Fundamentals of Data Science-Assignment 1

Sadiya Amreen- 2079690

2022-09-29

Problem 1 (15 points):

```
## Problem 1 (15 points):
##For this question, we will use the US census dataset from 1994, which is in adult.csv.
print(getwd())
```

```
## [1] "C:/Sadiya Studies/Data Science/DS441-Fundamts DS"
```

```
setwd("C:/Users/SADIYA/Downloads")
adult_data <- read.csv("adult.csv")
head(data)
```

```
##
## 1 function (..., list = character(), package = NULL, lib.loc = NULL,
## 2     verbose = getOption("verbose"), envir = .GlobalEnv, overwrite = TRUE)
## 3 {
## 4     fileExt <- function(x) {
## 5         db <- grepl("\\\\.[^.]+\\\\.(gz|bz2|xz)$", x)
## 6         ans <- sub(".*\\\\.", "", x)
```

```
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
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
```

```
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v stringr 1.4.1
## v tidyr   1.2.1     v forcats 0.5.2
## v readr   2.1.2
## -- Conflicts --------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(astsa)
library(ggplot2)
library(dplyr)
```

```
##a. First, we look at the summary statistics for all the variables. Based on those metrics, including t
##quartiles, compare two variables.
#What can you tell about their shape from these summaries?
```

```
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(adult_data$age)
```

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

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

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

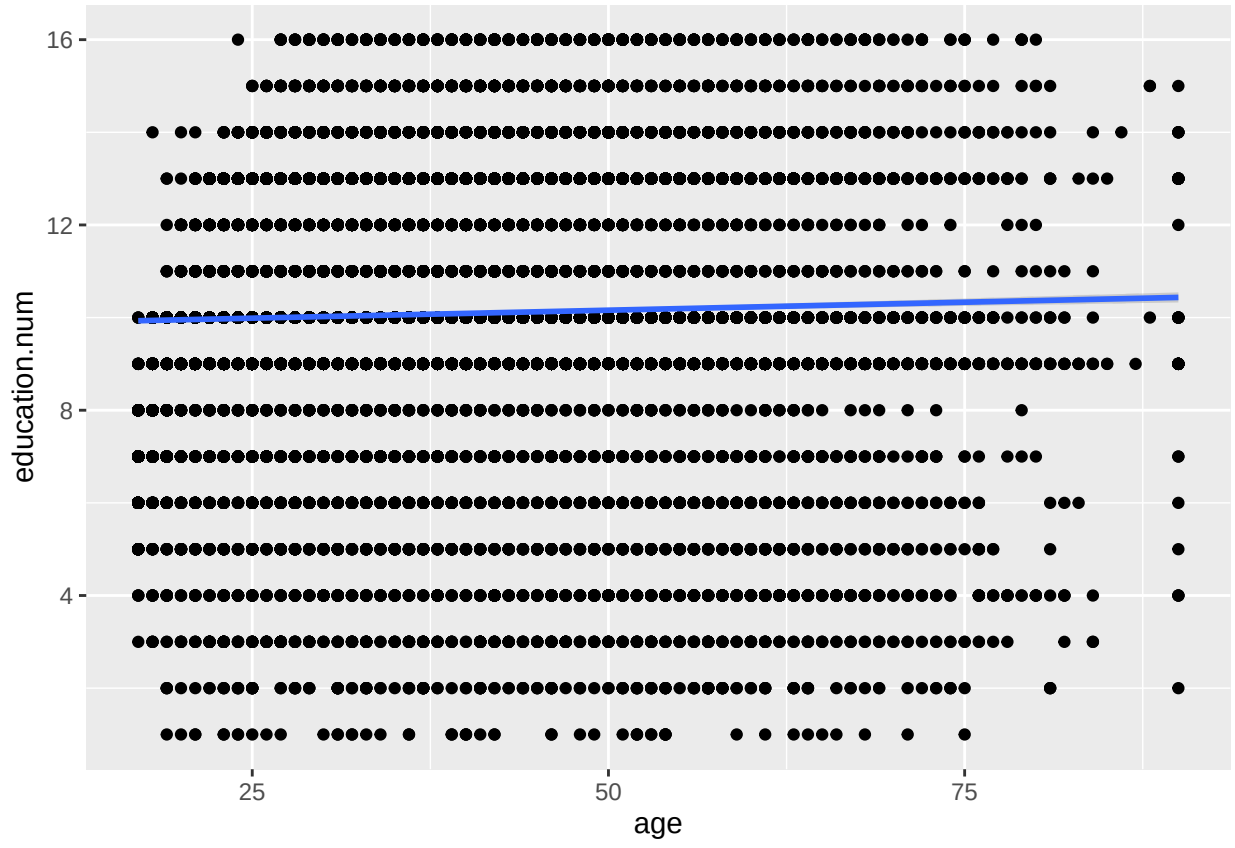
Inference:

As per Statistics, we can analyze shapes of a given dataset using summaries, the mean and the median both reflects the skewing, but the mean reflects the skewing the most. Generally for a given data,if the mean is less than the median, the shape is skewed to the left. Inversely, if the shape is skewed to the right, the median is less than the mean. So here, it is evident from the summaries of both #age and #education.num datasets that for respective datasets, mean > median , hence both shapes are Right skewed.

```
#b. Use a visualization to get a fine-grain comparison (you don't have to use QQ plots, though) of the
#distributions of those two variables.
```

```
ggplot(adult_data, aes(age, education.num)) + geom_point()+ geom_smooth(method=lm)
```

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




```
##Why did you choose the type of visualization that you chose?
  ##How do your part (a) assumptions compare to what you can see visually?
```

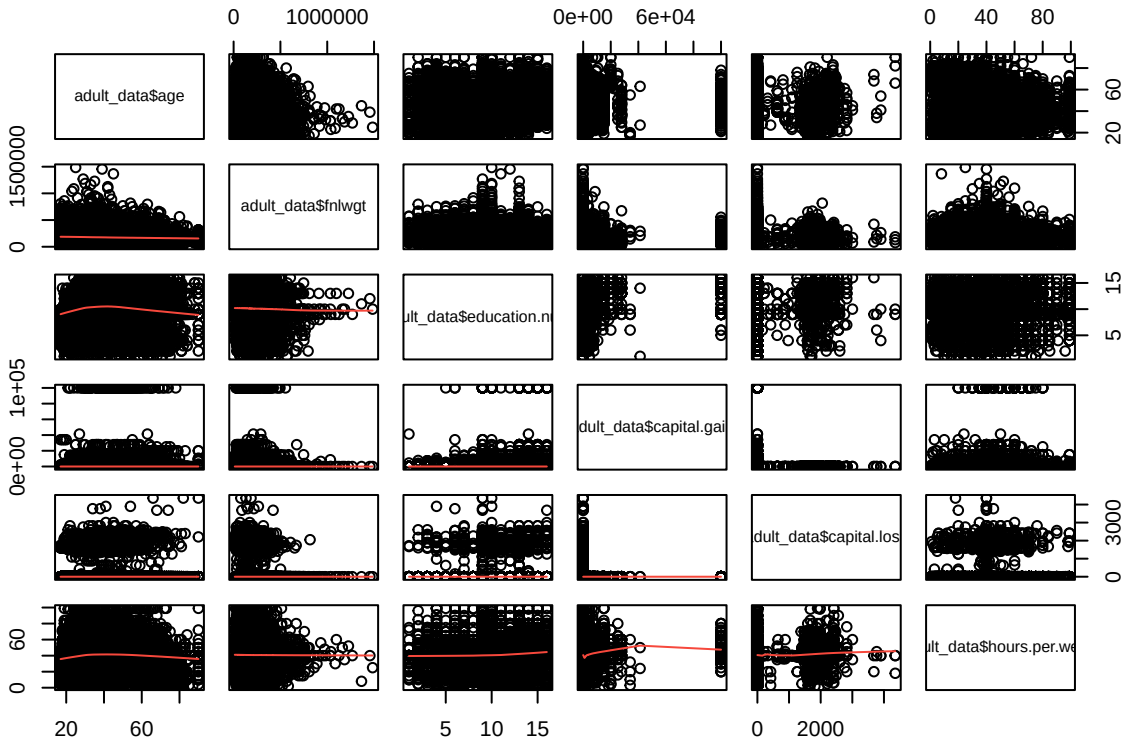
Inference:

i) Scatterplots vizualization is used here as the scatterplot [geom_point()] is most useful for displaying the relationship between two continuous variables. It can be used to compare continuous and categorical variable, or two categorical variables too. [geom_smooth()] helps the eye in seeing patterns even if there is the presence of overplotting. Used the geom_smooth() function for adding a regression line to our scatter plot and providing "method=lm" as an argument. We have set method=lm for lm stands as Linear Model, which plots a linear regression line.

ii) Our part (a) assumptions which were made on the shape of the curve for the data, regarding the skewess are similar to what we obtained here visually. It is pretty evident by the geom_smooth linear regression line that, both the dataset curves are right Skewed.

```
#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?

adult_data_num = select_if(adult_data,is.numeric)
pairs(~adult_data$age + adult_data$fnlwgt + adult_data$education.num + adult_data$capital.gain +
        adult_data$capital.loss + adult_data$ hours.per.week ,lower.panel = panel.smooth )
```



Inference:

Q)What does this view show you, that would be difficult to see looking at distributions?

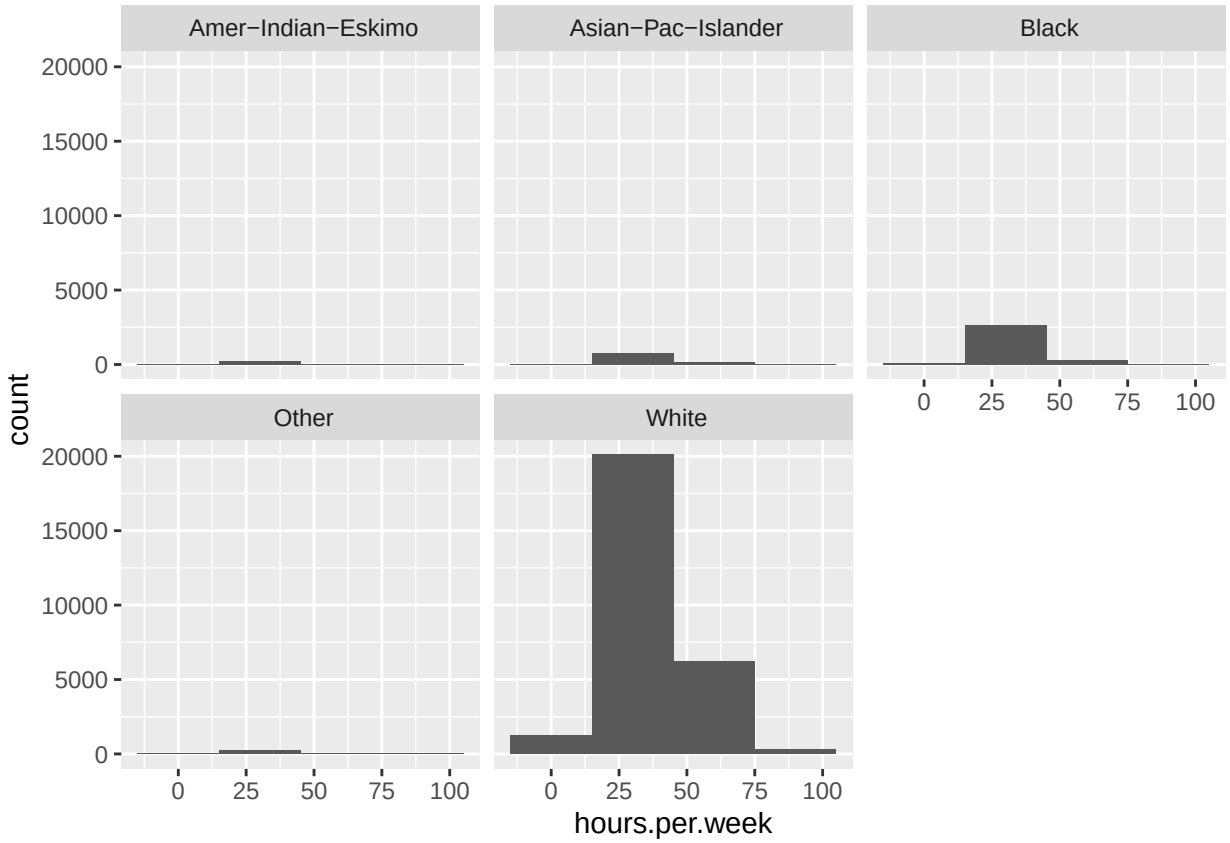
The Scatterplot matrix makes it easy to infer the correlations between the numeric variables which would have been diffcult to judge by just seeing at the distributions.

```
#d. These adult_data are a selection of US adults. It might not be a very balanced sample, though. Take
#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.


adult_data %>% group_by(adult_data$occupation) %>% summarise(n())
```

```
## # A tibble: 15 x 2
##    `adult_data$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
```

```
plots <-ggplot(adult_data ,aes(x=`hours.per.week`))+ geom_histogram(binwidth = 30)+facet_wrap(~race)
plots
```




```
##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
#categoricals.

table(adult_data$education, adult_data$marital.status)
```

```
##
##                  Divorced  Married-AF-spouse  Married-civ-spouse
##    10th               120                  0                 349
##    11th               130                  0                 354
##    12th                39                  0                 130
##    1st-4th             10                  0                  81
##    5th-6th             20                  0                 172
##    7th-8th             73                  0                 359
##    9th                 64                  0                 230
##    Assoc-acdm         203                  2                 460
##    Assoc-voc          234                  1                 689
##    Bachelors          546                  4                2768
##    Doctorate           33                  0                 286
##    HS-grad           1613                 13                4845
##    Masters            233                  0                1003
##    Preschool            1                  0                  20
##    Prof-school         55                  0                 412
##    Some-college      1069                  3                2818
##
##                  Married-spouse-absent  Never-married  Separated  Widowed
```

```
##    10th                 15          361         49         39
##    11th                 19          586         48         38
##    12th                  8          232         14         10
##    1st-4th              12           39          9         17
##    5th-6th              20           89         18         14
##    7th-8th              14          113         23         64
##    9th                   9          155         33         23
##    Assoc-acdm           12          337         30         23
##    Assoc-voc            13          362         42         41
##    Bachelors            68         1795         92         82
##    Doctorate             7           73          7          7
##    HS-grad             121         3089        406        414
##    Masters              17          404         25         41
##    Preschool             4           22          1          3
##    Prof-school           3           93          8          5
##    Some-college         76         2933        220        172
```

Inference:

explain a relationship between some of the values of the categoricals.

From the Cross tabulation visualization we can compare categoricals. For instance, we observe that Married-civ-spouses and Divorced cases have been witnessed the highest for HS-grad category.

Problem 2 (15 points):

```
## Problem 2 (15 points):
#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 t
#two original tables have in common. (Show a head of the resulting table.)


print(getwd())
```

```
## [1] "C:/Sadiya Studies/Data Science/DS441-Fundamts DS"
```

```
setwd("C:/Users/SADIYA/Downloads")
adult_dataevn <- read.csv("population_even.csv")

print(getwd())
```

```
## [1] "C:/Users/SADIYA/Downloads"
```

```
setwd("C:/Users/SADIYA/Downloads")
adult_dataodd <- read.csv("population_odd.csv")

finaldata <- adult_dataevn%>% left_join(adult_dataodd ,by="NAME")
 head(finaldata)
```

```
##   STATE.x       NAME POPESTIMATE2010 POPESTIMATE2012 POPESTIMATE2014
## 1       1    Alabama         4785437         4815588         4841799
## 2       2     Alaska          713910          730443          736283
## 3       4    Arizona         6407172         6554978         6730413
## 4       5   Arkansas         2921964         2952164         2967392
## 5       6 California        37319502        37948800        38596972
## 6       8   Colorado         5047349         5192647         5350101
##   POPESTIMATE2016 POPESTIMATE2018 STATE.y POPESTIMATE2011 POPESTIMATE2013
## 1         4863525         4887681       1         4799069         4830081
## 2          741456          735139       2          722128          737068
## 3         6941072         7158024       4              NA         6632764
## 4         2989918         3009733       5         2940667         2959400
## 5        39167117        39461588       6        37638369        38260787
## 6         5539215         5691287       8         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
```

```
head(adult_dataodd)
```

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

```
head(adult_dataevn)
```

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

```
## 5        39167117        39461588
## 6         5539215         5691287
```

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

 finaldata1 <- subset(finaldata,select = -c(STATE.y))
  head(finaldata1)
```

```
##   STATE.x       NAME POPESTIMATE2010 POPESTIMATE2012 POPESTIMATE2014
## 1       1    Alabama         4785437         4815588         4841799
## 2       2     Alaska          713910          730443          736283
## 3       4    Arizona         6407172         6554978         6730413
## 4       5   Arkansas         2921964         2952164         2967392
## 5       6 California        37319502        37948800        38596972
## 6       8   Colorado         5047349         5192647         5350101
##   POPESTIMATE2016 POPESTIMATE2018 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
```

```
#b. Rename columns to be just the year number.
rename_colmns <- c("STATE","NAME","2010","2012","2014","2016","2018", "2011",
                   "2013","2015","2017","2019")
names(finaldata1) <- rename_colmns
 head(finaldata1)
```

```
##   STATE       NAME     2010     2012     2014     2016     2018     2011
## 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
##       2013     2015     2017     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
```

```
#c. Reorder the columns to be in year order.
rordr_colmns = finaldata1 %>% select('STATE','NAME',str_sort(colnames(.),
                                                decreasing = FALSE, numeric = TRUE))

 head(rordr_colmns)
```

```
##   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 adult_data by replacing them with the average of the surrounding ye
#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.

  sum(is.na(rordr_colmns))
```

```
## [1] 5
```

```
   which(is.na(rordr_colmns))
```

```
## [1] 159 296 377 495 622
```

```
   rordr_colmns[rowSums(is.na(rordr_colmns)) > 0, ]
```

```
##    STATE      NAME     2010     2011     2012    2013     2014     2015
## 3      4   Arizona  6407172       NA  6554978 6632764  6730413  6829676
## 13    16     Idaho  1570746  1583910  1595324 1611206  1631112       NA
## 27    30   Montana   990697   997316  1003783 1013569  1021869  1030475
## 36    39      Ohio 11539336 11544663 11548923      NA 11602700 11617527
## 50    55 Wisconsin  5690475  5705288  5719960 5736754  5751525  5760940
##        2016     2017     2018     2019
## 3   6941072  7044008  7158024  7278717
## 13  1682380  1717715  1750536  1787065
## 27  1040859       NA  1060665  1068778
## 36 11634370 11659650 11676341 11689100
## 50  5772628  5790186  5807406       NA
```

```
   rordr_colmns[3,4] <- (rordr_colmns[3,3] + rordr_colmns[3,5])/2
   rordr_colmns[3,4]
```

```
## [1] 6481075
```

```
   rordr_colmns[13,8] <- (rordr_colmns[13,7] + rordr_colmns[13,9])/2
   rordr_colmns[13,8]
```

```
## [1] 1656746
```

```
   rordr_colmns[27,10] <- (rordr_colmns[27,9] + rordr_colmns[27,10])/2
   rordr_colmns[27,10]
```

```
## [1] NA
```

```
   rordr_colmns[36,6] <- (rordr_colmns[36,5]  + rordr_colmns[36,7])/2
   rordr_colmns[36,6]
```

```
## [1] 11575812
```

```
   rordr_colmns[50,12] <- rordr_colmns[50,11]
   rordr_colmns[50,12]
```

```
## [1] 5807406
```

```
   head(rordr_colmns)
```

```
##   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  6481075  6554978  6632764  6730413  6829676
## 4     5   Arkansas  2921964  2940667  2952164  2959400  2967392  2978048
## 5     6 California 37319502 37638369 37948800 38260787 38596972 38918045
## 6     8   Colorado  5047349  5121108  5192647  5269035  5350101  5450623
##       2016     2017     2018     2019
## 1  4863525  4874486  4887681  4903185
## 2   741456   739700   735139   731545
## 3  6941072  7044008  7158024  7278717
## 4  2989918  3001345  3009733  3017804
## 5 39167117 39358497 39461588 39512223
## 6  5539215  5611885  5691287  5758736
```

```
   #d. We can use some tidyverse aggregation to learn about the population.
##a. 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.
 finaldata2 <- rordr_colmns
    finaldata3 <- finaldata2 %>% rowwise() %>% mutate(MAX = max(c_across(`2010`:`2019`)) )
    head(finaldata3)
```

```
## # A tibble: 6 x 13
## # Rowwise:
##   STATE NAME      `2010` `2011` `2012` `2013` `2014` `2015` `2016` `2017` `2018`
##   <int> <chr>      <int>  <dbl>  <int>  <dbl>  <int>  <dbl>  <int>  <dbl>  <int>
## 1     1 Alabama   4.79e6 4.80e6 4.82e6 4.83e6 4.84e6 4.85e6 4.86e6 4.87e6 4.89e6
## 2     2 Alaska    7.14e5 7.22e5 7.30e5 7.37e5 7.36e5 7.37e5 7.41e5 7.40e5 7.35e5
## 3     4 Arizona   6.41e6 6.48e6 6.55e6 6.63e6 6.73e6 6.83e6 6.94e6 7.04e6 7.16e6
## 4     5 Arkansas  2.92e6 2.94e6 2.95e6 2.96e6 2.97e6 2.98e6 2.99e6 3.00e6 3.01e6
## 5     6 Californ~ 3.73e7 3.76e7 3.79e7 3.83e7 3.86e7 3.89e7 3.92e7 3.94e7 3.95e7
## 6     8 Colorado  5.05e6 5.12e6 5.19e6 5.27e6 5.35e6 5.45e6 5.54e6 5.61e6 5.69e6
## # ... with 2 more variables: `2019` <int>, MAX <dbl>
```

```
#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?
  popltn_all <- finaldata3 %>% mutate(total = sum(c_across(`2010`:`2019`)))
popltn_all
```

```
## # A tibble: 52 x 14
## # Rowwise:
##    STATE NAME     `2010` `2011` `2012` `2013` `2014` `2015` `2016` `2017` `2018`
##    <int> <chr>     <int>  <dbl>  <int>  <dbl>  <int>  <dbl>  <int>  <dbl>  <int>
##  1     1 Alabama  4.79e6 4.80e6 4.82e6 4.83e6 4.84e6 4.85e6 4.86e6 4.87e6 4.89e6
##  2     2 Alaska   7.14e5 7.22e5 7.30e5 7.37e5 7.36e5 7.37e5 7.41e5 7.40e5 7.35e5
##  3     4 Arizona  6.41e6 6.48e6 6.55e6 6.63e6 6.73e6 6.83e6 6.94e6 7.04e6 7.16e6
##  4     5 Arkansas 2.92e6 2.94e6 2.95e6 2.96e6 2.97e6 2.98e6 2.99e6 3.00e6 3.01e6
##  5     6 Califor~ 3.73e7 3.76e7 3.79e7 3.83e7 3.86e7 3.89e7 3.92e7 3.94e7 3.95e7
##  6     8 Colorado 5.05e6 5.12e6 5.19e6 5.27e6 5.35e6 5.45e6 5.54e6 5.61e6 5.69e6
##  7     9 Connect~ 3.58e6 3.59e6 3.59e6 3.59e6 3.59e6 3.59e6 3.58e6 3.57e6 3.57e6
##  8    10 Delaware 9.00e5 9.07e5 9.15e5 9.24e5 9.32e5 9.41e5 9.49e5 9.57e5 9.65e5
##  9    11 Distric~ 6.05e5 6.20e5 6.35e5 6.51e5 6.62e5 6.75e5 6.86e5 6.95e5 7.02e5
## 10    12 Florida  1.88e7 1.91e7 1.93e7 1.95e7 1.98e7 2.02e7 2.06e7 2.10e7 2.12e7
## # ... with 42 more rows, and 3 more variables: `2019` <int>, MAX <dbl>,
## #   total <dbl>
```

```
#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.
colSums(popltn_all[,3:12])
```

```
##      2010      2011      2012      2013      2014      2015      2016      2017
## 313043191 315244038 317465478 319585920 321835882 324114082 326347983        NA
##      2018      2019
## 329880855 331418189
```

```
#Problem 3 (15 points)

#Continuing with the data from Problem 2, let's create a graph of population over time for a few states
#(choose at least three yourself). This will require another data transformation, a reshaping. In order
#a line graph, we will need a variable that represents the year, so that it can be mapped to the x axis
#transformation to turn all those year columns into one column that holds the year, reducing the 10 yea
#columns down to 2 columns (year and population). Once the data are in the right shape, it will be no h
#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 lib
#super useful tool). Even after doing that, you have a string version of the year. R is seeing the 'wor
#two-zero-one-five instead of the number two thousand fifteen. It needs to be a number to work on a tim
#axis. There are many ways to fix this. You can look into type_convert or do more string parsing (e.g.,
#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).

rows <- popltn_all[c(3,5,10),2:12]
rows
```

```
## # A tibble: 3 x 11
## # Rowwise:
##   NAME     `2010` `2011` `2012` `2013` `2014` `2015` `2016` `2017` `2018` `2019`
##   <chr>     <int>  <dbl>  <int>  <dbl>  <int>  <dbl>  <int>  <dbl>  <int>  <int>
## 1 Arizona  6.41e6 6.48e6 6.55e6 6.63e6 6.73e6 6.83e6 6.94e6 7.04e6 7.16e6 7.28e6
## 2 Califor~ 3.73e7 3.76e7 3.79e7 3.83e7 3.86e7 3.89e7 3.92e7 3.94e7 3.95e7 3.95e7
## 3 Florida  1.88e7 1.91e7 1.93e7 1.95e7 1.98e7 2.02e7 2.06e7 2.10e7 2.12e7 2.15e7
```

```
popltn_col <- rows[,c(2:11)]
population <- pivot_longer(popltn_col,cols = 1:10, names_to = 'year',
                           values_to = 'pop')
population
```

```
## # A tibble: 30 x 2
##    year      pop
##    <chr>   <dbl>
##  1 2010  6407172
##  2 2011  6481075
##  3 2012  6554978
##  4 2013  6632764
##  5 2014  6730413
##  6 2015  6829676
##  7 2016  6941072
##  8 2017  7044008
##  9 2018  7158024
## 10 2019  7278717
## # ... with 20 more rows
```

```
state_namez <- rows[,1] # state names
state_namez
```

```
## # A tibble: 3 x 1
## # Rowwise:
##   NAME
##   <chr>
## 1 Arizona
## 2 California
## 3 Florida
```

```
 statez <- rep(c( 'Arizona', 'California','Florida'), each = 10)
statez
```

```
##  [1] "Arizona"    "Arizona"    "Arizona"    "Arizona"    "Arizona"
##  [6] "Arizona"    "Arizona"    "Arizona"    "Arizona"    "Arizona"
## [11] "California" "California" "California" "California" "California"
## [16] "California" "California" "California" "California" "California"
## [21] "Florida"    "Florida"    "Florida"    "Florida"    "Florida"
## [26] "Florida"    "Florida"    "Florida"    "Florida"    "Florida"
```

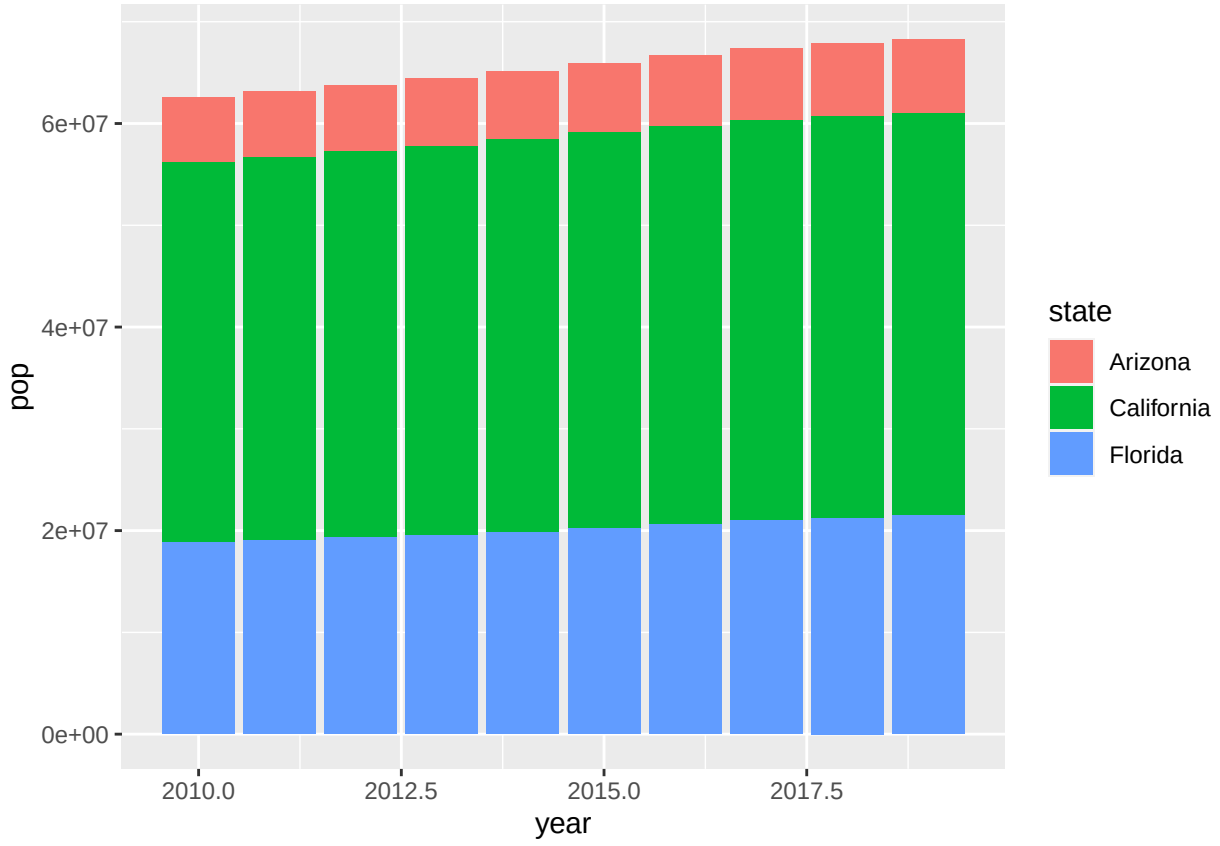
```
population$state <- statez
population$state
```

```
##  [1] "Arizona"    "Arizona"    "Arizona"    "Arizona"    "Arizona"
##  [6] "Arizona"    "Arizona"    "Arizona"    "Arizona"    "Arizona"
## [11] "California" "California" "California" "California" "California"
## [16] "California" "California" "California" "California" "California"
## [21] "Florida"    "Florida"    "Florida"    "Florida"    "Florida"
## [26] "Florida"    "Florida"    "Florida"    "Florida"    "Florida"
```

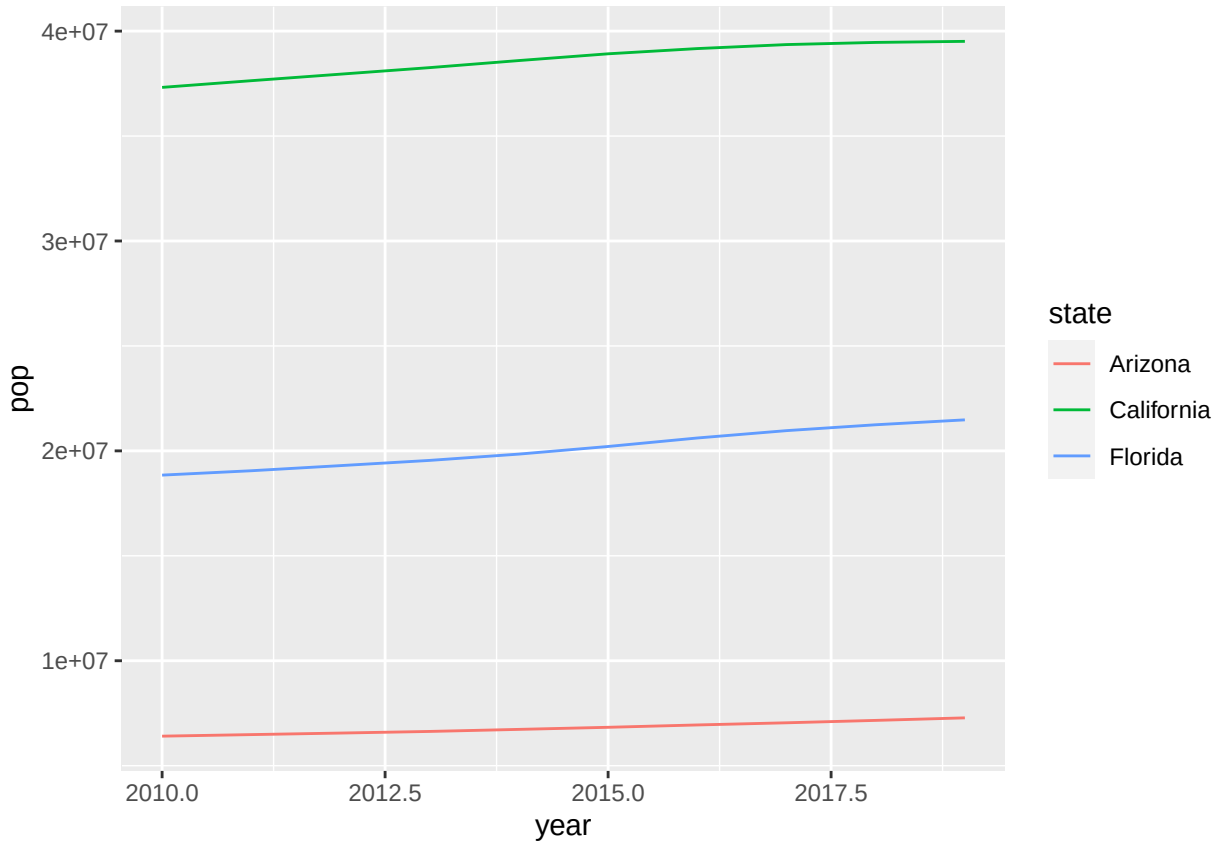
```
population$year <- as.integer(population$year) # convert to numeric
population$year
```

```
##  [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
```

```
p <- ggplot(population,aes(x=year,y=pop,fill=state)) +
 geom_col()
p
```




```
p <- ggplot(data = population,aes(x=year,y=pop)) +
  geom_line(aes(colour = state))
p
```




```
##Problem 4 (15 points)
##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.
```

1.Outdated data

This type of data can be solved with removing of those records from the database that are of no usage to the business . It happens because of newly generated fields which are missing in the old records. We can remove the redundant data or update those records.

2. Duplicate Data

Duplicates are among the worst delinquent of data pollution. Duplicates can form in a number of ways including data migration, via data exchanges through integrations, manual entry, and from batch imports. The most common duplicate objects are Contacts,Leads and Accounts.

```
##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?
##b. For the same data: can I predict if a customer will buy milk based on what else they bought?
##c. Suppose we have data listing items in individual purchases. What are different sets of
##products that are often purchased together?
```

a)

we can take do simple random sampling and later according to what they buy, we can cluster them.

b)

Yes, you can predict if the customer will buy milk based on what else they bought, by applying association analysis. It connects with products that are usually bought with it and then we can perform probability of it.

c)

This can be done with association analysis as well.

```
##c. Explain if each of the following is a data mining task
##a. Organizing the customers of a company according to education level.
##b. Computing the total sales of a company.
##c. Sorting a student database according to identification numbers.
##d. Predicting the outcomes of tossing a (fair) pair of dice.
##e. Predicting the future stock price of a company using historical records.
```

##Answer:

a) this is not a data mining task. It can be done by order function.
b) this is not a data mining task. Computing can be applied by simple arithmetic operations.
c) this is not a data mining task. a basic query search.
d) this is not a data mining task because Dice works on probability.
e) Yes, it is a data mining task -It can be implemented by creating a model to predict the continuous value for the stock price.