```
#High value customers identification for an E-Commerce company

#Installing necessary packages
install.packages("plyr")
install.packages("ggplot2")
install.packages("scales")
install.packages("NbClust")

# reading installed packages and atttaching it to project
library(dplyr)
library(ggplot2)
library(NbClust)
library(scales)

# reading data
ecom_data <- read.csv("Ecommerce.csv",header = T)

# data exploration
class(ecom_data)
View(ecom_data)
str(ecom_data)
summary(ecom_data)
head(ecom_data)
dim(ecom_data) # 541909  9

# Removing redundant column X
ecom_data_subset <- subset(ecom_data, select = -X)
View(ecom_data_subset)
str(ecom_data_subset)

# Checking for missing values
length(unique(ecom_data_subset$CustomerID)) # 4373
sum(is.na(ecom_data_subset$CustomerID)) # 135080
# ecom_data_subset <- subset(ecom_data_subset, is.na(data$CustomerID))
# ecom_data_subset <- subset(ecom_data_subset, Country == "United Kingdom")

mean(is.na(ecom_data_subset)) # 0.02769633 only 2.76 % data are having missing values so we
can ignore it.

pMiss <- function(x)
{
  sum(is.na(x))/length(x)*100
}
apply(ecom_data_subset,2,pMiss)
# apply(ecom_data_subset,1,pMiss)


# The mice package provides a nice function md.pattern() to get a better understanding
# of the pattern of missing data
library(mice)
md.pattern(ecom_data_subset)

# In order to visualize missing values
# install.packages("VIM",dependencies = T)
library(VIM)
aggr_plot <- aggr(ecom_data_subset, col=c('navyblue','red'), numbers=TRUE, sortVars=TRUE,
                  labels=names(ecom_data_subset), cex.axis=.7, gap=3,
                  ylab=c("Histogram of missing data","Pattern"))
```

```
# Let's see the number of unique invoices and unique customers.
length(unique(ecom_data_subset$InvoiceNo))
length(unique(ecom_data_subset$CustomerID))
# We now have a dataset of 23,494 unique invoices and 3,951 unique customers.

# Remove Quantity with negative values
pos_quant <- ecom_data_subset[ecom_data_subset$Quantity > 0,]
nrow(pos_quant) # 5,31,285

# changing date format
ecom_data_subset$InvoiceDate <- as.Date(ecom_data_subset$InvoiceDate,format = "%d-%b-%y")
#23-Nov-17
str(ecom_data_subset$InvoiceDate)

ecom_data_subset$InvoiceNo <- as.integer(ecom_data_subset$InvoiceNo)

# Add the column - amount_spent
ecom_data_subset['amount_spent'] = ecom_data_subset['Quantity'] *
ecom_data_subset['UnitPrice']

#Customer clusters vary by geography.
#So here we'll restrict the data to one geographic unit.
table(ecom_data_subset$Country)

#Let's see the number of unique invoices and unique customers.
length(unique(ecom_data_subset$InvoiceNo))
length(unique(ecom_data_subset$CustomerID))
#We now have a dataset of 25,900 unique invoices and 4,373 unique customers.

#To calculate the recency whic is no of days elapsed since customer last order
# and frequency refers to the no of invoices with purchases during the year variables
below,
# it will be necessary to distinguish invoices with purchases from invoices with returns.
#Identify returns
ecom_data_subset$item.return <- grepl("C", ecom_data_subset$InvoiceNo, fixed=TRUE)
ecom_data_subset$purchase.invoice <- ifelse(ecom_data_subset$item.return=="TRUE", 0, 1)
View(ecom_data_subset)


# Creating Customer Level Dataset

customers <- as.data.frame(unique(ecom_data_subset$CustomerID))
names(customers) <- "CustomerID"


# Recency #


# Adding a recency column by substracting the InvoiceDate from the (last InvoiceDate+1)
ecom_data_subset$recency <- as.Date("2017-12-08") - as.Date(ecom_data_subset$InvoiceDate)

# remove returns so only consider the data of most recent "purchase"
temp <- subset(ecom_data_subset, purchase.invoice == 1)

# Obtain no of days since most recent purchase
recency <- aggregate(recency ~ CustomerID, data=temp, FUN=min, na.rm=TRUE)
remove(temp)

# Add recency to customer data
```

```
customers <- merge(customers, recency, by="CustomerID", all=TRUE, sort=TRUE)
remove(recency)

customers$recency <- as.numeric(customers$recency)


# Frequency

customer.invoices <- subset(ecom_data_subset, select = c("CustomerID","InvoiceNo",
"purchase.invoice"))
customer.invoices <- customer.invoices[!duplicated(customer.invoices), ]
customer.invoices <- customer.invoices[order(customer.invoices$CustomerID),]
row.names(customer.invoices) <- NULL

# Number of invoices/year (purchases only)
annual.invoices <- aggregate(purchase.invoice ~ CustomerID, data=customer.invoices,
FUN=sum, na.rm=TRUE)
names(annual.invoices)[names(annual.invoices)=="purchase.invoice"] <- "frequency"

# Add # of invoices to customers data
customers <- merge(customers, annual.invoices, by="CustomerID", all=TRUE, sort=TRUE)
remove(customer.invoices, annual.invoices)

range(customers$frequency)
table(customers$frequency)

# Remove customers who have not made any purchases in the past year
customers <- subset(customers, frequency > 0)



# Monetary Value of Customers


# Total spent on each item on an invoice
# data$Amount <- data$Quantity * data$UnitPrice

# Aggregated total sales to customer
total.sales <- aggregate(amount_spent ~ CustomerID, data=ecom_data_subset, FUN=sum,
na.rm=TRUE)
names(total.sales)[names(total.sales)=="amount_spent"] <- "monetary"

# Add monetary value to customers dataset
customers <- merge(customers, total.sales, by="CustomerID", all.x=TRUE, sort=TRUE)
remove(total.sales)

# Identify customers with negative monetary value numbers, as they were presumably
# returning purchases from the preceding year
hist(customers$monetary)
customers$monetary <- ifelse(customers$monetary < 0, 0, customers$monetary) # reset
negative numbers to zero
hist(customers$monetary)



# Pareto Principle: 80/20 Rule


customers <- customers[order(-customers$monetary),]
high.cutoff <- 0.8 * sum(customers$monetary)
```

```
customers$high <- ifelse(cumsum(customers$monetary) <= high.cutoff, "Top 20%", "Bottom
80%")
customers$high <- factor(customers$high, levels=c("Top 20%", "Bottom 80%"), ordered=TRUE)
levels(customers$high)
round(prop.table(table(customers$high)), 2)

customers <- customers[order(customers$CustomerID),]


# Preprocess data


# Log-transform positively-skewed variables
customers$recency.log <- log(customers$recency)
customers$frequency.log <- log(customers$frequency)
customers$monetary.log <- customers$monetary + 0.1 # can't take log(0), so add a small
value to remove zeros
customers$monetary.log <- log(customers$monetary.log)

# Z-scores
customers$recency.z <- scale(customers$recency.log, center=TRUE, scale=TRUE)
customers$frequency.z <- scale(customers$frequency.log, center=TRUE, scale=TRUE)
customers$monetary.z <- scale(customers$monetary.log, center=TRUE, scale=TRUE)

View(customers)


# Visualize data


library(ggplot2)
library(scales)

# Original scale
scatter.1 <- ggplot(customers, aes(x = frequency, y = monetary))
scatter.1 <- scatter.1 + geom_point(aes(colour = recency, shape = pareto))
scatter.1 <- scatter.1 + scale_shape_manual(name = "80/20 Designation", values=c(17, 16))
scatter.1 <- scatter.1 + scale_colour_gradient(name="Recency\n(Days since Last Purchase))")
scatter.1 <- scatter.1 + scale_y_continuous(label=dollar)
scatter.1 <- scatter.1 + xlab("Frequency (Number of Purchases)")
scatter.1 <- scatter.1 + ylab("Monetary Value of Customer (Annual Sales)")
scatter.1

#This first graph uses the variables’ original metrics and is almost completely
uninterpretable.
#There’s a clump of data points in the lower left-hand corner of the plot, and then a few
outliers.
#This is why we log-transformed the input variables.

# Log-transformed
scatter.2 <- ggplot(customers, aes(x = frequency.log, y = monetary.log))
scatter.2 <- scatter.2 + geom_point(aes(colour = recency.log, shape = pareto))
scatter.2 <- scatter.2 + scale_shape_manual(name = "80/20 Designation", values=c(17, 16))
scatter.2 <- scatter.2 + scale_colour_gradient(name="Log-transformed Recency")
scatter.2 <- scatter.2 + xlab("Log-transformed Frequency")
scatter.2 <- scatter.2 + ylab("Log-transformed Monetary Value of Customer")
scatter.2


# Scaled variables
```

```
scatter.3 <- ggplot(customers, aes(x = frequency.z, y = monetary.z))
scatter.3 <- scatter.3 + geom_point(aes(colour = recency.z, shape = pareto))
scatter.3 <- scatter.3 + scale_shape_manual(name = "80/20 Designation", values=c(17, 16))
scatter.3 <- scatter.3 + scale_colour_gradient(name="Z-scored Recency")
scatter.3 <- scatter.3 + xlab("Z-scored Frequency")
scatter.3 <- scatter.3 + ylab("Z-scored Monetary Value of Customer")
scatter.3

##################################################
# Determining number of clusters through K-Means #
##################################################

preprocessed <- customers[,9:11]
clustmax <- 10 # specify the maximum number of clusters you want to try out

models <- data.frame(k=integer(),
                     tot.withinss=numeric(),
                     betweenss=numeric(),
                     totss=numeric(),
                     rsquared=numeric())

for (k in 1:clustmax )
{

  print(k)

  # Run kmeans
  # nstart = number of initial configurations; the best one is used
  # $iter will return the iteration used for the final model
  output <- kmeans(preprocessed, centers = k, nstart = 20)

  # Add cluster membership to customers dataset
  var.name <- paste("cluster", k, sep="_")
  customers[,(var.name)] <- output$cluster
  customers[,(var.name)] <- factor(customers[,(var.name)], levels = c(1:k))

  # Graph clusters
  cluster_graph <- ggplot(customers, aes(x = frequency.log, y = monetary.log))
  cluster_graph <- cluster_graph + geom_point(aes(colour = customers[,(var.name)]))
  colors <-
c('red','orange','green3','deepskyblue','blue','darkorchid4','violet','pink1','tan3','black')

  cluster_graph <- cluster_graph + scale_colour_manual(name = "Cluster Group",
values=colors)
  cluster_graph <- cluster_graph + xlab("Log-transformed Frequency")
  cluster_graph <- cluster_graph + ylab("Log-transformed Monetary Value of Customer")
  title <- paste("k-means Solution with", k, sep=" ")
  title <- paste(title, "Clusters", sep=" ")
  cluster_graph <- cluster_graph + ggtitle(title)
  print(cluster_graph)

  # Cluster centers in original metrics
  library(plyr)
  print(title)
  cluster_centers <- ddply(customers, .(customers[,(var.name)]), summarize,
                           monetary=round(median(monetary),2),# use median b/c this is the
raw, heavily-skewed data
                           frequency=round(median(frequency),1),
                           recency=round(median(recency), 0))
  names(cluster_centers)[names(cluster_centers)=="customers[, (var.name)]"] <- "Cluster"
```

```
  print(cluster_centers)
  cat("\n")
  cat("\n")

  # Collect model information
  models[k,("k")] <- k
  models[k,("tot.withinss")] <- output$tot.withinss # the sum of all within sum of squares
  models[k,("betweenss")] <- output$betweenss
  models[k,("totss")] <- output$totss # betweenss + tot.withinss
  models[k,("rsquared")] <- round(output$betweenss/output$totss, 3) # percentage of
variance explained by cluster membership
  assign("models", models, envir = .GlobalEnv)

  remove(output, var.name, cluster_graph, cluster_centers, title, colors)

}

remove(k)

# Graph variance explained by number of clusters
r2_graph <- ggplot(models, aes(x = k, y = rsquared))
r2_graph <- r2_graph + geom_point() + geom_line()
r2_graph <- r2_graph + scale_y_continuous(labels = scales::percent)
r2_graph <- r2_graph + scale_x_continuous(breaks = 1:clustmax)
r2_graph <- r2_graph + xlab("k (Number of Clusters)")
r2_graph <- r2_graph + ylab("Variance Explained")
r2_graph

# Graph within sums of squares by number of clusters
# Look for a "bend" in the graph, as with a scree plot
ss_graph <- ggplot(models, aes(x = k, y = tot.withinss))
ss_graph <- ss_graph + geom_point() + geom_line()
ss_graph <- ss_graph + scale_x_continuous(breaks = 1:clustmax)
ss_graph <- ss_graph + scale_y_continuous(labels = scales::comma)
ss_graph <- ss_graph + xlab("k (Number of Clusters)")
ss_graph <- ss_graph + ylab("Total Within SS")
ss_graph



# Using NbClust metrics to determine number of clusters

library(NbClust)
set.seed(1)
nc <- NbClust(preprocessed, min.nc=2, max.nc=7, method="kmeans")
table(nc$Best.n[1,])

nc$All.index # estimates for each number of clusters on 26 different metrics of model fit

barplot(table(nc$Best.n[1,]),
        xlab="Number of Clusters", ylab="Number of Criteria",
        main="Number of Clusters Chosen by Criteria")

# remove(preprocessed)
#Hierarchical Cluster######
custmr_data <- read.csv("Ecommerce.csv",header = T) # reading data
custmr_data <- na.omit(custmr_data) # data cleaning

View(custmr_data)
str(custmr_data)
```

```
custer_h <- dist(custmr_data,method = "euclidian") # distance matrix
fit <- hclust(custer_h,method = 'ward')
?hclust

groups <- cutree(fit, k = 3)
groups
custmr_data < cbind(custmr_data,ClusterNum = groups)

plot(fit)
rect.hclust(fit,k=3,border = 'red')
```