Project 2: Index Tracking

Group 11: Vishwa Bhuta, Zack Bilderback, Green Guo, Victoria Salido

2/16/2017

Our constuctFund function takes in a similarity matrix, the number of stocks to pick for the index, price matrix, shares matrix, unique tickers for stocks, and the dates of the portfolio. The function then formulates the integer program by constructing constraints and the coefficients that will maximize the similarity between the n stocks and their representatives in the fund. Once the integer program has been solved and q number of stocks have been picked to best represent the index, the function then calculates weights for each stock in the fund and returns these weights.

When creating our similarity matrix, we decided that using the cosine similarity of the daily returns of the stocks would be a good measure in creating an index. Since it doesn't take the magnitude of the vectors into account, we thought this would make our function generalizable in case we had an input with missing returns, or few/more return data. Our function begins by calculating the returns from the share and price matrices, similar to the first part of the problem. Next, it basically creates an empty 100x100 similarity matrix and then loops through each pair of vectors in the return matrix to find their cosine similarity to input in the similarity matrix.

Please refer below to see our code for the return matrix, and the code/visuals for #4 and #5, which will include our portfolios and the plot of the returns.

Question 1

```r
# calculate return matrix
returnMat = matrix(NA, length(unique_dates)-1, length(unique_tickers)) # initialize matrix for daily re

for (i in 1:length(unique_tickers)) { # loop through each unique ticker symbol
  prices = priceMat[, i] # get the prices for that symbol
  daily_returns = diff(prices)/prices[-length(prices)] # calculate the return
  return_dates = unique_dates[2:length(unique_dates)] # remove the first date which we cannot calculate
  returnMat[, i] = daily_returns # update return matrix
}

rownames(returnMat) = as.character(return_dates) # set rownames as return dates
colnames(returnMat) = unique_tickers # set colnames as tickers
# Sample matrix
head(returnMat)[,1:6]
```

```
##                    AAPL         ADBE          ADI           ADP
## 2012-01-04  0.005374097 -0.010150508 -0.001387732 -0.0011935365
## 2012-01-05  0.011101974  0.007072136  0.004446915  0.0074455373
## 2012-01-06  0.010453771  0.008426966 -0.006640841  0.0014598540
## 2012-01-09 -0.001586127 -0.006615599  0.019498607 -0.0038265306
## 2012-01-10  0.003580442  0.023484052  0.004371585 -0.0001829157
## 2012-01-11 -0.001630281  0.004109589  0.004080522 -0.0009147457
##                    ADSK          AKAM
## 2012-01-04 -0.019474197 -0.0148800486
## 2012-01-05  0.020191989 -0.0003082614
## 2012-01-06  0.004542505  0.0101757632
## 2012-01-09  0.003875969 -0.0045787546
## 2012-01-10  0.046653797  0.0162526832
## 2012-01-11 -0.010759299  0.0259505130
```

Above is the daily returns for the stocks (6 of which are displayed) using the 2012 price data.

Question 2

```
corrMat = cor(returnMat, use="pairwise.complete.obs") # calculate correlation matrix, rho

head(corrMat)[,1:6]
```

```
##           AAPL      ADBE       ADI       ADP      ADSK      AKAM
## AAPL 1.0000000 0.2851738 0.3265191 0.3418359 0.2816883 0.2679535
## ADBE 0.2851738 1.0000000 0.5660888 0.5495794 0.6250335 0.4222517
## ADI  0.3265191 0.5660888 1.0000000 0.6220763 0.5816736 0.3663231
## ADP  0.3418359 0.5495794 0.6220763 1.0000000 0.4948663 0.3881868
## ADSK 0.2816883 0.6250335 0.5816736 0.4948663 1.0000000 0.4160317
## AKAM 0.2679535 0.4222517 0.3663231 0.3881868 0.4160317 1.0000000
```

Above is an excerpt of the correlation matrix for the returns of the 100 stocks (6 of which are shown).

Question 3

```
# Change 25 to any number to test
weights = constructFund(corrMat,25,priceMat, sharesMat, unique_tickers, unique_dates)
```

Question 4

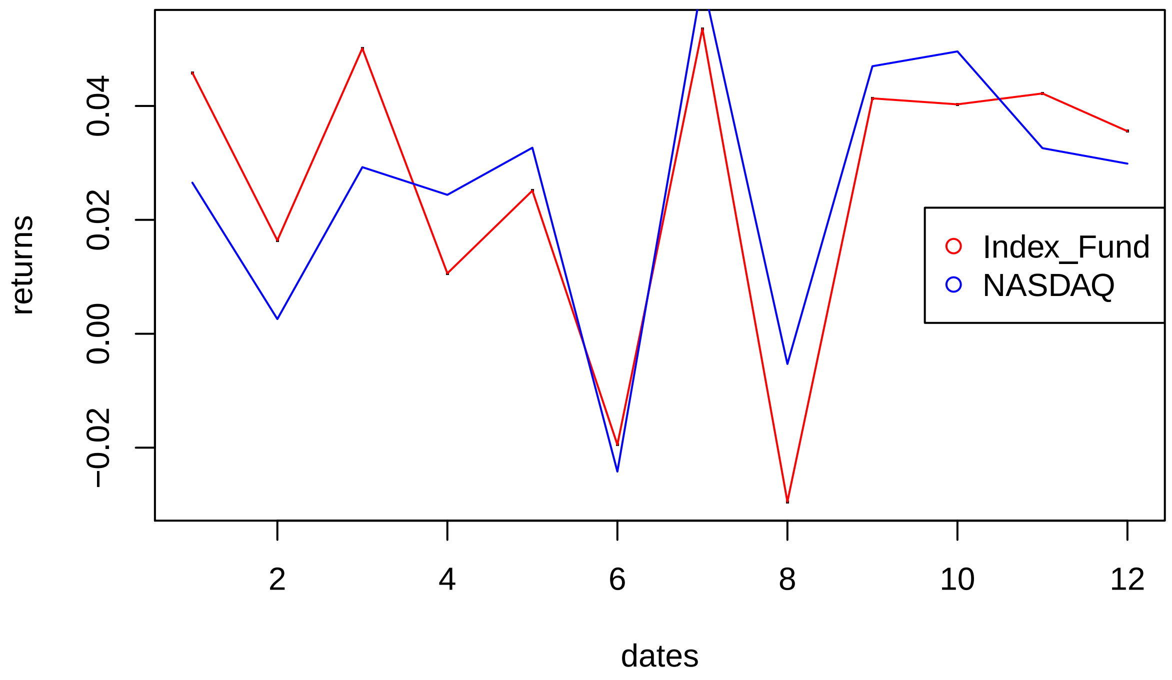
```
#assigning weights to each unique stock
#take those stocks that we will use in our portfolio (non-zero)
stocks=data.frame(unique_tickers,weights)
select=stocks[stocks$weights >0,]

#weights and tickers we want to use (25 selected)
weights=select$weights
select_stocks=select$unique_tickers

stock_prices = read.csv("N100StkPrices.csv", header=TRUE)

# accounting for changes in ticker names
# KFT changed to KRFT in Oct 2012.
stock_prices$TICKER[stock_prices$TICKER == "KFT"] = "KRFT"

# SXCI changed to CTRX in Jul 2012.
stock_prices$TICKER[stock_prices$TICKER == "SXCI"] = "CTRX"

# HANS changed to MNST in Jan 2012.
stock_prices$TICKER[stock_prices$TICKER == "HANS"] = "MNST"

#View(stock_prices) -- just checking to see if stock names actually changed
```

```r
date = apply(as.matrix(stock_prices$date), MARGIN=1, FUN="toString")
stock_prices$date=as.Date(date,"%Y%m%d")

#we are allowed to invest 1000000 into the fund
investment=1000000

#create a new data frame that only includes the new dates (for simplicity)
new_data=stock_prices[stock_prices$date>=as.Date('2012-12-31'),]

#extract the 25 stocks that we selected "
stocks_= new_data[new_data$TICKER %in% select_stocks,]
#make sure tickers are in correct order
stocks_= stocks_[with(stocks_, order(stocks_$TICKER)), ]
stocks_$weights=weights


#mulitply the investment by the weight of each stock -- this will allocate the amount of funds given to
#then divide the fund for each stock by the current stock price to determine how many of each to buy
stocks_$funds_per_stock=stocks_$weights * investment
stocks_$share_amount= stocks_$funds_per_stock/stocks_$PRC

#import NASDAQ stock data
Nasdaq=read.csv("N100Monthly.csv",header=TRUE)

#pull out the 25 stocks that our integer program had selected
index_port=Nasdaq[Nasdaq$TICKER %in% select_stocks,]
index_port= index_port[with(index_port, order(index_port$date,index_port$TICKER)), ]

#change date format
date = apply(as.matrix(index_port$date), MARGIN=1, FUN="toString")
index_port$date=as.Date(date,"%Y%m%d")

#for each month and each stock, multiply the amount of shares by the price
index_port$funds_per_stock=index_port$PRC * stocks_$share_amount


monthly_stock=aggregate(index_port$funds_per_stock~index_port$date,index_port,sum)

new_stocks=as.vector(monthly_stock$`index_port$funds_per_stock`)
new_stocks=append(new_stocks, investment, after=0)

returns=list()
for (i in seq(2,length(new_stocks))){
  returns=c(returns, (new_stocks[i]-new_stocks[i-1])/new_stocks[i-1])
}


#import stock imfo that was provided to compare performance
Nstocks=c(2660.93,2731.53,2738.58,2818.69,2887.44,2981.76,2909.60,3090.19,3073.81,3218.20,3377.73,3487.

Nasdaq_stocks=list()
for (i in seq(2,length(Nstocks))){
  Nasdaq_stocks=c(Nasdaq_stocks, (Nstocks[i]-Nstocks[i-1])/Nstocks[i-1])
}

dates=c(1,2,3,4,5,6,7,8,9,10,11,12)
```

```
#Print the stocks in portfolio and their weights:
print(select)
```

```
##    unique_tickers     weights
## 3             ADI 0.231247618
## 4             ADP 0.237772081
## 12           ATVI 0.004029073
## 15           BIDU 0.009399066
## 16           BIIB 0.057098560
## 18             CA 0.005733012
## 20           CERN 0.004552401
## 22           CHRW 0.006266820
## 27           CTRX 0.003301786
## 29           CTXS 0.030617392
## 36           ESRX 0.015077518
## 38           EXPE 0.052000598
## 40             FB 0.015181312
## 43           GILD 0.019051760
## 44           GMCR 0.002097689
## 53           KRFT 0.025741115
## 57           LMCA 0.012251822
## 58            MAR 0.235020419
## 60           MNST 0.002989607
## 66           NFLX 0.001756576
## 71           ORLY 0.010726518
## 87           TRIP 0.001860779
## 89           TSLA 0.001320275
## 95           VRTX 0.003107259
## 96            WDC 0.011798943
```

```
#Plot of portfolio returns against NASDAQ returns
library('ggplot2')
plot.new()

matplot(x=dates, y=returns,pch=".")
matlines(x=dates, y=returns, col='red')
matlines(x=dates, y=Nasdaq_stocks, col='blue')
legend("right", inset=0, legend=c("Index_Fund", "NASDAQ"), pch=1, col=c('red','blue'), horiz=FALSE)
```



As you can see, the portfolio created using a correlation matrix is decent at tracking the NASDAQ. It is especially good in the months of May-July, but it overperforms the NASDAQ in the early months and underperforms in the late months.

Q5

```
rho = similarityMat(priceMat, sharesMat, unique_tickers,unique_dates)
weights_2 = constructFund(rho,25,priceMat,sharesMat,unique_tickers,unique_dates)
```

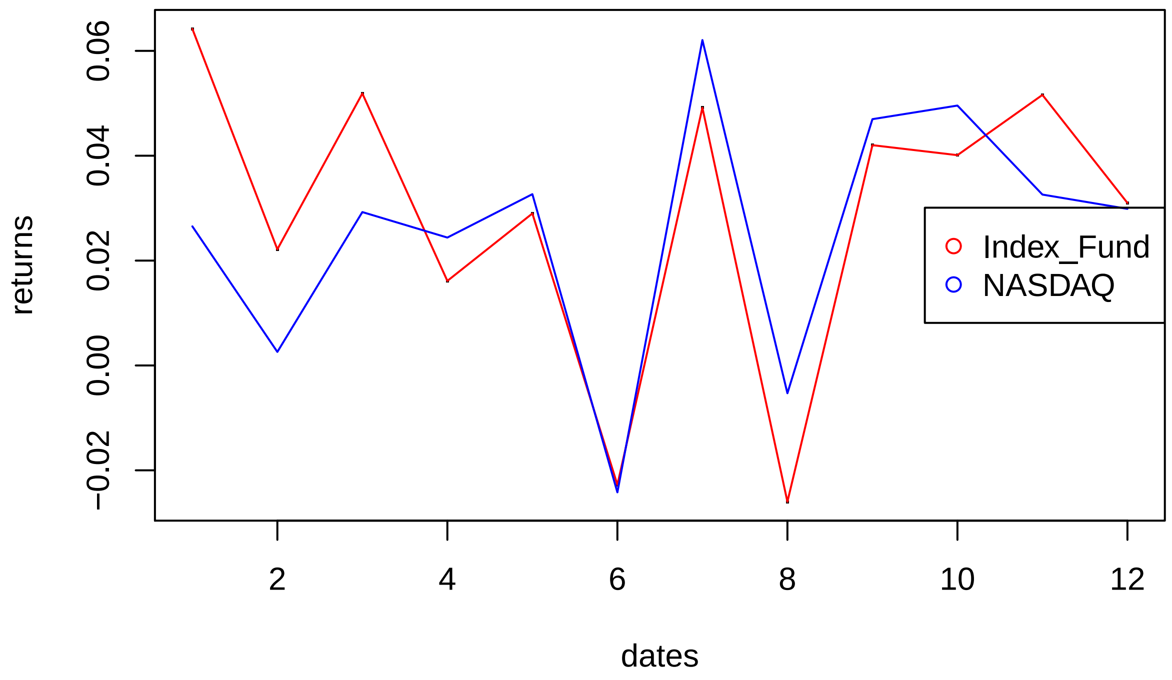
```
#assigning weights to each unique stock
#take those stocks that we will use in our portfolio (non-zero)
stocks=data.frame(unique_tickers,weights_2)
select=stocks[stocks$weights_2 >0,]

#weights and tickers we want to use (25 selected)
weights=select$weights
select_stocks=select$unique_tickers


#we are allowed to invest 1000000 into the fund
investment=1000000

#create a new data frame that only includes the new dates (for simplicity)
new_data=stock_prices[stock_prices$date>=as.Date('2012-12-31'),]
```

```
#extract the 25 stocks that we selected "
stocks_= new_data[new_data$TICKER %in% select_stocks,]
#make sure tickers are in correct order
stocks_= stocks_[with(stocks_, order(stocks_$TICKER)), ]
stocks_$weights=weights


#mulitply the investment by the weight of each stock -- this will allocate the amount of funds given to
#then divide the fund for each stock by the current stock price to determine how many of each to buy
stocks_$funds_per_stock=stocks_$weights * investment
stocks_$share_amount= stocks_$funds_per_stock/stocks_$PRC


#pull out the 25 stocks that our integer program had selected
index_port=Nasdaq[Nasdaq$TICKER %in% select_stocks,]
index_port= index_port[with(index_port, order(index_port$date,index_port$TICKER)), ]

#change date format
date = apply(as.matrix(index_port$date), MARGIN=1, FUN="toString")
index_port$date=as.Date(date,"%Y%m%d")

#for each month and each stock, multiply the amount of shares by the price
index_port$funds_per_stock=index_port$PRC * stocks_$share_amount

#View(index_port)

monthly_stock=aggregate(index_port$funds_per_stock~index_port$date,index_port,sum)

new_stocks=as.vector(monthly_stock$`index_port$funds_per_stock`)
new_stocks=append(new_stocks, investment, after=0)

returns=list()
for (i in seq(2,length(new_stocks))){
  returns=c(returns, (new_stocks[i]-new_stocks[i-1])/new_stocks[i-1])
}


#import stock info that was provided to compare performance
Nstocks=c(2660.93,2731.53,2738.58,2818.69,2887.44,2981.76,2909.60,3090.19,3073.81,3218.20,3377.73,3487.

Nasdaq_stocks=list()
for (i in seq(2,length(Nstocks))){
  Nasdaq_stocks=c(Nasdaq_stocks, (Nstocks[i]-Nstocks[i-1])/Nstocks[i-1])
}

dates=c(1,2,3,4,5,6,7,8,9,10,11,12)
```

```
#print stocks in our new portfolio and their weights
print(select)
```

```
##    unique_tickers    weights_2
## 3             ADI 0.2310230059
## 4             ADP 0.2348543363
## 12           ATVI 0.0040290727
## 15           BIDU 0.0093990660
## 16           BIIB 0.0761503191
## 20           CERN 0.0045524014
## 22           CHRW 0.0062668202
## 27           CTRX 0.0033017861
## 29           CTXS 0.0306173918
## 36           ESRX 0.0150775179
## 38           EXPE 0.0520005984
## 44           GMCR 0.0020976887
## 48           ILMN 0.0057330117
## 53           KRFT 0.0257411154
## 57           LMCA 0.0122518221
## 58            MAR 0.2350204190
## 60           MNST 0.0029896074
## 63            MWW 0.0002246126
## 66           NFLX 0.0169378883
## 71           ORLY 0.0107265182
## 87           TRIP 0.0018607787
## 89           TSLA 0.0013202745
## 94           VRSK 0.0029177449
## 95           VRTX 0.0031072591
## 96            WDC 0.0117989434
```

```r
#plot of portfolio returns against NASDAQ returns
plot.new()

matplot(x=dates, y=returns,pch=".")
matlines(x=dates, y=returns, col='red')
matlines(x=dates, y=Nasdaq_stocks, col='blue')
legend("right", inset=0, legend=c("Index_Fund", "NASDAQ"), pch=1, col=c('red','blue'), horiz=FALSE)
```



The portfolio we created using our cosine similarity matrix is a little better at tracking the NASDAQ. Overall, it is succeeding and failing in the same places as the previous index. It is overperforming the NASDAQ more than the previous portfolio in Jan-Apr, but underperforming it less from July-October.