Homework 1

SivaRamaKrishna(SRK) Yarra

2023-10-01

install.packages("dplyr",dependencies = TRUE) library(dplyr)

adult_data=read.csv("adult.csv",header=T) # Load required libraries library(dplyr)

#Problem 1(a)

```
# Summary statistics for age
adult_data=read.csv("adult.csv",header=T)
summary(adult_data)
```

```
##       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
```

```
summary_age <- summary(adult_data$age)
summary_age
```

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

```
summary_hours_per_week <- summary(adult_data$hours.per.week)
summary_hours_per_week
```

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

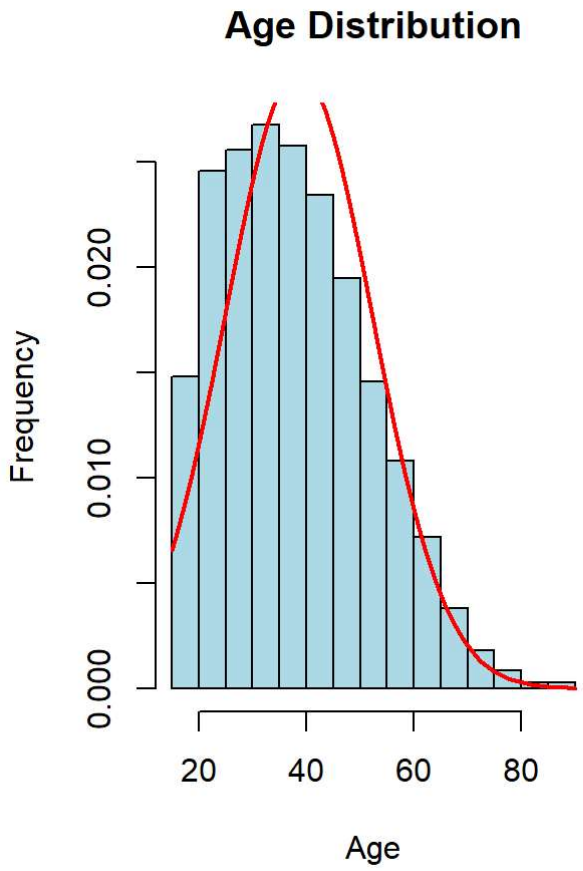
Explanation: This will provide us with summary statistics for each variable, including measures like the mean, median, quartiles (1st quartile, median/2nd quartile, and 3rd quartile), minimum, and maximum values. We are able to calculate summary statistics using adult dataset and able to provide data like hours and weekly as per age Summary (or descriptive) statistics are the first figures used to represent nearly every dataset. They also form the foundation for much more complicated computations and analyses. Thus, in spite of being composed of simple methods, they are essential to the analysis process

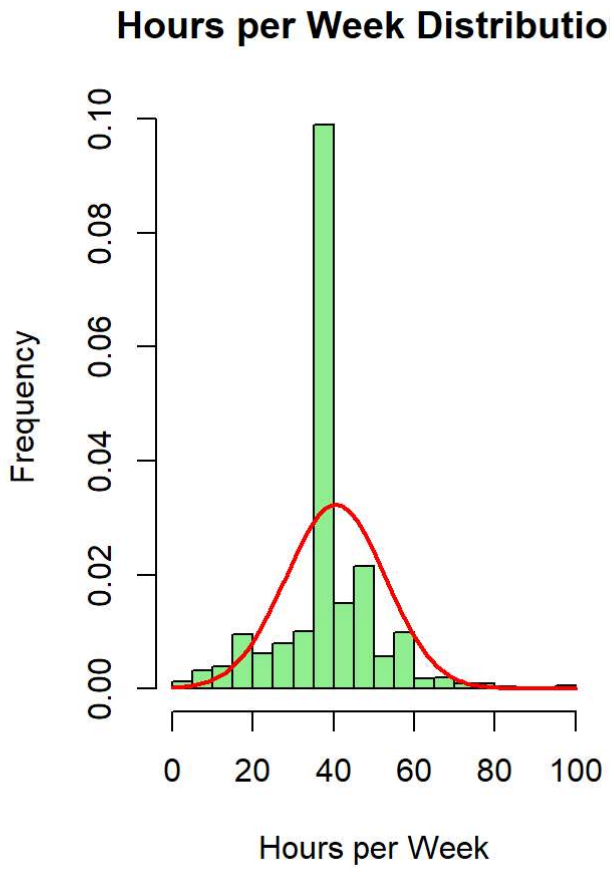
#Problem1(b)

```
par(mfrow = c(1, 2))  # Arrange plots side by side
mean<-mean(adult_data$age,na.rm=TRUE)
sd<-sd(adult_data$age,na.rm=TRUE)

mean1<-mean(adult_data$hours.per.week,na.rm=TRUE)
sd1<-sd(adult_data$hours.per.week,na.rm=TRUE)

hist(adult_data$age,main = "Age Distribution", xlab = "Age", ylab = "Frequency", col = "lightblu
e",freq=FALSE)
curve(dnorm(x,mean,sd),add=TRUE,col="red",lwd=2)
hist(adult_data$hours.per.week, main = "Hours per Week Distribution", xlab = "Hours per Week", y
lab = "Frequency", col = "lightgreen",freq=FALSE)
curve(dnorm(x,mean1,sd1),add=TRUE,col="red",lwd=2)
```





Explanation:In histograms, We can visually assess the distribution shape, whether it's symmetric, skewed, or bimodal. We are to visualize age distribution from the adult dataset and we are able to check the frequency per hour and weekly The histogram is a popular graphing tool. It is used to summarize discrete or continuous data that are measured on an interval scale. It is often used to illustrate the major features of the distribution of the data in a convenient form Histograms: Histograms are useful for visualizing the distribution of a single variable. They are particularly helpful when you want to examine the shape of the distribution, identify patterns, and assess whether it's symmetric, skewed, or bimodal. Histograms provide a detailed view of the data's frequency distribution.

Box Plots (Box-and-Whisker Plots): Box plots are valuable for comparing the distributions of multiple variables or identifying potential outliers within a single variable. They offer a summary view of the data's central tendency, spread (variability), skewness, and presence of outliers. Box plots are effective at highlighting the range, quartiles, and potential extreme values in the data.

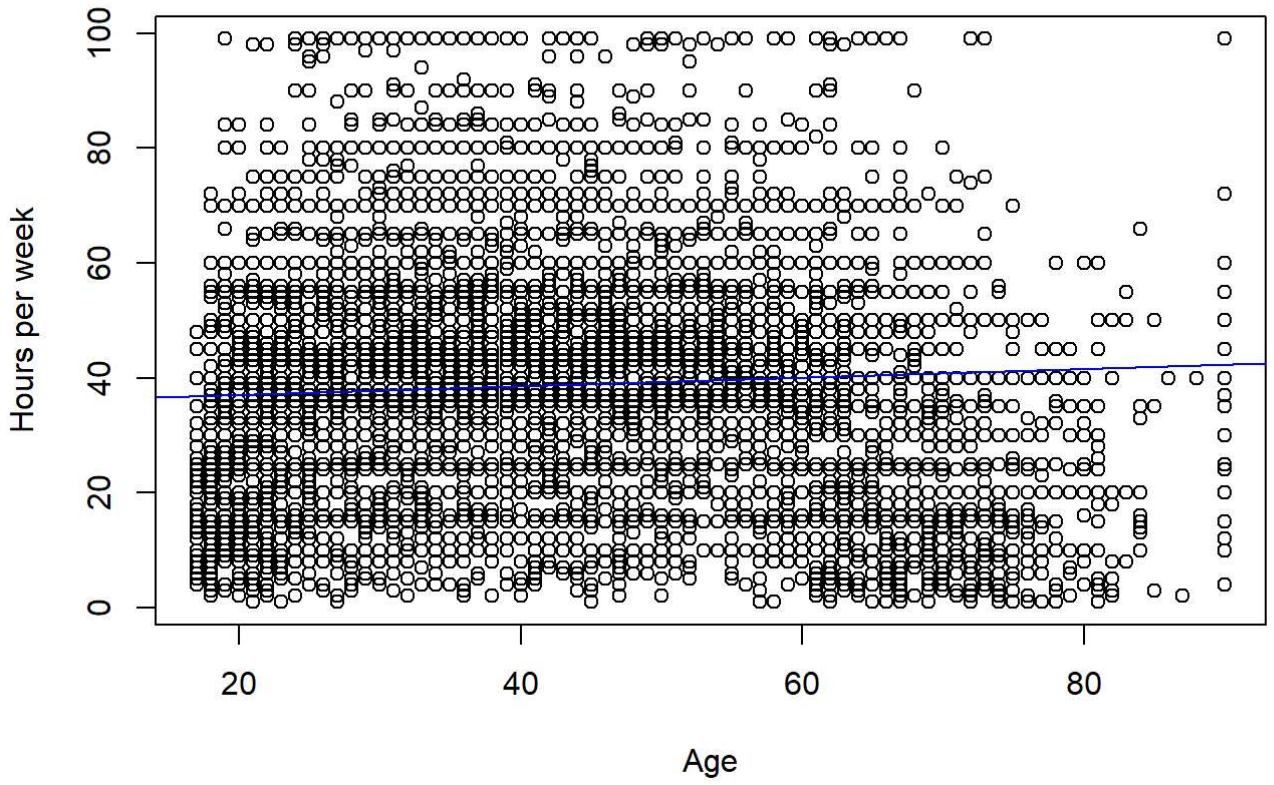
For central tendency (mean vs. median): You can compare the mean and median values obtained from the summary statistics to assess whether the distribution is symmetric or skewed. If the mean and median are close, the distribution tends to be symmetric. If they are significantly different, it may indicate skewness.

For spread (quartiles): The quartiles obtained from the summary statistics help you understand the range and spread of the data. You can compare them to the box plot to visually confirm the spread and identify potential outliers.

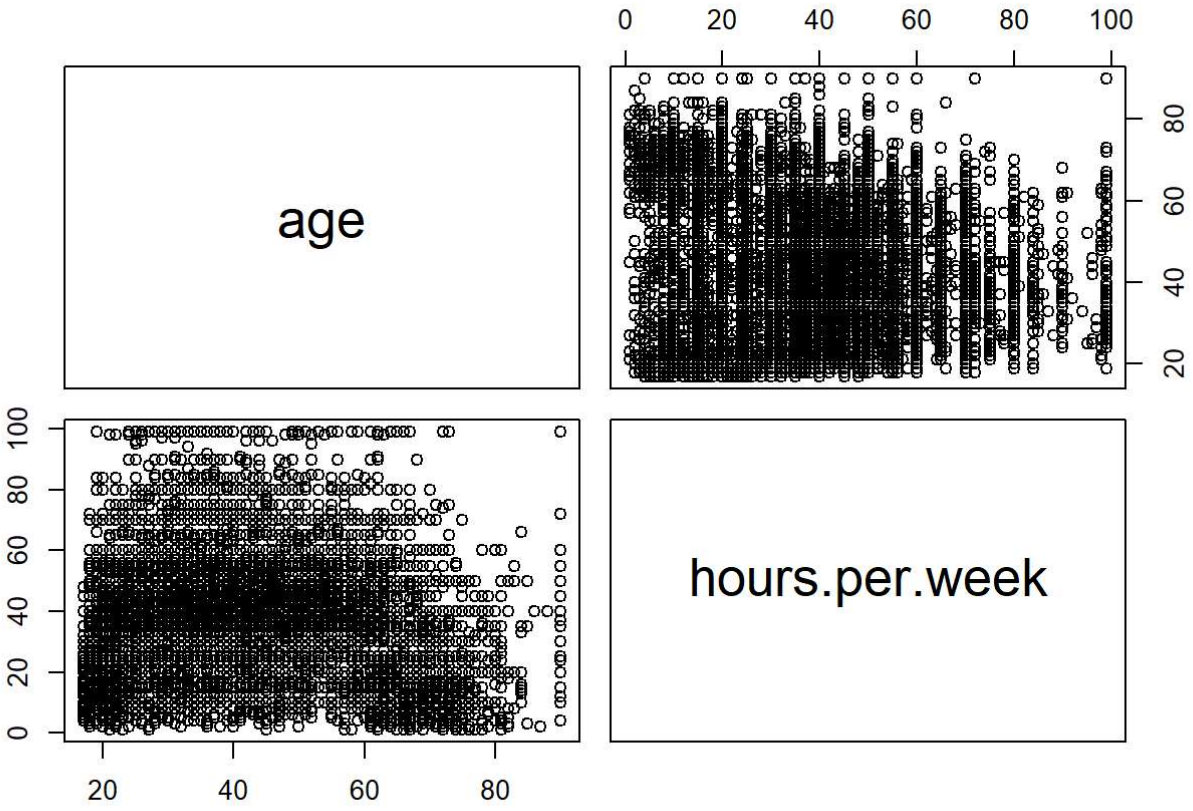
For skewness: Histograms can help visually identify whether the data is positively skewed (tail on the right) or negatively skewed (tail on the left). Skewness assumptions can be compared to the shape of the histogram. By combining the numerical summary statistics with visualizations like histograms and box plots, you can gain a more comprehensive understanding of the data's distribution and shape and validate or refine your assumptions about its characteristics

#Problem1(c)

```
# Create a scatterplot matrix of numerical variables
plot(adult_data$age,adult_data$hours.per.week,xlab="Age",ylab="Hours per week")
abline(lm(adult_data$age~adult_data$hours.per.week),col="blue")
```




```
numeric_vars <- adult_data[,c("age","hours.per.week")]
pairs(numeric_vars)
```



Explanation:Box plots help you identify the central tendency, spread, and presence of outliers in the data. They also provide insights into the distribution shape.

By comparing the summary statistics and visualizations of the two variables, you can assess their shapes and make observations about their central tendency, spread, skewness, and potential outliers.

We are able to visualize the way of Box plots provide a quick visual summary of the variability of values in a dataset. They show the median, upper and lower quartiles, minimum and maximum values, and any outliers in the dataset. Outliers can reveal mistakes or unusual occurrences in data

The view provided by box plots (box-and-whisker plots) offers several insights and information that can be challenging to discern when looking solely at distributions or histograms. Here are some key advantages of using box plots for data visualization:

Outliers Identification: Box plots are particularly effective at identifying potential outliers in the data. Outliers are data points that significantly differ from the bulk of the data. In a box plot, outliers are typically shown as individual points beyond the whiskers, making them visually distinct. Detecting outliers is more challenging when examining distributions or histograms alone.

Central Tendency and Spread: Box plots provide a clear representation of the central tendency (median) and the spread of the data (interquartile range). This information is easily discernible from the position of the box (median) and the length of the box (IQR) without the need to calculate these values separately.

Skewness: While histograms can indicate skewness in the data, box plots offer a quick visual assessment of skewness based on the box's orientation relative to the median. If the box is symmetrically distributed around the median, it suggests a symmetric distribution. A skewed distribution can be identified when the box is shifted towards one end.

Comparative Analysis: Box plots are useful for comparing the distributions of multiple variables side by side. When you have multiple box plots in the same visualization, it becomes easier to compare the central tendency, spread, and presence of outliers across different variables or groups.

Robustness to Extreme Values: Box plots are less sensitive to extreme values (outliers) than histograms. Outliers may not distort the shape of the box plot as much as they can impact the appearance of a histogram. This robustness makes it easier to focus on the central distribution.

Summary Information: Box plots offer a concise summary of key statistics, including quartiles (25th, 50th, and 75th percentiles), median, and potential outliers, all in a single view. This is valuable for quick exploratory data analysis

#Problem1(d)

```
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
```

```
# Count the occurrences of each category in the categorical variable
category_counts <- adult_data %>%
  group_by(workclass) %>%
  count()
category_counts1 <- adult_data %>%
  group_by(education) %>%
  count()
category_counts2 <- adult_data %>%
  group_by(marital.status) %>%
  count()
category_counts3 <- adult_data %>%
  group_by(occupation) %>%
  count()
category_counts4 <- adult_data %>%
  group_by(relationship) %>%
  count()
category_counts5 <- adult_data %>%
  group_by(race) %>%
  count()
category_counts6 <- adult_data %>%
  group_by(sex) %>%
  count()
category_counts7 <- adult_data %>%
  group_by(native.country) %>%
  count()

category_counts8 <- adult_data %>%
  group_by(income.bracket) %>%
  count()# Print the result

print(category_counts)
```

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

```
print(category_counts1)
```

```
## # A tibble: 16 × 2
## # Groups:   education [16]
##    education           n
##    <chr>           <int>
##  1 " 10th"           933
##  2 " 11th"          1175
##  3 " 12th"           433
##  4 " 1st-4th"        168
##  5 " 5th-6th"        333
##  6 " 7th-8th"        646
##  7 " 9th"            514
##  8 " Assoc-acdm"    1067
##  9 " Assoc-voc"     1382
## 10 " Bachelors"     5355
## 11 " Doctorate"      413
## 12 " HS-grad"      10501
## 13 " Masters"       1723
## 14 " Preschool"       51
## 15 " Prof-school"    576
## 16 " Some-college"  7291
```

```
print(category_counts2)
```

```
## # A tibble: 7 × 2
## # Groups:   marital.status [7]
##   marital.status              n
##   <chr>                   <int>
## 1 " Divorced"              4443
## 2 " Married-AF-spouse"       23
## 3 " Married-civ-spouse"   14976
## 4 " Married-spouse-absent"  418
## 5 " Never-married"        10683
## 6 " Separated"             1025
## 7 " Widowed"                993
```

```
print(category_counts3)
```

```
## # A tibble: 15 × 2
## # Groups:   occupation [15]
##    occupation              n
##    <chr>               <int>
##  1 " ?"                 1843
##  2 " Adm-clerical"      3770
##  3 " Armed-Forces"         9
##  4 " Craft-repair"      4099
##  5 " Exec-managerial"   4066
##  6 " Farming-fishing"    994
##  7 " Handlers-cleaners" 1370
##  8 " Machine-op-inspct" 2002
##  9 " Other-service"     3295
## 10 " Priv-house-serv"    149
## 11 " Prof-specialty"    4140
## 12 " Protective-serv"    649
## 13 " Sales"             3650
## 14 " Tech-support"       928
## 15 " Transport-moving"  1597
```

```
print(category_counts4)
```

```
## # A tibble: 6 × 2
## # Groups:   relationship [6]
##   relationship          n
##   <chr>             <int>
## 1 " Husband"        13193
## 2 " Not-in-family"   8305
## 3 " Other-relative"   981
## 4 " Own-child"       5068
## 5 " Unmarried"       3446
## 6 " Wife"            1568
```

```
print(category_counts5)
```

```
## # A tibble: 5 × 2
## # Groups:   race [5]
##   race                    n
##   <chr>               <int>
## 1 " Amer-Indian-Eskimo"   311
## 2 " Asian-Pac-Islander"  1039
## 3 " Black"               3124
## 4 " Other"                271
## 5 " White"              27816
```

```
print(category_counts6)
```

```
## # A tibble: 2 × 2
## # Groups:   sex [2]
##   sex           n
##   <chr>     <int>
## 1 " Female" 10771
## 2 " Male"   21790
```

```
print(category_counts7)
```

```
## # A tibble: 42 × 2
## # Groups:   native.country [42]
##    native.country            n
##    <chr>                 <int>
##  1 " ?"                    583
##  2 " Cambodia"              19
##  3 " Canada"               121
##  4 " China"                 75
##  5 " Columbia"              59
##  6 " Cuba"                  95
##  7 " Dominican-Republic"    70
##  8 " Ecuador"               28
##  9 " El-Salvador"          106
## 10 " England"               90
## # ℹ 32 more rows
```

```
print(category_counts8)
```

```
## # A tibble: 2 × 2
## # Groups:   income.bracket [2]
##   income.bracket     n
##   <chr>          <int>
## 1 " <=50K"       24720
## 2 " >50K"         7841
```

Explanation:

#Problem1(e)

```
library(ggplot2)
library(tidyr)

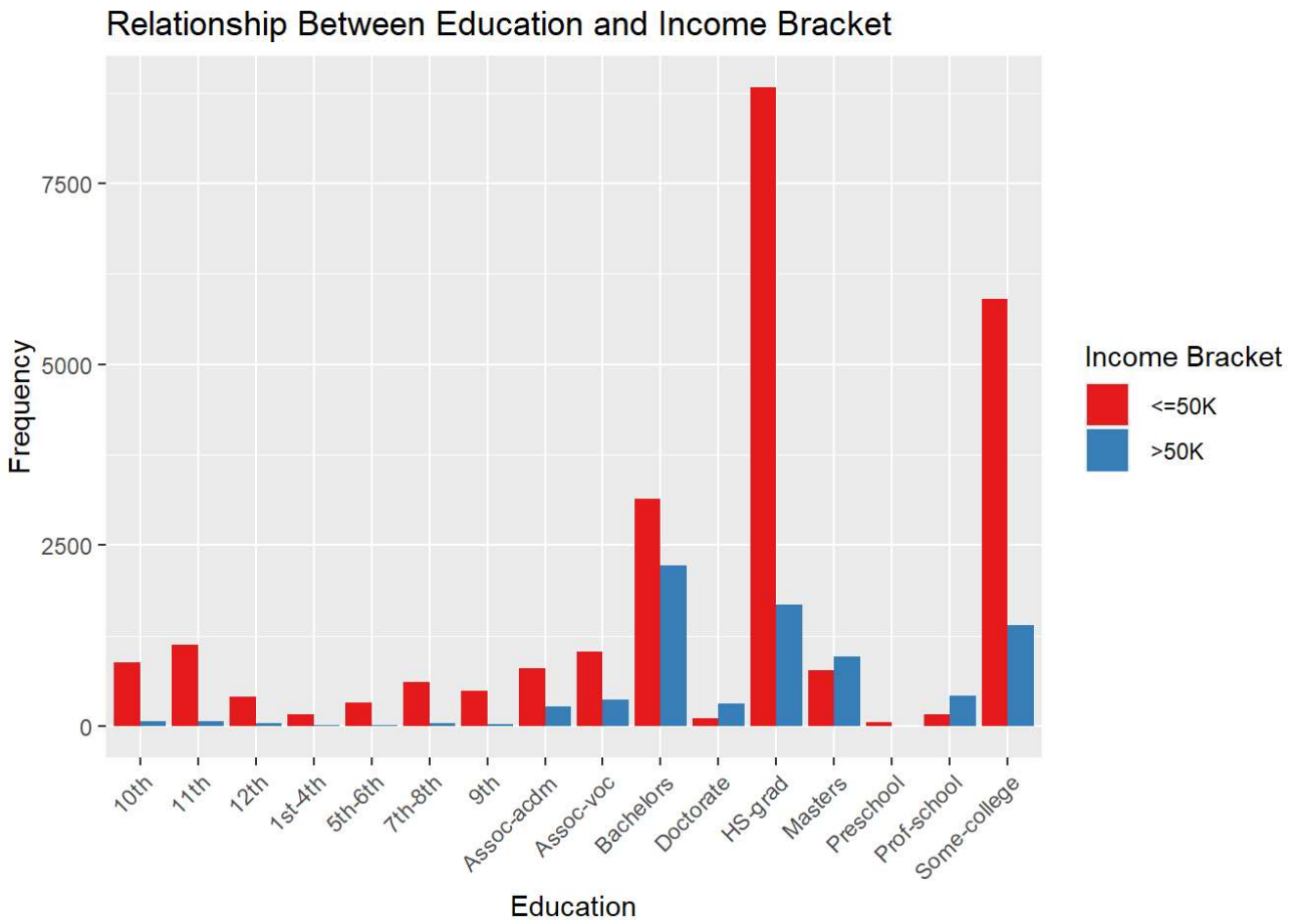
# Create a cross-tabulation of marital status and income bracket
cross_tab <- table(adult_data$education, adult_data$income.bracket)

# Create a bar plot to visualize the relationship
bar_plot <- as.data.frame(cross_tab) %>%
  ggplot(aes(x = Var1, y = Freq, fill = Var2)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(
    x = "Education",
    y = "Frequency",
    fill = "Income Bracket",
    title = "Relationship Between Education and Income Bracket"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_fill_brewer(palette = "Set1")

# Print the cross-tabulation and the bar plot
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
```

```
print(bar_plot)
```



Explanation: To explain the relationship between the values of the two categorical variables, you can observe the patterns in the stacked bar plot:

Look for patterns in the distribution of categories within each level of the other category. Are certain categories more prevalent when the other category takes on specific values? Assess whether there are significant differences in the frequency of combinations. Are there combinations that occur more frequently than others? Check for any trends or variations in the relationship. Are there any categories that stand out in terms of their association with specific values of the other category? By examining the visualization and considering the patterns and frequencies, you can draw conclusions about the relationship between the two categorical variables and identify any interesting or significant associations between their values

#Problem 2(a)

```
# Read population data for even years
even_years_data <- read.csv("population_even.csv")

# Read population data for odd years
odd_years_data <- read.csv("population_odd.csv")
library(dplyr)


# Merge the two data frames based on "State"
merged_data <- merge(even_years_data, odd_years_data, by = "STATE", all = TRUE)
merged_data
```

```
##    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
## 7      9          Connecticut         3579114         3594547         3594524
## 8     10             Delaware          899593          915179          932487
## 9     11 District of Columbia          605226          634924          662328
## 10    12              Florida        18845537        19297822        19845911
## 11    13              Georgia         9711881         9901430        10067278
## 12    15               Hawaii         1363963         1394804         1414538
## 13    16                Idaho         1570746         1595324         1631112
## 14    17             Illinois        12840503        12882510        12884493
## 15    18              Indiana         6490432         6537703         6593644
## 16    19                 Iowa         3050745         3076190         3109350
## 17    20               Kansas         2858190         2885257         2900475
## 18    21             Kentucky         4348181         4386346         4414349
## 19    22            Louisiana         4544532         4600972         4644013
## 20    23                Maine         1327629         1327729         1330513
## 21    24             Maryland         5788645         5886992         5957283
## 22    25        Massachusetts         6566307         6663005         6762596
## 23    26             Michigan         9877510         9897145         9929848
## 24    27            Minnesota         5310828         5376643         5451079
## 25    28          Mississippi         2970548         2983816         2990468
## 26    29             Missouri         5995974         6024367         6056202
## 27    30              Montana          990697         1003783         1021869
## 28    31             Nebraska         1829542         1853303         1879321
## 29    32               Nevada         2702405         2743996         2817628
## 30    33        New Hampshire         1316762         1324232         1333341
## 31    34           New Jersey         8799446         8844942         8864525
## 32    35           New Mexico         2064552         2087309         2089568
## 33    36             New York        19399878        19572932        19651049
## 34    37       North Carolina         9574323         9749476         9932887
## 35    38         North Dakota          674715          701176          737401
## 36    39                 Ohio        11539336        11548923        11602700
## 37    40             Oklahoma         3759944         3818814         3878187
## 38    41               Oregon         3837491         3899001         3963244
## 39    42         Pennsylvania        12711160        12767118        12788313
## 40    44         Rhode Island         1053959         1054621         1055936
## 41    45       South Carolina         4635649         4717354         4823617
## 42    46         South Dakota          816166          833566          849129
## 43    47            Tennessee         6355311         6453898         6541223
## 44    48                Texas        25241971        26084481        26964333
## 45    49                 Utah         2775332         2853375         2936879
## 46    50              Vermont          625879          626090          625214
## 47    51             Virginia         8023699         8185080         8310993
## 48    53           Washington         6742830         6897058         7054655
## 49    54        West Virginia         1854239         1856872         1849489
## 50    55            Wisconsin         5690475         5719960         5751525
## 51    56              Wyoming          564487          576305          582531
```

```
## 52    72          Puerto Rico         3721525         3634488         3534874
##    POPESTIMATE2016 POPESTIMATE2018               NAME.y POPESTIMATE2011
## 1          4863525         4887681              Alabama         4799069
## 2           741456          735139               Alaska          722128
## 3          6941072         7158024              Arizona              NA
## 4          2989918         3009733             Arkansas         2940667
## 5         39167117        39461588           California        37638369
## 6          5539215         5691287             Colorado         5121108
## 7          3578141         3571520          Connecticut         3588283
## 8           948921          965479             Delaware          907381
## 9           685815          701547 District of Columbia          619800
## 10        20613477        21244317              Florida        19053237
## 11        10301890        10511131              Georgia         9802431
## 12         1427559         1420593               Hawaii         1379329
## 13         1682380         1750536                Idaho         1583910
## 14        12820527        12723071             Illinois        12867454
## 15         6634304         6695497              Indiana         6516528
## 16         3131371         3148618                 Iowa         3066336
## 17         2910844         2911359               Kansas         2869225
## 18         4438182         4461153             Kentucky         4369821
## 19         4678135         4659690            Louisiana         4575625
## 20         1331317         1339057                Maine         1328284
## 21         6003323         6035802             Maryland         5839419
## 22         6823608         6882635        Massachusetts         6613583
## 23         9950571         9984072             Michigan         9882412
## 24         5522744         5606249            Minnesota         5346143
## 25         2987938         2981020          Mississippi         2978731
## 26         6087135         6121623             Missouri         6010275
## 27         1040859         1060665              Montana          997316
## 28         1905616         1925614             Nebraska         1840672
## 29         2917563         3027341               Nevada         2712730
## 30         1342307         1353465        New Hampshire         1320202
## 31         8870827         8886025           New Jersey         8828117
## 32         2091630         2092741           New Mexico         2080450
## 33        19633428        19530351             New York        19499241
## 34        10154788        10381615       North Carolina         9657592
## 35          754434          758080         North Dakota          685225
## 36        11634370        11676341                 Ohio        11544663
## 37         3926331         3940235             Oklahoma         3788379
## 38         4089976         4181886               Oregon         3872036
## 39        12782275        12800922         Pennsylvania        12745815
## 40         1056770         1058287         Rhode Island         1053649
## 41         4957968         5084156       South Carolina         4671994
## 42          862996          878698         South Dakota          823579
## 43         6646010         6771631            Tennessee         6399291
## 44        27914410        28628666                Texas        25645629
## 45         3041868         3153550                 Utah         2814384
## 46          623657          624358              Vermont          627049
## 47         8410106         8501286             Virginia         8101155
## 48         7294771         7523869           Washington         6826627
## 49         1831023         1804291        West Virginia         1856301
## 50         5772628         5807406            Wisconsin         5705288
```

```
## 51          584215          577601             Wyoming          567299
## 52         3406672         3193354          Puerto Rico         3678732
##    POPESTIMATE2013 POPESTIMATE2015 POPESTIMATE2017 POPESTIMATE2019
## 1          4830081         4852347         4874486         4903185
## 2           737068          737498          739700          731545
## 3          6632764         6829676         7044008         7278717
## 4          2959400         2978048         3001345         3017804
## 5         38260787        38918045        39358497        39512223
## 6          5269035         5450623         5611885         5758736
## 7          3594841         3587122         3573297         3565287
## 8           923576          941252          956823          973764
## 9           650581          675400          694906          705749
## 10        19545621        20209042        20963613        21477737
## 11         9972479        10178447        10410330        10617423
## 12         1408243         1422052         1424393         1415872
## 13         1611206              NA         1717715         1787065
## 14        12895129        12858913        12778828        12671821
## 15         6568713         6608422         6658078         6732219
## 16         3092997         3120960         3141550         3155070
## 17         2893212         2909011         2908718         2913314
## 18         4404659         4425976         4452268         4467673
## 19         4624527         4664628         4670560         4648794
## 20         1328009         1328262         1334612         1344212
## 21         5923188         5985562         6023868         6045680
## 22         6713315         6794228         6859789         6892503
## 23         9913065         9931715         9973114         9986857
## 24         5413479         5482032         5566230         5639632
## 25         2988711         2988471         2988510         2976149
## 26         6040715         6071732         6106670         6137428
## 27         1013569         1030475              NA         1068778
## 28         1865279         1891277         1915947         1934408
## 29         2775970         2866939         2969905         3080156
## 30         1326622         1336350         1348787         1359711
## 31         8856972         8867949         8885525         8882190
## 32         2092273         2089291         2091784         2096829
## 33        19624447        19654666        19589572        19453561
## 34         9843336        10031646        10268233        10488084
## 35          722036          754066          754942          762062
## 36              NA        11617527        11659650        11689100
## 37         3853214         3909500         3931316         3956971
## 38         3922468         4015792         4143625         4217737
## 39        12776309        12784826        12787641        12801989
## 40         1055081         1056065         1055673         1059361
## 41         4764080         4891938         5021268         5148714
## 42          842316          853988          872868          884659
## 43         6494340         6591170         6708799         6829174
## 44        26480266        27470056        28295273        28995881
## 45         2897640         2981835         3101042         3205958
## 46          626210          625216          624344          623989
## 47         8252427         8361808         8463587         8535519
## 48         6963985         7163657         7423362         7614893
## 49         1853914         1842050         1817004         1792147
```

```
## 50         5736754         5760940         5790186              NA
## 51          582122          585613          578931          578759
## 52         3593077         3473232         3325286         3193694
```

```
# Remove duplicate state ID column if it exists
merged_data <- merged_data %>%
  dplyr::select(-NAME.y)
merged_data
```

```
##    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
## 7      9         Connecticut         3579114         3594547         3594524
## 8     10            Delaware          899593          915179          932487
## 9     11 District of Columbia          605226          634924          662328
## 10    12             Florida        18845537        19297822        19845911
## 11    13             Georgia         9711881         9901430        10067278
## 12    15              Hawaii         1363963         1394804         1414538
## 13    16               Idaho         1570746         1595324         1631112
## 14    17            Illinois        12840503        12882510        12884493
## 15    18             Indiana         6490432         6537703         6593644
## 16    19                Iowa         3050745         3076190         3109350
## 17    20              Kansas         2858190         2885257         2900475
## 18    21            Kentucky         4348181         4386346         4414349
## 19    22           Louisiana         4544532         4600972         4644013
## 20    23               Maine         1327629         1327729         1330513
## 21    24            Maryland         5788645         5886992         5957283
## 22    25       Massachusetts         6566307         6663005         6762596
## 23    26            Michigan         9877510         9897145         9929848
## 24    27           Minnesota         5310828         5376643         5451079
## 25    28         Mississippi         2970548         2983816         2990468
## 26    29            Missouri         5995974         6024367         6056202
## 27    30             Montana          990697         1003783         1021869
## 28    31            Nebraska         1829542         1853303         1879321
## 29    32              Nevada         2702405         2743996         2817628
## 30    33       New Hampshire         1316762         1324232         1333341
## 31    34          New Jersey         8799446         8844942         8864525
## 32    35          New Mexico         2064552         2087309         2089568
## 33    36            New York        19399878        19572932        19651049
## 34    37      North Carolina         9574323         9749476         9932887
## 35    38        North Dakota          674715          701176          737401
## 36    39                Ohio        11539336        11548923        11602700
## 37    40            Oklahoma         3759944         3818814         3878187
## 38    41              Oregon         3837491         3899001         3963244
## 39    42        Pennsylvania        12711160        12767118        12788313
## 40    44        Rhode Island         1053959         1054621         1055936
## 41    45      South Carolina         4635649         4717354         4823617
## 42    46        South Dakota          816166          833566          849129
## 43    47           Tennessee         6355311         6453898         6541223
## 44    48               Texas        25241971        26084481        26964333
## 45    49                Utah         2775332         2853375         2936879
## 46    50             Vermont          625879          626090          625214
## 47    51            Virginia         8023699         8185080         8310993
## 48    53          Washington         6742830         6897058         7054655
## 49    54       West Virginia         1854239         1856872         1849489
## 50    55           Wisconsin         5690475         5719960         5751525
## 51    56             Wyoming          564487          576305          582531
```

```
## 52    72          Puerto Rico         3721525         3634488         3534874
##    POPESTIMATE2016 POPESTIMATE2018 POPESTIMATE2011 POPESTIMATE2013
## 1          4863525         4887681         4799069         4830081
## 2           741456          735139          722128          737068
## 3          6941072         7158024              NA         6632764
## 4          2989918         3009733         2940667         2959400
## 5         39167117        39461588        37638369        38260787
## 6          5539215         5691287         5121108         5269035
## 7          3578141         3571520         3588283         3594841
## 8           948921          965479          907381          923576
## 9           685815          701547          619800          650581
## 10        20613477        21244317        19053237        19545621
## 11        10301890        10511131         9802431         9972479
## 12         1427559         1420593         1379329         1408243
## 13         1682380         1750536         1583910         1611206
## 14        12820527        12723071        12867454        12895129
## 15         6634304         6695497         6516528         6568713
## 16         3131371         3148618         3066336         3092997
## 17         2910844         2911359         2869225         2893212
## 18         4438182         4461153         4369821         4404659
## 19         4678135         4659690         4575625         4624527
## 20         1331317         1339057         1328284         1328009
## 21         6003323         6035802         5839419         5923188
## 22         6823608         6882635         6613583         6713315
## 23         9950571         9984072         9882412         9913065
## 24         5522744         5606249         5346143         5413479
## 25         2987938         2981020         2978731         2988711
## 26         6087135         6121623         6010275         6040715
## 27         1040859         1060665          997316         1013569
## 28         1905616         1925614         1840672         1865279
## 29         2917563         3027341         2712730         2775970
## 30         1342307         1353465         1320202         1326622
## 31         8870827         8886025         8828117         8856972
## 32         2091630         2092741         2080450         2092273
## 33        19633428        19530351        19499241        19624447
## 34        10154788        10381615         9657592         9843336
## 35          754434          758080          685225          722036
## 36        11634370        11676341        11544663              NA
## 37         3926331         3940235         3788379         3853214
## 38         4089976         4181886         3872036         3922468
## 39        12782275        12800922        12745815        12776309
## 40         1056770         1058287         1053649         1055081
## 41         4957968         5084156         4671994         4764080
## 42          862996          878698          823579          842316
## 43         6646010         6771631         6399291         6494340
## 44        27914410        28628666        25645629        26480266
## 45         3041868         3153550         2814384         2897640
## 46          623657          624358          627049          626210
## 47         8410106         8501286         8101155         8252427
## 48         7294771         7523869         6826627         6963985
## 49         1831023         1804291         1856301         1853914
## 50         5772628         5807406         5705288         5736754
```

```
## 51          584215          577601          567299          582122
## 52         3406672         3193354         3678732         3593077
##    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
## 7          3587122         3573297         3565287
## 8           941252          956823          973764
## 9           675400          694906          705749
## 10        20209042        20963613        21477737
## 11        10178447        10410330        10617423
## 12         1422052         1424393         1415872
## 13              NA         1717715         1787065
## 14        12858913        12778828        12671821
## 15         6608422         6658078         6732219
## 16         3120960         3141550         3155070
## 17         2909011         2908718         2913314
## 18         4425976         4452268         4467673
## 19         4664628         4670560         4648794
## 20         1328262         1334612         1344212
## 21         5985562         6023868         6045680
## 22         6794228         6859789         6892503
## 23         9931715         9973114         9986857
## 24         5482032         5566230         5639632
## 25         2988471         2988510         2976149
## 26         6071732         6106670         6137428
## 27         1030475              NA         1068778
## 28         1891277         1915947         1934408
## 29         2866939         2969905         3080156
## 30         1336350         1348787         1359711
## 31         8867949         8885525         8882190
## 32         2089291         2091784         2096829
## 33        19654666        19589572        19453561
## 34        10031646        10268233        10488084
## 35          754066          754942          762062
## 36        11617527        11659650        11689100
## 37         3909500         3931316         3956971
## 38         4015792         4143625         4217737
## 39        12784826        12787641        12801989
## 40         1056065         1055673         1059361
## 41         4891938         5021268         5148714
## 42          853988          872868          884659
## 43         6591170         6708799         6829174
## 44        27470056        28295273        28995881
## 45         2981835         3101042         3205958
## 46          625216          624344          623989
## 47         8361808         8463587         8535519
## 48         7163657         7423362         7614893
## 49         1842050         1817004         1792147
```

```
## 50         5760940         5790186              NA
## 51          585613          578931          578759
## 52         3473232         3325286         3193694
```

```
# Remove ".x" suffix from column names

head(merged_data)
```

```
##   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 POPESTIMATE2011 POPESTIMATE2013
## 1         4863525         4887681         4799069         4830081
## 2          741456          735139          722128          737068
## 3         6941072         7158024              NA         6632764
## 4         2989918         3009733         2940667         2959400
## 5        39167117        39461588        37638369        38260787
## 6         5539215         5691287         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
```

#problem 2(b)

```
# Rename columns to just the year number
new_col_names <- c("STATE","NAME", "2010", "2011", "2012", "2013", "2014", "2015", "2016", "201
7", "2018", "2019")
colnames(merged_data) <- new_col_names
new_col_names
```

```
##  [1] "STATE" "NAME"  "2010"  "2011"  "2012"  "2013"  "2014"  "2015"  "2016"
## [10] "2017"  "2018"  "2019"
```

```
# Show the first few rows of the cleaned-up data
head(merged_data)
```

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

#Problem 2(c)

```
replace_missing_with_avg <- function(x) {
  for (i in 2:(length(x) - 1)) {
    if (is.na(x[i])) {
      x[i] <- (x[i - 1] + x[i + 1]) / 2
    }
  }
  return(x)
}


# Apply the custom function to the entire dataset
merged_data[, 2:ncol(merged_data)] <- lapply(
  merged_data[, 2:ncol(merged_data)],
  function(x) replace_missing_with_avg(x)
)
merged_data
```

```
##    STATE                 NAME     2010     2011     2012     2013     2014
## 1      1              Alabama  4785437  4815588  4841799  4863525  4887681
## 2      2               Alaska   713910   730443   736283   741456   735139
## 3      4              Arizona  6407172  6554978  6730413  6941072  7158024
## 4      5             Arkansas  2921964  2952164  2967392  2989918  3009733
## 5      6           California 37319502 37948800 38596972 39167117 39461588
## 6      8             Colorado  5047349  5192647  5350101  5539215  5691287
## 7      9          Connecticut  3579114  3594547  3594524  3578141  3571520
## 8     10             Delaware   899593   915179   932487   948921   965479
## 9     11 District of Columbia   605226   634924   662328   685815   701547
## 10    12              Florida 18845537 19297822 19845911 20613477 21244317
## 11    13              Georgia  9711881  9901430 10067278 10301890 10511131
## 12    15               Hawaii  1363963  1394804  1414538  1427559  1420593
## 13    16                Idaho  1570746  1595324  1631112  1682380  1750536
## 14    17             Illinois 12840503 12882510 12884493 12820527 12723071
## 15    18              Indiana  6490432  6537703  6593644  6634304  6695497
## 16    19                 Iowa  3050745  3076190  3109350  3131371  3148618
## 17    20               Kansas  2858190  2885257  2900475  2910844  2911359
## 18    21             Kentucky  4348181  4386346  4414349  4438182  4461153
## 19    22            Louisiana  4544532  4600972  4644013  4678135  4659690
## 20    23                Maine  1327629  1327729  1330513  1331317  1339057
## 21    24             Maryland  5788645  5886992  5957283  6003323  6035802
## 22    25        Massachusetts  6566307  6663005  6762596  6823608  6882635
## 23    26             Michigan  9877510  9897145  9929848  9950571  9984072
## 24    27            Minnesota  5310828  5376643  5451079  5522744  5606249
## 25    28          Mississippi  2970548  2983816  2990468  2987938  2981020
## 26    29             Missouri  5995974  6024367  6056202  6087135  6121623
## 27    30              Montana   990697  1003783  1021869  1040859  1060665
## 28    31             Nebraska  1829542  1853303  1879321  1905616  1925614
## 29    32               Nevada  2702405  2743996  2817628  2917563  3027341
## 30    33        New Hampshire  1316762  1324232  1333341  1342307  1353465
## 31    34           New Jersey  8799446  8844942  8864525  8870827  8886025
## 32    35           New Mexico  2064552  2087309  2089568  2091630  2092741
## 33    36             New York 19399878 19572932 19651049 19633428 19530351
## 34    37       North Carolina  9574323  9749476  9932887 10154788 10381615
## 35    38         North Dakota   674715   701176   737401   754434   758080
## 36    39                 Ohio 11539336 11548923 11602700 11634370 11676341
## 37    40             Oklahoma  3759944  3818814  3878187  3926331  3940235
## 38    41               Oregon  3837491  3899001  3963244  4089976  4181886
## 39    42         Pennsylvania 12711160 12767118 12788313 12782275 12800922
## 40    44         Rhode Island  1053959  1054621  1055936  1056770  1058287
## 41    45       South Carolina  4635649  4717354  4823617  4957968  5084156
## 42    46         South Dakota   816166   833566   849129   862996   878698
## 43    47            Tennessee  6355311  6453898  6541223  6646010  6771631
## 44    48                Texas 25241971 26084481 26964333 27914410 28628666
## 45    49                 Utah  2775332  2853375  2936879  3041868  3153550
## 46    50              Vermont   625879   626090   625214   623657   624358
## 47    51             Virginia  8023699  8185080  8310993  8410106  8501286
## 48    53           Washington  6742830  6897058  7054655  7294771  7523869
## 49    54        West Virginia  1854239  1856872  1849489  1831023  1804291
## 50    55            Wisconsin  5690475  5719960  5751525  5772628  5807406
## 51    56              Wyoming   564487   576305   582531   584215   577601
```

```
## 52    72          Puerto Rico  3721525  3634488  3534874  3406672  3193354
##        2015     2016     2017     2018     2019
## 1   4799069  4830081  4852347  4874486  4903185
## 2    722128   737068   737498   739700   731545
## 3   1831398  6632764  6829676  7044008  7278717
## 4   2940667  2959400  2978048  3001345  3017804
## 5  37638369 38260787 38918045 39358497 39512223
## 6   5121108  5269035  5450623  5611885  5758736
## 7   3588283  3594841  3587122  3573297  3565287
## 8    907381   923576   941252   956823   973764
## 9    619800   650581   675400   694906   705749
## 10 19053237 19545621 20209042 20963613 21477737
## 11  9802431  9972479 10178447 10410330 10617423
## 12  1379329  1408243  1422052  1424393  1415872
## 13  1583910  1611206  7140483  1717715  1787065
## 14 12867454 12895129 12858913 12778828 12671821
## 15  6516528  6568713  6608422  6658078  6732219
## 16  3066336  3092997  3120960  3141550  3155070
## 17  2869225  2893212  2909011  2908718  2913314
## 18  4369821  4404659  4425976  4452268  4467673
## 19  4575625  4624527  4664628  4670560  4648794
## 20  1328284  1328009  1328262  1334612  1344212
## 21  5839419  5923188  5985562  6023868  6045680
## 22  6613583  6713315  6794228  6859789  6892503
## 23  9882412  9913065  9931715  9973114  9986857
## 24  5346143  5413479  5482032  5566230  5639632
## 25  2978731  2988711  2988471  2988510  2976149
## 26  6010275  6040715  6071732  6106670  6137428
## 27   997316  1013569  1030475  4011309  1068778
## 28  1840672  1865279  1891277  1915947  1934408
## 29  2712730  2775970  2866939  2969905  3080156
## 30  1320202  1326622  1336350  1348787  1359711
## 31  8828117  8856972  8867949  8885525  8882190
## 32  2080450  2092273  2089291  2091784  2096829
## 33 19499241 19624447 19654666 19589572 19453561
## 34  9657592  9843336 10031646 10268233 10488084
## 35   685225   722036   754066   754942   762062
## 36 11544663  2287625 11617527 11659650 11689100
## 37  3788379  3853214  3909500  3931316  3956971
## 38  3872036  3922468  4015792  4143625  4217737
## 39 12745815 12776309 12784826 12787641 12801989
## 40  1053649  1055081  1056065  1055673  1059361
## 41  4671994  4764080  4891938  5021268  5148714
## 42   823579   842316   853988   872868   884659
## 43  6399291  6494340  6591170  6708799  6829174
## 44 25645629 26480266 27470056 28295273 28995881
## 45  2814384  2897640  2981835  3101042  3205958
## 46   627049   626210   625216   624344   623989
## 47  8101155  8252427  8361808  8463587  8535519
## 48  6826627  6963985  7163657  7423362  7614893
## 49  1856301  1853914  1842050  1817004  1792147
## 50  5705288  5736754  5760940  5790186  1185453
```

```
## 51   567299   582122   585613   578931   578759
## 52  3678732  3593077  3473232  3325286  3193694
```

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ forcats   1.0.0     ✓ readr     2.1.4
## ✓ lubridate 1.9.3     ✓ stringr   1.5.0
## ✓ purrr     1.0.2     ✓ tibble    3.2.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

#problem 2(d)(A)

```
max_population_by_state <- merged_data %>%
  rowwise() %>%
  mutate(Max_Population = max(c_across(starts_with("20")), na.rm = TRUE)) %>%
  select(STATE, NAME,Max_Population)

max_population_by_state
```

```
## # A tibble: 52 × 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
##  7     9 Connecticut                 3594841
##  8    10 Delaware                     973764
##  9    11 District of Columbia         705749
## 10    12 Florida                    21477737
## # ℹ 42 more rows
```

```
head(max_population_by_state)
```

```
## # A tibble: 6 × 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
```

```
library(dplyr)
```

#problem 2(d)(b)

```
# Calculate the total population across all years for each state (row-wise)
total_population_by_state <- merged_data %>%
  rowwise() %>%
  mutate(Total_Population = sum(c_across(starts_with("20")), na.rm = TRUE)) %>%
  select(STATE,NAME, Total_Population)
total_population_by_state
```

```
## # A tibble: 52 × 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
##  7     9 Connecticut                 35826676
##  8    10 Delaware                     9364455
##  9    11 District of Columbia         6636276
## 10    12 Florida                    201096314
## # ℹ 42 more rows
```

```
# Show the result
head(total_population_by_state)
```

```
## # A tibble: 6 × 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
```

#Problem 2(E)

```
total_us_population_2018 <- sum(merged_data$`2018`, na.rm = TRUE)
head(total_us_population_2018)
```

```
## [1] 331269652
```

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



```r
#problem 3

# Reshape the data from wide to long format
merged_data_long <- pivot_longer(merged_data, cols = starts_with("20"),
                                 names_to = "Year", values_to = "Population")
merged_data_long
```

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

```
merged_data_long$Year <- as.numeric(merged_data_long$Year)

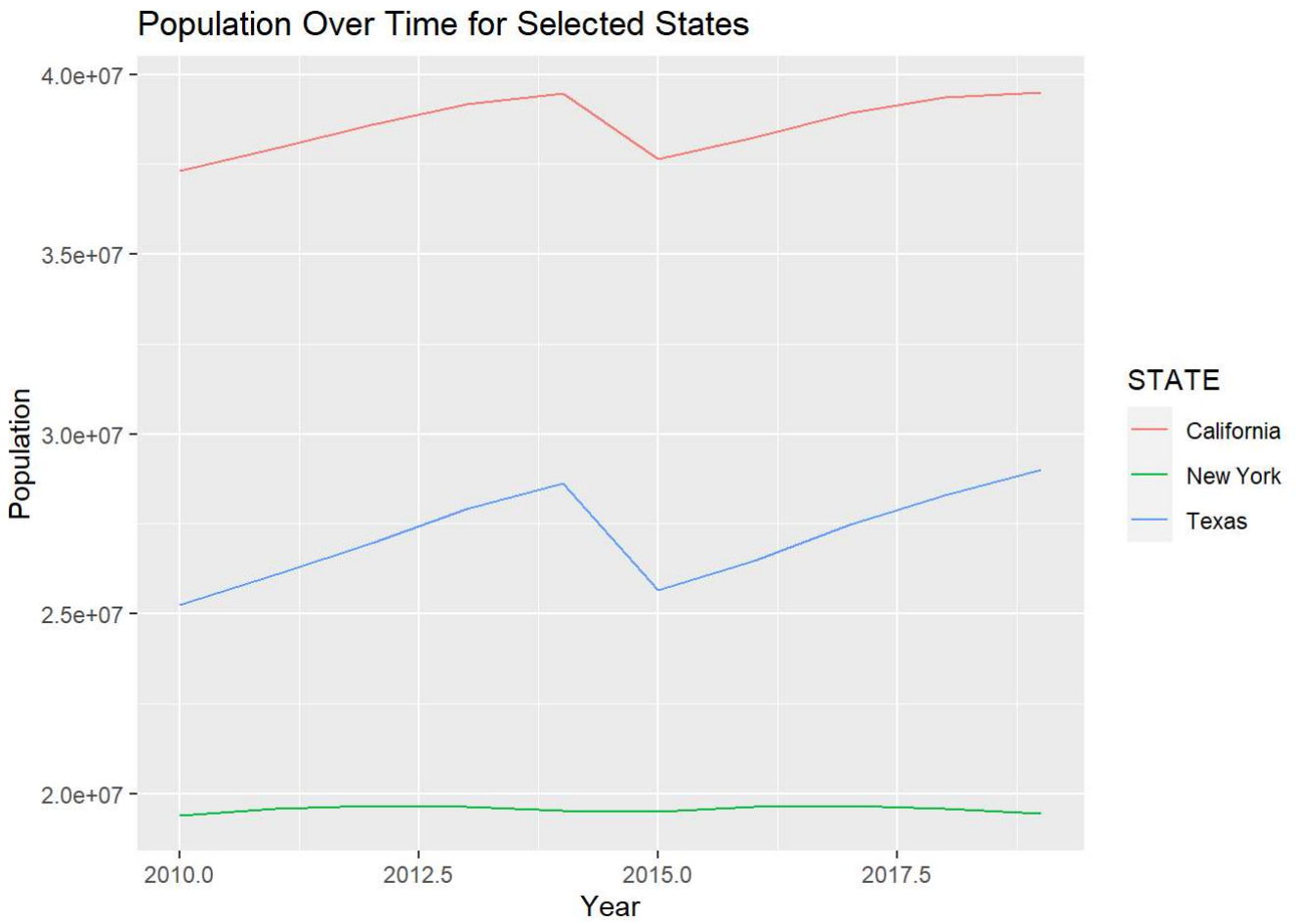
merged_data_long$Year
```

```
##   [1] 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014
##  [16] 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019
##  [31] 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014
##  [46] 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019
##  [61] 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014
##  [76] 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019
##  [91] 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014
## [106] 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019
## [121] 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014
## [136] 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019
## [151] 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014
## [166] 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019
## [181] 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014
## [196] 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019
## [211] 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014
## [226] 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019
## [241] 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014
## [256] 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019
## [271] 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014
## [286] 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019
## [301] 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014
## [316] 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019
## [331] 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014
## [346] 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019
## [361] 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014
## [376] 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019
## [391] 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014
## [406] 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019
## [421] 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014
## [436] 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019
## [451] 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014
## [466] 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019
## [481] 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014
## [496] 2015 2016 2017 2018 2019 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019
## [511] 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019
```

```
selected_states <- c("New York", "California", "Texas")  # Replace with your chosen states

plot_data <- merged_data_long %>%
  filter(NAME %in% selected_states)

ggplot(data = plot_data, aes(x = as.integer(Year), y = Population, color = NAME)) +
  geom_line() +
  labs(x = "Year", y = "Population", color = "STATE") +
  ggtitle("Population Over Time for Selected States")
```



#Problem 4 A. Describe two ways in which data can be dirty, and for each one, provide a potential solution??

```
Certainly! Data can become "dirty" or have a number of problems that reduce its usefulness. Here
are two typical problems with data quality and possible remedies in data science:

1.Absence of Data:

Problem: Missing data refers to a dataset that has one or more variables that are either unavail
able or insufficient. This may occur for a number of reasons, including incorrect data entry, su
rvey respondent non-response, or technical difficulties during data collecting.
Solution: Missing data can be addressed in a number of ways:
Imputation: Substituting estimated values derived from statistical techniques for missing value
s. Mean, median, and regression imputation are frequently used imputation techniques for numeric
al data. You can use the mode or strategies like the K-nearest neighbors (KNN) method for catego
rical data.

To lessen the possibility of missing data, make sure your data gathering procedures are robust.
This could involve making improvements to data entry procedures, survey design, and data validat
ion methods.

Analysis Methods: Some machine learning algorithms, such as tree-based algorithms like Random Fo
rests, can handle missing data directly. As an alternative, you might think about including data
imputation methods into your modeling procedure.

2.Inconsistent Data:

consistent formats, inconsistent data is produced. This may involve differences in date formats,
issues with units, or coding errors.
Solution: Data cleaning and standardization are necessary to address conflicting data:
Data cleaning involves finding and fixing problems in data entry such typos, wrong numbers, and
outliers. Inconsistencies can be found using tools like data validation tests and data profilin
g.

Data standardization: Assure that data are documented consistently. This could entail employing
standardized measuring units, converting all date formats to a single format, or developing codi
ng guidelines for categorical variables.

Data Transformation: Occasionally, you may need to modify the data to improve its consistency
```

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

B(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 is a data mining functionality.

Data Preparation: Choose the relevant columns that describe consumer purchases to prepare your d
ata. The columns should represent various facets of each client's purchasing activity, and the r
ows should be each individual consumer.

Choose whatever characteristics or characteristics of client purchases you want to take into acc
ount in the clustering analysis. The data may require preprocessing and transformation, such as
scaling or normalization.

Select an algorithm for clustering: Based on the characteristics of your data and the desired re
sult, choose an appropriate clustering procedure. K-Means, Hierarchical Clustering, and DBSCAN a
re examples of common clustering techniques.

How many clusters are there? Choose the number of clusters—in this case, five—that you want to m
ake. You can employ strategies like the elbow.

Establish the Number of Clusters: Choose the number of clusters (in this case, five) that you wi
sh to establish. The Elbow Method and Silhouette Score are two techniques you can use to figure
out how many clusters are ideal.

Apply Clustering: Use your data to run the selected clustering algorithm. Based on how they sho
p, each consumer will be put into one of the five clusters.

Interpretation: Examine the information to determine the traits of each cluster. Within each gro
up, look for trends and similarities in the purchasing habits of the customers.
```

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

Yes, you can predict whether a customer will buy milk based on what else they bought using a classification data mining functionality. This is a binary classification problem where you want to classify customers into two categories: "Will Buy Milk" or "Will Not Buy Milk" based on their purchase history. Here's how to approach it:

Data Mining Functionality: Classification

Data Preparation: Prepare your data by selecting the relevant columns that describe customer purchases. Each customer should be represented as a row, and the columns should represent different products or attributes of their purchase behavior.

Feature Selection and Engineering: Choose the features or attributes of customer purchases that you believe are relevant for predicting whether they will buy milk. These features can include the types of products purchased, the frequency of purchases, total spending, etc.

Data Splitting: Split your dataset into a training set and a testing set. The training set will be used to train the classification model, and the testing set will be used to evaluate its performance.

Select a Classification Algorithm: Choose a classification algorithm suitable for your data. Common classification algorithms include Logistic Regression, Decision Trees, Random Forests, Support Vector Machines, and Neural Networks.

Training the Model: Use the training data to train your chosen classification model. The model will learn the relationship between the selected features and the target variable (buying milk).

Model Evaluation: Evaluate the performance of your model using appropriate classification metrics such as accuracy, precision, recall, F1-score, and ROC-AUC on the testing data. This step will help you assess how well your model can predict whether a customer will buy milk.

Prediction: Once your model is trained and evaluated, you can use it to make predictions for new customers. Given a customer's purchase history, the model will predict whether they are likely to buy milk or not.

Interpretation and Insights: Analyze the model's predictions and gain insights into which features are most influential in predicting milk purchases. This can help you understand customer behavior better.

Model Improvement: Depending on the performance of your initial model, you may want to experiment with different algorithms or fine-tune hyperparameters to improve predictive accuracy.

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

We can use association rule mining, a data mining feature, to extract various sets of products t hat are frequently bought together from a dataset listing items in individual purchases. By spot ting patterns in the co-occurring goods in transactions, association rule mining enables you to learn which products are typically bought in pairs. Here's how to go about doing it:

Association Rule Mining is a feature of data mining.

Data Preparation: Make sure your data is properly organized, with each row indicating a single t ransaction of purchases and columns reflecting the items purchased or their identifiers.

Data Encoding: Transform your information into a binary format in which each column represents a different product and the values signify whether the product was present (1) or absent (0) durin g each transaction. This format is frequently called

Using the Apriori method, you can extract association rules from your data. By creating frequent itemsets—groups of items that regularly appear together—this algorithm creates association rule s.

Filter Rules: Use your specified minimum support and confidence criteria to sort the discovered association rules. Focusing on the most important associations is made easier by this stage.

Interpretation: Look at the association rules that were generated to find several groups of prod ucts that are frequently bought together. These rules are divided into two sections: antecedent (things purchased) and consequent (items likely to be purchased).

Post-processing: Depending on your unique business goals, you can further evaluate and refine th e discovered groupings of products. For marketing campaigns or recommendations, you might group products into product bundles, for instance.

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

C(a) Organizing the customers of a company according to education level??

In general, this is not regarded as a data mining task. It is more of an activity that organizes or sorts data. Customer organization by education level is a simple data manipulation task that doesn't require finding undiscovered patterns or generating predictions based on data.

C(b)Computing the total sales of a company??

Is it a Data Mining Task? No, computing the total sales of a company is a basic data aggregation task and not a data mining task. Data mining usually involves extracting patterns, insights, or predictive models from data, while calculating totals or aggregating values is a straightforward operation.

C(c)Sorting a student database according to identification numbers??

```
Is it a Data Mining Task? No, sorting a database based on identification numbers is a data manip
ulation task but not a data mining task. Data mining typically deals with more complex tasks lik
e pattern recognition, clustering, classification, or regression.
```

C(d)Predicting the outcomes of tossing a (fair) pair of dice??

```
forecasting the results of a fair pair of dice is not a data mining task; it is a statistical or
probabilistic task. While forecasting dice results is based on well-defined probability theory,
data mining often entails extracting patterns and knowledge from big databases.
```

C(e)Predicting the future stock price of a company using historical records??

```
This is a data mining endeavor, that much is true. Analysis of patterns and trends in previous s
tock price data is necessary to anticipate future prices when future stock prices are predicted
using historical data. Predictive analytics and time series forecasting, which are frequent data
mining jobs, fall under this category.
In conclusion, while some of the aforementioned jobs involve data manipulation or simple math, d
ata mining often entails more intricate procedures, such finding patterns, making predictions, o
r drawing conclusions from data.
```