Yewno Quant Test

Oded Loewenstein (orl2108)

December 26, 2019

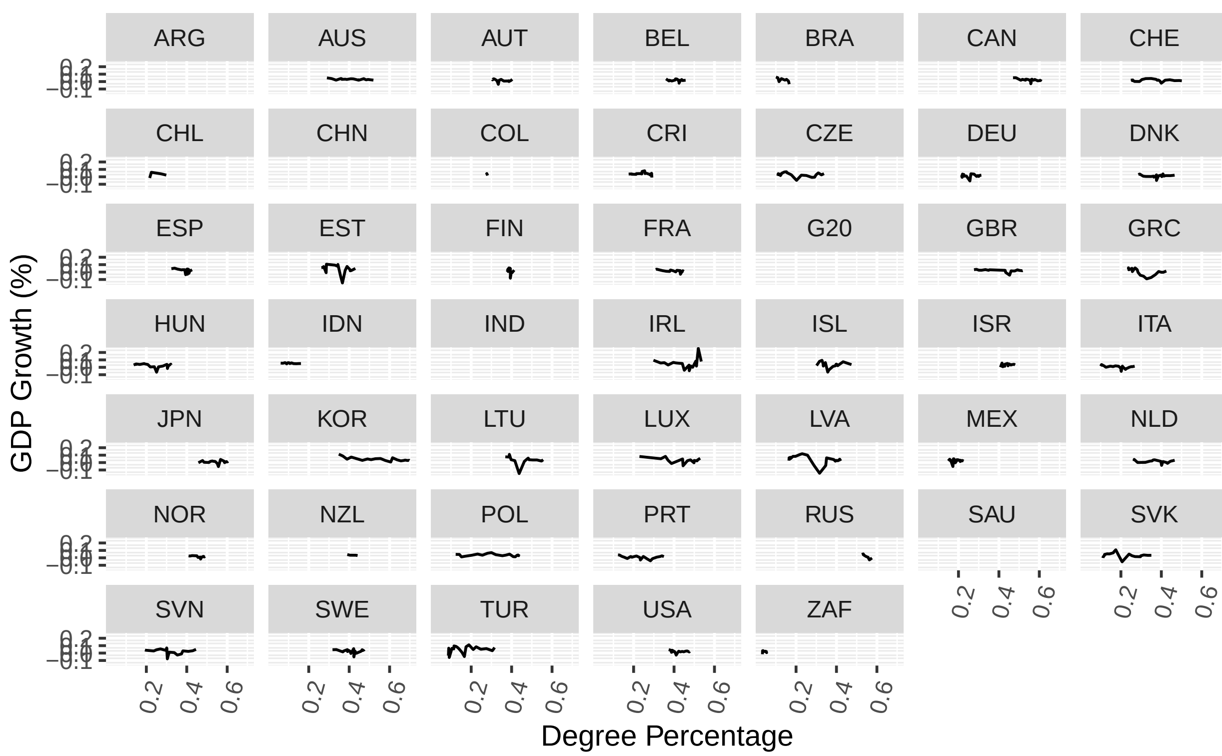
Problem 1

For this problem I needed to use freely available data to predict/explain macroeconomic indicators, with financial and economic data not included. The indicator I choose to explain is GDP, or more accurately, the change in GDP in each year (**GDP_Growth**). I first obtain data regarding the precentage of students within the age of 25-34 who have at least an undergraduate degree in each country (**Deg_Per**). The data for the described variables was obtain from the Unesco Institute For Statistics (http://data.uis.unesco.org/).

Since I feel that there are other variables of a non-financial/economic nature which could further predict the change in GDP, I add another variable that can mimic the economic status, the change in the number of passangers from each country (**Pass_Change**), obtained from the world bank website (https://data.worldbank.org/indicator/IS.AIR.PSGR?locations=AF).

I process the data and obtain a final data set, (data1 under in the data folder) and observe how the relation between the response and the predictor for each country:

```
data <- read.csv("data.csv")
data <- data[-c(which(data$TIME==1998),which(data$Code=="G20")),-c(3:6,9)]
colnames(data) <- c("Code","Country","Time","Deg_Per","GDP_Growth","Passengers","Pass_change")
data$GDP_Growth <- as.numeric(as.character(data$GDP_Growth))/100
data$Deg_Per <- data$Deg_Per/100
data$Passengers <- as.numeric(as.character(data$Passengers))
data$Pass_change <- as.numeric(as.character(data$Pass_change))
```



Clearly, this does not seem like a linear relationship exists, and therefore I should try to model this relationship in a non-linear method. The first method that comes to mind is the decision tree, because it is highly interpretable and does not have heavy underlying assumptions, which we may encounter in this kind of data (time series data due to trend or seasonality, normality, etc.).

I use the decision tree in both it's forms, and try to predict not only the I first try to predict GDP as a classification problem, or in other words, whether the GDP will increase, decrease or stay the same. In order to do so, I construct a new categorical variable with the values -1,0,1 (decrease, no change, increase) called **Indicator**.

```
data$Indicator <- as.factor(ifelse(is.na(data$GDP_Growth),0,ifelse(data$GDP_Growth>0,1,-1)))
```

I split the data into a training (80%) and test (20%) data set.

```
index <- sample(1:dim(data)[1],0.8*dim(data)[1], replace = F)
train <- data[index,]
test <- data[-index,]
```

Now, I can finally start with the modeling. I use 10-fold cross validation to validate which model performs best, testing different values for minimal split.

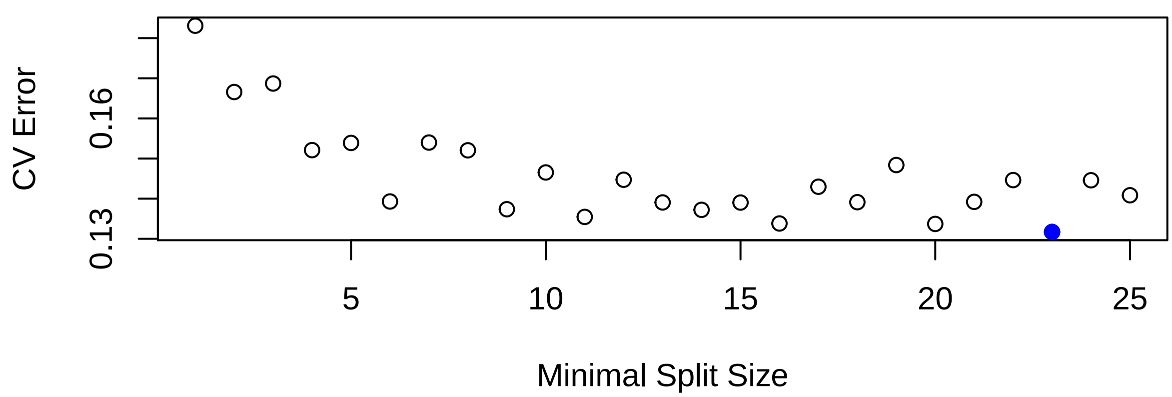
```
cross.tree <- function(K=10,minsplit = 20){
  fold <- createFolds(train$Indicator, k = K)
  validation.missclass <- rep(NA,K)
  for(k in 1:K){
    class.tree <- rpart(Indicator~Deg_Per+Pass_change, data = train[-fold[[k]],], control = rpart.contr
    train.pred <- predict(class.tree, newdata = train[fold[[k]],], type = "class")
    validation.missclass[k] <- mean(train$Indicator[fold[[k]]]!=train.pred)
    CV.error <- mean(validation.missclass)
  }
return(CV.error)
}

CV.mat <- matrix(1:25,25,1)
CV.error <- apply(X = CV.mat,MARGIN = 1,FUN = cross.tree, K=10)
CV.mat <- as.matrix(cbind(CV.mat,CV.error))
colnames(CV.mat) <- c("Minimal_Split","CV_Error")
Best.minsplit <- which.min(CV.mat[,2])
```

Minimal_Split	CV_Error
1	0.1830976
2	0.1665777
3	0.1686953
4	0.1520984
5	0.1538949
6	0.1393001
7	0.1539827
8	0.1520539
9	0.1373737
10	0.1465320
11	0.1354545
12	0.1447138
13	0.1390572
14	0.1372162
15	0.1390332
16	0.1338119
17	0.1429726
18	0.1391246
19	0.1483838
20	0.1337037
21	0.1392015
22	0.1446332
23	0.1317039
24	0.1445887
25	0.1408526

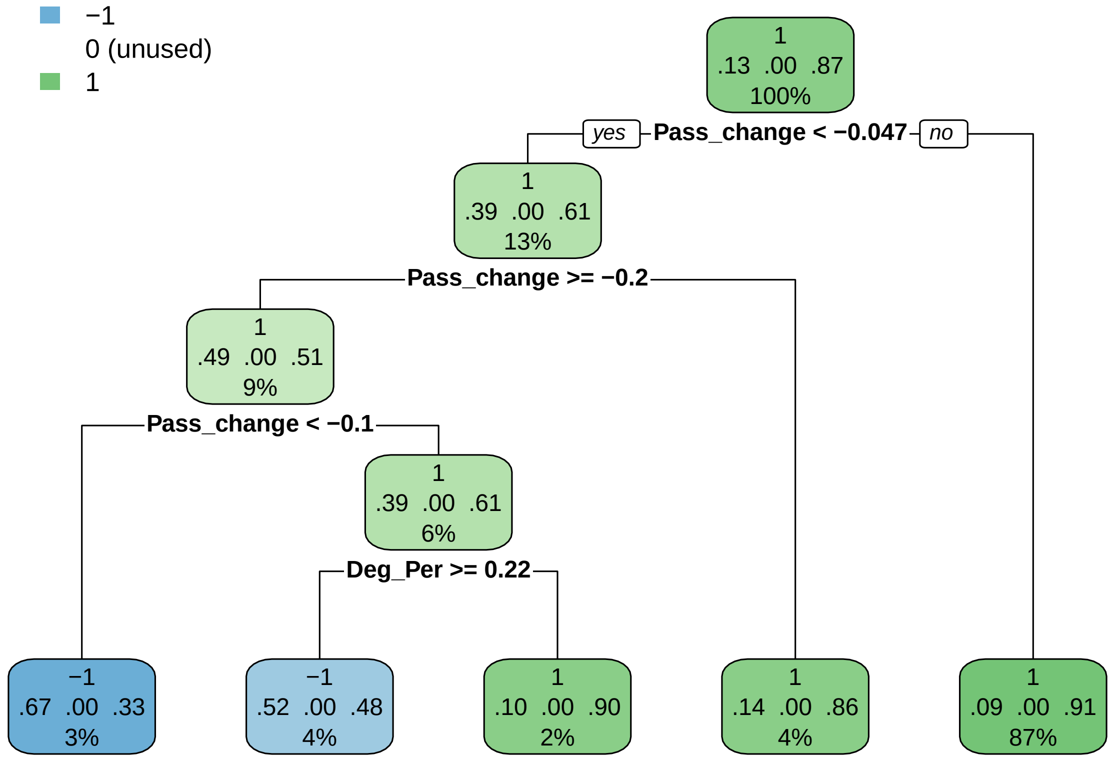
I find that with the best value for the minimal split size is **23**, which results in a cross validation missclassification rate of **13.2%** as described in the plot below (with the best value of one marked in blue). In other words, meaning that in **86.8%** we the our model predicts the direction of the GDP correctly.

Minimal_Split	CV_Error
23	0.1317039



Now, I can use the best value found above to train my classification tree:

```
class.tree <- rpart(Indicator~Deg_Per+Pass_change,
                    data = train, control =
                      rpart.control("minsplit" = Best.minsplit))
rpart.plot(class.tree)
```



The tree above helps us interpret how the change in GDP in a country for a given year is dependent on the change in passangers and in percentage of undergraduate students.

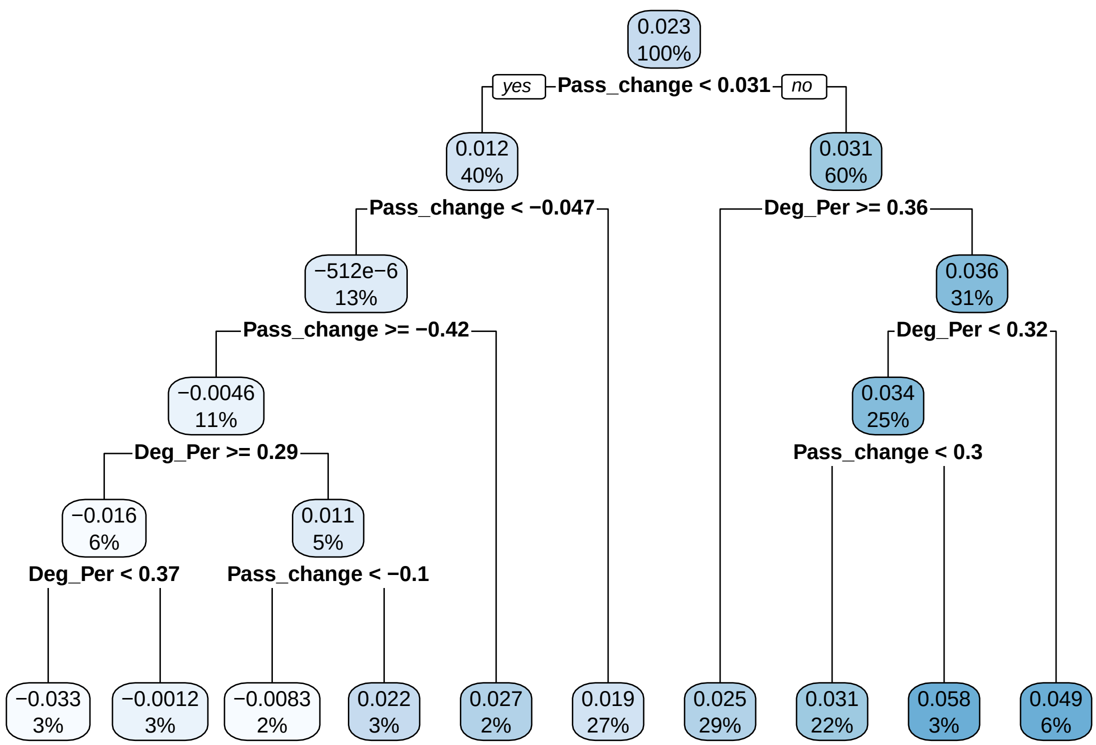
I find that the model predicts the direction of GDP out-of-sample pretty well, with a miss-classification rate of **16.05%** using the test sample:

```
test.pred <- predict(class.tree, newdata = test, type = "class")
test.missclass <- mean(as.numeric(as.character(test$Indicator))
                       !=as.numeric(as.character(test.pred)))
names(test.missclass) <- "Miss-Classification Error"
kable(t(test.missclass))
```

Miss-Classification Error
0.1605839

Now, I will try to predict the numerical change in GDP using a regression tree:

```
reg.tree <- rpart(GDP_Growth~Deg_Per+Pass_change, data = train)
rpart.plot(reg.tree)
```




```
test.pred.reg <- predict(reg.tree, newdata = test)
mse <- mean((as.numeric(as.character(test$GDP_Growth))
            -as.numeric(as.character(test.pred.reg)))^2)

names(mse) <- "Mean Squared Error"
kable(t(mse))
```

Mean Squared Error
0.0012053

For this analysis I achieve pretty nice results as well, with a mean squared error of **0.0012**.

Problem 2

In this problem I will implement a smart β strategy which is based on factor investing using the value criteria. This strategy attempts to gain excess returns by weighting a given benchmark given the value factor, or in other words, companies which have a low price compared to their fundamental value. I do this by inserting S&P500 index's 30 lowest positive *Price-to-Book* companies into my portfolio. This is a rather simple strategy and there are many other ways to invest in value companies, along with many other ways to invest according to other factors, but being a Fama-French enthusiastic and knowing the importance of the Book-to-Market ratio in their research made me want to try this version of value investing.

I start by importing the data and extract only the S&P500 companies data from it:

```
data.2 <- read.csv("fund.data2.csv", header = T) # Import Fundamental data of many companies
sp.ticker <- read.csv("sp500.csv", header = F) # Import SP500 Tickers
data.2 <- data.2[,-1]
Tickers <- as.character(data.2$Ticker)
head(data.2)
```

```
##   Ticker                        Name Price.to.Earnings Price.to.Book
## 1   FLWS      1 800 FLOWERS COM INC             20.83          2.72
## 2   SRCE            1ST SOURCE CORP             15.07          1.59
## 3    FOX           21st Century Fox             18.54          4.14
## 4    DDD             3D SYSTEMS CORP           -26.52          2.33
## 5    MMM                          3M             25.57         11.22
## 6   CAFD 8point3 Energy Partners LP                NA            NA
##   Book.to.Market Dividends Div_per_share Common.shares Ave_shares_diluted
## 1           0.37      0.00          0.00      64591371           66854500
## 2           0.63    -24.39          0.94      25965746           25952840
## 3           0.24  -1020.00          0.55    1852574153         1859500000
## 4           0.42      0.00          0.00     114180543                 NA
## 5           0.09  -3105.00          5.24     582287135          607150000
## 6             NA      0.00          0.00      79093305           43583000
##        MCAP Total._Equity Book_value_per_share
## 1    720.84        295.98                 4.58
## 2        NA        750.44                28.92
## 3  84236.55      21924.00                11.17
## 4   2045.08        587.07                 5.18
## 5 123317.88      10311.00                17.30
## 6        NA            NA                   NA
##                                    Sectore
## 1            Retail - Apparel & Specialty
## 2                                    Banks
## 3                            Entertainment
## 4                        Computer Hardware
## 5                      Industrial Products
## 6 Utilities - Independent Power Producers
```

```
head(sp.ticker)[,2]
```

```
## [1] MMM  ABT  ABBV ABMD ACN  ATVI
## 505 Levels: A AAL AAP AAPL ABBV ABC ABMD ABT ACN ADBE ADI ADM ADP ... ZTS
```

```
sp.ticker <- as.character(sp.ticker[,2])

extract_SP500 <- rep(NA,length(sp.ticker))
for(i in 1:length(extract_SP500)){
```

```
  extract_SP500[i] <- sp.ticker[i] %in% Tickers
}

sp.ticker <- sp.ticker[extract_SP500]
length(sp.ticker)
```

```
## [1] 482
```

I am left with 482 companies. I now need to extract the companies with the lowest *Price-to-Book* ratio (I will use a common rule of thumb which constructs the portfolio with the 30 companies with the lowest *Price-to-Book* ratio), removing all companies with negative book value since they are obviously not companies we would like to invest in:

```
ind <- rep(NA,length(sp.ticker))
for(i in 1:length(ind)){
  ind[i] <- which(Tickers==sp.ticker[i])
}

SP500.fund <- data.2[ind,]
dim(SP500.fund)
```

```
## [1] 482  13
```

```
sorted_SP500.fund <- SP500.fund[order(SP500.fund$Price.to.Book),]
sorted_SP500.fund <-
  sorted_SP500.fund[-which(sorted_SP500.fund$Price.to.Book<=0),] # removing negative BV
shortlist <- sorted_SP500.fund[1:30,c(1:2,13)]
```

I then extract the financial information for the companies in my portfolio for the last known quarter (Q4 2018):

```
start <- as.Date("2018-10-01")
end <- as.Date("2018-12-21")

# create list of stock tickers
TickerList <- as.character(shortlist$Ticker)

# read adjusted prices from Yahoo
adj.price <- NULL
for (Ticker in TickerList){
  adj.price <- cbind(adj.price, getSymbols(Ticker, from=start, to = end,
                                          verbose=FALSE, auto.assign=FALSE)[,6])
}

# keep only the dates that have adjusted prices for all tickers
adj.price <- adj.price[apply(adj.price,1,function(x) all(!is.na(x))),]
```

Now I can start calculating returns for my portfolio:

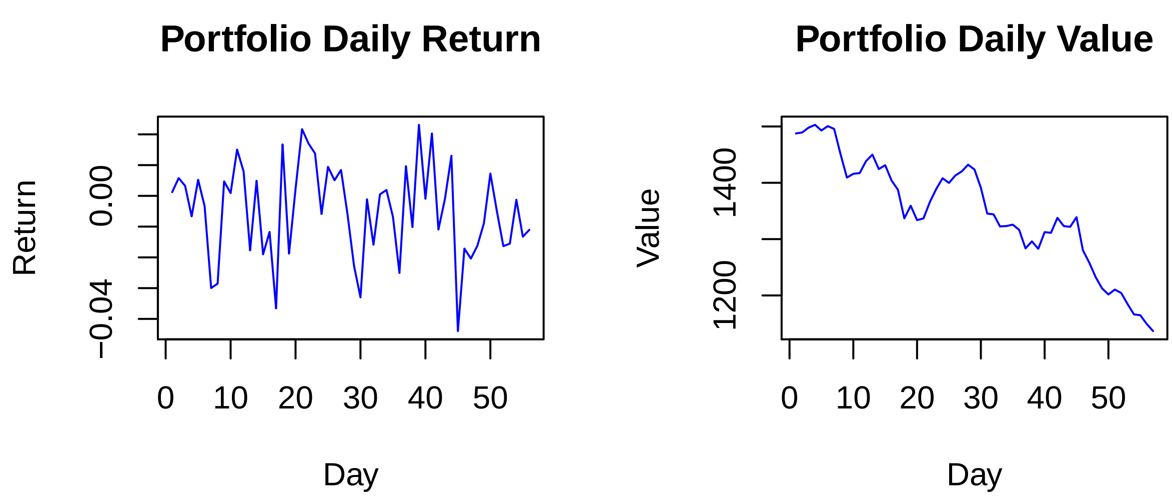
```
getSymbols("^IRX", from = start, to = end) # Extracting the risk free rate (3 months t-bill return)
rf <- IRX[,6]/100
```

```
portfolio <- rowSums(adj.price) # calculating the total price of the portfolio
# calculating the daily returns for my portfolio.
port.returns <- rep(NA,length(portfolio)-1) # calculating the daily returns for my portfolio.
for(i in 1:(length(portfolio)-1)){
  port.returns[i] <- portfolio[i+1]/portfolio[i]-1
}

excess.return <- port.returns-rf[2:length(rf)] # calculating the portfolio's daily excess return
names(excess.return) <- "Excess Return"
std.port <- stdev(excess.return) # standard deviation of the daily portfolio excess return
port.return <- portfolio[length(portfolio)]/portfolio[1]-1 # 3-months return on the portfolio
sharpe <- mean(excess.return)/std.port # sharpe ratio for my portfolio

cum.return <- c(1,rep(NA,length(port.returns))) # extracting the cummulative return
for(i in 1:length(cum.return)){
  cum.return[i+1] <- cum.return[i]*(1+port.returns[i])
}

par(mfrow = c(1,2))
plot(port.returns, xlab = "Day", ylab = "Return",
     main = "Portfolio Daily Return", type = "l", col = "blue")
plot(portfolio, xlab = "Day", ylab = "Value",
     main = "Portfolio Daily Value", type = "l", col = "blue")
```



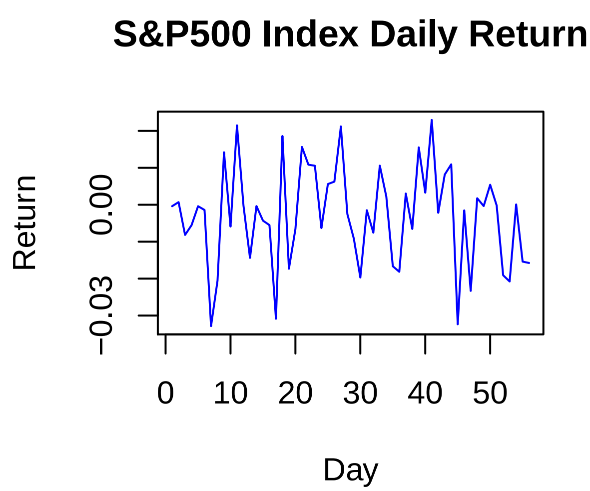
I will compare this portfolio with the S&P500 index ETF:

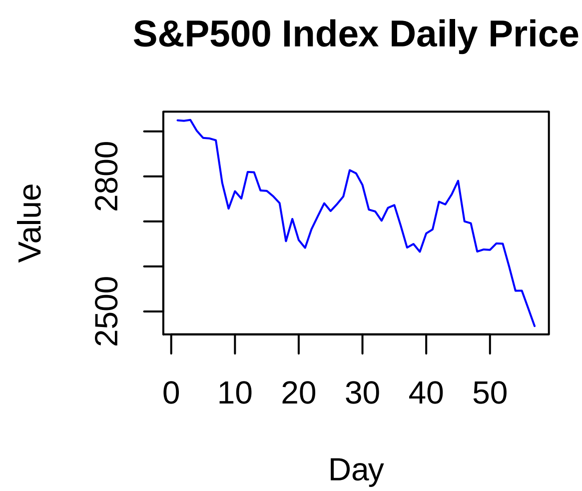
```r
getSymbols("^GSPC", from = start, to = end)
sp500.ind <- as.vector(GSPC[,6])
```

```r
sp500.returns <- rep(NA,length(sp500.ind)-1)
for(i in 1:(length(sp500.ind)-1)){
  sp500.returns[i] <- sp500.ind[i+1]/sp500.ind[i]-1
}

sp500.excess.return <- sp500.returns-rf[2:length(rf)] # calculating the ETF's daily excess return
names(sp500.excess.return) <- "Excess Return"
sp500.std.port <- stdev(sp500.excess.return) # standard deviation of the ETF's excess return
sp500.return <- sp500.ind[length(sp500.ind)]/sp500.ind[1]-1 # 3-months return on the S&P500 index
sp500.sharpe <- mean(sp500.excess.return)/sp500.std.port # sharpe ratio for S&P500 index

par(mfrow = c(1,2))
plot(sp500.returns, xlab = "Day", ylab = "Return",
     main = "S&P500 Index Daily Return", type = "l", col = "blue")
plot(sp500.ind, xlab = "Day", ylab = "Value",
     main = "S&P500 Index Daily Price", type = "l", col = "blue")
```





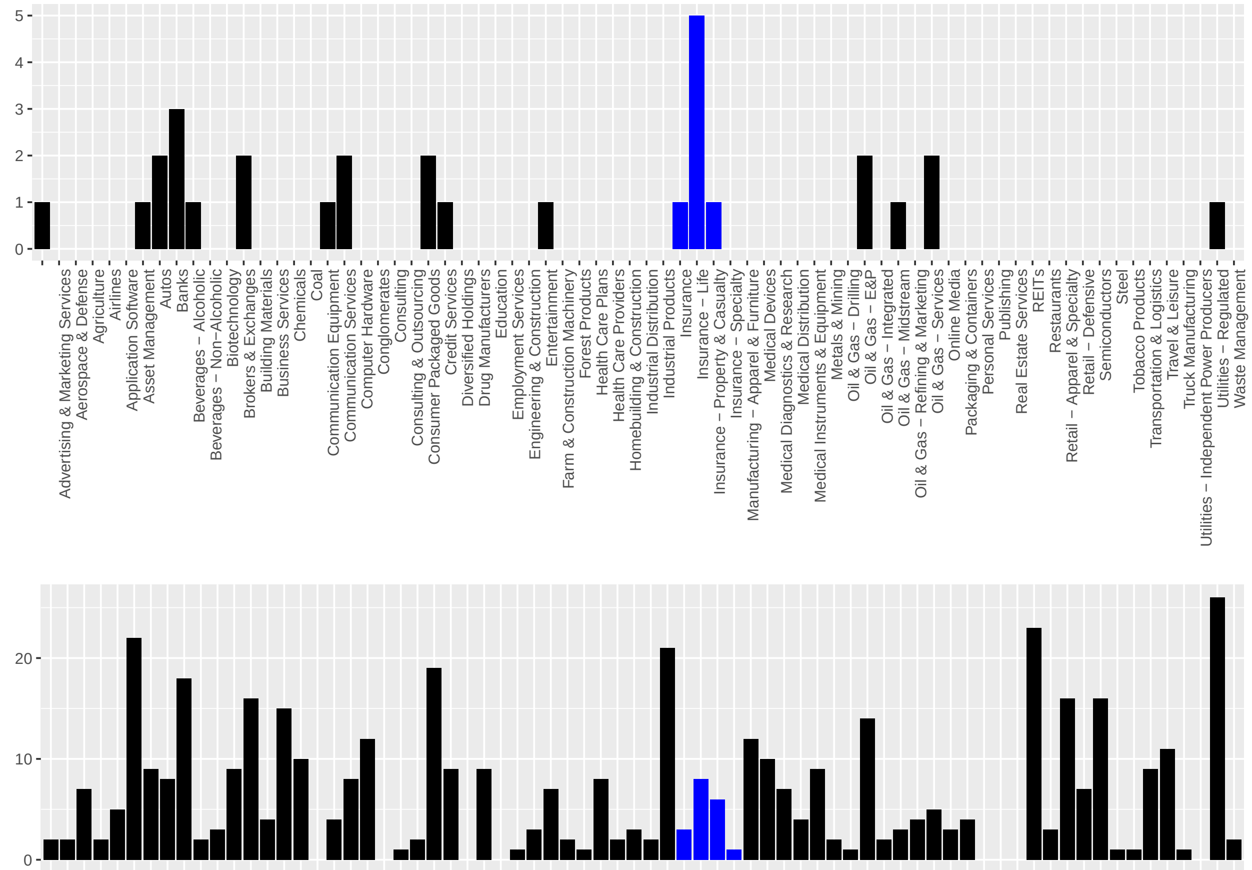
I will now compare my portfolio's results and the S&P500 ETF results:

```r
result.mat <- matrix(c(port.return,sharpe,sp500.return,sp500.sharpe),2,2)
colnames(result.mat) <- c("Value Portfolio","S&P500 Index")
rownames(result.mat) <- c("3 Months Return","Sharpe Ratio")
kable(result.mat)
```

	Value Portfolio	S&P500 Index
3 Months Return	-0.2358858	-0.1563194
Sharpe Ratio	-1.7968147	-1.9172294

We can immediately notice that returns are negative. That's because in the last 3 month the market suffered losses, specifically in December. We can also notice that although the sharpe ratio for the portfolio is higher than for S&P500 (well, less negative), the negative return on the portfolio is higher.

Trying to understand what the cause could be, the first thing that came to mind is that there perhaps the companies in our portfolio are from specific sectors which suffered higher losses in the discussed period. I observed the distribution of sectors for both the portfolio and 482 S&P500 companies, and found that there is a large amount of companies from the insurance sector in my portfolio, which is not representive of the S&P distribution, as can be described in the image below:



I therefore repeat the process above with insurance companies removed from the 482 cmopany data base, but find that the results don't change significantly:

```r
insurance.ind <-
  which(grepl("Insurance",as.character(sorted_SP500.fund$Sectore))) # finding the insurance companies
sorted_SP500.fund <- sorted_SP500.fund[-insurance.ind,] # removing insurance companies
new.shortlist <- sorted_SP500.fund[1:30,c(1:2,13)]

new.TickerList <- as.character(new.shortlist$Ticker)

# read closing prices from Yahoo keeping only the closing prices

new.adj.price <- NULL
for (Ticker in new.TickerList){
  new.adj.price <- cbind(new.adj.price, getSymbols(Ticker, from=start, to = end,
                                                   verbose=FALSE, auto.assign=FALSE)[,6])
}
# keep only the dates that have closing prices for all tickers

new.adj.price <- new.adj.price[apply(new.adj.price,1,function(x) all(!is.na(x))),]
new.portfolio <- rowSums(new.adj.price)
new.port.returns <- rep(NA,length(portfolio)-1)
for(i in 1:(length(portfolio)-1)){
  new.port.returns[i] <- new.portfolio[i+1]/new.portfolio[i]-1
}

new.excess.return <- new.port.returns-rf[2:length(rf)]
names(new.excess.return) <- "Excess Return"
new.std.port <- stdev(new.excess.return)
new.port.return <- new.portfolio[length(portfolio)]/new.portfolio[1]-1
new.sharpe <- mean(new.excess.return)/new.std.port

new.result.mat <- matrix(c(new.port.return,new.sharpe,sp500.return,sp500.sharpe),2,1)
colnames(new.result.mat) <- c("Value Portfolio")
rownames(new.result.mat) <- c("3 Months Return","Sharpe Ratio")
kable(new.result.mat)
```

	Value Portfolio
3 Months Return	-0.2412815
Sharpe Ratio	-1.8573680

Problem 3

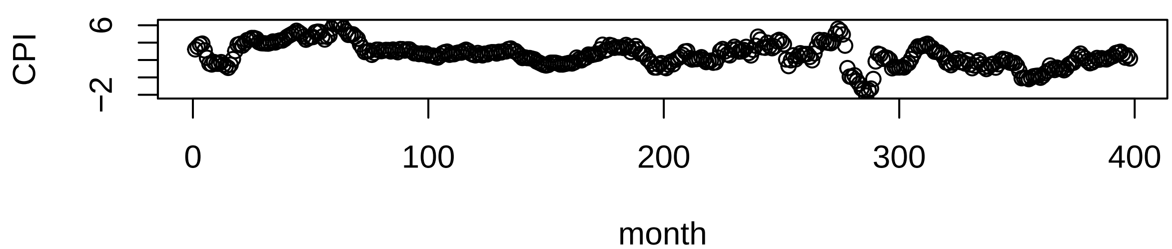
Part (a)

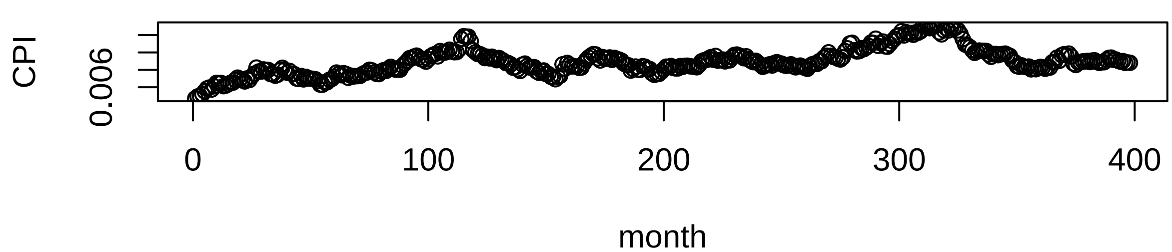
For this problem, I choose to use US monthly CPI to predict the USD-JPY exchange rate To do this, I collect the CPI data from the OECD website (https://data.oecd.org/price/inflation-cpi.htm) and the USD/JPY exchange rate from investing.com (https://www.investing.com/currencies/usd-jpy-historical-data). Both raw data sets and the processed data set I will be using (data3) exist in the GitHub folder.

Part (b)

I start with analyzing the CPI:

```
data.3 <- read.csv("data3.csv")
par(mfrow = c(2,1))
plot(data.3$CPI,ylab = "CPI", xlab = "month")
plot(1/data.3$USD_JPY,ylab = "CPI", xlab = "month")
```






```
kable(t(quantile(data.3$CPI)))
```

0%	25%	50%	75%	100%
-2.097161	1.726692	2.649503	3.39356	6.289809

By observing the CPI, we see it flactuates substantial over the given period (1985-2018).

Part (c)

I check whether the data source I used (CPI) cointegrates with the USD-JPY exchange rate using the Phillips-Ouliaris Cointegration test and find that it does:cointeg.test$p.value

```
po.mat <- data.3[,4:5]
cointeg.test <- po.test(po.mat)
alpha <- 0.05 # setting a 0.05 significance level
ifelse(cointeg.test$p.value <= alpha, "Cointegration", "No Cointegration")
```

```
## [1] "Cointegration"
```

Part (d)