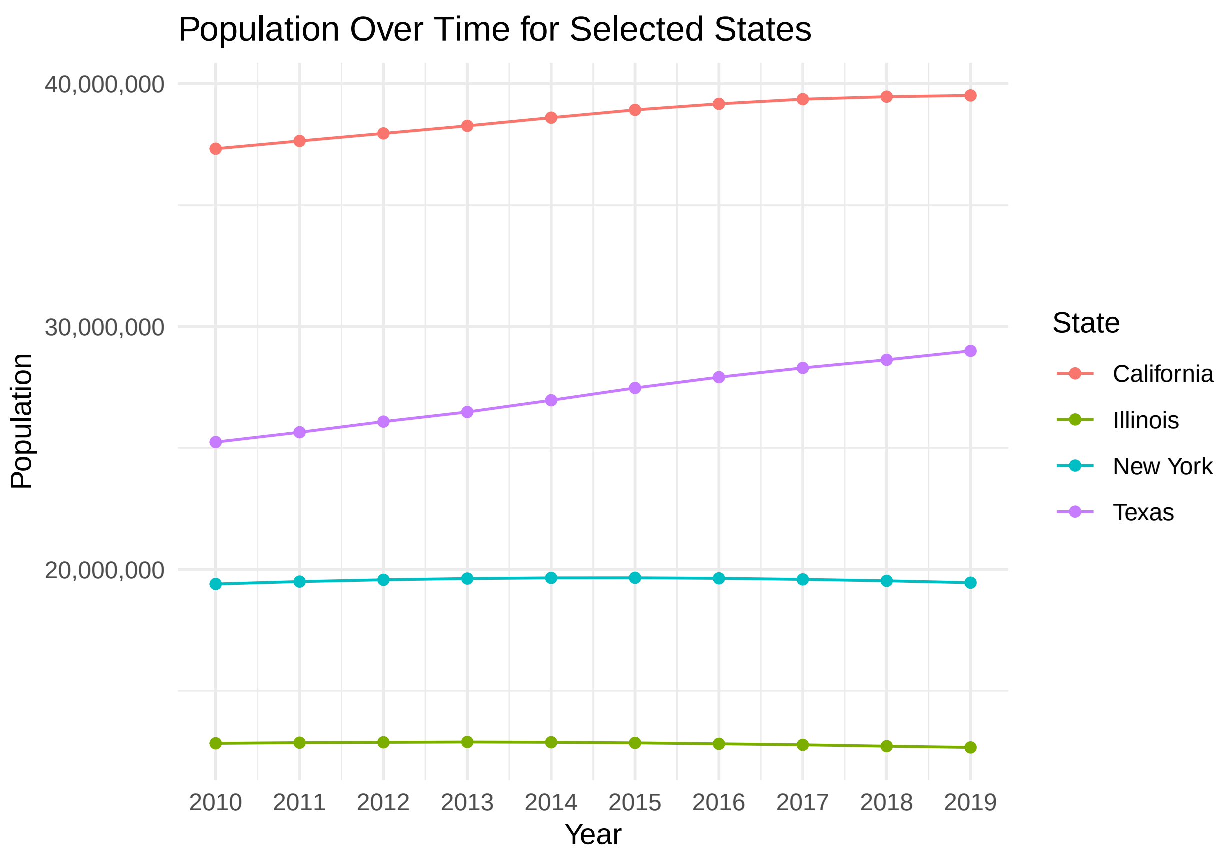
Filter the data for selected states

```
pop_reshape_selected_states <- pop_reshape %>%filter(NAME %in% states)
head(pop_reshape_selected_states)
```

```
## # A tibble: 6 x 4
##   STATE NAME        year population
##   <int> <chr>      <int>      <dbl>
## 1     6 California  2010   37319502
## 2     6 California  2011   37638369
## 3     6 California  2012   37948800
## 4     6 California  2013   38260787
## 5     6 California  2014   38596972
## 6     6 California  2015   38918045
```

Create a line graph

```
ggplot(pop_reshape_selected_states, aes(x = as.integer(year), y = population,
                                        group = NAME, color = NAME)) +
  geom_line() +
  geom_point() +
  scale_x_continuous(breaks = seq(min(pop_reshape_selected_states$year), max(pop_reshape_selected_state
  scale_y_continuous(labels = scales::comma) +
  labs(title = "Population Over Time for Selected States", x = "Year",
       y = "Population", color = "State") +
  theme_minimal()
```



#Problem 4

This problem is short answer questions only. No code is needed.

a. Describe two ways in which data can be dirty, and for each one, provide a potential solution.

- Inconsistent data occurs when different sources provide data in varying formats, such as different date formats (e.g., dd/mm/yyyy vs. mm/dd/yyyy). To address this, data should be standardized to a single format across all sources, ensuring consistency.

- Noisy data includes outliers or incorrect values, like a negative salary (-1000). One solution is to apply data cleaning techniques such as outlier detection and removal, or imputation to replace incorrect values with estimates based on the rest of the data.

b. Explain which data mining functionality you would use to help with each of these data questions.

a) Suppose we have data where each row is a customer and we have columns that describe their purchases. What are five groups of customers who buy similar things?

Clustering algorithms are used to find the groups or clusters of customers who buy similar things based on their type of purchases. Example K-means algorithm iteratively assigns data points to the nearest cluster centroid and then updates the centroids based on the mean of the data points in each cluster.

b) For the same data: can I predict if a customer will buy milk based on what else they bought?

To predict whether a customer is likely to purchase milk based on their other purchases, one would utilize a classification approach. This involves employing algorithms such as logistic regression, decision trees, or random forests to develop a model that can forecast the probability of a customer buying milk based on their purchasing behaviour.

c) Suppose we have data listing items in individual purchases. What are different sets of products that are often purchased together?

To discover common sets of products frequently purchased together, one would employ association rule mining, particularly leveraging the Apriori algorithm. This technique identifies frequent items, which are groups of items that tend to co-occur in transactions. These item sets can then be used to identify patterns of products commonly bought together.

c. Explain if each of the following is a data mining task

a) Organizing the customers of a company according to education level.

Answer: No. This is a simple data organization task and does not involve extracting patterns or knowledge from data, which is the essence of data mining.

b) Computing the total sales of a company. Answer: No. This is a basic arithmetic calculation and does not involve extracting patterns or knowledge from data.

c) Sorting a student database according to identification numbers. Answer: No. This is a basic database operation and does not involve extracting patterns or knowledge from data.

d) Predicting the outcomes of tossing a (fair) pair of dice. Answer: No. This is a probability calculation and does not involve extracting patterns or knowledge from data.

e) Predicting the future stock price of a company using historical records. Answer: Yes. This is a data mining task as it involves building a model to predict future stock prices based on historical data, which requires extracting patterns and knowledge from the data.