The following libraries are necessary and need it to be downloaded and up to date, in order to develop the analysis required.

FB1 is a named assigned to the data provided, it can be renamed if the sponsor chooses to do so.

```
# Load in library
library(rpart)
library(rpart.plot)
library(readr)
library(dplyr)
library(foreign)
library(ggplot2)
library(class)
library(useful)
library(tidyverse)
library(caret)
library(gbm)
library(ISLR)
library(skimr)
library(party)
library(tree)
library(DAAG)
library(mlbench)
library(pROC)
library(rattle)

options(scipen = 999)

FB1 <- read_csv("FB1.csv")
fb1<- FB1[-1]
```

As stated on the code the following variables were eliminated because there were no values at all. Therefore, the variables are meaningless to our analysis.

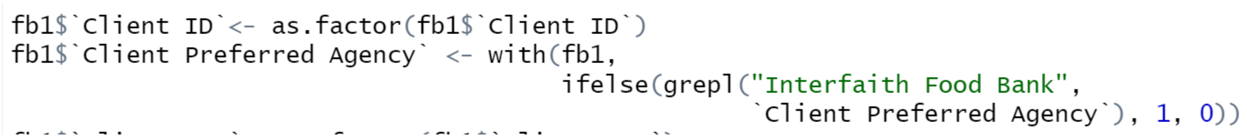
```
#eliminating the following rows because they have no values at all
fb1 %>% select(-c(`Visited Agency`, `Visited Agency (Short Name)`,
                  `Client Race Category`, `Client Student Loans`,
                  `Client Scholarships`, `Client No Income`,
                  `Client Status`))-> fb1
```

Next, the variables containing only dates related to customers were assigned to a variable just to make the regression tree more visible. Dates do not affect our analysis, but the variables were not eliminated, rather they were assigned just in case they are required for a specific analysis.

```
#these data set is assigned because it contains only dates
fb1 %>% select(c(`Client First Visit-Date`, `Client First Visit- Personal Tab`,
                 `Client CSFP Initial Certification Date`,
                 `Client CSFP Next Recertification Date`,
                 `Client CSFP Next Check-In Date`, `Recorded At`))-> Datfb1
```

After the variables containing dates were saved and assigned to a Datfb1 variable, then those variables are eliminated from our data set.

```
fb1 %>% select(-c(`Client First Visit-Date`, `Client First Visit- Personal Tab`,
                  `Client CSFP Initial Certification Date`,
                  `Client CSFP Next Recertification Date`,
                  `Client CSFP Next Check-In Date`, `Recorded At`))-> fb1
#fb1$`Client First Visit- Personal Tab`<- as.Date(fb1$`Client First Visit- Personal Tab`,
# format = "%d-%m-%Y")
##################################################################################
```



```r
fb1$`Client ID`<- as.factor(fb1$`Client ID`)
fb1$`Client Preferred Agency` <- with(fb1,
                                      ifelse(grepl("Interfaith Food Bank",
                                                   `Client Preferred Agency`), 1, 0))
```

Variables characterized as characters need to be converted to factors because R does not handle characters as values, rather as factors. Notice the variable Client ID contains numbers but for our analysis the Client ID variable will helps us identify a specific client. Thus, it will be considered a categorical variable but needs to be converted to factor.

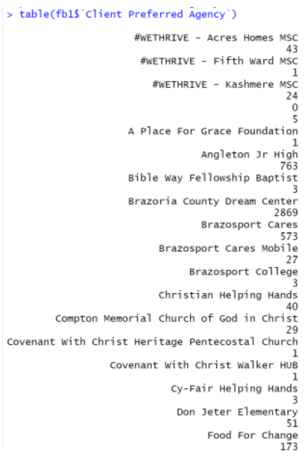
The next function in R Studio is creating a new column our data frame fb1 named "Client Preferred Agency", and populating it with binary values (either 1 or 0) based on whether the agency name "Interfaith Food Bank" appears in the original "Client Preferred Agency" column.fb1$'Client Preferred Agency': This accesses the "Client Preferred Agency" column in the fb1 data frame using the $ operator.

Here's a step-by-step breakdown of what's happening in the code:

1. **<- with(fb1, ...)**: This specifies that the subsequent code will be executed within the context of the **fb1** data frame.
2. **ifelse(grepl("Interfaith Food Bank",** Client Preferred Agency**), 1, 0)**: This is a conditional statement that checks whether the string "Interfaith Food Bank" is present in the "Client Preferred Agency" column of **fb1**. If it is, the function returns 1. If it isn't, the function returns 0. The **ifelse** function in R allows for vectorized if-else statements, which means it can operate on entire columns of data at once.
3. The result of step 3 is then assigned to the new column "Client Preferred Agency" in **fb1**.

In summary, this code is creating a new binary column that indicates whether or not each row in the original "Client Preferred Agency" column contains the string "Interfaith Food Bank".

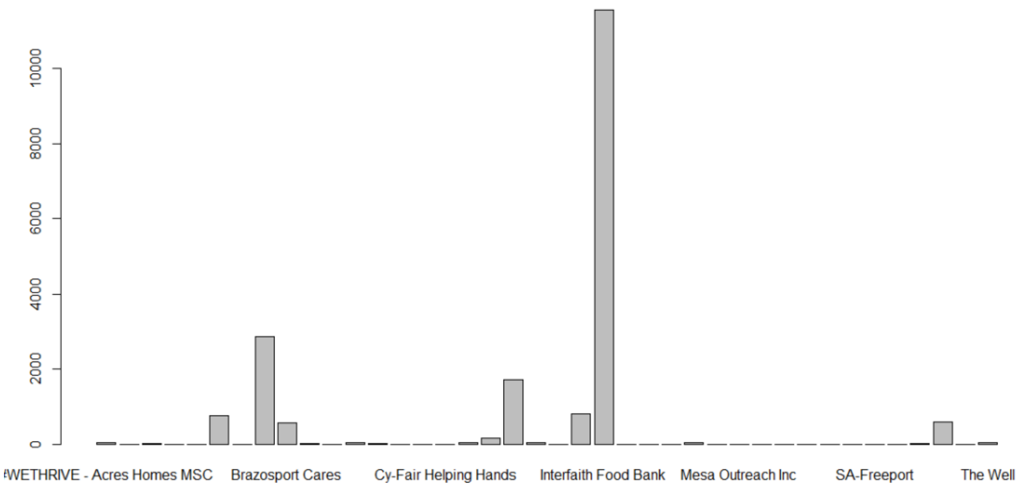
In addition “Interfaith Food Bank” was chosen because diving through the Client Preferred Agency database we observe the agency Interfaith Food Bank had the most visitors among the rest of the agencies. Notice that this can be adjusted and if wanted to test the regression trees, all is needed is to change the variable name.



```
> table(fb1$`Client Preferred Agency`)

                     #WETHRIVE - Acres Homes MSC
                                              43
                      #WETHRIVE - Fifth Ward MSC
                                               1
                        #WETHRIVE - Kashmere MSC
                                              24
                                               0
                                               5
                    A Place For Grace Foundation
                                               1
                                Angleton Jr High
                                             763
                    Bible Way Fellowship Baptist
                                               3
                    Brazoria County Dream Center
                                            2869
                                Brazosport Cares
                                             573
                         Brazosport Cares Mobile
                                              27
                              Brazosport College
                                               3
                         Christian Helping Hands
                                              40
        Compton Memorial Church of God in Christ
                                              29
Covenant With Christ Heritage Pentecostal Church
                                               1
                 Covenant With Christ Walker HUB
                                               1
                           Cy-Fair Helping Hands
                                               3
                            Don Jeter Elementary
                                              51
                                 Food For Change
                                             173
```

If necessary the Client Preferred Agency can be plotted to confirm our decision, using the following command:

```
plot(fb1$`Client Preferred Agency`)
```



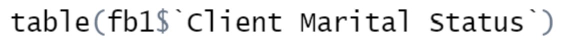
```
fb1$`Client Marital Status` <- factor(fb1$`Client Marital Status`,
                                      levels = c("0", "common_law", "declined_to_answer","did_not_ask",
                                                 "divorced", "do_not_know", "married", "separated",
                                                 "single", "widowed"),
                                      labels = c(0,1,2,3,4,5,6,7,8,9))
```

This function in R Studio is creating a new factor variable in the **fb1** data frame called "Client Marital Status", based on the values in the existing "Client Marital Status" column.

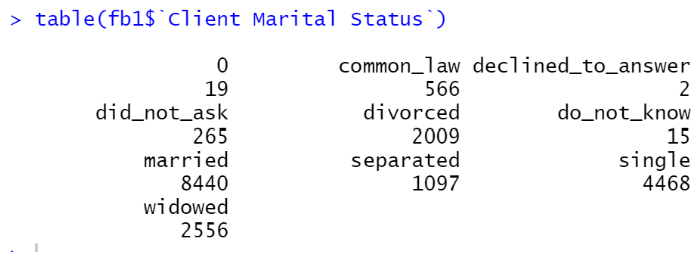
Here's a step-by-step breakdown of what's happening in the code:

1. **fb1$'Client Marital Status'**: This accesses the "Client Marital Status" column in the **fb1** data frame using the **$** operator.
2. **factor(fb1$'Client Marital Status', ...)**: This function creates a new factor variable based on the values in the "Client Marital Status" column of **fb1**. Factors are a special data type in R used for categorical variables.
3. **levels = c("0", "common_law", "declined_to_answer", "did_not_ask", "divorced", "do_not_know", "married", "separated", "single", "widowed")**: This specifies the levels of the new factor variable. In this case, the factor levels are a set of character strings that correspond to the possible values in the original "Client Marital Status" column.
4. **labels = c(0,1,2,3,4,5,6,7,8,9)**: This specifies the labels to be assigned to each factor level. In this case, the function is assigning numeric values to each factor level, with "0" being assigned the label "0", "common_law" being assigned the label "1", and so on.
5. The result of step 2 is then assigned to the new column "Client Marital Status" in **fb1**

Using the following function, we can extract all the characteristics on the Client Marital Status:

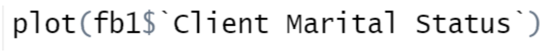


```
table(fb1$`Client Marital Status`)
```

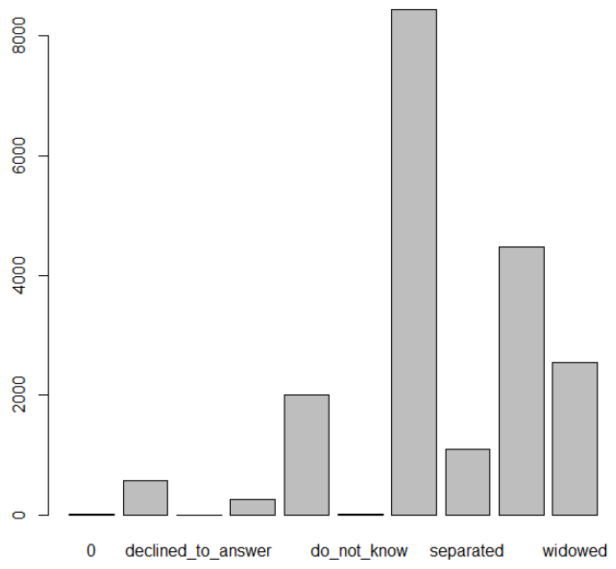


```
> table(fb1$`Client Marital Status`)

                  0          common_law declined_to_answer
                 19                 566                  2
        did_not_ask            divorced        do_not_know
                265                2009                 15
            married           separated             single
               8440                1097               4468
            widowed
               2556
```

In addition, the function can be plot to visualize the characteristics of the variable, using the following function:

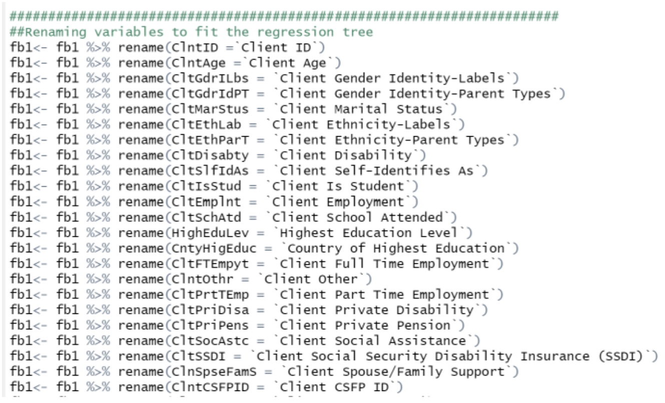


```
plot(fb1$`Client Marital Status`)
```



The same procedure was applied to the rest of the variables.

The following code we just rename the variables again to better fit the regression tree.



```
###########################################################################
##Renaming variables to fit the regression tree
fb1<- fb1 %>% rename(ClntID =`Client ID`)
fb1<- fb1 %>% rename(ClntAge =`Client Age`)
fb1<- fb1 %>% rename(CltGdrILbs = `Client Gender Identity-Labels`)
fb1<- fb1 %>% rename(CltGdrIdPT = `Client Gender Identity-Parent Types`)
fb1<- fb1 %>% rename(CltMarStus = `Client Marital Status`)
fb1<- fb1 %>% rename(CltEthLab = `Client Ethnicity-Labels`)
fb1<- fb1 %>% rename(CltEthParT = `Client Ethnicity-Parent Types`)
fb1<- fb1 %>% rename(CltDisabty = `Client Disability`)
fb1<- fb1 %>% rename(CltSlfIdAs = `Client Self-Identifies As`)
fb1<- fb1 %>% rename(CltIsStud = `Client Is Student`)
fb1<- fb1 %>% rename(CltEmplnt = `Client Employment`)
fb1<- fb1 %>% rename(CltSchAtd = `Client School Attended`)
fb1<- fb1 %>% rename(HighEduLev = `Highest Education Level`)
fb1<- fb1 %>% rename(CntyHigEduc = `Country of Highest Education`)
fb1<- fb1 %>% rename(CltFTEmpyt = `Client Full Time Employment`)
fb1<- fb1 %>% rename(ClntOthr = `Client Other`)
fb1<- fb1 %>% rename(CltPrtTEmp = `Client Part Time Employment`)
fb1<- fb1 %>% rename(CltPriDisa = `Client Private Disability`)
fb1<- fb1 %>% rename(CltPriPens = `Client Private Pension`)
fb1<- fb1 %>% rename(CltSocAstc = `Client Social Assistance`)
fb1<- fb1 %>% rename(CltSSDI = `Client Social Security Disability Insurance (SSDI)`)
fb1<- fb1 %>% rename(ClnSpseFamS = `Client Spouse/Family Support`)
fb1<- fb1 %>% rename(ClntCSFPID = `Client CSFP ID`)
```

names(fb1): will print out all the variable names in the R console

attach(fb1): attaching the names will help you obtained the information needed in the dataset

sum(is.na(fb1)) : count total missing values in entire data frame

apply(X = is.na(fb1), MARGIN = 2, FUN = sum): prints the number of N/A values for each variable as a list

fb1 <- fb1[complete.cases(fb1),]: deletes all the missing values in the dataset without affecting the analysis.

sum(is.na(fb1)): prints the number of missing values after using the complete case function above

dim(fb1): prints out the numbers of rows, and columns. For our dataset the following is true: 19,428 101

```r
set.seed(84310)
partition <- createDataPartition(y = fb1$`MothHoldInc`,
                                 p = 0.8, list = FALSE)
```

Setting a seed guarantees the same output no matter how many times you run the program.

The partition variable above is creating a partition of the data in the **fb1** data frame by randomly selecting a subset of rows for training a machine learning model.

Here's a step-by-step breakdown of what's happening in the code:

1. **fb1$'MothHoldInc'**: This accesses the "MothHoldInc" column in the **fb1** data frame using the **$** operator. This column likely contains the target variable for the machine learning model.
2. **createDataPartition(y = fb1$'MothHoldInc', p = 0.8, list = FALSE)**: This function is part of the **caret** package in R, which provides tools for training and evaluating machine learning models.
3. **y = fb1$'MothHoldInc'**: This specifies the target variable for the machine learning model.
4. **p = 0.8**: This specifies the proportion of the data to be included in the training set. In this case, 80% of the data will be used for training.
5. **list = FALSE**: This specifies that the function should return a vector of row indices rather than a list of data frames.
6. The result of step 2 is then assigned to the **partition** variable.

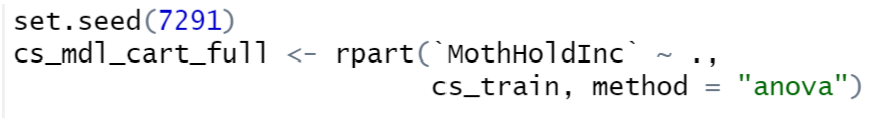
The following functions split the data into two parts, one as a training set and the rest as a test.

```r
cs_train <- fb1[partition, ]
cs_test <- fb1[-partition, ]
```

This function in R Studio is building a decision tree regression model on the **cs_train** data frame, using the Recursive Partitioning and Regression Trees (rpart) algorithm, to predict the target variable **MothHoldInc**.

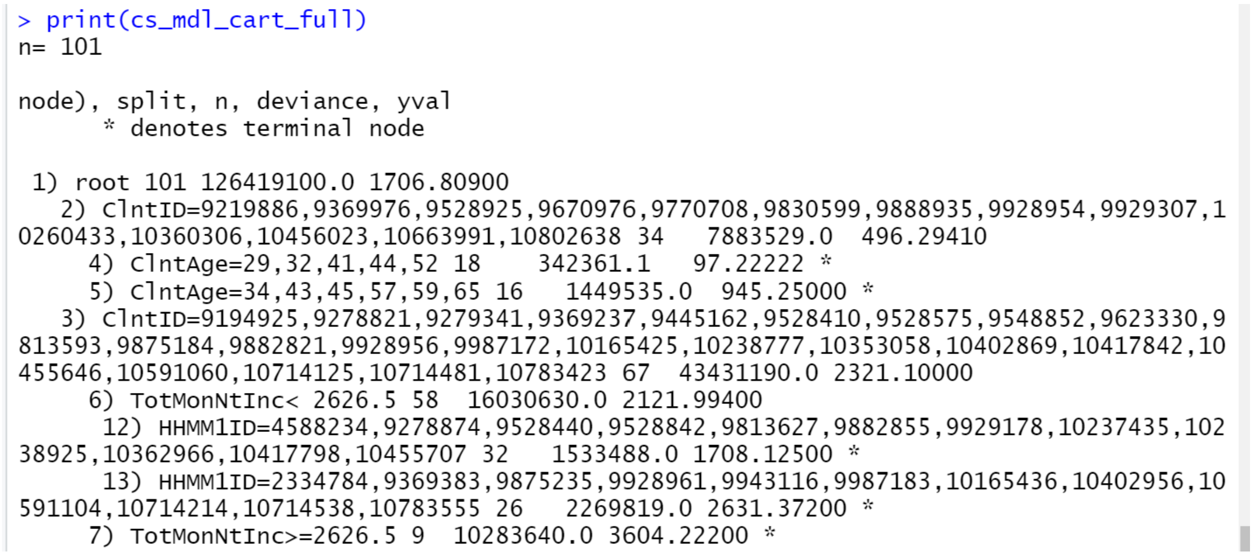
Here's a step-by-step breakdown of what's happening in the code:

1. **rpart(**MothHoldInc **~ ., cs_train, method = "anova")**: This function is using the **rpart** algorithm to build a decision tree model, where **MothHoldInc** is the target variable to be predicted and **.** represents all the other variables in the **cs_train** data frame.
2. **cs_train**: This is the training data frame that the model will be fit to.
3. **method = "anova"**: This specifies the method used for splitting the data at each node of the decision tree. In this case, the **anova** method is used to perform an analysis of variance to determine the best variable to split the data.
4. The resulting decision tree model is then assigned to the **cs_mdl_cart_full** variable.



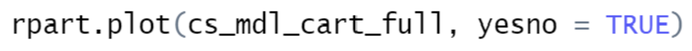
```
set.seed(7291)
cs_mdl_cart_full <- rpart(`MothHoldInc` ~ .,
                          cs_train, method = "anova")
```

The function **print(cs_mdl_cart_full)** will allow us to print the information that will be used to generate the regression tree. Notice, Client ID, Client Age, Total Monthly Income, HH Member 1 ID, and Total Monthly Net Income, were the most important variables in our analysis.

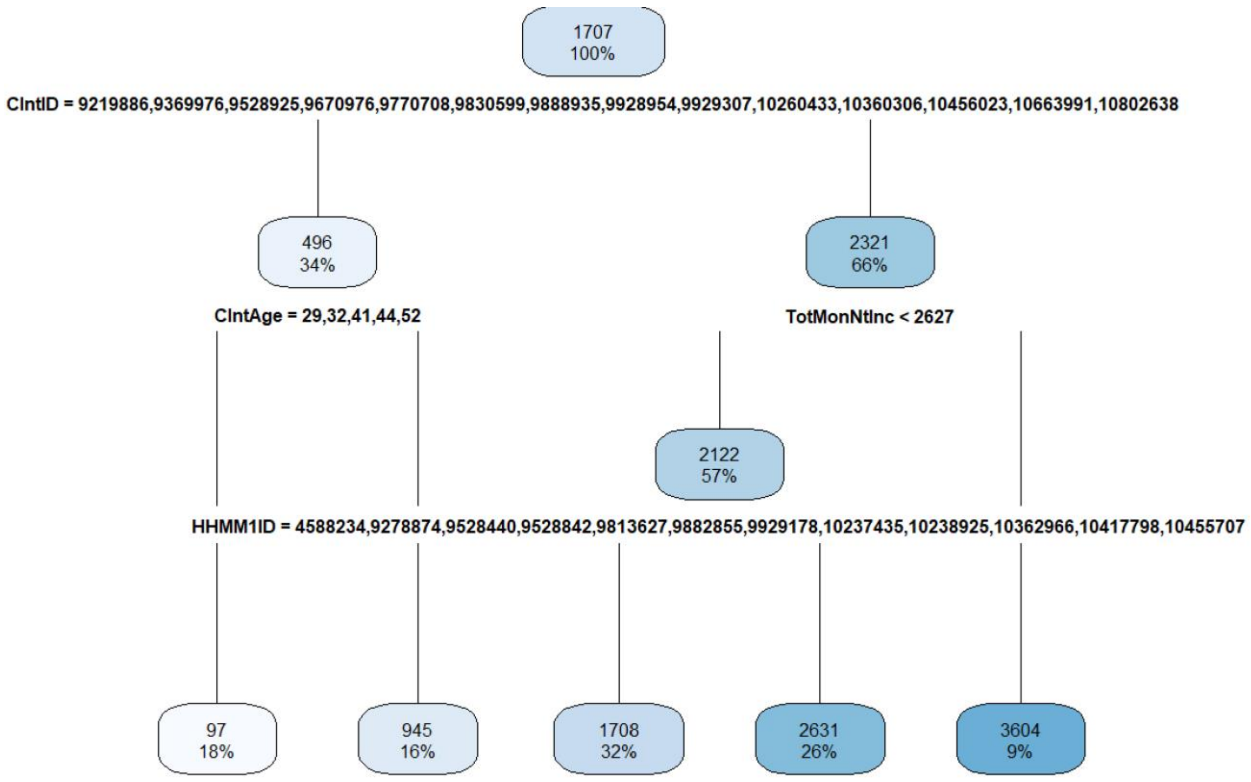


```
> print(cs_mdl_cart_full)
n= 101

node), split, n, deviance, yval
      * denotes terminal node

 1) root 101 126419100.0 1706.80900
   2) ClntID=9219886,9369976,9528925,9670976,9770708,9830599,9888935,9928954,9929307,1
0260433,10360306,10456023,10663991,10802638 34   7883529.0  496.29410
     4) ClntAge=29,32,41,44,52 18    342361.1   97.22222 *
     5) ClntAge=34,43,45,57,59,65 16   1449535.0  945.25000 *
   3) ClntID=9194925,9278821,9279341,9369237,9445162,9528410,9528575,9548852,9623330,9
813593,9875184,9882821,9928956,9987172,10165425,10238777,10353058,10402869,10417842,10
455646,10591060,10714125,10714481,10783423 67  43431190.0 2321.10000
     6) TotMonNtInc< 2626.5 58  16030630.0 2121.99400
      12) HHM1ID=4588234,9278874,9528440,9528842,9813627,9882855,9929178,10237435,102
38925,10362966,10417798,10455707 32   1533488.0 1708.12500 *
      13) HHM1ID=2334784,9369383,9875235,9928961,9943116,9987183,10165436,10402956,10
591104,10714214,10714538,10783555 26   2269819.0 2631.37200 *
     7) TotMonNtInc>=2626.5 9  10283640.0 3604.22200 *
```

The following function will plot the tree for a better visualization of the results.

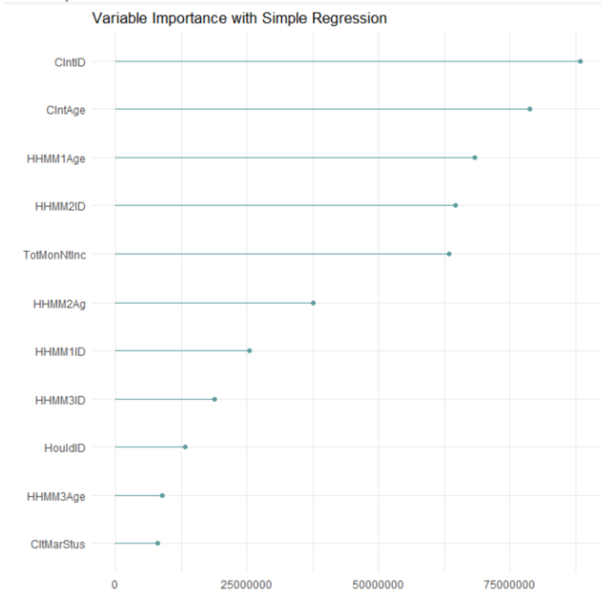


```
rpart.plot(cs_mdl_cart_full, yesno = TRUE)
```


1707
100%
ClntID = 9219886,9369976,9528925,9670976,9770708,9830599,9888935,9928954,9929307,10260433,10360306,10456023,10663991,10802638
496
34%
2321
66%
ClntAge = 29,32,41,44,52
TotMonNtInc < 2627
2122
57%
HHMM1ID = 4588234,9278874,9528440,9528842,9813627,9882855,9929178,10237435,10238925,10362966,10417798,10455707
97
18%
945
16%
1708
32%
2631
26%
3604
9%

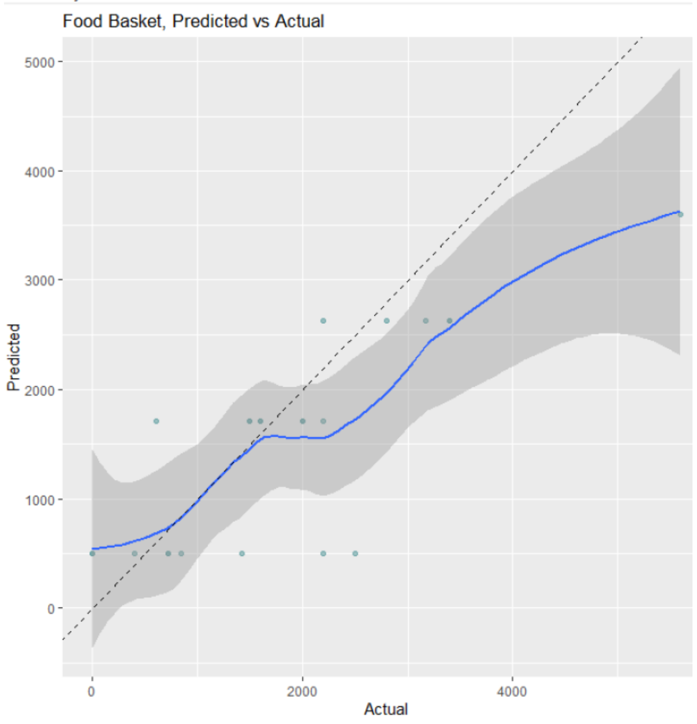
The following function will plot the most important variables:

```
cs_mdl_cart$variable.importance %>%
  data.frame() %>%
  rownames_to_column(var = "Feature") %>%
  rename(Overall = '.') %>%
  ggplot(aes(x = fct_reorder(Feature, Overall), y = Overall)) +
  geom_pointrange(aes(ymin = 0, ymax = Overall), color = "cadetblue", size = .3) +
  theme_minimal() +
  coord_flip() +
  labs(x = "", y = "", title = "Variable Importance with Simple Regression")

cs_preds_cart <- predict(cs_mdl_cart, cs_test, type = "vector")
```



There are less than 2,000 possible predicted values binning the observations.

```
data.frame(Predicted = cs_preds_cart, Actual = cs_test$MothHoldInc) %>%
  ggplot(aes(x = Actual, y = Predicted)) +
  geom_point(alpha = 0.6, color = "cadetblue") +
  geom_smooth() +
  geom_abline(intercept = 0, slope = 1, linetype = 2) +
  labs(title = "Food Basket, Predicted vs Actual")
```



The following functions helps us predict the accuracy of our model:

```
p <- predict(cs_mdl_cart_full, cs_train)
```

```
sqrt(mean(cs_train$`MothHoldInc`-p)^2)
```

```
#sually, the larger the R2, the better the regression model fits
#your observations, r square, 0.826503
(cor(cs_train$`MothHoldInc`, p))^2
```

Our model turns out to have an 83% of accuracy.