Capstone Project - Cross sell and Up sell of Product

Rahul Saha - EISIN171819

31 December 2018

We are going to explore the data that are generated through internal and external source for the project. Building such a pipeline is an important and critical process to perform Analytics. Here, we shall use Unsupervised Learning Technique - K Means primary to perform some supervised clusters. We are going to build multiple cluster and check for the product that are currently consumed and based on that we shall recommend other products to the customer.

Mock Data Preparation

```
library(readxl)
```

```
#Input Data from excel with lookup table
lookupTable <- read_excel("C:\\Users\\rahul\\Downloads\\Codeathon Next OneDrive_1_08-12-2018\\LookupTable.xlsx")
```

```
#Locally assign lookup values
Gender_ls <- lookupTable$Gender
Marital_Status_ls <- lookupTable$'Marital Status'
Locality_Type_ls <- lookupTable$`Locality Type`
State_ls = lookupTable$State
Education_ls = lookupTable$Education
Profession_ls = lookupTable$Profession
Organisation_ls = lookupTable$Organisation
Employment_Status_ls = lookupTable$'Employment Status'
Smoking_ls = lookupTable$Smoking
Health_Condition_ls = lookupTable$'Health Condition'
Purchase_Channel_ls = lookupTable$'Purchase Channel'
```

```
#Removing NA values
Gender_ls = Gender_ls[is.na(Gender_ls)==FALSE]
Marital_Status_ls = Marital_Status_ls[is.na(Marital_Status_ls)==FALSE]
Locality_Type_ls = Locality_Type_ls[is.na(Locality_Type_ls)==FALSE]
State_ls = State_ls[is.na(State_ls)==FALSE]
Education_ls = Education_ls[is.na(Education_ls)==FALSE]
Profession_ls = Profession_ls[is.na(Profession_ls)==FALSE]
Organisation_ls = Organisation_ls[is.na(Organisation_ls)==FALSE]
Employment_Status_ls = Employment_Status_ls[is.na(Employment_Status_ls)==FALSE]
Smoking_ls = Smoking_ls[is.na(Smoking_ls)==FALSE]
Health_Condition_ls = Health_Condition_ls[is.na(Health_Condition_ls)==FALSE]
Purchase_Channel_ls = Purchase_Channel_ls[is.na(Purchase_Channel_ls)==FALSE]
```

```
#Package to generate positive random normals
library(truncnorm)
```

```
## Warning: package 'truncnorm' was built under R version 3.5.1
```

```
df <- data.frame(ds.Gender_ls = sample(Gender_ls,5000,replace = T),
                 ds.Marital_Status_ls = sample(Marital_Status_ls,5000,replace = T,prob = c(0.7,0.2,0.1)),
                 ds.Locality_Type_ls = sample(Locality_Type_ls,5000,replace = T),
                 ds.State_ls = sample(State_ls,5000,replace = T),
                 ds.Education_ls = sample(Education_ls,5000,replace = T,prob = c(0.3,0.4,0.2,0.1)),
                 ds.Primary_Purchase_Channel_ls = sample(Purchase_Channel_ls,5000,replace = T,prob = c(0.3,0.3,0.2,0.2)))
df =  cbind(df,
            ds.Organisation_ls = sample(Organisation_ls,5000,replace = T,prob = c(0.2,0.14,0.15,0.12,0.15,0.01,0.08,0.1,0.05
)),
            ds.Employment_Status_ls = sample(Employment_Status_ls,5000,replace = T, prob = c(0.05,0.6,0.15,0.2)),
            ds.Health_Condition_ls = sample(Health_Condition_ls,5000,replace = T, prob = c(0.25,0.3,0.45)),
            ds.Secondary_Purchase_Channel_ls = sample(Purchase_Channel_ls,5000,replace = T, prob = c(0.3,0.3,0.2,0.2)))


df$ds.Profession_ls = ifelse(df$ds.Employment_Status_ls == "Retired","Retired","NA")


df$ds.Profession_ls[df$ds.Profession_ls=="NA"] = sample(Profession_ls,4061,replace = T,prob = c(0.01,0.1,0.05,0.09,0.15,0.1,
0.05,0.05,0.05,0.08,0.02,0.00,0.1,0.15))
```

```
## Warning in df$ds.Profession_ls[df$ds.Profession_ls == "NA"] =
## sample(Profession_ls, : number of items to replace is not a multiple of
## replacement length
```

```
df$ds.age = ifelse(df$ds.Employment_Status_ls == "Retired",round(rtruncnorm(939,a=40, b=85, mean=55, sd=5),0),
                   ifelse(df$ds.Profession_ls == "Student",round(rtruncnorm(38 ,a=20, b=45, mean=28, sd=5),0),
                          round(rtruncnorm(4023 ,a=23, b=70, mean=45, sd=5),0)))

df$ds.Parents = ifelse(df$ds.Employment_Status_ls == "Retired",round(rtruncnorm(939,a=0, b=1, mean=0, sd=1),0),
                       ifelse(df$ds.Profession_ls == "Student",round(rtruncnorm(38 ,a=1, b=2, mean=2, sd=1),0),
                              round(rtruncnorm(4023 ,a=1, b=2, mean=1.25, sd=1),0)))

df$ds.family.size = ifelse(df$ds.age >= 20 & df$ds.age < 30, sample(c(3,4,5),replace=T,prob = c(0.2,0.6,0.2)),
                           ifelse(df$ds.age >= 30 & df$ds.age < 40, sample(c(4,5,6),replace=T,prob = c(0.1,0.7,0.2)),
                                  sample(c(3,4,5),replace=T,prob = c(0.1,0.6,0.3))))

df$ds.Children <- ifelse(df$ds.age >= 20 & df$ds.age < 30, sample(c(0,1,2),replace=T,prob = c(0.2,0.45,0.35)),
                         ifelse(df$ds.age >= 30 & df$ds.age < 40, sample(c(0,1,2),replace=T,prob = c(0.1,0.6,0.3)),
                                sample(c(2,3,4),replace=T,prob = c(0.5,0.35,0.15))))

df$ds.Avg.Family.Age <- ifelse(df$ds.age >= 20 & df$ds.age < 30 & df$ds.Children > 2,rtruncnorm(5000,a=30, b=40, mean=35,sd=
3),ifelse(df$ds.age >= 30 & df$ds.age < 45 & df$ds.Children > 1, rtruncnorm(5000,a=30, b=50, mean=40, sd=3),
                         rtruncnorm(5000,a=40, b=60, mean=50, sd=3)))

df$ds.Annual.income = round(rtruncnorm(n=5000, a=20000, b=185000, mean=50000, sd=10000),0)
df$ds.Avg.annual.inc = round(rtruncnorm(n=5000, a=20000, b=185000, mean=50000, sd=10000),0)
df$ds.Annual.Expenses  = df$ds.Annual.income - round(rtruncnorm(n=5000, a=10000, b=105000, mean=35000, sd=7000),0)

df$ds.Saving.Amount = df$ds.Annual.income - df$ds.Annual.Expenses
df$ds.Credit.Cards = round(rtruncnorm(n=5000, a=0, b=15, mean=4, sd=1),0)
df$ds.Two.Wheelers = round(rtruncnorm(n=5000, a=0, b=4, mean=2, sd=1),0)
df$ds.Four.Wheelers = round(rtruncnorm(n=5000, a=0, b=2, mean=0, sd=0.5),0)
df$ds.Bank.Accounts = round(rtruncnorm(n=5000, a=1, b=6, mean=3, sd=1),0)
df$ds.Houses = round(rtruncnorm(n=5000, a=1, b=4, mean=1, sd=1),0)
df$ds.Estates  = round(rtruncnorm(n=5000, a=1, b=3, mean=1, sd=0.5),0)

df$ds.Yearly.Travel.Dist.Air = ifelse(round(rtruncnorm(n=5000, a=5000, b=50000, mean=15000, sd=1000),0)<0,round(rtruncnorm(n
=5000, a=5000, b=50000, mean=15000, sd=1000),0)*-1,round(rtruncnorm(n=5000, a=5000, b=50000, mean=15000, sd=1000),0))

df$ds.Yearly.Travel.Dist.Road = ifelse(round(rtruncnorm(n=5000, a=20000, b=80000, mean=50000, sd=10000),0)<0,round(rtruncnor
m(n=5000, a=20000, b=80000, mean=50000, sd=10000),0)*-1,round(rtruncnorm(n=5000, a=20000, b=80000, mean=50000, sd=10000),0))

df$ds.Policy.Face.Value = round(rtruncnorm(n=5000, a=200000, b=2550000, mean=100000, sd=20000),0)

df$ds.Claimed.Amount = df$ds.Policy.Face.Value*sample(c(0,0.2,0.5,0.8,1),replace=T,prob = c(0.55,0.15,0.15,0.1,0.05))

df$ds.Yearly.Premium = df$ds.Annual.Expenses * round(rtruncnorm(n=5000, a=5, b=20, mean=10, sd=2),0)/100

df$ds.YearOfPurchase.1st_Prod = sample(c(2000:2009), 5000, replace = TRUE)

df$ds.YearOfPurchase.LastProd = df$ds.YearOfPurchase.1st_Prod + sample(0:8, 5000, replace = TRUE)

#Children Plan
df$ds.prod1 = ifelse(df$ds.Children == 0, sample(0:1,replace=T,prob = c(0.9,0.1)),sample(0:1,replace=T,prob = c(0.1,0.9)))

#Senior Citizen Plan
df$ds.prod2 = ifelse(df$ds.age < 60, sample(0:1,replace=T,prob = c(0.9,0.1)),sample(0:1,replace=T,prob = c(0.2,0.8)))

#Car Insurance Plan
df$ds.prod3 = ifelse(df$ds.Four.Wheelers > 0, sample(0:1,replace=T,prob = c(0,0.9)),sample(0:1,replace=T,prob = c(1,0)))

#Bike Insurance Plan
df$ds.prod4 = ifelse(df$ds.Two.Wheelers > 0, sample(0:1,replace=T,prob = c(0,0.9)),sample(0:1,replace=T,prob = c(1,0)))

#Professional Insurance Plan
df$ds.prod5 = ifelse(df$ds.Profession_ls == "Student", sample(0:1,replace=T,prob = c(1,0)),sample(0:1,replace=T,prob = c(0.1
5,0.85)))

#Farmer Insurance Plan
df$ds.prod6 = ifelse(df$ds.Profession_ls == "Farmer", sample(0:1,replace=T,prob = c(1,0)),sample(0:1,replace=T,prob = c(0.15
,0.85)))

#High Income Insurance Plan
df$ds.prod7 = ifelse(df$ds.Annual.income <= 50000, sample(0:1,replace=T,prob = c(0.8,0.2)),sample(0:1,replace=T,prob = c(0.2
,0.8)))

#Estate Insurance Plan
df$ds.prod8 = ifelse(df$ds.Estates < 2, sample(0:1,replace=T,prob = c(0.95,0.15)),sample(0:1,replace=T,prob = c(0.15,0.95)))

#Health Insurance Plan
df$ds.prod9 = ifelse(df$ds.Health_Condition_ls == "Below Avg", sample(0:1,replace=T,prob = c(0.2,0.8)),sample(0:1,replace=T,
prob = c(0.1,0.9)))

#Retirement Plan
df$ds.prod10 = sample(0:1,5000,replace = TRUE,prob = c(0.10,0.90))
```

```
#write.csv(df, file = "C:\\Users\\rahul\\Downloads\\Codeathon Next OneDrive_1_08-12-2018\\codeathon_input1.csv")
```

Drawing Analytics

Step 1 - File/Data Input:

1 a. Reading data from csv file:

```
inputdf <- read.csv("C:\\Users\\rahul\\Downloads\\Codeathon Next OneDrive_1_08-12-2018\\codeathon_input1.csv")
str(inputdf)
```

```
## 'data.frame':    5000 obs. of  44 variables:
##  $ X                               : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ ds.Gender_ls                    : Factor w/ 2 levels "Female","Male": 1 1 2 2 2 1 1 1 1 1 ...
##  $ ds.Marital_Status_ls            : Factor w/ 3 levels "Divorced","Married",..: 3 2 2 2 2 2 3 3 2 3 ...
##  $ ds.Locality_Type_ls             : Factor w/ 3 levels "City","Town",..: 1 1 1 3 2 1 1 1 3 2 ...
##  $ ds.State_ls                     : Factor w/ 51 levels "Alabama","Alaska",..: 44 25 3 44 9 7 49 20 20 33 ...
##  $ ds.Education_ls                 : Factor w/ 4 levels "Graduate","Masters",..: 2 4 1 1 4 1 1 4 3 2 ...
##  $ ds.Primary_Purchase_Channel_ls  : Factor w/ 4 levels "Broker","Direct",..: 2 2 4 2 1 3 4 3 4 3 ...
##  $ ds.Organisation_ls              : Factor w/ 9 levels "Administration",..: 7 5 2 5 7 2 1 5 1 6 ...
##  $ ds.Employment_Status_ls         : Factor w/ 4 levels "Active","Retired",..: 2 1 1 1 3 3 1 3 1 1 ...
##  $ ds.Health_Condition_ls          : Factor w/ 3 levels "Above Avg","Average",..: 2 3 3 2 2 2 2 2 3 3 ...
##  $ ds.Secondary_Purchase_Channel_ls: Factor w/ 4 levels "Broker","Direct",..: 2 2 1 3 2 1 2 3 1 4 ...
##  $ ds.Profession_ls                : Factor w/ 14 levels "Admin - operation",..: 11 9 5 9 9 4 7 1 3 1 ...
##  $ ds.age                          : int  50 54 49 50 45 44 49 43 54 38 ...
##  $ ds.Parents                      : int  1 1 2 2 2 1 2 1 2 1 ...
##  $ ds.family.size                  : int  5 4 4 5 4 4 5 4 4 5 ...
##  $ ds.Children                     : int  2 2 3 2 2 3 2 2 3 2 ...
##  $ ds.Avg.Family.Age               : num  48.5 54.1 51.6 43.3 57.3 ...
##  $ ds.Annual.income                : int  42399 53036 56773 52577 37594 53683 62026 61673 44978 50466 ...
##  $ ds.Avg.annual.inc               : int  29845 36072 35141 43767 64364 41437 45846 42865 63552 37065 ...
##  $ ds.Annual.Expenses              : int  11527 17676 30734 18922 1847 21355 36382 25829 9982 18160 ...
##  $ ds.Saving.Amount                : int  30872 35360 26039 33655 35747 32328 25644 35844 34996 32306 ...
##  $ ds.Credit.Cards                 : int  4 3 4 4 5 5 5 4 5 6 ...
##  $ ds.Two.Wheelers                 : int  2 2 0 2 1 2 1 1 1 3 ...
##  $ ds.Four.Wheelers                : int  0 1 0 1 0 0 0 0 0 0 ...
##  $ ds.Bank.Accounts                : int  3 2 3 3 3 3 2 4 3 3 ...
##  $ ds.Houses                       : int  1 3 1 2 2 1 1 2 1 2 ...
##  $ ds.Estates                      : int  1 1 1 1 1 2 1 1 1 2 ...
##  $ ds.Yearly.Travel.Dist.Air       : int  14467 13923 15162 12581 13417 15695 16466 16818 14074 13909 ...
##  $ ds.Yearly.Travel.Dist.Road      : int  32589 30853 55229 72063 47029 50101 47385 63616 60448 59141 ...
##  $ ds.Policy.Face.Value            : int  200308 201418 209434 211959 202983 200361 201920 202095 200591 203923 ...
##  $ ds.Claimed.Amount               : num  0 161134 41887 105980 0 ...
##  $ ds.Yearly.Premium               : num  1383 1944 2459 1892 203 ...
##  $ ds.YearOfPurchase.1st_Prod      : int  2008 2000 2004 2005 2006 2004 2009 2003 2002 2003 ...
##  $ ds.YearOfPurchase.LastProd      : int  2013 2005 2011 2007 2006 2009 2012 2004 2005 2003 ...
##  $ ds.prod1                        : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ ds.prod2                        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ds.prod3                        : int  0 1 0 1 0 0 0 0 0 0 ...
##  $ ds.prod4                        : int  1 1 0 1 1 1 1 1 1 1 ...
##  $ ds.prod5                        : int  0 1 0 1 0 1 0 1 0 1 ...
##  $ ds.prod6                        : int  0 1 0 1 0 1 0 1 0 1 ...
##  $ ds.prod7                        : int  0 1 1 1 0 1 1 1 0 1 ...
##  $ ds.prod8                        : int  0 0 0 0 0 1 0 0 0 1 ...
##  $ ds.prod9                        : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ ds.prod10                       : int  1 1 1 1 1 1 1 1 1 1 ...
```

```
# We shall ignore repetative column for Purchase_Channel
```

Summary:

Data comprises of both categorical, discrite and numeric type

```
summary(inputdf)
```

```
##        X        ds.Gender_ls  ds.Marital_Status_ls ds.Locality_Type_ls
##  Min.   :   1   Female:2550   Divorced: 524        City   :1703
##  1st Qu.:1251   Male  :2450   Married :3546        Town   :1641
##  Median :2500                 Single  : 930        Village:1656
##  Mean   :2500
##  3rd Qu.:3750
##  Max.   :5000
##
##          ds.State_ls         ds.Education_ls
##  Massachusetts : 119   Graduate      :1970
##  North Carolina: 118   Masters       :1012
##  Illinois      : 117   PhD           : 488
##  Maryland      : 117   Under Graduate:1530
##  North Dakota  : 117
##  South Carolina: 116
##  (Other)       :4296
##    ds.Primary_Purchase_Channel_ls       ds.Organisation_ls
##  Broker           : 965           IT             :972
##  Direct           :1494           Administration :768
##  Independent Agent:1549           Retails        :749
##  TPA              : 992           Metal & Mineral:669
##                                   ITES-BPO       :642
##                                   Agriculture    :517
##                                   (Other)        :683
##  ds.Employment_Status_ls ds.Health_Condition_ls
##  Active    :3034         Above Avg:2235
##  Retired   :1017         Average  :1525
##  Terminated: 726         Below Avg:1240
##  Unemployed: 223
##
##
##
##   ds.Secondary_Purchase_Channel_ls          ds.Profession_ls
##  Broker           : 983            Retired          :1017
##  Direct           :1498            Industry Worker  : 601
##  Independent Agent:1533            Engineer         : 575
##  TPA              : 986            Teacher          : 404
##                                    Driver           : 395
##                                    Admin - operation: 392
##                                    (Other)          :1616
##      ds.age        ds.Parents    ds.family.size   ds.Children
##  Min.   :21.00   Min.   :0.000   Min.   :4.000   Min.   :1.000
##  1st Qu.:43.00   1st Qu.:1.000   1st Qu.:4.000   1st Qu.:2.000
##  Median :47.00   Median :1.000   Median :4.000   Median :2.000
##  Mean   :46.97   Mean   :1.275   Mean   :4.365   Mean   :2.219
##  3rd Qu.:51.00   3rd Qu.:2.000   3rd Qu.:5.000   3rd Qu.:3.000
##  Max.   :67.00   Max.   :2.000   Max.   :5.000   Max.   :3.000
##
##  ds.Avg.Family.Age ds.Annual.income ds.Avg.annual.inc ds.Annual.Expenses
##  Min.   :30.37     Min.   :20938    Min.   :21305     Min.   :-23221
##  1st Qu.:42.92     1st Qu.:43480    1st Qu.:43249     1st Qu.:  7106
##  Median :48.33     Median :50294    Median :50193     Median : 15264
##  Mean   :47.07     Mean   :50317    Mean   :50104     Mean   : 15249
##  3rd Qu.:51.10     3rd Qu.:56963    3rd Qu.:56785     3rd Qu.: 23443
##  Max.   :59.20     Max.   :86313    Max.   :83819     Max.   : 59834
##
##  ds.Saving.Amount ds.Credit.Cards ds.Two.Wheelers ds.Four.Wheelers
##  Min.   :12072    Min.   :0.000   Min.   :0.000   Min.   :0.0000
##  1st Qu.:30384    1st Qu.:3.000   1st Qu.:1.000   1st Qu.:0.0000
##  Median :35070    Median :4.000   Median :2.000   Median :0.0000
##  Mean   :35069    Mean   :4.013   Mean   :1.985   Mean   :0.3212
##  3rd Qu.:39863    3rd Qu.:5.000   3rd Qu.:3.000   3rd Qu.:1.0000
##  Max.   :63641    Max.   :7.000   Max.   :4.000   Max.   :2.0000
##
##  ds.Bank.Accounts   ds.Houses       ds.Estates
##  Min.   :1.000    Min.   :1.000   Min.   :1.000
##  1st Qu.:2.000    1st Qu.:1.000   1st Qu.:1.000
##  Median :3.000    Median :2.000   Median :1.000
##  Mean   :3.035    Mean   :1.761   Mean   :1.305
##  3rd Qu.:4.000    3rd Qu.:2.000   3rd Qu.:2.000
##  Max.   :6.000    Max.   :4.000   Max.   :3.000
##
##  ds.Yearly.Travel.Dist.Air ds.Yearly.Travel.Dist.Road ds.Policy.Face.Value
##  Min.   :10611             Min.   :20638              Min.   :200001
##  1st Qu.:14319             1st Qu.:43064              1st Qu.:201105
##  Median :14999             Median :49705              Median :202646
##  Mean   :14996             Mean   :49782              Mean   :203752
##  3rd Qu.:15658             3rd Qu.:56513              3rd Qu.:205296
##  Max.   :18828             Max.   :79802              Max.   :238980
##
##  ds.Claimed.Amount ds.Yearly.Premium ds.YearOfPurchase.1st_Prod
##  Min.   :     0    Min.   :-2501.6   Min.   :2000
```

```
##  1st Qu.:     0    1st Qu.:  672.7   1st Qu.:2002
##  Median : 40547    Median : 1463.2   Median :2004
##  Mean   : 61136    Mean   : 1531.7   Mean   :2005
##  3rd Qu.:102709    3rd Qu.: 2318.3   3rd Qu.:2007
##  Max.   :184594    Max.   : 7059.8   Max.   :2009
##
##  ds.YearOfPurchase.LastProd    ds.prod1    ds.prod2         ds.prod3
##  Min.   :2000               Min.   :1   Min.   :0.0000   Min.   :0.000
##  1st Qu.:2006               1st Qu.:1   1st Qu.:0.0000   1st Qu.:0.000
##  Median :2008               Median :1   Median :0.0000   Median :0.000
##  Mean   :2008               Mean   :1   Mean   :0.0188   Mean   :0.319
##  3rd Qu.:2011               3rd Qu.:1   3rd Qu.:0.0000   3rd Qu.:1.000
##  Max.   :2017               Max.   :1   Max.   :1.0000   Max.   :1.000
##
##     ds.prod4         ds.prod5         ds.prod6         ds.prod7
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00
##  1st Qu.:1.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00
##  Median :1.0000   Median :0.0000   Median :0.0000   Median :1.00
##  Mean   :0.9574   Mean   :0.4954   Mean   :0.4798   Mean   :0.51
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.00
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00
##
##     ds.prod8        ds.prod9   ds.prod10
##  Min.   :0.000   Min.   :1   Min.   :0.0000
##  1st Qu.:0.000   1st Qu.:1   1st Qu.:1.0000
##  Median :0.000   Median :1   Median :1.0000
##  Mean   :0.158   Mean   :1   Mean   :0.9042
##  3rd Qu.:0.000   3rd Qu.:1   3rd Qu.:1.0000
##  Max.   :1.000   Max.   :1   Max.   :1.0000
##
```

Step 2 - Exploratory Analysis:

2 a) Loading library

```
library(ggplot2) #For Graphical Visualization
```

```
## Warning: package 'ggplot2' was built under R version 3.5.1
```

```
library(forcats) #For ordering bar cart, based on the frequency distribution
```

2 b) Initial Visualization

Gender:

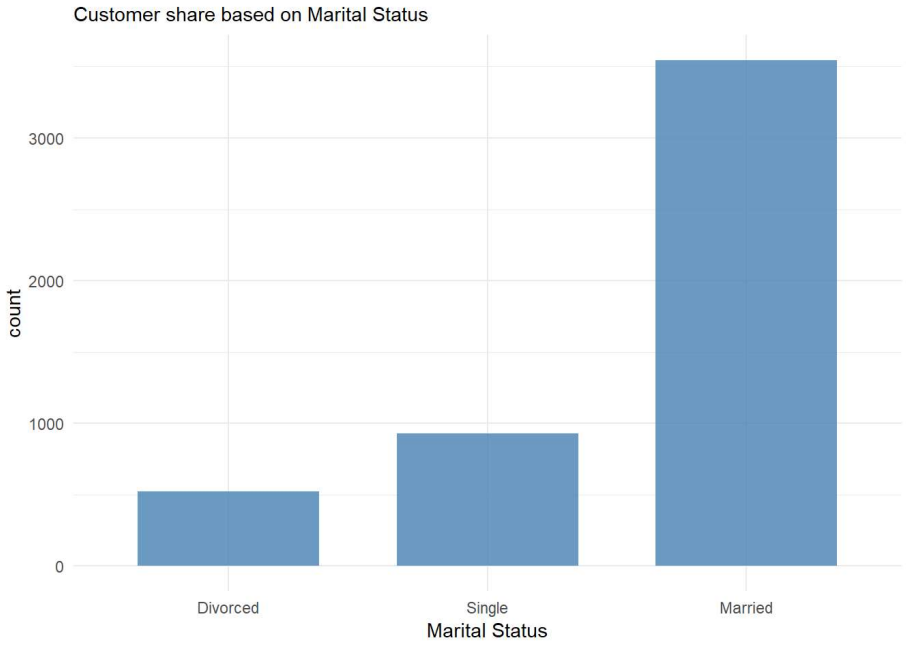
Customers are almost equally divided between Gender.

```
a <- ggplot(inputdf, aes(x = ds.Gender_ls))
a +  geom_bar(aes(fill = ds.Gender_ls), width=0.7, alpha=0.8) + labs(x = "Gender") +
  theme_minimal()
```

Marital_Status:

As per normal customer behaviour, married are more compared to single and divorce. Since, here the product is Insurance Product and majority customer are married, and customer are more careful to secure family's economy condition.

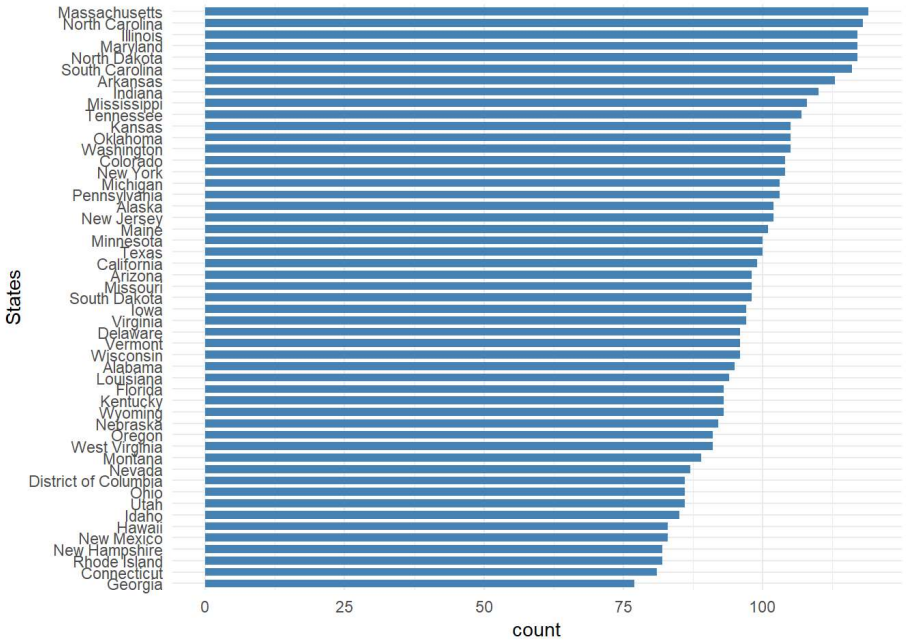
```
a <- ggplot(inputdf, aes(x = fct_rev(fct_infreq(ds.Marital_Status_ls))))
a +  geom_bar(fill="steelblue", width=0.7, alpha=0.8)+ labs(x = "Marital Status", subtitle = "Customer share based on Marita
l Status") +
  theme_minimal()
```



States:

Customers are present across all the US states. Some states are having more customers compare to other states. Massachusetts is having maximum customer, does this city population is more compare to other? Does the channel partners are more active? We can answer more such insight. But here we are focusing on our scope to recommend some product to certain group.

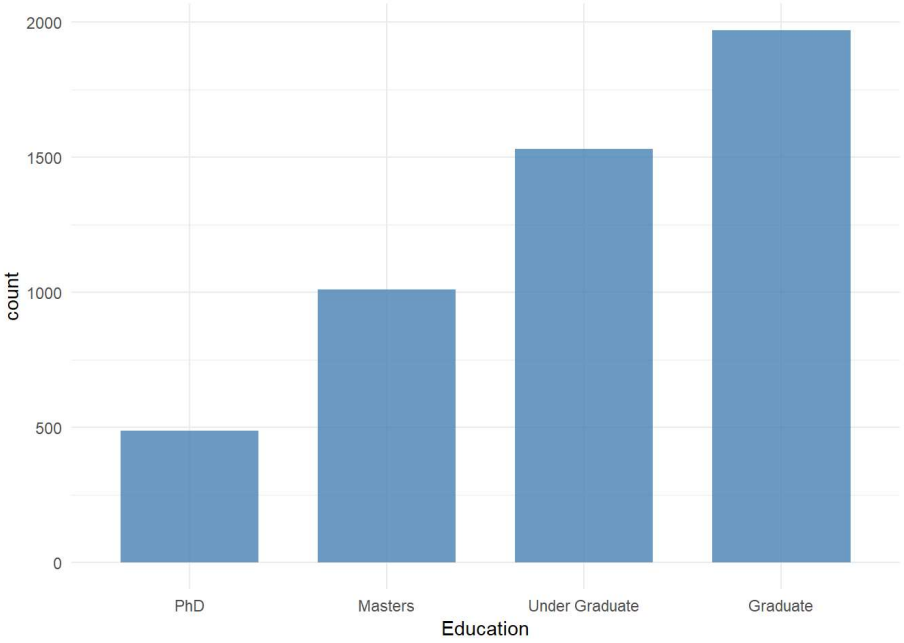
```
a <- ggplot(inputdf, aes(x = fct_rev(fct_infreq(ds.State_ls))))
a +  geom_bar(fill="steelblue",width=0.7) + labs(x = "States") + coord_flip() +
  theme_minimal() + theme(legend.position="none")
```



Education:

It shows general trend that limited people go for higher eduction. So do customer based for an Insurance Product.

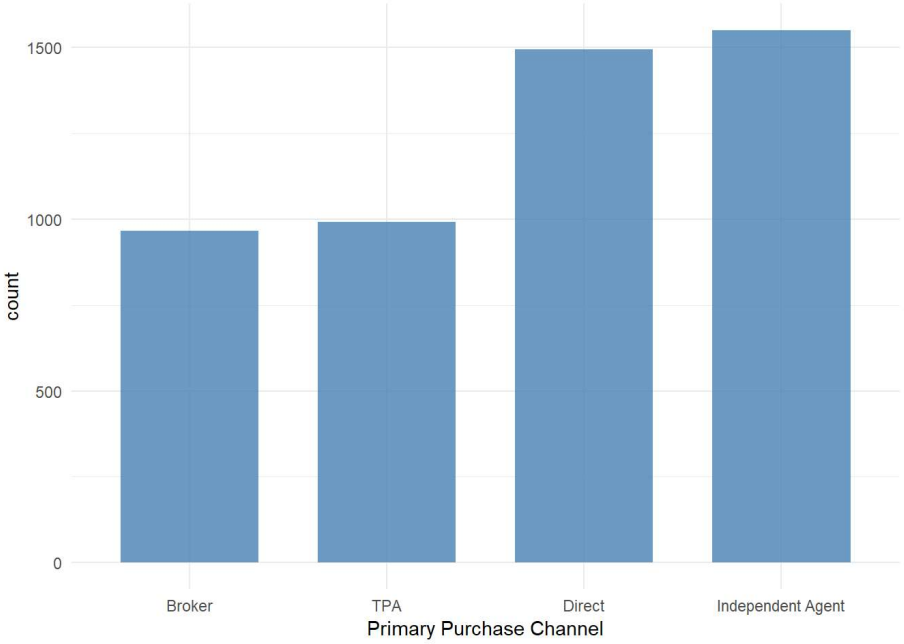
```
a <- ggplot(inputdf, aes(x = fct_rev(fct_infreq(ds.Education_ls))))
a +  geom_bar(fill="steelblue", alpha=0.8,width=0.7) + labs(x = "Education") +
  theme_minimal()
```



Primary Purchase Channel:

Show which particular purchase channel is more active and customer are preferring.

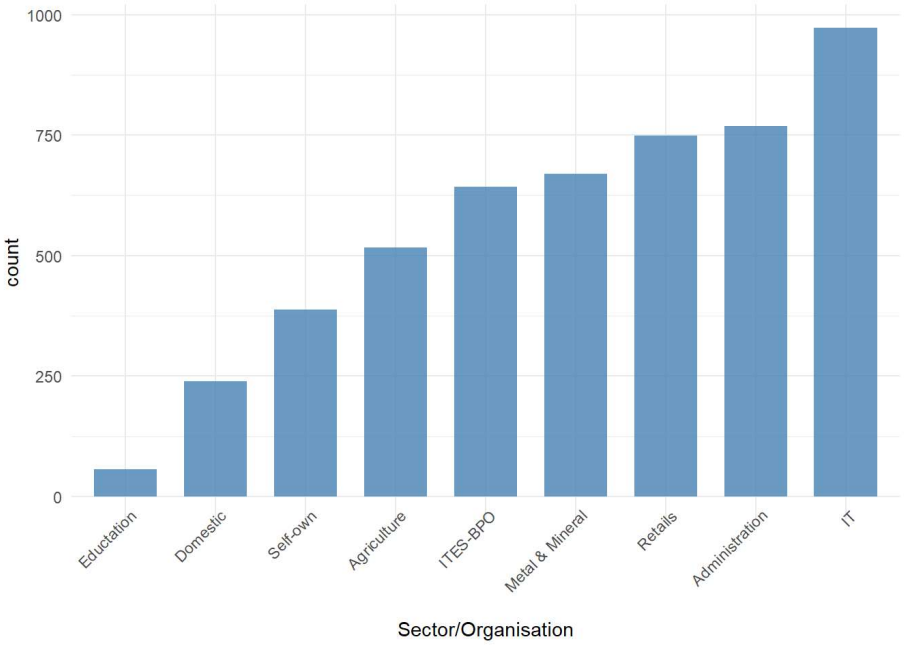
```
a <- ggplot(inputdf, aes(x = fct_rev(fct_infreq(ds.Primary_Purchase_Channel_ls))))
a +  geom_bar(fill="steelblue", alpha=0.8,width=0.7) + labs(x = "Primary Purchase Channel") +
  theme_minimal()
```



Organisation:

Show which selector is having more customer share.

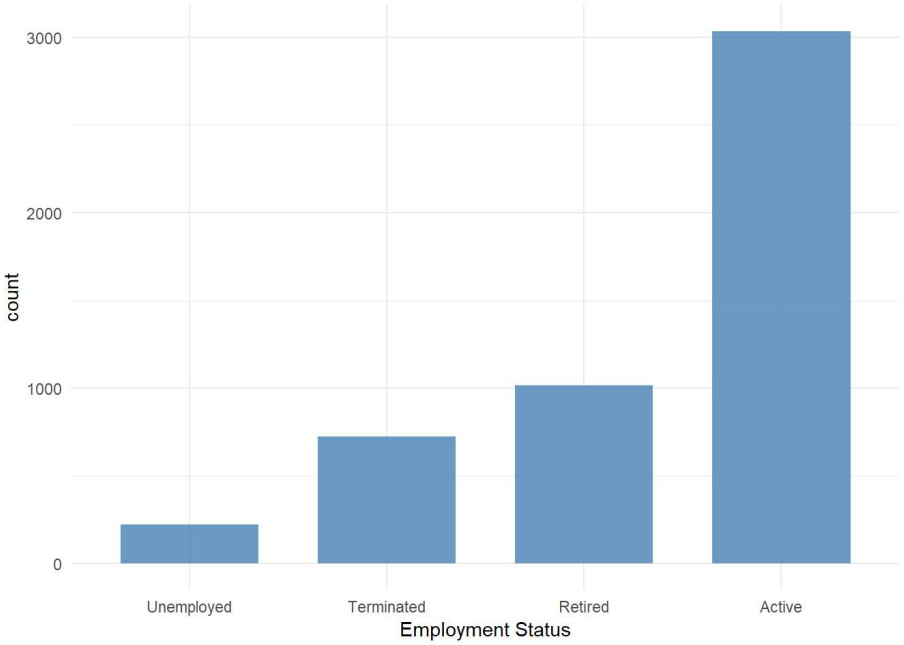
```
library(gridExtra)
rotatedAxisElementText = function(angle,position='x'){
  angle     = angle[1];
  position  = position[1]
  positions = list(x=0,y=90,top=180,right=270)
  if(!position %in% names(positions))
    stop(sprintf("'position' must be one of [%s]",paste(names(positions),collapse=", ")),call.=FALSE)
  if(!is.numeric(angle))
    stop("'angle' must be numeric",call.=FALSE)
  rads  = (angle - positions[[ position ]])*pi/180
  hjust = 0.3*(4 - sin(rads))
  vjust = 0.3*(3.5 + cos(rads))
  element_text(angle=angle,vjust=vjust,hjust=hjust)
}
a <- ggplot(inputdf, aes(x = fct_rev(fct_infreq(ds.Organisation_ls))))
a +  geom_bar(fill="steelblue", alpha=0.8,width=0.7) + labs(x= "Sector/Organisation") +
  theme_minimal() + theme(axis.text.x = rotatedAxisElementText(45,'x'))
```



Employment Status:

On what stage, do the customer maintain an insurance product. Better to push a product to employee with Active status.

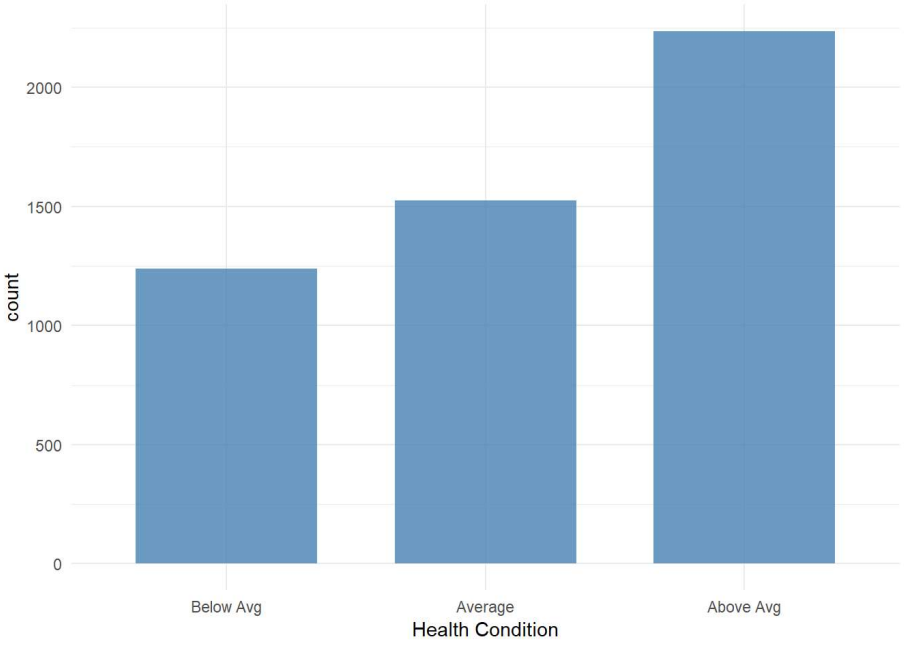
```
a <- ggplot(inputdf, aes(x = fct_rev(fct_infreq(ds.Employment_Status_ls))))
a +  geom_bar(fill="steelblue", alpha=0.8,width=0.7) + labs(x= "Employment Status") +
  theme_minimal()
```



Health Condition:

On what Health condition, do the customer maintain an insurance product. With more Average and Above Average health condition, Insurance company suppose to be in profitable condition.

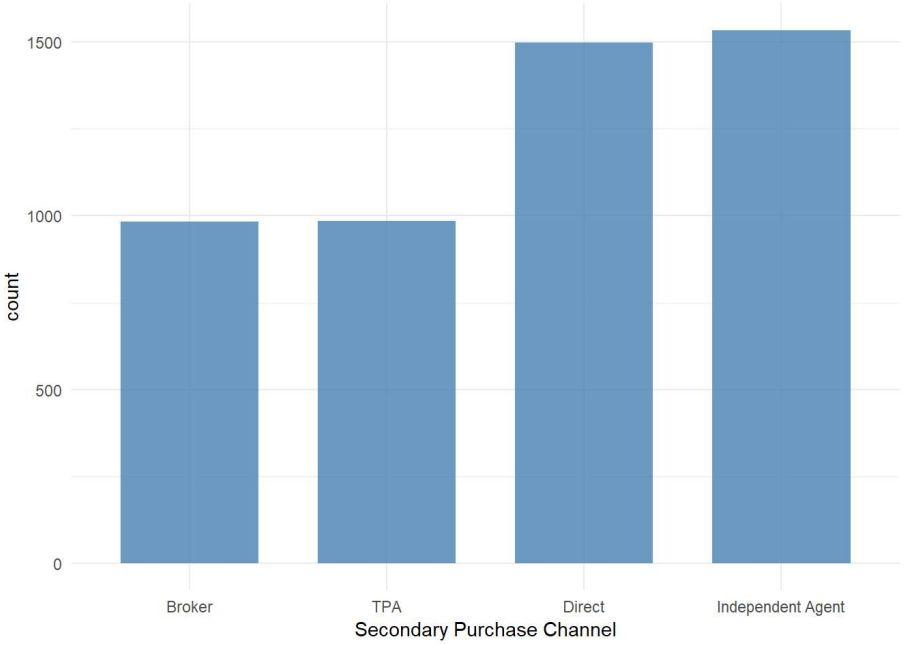
```
a <- ggplot(inputdf, aes(x = fct_rev(fct_infreq(ds.Health_Condition_ls))))
a +  geom_bar(fill="steelblue", alpha=0.8,width=0.7) + labs(x= "Health Condition") +
  theme_minimal()
```



Secondary Purchase Channel:

Show which particular purchase channel is more active and customer are preferring.

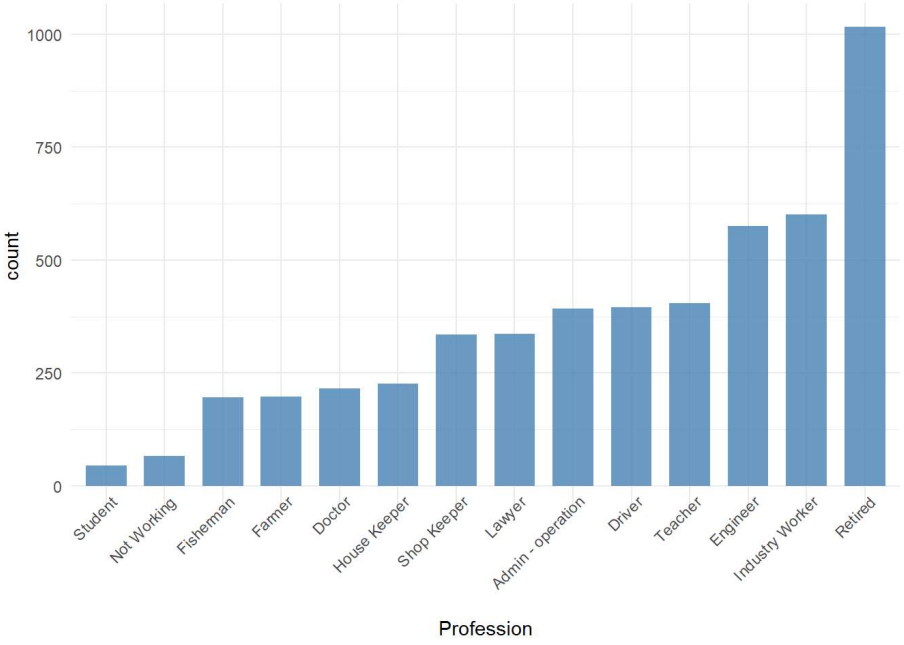
```
a <- ggplot(inputdf, aes(x = fct_rev(fct_infreq(ds.Secondary_Purchase_Channel_ls))))
a +  geom_bar(fill="steelblue", alpha=0.8,width=0.7) + labs(x = "Secondary Purchase Channel") +
  theme_minimal()
```



Profession:

Stats for the company about their customers and their professions.

```
a <- ggplot(inputdf, aes(x = fct_rev(fct_infreq(ds.Profession_ls))))
a +  geom_bar(fill="steelblue", alpha=0.8,width=0.7) + labs(x= "Profession") +
  theme_minimal() + theme(axis.text.x = rotatedAxisElementText(45,'x'))
```




```
# Don't get confused with the Retired bar, prior profession before retirement are not consider.
```

Lets check for the products if having any direct connection with the categorical values.

```
# Loading library tidyr, to convert the wide-table to long-table.
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 3.5.1
```

```
# Only selecting categorical fields and the products
prod.widetable <- inputdf[,c(1:10,12,35:44)]

# Converting to convert the wide-table to long-table
prod.longtable <- gather(prod.widetable, condition, measurement, ds.prod1:ds.prod10, factor_key=TRUE)

str(prod.longtable)
```

```
## 'data.frame':    50000 obs. of  13 variables:
##  $ X                              : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ ds.Gender_ls                   : Factor w/ 2 levels "Female","Male": 1 1 2 2 2 1 1 1 1 1 ...
##  $ ds.Marital_Status_ls           : Factor w/ 3 levels "Divorced","Married",..: 3 2 2 2 2 2 3 3 2 3 ...
##  $ ds.Locality_Type_ls            : Factor w/ 3 levels "City","Town",..: 1 1 1 3 2 1 1 1 3 2 ...
##  $ ds.State_ls                    : Factor w/ 51 levels "Alabama","Alaska",..: 44 25 3 44 9 7 49 20 20 33 ...
##  $ ds.Education_ls                : Factor w/ 4 levels "Graduate","Masters",..: 2 4 1 1 4 1 1 4 3 2 ...
##  $ ds.Primary_Purchase_Channel_ls: Factor w/ 4 levels "Broker","Direct",..: 2 2 4 2 1 3 4 3 4 3 ...
##  $ ds.Organisation_ls             : Factor w/ 9 levels "Administration",..: 7 5 2 5 7 2 1 5 1 6 ...
##  $ ds.Employment_Status_ls        : Factor w/ 4 levels "Active","Retired",..: 2 1 1 1 3 3 1 3 1 1 ...
##  $ ds.Health_Condition_ls         : Factor w/ 3 levels "Above Avg","Average",..: 2 3 3 2 2 2 2 2 3 3 ...
##  $ ds.Profession_ls               : Factor w/ 14 levels "Admin - operation",..: 11 9 5 9 9 4 7 1 3 1 ...
##  $ condition                      : Factor w/ 10 levels "ds.prod1","ds.prod2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ measurement                    : int  1 1 1 1 1 1 1 1 1 1 ...
```

We don’t see much difference in proportion based on the below category

```
prop.table(ftable(prod.longtable[,2],prod.longtable$measurement, prod.longtable$condition))*100
```

```
##           ds.prod1 ds.prod2 ds.prod3 ds.prod4 ds.prod5 ds.prod6 ds.prod7 ds.prod8 ds.prod9 ds.prod10
##
## Female 0     0.000    5.012    3.448    0.208    2.526    2.622    2.500    4.274    0.000     0.468
##        1     5.100    0.088    1.652    4.892    2.574    2.478    2.600    0.826    5.100     4.632
## Male   0     0.000    4.800    3.362    0.218    2.520    2.580    2.400    4.146    0.000     0.490
##        1     4.900    0.100    1.538    4.682    2.380    2.320    2.500    0.754    4.900     4.410
```

We don’t see much difference in proportion based on the below category

```
prop.table(ftable(prod.longtable[,3],prod.longtable$measurement, prod.longtable$condition))*100
```

```
##             ds.prod1 ds.prod2 ds.prod3 ds.prod4 ds.prod5 ds.prod6 ds.prod7 ds.prod8 ds.prod9 ds.prod10
##
## Divorced 0     0.000    1.030    0.682    0.044    0.542    0.552    0.522    0.898    0.000     0.100
##          1     1.048    0.018    0.366    1.004    0.506    0.496    0.526    0.150    1.048     0.948
## Married  0     0.000    6.958    4.854    0.322    3.558    3.680    3.440    5.936    0.000     0.690
##          1     7.092    0.134    2.238    6.770    3.534    3.412    3.652    1.156    7.092     6.402
## Single   0     0.000    1.824    1.274    0.060    0.946    0.970    0.938    1.586    0.000     0.168
##          1     1.860    0.036    0.586    1.800    0.914    0.890    0.922    0.274    1.860     1.692
```

Bayesian network Algorithm to see if Products is having any dependency

```
library(bnlearn)
```

```
## Warning: package 'bnlearn' was built under R version 3.5.1
```

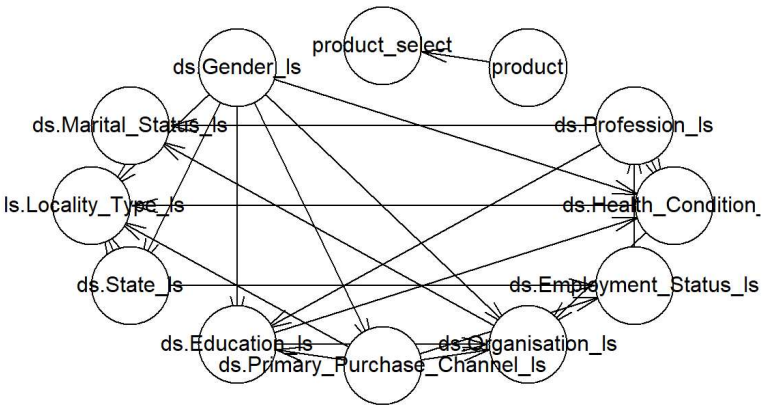
```
##
## Attaching package: 'bnlearn'
```

```
## The following object is masked from 'package:stats':
##
##     sigma
```

Score-based Learning Algorithms

Hill-Climbing (hc): a hill climbing greedy search on the space of the directed graphs. The optimized implementation uses score caching, score decomposability and score equivalence to reduce the number of duplicated tests

```
bn_df <- data.frame(prod.longtable[,2:11], product = prod.longtable[,12],product_select = as.factor(prod.longtable$measureme
nt))
res <- hc(bn_df)
plot(res)
```



2 c) Initial Visualization and Normality test for continous variables

```
library(nortest)
```

Normality test for “ds.age”: Fail Test

- Since p-value is less than 0.05, the distribution is not normal

```
ad.test(inputdf[,13])
```

```
##
##  Anderson-Darling normality test
##
## data:  inputdf[, 13]
## A = 10.873, p-value < 2.2e-16
```

```
cvm.test(inputdf[,13])
```

```
## Warning in cvm.test(inputdf[, 13]): p-value is smaller than 7.37e-10,
## cannot be computed more accurately
```

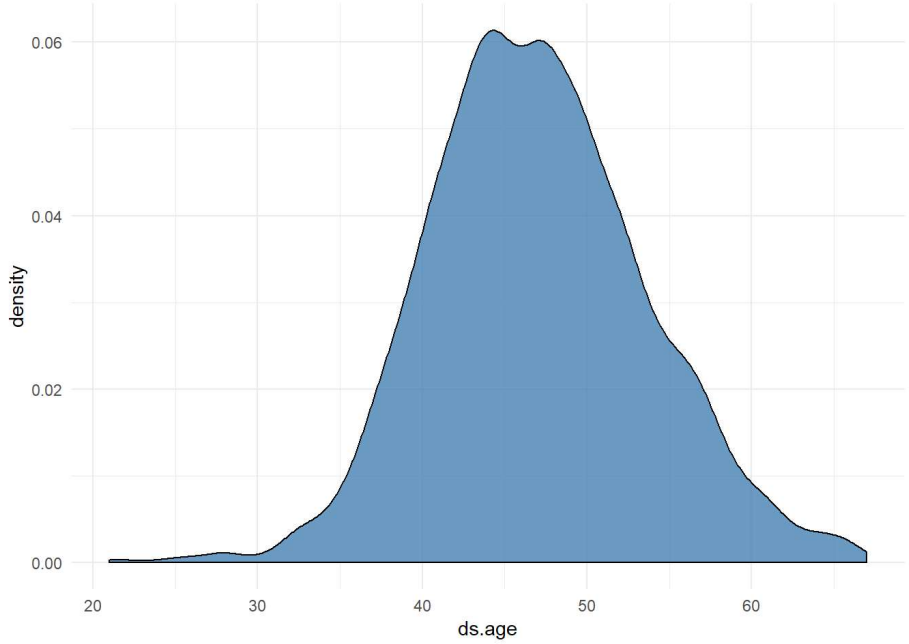
```
##
##  Cramer-von Mises normality test
##
## data:  inputdf[, 13]
## W = 1.8725, p-value = 7.37e-10
```

```
pearson.test(inputdf[,13])
```

```
##
##  Pearson chi-square normality test
##
## data:  inputdf[, 13]
## P = 8750.5, p-value < 2.2e-16
```

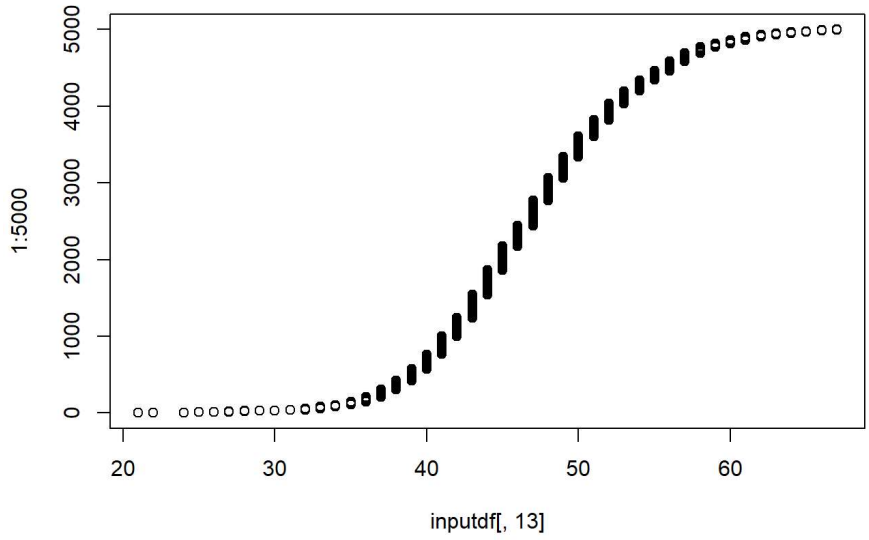
- Density Plot: Looks like close to Normal Distribution but not a perfect one

```
a <- ggplot(inputdf, aes(x = ds.age))
a +  geom_density(fill="steelblue", alpha=0.8) + theme_minimal()
```



- QQ Plot: To check the normality distribution graphically

```
qqplot(x=inputdf[,13],y=1:5000)
```



Normality test for “ds.Parents” : Fail Test

- Since p-value is less than 0.05, the distribution is not normal

```
ad.test(inputdf[,14])
```

```
##
##  Anderson-Darling normality test
##
## data:  inputdf[, 14]
## A = 510.61, p-value < 2.2e-16
```

```
cvm.test(inputdf[,14])
```

```
## Warning in cvm.test(inputdf[, 14]): p-value is smaller than 7.37e-10,
## cannot be computed more accurately
```

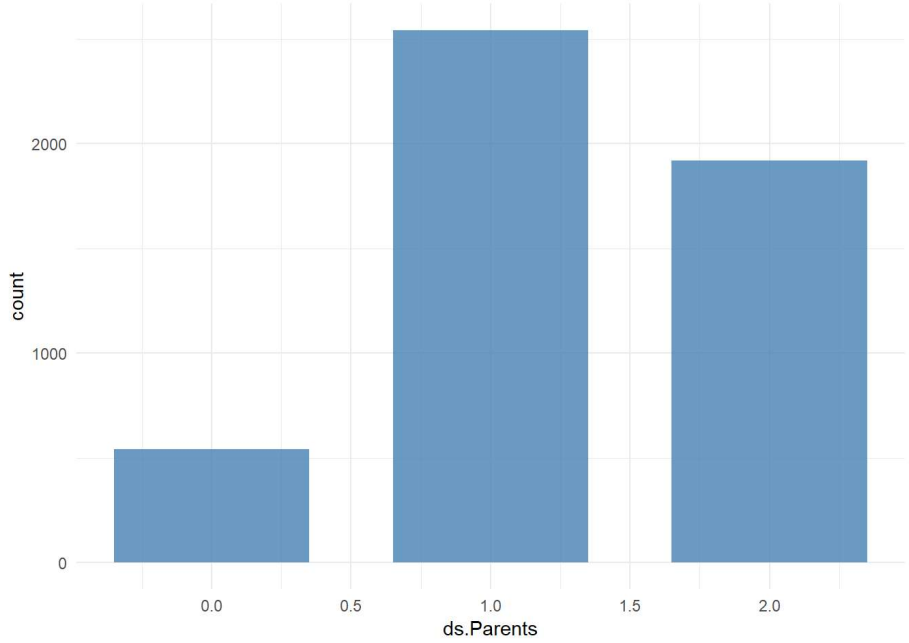
```
##
##  Cramer-von Mises normality test
##
## data:  inputdf[, 14]
## W = 88.336, p-value = 7.37e-10
```

```
pearson.test(inputdf[,14])
```

```
##
##  Pearson chi-square normality test
##
## data:  inputdf[, 14]
## P = 122220, p-value < 2.2e-16
```

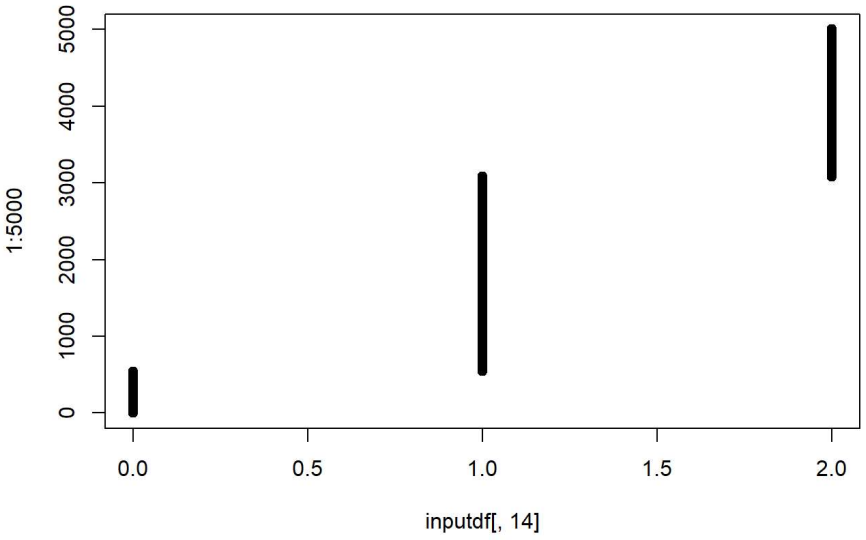
- Bar/Density Plot:

```
a <- ggplot(inputdf, aes(x = ds.Parents))
a +  geom_bar(fill="steelblue",  width = 0.7, alpha=0.8) + theme_minimal()
```



- QQ Plot: To check the normality distribution graphically

```
qqplot(x=inputdf[,14],y=1:5000)
```



Normality test for “ds.family.size”: Fail Test

- Since p-value is less than 0.05, the distribution is not normal

```
ad.test(inputdf[,15])
```

```
##
##  Anderson-Darling normality test
##
## data:  inputdf[, 15]
## A = 985.81, p-value < 2.2e-16
```

```
cvm.test(inputdf[,15])
```

```
## Warning in cvm.test(inputdf[, 15]): p-value is smaller than 7.37e-10,
## cannot be computed more accurately
```

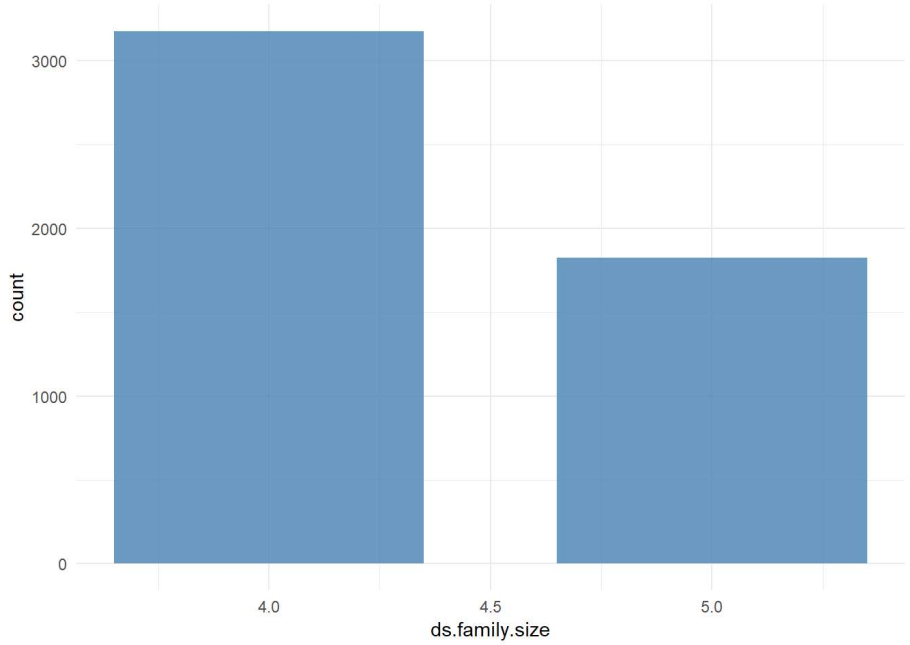
```
##
##  Cramer-von Mises normality test
##
## data:  inputdf[, 15]
## W = 169.01, p-value = 7.37e-10
```

```
pearson.test(inputdf[,15])
```

```
##
##  Pearson chi-square normality test
##
## data:  inputdf[, 15]
## P = 158620, p-value < 2.2e-16
```

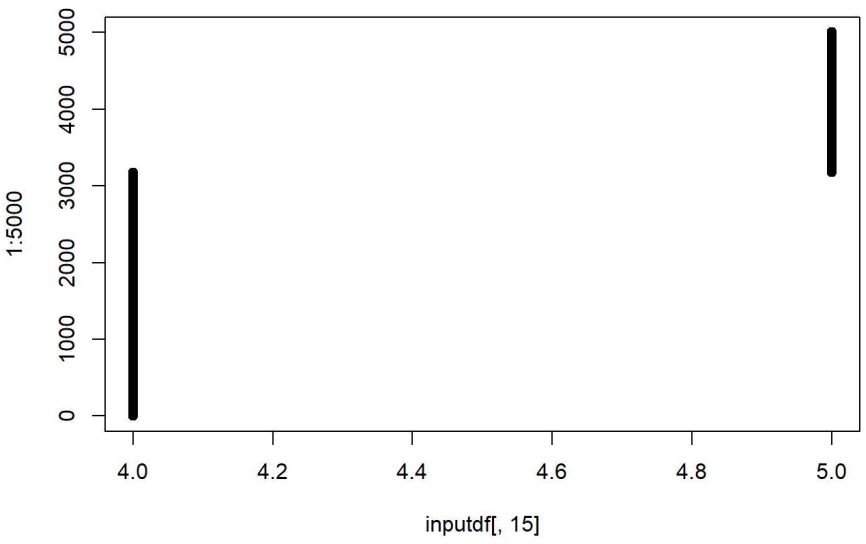
- Bar/Density Plot:

```
a <- ggplot(inputdf, aes(x = ds.family.size))
a +  geom_bar(fill="steelblue",  width = 0.7, alpha=0.8) + theme_minimal()
```



- QQ Plot: To check the normality distribution graphically

```
qqplot(x=inputdf[,15],y=1:5000)
```



Normality test for “ds.Children”: Fail Test

- Since p-value is less than 0.05, the distribution is not normal

```
ad.test(inputdf[,16])
```

```
##
##  Anderson-Darling normality test
##
## data:  inputdf[, 16]
## A = 666.94, p-value < 2.2e-16
```

```
cvm.test(inputdf[,16])
```

```
## Warning in cvm.test(inputdf[, 16]): p-value is smaller than 7.37e-10,
## cannot be computed more accurately
```

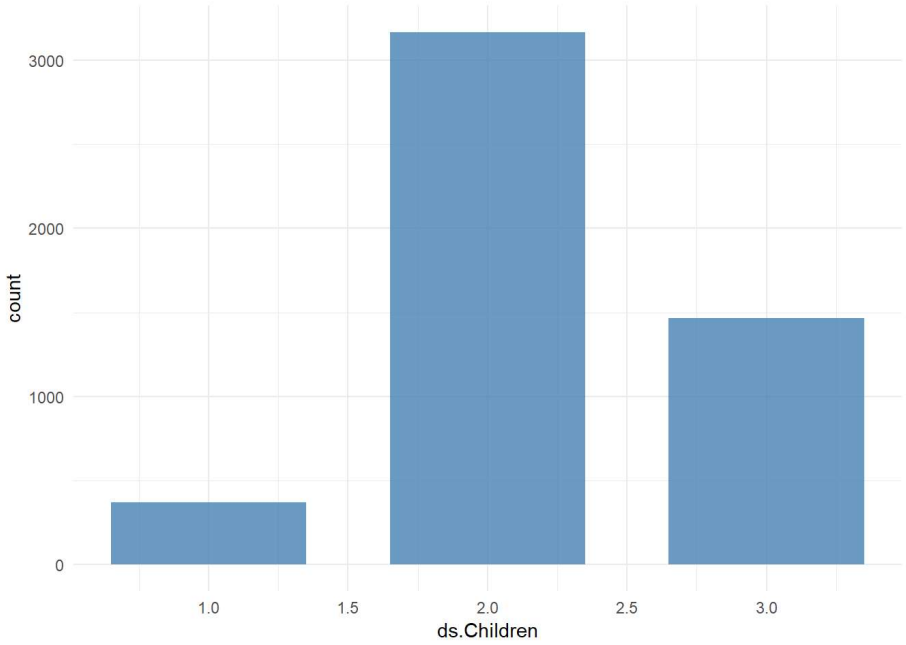
```
##
##  Cramer-von Mises normality test
##
## data:  inputdf[, 16]
## W = 127.7, p-value = 7.37e-10
```

```
pearson.test(inputdf[,16])
```

```
##
##  Pearson chi-square normality test
##
## data:  inputdf[, 16]
## P = 145040, p-value < 2.2e-16
```

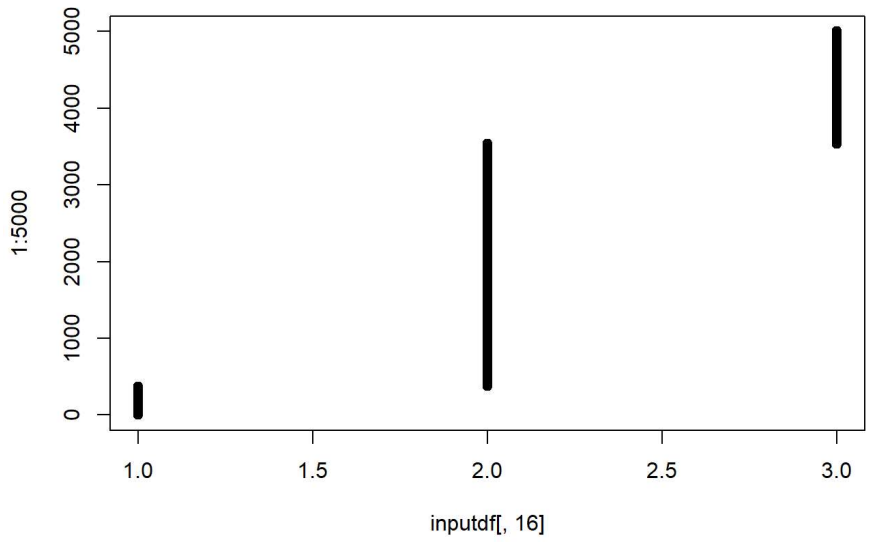
- Bar/Density Plot:

```
a <- ggplot(inputdf, aes(x = ds.Children))
a +  geom_bar(fill="steelblue", width = 0.7, alpha=0.8) + theme_minimal()
```



- QQ Plot: To check the normality distribution graphically

```
qqplot(x=inputdf[,16],y=1:5000)
```



Normality test for “ds.Avg.Family.Age”: Fail Test

- Since p-value is less than 0.05, the distribution is not normal

```
ad.test(inputdf[,17])
```

```
##
##  Anderson-Darling normality test
##
## data:  inputdf[, 17]
## A = 80.211, p-value < 2.2e-16
```

```
cvm.test(inputdf[,17])
```

```
## Warning in cvm.test(inputdf[, 17]): p-value is smaller than 7.37e-10,
## cannot be computed more accurately
```

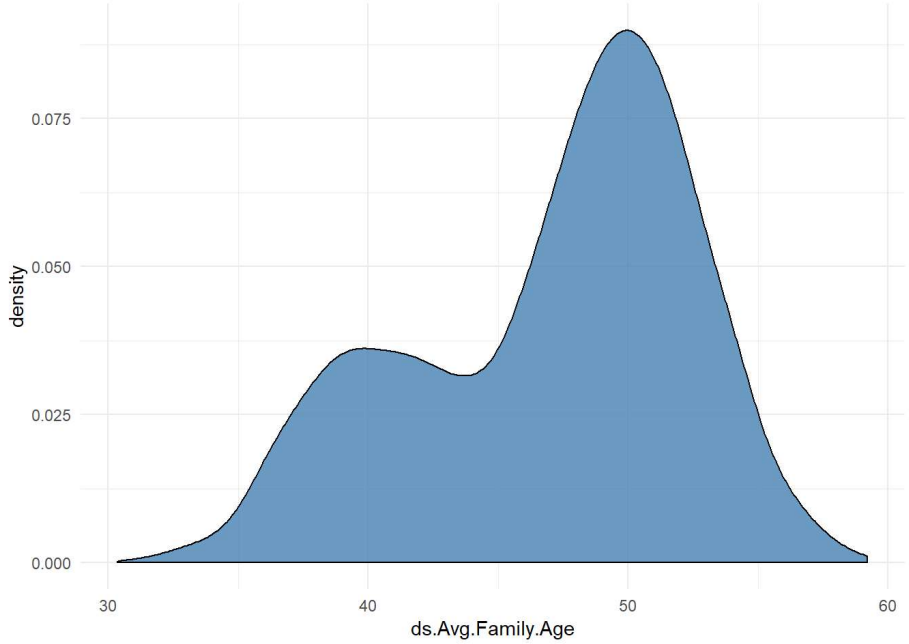
```
##
##  Cramer-von Mises normality test
##
## data:  inputdf[, 17]
## W = 14.532, p-value = 7.37e-10
```

```
pearson.test(inputdf[,17])
```

```
##
##  Pearson chi-square normality test
##
## data:  inputdf[, 17]
## P = 955.63, p-value < 2.2e-16
```

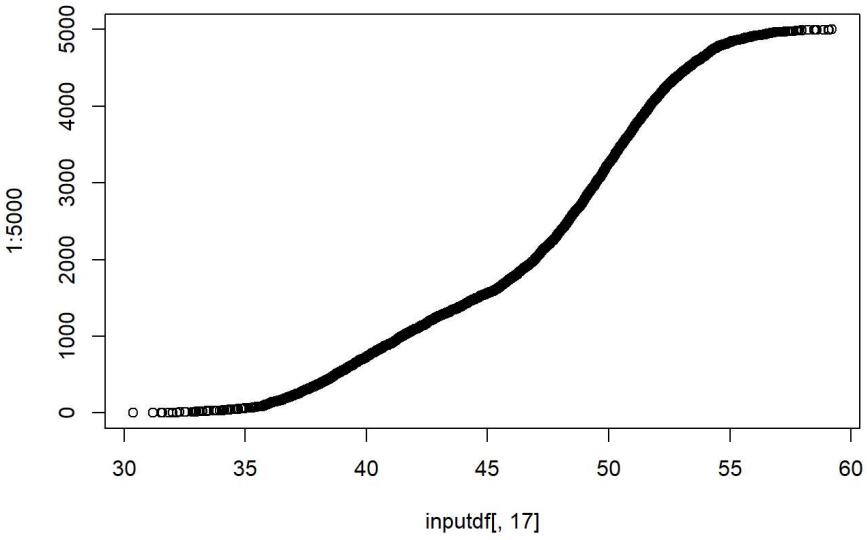
- Density Plot: Check for Normal Distribution

```
a <- ggplot(inputdf, aes(x = ds.Avg.Family.Age))
a +  geom_density(fill="steelblue", alpha=0.8) + theme_minimal()
```



- QQ Plot: To check the normality distribution graphically

```
qqplot(x=inputdf[,17],y=1:5000)
```



Normality test for “ds.Annual.income”: Pass Test

- Since p-value is less than 0.05, the distribution is not normal

```
ad.test(inputdf[,18])
```

```
##
##  Anderson-Darling normality test
##
## data:  inputdf[, 18]
## A = 0.43231, p-value = 0.3041
```

```
cvm.test(inputdf[,18])
```

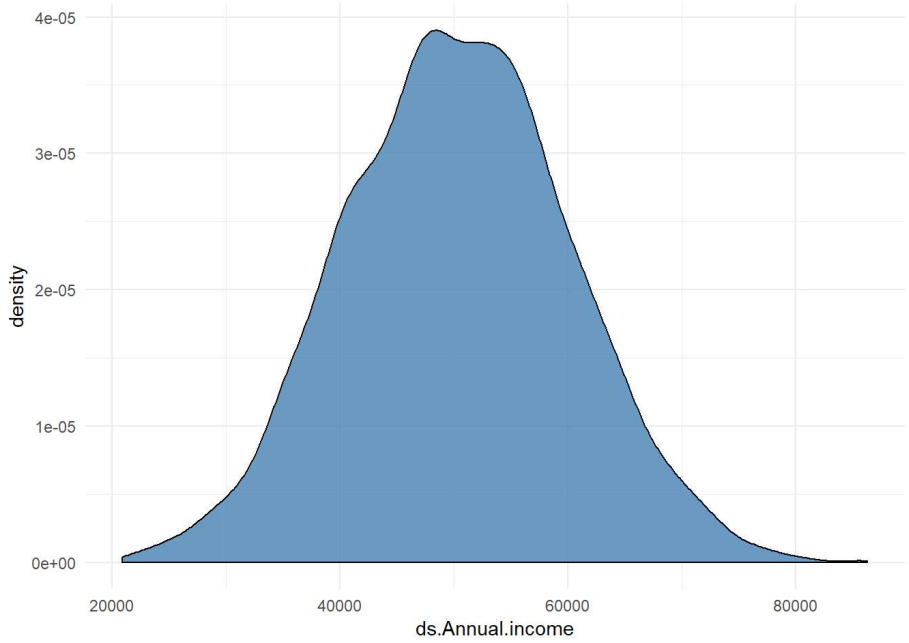
```
##
##  Cramer-von Mises normality test
##
## data:  inputdf[, 18]
## W = 0.064548, p-value = 0.3296
```

```
pearson.test(inputdf[,18])
```

```
##
##  Pearson chi-square normality test
##
## data:  inputdf[, 18]
## P = 62.927, p-value = 0.3062
```

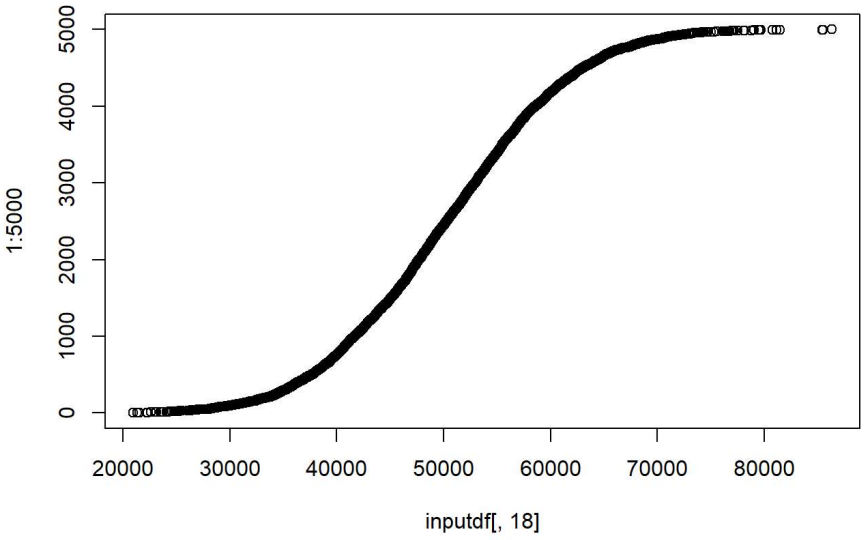
- Density Plot: Looks like Normal Distribution

```
a <- ggplot(inputdf, aes(x = ds.Annual.income))
a +  geom_density(fill="steelblue", alpha=0.8) + theme_minimal()
```



- QQ Plot: To check the normality distribution graphically

```
qqplot(x=inputdf[,18],y=1:5000)
```



Normality test for “ds.Avg.annual.inc”: Pass Test

- Since p-value is less than 0.05, the distribution is not normal

```
ad.test(inputdf[,19])
```

```
##
##  Anderson-Darling normality test
##
## data:  inputdf[, 19]
## A = 0.53137, p-value = 0.1744
```

```
cvm.test(inputdf[,19])
```

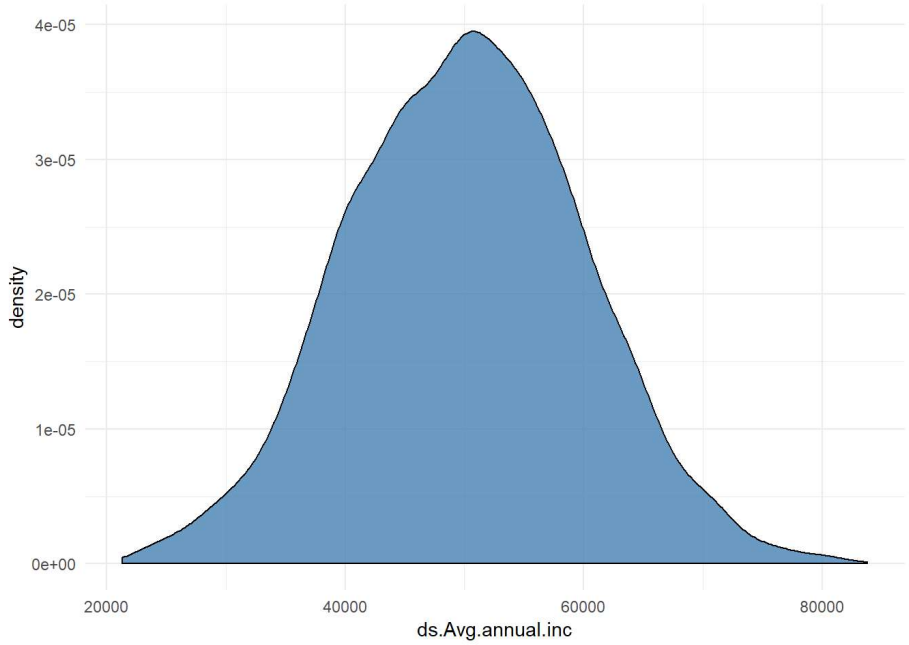
```
##
##  Cramer-von Mises normality test
##
## data:  inputdf[, 19]
## W = 0.088778, p-value = 0.1595
```

```
pearson.test(inputdf[,19])
```

```
##
##  Pearson chi-square normality test
##
## data:  inputdf[, 19]
## P = 69.881, p-value = 0.1364
```

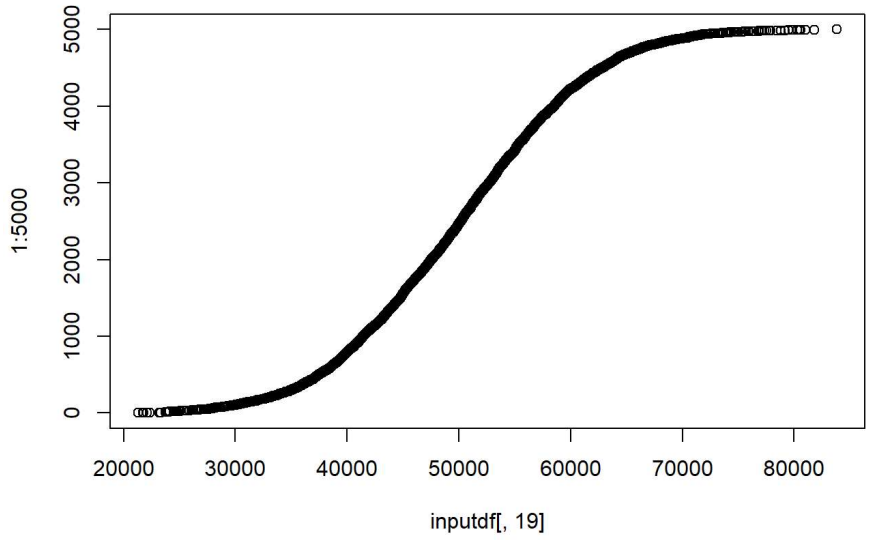
- Density Plot: Check for Normal Distribution

```
a <- ggplot(inputdf, aes(x = ds.Avg.annual.inc))
a +  geom_density(fill="steelblue", alpha=0.8) + theme_minimal()
```



- QQ Plot: To check the normality distribution graphically

```
qqplot(x=inputdf[,19],y=1:5000)
```



Normality test for “ds.Annual.Expenses”: Pass Test

- Since p-value is less than 0.05, the distribution is not normal

```
ad.test(inputdf[,20])
```

```
##
##  Anderson-Darling normality test
##
## data:  inputdf[, 20]
## A = 0.32922, p-value = 0.5157
```

```
cvm.test(inputdf[,20])
```

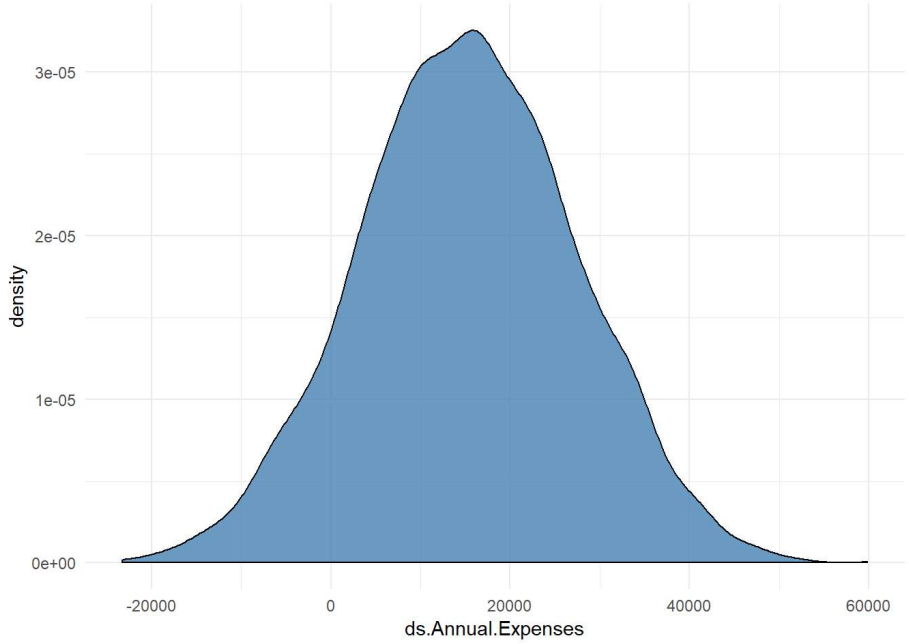
```
##
##  Cramer-von Mises normality test
##
## data:  inputdf[, 20]
## W = 0.040521, p-value = 0.6722
```

```
pearson.test(inputdf[,20])
```

```
##
##  Pearson chi-square normality test
##
## data:  inputdf[, 20]
## P = 61.536, p-value = 0.3507
```

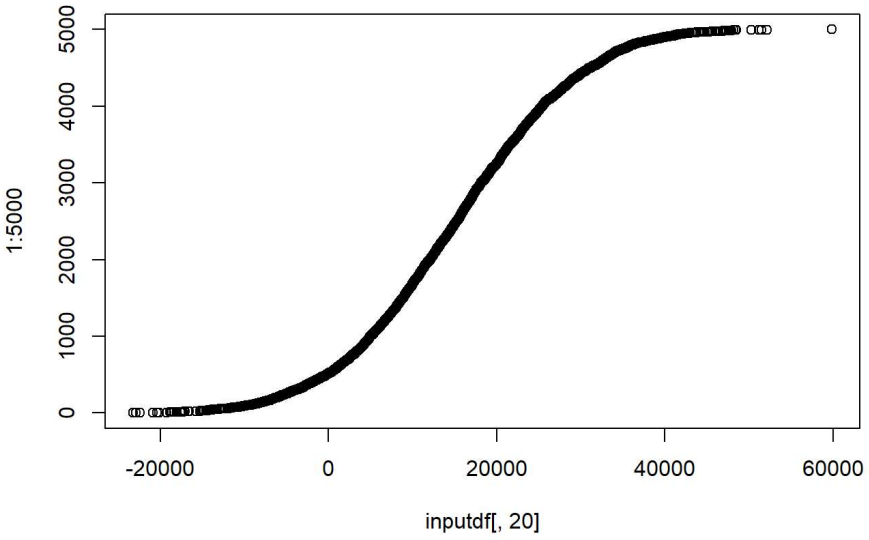
- Density Plot: Check for Normal Distribution

```
a <- ggplot(inputdf, aes(x = ds.Annual.Expenses))
a +  geom_density(fill="steelblue", alpha=0.8) + theme_minimal()
```



- QQ Plot: To check the normality distribution graphically

```
qqplot(x=inputdf[,20],y=1:5000)
```



Normality test for “ds.Saving.Amount”: Pass Test

- Since p-value is less than 0.05, the distribution is not normal

```
ad.test(inputdf[,21])
```

```
##
##  Anderson-Darling normality test
##
## data:  inputdf[, 21]
## A = 0.12911, p-value = 0.9835
```

```
cvm.test(inputdf[,21])
```

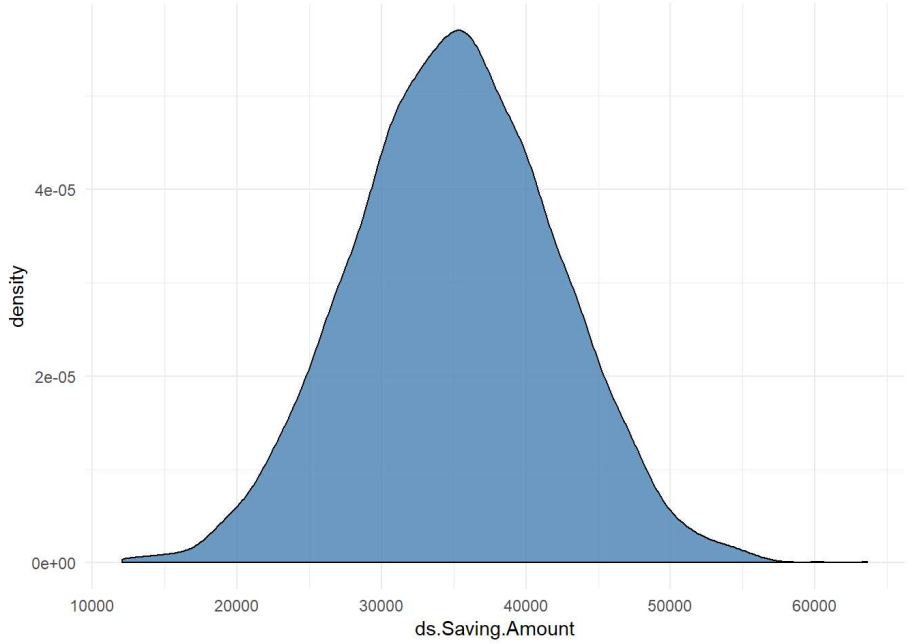
```
##
##  Cramer-von Mises normality test
##
## data:  inputdf[, 21]
## W = 0.016617, p-value = 0.9892
```

```
pearson.test(inputdf[,21])
```

```
##
##  Pearson chi-square normality test
##
## data:  inputdf[, 21]
## P = 36.916, p-value = 0.986
```

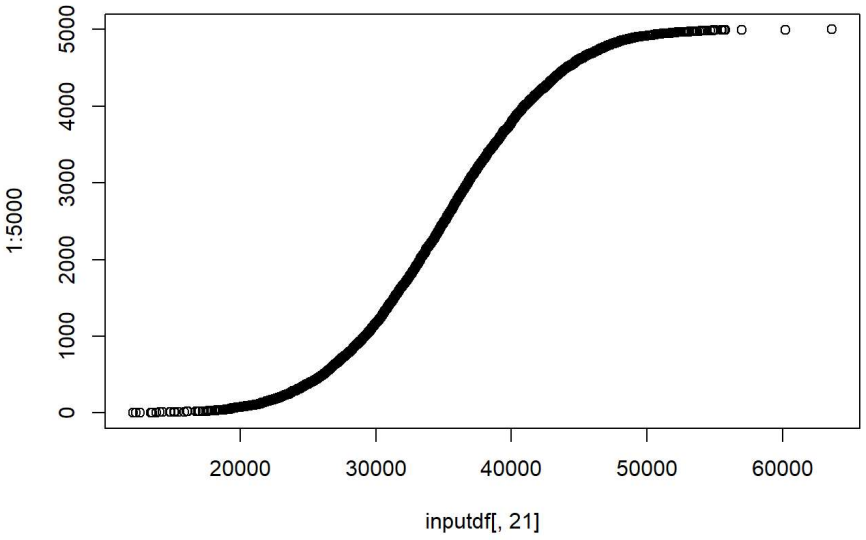
- Density Plot: Check for Normal Distribution

```
a <- ggplot(inputdf, aes(x = ds.Saving.Amount))
a +  geom_density(fill="steelblue", alpha=0.8) + theme_minimal()
```



- QQ Plot: To check the normality distribution graphically

```
qqplot(x=inputdf[,21],y=1:5000)
```



Normality test for “ds.Credit.Cards”: Fail Test

- Since p-value is less than 0.05, the distribution is not normal

```
ad.test(inputdf[,22])
```

```
##
##  Anderson-Darling normality test
##
## data:  inputdf[, 22]
## A = 192.1, p-value < 2.2e-16
```

```
cvm.test(inputdf[,22])
```

```
## Warning in cvm.test(inputdf[, 22]): p-value is smaller than 7.37e-10,
## cannot be computed more accurately
```

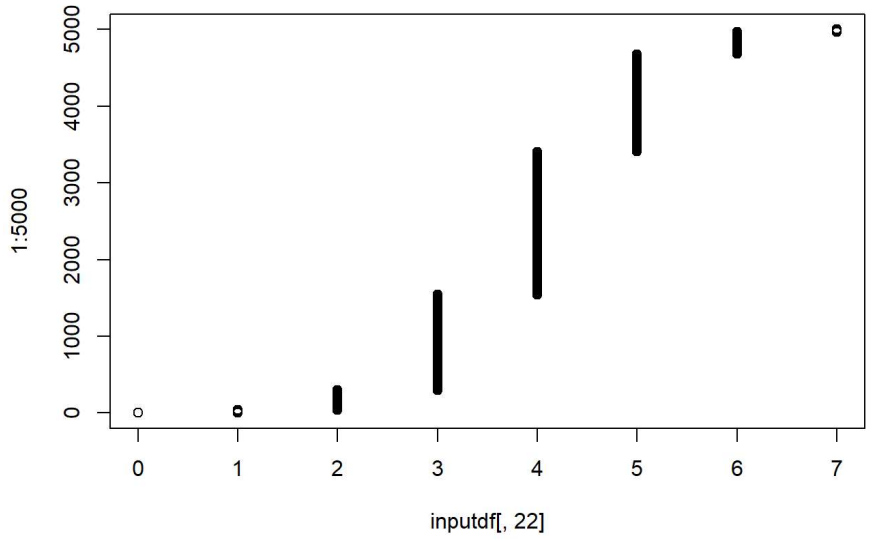
```
##
##  Cramer-von Mises normality test
##
## data:  inputdf[, 22]
## W = 36.119, p-value = 7.37e-10
```

```
pearson.test(inputdf[,22])
```

```
##
##  Pearson chi-square normality test
##
## data:  inputdf[, 22]
## P = 77848, p-value < 2.2e-16
```

- QQ Plot: To check the normality distribution graphically

```
qqplot(x=inputdf[,22],y=1:5000)
```



Normality test for “ds.Two.Wheelers”: Fail Test

- Since p-value is less than 0.05, the distribution is not normal

```
ad.test(inputdf[,23])
```

```
##
##  Anderson-Darling normality test
##
## data:  inputdf[, 23]
## A = 244.41, p-value < 2.2e-16
```

```
cvm.test(inputdf[,23])
```

```
## Warning in cvm.test(inputdf[, 23]): p-value is smaller than 7.37e-10,
## cannot be computed more accurately
```

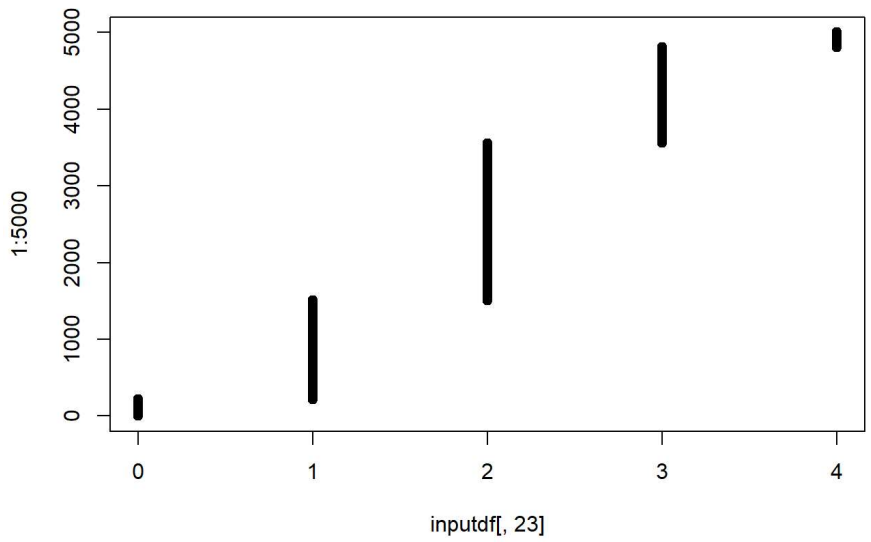
```
##
##  Cramer-von Mises normality test
##
## data:  inputdf[, 23]
## W = 44.999, p-value = 7.37e-10
```

```
pearson.test(inputdf[,23])
```

```
##
##  Pearson chi-square normality test
##
## data:  inputdf[, 23]
## P = 86766, p-value < 2.2e-16
```

- QQ Plot: To check the normality distribution graphically

```
qqplot(x=inputdf[,23],y=1:5000)
```



Normality test for “ds.Four.Wheelers”: Fail Test

- Since p-value is less than 0.05, the distribution is not normal

```
ad.test(inputdf[,24])
```

```
##
##  Anderson-Darling normality test
##
## data:  inputdf[, 24]
## A = 1038.6, p-value < 2.2e-16
```

```
cvm.test(inputdf[,24])
```

```
## Warning in cvm.test(inputdf[, 24]): p-value is smaller than 7.37e-10,
## cannot be computed more accurately
```

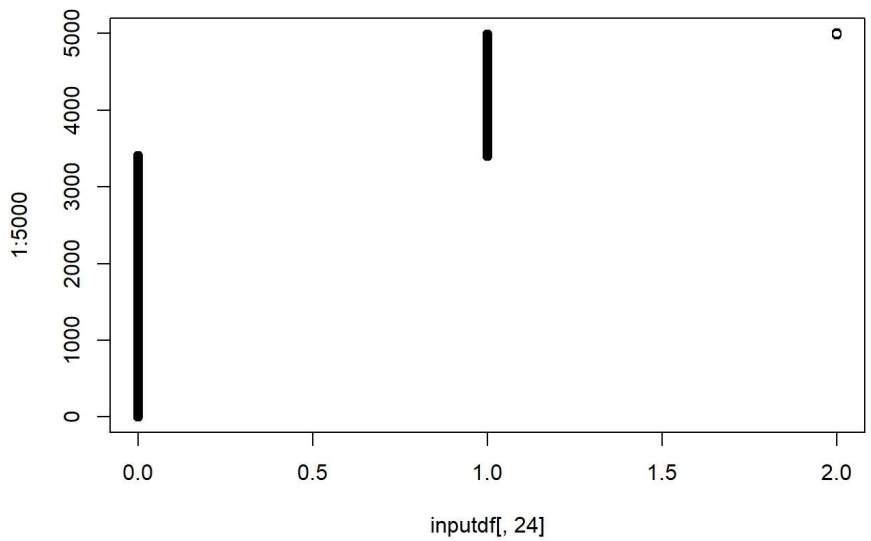
```
##
##  Cramer-von Mises normality test
##
## data:  inputdf[, 24]
## W = 185.6, p-value = 7.37e-10
```

```
pearson.test(inputdf[,24])
```

```
##
##  Pearson chi-square normality test
##
## data:  inputdf[, 24]
## P = 167060, p-value < 2.2e-16
```

- QQ Plot: To check the normality distribution graphically

```
qqplot(x=inputdf[,24],y=1:5000)
```



Normality test for “ds.Bank.Accounts”: Fail Test

- Since p-value is less than 0.05, the distribution is not normal

```
ad.test(inputdf[,25])
```

```
##
##  Anderson-Darling normality test
##
## data:  inputdf[, 25]
## A = 205.2, p-value < 2.2e-16
```

```
cvm.test(inputdf[,25])
```

```
## Warning in cvm.test(inputdf[, 25]): p-value is smaller than 7.37e-10,
## cannot be computed more accurately
```

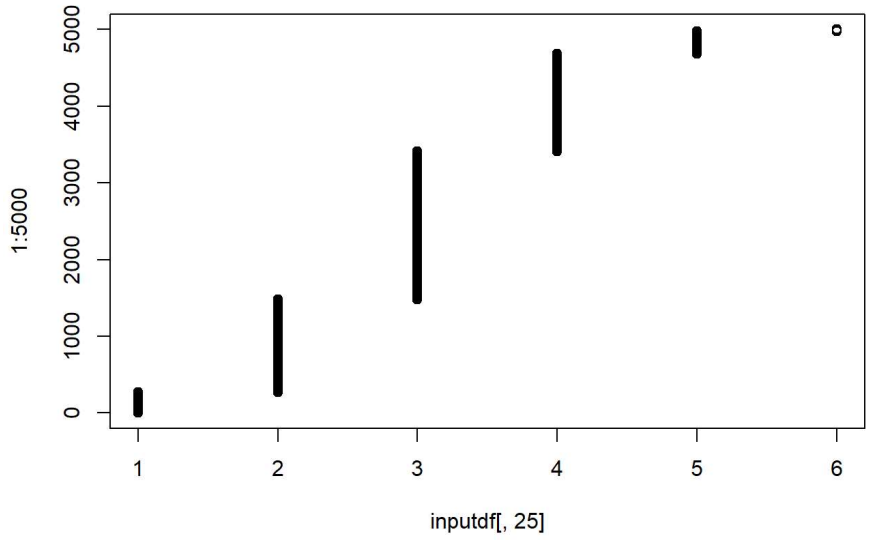
```
##
##  Cramer-von Mises normality test
##
## data:  inputdf[, 25]
## W = 38.507, p-value = 7.37e-10
```

```
pearson.test(inputdf[,25])
```

```
##
##  Pearson chi-square normality test
##
## data:  inputdf[, 25]
## P = 80002, p-value < 2.2e-16
```

- QQ Plot: To check the normality distribution graphically

```
qqplot(x=inputdf[,25],y=1:5000)
```



Normality test for “ds.Houses”: Fail Test

- Since p-value is less than 0.05, the distribution is not normal

```
ad.test(inputdf[,26])
```

```
##
##  Anderson-Darling normality test
##
## data:  inputdf[, 26]
## A = 458.52, p-value < 2.2e-16
```

```
cvm.test(inputdf[,26])
```

```
## Warning in cvm.test(inputdf[, 26]): p-value is smaller than 7.37e-10,
## cannot be computed more accurately
```

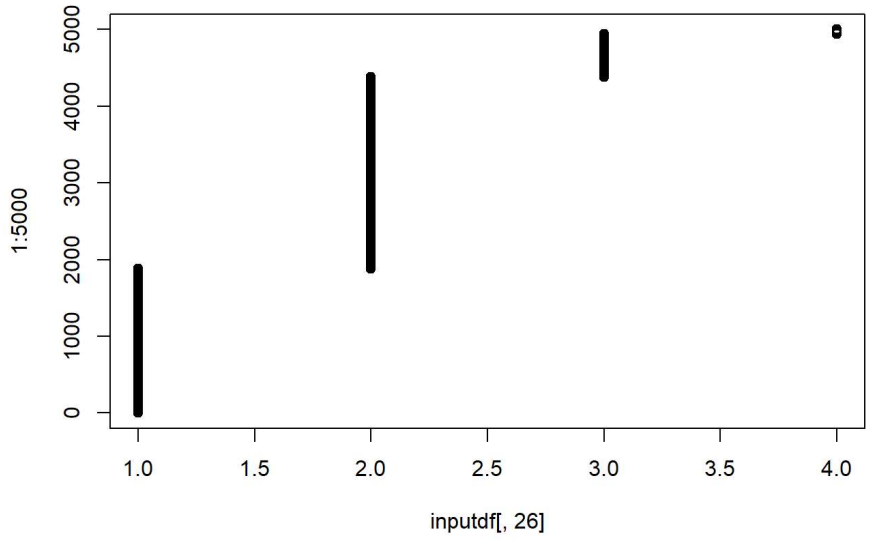
```
##
##  Cramer-von Mises normality test
##
## data:  inputdf[, 26]
## W = 80.562, p-value = 7.37e-10
```

```
pearson.test(inputdf[,26])
```

```
##
##  Pearson chi-square normality test
##
## data:  inputdf[, 26]
## P = 118130, p-value < 2.2e-16
```

- QQ Plot: To check the normality distribution graphically

```
qqplot(x=inputdf[,26],y=1:5000)
```



Normality test for “ds.Estates”: Fail Test

- Since p-value is less than 0.05, the distribution is not normal

```
ad.test(inputdf[,27])
```

```
##
##  Anderson-Darling normality test
##
## data:  inputdf[, 27]
## A = 1065.1, p-value < 2.2e-16
```

```
cvm.test(inputdf[,27])
```

```
## Warning in cvm.test(inputdf[, 27]): p-value is smaller than 7.37e-10,
## cannot be computed more accurately
```

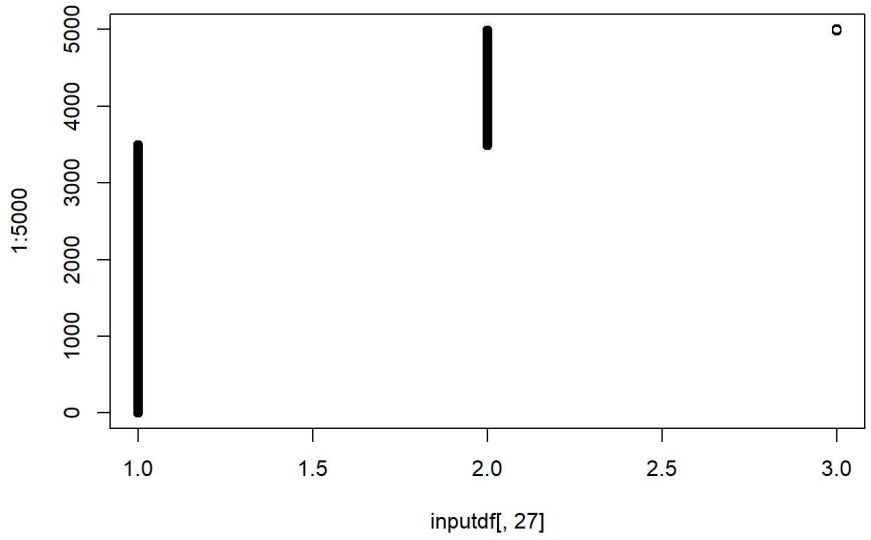
```
##
##  Cramer-von Mises normality test
##
## data:  inputdf[, 27]
## W = 193.22, p-value = 7.37e-10
```

```
pearson.test(inputdf[,27])
```

```
##
##  Pearson chi-square normality test
##
## data:  inputdf[, 27]
## P = 170900, p-value < 2.2e-16
```

- QQ Plot: To check the normality distribution graphically

```
qqplot(x=inputdf[,27],y=1:5000)
```



Normality test for “ds.Yearly.Travel.Dist.Air”: Pass Test

- Since p-value is less than 0.05, the distribution is not normal

```
ad.test(inputdf[,28])
```

```
##
##  Anderson-Darling normality test
##
## data:  inputdf[, 28]
## A = 0.4409, p-value = 0.2901
```

```
cvm.test(inputdf[,28])
```

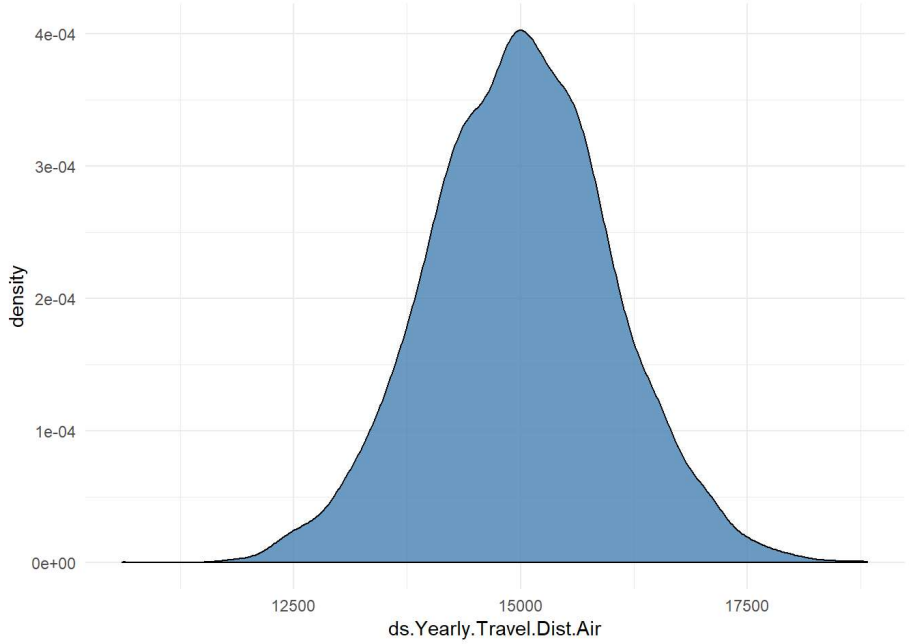
```
##
##  Cramer-von Mises normality test
##
## data:  inputdf[, 28]
## W = 0.063338, p-value = 0.3419
```

```
pearson.test(inputdf[,28])
```

```
##
##  Pearson chi-square normality test
##
## data:  inputdf[, 28]
## P = 50.58, p-value = 0.7447
```

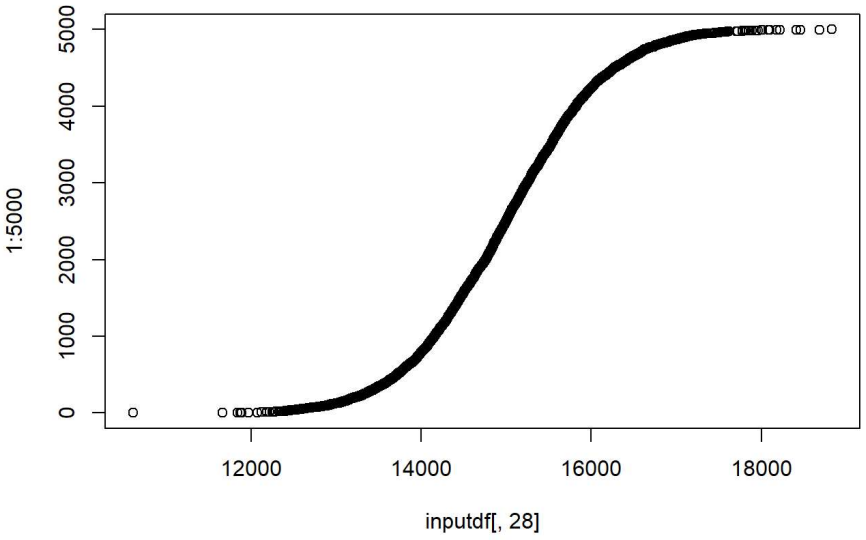
- Density Plot: Check for Normal Distribution

```
a <- ggplot(inputdf, aes(x = ds.Yearly.Travel.Dist.Air))
a +  geom_density(fill="steelblue", alpha=0.8) + theme_minimal()
```



- QQ Plot: To check the normality distribution graphically

```
qqplot(x=inputdf[,28],y=1:5000)
```



Normality test for “ds.Yearly.Travel.Dist.Road”: Pass Test

- Since p-value is less than 0.05, the distribution is not normal

```
ad.test(inputdf[,29])
```

```
##
##  Anderson-Darling normality test
##
## data:  inputdf[, 29]
## A = 0.32962, p-value = 0.515
```

```
cvm.test(inputdf[,29])
```

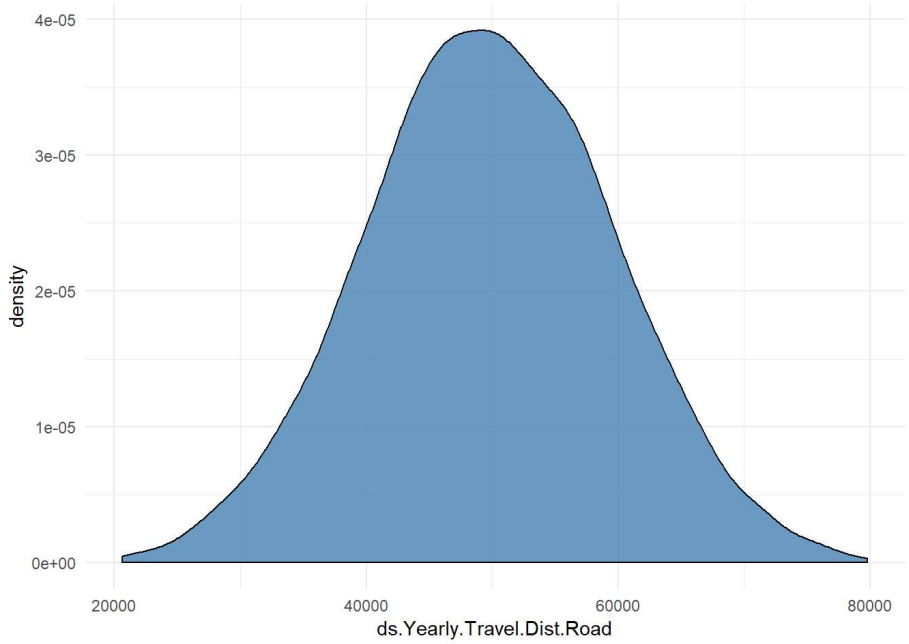
```
##
##  Cramer-von Mises normality test
##
## data:  inputdf[, 29]
## W = 0.055444, p-value = 0.4344
```

```
pearson.test(inputdf[,29])
```

```
##
##  Pearson chi-square normality test
##
## data:  inputdf[, 29]
## P = 72.736, p-value = 0.09215
```

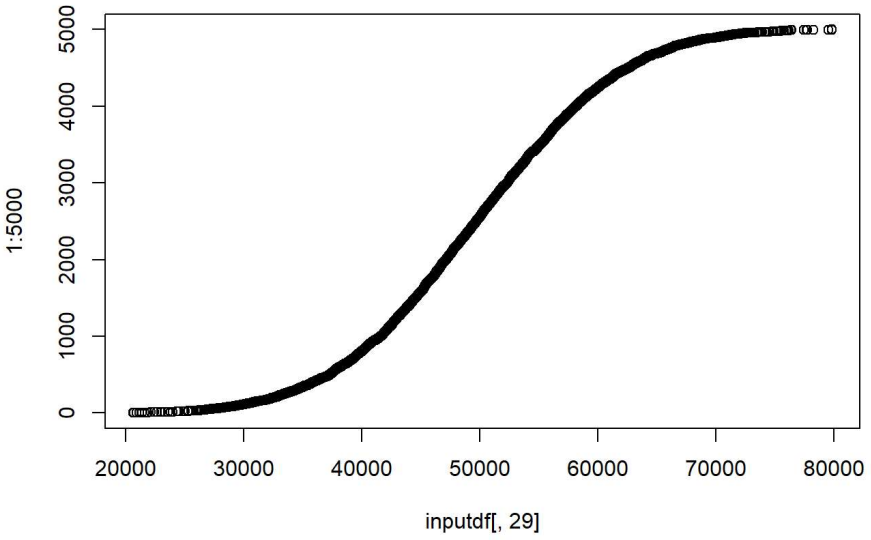
- Density Plot: Check for Normal Distribution

```
a <- ggplot(inputdf, aes(x = ds.Yearly.Travel.Dist.Road))
a +  geom_density(fill="steelblue", alpha=0.8) + theme_minimal()
```



- QQ Plot: To check the normality distribution graphically

```
qqplot(x=inputdf[,29],y=1:5000)
```



Normality test for “ds.Policy.Face.Value”: Fail Test

- Since p-value is less than 0.05, the distribution is not normal

```
ad.test(inputdf[,30])
```

```
##
##  Anderson-Darling normality test
##
## data:  inputdf[, 30]
## A = 205.61, p-value < 2.2e-16
```

```
cvm.test(inputdf[,30])
```

```
## Warning in cvm.test(inputdf[, 30]): p-value is smaller than 7.37e-10,
## cannot be computed more accurately
```

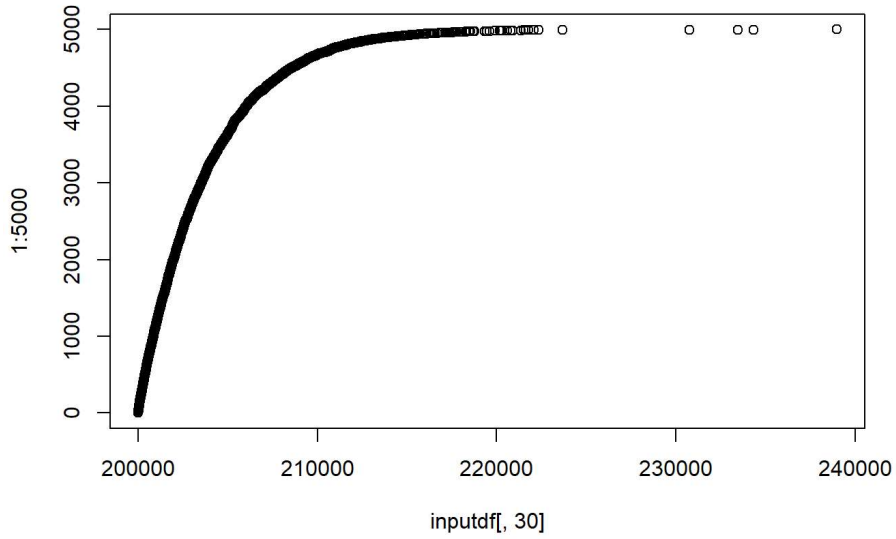
```
##
##  Cramer-von Mises normality test
##
## data:  inputdf[, 30]
## W = 34.765, p-value = 7.37e-10
```

```
pearson.test(inputdf[,30])
```

```
##
##  Pearson chi-square normality test
##
## data:  inputdf[, 30]
## P = 3239.5, p-value < 2.2e-16
```

- QQ Plot: To check the normality distribution graphically

```
qqplot(x=inputdf[,30],y=1:5000)
```



Normality test for “ds.Claimed.Amount”: Fail Test

- Since p-value is less than 0.05, the distribution is not normal

```
ad.test(inputdf[,31])
```

```
##
##  Anderson-Darling normality test
##
## data:  inputdf[, 31]
## A = 368.62, p-value < 2.2e-16
```

```
cvm.test(inputdf[,31])
```

```
## Warning in cvm.test(inputdf[, 31]): p-value is smaller than 7.37e-10,
## cannot be computed more accurately
```

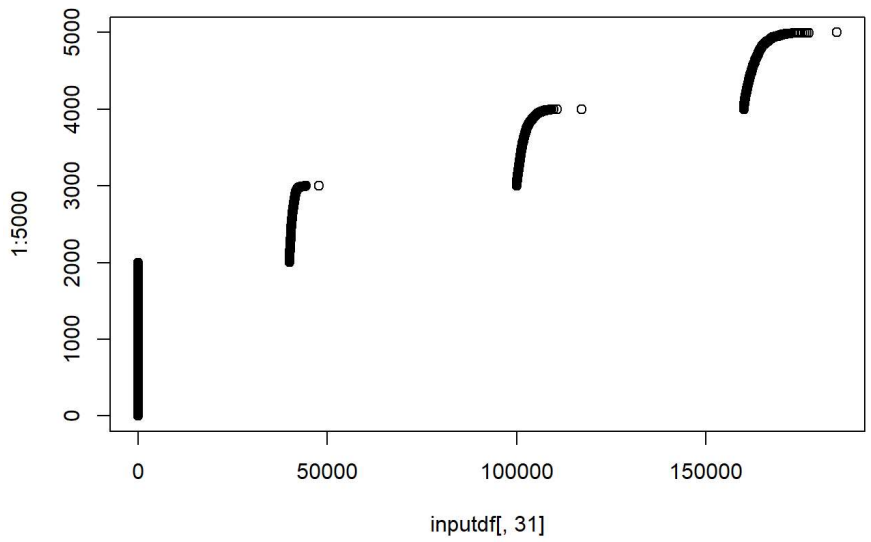
```
##
##  Cramer-von Mises normality test
##
## data:  inputdf[, 31]
## W = 56.999, p-value = 7.37e-10
```

```
pearson.test(inputdf[,31])
```

```
##
##  Pearson chi-square normality test
##
## data:  inputdf[, 31]
## P = 66609, p-value < 2.2e-16
```

- QQ Plot: To check the normality distribution graphically

```
qqplot(x=inputdf[,31],y=1:5000)
```



Normality test for “ds.Yearly.Premium”: Pass Test

- Since p-value is less than 0.05, the distribution is not normal

```
ad.test(inputdf[,32])
```

```
##
##  Anderson-Darling normality test
##
## data:  inputdf[, 32]
## A = 5.1257, p-value = 1.158e-12
```

```
cvm.test(inputdf[,32])
```

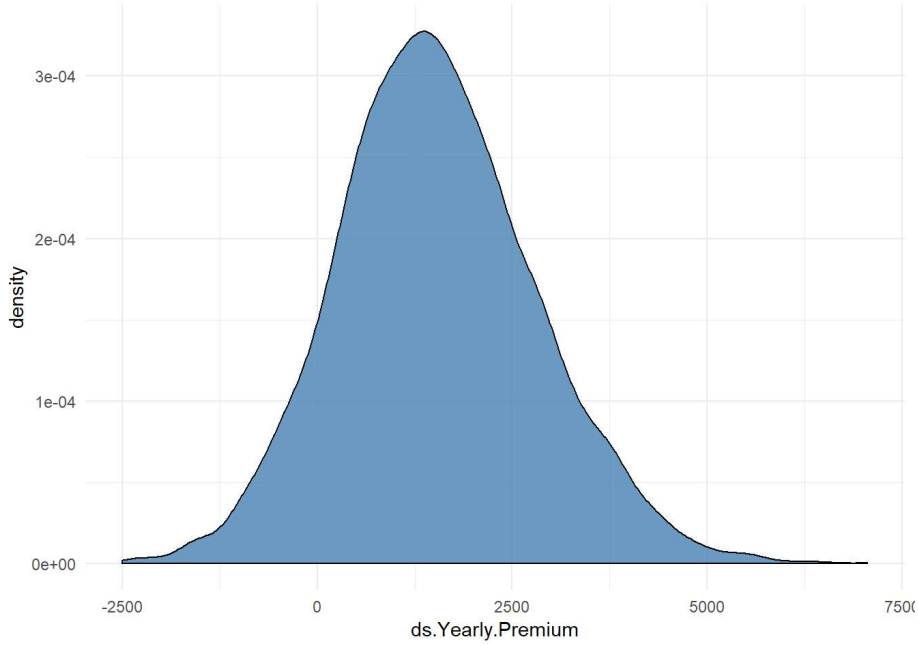
```
##
##  Cramer-von Mises normality test
##
## data:  inputdf[, 32]
## W = 0.89283, p-value = 4.437e-09
```

```
pearson.test(inputdf[,32])
```

```
##
##  Pearson chi-square normality test
##
## data:  inputdf[, 32]
## P = 90.352, p-value = 0.004185
```

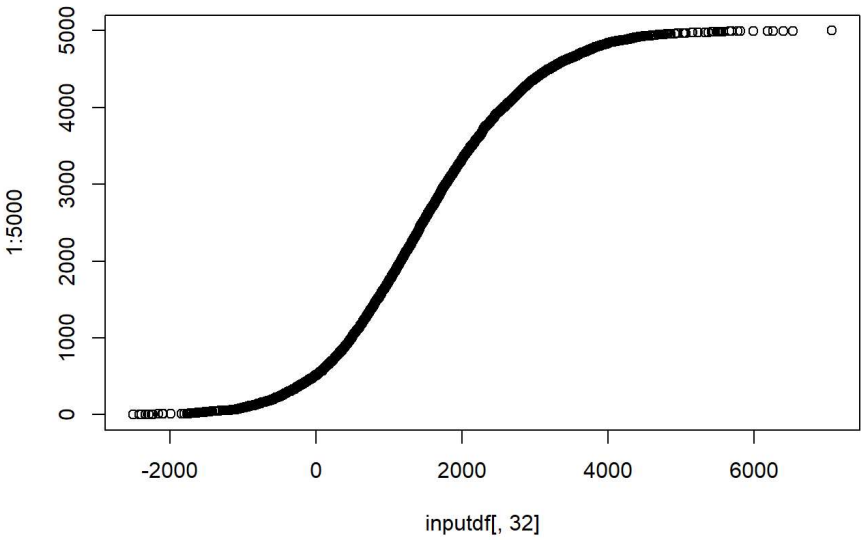
- Density Plot: Check for Normal Distribution

```
a <- ggplot(inputdf, aes(x = ds.Yearly.Premium))
a +  geom_density(fill="steelblue", alpha=0.8) + theme_minimal()
```



- QQ Plot: To check the normality distribution graphically

```
qqplot(x=inputdf[,32],y=1:5000)
```



Normality test for “ds.YearOfPurchase.1st_Prod”: Fail Test

- Since p-value is less than 0.05, the distribution is not normal

```
ad.test(inputdf[,33])
```

```
##
##  Anderson-Darling normality test
##
## data:  inputdf[, 33]
## A = 85.1, p-value < 2.2e-16
```

```
cvm.test(inputdf[,33])
```

```
## Warning in cvm.test(inputdf[, 33]): p-value is smaller than 7.37e-10,
## cannot be computed more accurately
```

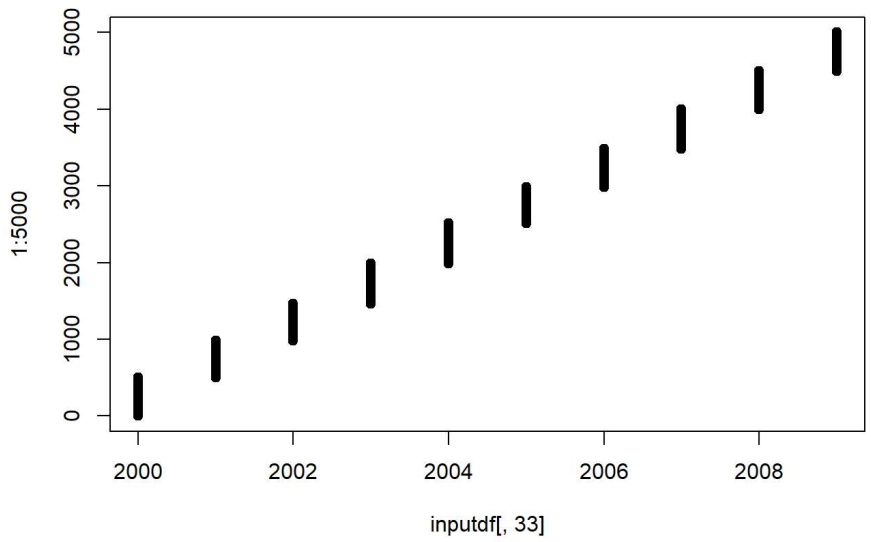
```
##
##  Cramer-von Mises normality test
##
## data:  inputdf[, 33]
## W = 11.724, p-value = 7.37e-10
```

```
pearson.test(inputdf[,33])
```

```
##
##  Pearson chi-square normality test
##
## data:  inputdf[, 33]
## P = 25531, p-value < 2.2e-16
```

- QQ Plot: To check the normality distribution graphically

```
qqplot(x=inputdf[,33],y=1:5000)
```



Normality test for “ds.YearOfPurchase.LastProd”: Fail Test

- Since p-value is less than 0.05, the distribution is not normal

```
ad.test(inputdf[,34])
```

```
##
##  Anderson-Darling normality test
##
## data:  inputdf[, 34]
## A = 19.184, p-value < 2.2e-16
```

```
cvm.test(inputdf[,34])
```

```
## Warning in cvm.test(inputdf[, 34]): p-value is smaller than 7.37e-10,
## cannot be computed more accurately
```

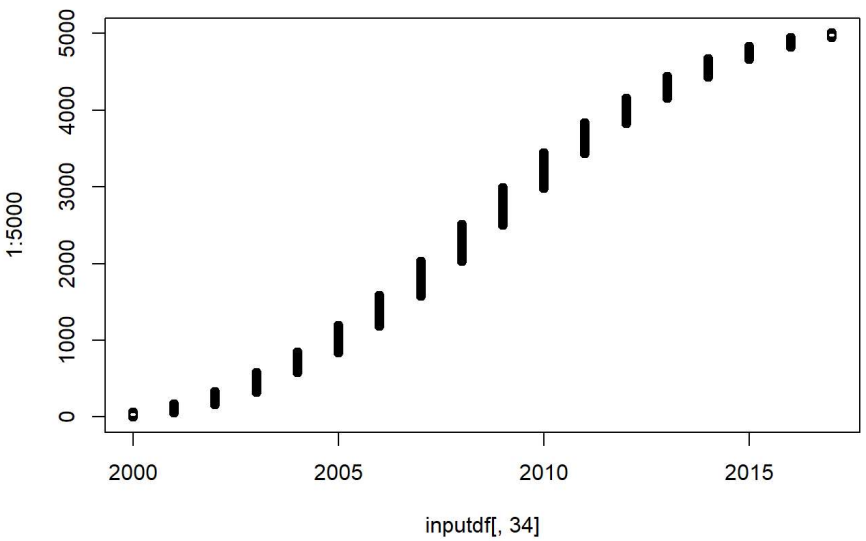
```
##
##  Cramer-von Mises normality test
##
## data:  inputdf[, 34]
## W = 3.0417, p-value = 7.37e-10
```

```
pearson.test(inputdf[,34])
```

```
##
##  Pearson chi-square normality test
##
## data:  inputdf[, 34]
## P = 16312, p-value < 2.2e-16
```

- QQ Plot: To check the normality distribution graphically

```
qqplot(x=inputdf[,34],y=1:5000)
```



Step 3 - Analysis for Clustering:

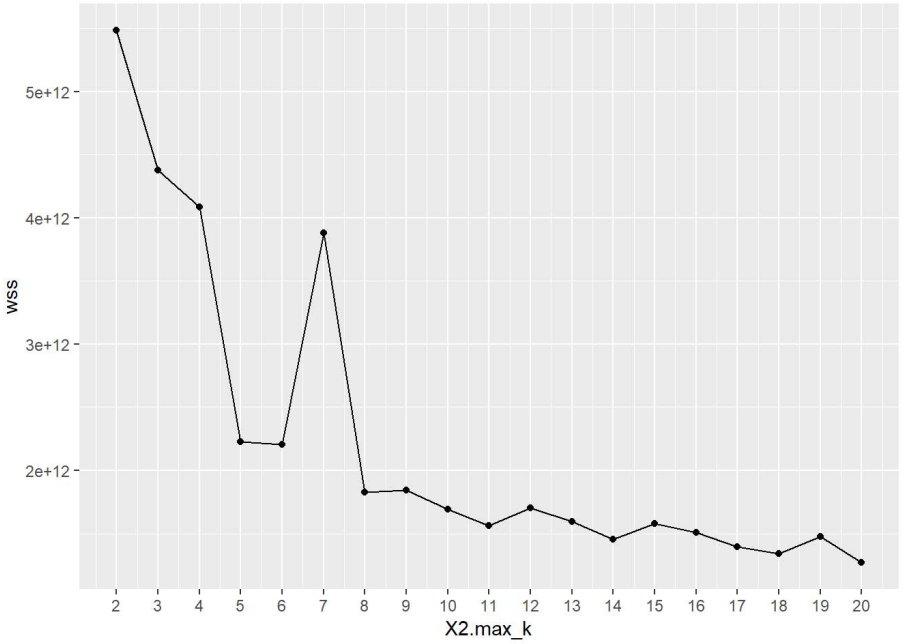
K-Means Clustering

Initially, lets pull all the numeric variable to build cluster. Variable not normally distributed will be of lesser important but just to experiment, we are considering.

```
prod_df <- inputdf[,13:34]
prod_df <- na.omit(prod_df)
```

Test to calculate the optimal number of clusters

```
kmean_withinss <- function(k) {
  p_cluster <- kmeans(prod_df, k)
  return (p_cluster$tot.withinss)
}

# Set maximum cluster
max_k <-20
# Run algorithm over a range of k
wss <- sapply(2:max_k, kmean_withinss)

# Create a data frame to plot the graph
elbow <-data.frame(2:max_k, wss)

library(ggplot2)
# Plot the graph with gglop
ggplot(elbow, aes(x = X2.max_k, y = wss)) +
  geom_point() +
  geom_line() +
  scale_x_continuous(breaks = seq(1, 20, by = 1))
```



Based on the above elbow, we can select K-Means clustering between 3-7 centers.

- Trying with 5 centers

```
p_cluster_2 <-kmeans(prod_df, centers = 5, nstart = 25)

p_cluster_2$betweenss
```

```
## [1] 2.021397e+13
```

```
# Very large distance the centroids of the clusters in the final partition are from one another.
```

- Loading the required library for Cluster plot

```
library(tidyverse)  # data manipulation
```

```
## Warning: package 'tidyverse' was built under R version 3.5.1
```

```
## -- Attaching packages --------------------------------- tidyverse 1.2.1 --
```

```
## v tibble  1.4.2     v purrr   0.2.5
## v readr   1.1.1     v dplyr   0.7.6
## v tibble  1.4.2     v stringr 1.3.1
```

```
## Warning: package 'purrr' was built under R version 3.5.1
```

```
## Warning: package 'dplyr' was built under R version 3.5.1
```

```
## Warning: package 'stringr' was built under R version 3.5.1
```

```
## -- Conflicts ------------------------------------ tidyverse_conflicts() --
## x dplyr::combine() masks gridExtra::combine()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
```

```
library(cluster)    # clustering algorithms
```

```
## Warning: package 'cluster' was built under R version 3.5.1
```

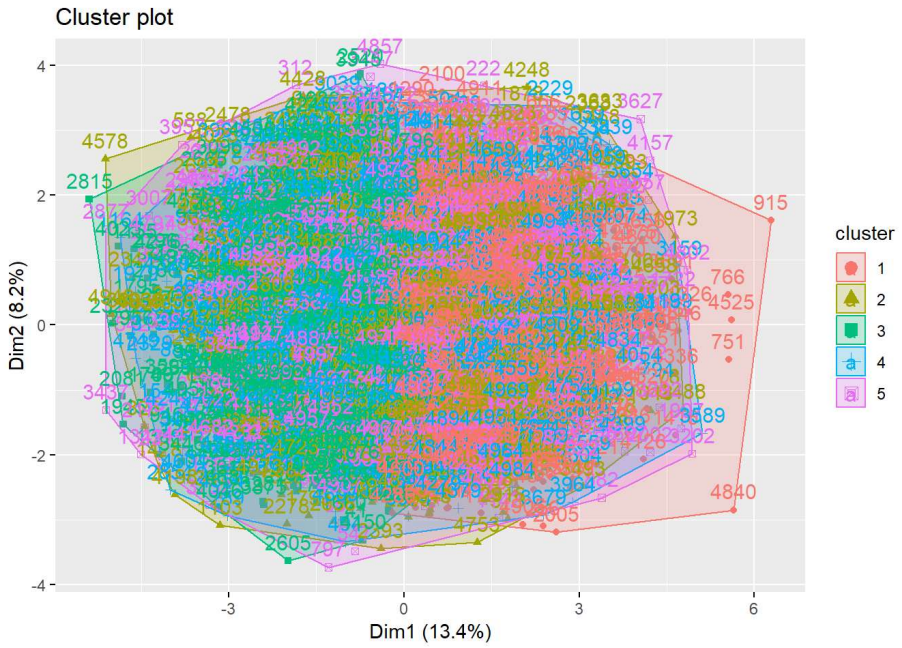
```
library(factoextra) # clustering algorithms & visualization
```

```
## Warning: package 'factoextra' was built under R version 3.5.1
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

- Plotting the clusters: Dim1+Dim2 = 21.6%

```
fviz_cluster(p_cluster_2, data = prod_df)
```



- Trying with 7 centers

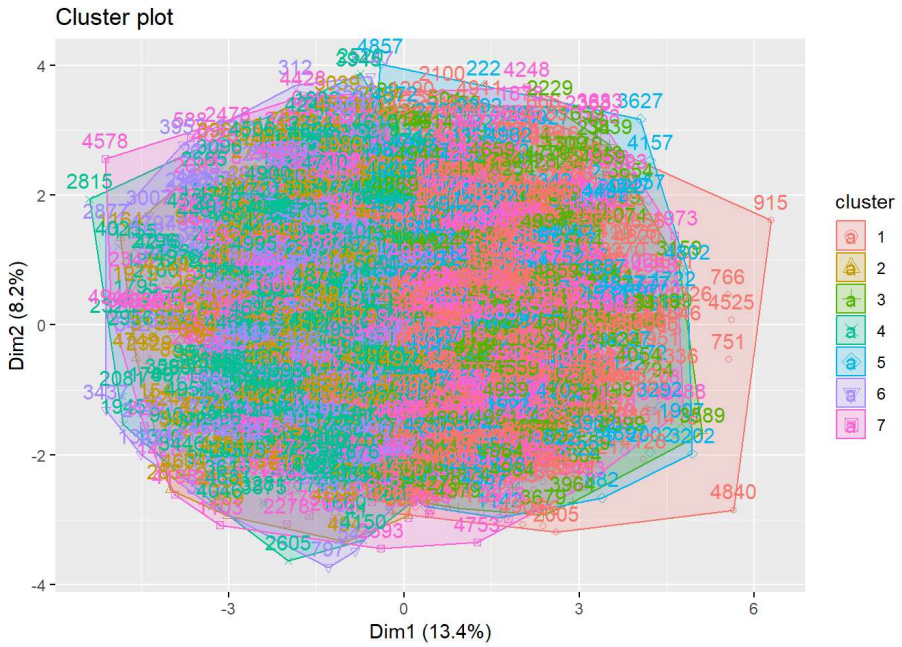
```
p_cluster_2a <-kmeans(prod_df, centers = 7, nstart = 25)

p_cluster_2a$betweenss
```

```
## [1] 2.05277e+13
```

```
# Very large distance the centroids of the clusters in the final partition are from one another.
```

- Plotting the clusters: Dim1+Dim2 = 21.6%

```
fviz_cluster(p_cluster_2a, data = prod_df)
```

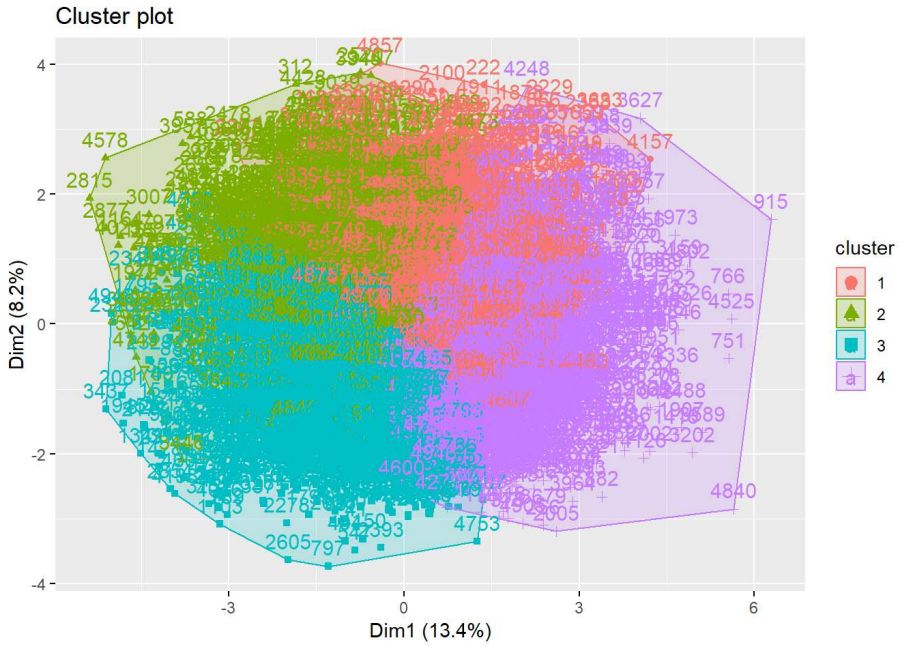


- Lets try to scale the variable and plot the cluster

```
sc_df <- scale(inputdf[, 13:34])

# Plotting with 4 Centriod
sc_cluster <- kmeans(sc_df,4)

sc_cluster$betweenss
```

```
## [1] 16513.48
```

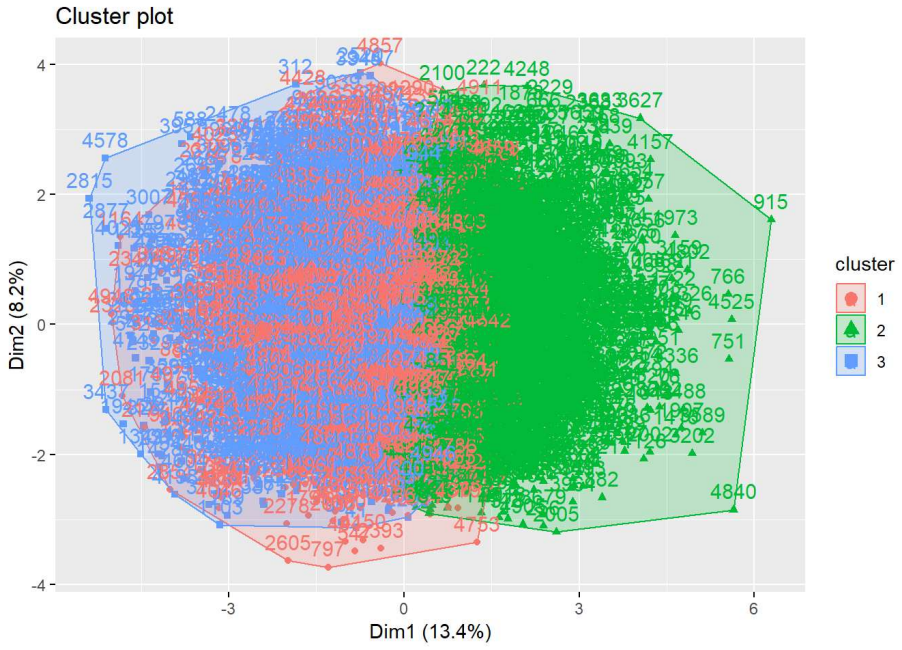
```
fviz_cluster(sc_cluster, data = sc_df)
```




```
# Plotting with 4 Centriod
sc_cluster_2 <-kmeans(sc_df, centers = 3, nstart = 25)

sc_cluster_2$betweenss
```

```
## [1] 13377.24
```

```
fviz_cluster(sc_cluster_2, data = sc_df)
```

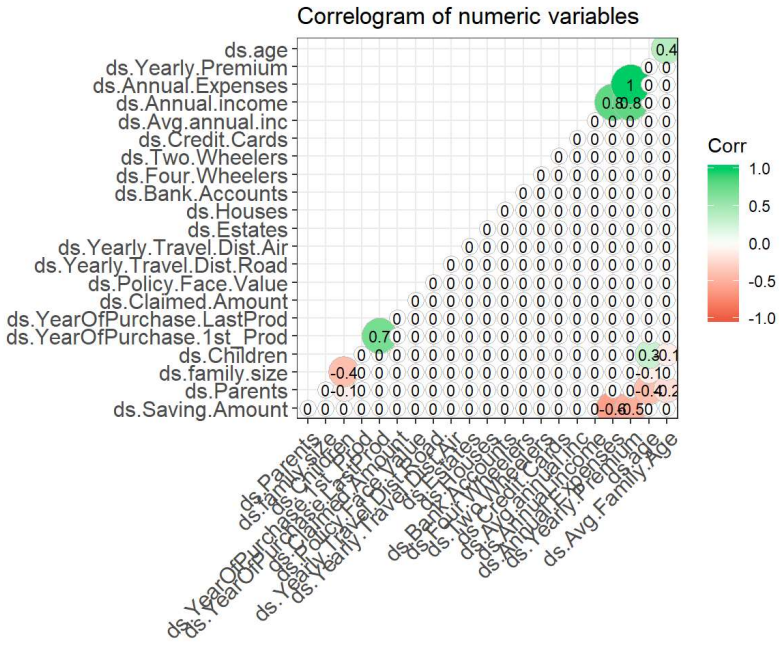


- Loading the required library for Correlogram plot

```
library(ggplot2)
library(ggcorrplot)
```

```
## Warning: package 'ggcorrplot' was built under R version 3.5.1
```

```
# Correlation matrix
corr <- round(cor(prod_df), 1)

# Plotting Correlogram:
ggcorrplot(corr, hc.order = TRUE,
           type = "lower",
           lab = TRUE,
           lab_size = 3,
           method="circle",
           colors = c("tomato2", "white", "springgreen3"),
           title="Correlogram of numeric variables",
           ggtheme=theme_bw)
```



- ds.YearOfPurchase.1st_Prod(21) ds.YearOfPurchase.LastProd(22) - Having high correlation, can select 1 of the 2
- ds.Annual.Expenses(8) ds.Yearly.Premium(20) ds.Annual.income(6) - Having high correlation, can select 1 of the 3

```
## Warning: package 'Hmisc' was built under R version 3.5.1
```

```
## Loading required package: lattice
```

```
## Loading required package: survival
```

```
## Loading required package: Formula
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:dplyr':
##
##     src, summarize
```

```
## The following object is masked from 'package:bnlearn':
##
##     impute
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
## Warning: package 'REdaS' was built under R version 3.5.1
```

```
## Loading required package: grid
```

```
## Warning: package 'ppcor' was built under R version 3.5.1
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
## Warning: package 'psych' was built under R version 3.5.1
```

```
##
## Attaching package: 'psych'
```

```
## The following object is masked from 'package:Hmisc':
##
##     describe
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
##
## Attaching package: 'psy'
```

```
## The following object is masked from 'package:psych':
##
##     wkappa
```

```
colnames(prod_df)
```

```
##  [1] "ds.age"                     "ds.Parents"
##  [3] "ds.family.size"             "ds.Children"
##  [5] "ds.Avg.Family.Age"          "ds.Annual.income"
##  [7] "ds.Avg.annual.inc"          "ds.Annual.Expenses"
##  [9] "ds.Saving.Amount"           "ds.Credit.Cards"
## [11] "ds.Two.Wheelers"            "ds.Four.Wheelers"
## [13] "ds.Bank.Accounts"           "ds.Houses"
## [15] "ds.Estates"                 "ds.Yearly.Travel.Dist.Air"
## [17] "ds.Yearly.Travel.Dist.Road" "ds.Policy.Face.Value"
## [19] "ds.Claimed.Amount"          "ds.Yearly.Premium"
## [21] "ds.YearOfPurchase.1st_Prod" "ds.YearOfPurchase.LastProd"
```

Principal Component Analysis

Identify variables with more variance and perform clustering

```
pca_out2 <- principal(sc_df[,c(1:22)],nf=4,rotate='varimax')
```

```
## Warning in cor.smooth(r): Matrix was not positive definite, smoothing was
## done
```

```
## Warning in principal(sc_df[, c(1:22)], nf = 4, rotate = "varimax"): The
## matrix is not positive semi-definite, scores found from Structure loadings
```

```
varimax_pca2 <- pca_out2$loadings[1:22,]
apply(varimax_pca2,1,function(x) sum(x*x)) #Communalities
```

```
##                     ds.age                  ds.Parents
##               0.7535796220               0.4356761230
##             ds.family.size                ds.Children
##               0.6463390569               0.7467574309
##          ds.Avg.Family.Age           ds.Annual.income
##               0.5444752709               0.6628705370
##          ds.Avg.annual.inc         ds.Annual.Expenses
##               0.0005076003               0.9893295570
##           ds.Saving.Amount            ds.Credit.Cards
##               0.3223337904               0.0015849894
##            ds.Two.Wheelers           ds.Four.Wheelers
##               0.0052052213               0.0043099667
##           ds.Bank.Accounts                ds.Houses
##               0.0073282915               0.0041448463
##                 ds.Estates  ds.Yearly.Travel.Dist.Air
##               0.0051015949               0.0015131125
## ds.Yearly.Travel.Dist.Road       ds.Policy.Face.Value
##               0.0016868569               0.0056870930
##          ds.Claimed.Amount          ds.Yearly.Premium
##               0.0033330169               0.9574014823
## ds.YearOfPurchase.1st_Prod ds.YearOfPurchase.LastProd
##               0.8658974293               0.8668579951
```

```
varimax_pca2 <- ifelse(abs(varimax_pca2)>0.4,varimax_pca2,NA)  #disregarding loadings where abs <0.4
varimax_pca2 <- as.data.frame(round(varimax_pca2,3))
varimax_pca2[order(-varimax_pca2$RC1,-varimax_pca2$RC2,-varimax_pca2$RC3,-varimax_pca2$RC4),]
```

```
##                               RC1   RC3    RC2    RC4
## ds.Annual.Expenses          0.992    NA     NA     NA
## ds.Yearly.Premium           0.976    NA     NA     NA
## ds.Annual.income            0.811    NA     NA     NA
## ds.Saving.Amount           -0.567    NA     NA     NA
## ds.age                         NA    NA  0.845     NA
## ds.Avg.Family.Age              NA    NA  0.718     NA
## ds.Parents                     NA    NA -0.659     NA
## ds.YearOfPurchase.LastProd     NA 0.930     NA     NA
## ds.YearOfPurchase.1st_Prod     NA 0.929     NA     NA
## ds.Children                    NA    NA     NA  0.844
## ds.family.size                 NA    NA     NA -0.802
## ds.Avg.annual.inc              NA    NA     NA     NA
## ds.Credit.Cards                NA    NA     NA     NA
## ds.Two.Wheelers                NA    NA     NA     NA
## ds.Four.Wheelers               NA    NA     NA     NA
## ds.Bank.Accounts               NA    NA     NA     NA
## ds.Houses                      NA    NA     NA     NA
## ds.Estates                     NA    NA     NA     NA
## ds.Yearly.Travel.Dist.Air      NA    NA     NA     NA
## ds.Yearly.Travel.Dist.Road     NA    NA     NA     NA
## ds.Policy.Face.Value           NA    NA     NA     NA
## ds.Claimed.Amount              NA    NA     NA     NA
```

```
colnames(sc_df)
```

```
##  [1] "ds.age"                     "ds.Parents"
##  [3] "ds.family.size"             "ds.Children"
##  [5] "ds.Avg.Family.Age"          "ds.Annual.income"
##  [7] "ds.Avg.annual.inc"          "ds.Annual.Expenses"
##  [9] "ds.Saving.Amount"           "ds.Credit.Cards"
## [11] "ds.Two.Wheelers"            "ds.Four.Wheelers"
## [13] "ds.Bank.Accounts"           "ds.Houses"
## [15] "ds.Estates"                 "ds.Yearly.Travel.Dist.Air"
## [17] "ds.Yearly.Travel.Dist.Road" "ds.Policy.Face.Value"
## [19] "ds.Claimed.Amount"          "ds.Yearly.Premium"
## [21] "ds.YearOfPurchase.1st_Prod" "ds.YearOfPurchase.LastProd"
```

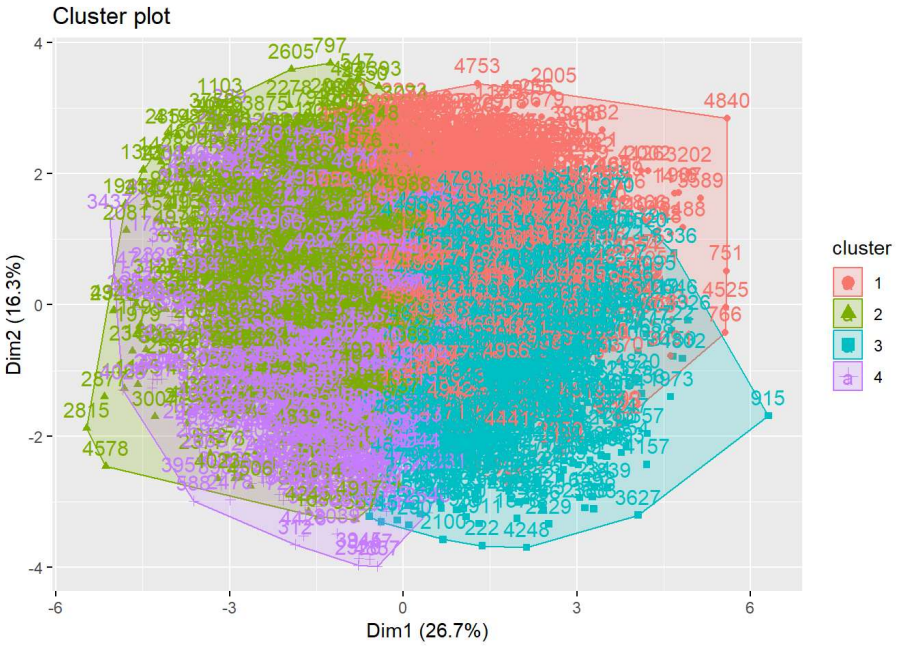
- Clustering - Selecting all the variable with high variance in the RC1-4. But randomly select centriod = 4.

```
sc_df0 <- sc_df[,c(1,2,3,4,5,6,8,9,20,21,22)]

sc_cluster0 <- kmeans(sc_df0,4)

sc_cluster0$betweenss
```

```
## [1] 16172.15
```

- With centriod = 4. Improvement Achieved Dim1+Dim2 = 43%

```
fviz_cluster(sc_cluster0, data = sc_df0)
```



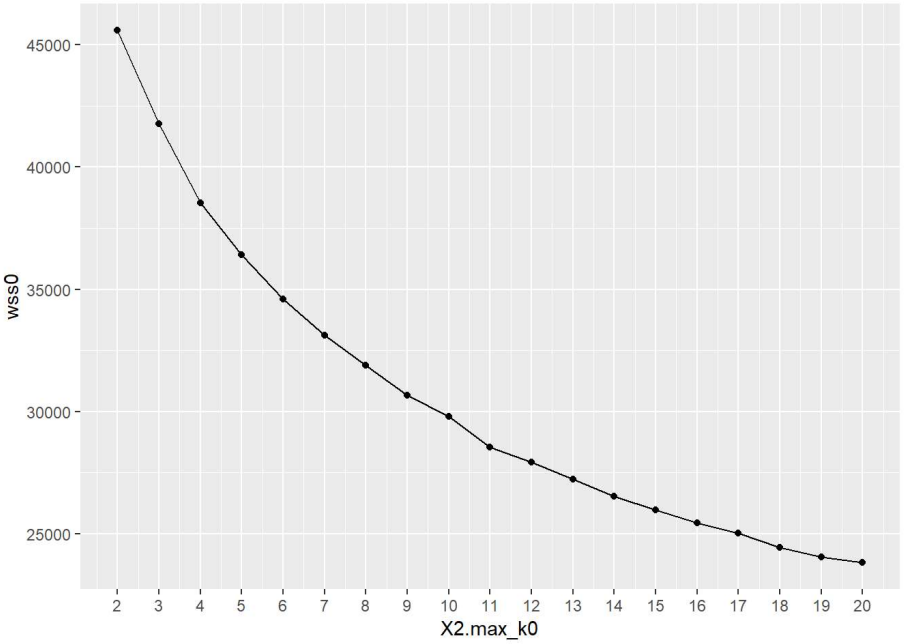
- Lets get the elbow for better estimate

```
kmean_withinss <- function(k) {
  p_cluster0 <- kmeans(sc_df0, k)
  return (p_cluster0$tot.withinss)
}

# Set maximum cluster
max_k0 <-20
# Run algorithm over a range of k
wss0 <- sapply(2:max_k0, kmean_withinss)
```

```
## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations
```

```
# Create a data frame to plot the graph
elbow0 <-data.frame(2:max_k0, wss0)
```

```
library(ggplot2)

# Plot the graph with gglop
ggplot(elbow0, aes(x = X2.max_k0, y = wss0)) +
  geom_point() +
  geom_line() +
  scale_x_continuous(breaks = seq(1, 20, by = 1))
```



- With centriod = 7.

```
sc_cluster_0 <-kmeans(sc_df0, centers = 7, nstart = 25)
```

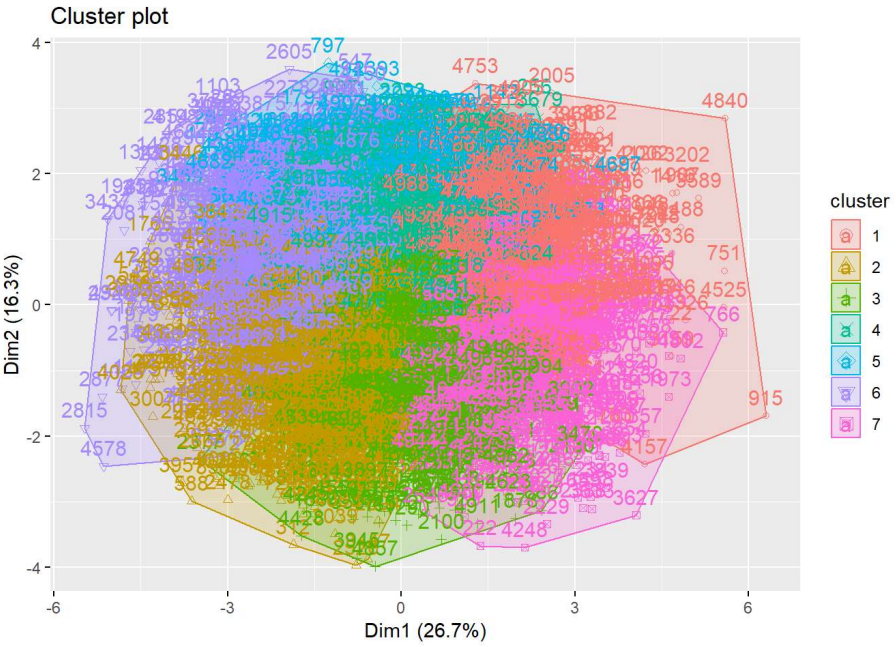
```
## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations
```

```
sc_cluster_0$betweenss
```

```
## [1] 22014.15
```

- Improvement Achieved Dim1+Dim2 = 43%

```
fviz_cluster(sc_cluster_0, data = sc_df0)
```



- With centriod = 4

We are selecting 4 parameters with maximum variance from the 4 RCs

```
sc_df1 <- sc_df[,c(1,4,8,21)]

sc_cluster1 <- kmeans(sc_df1,4) # Randomly putting 4

sc_cluster1$size
```

```
## [1] 1464  513 1472 1551
```

- Proportion of customers on each clusters

```
sc_cluster1$size/5000*100
```

```
## [1] 29.28 10.26 29.44 31.02
```

- Distance between SS

```
sc_cluster1$betweenss
```

```
## [1] 8524.487
```

- Improvement Achieved Dim1+Dim2 = 58.6%

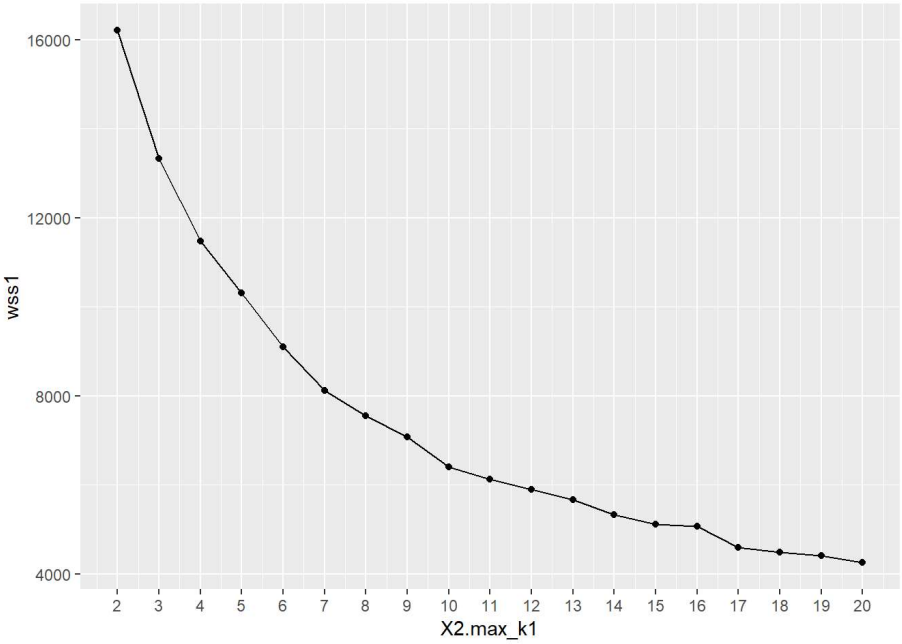
```
fviz_cluster(sc_cluster1, data = sc_df1)
```

- Lets get the elbow for better estimate

```
kmean_withinss <- function(k) {
  p_cluster1 <- kmeans(sc_df1, k)
  return (p_cluster1$tot.withinss)
}

# Set maximum cluster
max_k1 <-20
# Run algorithm over a range of k
wss1 <- sapply(2:max_k1, kmean_withinss)
```

```
## Warning: did not converge in 10 iterations
```

```
# Create a data frame to plot the graph
elbow <-data.frame(2:max_k1, wss1)


library(ggplot2)

# Plot the graph with gglop
ggplot(elbow, aes(x = X2.max_k1, y = wss1)) +
  geom_point() +
  geom_line() +
  scale_x_continuous(breaks = seq(1, 20, by = 1))
```



- With centriod = 5

```
sc_cluster_3 <-kmeans(sc_df1, centers = 5, nstart = 25)
```

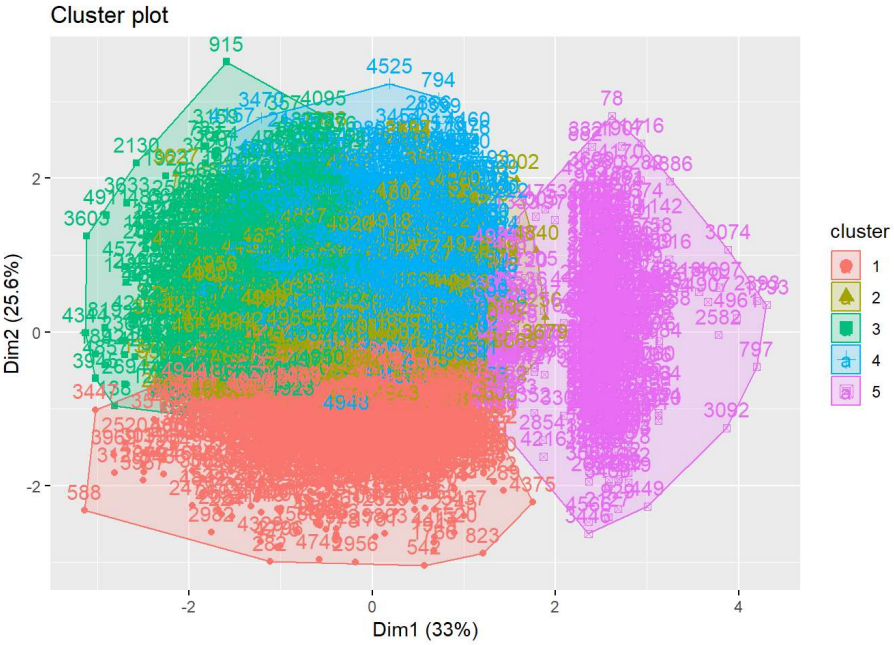
```
## Warning: did not converge in 10 iterations
```

- Distance between SS

```
sc_cluster_3$betweenss
```

```
## [1] 9880.96
```

```
fviz_cluster(sc_cluster_3, data = sc_df1)
```



- With centriod = 3

```
sc_cluster_4 <-kmeans(sc_df1, centers = 3, nstart = 25)

sc_cluster_4$size
```

```
## [1] 1793 1464 1743
```

- Distance between SS

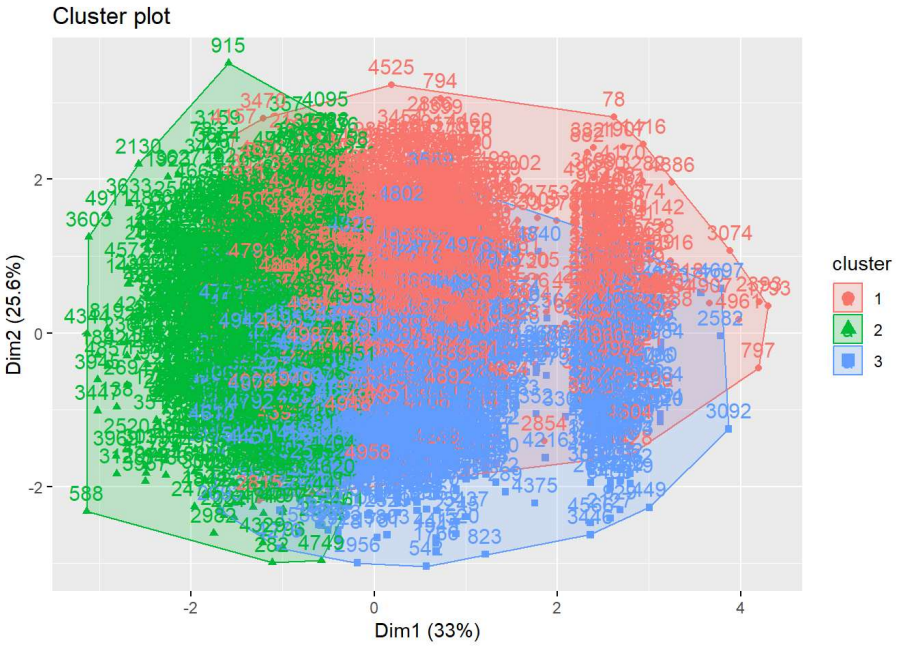
```
sc_cluster_4$betweenss
```

```
## [1] 6703.373
```

```
sc_cluster_4$size/5000*100
```

```
## [1] 35.86 29.28 34.86
```

```
fviz_cluster(sc_cluster_4, data = sc_df1)
```




```
sc_cluster_5 <-kmeans(sc_df1, centers = 2, nstart = 25)

sc_cluster_5$size
```

```
## [1] 1464 3536
```

- Proportion of customers on each clusters Not that uniformly distributed. So, lets proceed with 3 clusters.

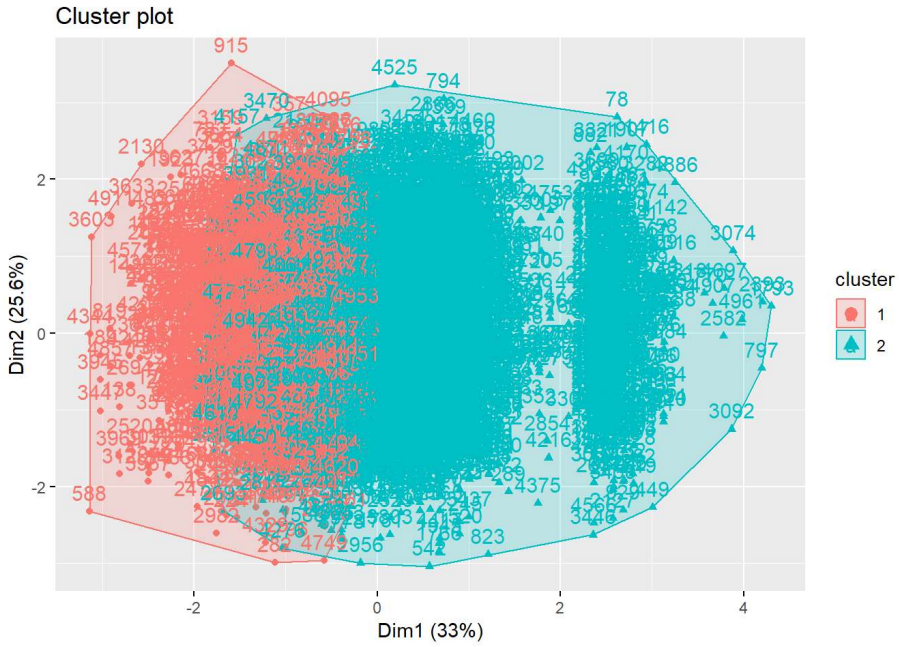
```
sc_cluster_5$size/5000*100
```

```
## [1] 29.28 70.72
```

```
sc_cluster_5$betweenss
```

```
## [1] 4061.059
```

```
fviz_cluster(sc_cluster_5, data = sc_df1)
```



Based on all the clusters with different numbers of centriods. Here, my recommendation shall be 3-4 based on the data.

- I am proceeding my analysis with 3 clusters

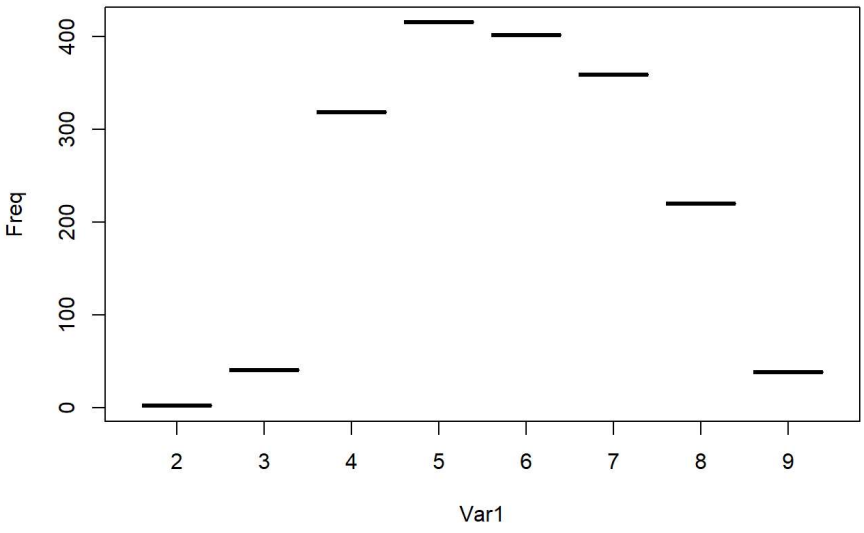
```
sc_df_final3 <- data.frame(inputdf[,1:44],sc_cluster_4$cluster)
```

Grouping customer based on the number of the product they bought

```
library(dplyr)
cluster3_1 = sc_df_final3 %>%
      filter(sc_df_final3$sc_cluster_4.cluster == "1")

ftable(apply(cluster3_1[,35:44], 1, sum))
```

```
##    2   3   4   5   6   7   8   9
##
##    2  40 318 415 401 359 220  38
```

```
x1 = ftable(apply(cluster3_1[,35:44], 1, sum))
x1 = as.data.frame(x1)
plot(x1, type = "h")
```

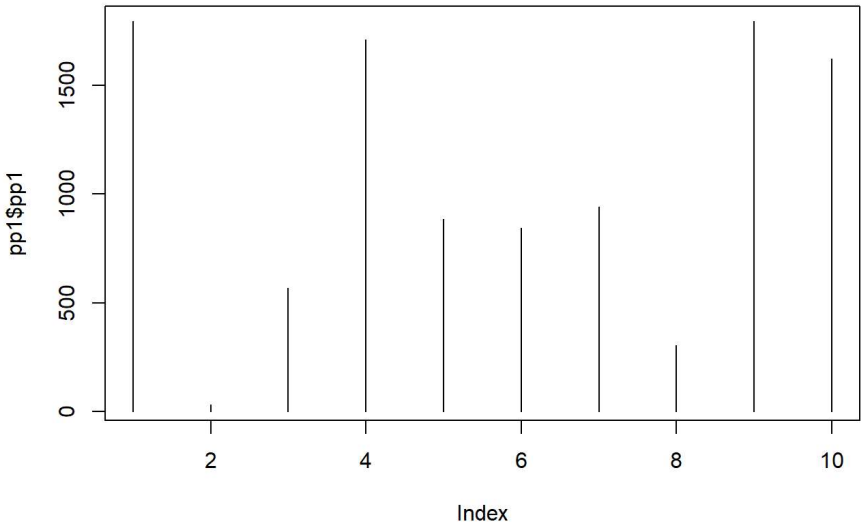


From the above plot we can understand that there are lot of customer who has bought 5-7 products. So, there is a good tendency to buy 5-7 products. So, we can identify customer who has bought 3-4 products and try to promote

some popular products to the customer.

Identifying the popular products within the cluster, having better market share.

```
pp1 = apply(cluster3_1[,35:44], 2, sum)

pp1 = as.data.frame(pp1)
plot(pp1$pp1, type = "h")
```

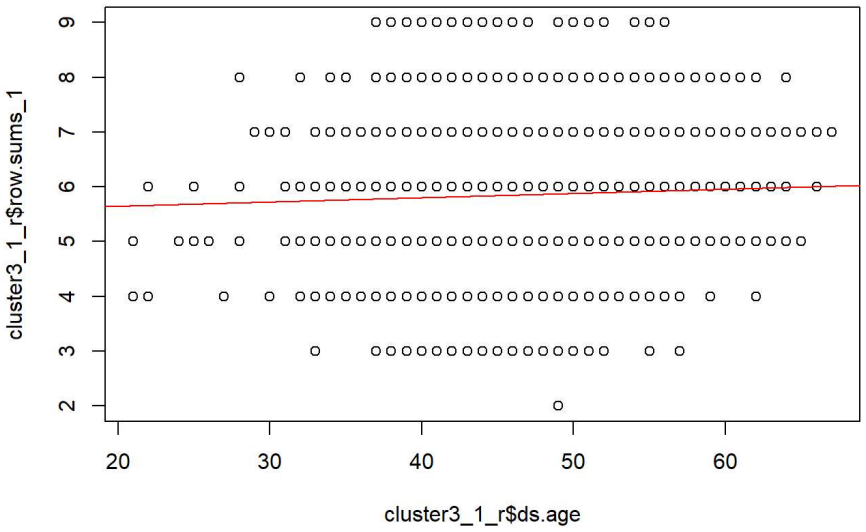


Checking if we can identify any trend of buying product based on the customer age

Cluster 1 - Having customers with more age compare to the Cluster 2 & 3

Customers using 4 products are generally aged below 55. So, those who are in the age band 55-60 can buy some more products.

```
cluster3_1_r = cbind(cluster3_1, row.sums_1 = apply(cluster3_1[,35:44], 1, sum))

fit_clus3_1 = lm(row.sums_1 ~ ds.age,data = cluster3_1_r)

plot.ts(cluster3_1_r$ds.age,cluster3_1_r$row.sums_1)
  abline(fit_clus3_1,col = "red")
```



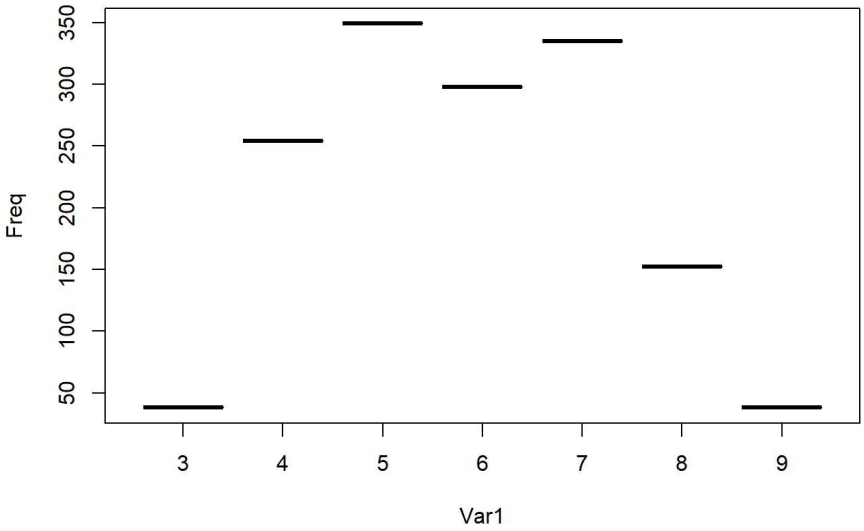
Grouping customer based on the number of the product they bought

```
cluster3_2 = sc_df_final3 %>%
         filter(sc_df_final3$sc_cluster_4.cluster == "2")
ftable(apply(cluster3_2[,35:44], 1, sum))
```

```
##    3   4   5   6   7   8   9
##
##   38 254 349 298 335 152  38
```

Nos. of customers buying nos. of products

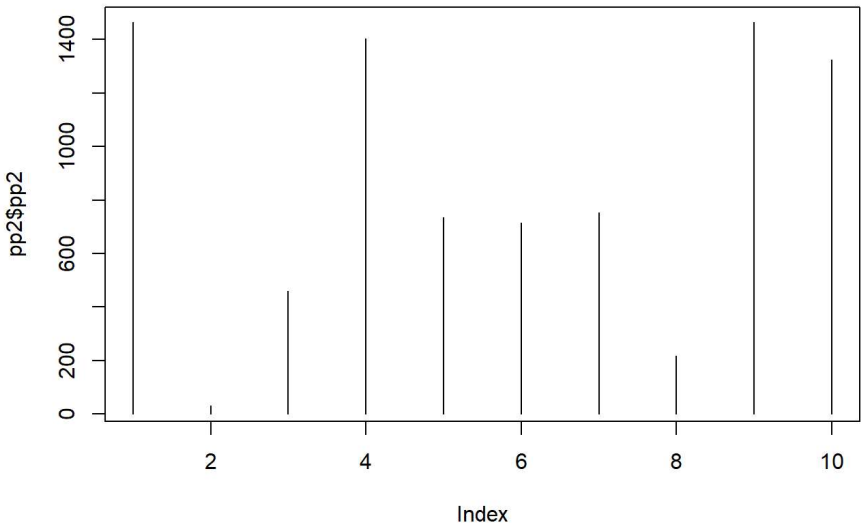
```
x2 = ftable(apply(cluster3_2[,35:44], 1, sum))
x2 = as.data.frame(x2)
plot(x2, type = "h", title = "Frequency distribution of customers with nos of products")
```



From the above plot we can understand that there are lot of customer who has bought 5-7 products. So, there is a good tendency to buy 5-7 products. So, we can identify customer who has bought 3-5 products and try to promote some popular products to the customer.

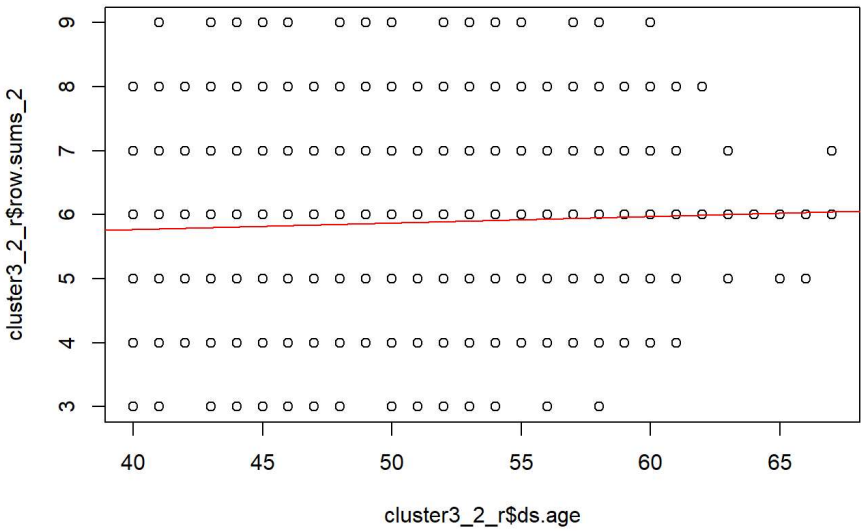
Identifying the popular products within the cluster

```
pp2 = apply(cluster3_2[,35:44], 2, sum)

pp2 = as.data.frame(pp2)
plot(pp2$pp2, type = "h")
```



Checking if we can identify any trend of buying product based on the customer age

```
cluster3_2_r = cbind(cluster3_2, row.sums_2 = apply(cluster3_2[,35:44], 1, sum))

fit_clus3_2 = lm(row.sums_2 ~ ds.age,data = cluster3_2_r)

plot.ts(cluster3_2_r$ds.age,cluster3_2_r$row.sums_2)
  abline(fit_clus3_2,col = "red")
```



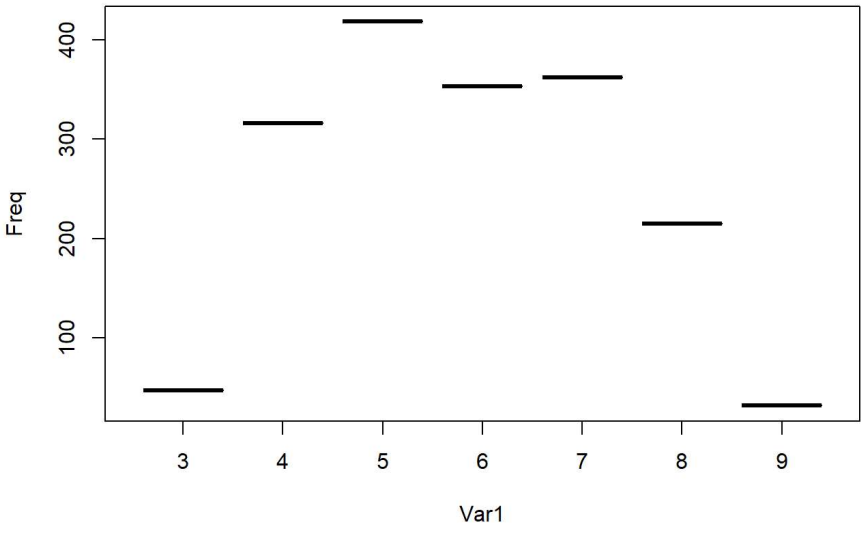
Grouping customer based on the number of the product they bought

```
cluster3_3 = sc_df_final3 %>%
         filter(sc_df_final3$sc_cluster_4.cluster == "3")

ftable(apply(cluster3_3[,35:44], 1, sum))
```

```
##    3   4   5   6   7   8   9
##
##   47 316 418 353 362 215  32
```

Nos. of customers buying nos. of products

```
x3 = ftable(apply(cluster3_3[,35:44], 1, sum))
x3 = as.data.frame(x3)
plot(x3, type = "h")
```

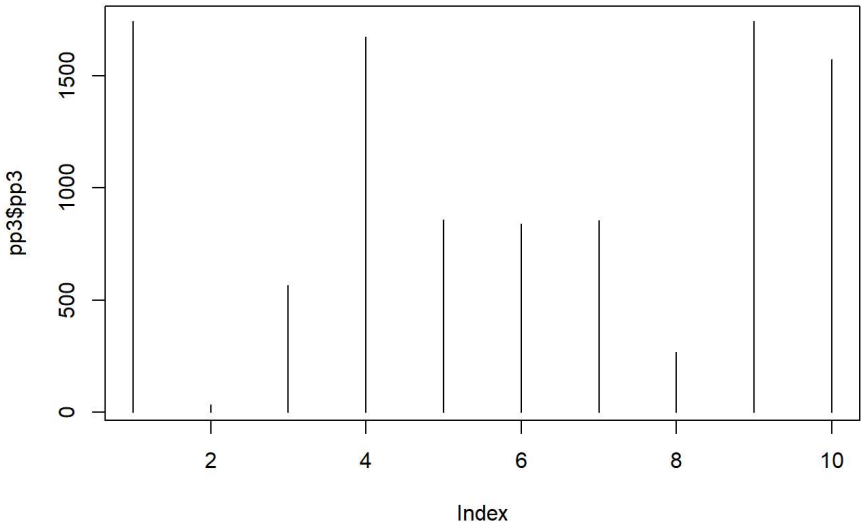


From the above plot we can understand that there are lot of customer who has bought 5-7 products. So, there is a good tendency to buy 5-7 products. So, we can identify customer who has bought 3-5 products and try to promote

some popular products to the customer.

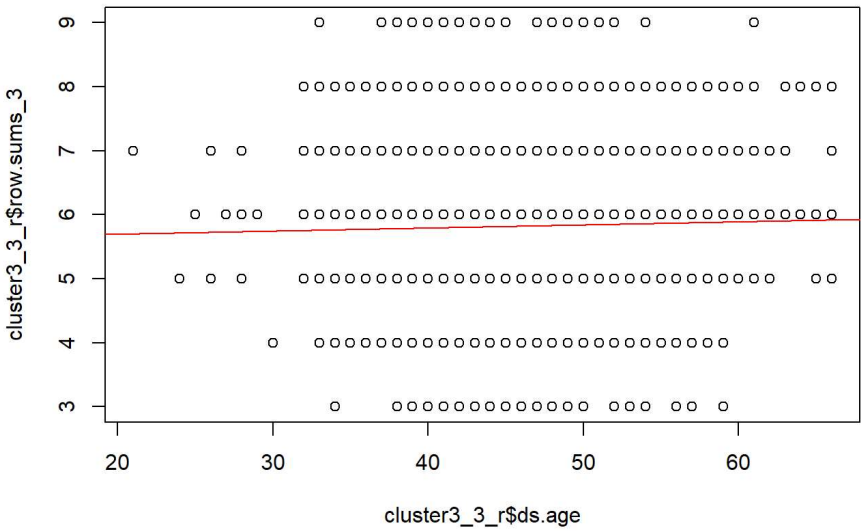
Identifying the popular products within the cluster

```
pp3 = apply(cluster3_3[,35:44], 2, sum)

pp3 = as.data.frame(pp3)
plot(pp3$pp3, type = "h")
```



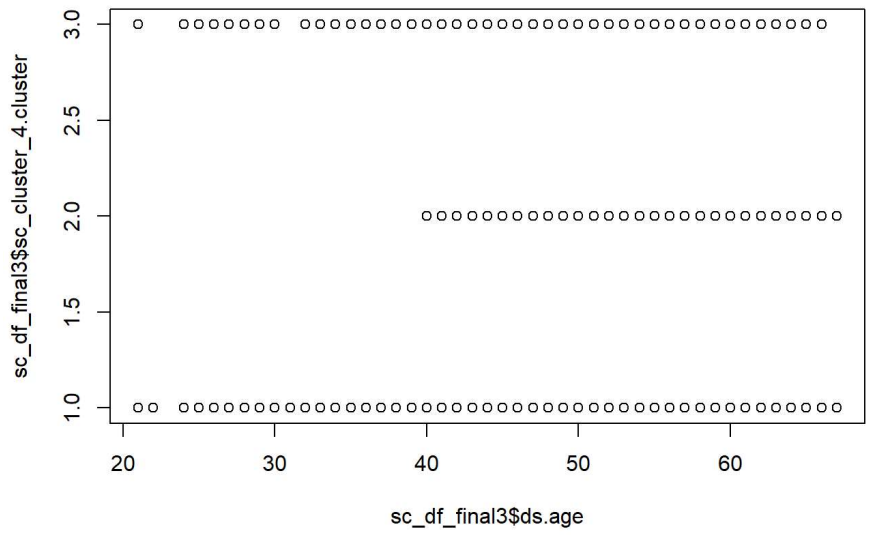
Checking if we can identify any trend of buying product based on the customer age

```
cluster3_3_r = cbind(cluster3_3, row.sums_3 = apply(cluster3_3[,35:44], 1, sum))

fit_clus3_3 = lm(row.sums_3 ~ ds.age,data = cluster3_3_r)

plot.ts(cluster3_3_r$ds.age,cluster3_3_r$row.sums_3)
  abline(fit_clus3_3,col = "red")
```



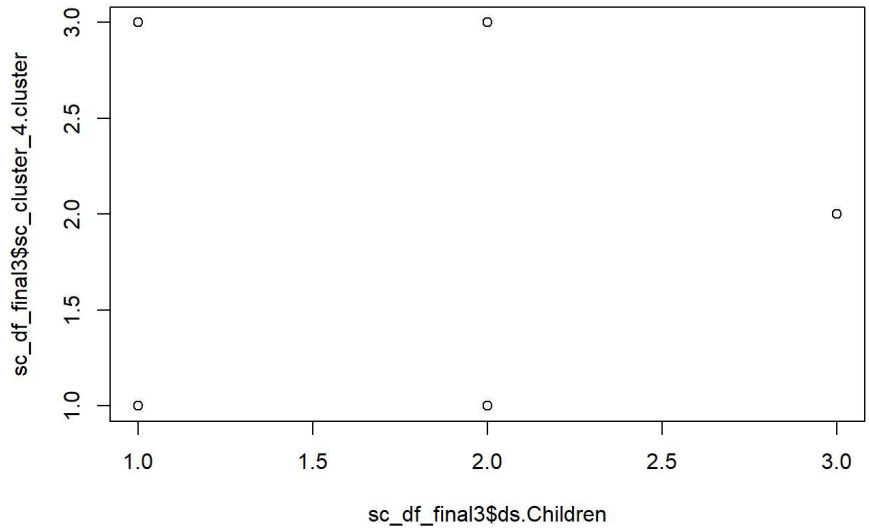
Cluster with age

```
plot.ts(sc_df_final3$ds.age,sc_df_final3$sc_cluster_4.cluster)
```



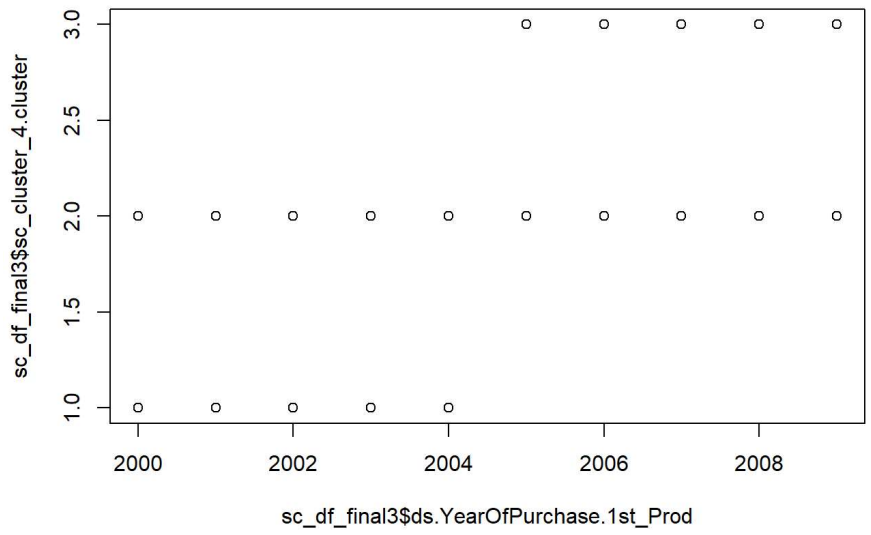
Cluster 1 is having customer with children more than 2

```
plot.ts(sc_df_final3$ds.Children,sc_df_final3$sc_cluster_4.cluster)
```



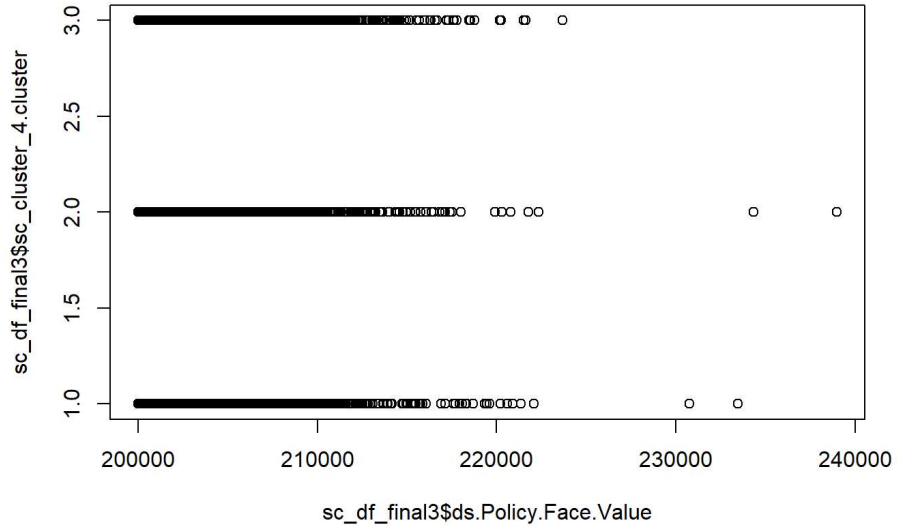
Customer belongs to Cluster 3 has bought 1st Product very recently. Customer belongs to Cluster 2 has bought 1st Product long back. Customer belongs to Cluster 1 has bought 1st Product for longer span.

```
plot(sc_df_final3$ds.YearOfPurchase.1st_Prod,sc_df_final3$sc_cluster_4.cluster)
```



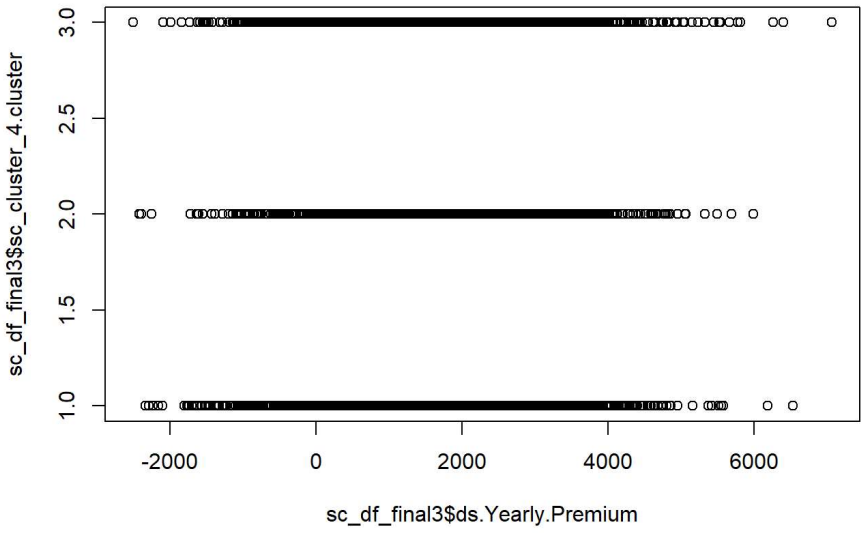
Checking if Policy.Face.Value is having relation with cluster. No relation.

```
plot(sc_df_final3$ds.Policy.Face.Value,sc_df_final3$sc_cluster_4.cluster)
```



Checking Yearly.Premium if any relation with clusters. No differenciating relations.

```
plot(sc_df_final3$ds.Yearly.Premium,sc_df_final3$sc_cluster_4.cluster)
```



Let us use Apriori algorithm, to check if the products are having any rules, and based on it if we can find out some hidden secrets, that is not visible to us till now.

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following object is masked from 'package:tidyr':
##
##     expand
```

```
##
## Attaching package: 'arules'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following object is masked from 'package:bnlearn':
##
##     discretize
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

Converting to transactions to come up with rules. Let us assume that each customer and their product selection is part of a transaction.

```
for (i in 35:44){
  sc_df_final3[,i] = as.factor(sc_df_final3[,i])
}

prod_transaction = as(sc_df_final3[,35:44], "transactions")
apriori(data = prod_transaction, parameter = NULL)
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5     0.1      1
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 500
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[18 item(s), 5000 transaction(s)] done [0.01s].
## sorting and recoding items ... [15 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 5 6 7 8 9 10 done [0.00s].
## writing ... [11148 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```
## set of 11148 rules
```

```
rules <- apriori(prod_transaction,
    parameter = list(supp = 0.5, conf = 0.9, target = "rules"))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.9    0.1    1 none FALSE            TRUE       5     0.5      1
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 2500
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[18 item(s), 5000 transaction(s)] done [0.01s].
## sorting and recoding items ... [10 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 done [0.00s].
## writing ... [356 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
summary(rules)
```

```
## set of 356 rules
##
## rule length distribution (lhs + rhs):sizes
##   1   2   3   4   5   6
##   5  41 102 119  71  18
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   3.000   4.000   3.742   4.250   6.000
##
## summary of quality measures:
##     support         confidence          lift            count
##  Min.   :0.5000   Min.   :0.9009   Min.   :0.9823   Min.   :2500
##  1st Qu.:0.5340   1st Qu.:0.9567   1st Qu.:0.9992   1st Qu.:2670
##  Median :0.6506   Median :0.9830   Median :1.0000   Median :3253
##  Mean   :0.6825   Mean   :0.9731   Mean   :1.0465   Mean   :3413
##  3rd Qu.:0.8232   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:4116
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.9315   Max.   :5000
##
## mining info:
##              data ntransactions support confidence
##  prod_transaction          5000     0.5        0.9
```

Inspecting top 20 rules.

```
inspect(rules[1:20])
```

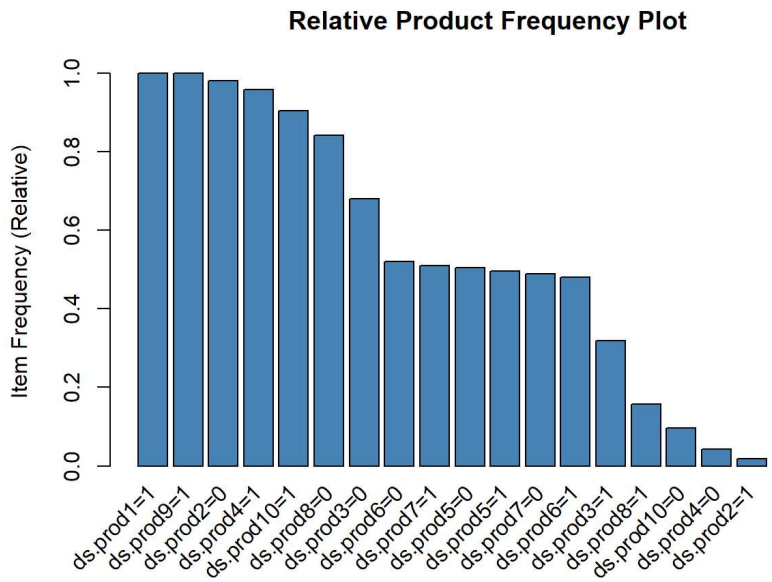
```
##      lhs             rhs           support confidence lift      count
## [1]  {}           => {ds.prod10=1} 0.9042  0.9042000  1.0000000 4521
## [2]  {}           => {ds.prod4=1}  0.9574  0.9574000  1.0000000 4787
## [3]  {}           => {ds.prod2=0}  0.9812  0.9812000  1.0000000 4906
## [4]  {}           => {ds.prod1=1}  1.0000  1.0000000  1.0000000 5000
## [5]  {}           => {ds.prod9=1}  1.0000  1.0000000  1.0000000 5000
## [6]  {ds.prod5=0} => {ds.prod6=0}  0.5000  0.9908839  1.9048133 2500
## [7]  {ds.prod6=0} => {ds.prod5=0}  0.5000  0.9611688  1.9048133 2500
## [8]  {ds.prod5=0} => {ds.prod8=0}  0.5022  0.9952438  1.1819997 2511
## [9]  {ds.prod5=0} => {ds.prod1=1}  0.5046  1.0000000  1.0000000 2523
## [10] {ds.prod5=0} => {ds.prod9=1}  0.5046  1.0000000  1.0000000 2523
## [11] {ds.prod7=1} => {ds.prod1=1}  0.5100  1.0000000  1.0000000 2550
## [12] {ds.prod7=1} => {ds.prod9=1}  0.5100  1.0000000  1.0000000 2550
## [13] {ds.prod6=0} => {ds.prod8=0}  0.5130  0.9861592  1.1712104 2565
## [14] {ds.prod6=0} => {ds.prod2=0}  0.5014  0.9638601  0.9823278 2507
## [15] {ds.prod6=0} => {ds.prod1=1}  0.5202  1.0000000  1.0000000 2601
## [16] {ds.prod6=0} => {ds.prod9=1}  0.5202  1.0000000  1.0000000 2601
## [17] {ds.prod3=0} => {ds.prod10=1} 0.6140  0.9016153  0.9971414 3070
## [18] {ds.prod3=0} => {ds.prod4=1}  0.6506  0.9553598  0.9978690 3253
## [19] {ds.prod3=0} => {ds.prod2=0}  0.6694  0.9829662  1.0018001 3347
## [20] {ds.prod3=0} => {ds.prod1=1}  0.6810  1.0000000  1.0000000 3405
```

Library for graphical representation

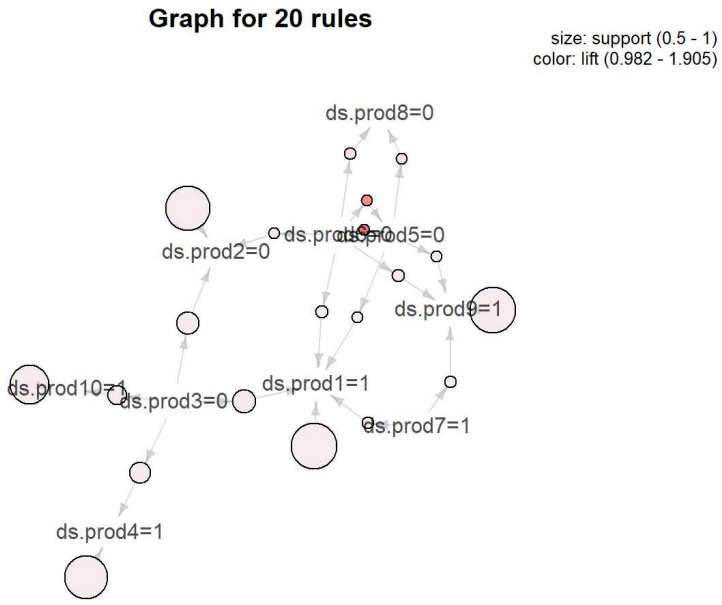
```
library(arulesViz)
```

Checking relative frequency distribution

```
arules::itemFrequencyPlot(prod_transaction,topN=20,col="steelblue",main='Relative Product Frequency Plot',type="relative",ylab="Item Frequency (Relative)")
```




```
plot(rules[1:20],method = "graph")
```



Conlcusion:

Based on the analysis it seems that, we can consider 3 clusters for equal distribution of population amoung the 3 cluster for business operation. Age, Children are some of the more distinguishing factors. We can see that till a certain age customer keep buying insurance products and count of products they avail can reached to even 9 products. But high tendency on the customers to buy product at the range of 6-8. So based on the recommendation, we can suggest mid age customer age 30-45 to buy products if they are less than 6 products in their portfolio. Products 1, 4, 9 & 10 are the most popular products across all the clusters. So when we refer the dictionary or variable details we see that **Product 1 - Children Plan**, **Product 4 - Bike Insurance Plan**, **Product 9 - Health Insurance Plan**, **Product 10 - Retirement Insurance Plan** are the popular product and it really make sense that across various cluster, these insurance products are very much needed. We can understand that insurance are the product that are purchased for long term goal and they are maintain. In case of very senior customer, we see that need or benefit received from insurance are predominating. We also observe that income and expenses are based on the time one is earning and lesser impact due to age. Since, experience might get more or the price might be higher for the aged professionals but younger generation are energatic to earn a similar share. We also see that age between 20-30 has lesser insurance product, seems that at initial phase they are interested to fullfil their dezired and then they feel to secure the future. We also see some products are not that preferred products. It needs further research and Product R&D team need to build new Product so that market share for the product can be increase.

This study will differ on case by case depending on the data but will throw some insight on the society and direction to the business to gain profit to recommend products where there might be some need in near future.

Note: Since as part of this project, we perform algorithm like K Means Clustering, Apriori and Bayesian network; I have not split data into test and train.