NAME: OMAR USMAN UPPAL

CMS ID: 203515

ASSIGNMENT 2

CS823 Advance Topics in Databases [Big Data Analytics}

SECTION 1. HOW TO LOAD DATA IN R

In the field of data science and big data there are two common types of data formats that we use to load DATA in R.

1. Load Data from Files
2. Load Data from Relational Databases

In this section we will focus on the first type only.

1.1 LOAD DATA FROM FILES

1.1.1 EXAMPLE 1:

In this example the data we will use has two important points.

1. Data is read from the URL
2. Data is well-structured

READ DATA

“read.table” is an R command used to read the data from different sources. It can read the data from the files stored on the system as well as directly from the urls.

```
cardata <- read.table('http://www.win-vector.com/dfiles/car.data.csv',
sep=',',header=T)
```

EXAMINE DATA

Now, the data is loaded in data frame object –cardata– we will apply different R commands to examine the data.

```
class(cardata)
```

```
## [1] "data.frame"
```

class() commands gives information about the type of R object which in this case is **“data.frame”**

```
dim(cardata)
```

```
## [1] 1728    7
```

dim() commands gives information about the dimenstion of the data i.e., the number of rows and number of columns which in this case is **“1728 rows and 7 columns”**

```
summary(cardata)
```

```
##    buying       maint        doors     persons      lug_boot    safety
##  high :432   high :432   2    :432   2   :576   big  :576   high:576
##  low  :432   low  :432   3    :432   4   :576   med  :576   low :576
##  med  :432   med  :432   4    :432   more:576   small:576   med :576
##  vhigh:432   vhigh:432   5more:432
##    rating
##  acc  : 384
##  good :  69
##  unacc:1210
##  vgood:  65
```

summary() commands is used to get summary of almost any R object. In this case the **summary (cardata)** gives information about the **distribution** of the data.

```
summary(cardata$buying)
```

```
##  high   low   med vhigh
##   432   432   432   432
```

summary(cardata$buying) command can be used to get information about the specific column by provding it arguments. In this case the the command shows the distribtuion of specific column *buying*.

help command gives documentation of a class. Using **help(class(cardata)** shows the helpful information in the side window.

EXAMPLE 1 SUMMARY

After executing different commands above, the information we get from the output can be summarized in below points.

- 7 columns have the headins that help us understand about the information in the columns
- Each car in the dataset have 2, 3, 4 or 5more doors. We have to understand what 5more means? Does it mean 5 or more doors?

- 432 cars are 2 doors
- 432 cars are 3 doors
- Each car in the dataset can seat 2, 4 or more persons.
- 576 cars are 4-seaters
- Similary other columns *'buying', 'maint', 'lug-boot', 'safety', 'rating'* also provide information about the dataset that we can sue for further analysis.

1.1.2 EXAMPLE 2:

In this example the data we will use has two important points.

1. Data is read from the file stored in the system
2. Data is Less-structured

READ DATA

We will read data from the *file(german.data)* placed in local folder *C:/MyRwork/Big Data*. The file is downloaded from the github link https://github.com/WinVector/zmPDSwR/tree/master/Statlog (/url).

To read the data first we will set the working directory to folder where our file is placed. Than we will use the read.table() command to load the data into our data frame object.

```
setwd("C:/MyRwork/Big Data")
creditdata <- read.table('german.data',sep='',stringsAsFactors=F,header=F)
```

creditdata is the new data frame object that now contains the data read from the file.

EXAMINE DATA

Now we will execute the three commands as we did in Example 1 and see what information we get from it.

```
class(creditdata)
```

```
## [1] "data.frame"
```

```
dim(creditdata)
```

```
## [1] 1000   21
```

```
summary(creditdata)
```

```
##       V1                  V2            V3                 V4
##  Length:1000        Min.   : 4.0   Length:1000        Length:1000
##  Class :character   1st Qu.:12.0   Class :character   Class :character
##  Mode  :character   Median :18.0   Mode  :character   Mode  :character
##                     Mean   :20.9
##                     3rd Qu.:24.0
##                     Max.   :72.0
##        V5             V6                 V7                 V8
##  Min.   :  250   Length:1000        Length:1000        Min.   :1.000
##  1st Qu.: 1366   Class :character   Class :character   1st Qu.:2.000
##  Median : 2320   Mode  :character   Mode  :character   Median :3.000
##  Mean   : 3271                                         Mean   :2.973
##  3rd Qu.: 3972                                         3rd Qu.:4.000
##  Max.   :18424                                         Max.   :4.000
##       V9                V10                V11            V12
##  Length:1000        Length:1000        Min.   :1.000   Length:1000
##  Class :character   Class :character   1st Qu.:2.000   Class :character
##  Mode  :character   Mode  :character   Median :3.000   Mode  :character
##                                        Mean   :2.845
##                                        3rd Qu.:4.000
##                                        Max.   :4.000
##       V13            V14                V15                V16
##  Min.   :19.00   Length:1000        Length:1000        Min.   :1.000
##  1st Qu.:27.00   Class :character   Class :character   1st Qu.:1.000
##  Median :33.00   Mode  :character   Mode  :character   Median :1.000
##  Mean   :35.55                                         Mean   :1.407
##  3rd Qu.:42.00                                         3rd Qu.:2.000
##  Max.   :75.00                                         Max.   :4.000
##      V17                V18            V19                V20
##  Length:1000        Min.   :1.000   Length:1000        Length:1000
##  Class :character   1st Qu.:1.000   Class :character   Class :character
##  Mode  :character   Median :1.000   Mode  :character   Mode  :character
##                     Mean   :1.155
##                     3rd Qu.:1.000
##                     Max.   :2.000
##       V21
##  Min.   :1.0
##  1st Qu.:1.0
##  Median :1.0
##  Mean   :1.3
##  3rd Qu.:2.0
##  Max.   :2.0
```

class() and *dim()* commands show that our object is of type *data.frame* with dimensions of *1000 rows x 21 columns*. However the exection of *summary* commands shows the distribution but we cannot get the information what it actually means. The data is an incomprehensible block of codes with no meaningful explanations.

Hence we will introduce another step here before we can *EXAMINE THE DATA*.

TRANSFORM DATA

This data is stored as tabular data without headers; it uses a cryptic encoding of values that requires the dataset's accompanying documentation to untangle. Details of the German bank credit dataset can be found at http://mng.bz/mZbu (/url).

We will start by printing the first 3 rows of the dataset.

```
print(creditdata[1:3,])
```

```
##    V1 V2  V3  V4   V5  V6  V7 V8  V9  V10 V11  V12 V13  V14  V15 V16  V17
## 1 A11  6 A34 A43 1169 A65 A75  4 A93 A101   4 A121  67 A143 A152   2 A173
## 2 A12 48 A32 A43 5951 A61 A73  2 A92 A101   2 A121  22 A143 A152   1 A173
## 3 A14 12 A34 A46 2096 A61 A74  2 A93 A101   3 A121  49 A143 A152   1 A172
##   V18  V19  V20 V21
## 1   1 A192 A201   1
## 2   1 A191 A201   2
## 3   2 A191 A201   1
```

We can notice that we get the exact same three rows we saw in the *german.data* file with the addition of column names V1 through V21. We can change the column names to something meaningful using the detail available on the dataset link.

```
colnames(creditdata) <- c('Status.of.existing.checking.account',
                'Duration.in.month', 'Credit.history', 'Purpose',
                'Credit.amount', 'Savings account/bonds',
                'Present.employment.since','Installment.rate.in.percentage.of.disposa
ble.income',
                'Personal.status.and.sex', 'Other.debtors/guarantors',
                'Present.residence.since', 'Property', 'Age.in.years',
                'Other.installment.plans', 'Housing',
                'Number.of.existing.credits.at.this.bank', 'Job',
                'Number.of.people.being.liable.to.provide.maintenance.for',
                'Telephone', 'foreign.worker', 'Good.Loan')

creditdata$Good.Loan <- as.factor(ifelse(creditdata$Good.Loan==1,'GoodLoan','BadLoa
n'))

print(creditdata[1:3,])
```

```
##   Status.of.existing.checking.account Duration.in.month Credit.history
## 1                                 A11                6            A34
## 2                                 A12               48            A32
## 3                                 A14               12            A34
##   Purpose Credit.amount Savings account/bonds Present.employment.since
## 1     A43          1169                  A65                       A75
## 2     A43          5951                  A61                       A73
## 3     A46          2096                  A61                       A74
##   Installment.rate.in.percentage.of.disposable.income
## 1                                                   4
## 2                                                   2
## 3                                                   2
##   Personal.status.and.sex Other.debtors/guarantors Present.residence.since
## 1                     A93                    A101                       4
## 2                     A92                    A101                       2
## 3                     A93                    A101                       3
##   Property Age.in.years Other.installment.plans Housing
## 1     A121          67                    A143    A152
## 2     A121          22                    A143    A152
## 3     A121          49                    A143    A152
##   Number.of.existing.credits.at.this.bank  Job
## 1                                       2 A173
## 2                                       1 A173
## 3                                       1 A172
##   Number.of.people.being.liable.to.provide.maintenance.for Telephone
## 1                                                         1      A192
## 2                                                         1      A191
## 3                                                         2      A191
##   foreign.worker Good.Loan
## 1           A201  GoodLoan
## 2           A201   BadLoan
## 3           A201  GoodLoan
```

colnames command is used to change the column names and we can see that they are now giving information about the data in the respective columns.

creditdata$Good.Loan means that want to do some operation on our data frame object **creditdata** and specifically its column 21 which we renamed to **Good.Loan**. In the single line of code in which **creditdata$Good.Loan** has been assigned new values following operations were done.

- 1 in the column was replaced with “GoodLoan” while 0 was replaced with “BadLoad”
- Once all the values were replaced the **creditdata$Good.Loan** object type encoded to *factor*.

Comparing the result of *print(creditdata[1:3,])* with earlier results we can see the change in the last column which shows meaningful explanation.

MAPPING FUNCTION FOR A-* CODES

Reading the data documentation further tells us that it has a dictionary of the meanings of all of the cryptic A* codes. For example, it says in column 4 (now called Purpose, meaning the purpose of the loan) that the code A40 is a new car loan, A41 is a used car loan, and so on. We will create a mapping list that we will use to replace A* codes with their meanings.

```
mapping <- list('A11'='... < 0 DM',
 'A12'='0 <= ... < 200 DM',
 'A13'='... >= 200 DM / salary assignments for at least 1 year',
 'A14'='no checking account',
 'A30'='no credits taken/all credits paid back duly',
 'A31'='all credits at this bank paid back duly',
 'A32'='existing credits paid back duly till now',
 'A33'='delay in paying off in the past',
 'A34'='critical account/other credits existing (not at this bank)',
 'A40'='car (new)',
 'A41'='car (used)',
 'A42'='furniture/equipment',
 'A43'='radio/television',
 'A44'='domestic appliances',
 'A45'='repairs',
 'A46'='education',
 'A47'='(vacation - does not exist?)',
 'A48'='retraining',
 'A49'='business',
 'A410'='others',
 'A61'='... < 100 DM',
 'A62'='100 <= ... < 500 DM',
 'A63'='500 <= ... < 1000 DM',
 'A64'='.. >= 1000 DM',
 'A65'='unknown/ no savings account',
 'A71'='unemployed',
 'A72'='... < 1 year',
 'A73'='1 <= ... < 4 years',
 'A74'='4 <= ... < 7 years',
 'A75'='.. >= 7 years',
 'A91'='male : divorced/separated',
 'A92'='female : divorced/separated/married',
 'A93'='male : single',
 'A94'='male : married/widowed',
 'A95'='female : single',
 'A101'='none',
 'A102'='co-applicant',
 'A103'='guarantor',
 'A121'='real estate',
 'A122'='if not A121 : building society savings agreement/life insurance',
 'A123'='if not A121/A122 : car or other, not in attribute 6',
 'A124'='unknown / no property',
 'A141'='bank',
 'A142'='stores',
 'A143'='none',
 'A151'='rent',
 'A152'='own',
 'A153'='for free',
```

```
 'A171'='unemployed/ unskilled - non-resident',
 'A172'='unskilled - resident',
 'A173'='skilled employee / official',
 'A174'='management/ self-employed/highly qualified employee/ officer',
 'A191'='none',
 'A192'='yes, registered under the customers name',
 'A201'='yes',
 'A202'='no')
```

lists is R's structure that maps strings to arbitrary objects. In the next line of code we will use the **mapping** structure to replace the A* codes in the data frame object **creditdata**

```
for(i in 1:(dim(creditdata))[2]) {
if(class(creditdata[,i])=='character') {
creditdata[,i] <- as.factor(as.character(mapping[creditdata[,i]]))
}
}
```

Above lines of codes were the last step of **TRANSFORM DATA** step. Once the complete code is executed we will have transformed data in the rows of **creditdata** object. Following actions were done in the above lines of code.

- for loop is executed 21 times
- IF the class type of the column is **character** than A* codes are mapped with he meaning.
- Object type is encoded to type **factor**.
- Changes are stored back to **creditdata**

EXAMINE THE TRANSFORMED DATA

We can now easily examine the purpose of the first three loans with the command print(creditdata [1:3,'Purpose']). The purpose of first three loans can be seen plus additional information that there are 10 differents purposes of loan in the given dataset.

```
print(creditdata[1:3,'Purpose'])
```

```
## [1] radio/television radio/television education
## 10 Levels: business car (new) car (used) domestic appliances ... retraining
```

```
summary(creditdata$Purpose)
```

```
##            business           car (new)          car (used)
##                  97                 234                 103
## domestic appliances           education furniture/equipment
##                  12                  50                 181
##              others    radio/television             repairs
##                  12                 280                  22
##          retraining
##                   9
```

summary(creditdata$Purpose) is used to find the distribution of loan purpose.

We can also start to investigate the relation of loan type to other attributes as shown in the final two listings.

```
table(creditdata$Credit.history,creditdata$Good.Loan)
```

```
##
##                                                        BadLoan
##   all credits at this bank paid back duly                   28
##   critical account/other credits existing (not at this bank)      50
##   delay in paying off in the past                           28
##   existing credits paid back duly till now                 169
##   no credits taken/all credits paid back duly               25
##
##                                                       GoodLoan
##   all credits at this bank paid back duly                   21
##   critical account/other credits existing (not at this bank)     243
##   delay in paying off in the past                           60
##   existing credits paid back duly till now                 361
##   no credits taken/all credits paid back duly               15
```

The above command shows relationship between *“Credit.history”* and *“Good.Loan”*

Similarly the command below shows relationship between *“Personal.status.and.sex”* and *“Good.Loan”*. The table shows that 146 single males contribute to bad loans while 402 single males contribute to Good Loans which makes approx 1:3.

```
table(creditdata$Personal.status.and.sex,creditdata$Good.Loan)
```

```
##
##                                    BadLoan GoodLoan
##   female : divorced/separated/married     109      201
##   male : divorced/separated                20       30
##   male : married/widowed                   25       67
##   male : single                           146      402
```

SECTION 2. EXPLORE DATA IN R

In this section we will use different techniques to explore data. For this section we will use another dataset known as **custdata** placed at https://github.com/WinVector/ zmPDSwR/tree/master/Custdata (/url). We have placed the file in our working directory and we will read from there.

2.1 USING SUMMARY STATISTICS TO SPOT PROBLEMS

2.1.1 READ DATA

```
customerdata <- read.table('custdata.tsv',header=TRUE,sep="\t",quote="", fill=FALSE)
```

2.1.2 EXAMINE DATA

```
class(customerdata)
```

```
## [1] "data.frame"
```

```
dim(customerdata)
```

```
## [1] 1000   11
```

```
summary(customerdata)
```

```
##      custid        sex     is.employed         income
##  Min.   :   2068   F:440   Mode :logical   Min.   : -8700
##  1st Qu.: 345667   M:560   FALSE:73        1st Qu.: 14600
##  Median : 693403           TRUE :599       Median : 35000
##  Mean   : 698500           NA's :328       Mean   : 53505
##  3rd Qu.:1044606                           3rd Qu.: 67000
##  Max.   :1414286                           Max.   :615000
##
##              marital.stat health.ins
##  Divorced/Separated:155   Mode :logical
##  Married           :516   FALSE:159
##  Never Married     :233   TRUE :841
##  Widowed           : 96
##
##
##
##                        housing.type recent.move      num.vehicles
##  Homeowner free and clear    :157   Mode :logical   Min.   :0.000
##  Homeowner with mortgage/loan:412   FALSE:820       1st Qu.:1.000
##  Occupied with no rent       : 11   TRUE :124       Median :2.000
##  Rented                      :364   NA's :56        Mean   :1.916
##  NA's                        : 56                   3rd Qu.:2.000
##                                                     Max.   :6.000
##                                                     NA's   :56
##       age              state.of.res
##  Min.   :  0.0   California  :100
##  1st Qu.: 38.0   New York    : 71
##  Median : 50.0   Pennsylvania: 70
##  Mean   : 51.7   Texas       : 56
##  3rd Qu.: 64.0   Michigan    : 52
##  Max.   :146.7   Ohio        : 51
##                  (Other)     :600
```

Looking at the result we get the to know that **customerdata** is a “data frame”" of size **1000x11**. Moreover all the 11 columns have well defined names that can be used to explore the data.

summary command provides variety of **summary statistics**(*mean, variance, median, min, max and quantile*) on the numerical columns of the data frame, and count statistics on any categorical columns.

summary also helps in spotting the potential problems (*missing data* or *unlikely values*) in the data.

2.1.3 Typical Problems

The most common problems in the dataset are *missing values, invalid values and outliers*. *Data ranges* that are too narrow or wide can also be problem. We can spot such problems as a result of **summary** command but we also use visual tools to spot time as it is not easy to detect just with reading the tabular form of the data.

MISSING VALUES:

- We can see that the variable *is.employed* has *328 NA's* which means 30% data is missing.
- Similary three more variables, *housing.type, recent.move and num.vehicles*, have 56 missing values.

INVALID VALUES AND OUTLIERS:

- The variable *income* has negative value. Can income be negative? Mean of income is 53500 but max value is 615,000 which is a very high value consider the other values of same variable.
- The variable *age* has age 0 as well as age 146.7 of the clients. These are unexpected values for such type of dataset and they could be outliers.

DATA RANGE:

- The variable *income* has range from less than zero to more than half a million dollars. Is it a valid range or data has some error.

UNITS:

- The *income* data represent yearly wages in units of $1000. We have de fine da new variable_{c}ustomerdata$ Income_ to better understand it.

```
customerdata$Income = customerdata$income/1000
summary(customerdata$Income)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    -8.7    14.6    35.0    53.5    67.0   615.0
```

OBSERVATIONS:

The observations we gathered need to be further explored. If the values are missing that what could be possible reason for it? Maybe missing values have some meaning that we can discuss with the customer. We have to decide an appropriate action regarding missing data, should we include them or delete them or convert them to some appropriate values?

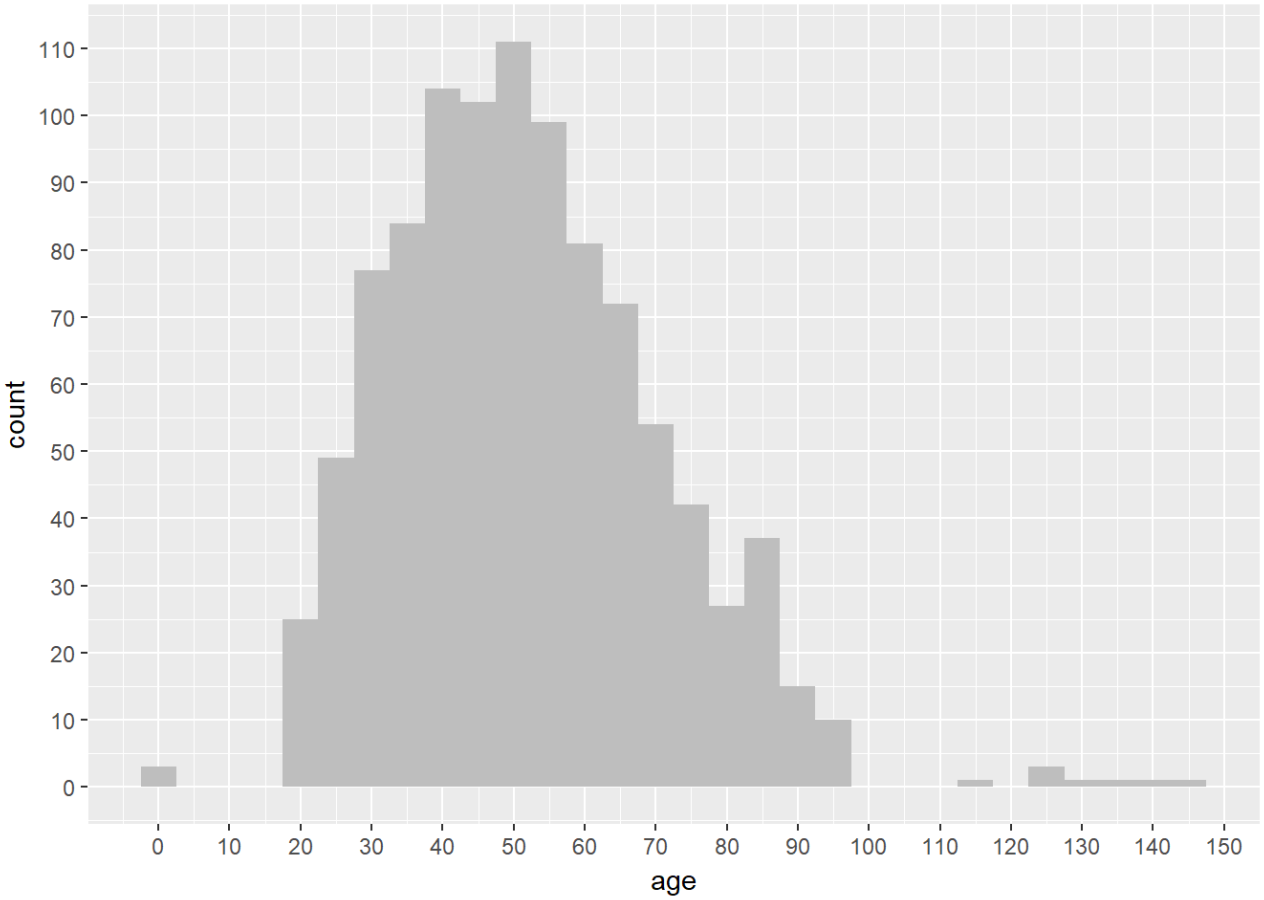
Similarly we have to find reaon for invalid values like negative values of income. They may also have some special meaning and than we have to take appropriate action regarding such values.

2.2 USING GRAPHICS and VISUALIZATION TO SPOT PROBLEMS

2.2.1 VISUALIZATION OF SINGLE VARIABLE

HISTOGRAM

```
library(ggplot2)
ggplot(customerdata) + geom_histogram(aes(x=age), binwidth=5, fill="grey")+
  scale_x_continuous(breaks = seq(0,160,10)) + scale_y_continuous(breaks = seq(0,220,1
0))
```

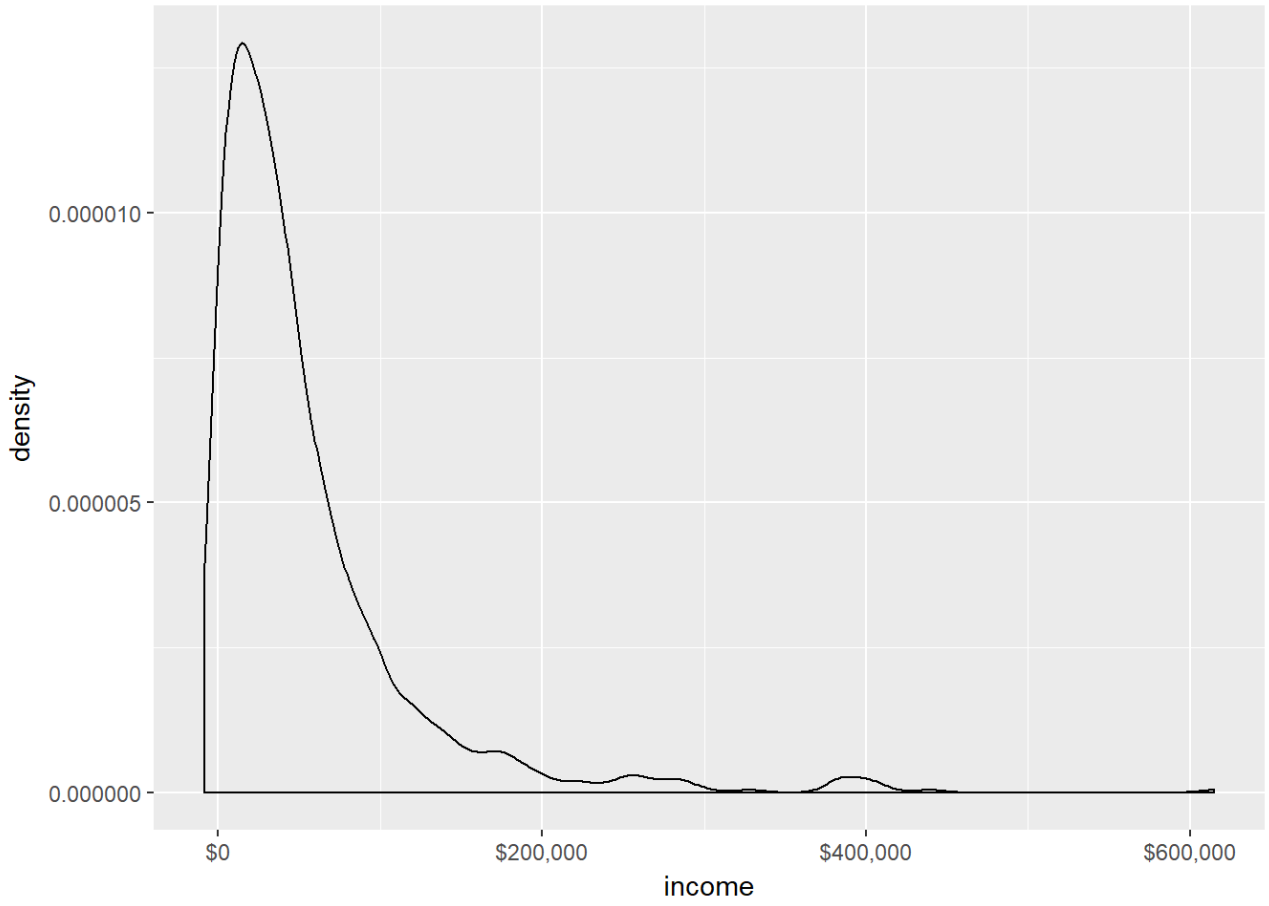


- Binwidth parameter tells the geom_histogram call to make bins of 5 years. Histogram of variable *age* shows that there are outliers in the data.
- One of the disadvantage of Histogram is that we have to set bin size in advance and it may not reflect the true information.

DENSITY PLOTS

Density plots can be thought of as “continous histogram” of a variable. The area under the density plot is equal to 1. We will plot the data of *income* variable in this case.

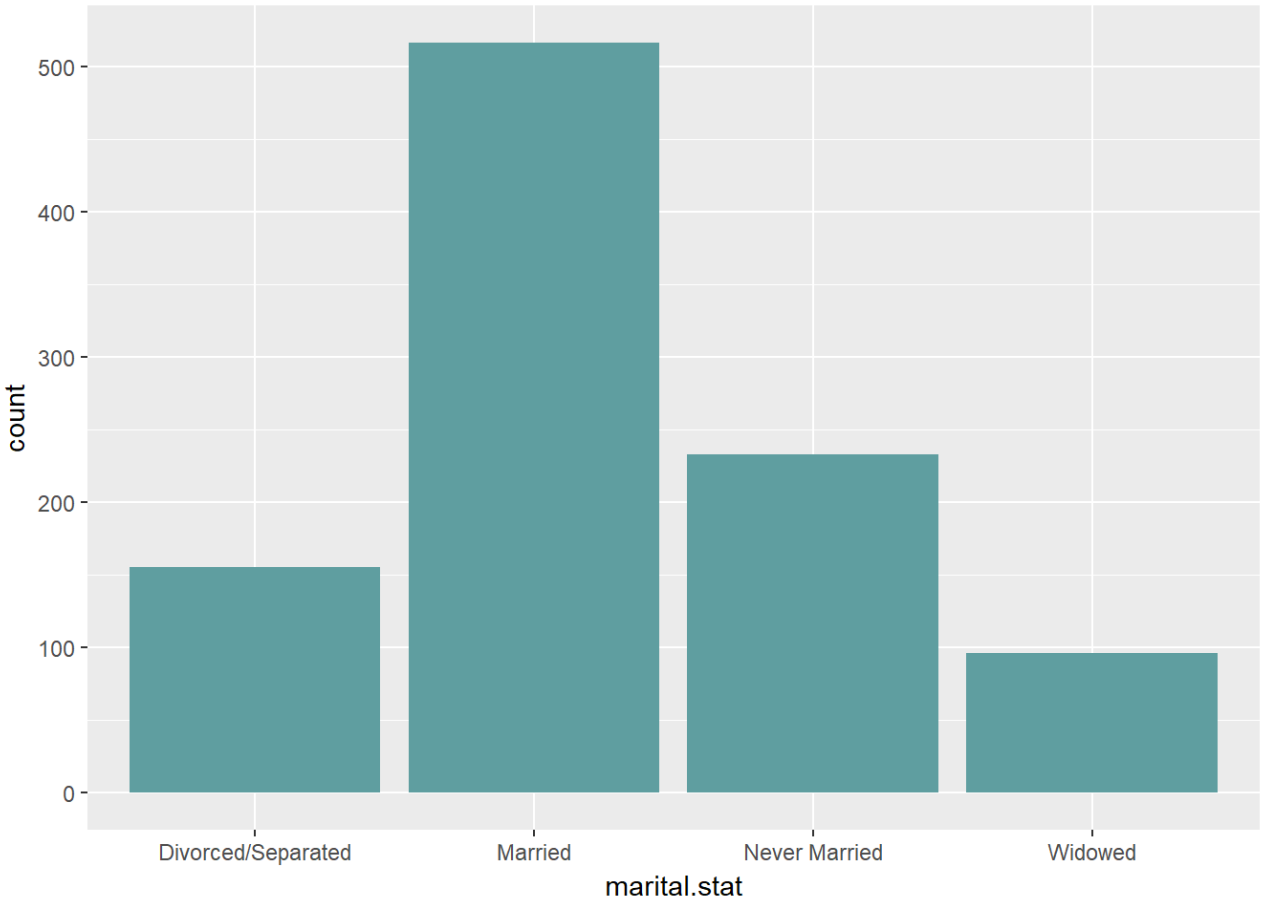
```
library(scales)
ggplot(customerdata) + geom_density(aes(x=income)) + scale_x_continuous(labels=dolla
r) +
   scale_y_continuous(labels=comma)
```



- Graph shows the distribution is concentrated at low end and so it is positively skewed. We use Density plots to see the overall shape of the curve.

BAR CHART

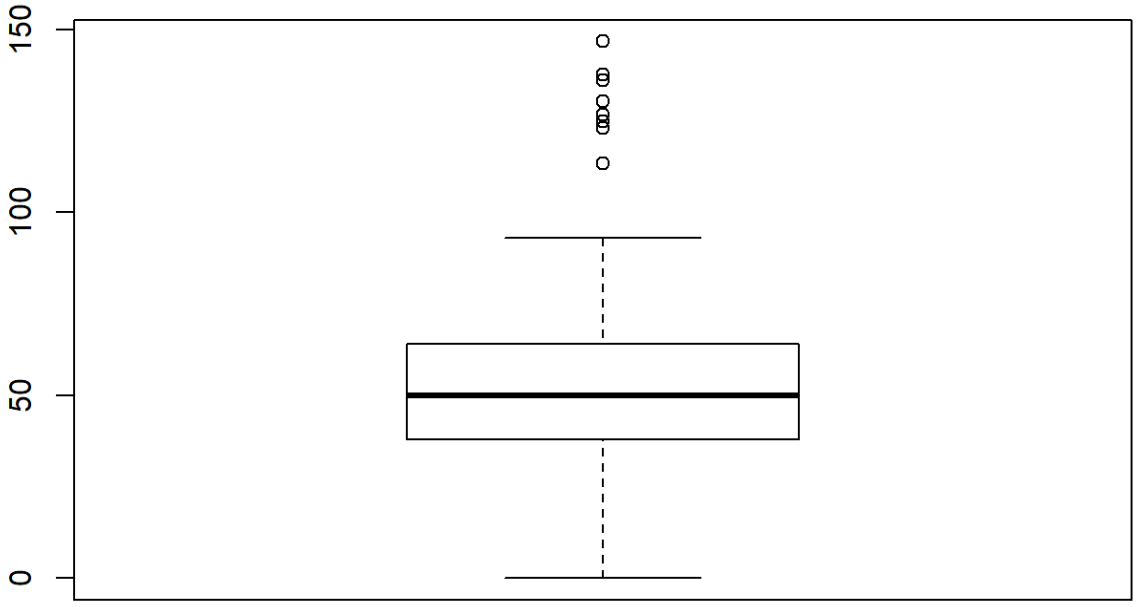
```
ggplot(customerdata) + geom_bar(aes(x=marital.stat), fill="cadetblue")
```



In the above bar graph we can see 4 caterogies of *martial.stat* on horizontal x-axis while *frequency* of each category on the vertical x-axis

BOX PLOT

```
boxplot(customerdata$age)
```



The box plot shows the median, minimum, maximum , first quartile, third quartile values of the age variable.

2.2.2 VISUALIZATION OF RELATION BETWEEN TWO VARIABLEs

CORRELATION BETWEEN AGE AND INCOME

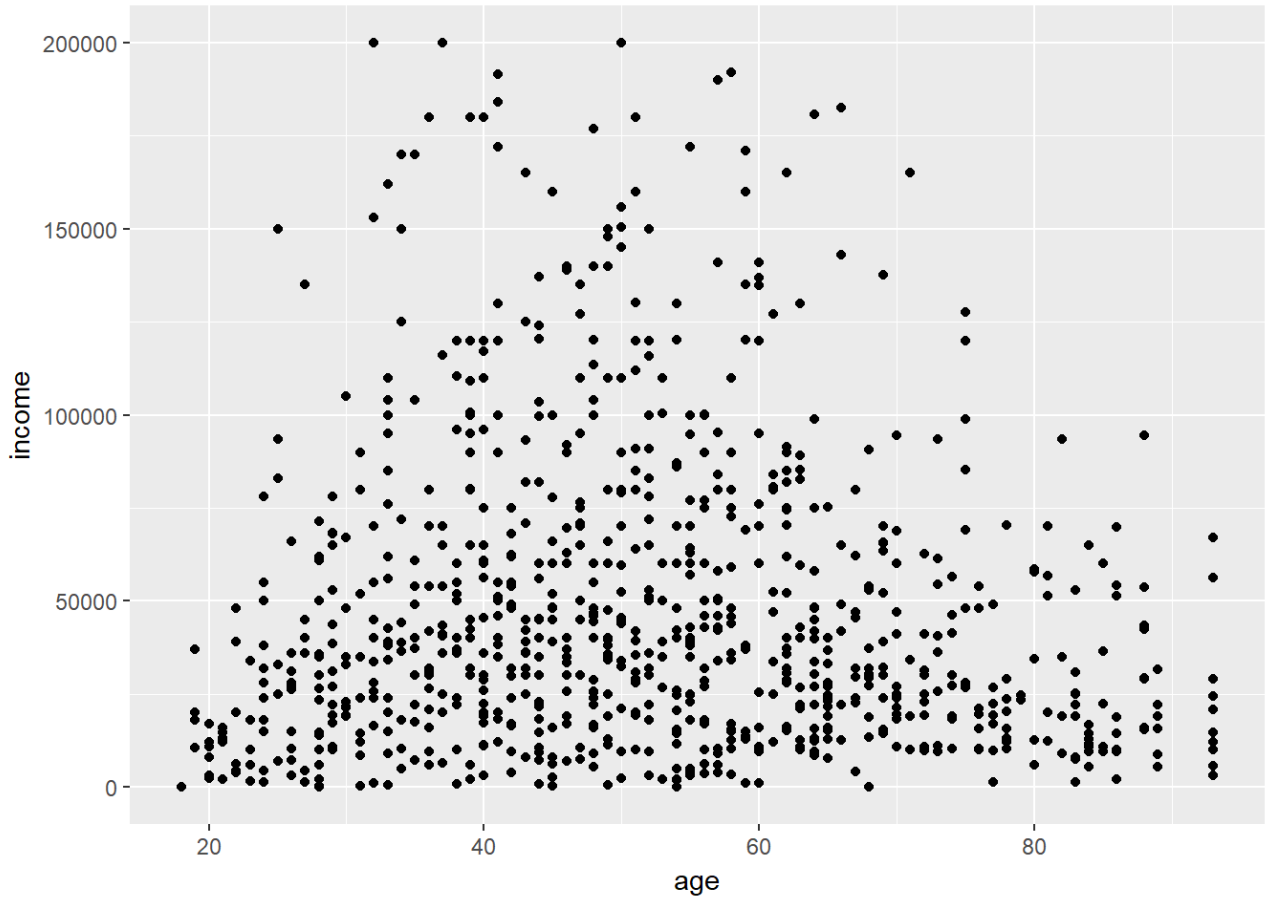
The negative correlation shows that if we increase the age than the income decreases as per the dataset.

```
custdata2 <- subset(customerdata, (customerdata$age > 0 & customerdata$age < 100 & cus
tomerdata$income > 0))
cor(custdata2$age, custdata2$income)
```

```
## [1] -0.02240845
```

SCATTER PLOTS AND SMOOTHING CURVES

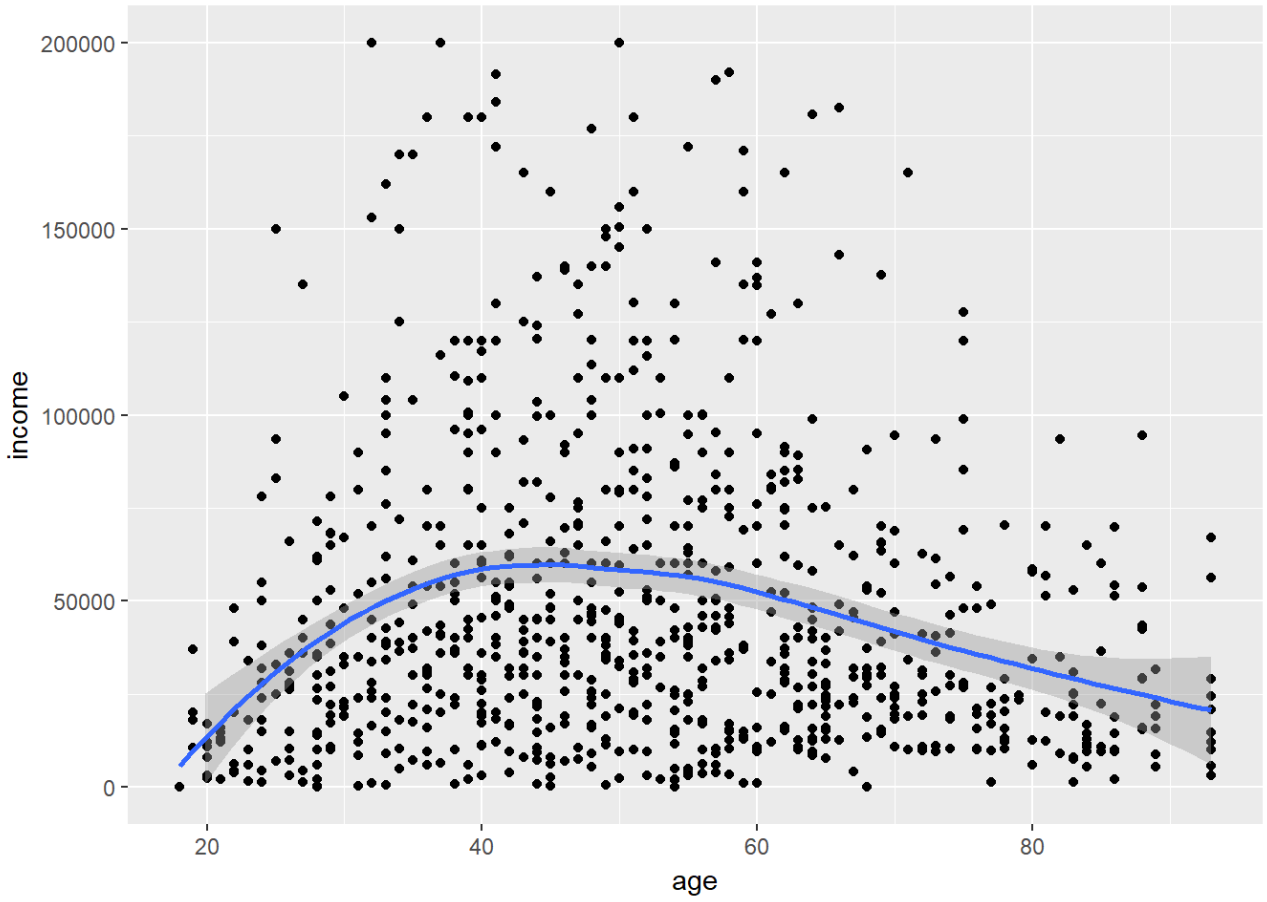
```
ggplot(custdata2, aes(x=age, y=income)) + geom_point() + ylim(0, 200000)
```



The scatter plot shows that income increases between the age 20-57 but it tends to decrease after the age of 57.

```
ggplot(custdata2, aes(x=age, y=income)) + geom_point() + geom_smooth() + ylim(0, 200000)
```

```
## `geom_smooth()` using method = 'loess'
```

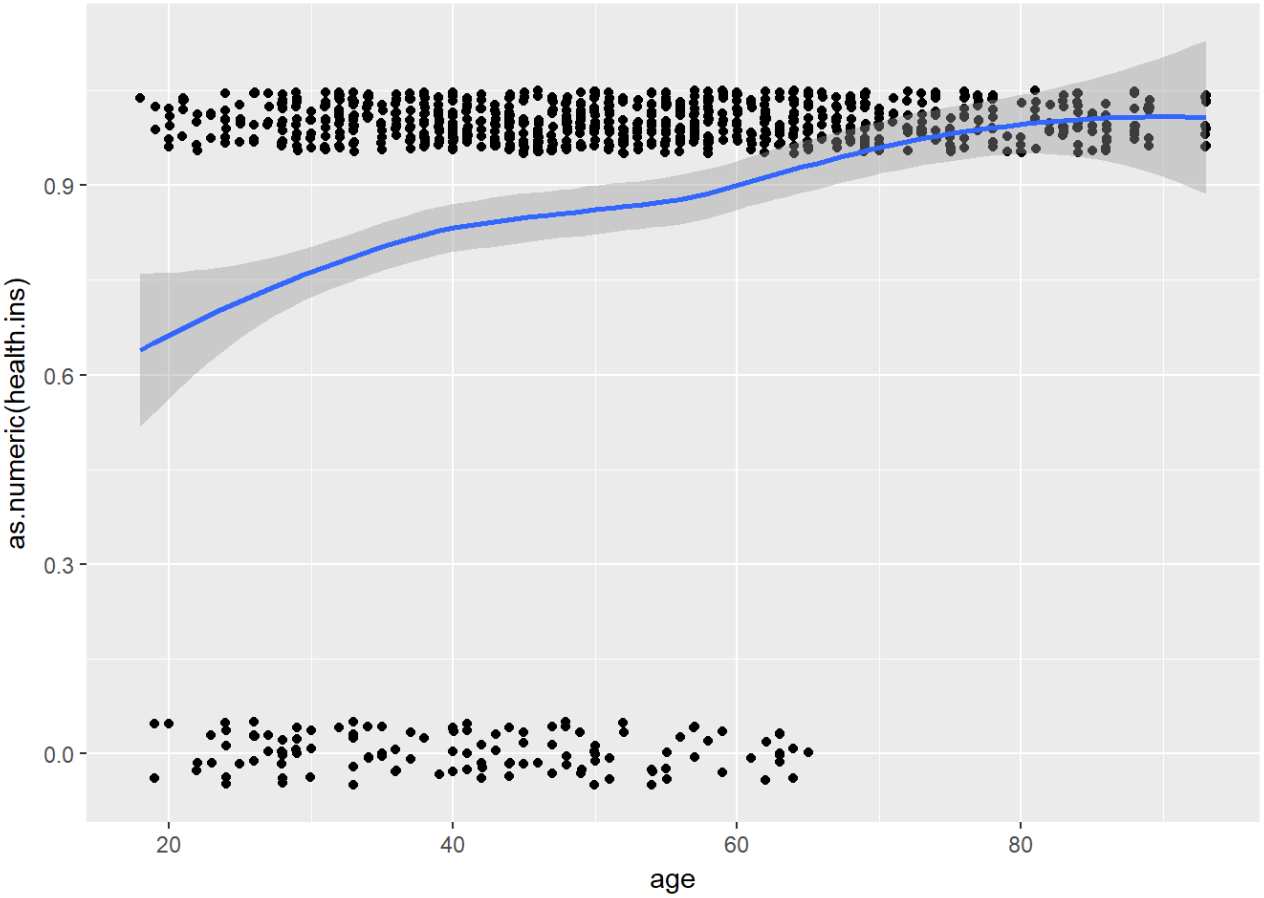


In this graph the rows with missing values are removed and it clarifies more clearly the range of age in which the income increases and than decreases.

Plotting the distribution of health.ins as a function of age

```
ggplot(custdata2, aes(x=age, y=as.numeric(health.ins))) + geom_point(position=position
_jitter(w=0.05, h=0.05)) + geom_smooth()
```

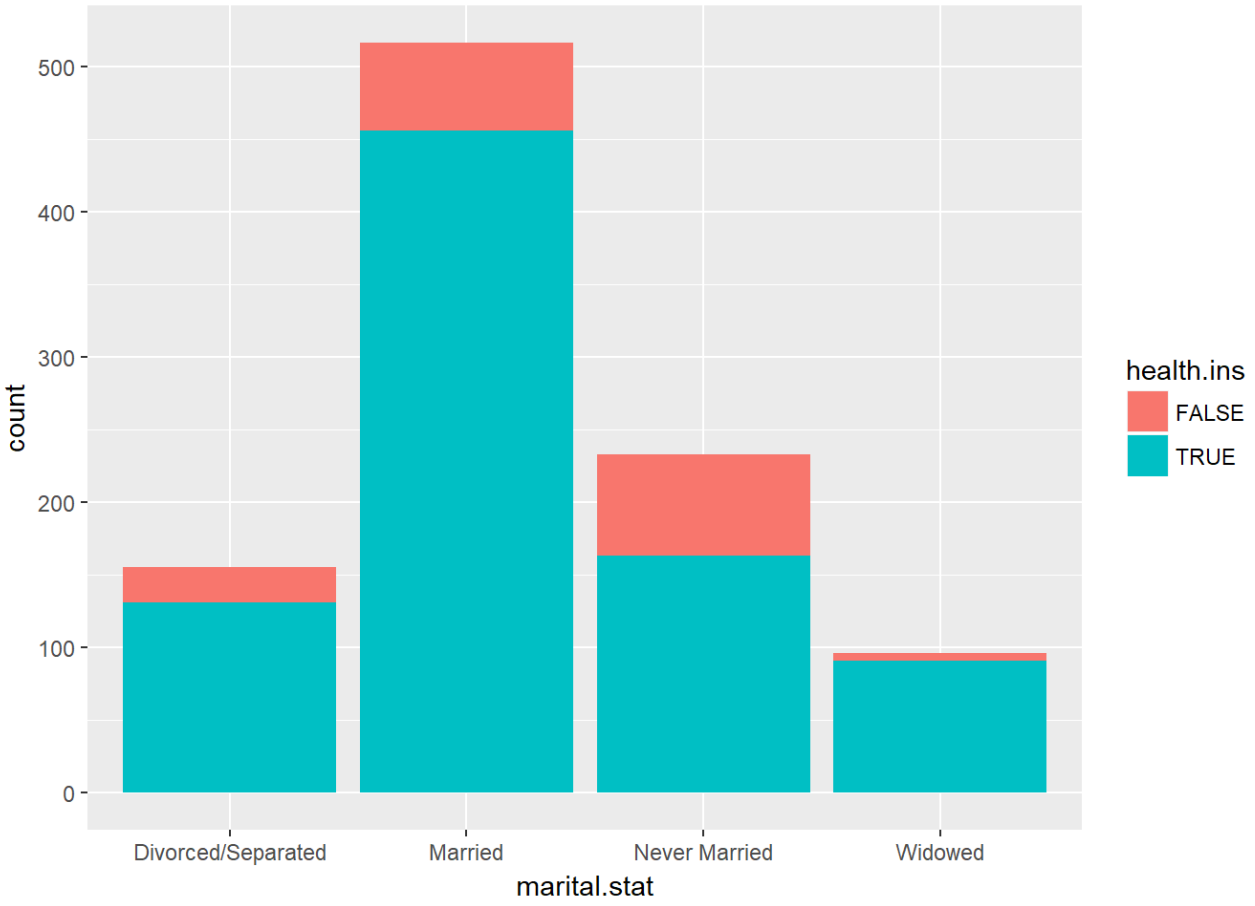
```
## `geom_smooth()` using method = 'loess'
```



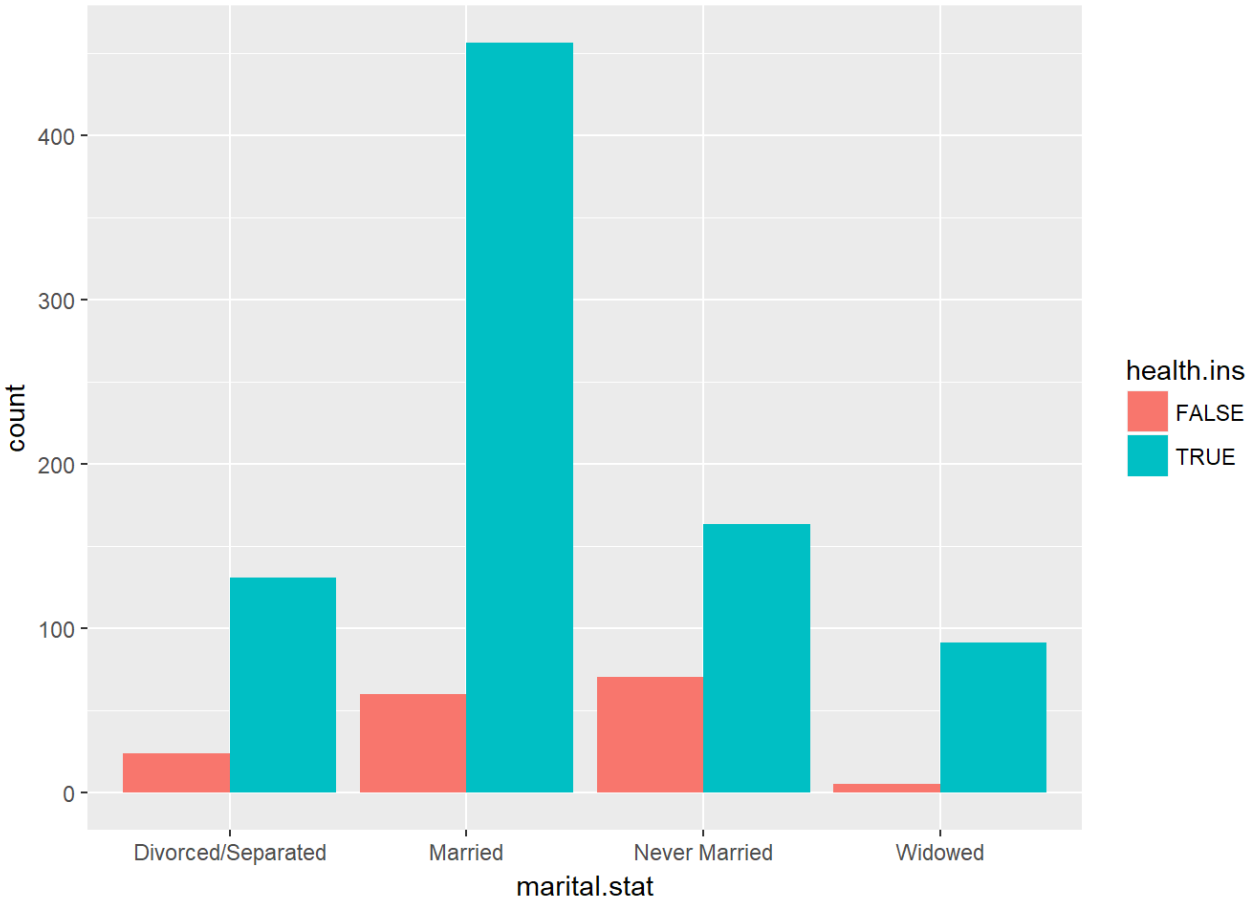
The above scatter plot is another visualization type showing a relationship between a continuous variable(age) and a Boolean(health.ins). The smoothing curve shows the fraction of customers with health insurance, as a function of age.

BAR CHARTS FOR TWO CATEGORICAL VARIABLES

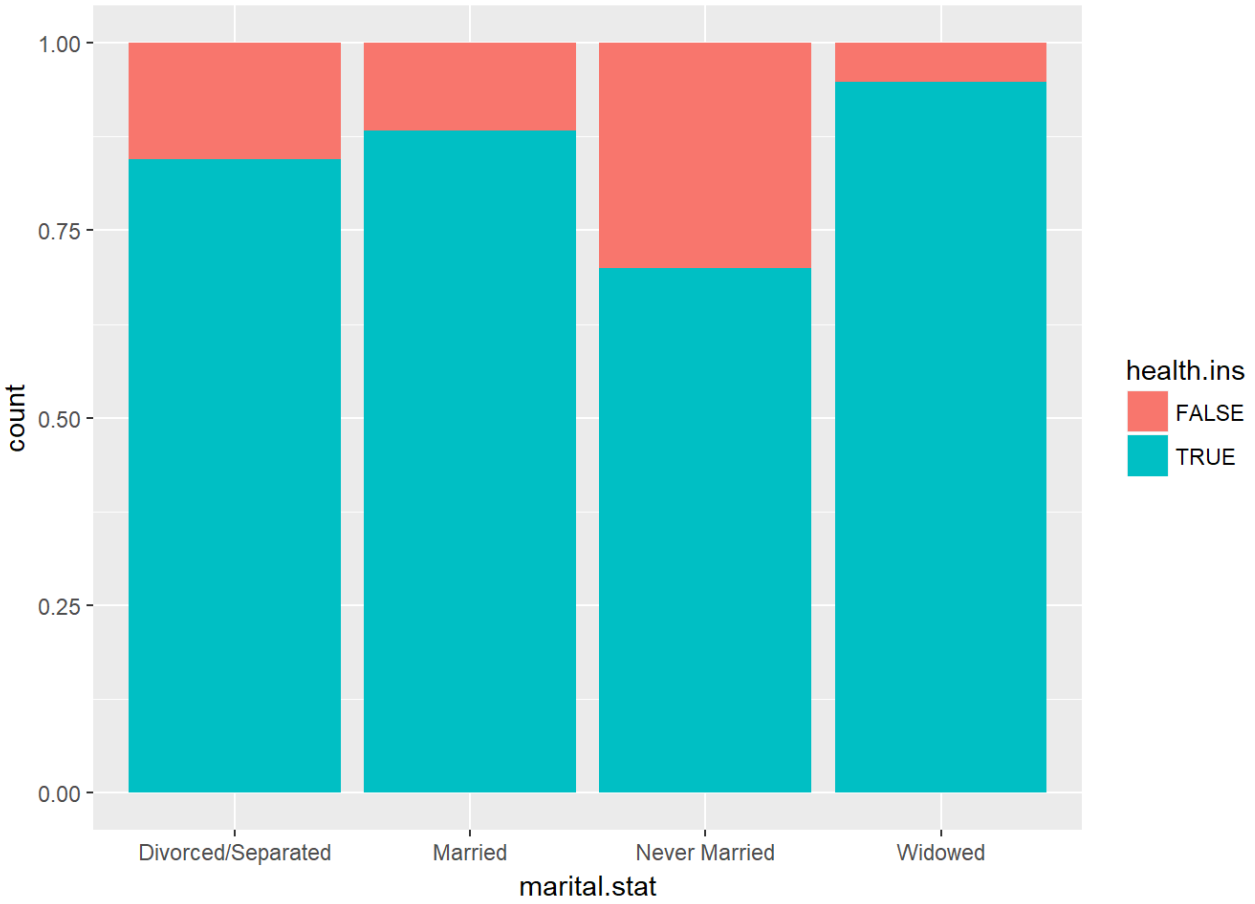
```
ggplot(customerdata) + geom_bar(aes(x=marital.stat, fill=health.ins))
```




```
ggplot(customerdata) + geom_bar(aes(x=marital.stat, fill=health.ins), position="dodg
e")
```




```
ggplot(customerdata) + geom_bar(aes(x=marital.stat, fill=health.ins), position="fill")
```



Different styles of bar charts are displayed. Pink color represents fraction of customers uninsured in the catergory of *martial.stat* while blue color shows the insured customers.

SECTION 3. MANAGING DATA

In this section we will apply R commands to

1. Fix the data issues
2. Organize the data for the modelling process.

We will start by reading the data from a new dataset that is *example-Data.rData* from https://github.com/WinVector/zmPDSwR/tree/master/Custdata (\url).

On executing the load command new variables are created in global environment

```
load("exampleData.rData")
```

```
summary(custdata)
```

```
##       state.of.res     custid        sex     is.employed
##  California  :114   Min.   :   2068   F:440   Mode :logical
##  New York    : 94   1st Qu.: 345667   M:560   FALSE:73
##  Pennsylvania: 63   Median : 693403           TRUE :599
##  Ohio        : 59   Mean   : 698500           NA's :328
##  Illinois    : 52   3rd Qu.:1044606
##  Texas       : 51   Max.   :1414286
##  (Other)     :567
##      income                   marital.stat health.ins
##  Min.   : -8700   Divorced/Separated:155   Mode :logical
##  1st Qu.: 14600   Married           :516   FALSE:159
##  Median : 35000   Never Married     :233   TRUE :841
##  Mean   : 53505   Widowed           : 96
##  3rd Qu.: 67000
##  Max.   :615000
##
##                        housing.type recent.move      num.vehicles
##  Homeowner free and clear    :157   Mode :logical   Min.   :0.000
##  Homeowner with mortgage/loan:412   FALSE:820       1st Qu.:1.000
##  Occupied with no rent       : 11   TRUE :124       Median :2.000
##  Rented                      :364   NA's :56        Mean   :1.916
##  NA's                        : 56                   3rd Qu.:2.000
##                                                     Max.   :6.000
##                                                     NA's   :56
##       age        is.employed.fix1   age.normalized     Median.Income
##  Min.   :  0.0   Length:1000        Min.   :-2.74074   Min.   :37427
##  1st Qu.: 38.0   Class :character   1st Qu.:-0.72626   1st Qu.:44819
##  Median : 50.0   Mode  :character   Median :-0.09011   Median :50118
##  Mean   : 51.7                      Mean   : 0.00000   Mean   :50919
##  3rd Qu.: 64.0                      3rd Qu.: 0.65207   3rd Qu.:55534
##  Max.   :146.7                      Max.   : 5.03516   Max.   :68187
##
##   income.norm            gp            income.lt.30K      age.range
##  Min.   :-0.1956   Min.   :0.0002281   Mode :logical   [0,25]  : 56
##  1st Qu.: 0.2812   1st Qu.:0.2618117   FALSE:562       (25,65] :732
##  Median : 0.6712   Median :0.5127602   TRUE :438       (65,Inf]:212
##  Mean   : 1.0781   Mean   :0.5016471
##  3rd Qu.: 1.3508   3rd Qu.:0.7405944
##  Max.   :11.7870   Max.   :0.9988350
##
##      Income
##  Min.   :     0
##  1st Qu.: 25000
##  Median : 45000
##  Mean   : 66199
##  3rd Qu.: 82000
##  Max.   :615000
##  NA's   :328
```

Obsevation

- 56 missing values in variables *housing.type*, *recent.move* and *num.vehicles*.
- 328 missing values in *Income* and *Is.employed* variables.
- The *Median.Income.y* , *Median.Income.x* and *Median.Income* have same information

3.1 CLEANING DATA

To clean the data first we will resolve the problem of missing values.

3.1.1 Check Location of missing data:

56 missing values in 3 variables

```
summary(custdata[is.na(custdata$housing.type), c("recent.move","num.vehicles")])
```

```
##  recent.move     num.vehicles
##  Mode:logical   Min.   : NA
##  NA's:56        1st Qu.: NA
##                 Median : NA
##                 Mean   :NaN
##                 3rd Qu.: NA
##                 Max.   : NA
##                 NA's   :56
```

We observed that there were three variables with 56 missing values. Here we will check that are those rows the same or different. Therefore in the *is.na* command we filter out *housing.type* missing values and compare with variables *recent.move* and *num.vehicles*. The result shows that the same 56 rows have missing values against the three variables.

We will the drop the rows with the missing values as they are less in number and probably it is save to drop them.

328 missing values in "is.employed"

```
custdata$is.employed.fix <- ifelse(is.na(custdata$is.employed), "missing", ifelse(cust
data$is.employed==T, "employed", "not employed"))
summary(as.factor(custdata$is.employed.fix))
```

```
##     employed      missing not employed
##          599          328           73
```

is.employed variable has 328 missing values which is one third of the customers. Here we created a new category for the variable called **missing**. We can see in the output the new category.

3.1.2 MISSING VALUES IN NUMERIC DATA

```
summary(custdata$Income)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##       0   25000   45000   66199   82000  615000     328
```

In this data there are 328 missing values.

3.1.3 FILL MISSING VALUES

If missing randomly

Here to fill the missing values on way is to replace by mean. We assume that the customers with missing income are distributed the same way as the others so using the mean the estimate will be correct on average

```
meanIncome <- mean(custdata$Income, na.rm=T)
Income.fix <- ifelse(is.na(custdata$Income), meanIncome, custdata$Income)
summary(Income.fix)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0   35000   66199   66199   66199  615000
```

```
summary(custdata)
```

```
##       state.of.res     custid        sex     is.employed
##  California  :114   Min.   :   2068   F:440   Mode :logical
##  New York    : 94   1st Qu.: 345667   M:560   FALSE:73
##  Pennsylvania: 63   Median : 693403           TRUE :599
##  Ohio        : 59   Mean   : 698500           NA's :328
##  Illinois    : 52   3rd Qu.:1044606
##  Texas       : 51   Max.   :1414286
##  (Other)     :567
##      income                   marital.stat health.ins
##  Min.   : -8700   Divorced/Separated:155   Mode :logical
##  1st Qu.: 14600   Married           :516   FALSE:159
##  Median : 35000   Never Married     :233   TRUE :841
##  Mean   : 53505   Widowed           : 96
##  3rd Qu.: 67000
##  Max.   :615000
##
##                        housing.type recent.move      num.vehicles
##  Homeowner free and clear    :157   Mode :logical   Min.   :0.000
##  Homeowner with mortgage/loan:412   FALSE:820       1st Qu.:1.000
##  Occupied with no rent       : 11   TRUE :124       Median :2.000
##  Rented                      :364   NA's :56        Mean   :1.916
##  NA's                        : 56                   3rd Qu.:2.000
##                                                     Max.   :6.000
##                                                     NA's   :56
##       age        is.employed.fix1   age.normalized     Median.Income
##  Min.   :  0.0   Length:1000        Min.   :-2.74074   Min.   :37427
##  1st Qu.: 38.0   Class :character   1st Qu.:-0.72626   1st Qu.:44819
##  Median : 50.0   Mode  :character   Median :-0.09011   Median :50118
##  Mean   : 51.7                      Mean   : 0.00000   Mean   :50919
##  3rd Qu.: 64.0                      3rd Qu.: 0.65207   3rd Qu.:55534
##  Max.   :146.7                      Max.   : 5.03516   Max.   :68187
##
##   income.norm            gp            income.lt.30K      age.range
##  Min.   :-0.1956   Min.   :0.0002281   Mode :logical   [0,25]  : 56
##  1st Qu.: 0.2812   1st Qu.:0.2618117   FALSE:562       (25,65] :732
##  Median : 0.6712   Median :0.5127602   TRUE :438       (65,Inf]:212
##  Mean   : 1.0781   Mean   :0.5016471
##  3rd Qu.: 1.3508   3rd Qu.:0.7405944
##  Max.   :11.7870   Max.   :0.9988350
##
##      Income       is.employed.fix
##  Min.   :     0   Length:1000
##  1st Qu.: 25000   Class :character
##  Median : 45000   Mode  :character
##  Mean   : 66199
##  3rd Qu.: 82000
##  Max.   :615000
##  NA's   :328
```

If missing systematically

We convert the numeric data into categorical data in such case.

First we make a vector of income ranges.

```
breaks <-c(0, 10000, 50000, 100000, 250000, 1000000)
```

We Cut the data into income ranges and we include the lowest value.

```
Income.groups <- cut(custdata$Income, breaks=breaks, include.lowest=T)
summary(Income.groups)
```

```
##       [0,1e+04]   (1e+04,5e+04]   (5e+04,1e+05] (1e+05,2.5e+05]
##              63             312             178              98
## (2.5e+05,1e+06]            NA's
##              21             328
```

Then * missing values category are given a new name as *"no income"* and * the class type is convered to factor to make them as categories.

```
Income.groups <- as.character(Income.groups)
Income.groups <- ifelse(is.na(Income.groups), "no income", Income.groups)
summary(as.factor(Income.groups))
```

```
##   (1e+04,5e+04] (1e+05,2.5e+05] (2.5e+05,1e+06]   (5e+04,1e+05]
##             312              98              21             178
##       [0,1e+04]       no income
##              63             328
```

3.2 DATA TRANSFORMATION

3.2.1 Normalizing income by state

medianincome is a global variable created when the excuted the ***load*** command.

```
summary(medianincome)
```

```
##         State    Median.Income
##            : 1   Min.   :37427
##  Alabama   : 1   1st Qu.:47483
##  Alaska    : 1   Median :52274
##  Arizona   : 1   Mean   :52655
##  Arkansas  : 1   3rd Qu.:57195
##  California: 1   Max.   :68187
##  (Other)   :46
```

We also normalize the income by *Median.Income*.

```
custdata$income.norm <- with(custdata, income/Median.Income)
summary(custdata$income.norm)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.1956  0.2812  0.6712  1.0781  1.3508 11.7870
```

3.2.2 Converting age into ranges

In the brks we have selected the range of interest. There are three range categories which are

- 0-25
- 25-65
- 65-Inf

```
brks <- c(0, 25, 65, Inf)
```

Next we cut the data into age ranges. The output of *cut* is factor variable.

```
custdata$age.range <- cut(custdata$age, breaks=brks, include.lowest=T)
summary(custdata$age.range)
```

```
##   [0,25]  (25,65] (65,Inf]
##       56      732      212
```

3.2.3 NORMALIZATION AND RESCALING

```
summary(custdata$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    38.0    50.0    51.7    64.0   146.7
```

In this example we have normalized the *age* variable. Less than 1 signifies very young customer.

```
meanage <- mean(custdata$age)
custdata$age.normalized <- custdata$age/meanage
summary(custdata$age.normalized)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.7350  0.9671  1.0000  1.2379  2.8372
```

3.2.4 Summarizing age

Another way of normalizing the data is to use mean and standard deviation. This is especially useful when the data distribution is roughly symmetrical.

- Customers less than -1 signifies customers youger than typical
- Customers greater than 1 signify customers older than typical.

```
meanage <- mean(custdata$age)
stdage <- sd(custdata$age)
custdata$age.normalized <- (custdata$age-meanage)/stdage
summary(custdata$age.normalized)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.74074 -0.72626 -0.09011  0.00000  0.65207  5.03516
```