Index Tracking Using Integer Programming

Constructing an index fund that tracks a specific broad market index could be done simply purchasing all the stocks in the index, with the same exact weights as in the index. However this approach is impractical (many small positions) and expensive (rebalancing costs may be incurred frequently, price response to trading). An index fund with q stocks, where q is substantially smaller than the size of the target population (n), seems desirable. We choose the stocks in our fund by picking the q stocks which best represent all the stocks in the index.

Cleaning NASDAQ data and pre-processing

```
rm(list = ls())
library(lpSolve)

data = read.csv("N100StkPrices.csv", header=TRUE)
data = na.omit(data)
ticker = data$TICKER

# spun off MDLZ
delete = seq(1, dim(data)[1])[ticker == "MDLZ"]
data = data[-delete, ]

date = apply(as.matrix(data$date), MARGIN=1, FUN="toString")
date = as.Date(date, "%Y%m%d")
ticker = data$TICKER
price = data$PRC
shares = data$SHROUT

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

# SXCI changed to CTRX in Jul 2012.
ticker[ticker == "SXCI"] = "CTRX"

# HANS changed to MNST in Jan 2012.
ticker[ticker == "HANS"] = "MNST"

# convert prices to a matrix, arranged by rows of dates and columns of tickers
unique_dates = sort(unique((date)))
unique_tickers = sort(unique(ticker))

priceMat = matrix(NA, length(unique_dates), length(unique_tickers))
sharesMat = matrix(0, length(unique_dates), length(unique_tickers))

for (i in 1:length(unique_tickers)) { # loop to create price and shares matrices
  tic = unique_tickers[i]
  idx = is.element(unique_dates, date[ticker==tic])

  priceMat[idx, i] = price[ticker==tic]
```

```
  sharesMat[idx, i] = shares[ticker==tic]
}

rownames(priceMat) = as.character(unique_dates)
rownames(sharesMat) = as.character(unique_dates)

rm(list = c("data", "delete", "i", "idx", "price", "shares", "tic", "ticker", "date"))
```

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

Creating the cosine 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.

```
similarityMat = function(priceMat, sharesMat, unique_tickers, unique_dates){
  # calculate return matrix
  returnMat = matrix(NA, length(unique_dates)-1, length(unique_tickers))
  # initialize matrix for daily returns

  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
    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

  #create cosine similarity matrix
  cosMat = matrix(NA, length(unique_tickers), length(unique_tickers))

  for (row in 1:length(unique_tickers)) {
    A = returnMat[,row] #define first vector for simmilarity calc
    for (col in 1:length(unique_tickers)) {
      B = returnMat[,col] #define second vector for simmilarity calc; gets looped through
      cosMat[row,col] = sum(A*B)/sqrt(sum(A^2)*sum(B^2)) #calculate cosine similarity
    }
  }
  return(cosMat)
}
```

Constructing the optimal fund weights

Our constuctFund function takes in a similarity matrix, the number of stocks to pick for the index, stock 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.

```
constructFund <- function(rho, q, priceMat, sharesMat, unique_tickers, unique_dates) {
  library(lpSolveAPI)

  N = length(unique_tickers)

  # The X coefficients come from the similiarity matrix because the
  # x variables represent the stocks that are similar to one another.
  # Y the variables represent which stocks are actually in the fund.
  # The coefficients for the y variables are all 0 because we are maximizing
  # the similarity between the n stocks and their representatives.
  #
  coeff.x <- as.vector(rho) #rho = corrMat
  coeff.y <- rep(0,N)

  c <- append(coeff.x, coeff.y)

  # create an LP model with 0 constraints (we will add constraints later)
  # and n decision variables -To pick q out of n stocks for our portfolio
  #we need n variables where n is the number of possible stocks
  idxLP <- make.lp(0,N^2+N)

  # set objective coefficients
  set.objfn(idxLP, c)

  # set objective direction
  lp.control(idxLP,sense = 'max')

  # add the constraints

  # 1st Constraint - This constraint says that there can only be
  #"q" amount of stocks in the index, which are represented by the y variables
  coeff.1 <- c(rep(0, N^2), rep(1, N))
  add.constraint(idxLP,coeff.1,"=",q)

  # 2nd Constraint - This constraint says that each stock i
  #has exactly one representaive stock j in the fund
  for(j in 1:N){

    # This is just updating indices so we know which variables get coefficients
    coeff.2 <- rep(0, N^2+N)

    start.idx <- N*(j-1) + 1
    end.idx <- N*j
    indices <- c(start.idx:end.idx)

    coeff.2[indices] <- 1

    add.constraint(idxLP,coeff.2, "=",1)
  }
```

```
  #3rd Constraint - This constraint says that stock i can be
  # represented by stock j in the fund only if j is in the fund

  # x constraints (10,000 x 10,000)
  x.mat <- diag(1,N^2)

  # y constraints (10,000 x 100)
  y.mat <- matrix(0,0,N)

  for (j in 1:N) {
    y.mat.temp <- matrix(0,N,N)
    y.mat.temp[1:N,] <- diag(-1,N)
    y.mat <- rbind(y.mat, y.mat.temp)
  }

  # Final matrix for 3rd constraint
  mat.3 <- cbind(x.mat, y.mat)

  for (i in 1:nrow(mat.3)) {
    coeff.3 = as.vector(mat.3[i,])
    add.constraint(idxLP,coeff.3, "<=",0)
  }

  # solve the model, if this return 0 an optimal solution is found
  set.type(idxLP, c(1:N^2+N),"binary")
  #write.lp(idxLP,'idxLP',type='lp')
  solve(idxLP)
  solution = get.variables(idxLP)
  x = solution[1:N^2]
  y = tail(solution, N)

  # Preprocessing for weight calculations
  end.values <- tail(sharesMat,1) * tail(priceMat,1)
  sum.values = sum(end.values)
  x.mat <- matrix(x, nrow = N, ncol = N, byrow = TRUE)

  # Calculate weights for each stock j

  weights <- c()
  for (i in 1:N) {
    weights = c(weights, sum(x.mat[,i] * end.values)/sum.values)
  }
  weights <- tail(weights,N)

  return(weights)
}
```

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

Creating fund

Here we look at how much of each stock to buy if we had $1M to invest, and then track the performance of our fund in 2013 against the NASDAQ.

```
#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

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 each of the 25 stocks
#then divide fund for each stock by the current stock price to determine how many 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
```

```r
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,
          3090.19,3073.81,3218.20,3377.73,3487.82,3592.00)

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)
```

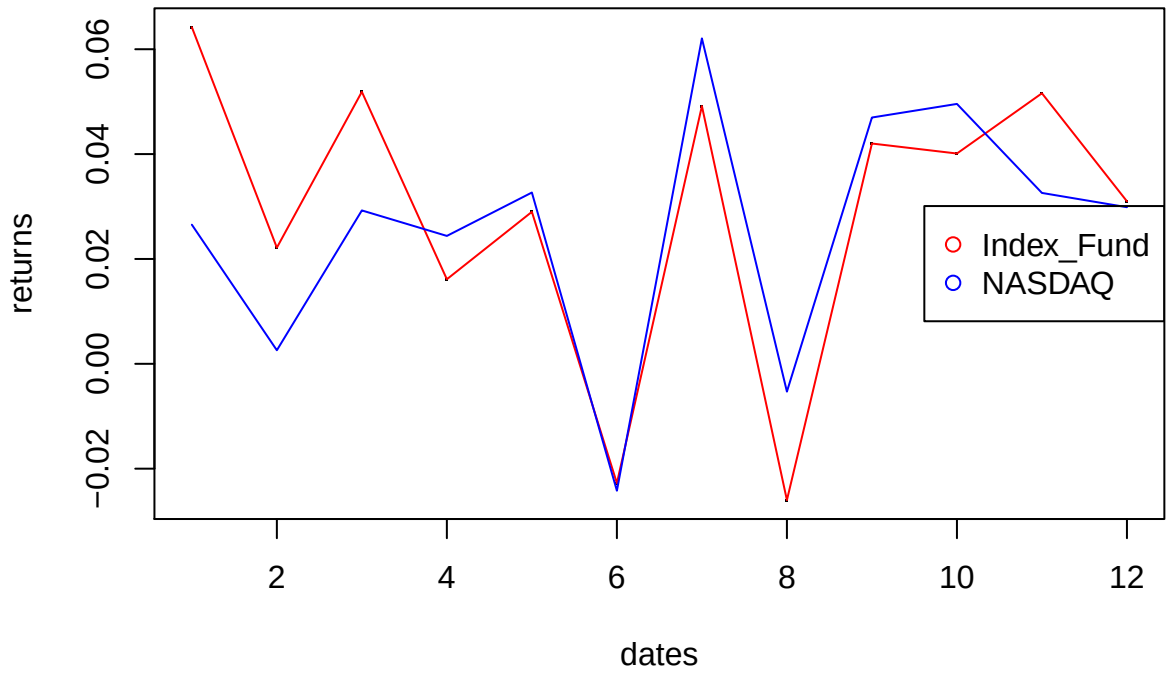
```r
#Print the stocks in portfolio and their weights:
print(select)
```

```
##    unique_tickers      weights
## 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
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.