A Quantitative Analysis Project with R markdown

AdetolaBabatunde

2024-07-04

Question 1 : Data exploration and preprocessing:

Please use the two (2) datasets: "Ident.csv" and "assessment.csv" attached to perform the analysis. When saving and submitting your final dataset, please save as CSV format with the proposed nomenclature: "firstname_lastname.csv" or, as an example, "John_Doe.csv".
getwd()

```r
#-------------------------------------------------------------
# Question (1) - Data exploration and preprocessing
#------------------------------------------------------------
#-------------------------------------------------------------
# Check the working directory and set working directory
#------------------------------------------------------------
getwd()
```

```
## [1] "/Users/adetolababatunde/Desktop/CRA DataAnalytics"
```

```r
#setwd("/Users/adetolababatunde/Desktop/CRA DataAnalytics")
setwd("~/Desktop/CRA DataAnalytics")
#------------------------------------------------------------
# Import the dataset: Ident.csv
#------------------------------------------------------------
Ident_data = read.csv("Ident.csv", sep=",", header=TRUE) # 1 row per client
Ident = Ident_data # created exact copy of the "Ident" dataset to avoid altering the main data
# check the data type/structure
str(Ident) # The data structure of Ident data set has to be changed to be
```

```
## 'data.frame':    1635 obs. of  8 variables:
##  $ clt_id      : int  585 457 541 212 1390 513 224 1017 942 526 ...
##  $ clt_prov    : int  1 1 2 10 12 6 2 3 4 13 ...
##  $ clt_date    : chr  "1998-06-05" "1970-11-13" "1942-09-02" "1995-08-16" ...
##  $ clt_imm_date: chr  "2004-04-26" "2003-04-25" "1964-07-01" "2008-02-24" ...
##  $ clt_fil     : int  1 1 2 0 1 0 0 2 1 2 ...
##  $ clt_lang    : int  1 1 0 0 0 0 1 0 0 0 ...
##  $ clt_drink   : int  0 0 1 0 1 0 1 0 1 0 ...
##  $ clt_child   : chr  "0" "1" "3" "0" ...
```

```r
#consistent with the data dictionary.

#check for number of missing values
sum(is.na(Ident)) # no missing values EXCEPT for 99999 and XX
```

```
## [1] 0
```

```
#under clt_lang & clt_child respectively.
# which makes it inconsistent with the data dictionary for Ident dataset.

# verify that all NULL/NAs are 0
colSums(is.na(Ident)) # show each column and their respective number of missing values
```

```
##       clt_id     clt_prov     clt_date clt_imm_date      clt_fil     clt_lang
##            0            0            0            0            0            0
##    clt_drink    clt_child
##            0            0
```

Step 1.1: Find all problematic observations inconsistent with the data dictionary in the "Ident" dataset and DELETE the entire row from the "Ident" dataset where such inconsistency appears.

```
#-----------------------------------------------------------------------------------------------------
# Step 1.1: Finding all problematic observations inconsistent with data
#dictionary for Ident dataset  and deleting the entire row.
#-----------------------------------------------------------------------------------------------------
# Make client ID numeric
Ident$clt_id = as.numeric(Ident$clt_id)
# Make Provincial code categorical (nominal)
Ident$clt_prov = as.factor(Ident$clt_prov)

# Make Date of birth be in date format YYYY-MM-DD (Note: oldest not more than 100 yrs)
Ident$clt_date = as.Date(Ident$clt_date)
max(Ident$clt_date);min(Ident$clt_date) # maximum and minimum date
```

```
## [1] "2002-12-11"

## [1] "1800-01-01"
```

```
# creating an age column (clt_age_yrs) to be able to trim the "Ident" dataset to not more than 100 year
library(eeptools)
```

```
## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.3.1

## Warning in check_dep_version(): ABI version mismatch:
## lme4 was built with Matrix ABI version 1
## Current Matrix ABI version is 0
## Please re-install lme4 from source or restore original 'Matrix' package
```

```
# cannot be more than 100 years old
Ident$clt_age_yrs = floor(age_calc(Ident$clt_date, max(Ident$clt_date), units = "years"))
# now, delete all the individuals that are more than 100 years old
Ident_updated = Ident[-which(Ident$clt_age_yrs>100),]

# Make Immigration date be in date format YYYY-MM-DD (Note: cannot be earlier than the dob)
Ident_updated$clt_imm_date = as.Date(Ident_updated$clt_imm_date)
# check if all the dob is less than the Imm_date except for born in Canada "0000-01-01"
max(Ident_updated$clt_imm_date);min(Ident_updated$clt_imm_date) # born in Canada is "0000-01-01"
```

```
## [1] "2020-11-30"
```

```
## [1] "0000-01-01"
```

```
born_in_canada = Ident_updated[which(Ident_updated$clt_imm_date == as.Date("0000-01-01")),]
# dataframe of those born in Canada
sum(Ident_updated$clt_imm_date < Ident_updated$clt_date); NROW(born_in_canada$clt_id)
```

```
## [1] 325
```

```
## [1] 325
```

```
# only those born in Canada have earlier clt_imm_date than DOB (clt_date)

# Make On-time filing categorical (nominal)
Ident_updated$clt_fil = as.factor(Ident_updated$clt_fil)

# Make Preferred official language categorical (nominal)
# This showed that there is 99999 which should be excluded
unique(as.factor(Ident_updated$clt_lang))
```

```
## [1] 1     0     99999
## Levels: 0 1 99999
```

```
# Verify how many rows have 99999
all_99999 = Ident_updated[which(Ident_updated$clt_lang=="99999"),]; NROW(all_99999)
```

```
## [1] 16
```

```
# Delete the rows with "99999" from the Ident_updated dataset
# remove the rows with "99999" from the dataset
Ident_updated = Ident_updated[-which(Ident_updated$clt_lang=="99999"),]
# Make Preferred official language categorical (nominal)
Ident_updated$clt_lang = as.factor(Ident_updated$clt_lang)


# Make Preferred drink categorical (nominal)
 # This showed that everything is fine
unique(as.factor(Ident_updated$clt_drink))
```

```
## [1] 0 1
## Levels: 0 1
```

```
Ident_updated$clt_drink = as.factor(Ident_updated$clt_drink)

# make Number of children numeric
# This showed that there is XX which should be excluded
unique(as.factor(Ident_updated$clt_child))
```

```
## [1] 0  1  3  2  4  5  XX
## Levels: 0 1 2 3 4 5 XX
```

```
# Verify how many rows have XX
all_XX = Ident_updated[which(Ident_updated$clt_child=="XX"),]; NROW(all_XX)
```

```
## [1] 16
```

```
# Delete the rows with "XX" from the Ident_updated dataset
# remove the rows with "XX" from the dataset
Ident_updated = Ident_updated[-which(Ident_updated$clt_child=="XX"),]
# make Number of children numeric
Ident_updated$clt_child = as.numeric(Ident_updated$clt_child)

# check the data type/structure again to see that everything
#aligns exactly with the data dictionary for "Ident" data set
str(Ident_updated) # all align with the data dictionary for "Ident" dataset
```

```
## 'data.frame':    1587 obs. of  9 variables:
##  $ clt_id      : num  585 457 541 212 1390 ...
##  $ clt_prov    : Factor w/ 13 levels "1","2","3","4",..: 1 1 2 10 12 6 2 3 4 13 ...
##  $ clt_date    : Date, format: "1998-06-05" "1970-11-13" ...
##  $ clt_imm_date: Date, format: "2004-04-26" "2003-04-25" ...
##  $ clt_fil     : Factor w/ 3 levels "0","1","2": 2 2 3 1 2 1 1 3 2 3 ...
##  $ clt_lang    : Factor w/ 2 levels "0","1": 2 2 1 1 1 1 2 1 1 1 ...
##  $ clt_drink   : Factor w/ 2 levels "0","1": 1 1 2 1 2 1 2 1 2 1 ...
##  $ clt_child   : num  0 1 3 0 2 0 4 4 0 4 ...
##  $ clt_age_yrs : num  4 32 60 7 35 55 28 22 48 21 ...
```

```
# End data for Step 1.1:
# removing the column we used to remove more than 100 years
Ident_updated = Ident_updated[,-ncol(Ident_updated)]
```

Step 1.2: Merge the "Ident" dataset and the "Assess" dataset together using the common variable. Use the list of [clt_id] in the "Ident" dataset as your full population. Keep all columns from both datasets. Keep only rows with the most recent assessments for each taxpayer [clt_id].

```
#-----------------------------------------------------------------------------------------------------
# Step 1.2: Merging "Ident" dataset and the "Assess" dataset
# and keeping the most recent assessments for each taxpayer
#-----------------------------------------------------------------------------------------------------

#------------------------------------------------------------
# Import the dataset: assessment.csv
#------------------------------------------------------------
assessment = read.csv("assessment.csv", sep=",", header=TRUE)

# check the data type/structure
str(assessment)
```

```
## 'data.frame':    8000 obs. of  5 variables:
##  $ clt_id    : int  1390 870 251 1187 141 405 1287 836 1479 602 ...
##  $ clt_tax_nb: int  4 4 10 1 8 6 10 6 7 1 ...
##  $ clt_emp_1 : int  176347 NA 160820 NA 196112 58262 125429 142247 19095 161149 ...
##  $ clt_emp_2 : int  NA 221318 NA 78867 NA NA NA NA NA NA ...
##  $ clt_emp_3 : int  13341 38298 21471 6544 NA 15044 NA NA 31052 NA ...
```

```r
# Data Cleaning:
# check for number of missing values
sum(is.na(assessment)) # about 13102 missing values
```

```
## [1] 13102
```

```r
# verify that all NULL/NAs are 0
# show each column and their respective number of missing values
colSums(is.na(assessment))
```

```
##     clt_id clt_tax_nb  clt_emp_1  clt_emp_2  clt_emp_3
##          0          0       1980       6399       4723
```

```r
assessment$clt_id = as.numeric(assessment$clt_id) # make client ID numeric
assessment$clt_tax_nb = as.numeric(assessment$clt_tax_nb) # Make assessing number numeric
assessment$clt_emp_1 = as.numeric(assessment$clt_emp_1) # Make employment income numeric
assessment$clt_emp_2 = as.numeric(assessment$clt_emp_2) # Make self-employment income numeric
assessment$clt_emp_3 = as.numeric(assessment$clt_emp_3) # Make Other income numeric

# change all NULL/NAs to 0 (zeros)
assessment[is.na(assessment)] = 0 # assume that all NULLS/NAs are zeros (0)

# verify that all NULL/NAs are 0
sum(colSums(assessment==0)) # this values matches the values when I checked for missing value earlier
```

```
## [1] 13102
```

```r
# Merging the Ident to Assess dataset using clt_id in the ident as full population
Ident_Assess = merge(Ident_updated, assessment, by.x = "clt_id")


# use the R package "data.table" to obtain the most recent assessment for each taxpayer
# install.packages("data.table")
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 4.3.1
```

```r
Ident_Assess_updated = as.data.table(Ident_Assess)
Ident_Assess_updated = Ident_Assess_updated[Ident_Assess_updated[, .I[clt_tax_nb == max(clt_tax_nb)], by

# to debug
Ident_Assess_updated[Ident_Assess_updated$clt_id==45,]
```

```
##    clt_id clt_prov   clt_date clt_imm_date clt_fil clt_lang clt_drink clt_child
##     <num>   <fctr>     <Date>       <Date>  <fctr>   <fctr>    <fctr>     <num>
## 1:     45        2 1940-01-06   1977-04-25       1        0         1         5
##    clt_tax_nb clt_emp_1 clt_emp_2 clt_emp_3
##         <num>     <num>     <num>     <num>
## 1:          8     24638         0     17680
```

```
# End of Step 1.2: Final data is Ident_Assess_updated
```

Step 1.3: Calculate age [clt_age] where the age of an individual is based on the age of the taxpayer on December 31, 2021. Truncate all decimals if any.

```
#----------------------------------------------------------------------------------------------------
# Step 1.3: Calculate age (ctl_age) based on December 31, 2021 and truncate all decimals.
#----------------------------------------------------------------------------------------------------
# install.packages("eeptools");
library(eeptools)
# Calculate age (clt_age) based on the age of the taxpayer on December 31, 2021
Ident_Assess_updated$clt_age = floor(age_calc(Ident_Assess_updated$clt_date,as.Date("2021-12-31"), units
```

Step 1.4: Calculate total income [clt_emp_tot] where an individual total income is the sum of employment income, self-employment income, and other income.

```
#----------------------------------------------------------------------------------------------------
# Step 1.4: Calculate the total income (clt_emp_tot)
# where Total Income = sum of employment income + self-employment income + other income.
#----------------------------------------------------------------------------------------------------
Ident_Assess_updated$clt_emp_tot = Ident_Assess_updated$clt_emp_1 + Ident_Assess_updated$clt_emp_2 + Ide
```

Step 1.5: Calculate the number of years [clt_yrs_can] an individual lived in Canada as of December 31, 2021. For immigrants, it would be calculated based on immigration date and for those born in Canada, it would be calculated based on their birth date. Truncate all decimals if any.

```
#----------------------------------------------------------------------------------------------------
# Step 1.5: Calculate the number of years (clt_yrs_can) lived in Canada as of December 31, 2021
# For immigrants: use immigration date
# For Canadian born: use birth date and truncate all decimals for all.
#----------------------------------------------------------------------------------------------------

# dataframe of those born in Canada
born_in_canada_updated = Ident_Assess_updated[which(Ident_Assess_updated$clt_imm_date == as.Date("0000-0
# calculating the number of years (clt_yrs_can) lived in Canada as of December 31, 2021
born_in_canada_updated$clt_yrs_can = floor(age_calc(born_in_canada_updated$clt_date,as.Date("2021-12-31

# dataframe of Immigrants
Immigrants = Ident_Assess_updated[which(Ident_Assess_updated$clt_imm_date != as.Date("0000-01-01")),]
# calculating the number of years (clt_yrs_can) lived in Canada as of December 31, 2021
Immigrants$clt_yrs_can = floor(age_calc(Immigrants$clt_imm_date,as.Date("2021-12-31"), units = "years")

# The Final data set
Ident_Assess_Final = as.data.frame(rbind.data.frame(born_in_canada_updated, Immigrants))
```

Step 1.6: Export and save your final combined dataset as CSV format with the following naming convention: "firstname_lastname.csv" or, as an example, "John_Doe.csv".

```
#----------------------------------------------------------------------------------------------------
# Step 1.6: Saving and Exporting the final combined dataset as CSV format
#----------------------------------------------------------------------------------------------------
write.csv(Ident_Assess_Final, file = "~/Desktop/CRA DataAnalytics//Adetola_Babatunde.csv")
#-------------------------------------------------------------
```

Question 2: Quantitative analysis and visulization

- For this question, please use the dataset "Clean_Q2.csv" and not the one you created in Question 1. Assume all NULLs are zeros.
- Create one "box plot" (also known as "box-and-whisker plot"), by plotting the variable total income ([clt_inc]) on the y-axis versus each of the following age ranges on the x-axis: 0-20 21-40 41-60 61-80 81-100

```r
# Question (2) - Quantitative analysis and visualization
#-----------------------------------------------------------
#------------------------------------------------------------
# Check the working directory and set working directory
#-----------------------------------------------------------
getwd()
```

```
## [1] "/Users/adetolababatunde/Desktop/CRA DataAnalytics"
```

```r
setwd("~/Desktop/CRA DataAnalytics")
#-----------------------------------------------------------
# Import the dataset: Clean_Q2.csv
#-----------------------------------------------------------
Clean_data = read.csv("Clean_Q2.csv", sep=",", header=TRUE) #
Clean = Clean_data # created exact copy of the "Clean_Q2" dataset to avoid altering the main data
# check the data type/structure
str(Clean) # The data structure of Clean_Q2 data set has to be changed to be consistent with the data d
```

```
## 'data.frame':    611 obs. of  8 variables:
##  $ indv_id      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ clt_inc      : int  7284 439923 195093 44104 87580 162512 233058 211235 422463 316784 ...
##  $ clt_ind_age  : int  10 28 86 54 12 27 79 81 2 29 ...
##  $ clt_sex      : int  0 1 1 2 0 1 0 1 2 0 ...
##  $ clt_mstatus  : int  0 0 0 1 1 1 0 1 1 1 ...
##  $ clt_method   : int  0 0 1 1 1 0 0 1 0 0 ...
##  $ clt_animal   : int  1 0 0 0 0 1 0 1 0 0 ...
##  $ clt_education: int  4 2 3 1 5 4 3 4 2 1 ...
```

```r
# check for number of missing values
sum(is.na(Clean))
```

```
## [1] 0
```

```r
# which makes it inconsistent with the data dictionary for Clean_Q2 dataset.
# verify that all NULL/NAs are 0
colSums(is.na(Clean)) # show each column and their respective number of missing values
```

```
##       indv_id       clt_inc   clt_ind_age       clt_sex   clt_mstatus
##             0             0             0             0             0
##    clt_method    clt_animal clt_education
##             0             0             0
```

```r
# Make client ID numeric
Clean$indv_id = as.numeric(Clean$indv_id) # not clt_id as stated in the data dictionary for Clean_Q2

# Make Total income numeric
Clean$clt_inc = as.numeric(Clean$clt_inc)
#unique(Clean$clt_inc)

# Make Age numeric
Clean$clt_ind_age = as.numeric(Clean$clt_ind_age)
unique(Clean$clt_ind_age)
```

```
##  [1] 10 28 86 54 12 27 79 81  2 29 77 60 36 18 63 35 37 15 48 41 43 13 33 74  3
## [26] 55 20 46 34 39 22 76 19 40 64  7 69 57 14 72 26 71 51 80 62 75  9  6 70 50
## [51] 52 85 83 47  8 73 45 84 49 11  1 38 25 42 78 44 66 56 82 58  5 61 31 32 24
## [76]  4 30 21 17 16 59 68 67 65 53 23
```

```r
# Make Gender categorical (nominal)
Clean$clt_sex = as.factor(Clean$clt_sex)
unique(Clean$clt_sex)
```

```
## [1] 0 1 2
## Levels: 0 1 2
```

```r
# Make Marital Status categorical (nominal)
Clean$clt_mstatus = as.factor(Clean$clt_mstatus)
unique(Clean$clt_mstatus)
```

```
## [1] 0 1
## Levels: 0 1
```

```r
# Make Filing method categorical (nominal)
Clean$clt_method = as.factor(Clean$clt_method)
unique(Clean$clt_method)
```

```
## [1] 0 1
## Levels: 0 1
```

```r
# Make Preferred pet categorical (nominal)
Clean$clt_animal = as.factor(Clean$clt_animal)
unique(Clean$clt_animal)
```

```
## [1] 1 0
## Levels: 0 1
```

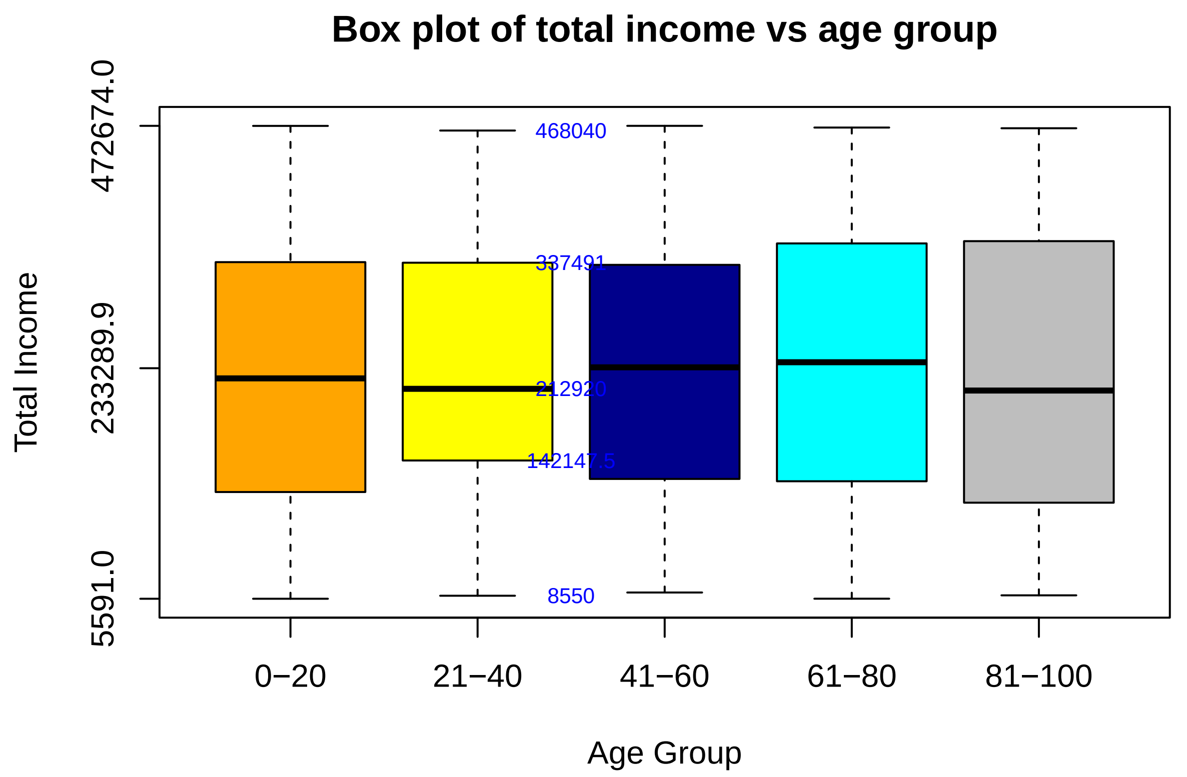
```r
# Make Education level categorical (ordinal)
Clean$clt_education = ordered(Clean$clt_education)
unique(Clean$clt_education)
```

```
## [1] 4 2 3 1 5
## Levels: 1 < 2 < 3 < 4 < 5
```

```
# create the age group
Clean$ctl_age_group = ifelse(Clean$clt_ind_age<=20, "0-20"
                      ,ifelse(Clean$clt_ind_age>=21 & Clean$clt_ind_age<=40, "21-40"
                      ,ifelse(Clean$clt_ind_age>=41 & Clean$clt_ind_age<=60, "41-60"
                      ,ifelse(Clean$clt_ind_age>=61 & Clean$clt_ind_age<=80, "61-80"
                      ,"81-100"))))
# Make Age Group categorical (ordinal)
Clean$ctl_age_group = ordered(Clean$ctl_age_group)
unique(Clean$ctl_age_group)
```

```
## [1] 0-20   21-40  81-100 41-60  61-80
## Levels: 0-20 < 21-40 < 41-60 < 61-80 < 81-100
```

```
# The box plot of total income vs age group
clean_boxplot = boxplot(clt_inc ~ ctl_age_group, data = Clean
        ,main = "Box plot of total income vs age group"
        ,xlab = "Age Group"
        ,ylab = "Total Income"
        ,col = c("orange", "yellow", "darkblue", "cyan", "grey")
        ,yaxt="n")
axis(2, at=round(c(min(Clean$clt_inc), mean(Clean$clt_inc), max(Clean$clt_inc)),1),
     labels = T)
text(x = 2.5,
     y = boxplot.stats(clean_boxplot$stats[,2])$stats,
     labels = boxplot.stats(clean_boxplot$stats[,2])$stats,
     cex = 0.7,
     col = "blue")
```



```
# Extract the approximate statistics from boxplot
clean_boxplot;
```

```
## $stats
##        [,1]     [,2]     [,3]   [,4]   [,5]
## [1,]   5591   8550.0  11757.0   5646   8963
## [2,] 111004 142147.5 123934.0 121630 100456
## [3,] 223238 212920.0 234162.0 239177 211235
## [4,] 338054 337491.0 335387.5 356554 358783
## [5,] 472631 468040.0 472674.0 471010 470306
##
## $n
## [1] 138 156 147 129  41
##
## $conf
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,] 192700.1 188208.8 206606.2 206496.4 147491.6
## [2,] 253775.9 237631.2 261717.8 271857.6 274978.4
##
## $out
## numeric(0)
##
## $group
## numeric(0)
##
## $names
## [1] "0-20"   "21-40"  "41-60"  "61-80"  "81-100"
```

```
clean_boxplot$stats; # quartile summaries
```

```
##        [,1]     [,2]     [,3]   [,4]   [,5]
## [1,]   5591   8550.0  11757.0   5646   8963
## [2,] 111004 142147.5 123934.0 121630 100456
## [3,] 223238 212920.0 234162.0 239177 211235
## [4,] 338054 337491.0 335387.5 356554 358783
## [5,] 472631 468040.0 472674.0 471010 470306
```

```
clean_boxplot$stats[3,] # median
```

```
## [1] 223238 212920 234162 239177 211235
```

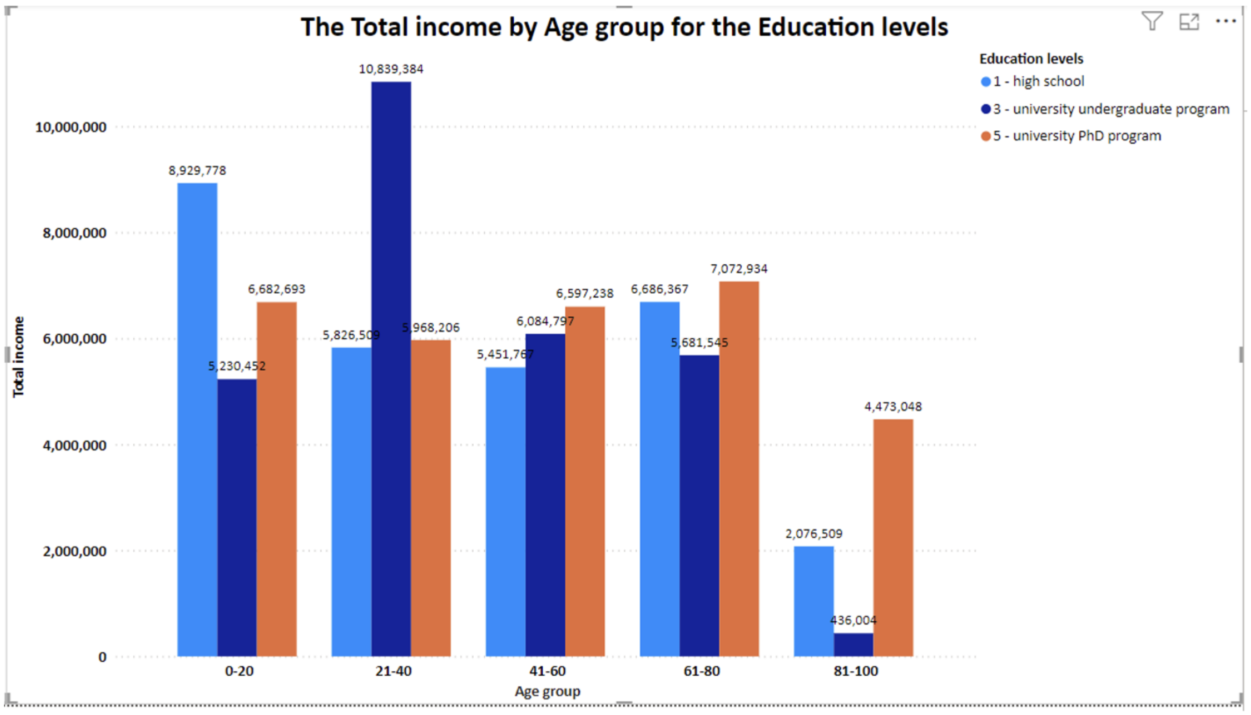
```
#------------------------------------------------------------
```

Question 3: Quantitative analysis and visualization

- For this question, please use the dataset "Clean_Q2.csv" and its associated data dictionary shown in Question 2. Assume all NULLs are zeros. **A)** Create one visual that shows total income ([clt_inc]) by age ([clt_ind_age]) for the following education levels ([clt_education]): high school university undergraduate program

university PhD program.
Please ensure to put an appropriate title, appropriate axes labels, legends (if needed), and any other elements in your visual to improve readability and avoid overplotting, and then place the plot in the following space allotted below –

Solution ___



B)In addition to outliers and interquartile range, what are the other 5 components of any "box plot" (also known as "box-and-whisker plot")? Use the box plot that you created in Question 2a above and give an approximate value of total income ([clt_inc]) for each of the 5 components in the box plot for the age range 21-40 (5 points) (maximum of 150 words).

Solution

In addition to the outliers and interquartile range, the other 5 components of a box plot are: 1. **Minimum:** the smallest number in the dataset of interest 2. **First quartile,** Q1 (also known as the 25th percentile) 3. **Median:** is the middle value in the dataset of interest 4.**Third quartile:** denoted by Q3 (also known as the 75th percentile) is the median of the upper half of the dataset of interest [1]. 5. **Maximum:** the largest number in the dataset of interest **The approximate value of total income ([clt_inc]) for each of the 5 components for the age group 21-40 are:** 1. The minimum = 8,550 2. First quartile = 142,148 3. Median = 212,920 4. Third quartile = 337,491 5. Maximum = 468,040