Appreciate Data Challenge

Stock Recommendation System Based on Market and Customer Demographics

University of Connecticut

December 9th, 2020

Team 5

Deexith Reddy, Saivalini Durvasula, Jette Hovad

Contents

1. Recommending Stocks to Portfolios based on Market Prices

Recommendations to customers will be based on two different approaches. Firstly, we will use market prices to recommend stocks, which provides the best return given a certain risk level and Sharpe ratio.

Secondly, we will use customer demographics to come up with stock recommendations, discussed in Section-2.

1.2 Data Collection

We used historical data from Yahoo Finance with a timeframe from Jan 1, 2015 to Dec 1st,2020. The time period is long enough to encompass longer economic cycles; therefore, data will not be affected by events like the pandemic, elections or other events, which can affect stock prices short term.

The tickers used to find stock prices can be found in the excel sheet provided by Appreciate. The working directory in R should be updated and this is where the output files with the recommended portfolios will be saved.

```
# Set Working Directory - this is where the output will be finally saved.
setwd("C:\\Users\\XXXXX\\Downloads")
```

The following R packages are required:

```
library(PortfolioAnalytics)
library(openxlsx)
library(ROI)
library(ROI.plugin.glpk)
library(ROI.plugin.quadprog)
library(ROI)
library(tidyquant) # tq_get
library(dplyr) # group_by
library(timetk)
library(forcats)# fct_reorder
library(tidyr) # spread()
```

```
# Read the training data given by the Appreciate Team:
```

(Note: The data files are not read in the order of the files, data1 contains data from sheet-2 of the training data and so on, please use the same variable names to help replicate the code)

```
data1 = read.xlsx("20201022-Training Dataset_v1(1) (3).xlsx",sheet=2)

data2 = read.xlsx("20201022-Training Dataset_v1(1) (3).xlsx",sheet=3)

data3 = read.xlsx("20201022-Training Dataset_v1(1) (3).xlsx",sheet=1)
```

1.3 R code for Data Collection

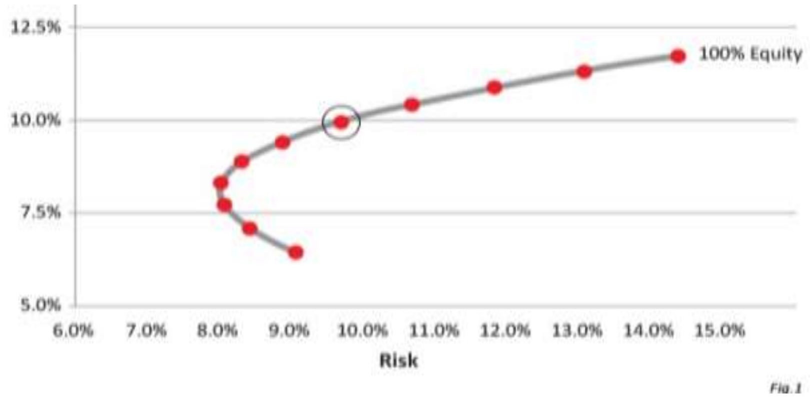
This is the code used to collect stock prices from Yahoo Finance [(1)]. The stock prices are collected for those tickers given in the training data by the Appreciate team and each ticker represents a stock.

```
tickers = c(unique(data1$Ticker))

data <- data.frame(tq_get(tickers,

             from = "2015-01-01",

             to = "2020-12-01",

             get = "stock.prices"))
```

1.4 The Efficient Frontier

The recommendation system will be based on the efficient frontier theory, which is part of modern portfolio theory.

It allows us to find the portfolios with maximum return for a given level of risk. Thereby, we can recommend the stocks that provide the investor with a portfolio, which has the highest return for a given level of risk. See below example of an efficient frontier.



www.proactiveadvisormagazine.com

The investor picks combinations of stocks, which maximizes return for instance for 10% risk, also called volatility, the investor can achieve maximum return 10%

In the following sections parts of our code will be connected to the efficient portfolio theory to explain our logic behind the recommendation system. The full code can be found in the R file shared in the zip folder.

1.5 Calculating the Return of the Stocks

We used daily adjusted prices to calculate the portfolio returns.

Firstly, logarithmic daily returns are calculated for all the tickers downloaded from yahoo finance that are relevant to our training data.

The code is as shown below. [(1)]

```
log_ret_tidy <- data %>%
  group_by(symbol) %>%
  tq_transmute(select = adjusted,
               mutate_fun = periodReturn,
               period = 'daily',
               col_rename = 'ret',
               type = 'log')
```

The data is then converted to the wide format and a time series xts() object subsequently.

```
log_ret_xts <- log_ret_tidy %>%
  spread(symbol, value = ret) %>%
  tk_xts()
```

When working with returns from the stocks, logarithmic mean is recommended because it is time additive. Mean return of the daily stocks is calculated with the code below. The colMeans() will give us the mean of logarithmic daily returns of each stock or ticker.

This is the code for calculating the mean returns:

```
   Cust_1_DR=subset(log_ret_xts, select = c(Cust_1_tick$Ticker, tickers[!tickers %in%
Cust_1_tick$Ticker][i]))
   Cust_1_DR[is.na(Cust_1_DR)] <- 0
   Cust_1_MR = data.frame("Ticker" = colnames(Cust_1_DR),"Returns"=
colMeans(Cust_1_DR,na.rm=TRUE))
```

```
rownames(Cust_1_MR) <- 1:nrow(Cust_1_MR)
```

1.6 Calculating the Covariance

Then we calculate the covariance matrix for all stocks. We will annualize it by multiplying by 252. The lower the covariance between the stocks the more we can lower the risk of the customers portfolio.

Below is the formula for covariance.

$$\text{cov}(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X}) \times (Y_i - \bar{Y})}{n-1}$$

(Bergman, David pp module 4, Statistics in Business Analytics)

This is the code for calculating covariance:

```
Cust_1_cov_mat <- round(cov(Cust_1_DR, use = "complete.obs") * 252,5)
```

The covariance matrix of the stocks is necessary for calculating the risk of the portfolio.

1.7 Calculating the Risk and Return of Existing Customer Portfolios

Now we have found the mean returns and covariance of the individual stocks, to further calculate the portfolio risk and return we also need the weights of the assets in the portfolio.

This is the formula for portfolio return:

$$E(R_P) = W_1 E(R_1) + W_2 E(R_2)$$

(Luehrman Timothy)

. Below is the formula for portfolio risk:

$$\sigma_P^2 = W_1^2 \sigma_1^2 + W_2^2 \sigma_2^2 + 2W_1 W_2 \text{Cov}(R_1, R_2)$$

(Luehrman Timothy)

Where W are the weight of the assets and R the returns.

The recommended weights were found by using an optimization method from the package "PortfolioAnalytics". We selected the optimization method "ROI" as it gave the highest Sharpe Ratio amongst the other optimization algorithms in the package.

The objectives added to the Optimization algorithm are:

1. Maximum Return
2. Minimum Risk

3. Maximize Sharpe Ratio (discussed in the next section)

Subject to constraints:

1. No cash in portfolio (full investment), indicating 100% investment, adding up the weights of the stocks in the portfolio to 1.
2. No short selling (long only), which by default considers the minimum weight to be assigned to any stock as 0, and the maximum weight to be assigned to any stock as 1.

The code below shows the assignment of the above objectives and constraints:

```
init.portf <- portfolio.spec(assets=colnames(Cust_1_DR))

init.portf <- add.constraint(portfolio=init.portf, type="full_investment")

init.portf <- add.constraint(portfolio=init.portf, type="long_only")

init.portf <- add.objective(portfolio=init.portf, type="return", name="mean")

init.portf <- add.objective(portfolio=init.portf, type="risk", name="StdDev")
```

1.8 Sharpe Ratio

The portfolio weights mentioned above will be determined based on maximum Sharpe ratio.

The Sharpe Ratio compares the excess return over the risk-free rate to the risk. It allows us to rank the stocks according to which stock gives the highest return for a given risk level. The risk-free rate is determined to be 3%

$$S_a = \frac{E\left[R_a - R_b\right]}{\sigma_a}$$

Where:

E(Ra) is the expected return of portfolio

E(Rb) is the risk-free rate, which is 3%

Standard deviation of a is the risk of the customer portfolios.

The below code optimizes the weights of the portfolio based on maxSR function which is set to TRUE in the code, and recommends weights that lead to maximum Sharpe Ratios.

```
maxSR <- optimize.portfolio(R=Cust_1_DR, portfolio=init.portf,
                    optimize_method="ROI",
                    Rf = 0.03,
                    maxSR=TRUE, trace=TRUE)
```

1.9 Recommending New Stocks to Existing Customer Portfolios Based on Market

We created a loop, which calculates the new weights and Sharpe ratios for each of the customers' existing portfolio by adding one new stock at a time, that is not present in the customer's Portfolio.

In every iteration, the weights are obtained for the updated portfolio with one stock each, through the optimization algorithm, and these weights are then used to calculate the new portfolio returns and Sharpe Ratio.

Finally, for each customer, the various new portfolios created are sorted based on the Sharpe Ratio, and top stocks that increase the Sharpe ratio along with the rebalanced portfolio weights are returned as the output.

Code can be seen below, but consider to see the R file shared for the complete code.

```
Cust_1_port_ret <- sum(maxSR$weights * Cust_1_MR$Returns)
  Cust_1_port_ret[i] <- ((Cust_1_port_ret + 1)^252) - 1
  Cust_1_port_sd[i] <- sqrt(t(maxSR$weights) %*% (Cust_1_cov_mat  %*% maxSR$weights))
  SR[i] <- (Cust_1_port_ret[i]-0.03)/Cust_1_port_sd[i]
  best <-  data.frame(round(maxSR$weights,10), SR[i], Cust_1_port_ret[i],Cust_1_port_sd[i])
  best1 <- rbind(best1,best)
 }
 best1 = best1[order(best1$SR, decreasing = TRUE),]
 best1 = cbind(rownames(best1), best1, stringsAsFactors = FALSE)
 rownames(best1) = 1:nrow(best1)
 colnames(best1)=c("Ticker","Weights","SharpeRatio","PortfolioReturn","PortfolioRisk")
 A=best1[row.names(best1[seq((length(Cust_1_tick$Ticker)+1), nrow(best1),
(length(Cust_1_tick$Ticker)+1)),][2]>0),]
 colnames(A)=c("Ticker","Weights","SharpeRatio","PortfolioReturn","PortfolioRisk")
 Sug_Tick=head(A[which(A$Weights > 0),])[1]
 Total=best1[best1$SharpeRatio %in% head(A[which(A$Weights > 0),])$SharpeRatio,]
 for (i in 1 : length(unique(Total$SharpeRatio))){
  Check = split(Total, fct_inorder(factor(Total$SharpeRatio),))
  assign(paste0("Portfolio",i),Check[[i]])
 }
 Final_Suggestions <- list()
 Final_Suggestions$Top_Tickers = Sug_Tick[1]
 Final_Suggestions$Portfolio = mget(ls(pattern = "^Portfolio.$"))
 cat(paste0("Top Stock Recommendations for your existing Portfolio are:", Sug_Tick[1]),sep="\n")
 return(Final_Suggestions)
}}
data5 = data1
```

1.10 Storing Data

The code has been created such that the stock recommendations with the rebalanced weights and metrics will be saved to your local working directory, in separate folders for each customer. (In this case, there would be 1000 folders for the 1000 unique customers in the training data given to us).

This process can take up to 4 hours to run for all the 1000 customers.

In the folders in your local working directory, you will find:

1. One sheet named "Rebalanced-Portfolio-Metrics-Market" that comprises the top optimal portfolios with optimized weights and one new stock added. Portfolio1.Ticker represents the portfolio, which resulted in the highest Sharpe ratio. Portfolio2.Ticker the second highest Sharpe ratio and so forth.

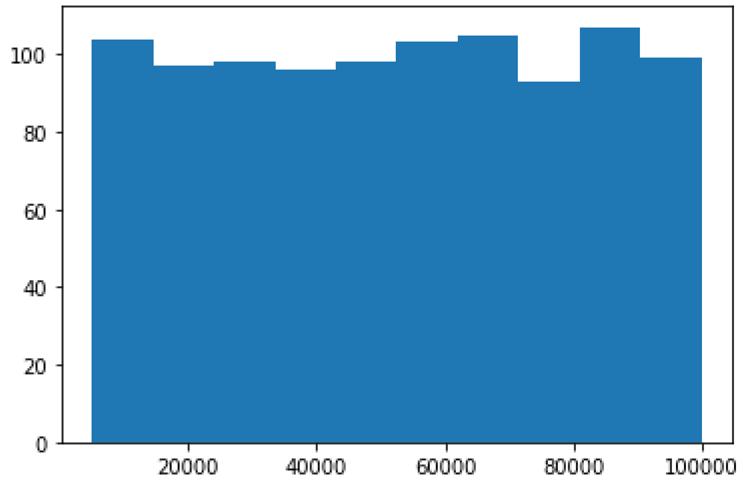
2. One other sheet named "Top-Tickers-Suggested-Market" that consists of the Ticker labels suitable for the particular customers portfolio, sorted in the order of highest Sharpe Ratio to lowest. as discussed above, the recommendations for top new stocks and portfolios will be saved in individual customer folders in your local working directory.

The code for the same is shown below:

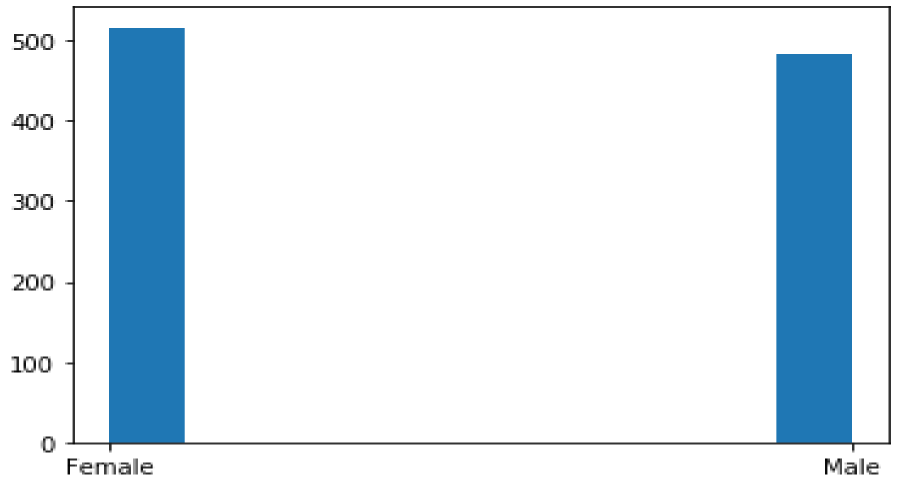
```
for (i in 1: length(unique(data5$Customer_ID))){
  cat(paste0("Customer:", unique(data5$Customer_ID)[i], sep = "\n"))
  cust = new2(data5[data5$Customer_ID==unique(data5$Customer_ID)[i],])
  dir.create(paste0("Customer"," ",unique(data5$Customer_ID)[i]), showWarnings = FALSE)
  write.csv(cust$Top_Tickers, file.path(paste0("Customer"," ",unique(data5$Customer_ID)[i]), "Top-
Tickers-Suggested-Market.csv"), row.names=FALSE)
  write.csv(cust$Portfolio, file.path(paste0("Customer"," ",unique(data5$Customer_ID)[i]), "Rebalanced-
Portfolio-Metrics-Market.csv"), row.names=FALSE)
}
```

2. Recommending Stocks based on Customer Demographics Information

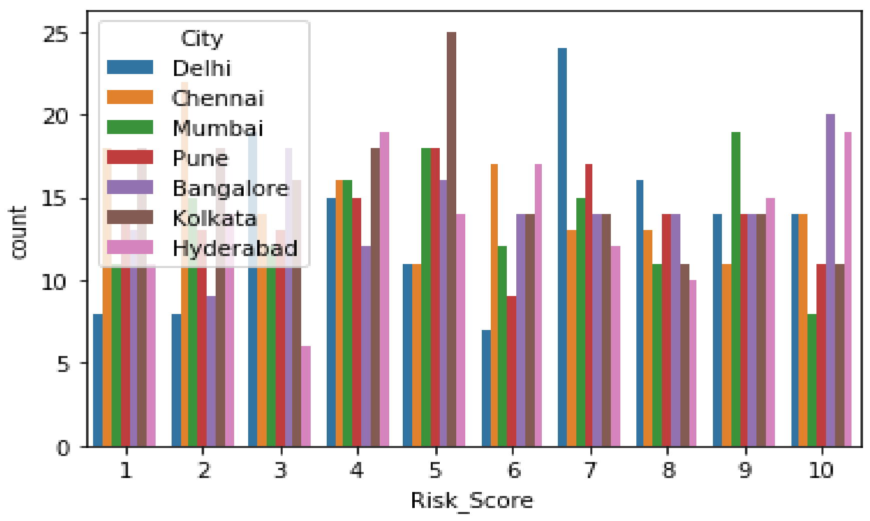
Customer information based on age, income, risk score, gender and city of residence are given. We had seen that the data is well balanced from the histograms shown below.



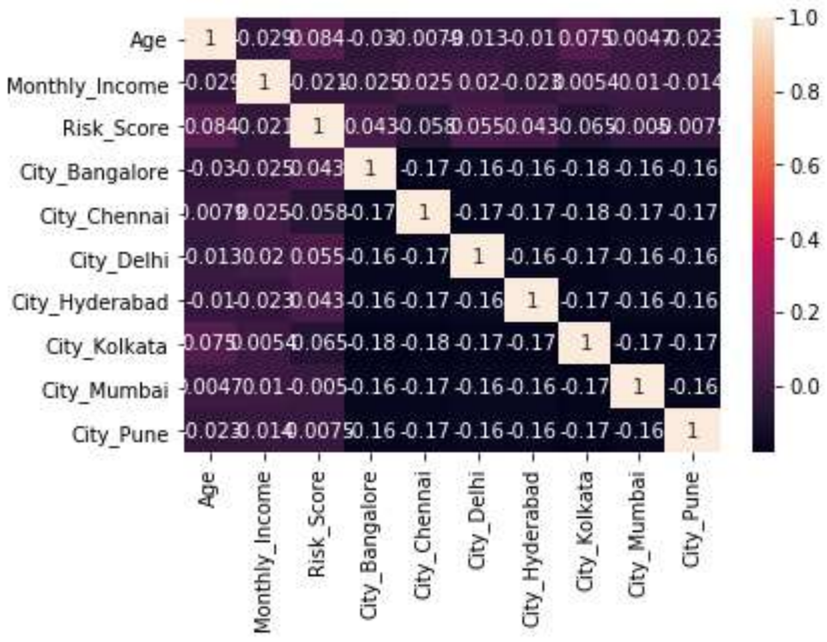
Well Balanced dataset for income on x-axis



Well Balanced dataset for gender on x-axis



City and risk score also well balance



No correlation between variables. We rule out regression and other classification methods

Similarly, age was also well balanced. This shows that there is no need to do sampling as the data is well balanced. Hence, even clustering will lead to many clusters as every variable will have an equal number of observations in each area.

For recommendation, we use USER-USER type of similarity based on the Cosine similarity of users, taking information from the demographics.

COSINE SIMILARITY:

If we represent any information in a multi-dimensional space, we can calculate the distance between those information points. While the Euclidean Distance between those two points might be a good indicator whether they are similar or not, it is sensitive to frequency. When we consider multidimensional space, increase in frequency will lead to growing distance between the two vectors and tricks the formula.

Hence, we use cosine similarity. Here, the cosine of the angle is measured and more the value, closer are the two vectors.

$$\boxed{\cos(\beta) = \frac{\hat{v} \cdot \hat{w}}{||\hat{v}||\,||\hat{w}||}}$$

Required Packages:

```
install.packages('plyr')
library('fastDummies')
library(magrittr)
library(tidyverse)
library(rlist)
library(pipeR)
library(float)
library(coop)
library(mapply)
library(tm)
library(purrr)
library(data.table)
library(plyr)
```

Data preparation:

1) We first prepare dummy variables for the gender and delete the original gender column.

```
data3 <- dummy_cols(data3, select_columns = 'Gender')

data3 = select(data3, -Gender)
```

2) Then we prepare categorical variables for the city by using categorical codes. We write a function for this. We also make customer ID as row names (Index) and delete the original city column,

```
encode_ordinal <- function(x, order = unique(x)) {
            x <- as.numeric(factor(x, levels = order, exclude = NULL))
            x
          }
data3[["city_encoded"]] <- encode_ordinal(data3[["City"]])
data3 = select(data3, -City)
rownames(data3) <- data3$Customer_ID
data3 = select(data3, -Customer_ID)
```

3) We then create a nested list that contains all the information for a customer in form of lists.
   For example Customer 5: [Age,Income,Risk_Score,GenderFemale,GenderMale,CityCode]
   [35, 50000, 4, 1, 0, 5]

```
d<-list()

for (i in 1:1000){
  h<-list()
  h<-list(as.numeric(data3[i,]))
  d<-append(d,h)
}
```

4) We set a list of tickers indexed by customer number. This is the current portfolio of the customer.

```
k<-c()
for (i in 1:1000){
 c<-c()
 c<-c(data1 %>% filter(Customer_ID == i) %>% select(Ticker))
 k<-append(k,c)}
```

Function:

5) Recommender function compared to closest customer:

```
recommender <- function(n){
 c<-c()
 e<-c()
 f<-c()
 for (j in 1:1000){
  if (j!=n) {
   cos_sim=coop::cosine(unlist(d[n]),unlist(d[j]))
   c<-c(c,cos_sim)
   f<-c(f,j)
   e<-c(e,k[j])
  }
 }
 y<-mapply(c,c,e, SIMPLIFY=F)
 y<-y[order(-c)]
 x <- y[1]
 x<-map(x, tail, -1)
 x<-unlist(x)
 return(sample(x,5))


}
```

6) Recommender function for new customers will take in information and calculate cosine similarity with all customers to give the nearest portfolio.

```
recommender1 <- function(p,q,r,s,t,u){



 c<-c()
 e<-c()
 a<-c(p,q,r,s,t,u)
 for (i in 1:1000){
   cos_sim=coop::cosine(a,unlist(d[i]))
```

```
  c<-c(c,cos_sim)
  e<-c(e,k[i])

 }
 y<-mapply(c,c,e, SIMPLIFY=F)
 y<-y[order(-c)]
 x <- y[1]
 x<-map(x, tail, -1)
 x<-unlist(x)
 return(sample(x,5))

}
```

7) Similarly, the customer can choose how many similar customers he wants to check with. We can pass an argument “m”.

```
recommenders <- function(n,m){

  c<-c()
  e<-c()
  f<-c()
  for (j in 1:1000){
    if (j!=4) {
      cos_sim=coop::cosine(unlist(d[n]),unlist(d[j]))
      c<-c(c,cos_sim)
      f<-c(f,j)
      e<-c(e,k[j])
    }
  }
  y<-mapply(c,c,e, SIMPLIFY=F)
  y<-y[order(-c)]
  w<-c()
  for (a in 1:m){
  x <- y[a]
  x<-map(x, tail, -1)
  x<-unlist(x)
  w<-c(w,x)
  }
  w<-unlist(w)
  return(sample(w,5))

}




recommenders1 <- function(p,q,r,s,t,u,m){


  c<-c()
  e<-c()
  a<-c(p,q,r,s,t,u)
  for (i in 1:1000){
    cos_sim=coop::cosine(a,unlist(d[i]))
    c<-c(c,cos_sim)
    e<-c(e,k[i])

  }
  y<-mapply(c,c,e, SIMPLIFY=F)
  y<-y[order(-c)]
  w<-c()
  for (a in 1:m){
    x <- y[a]
    x<-map(x, tail, -1)
    x<-unlist(x)
    w<-c(w,x)
  }
  w<-unlist(w)
  return(sample(w,5))

}
```

Each function will output a sample of five stocks based on the similar customer present in database.

2.1 Storing Data Example

Portfolio preference storage:

Finally, as discussed above, the recommendations for top new stocks and portfolios will be saved in individual customer folders in your local working directory. The code for the same is shown below:

```
for (i in 1: length(unique(data5$Customer_ID))){
  cat(paste0("Customer:", unique(data5$Customer_ID)[i], sep = "\n"))
  cust = new2(data5[data5$Customer_ID==unique(data5$Customer_ID)[i],])
  dir.create(paste0("Customer"," ",unique(data5$Customer_ID)[i]), showWarnings = FALSE)
  write.csv(cust$Top_Tickers, file.path(paste0("Customer"," ",unique(data5$Customer_ID)[i]), "Top-
Tickers-Suggested-Market.csv"), row.names=FALSE)
  write.csv(cust$Portfolio, file.path(paste0("Customer"," ",unique(data5$Customer_ID)[i]), "Rebalanced-
Portfolio-Metrics-Market.csv"), row.names=FALSE)
}
```

Demographic data is stored in the similar folder as above:

Demographic preference storage:

```
for (i in 1:2){
```

```
g<-recommender(i)
write.csv(g, file.path(paste0("Customer"," ",i),
                "Demographic Suggestions.csv"), row.names=FALSE)
}
```

3. References

1. https://www.codingfinance.com/post/2018-05-31-portfolio-opt-in-r/ (Code used for Yahoo finance data extraction + formulas for annualized returns, risk, was used from this source)
2. https://cran.r-project.org/web/packages/PortfolioAnalytics/PortfolioAnalytics.pdf
3. Luehrman, Timothy A. Risk and Return 1: Stock Returns and diversification