The Buffet Bet - Basket of Funds vs. All Passive

Stats 107 Final Project

David Wihl

December 6, 2016

Abstract

Type: Simulating a trading strategy. Ten years ago, Warren Buffett made a famous bet : the ultimate passive investment, the Vanguard S&P 500 Index Fund Admiral (VFIAX), vs. a basket of funds determined by a team of hedge fund experts. Whoever had a lower return after ten years would donate $1 million to the charity of choice of the winner. After eight years, it looks increasingly likely that Buffett will win the bet.

Contents

1 Introduction

Can a basket of funds beat the market in long term?

Our project aims to create a Mutual Fund selector for semi-passive investor. A typical Vanguard customer may have to choose from over one hundred mutual funds without any guidance. We will create a portfolio to maximize returns over the long term using several common trading strategies, such as maximizing the Sharpe ratio. In order to make this more realistic, we will take into account fees and transaction costs. Since this is aimed at a typical Vanguard retail customer, no shorting will be allowed.

We will not account for taxes on the assumption that the funds are in a tax deferred account such as 401(k) or IRA. For this reason, we will not evaluate ETFs which are at a disadvantage in a tax deferred account.

The portfolio will start with $100,000, which is the current average 401(k) balance (source).

The overall time period will be evaluated whenever possible on a sixteen year history which is the maximum provided by Yahoo via `quantmod`.

The baseline or benchmark will be investing this amount in VFIAX and leaving it there for the duration. The first section examines VFIAX to determine if it is a valid proxy for market both in terms of similiar results and reasonable expenses. We will also examine how VFIAX compares to other Vanguard mutual funds in terms of CAGR and Volatility in order to motivate several trading strategies.

The starting fund in the basket will be VTHRX, which is one of the typical Vanguard Age-Adjusted balanced funds and a common default fund for many retirement plans.

We will simulate whether rebalancing on a monthly, quarterly, semi-annual or annual basis is the best strategy.

Time permitting, we will evaluate these trading strategies over six windows of ten years each for a more robust estimation of performance.

2 Baseline

2.1 Importing List of Vanguard Mutual Funds

From Vanguard, I obtained a list of the 168 currently offered mutual funds including published expense ratio. I converted this into a CSV for easy import into R.

Once the list of funds was imported, I used `quantmod` to read in historical daily adjusted data for the entire date range available from Yahoo. Since the earliest available date was November 13, 2000, I chose an even 16 year end period, ending November 13, 2016. While more data would have been preferable, I felt that this was a reasonable sample given it included two significant market corrections in 2001 and again in 2009.

Given the size of data (168 securities, 16 years of daily data), I cached the data locally to allow fast iteration on many models. See appendix for all code.

2.2 Summary Statistics

Several simple descriptive statistics were created: Total Return, Average Daily Return, Standard Deviation / Risk, Sharpe Ratio and CAGR. The following tables summarize the best and worst mutual funds.

Interestingly, in performing these summary statistics, I determined that `quantmod` does not use adjusted price to calculate returns. Without explicitly using adjusted pricing, the default calculation is wrong. I found this going through the quantmod sources and then confirmed it at http://stackoverflow.com/questions/34772616/r-function-periodreturn-not-computing-returns-using-adjusted-closing-prices As an extra precaution, I adjusted values upon retrieving the data using `getSymbol`.

Table 1: Best Sharpe Ratio

Ticker	Fund.Name	Expenses	CAGR	Sharpe
VSCSX	Vanguard Short Term Corp Bd Index Adm	0.10	0.03	0.11
VFSTX	Vanguard Short-Term Investment Grade Inv	0.20	0.04	0.11
VTABX	Vanguard Total Intl Bond Index Adm	0.14	0.04	0.11

Table 2: Worst Sharpe Ratio

Ticker	Fund.Name	Expenses	CAGR	Sharpe
VTIPX	Vanguard Shrt-Term Infl-Prot Sec Idx Inv	0.17	0.00	0
VDVIX	Vanguard Developed Mkts Index Inv	0.20	-0.01	0
VMMSX	Vanguard Emerging Mkts Select Stock Inv	0.93	-0.01	0

Table 3: Best CAGR

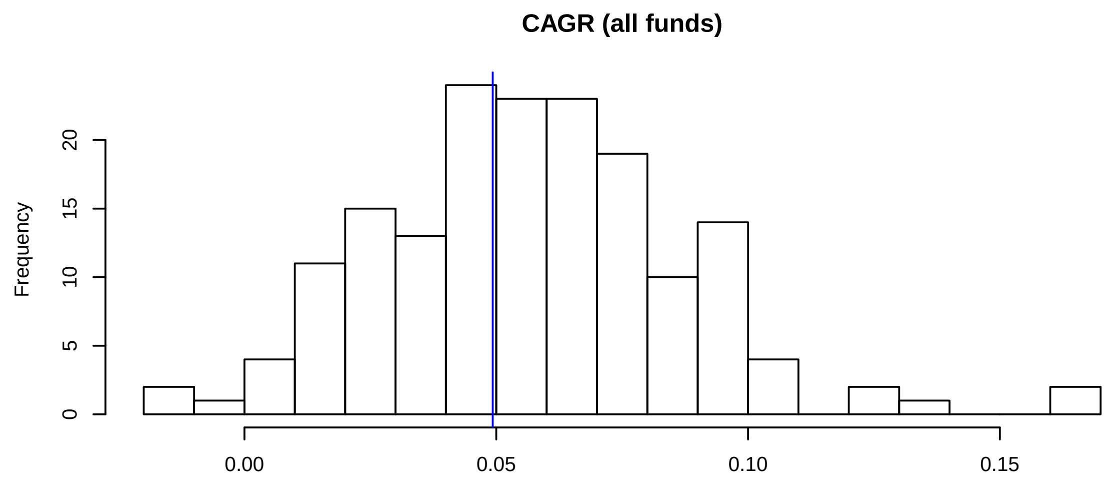
Ticker	Fund.Name	Expenses	CAGR	Sharpe
VSIAX	Vanguard Small-Cap Value Index Adm	0.08	0.16	0.03
VMVAX	Vanguard Mid-Cap Value Index Adm	0.08	0.16	0.07
VSGAX	Vanguard Small-Cap Growth Index Adm	0.08	0.13	0.05

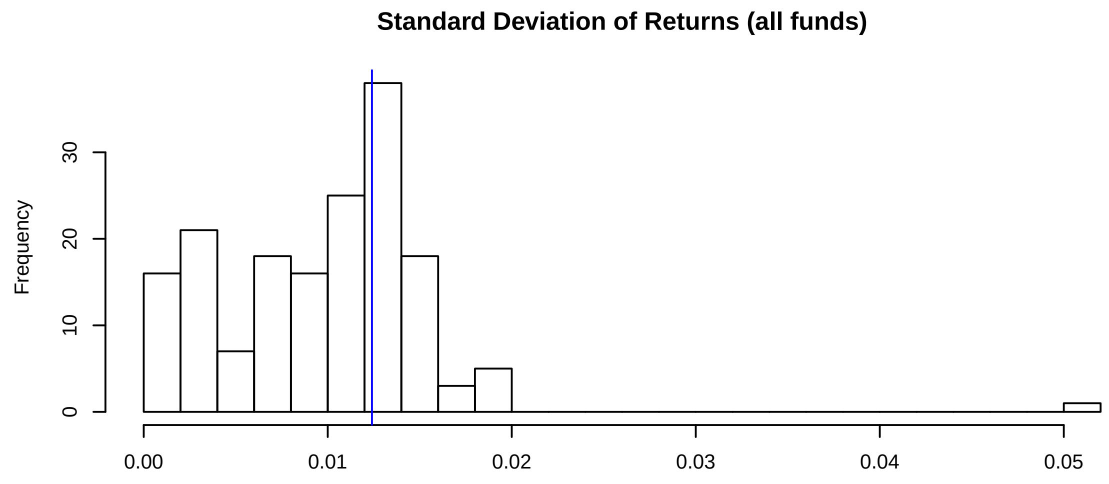
Table 4: Worst CAGR

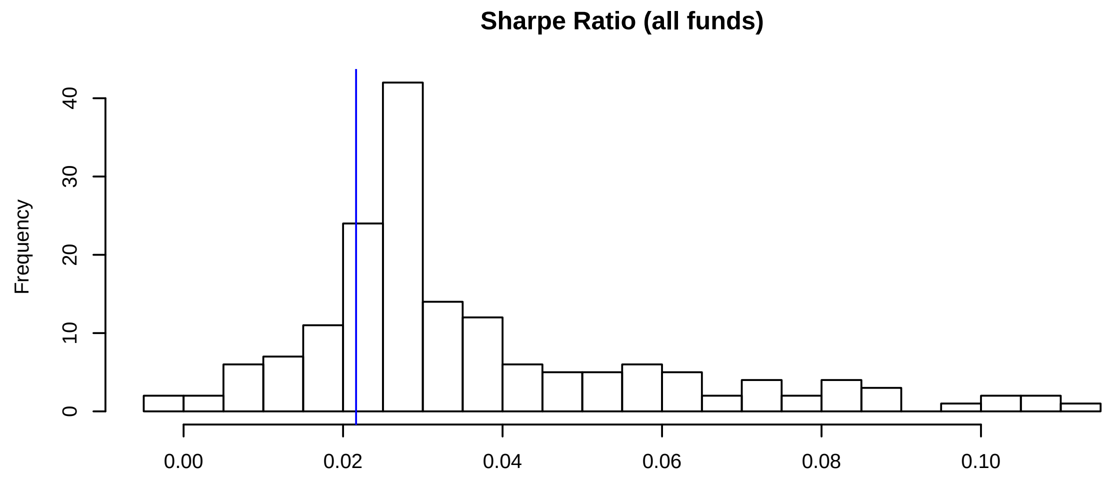
Ticker	Fund.Name	Expenses	CAGR	Sharpe
VMMSX	Vanguard Emerging Mkts Select Stock Inv	0.93	-0.01	0
VDVIX	Vanguard Developed Mkts Index Inv	0.20	-0.01	0
VTIPX	Vanguard Shrt-Term Infl-Prot Sec Idx Inv	0.17	0.00	0

VMMSX is particularly bad: high expenses, low returns and low Sharpe Ratio.

The following histograms show the distribution of all funds. VFIAX is the blue line.


CAGR (all funds)
Frequency
0
5
10
15
20
0.00
0.05
0.10
0.15


Standard Deviation of Returns (all funds)
Frequency
0
10
20
30
0.00
0.01
0.02
0.03
0.04
0.05


Sharpe Ratio (all funds)
Frequency
0
10
20
30
40
0.00
0.02
0.04
0.06
0.08
0.10

Based on these summary statistics and distributions, it appears that there are better mutual fund choices available than VFIAX, albeit at slightly higher risk for higher reward.

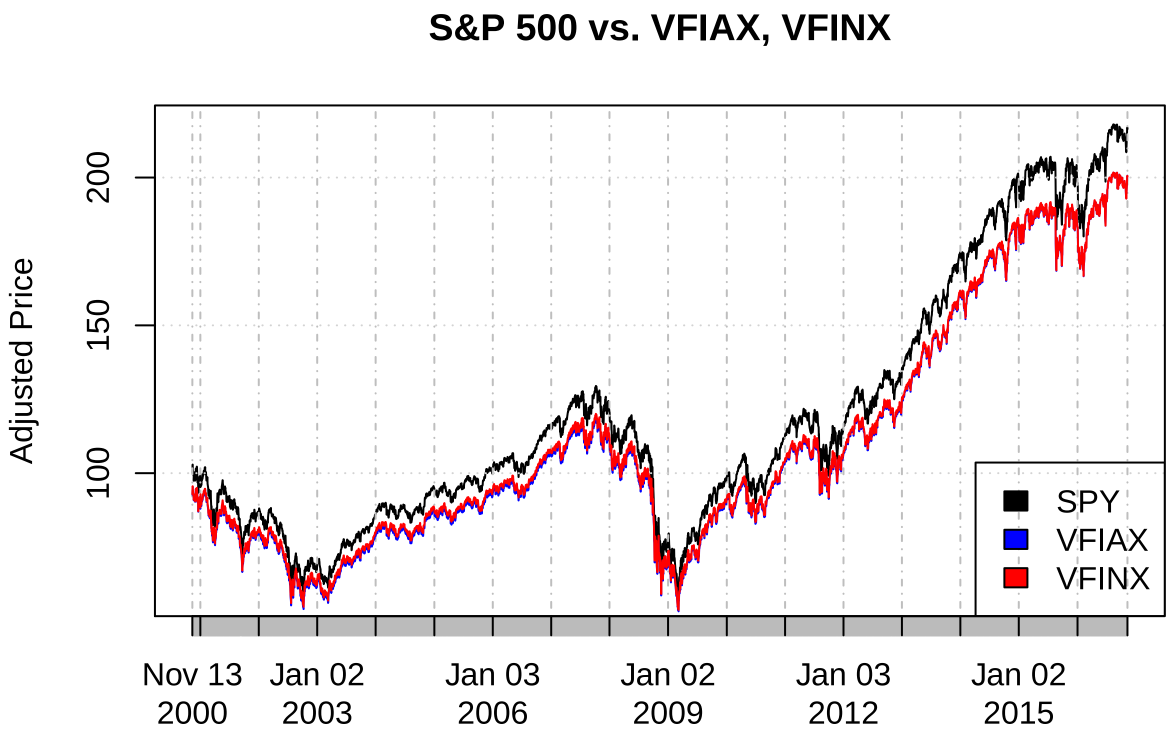
2.3 Expense Ratios

Every mutual fund has expense ratios which is the principle way that a company like Vanguard makes money. An expense ratio is a mutual fund's annual operating expense, expressed as a percentage of the fund's average net assets. It's calculated annually and removed from the fund's earnings before they're distributed to investors, directly reducing investors' returns. Since expenses were taken directly from assets, no special calculations needed to be made for expenses, since the costs were already reflected in share prices, and therefore the returns.

For VFIAX, Vanguard claims a 0.05% expense ratio. These are considered "admiral" class shares. For smaller investors, there are "investor" class shares (VFINX) with a 0.16% expense ratio. This expense has gone down over time. We shall now examine if this matches reality and how returns deviate from the benchmark S&P 500.

Metric	VFINX (Investor)	VFIAX (Admiral)
Claimed Expense	0.15%	0.05%
Cum. Exp. 16 Yrs	2.4%	0.8%
Actual Expense	0.8%	0.4%
Difference	67.1%	50%
CAGR	4.9166%	4.9299%

SPY CAGR: 4.9446%



By these calculations, expenses add up over time but are lower than advertised. Admiral shares do not offer significant benefit over Investor shares even though the expenses are purported to be 1/3. To be precise, there are less than 2 basis points separating their CAGR. For a very small cost, typically 1.5 basis points of CAGR for Admiral shares, Vanguard index funds offer significant convenience.

3 Trading Methods

From the previous section, we have established that VFIAX:

- represents the S&P 500 market without significant overhead
- it seems possible to do better

We will now iterate through a number of trading strategies to determine which trading models can best achieve this potential.

3.1 Age Based Balanced Fund

3.2 Pick Another Fund

3.3 Diversified Static Set of Funds

3.4 Sharpe Ratio / Efficient Frontier

3.5 Pairs Trading

3.6 Moving Averages

3.7 Momentum Trading

3.8 Fama French Three Factor

3.9 Fama French Five Factor

3.10 Minimum Variance

4 Conclusions and Discussions

5 References

This entire project code including source data can be found on GitHub (https://github.com/wihl/stats107-project)

List of Vanguard funds, including fees and transaction costs, supplied by Vanguard https://investor.vanguard.com/mutual-funds/list#/mutual-funds/asset-class/month-end-returns

The Buffet Bet, http://longbets.org/362/

Fortune, The Buffett Bet http://fortune.com/2016/05/11/warren-buffett-hedge-fund-bet/

6 Appendix A - Code

6.1 Preparing Report

The following is the code used to prepare this report:

```
# Load libraries
library(ggplot2)
library(quantmod)
library(knitr)
#
# Import List of Securities
#
loadRData = function(filename) {
  # Source: http://stackoverflow.com/questions/5577221/how-can-i-load-an-object-into-a-variable-name-th
  load(filename)
  get(ls()[ls()!="filename"])
}
fromDate = "2000-11-13"
toDate   = "2016-11-13"
funds    = read.csv("data/vanguard.csv",header = T, stringsAsFactors = F)
# Load from previously cached download
stockDataEnv= loadRData("data/stockData.RData")
fundData = mget(funds$Ticker,stockDataEnv)
#
# Generate Summary Statistics
#
for (i in 1:nrow(funds)) {
  sym = funds$Ticker[i]
  data = eval(parse(text=paste("fundData$",sym,sep="")))
  if(!is.null(data)){
    # we have the stockdata; Calculate summary statistics
    funds$startDate[i] = as.Date(index(data[1]))
    funds$endDate[i] = as.Date(index(last(data[])))
    startPrice = as.numeric(Ad(data[1]))
    endPrice = as.numeric(Ad(last(data[])))
    funds$totalRet[i] = (endPrice - startPrice) / startPrice
    dailyRets = dailyReturn(Ad(data))
    funds$stdDev[i] = sd(dailyRets)
    funds$avgRet[i] = mean(dailyRets)
    funds$Sharpe[i] = funds$avgRet[i] / funds$stdDev[i]
    # CAGR
    years =  as.numeric((as.Date(funds$endDate[i]) -
                           as.Date(funds$startDate[i])))/365
    funds$CAGR[i] = ((endPrice / startPrice)^(1/years)) - 1

  }
}
#
# Generate Tables of Top Performers
#
colsToDisplay = c("Ticker","Fund.Name","Expenses","CAGR","Sharpe")
kable(head(funds[order(-funds$Sharpe),colsToDisplay],n=3), caption="Best Sharpe Ratio",
      row.names = F,digits=2)
```

```r
kable(head(funds[order(funds$Sharpe),colsToDisplay],n=3), caption="Worst Sharpe Ratio",
      row.names = F,digits=2,n=3)

kable(head(funds[order(-funds$CAGR),colsToDisplay],n=3), caption="Best CAGR",
      row.names = F,digits=2,n=3)

kable(head(funds[order(funds$CAGR),colsToDisplay],n=3), caption="Worst CAGR",
      row.names = F,digits=2,n=3)

#
# Show histograms with base VFIAX case
#
par(mfrow=c(3,1))

# CAGR
#ggplot(funds, aes(x=CAGR)) +
#    geom_histogram(binwidth=.5, colour="black", fill="white") +
#    geom_vline(aes(xintercept=funds[funds$Ticker=="VFIAX",]$CAGR),
#                color="red", linetype="dashed", size=1)

hist(funds$CAGR,main="CAGR (all funds)",xlab="",breaks=20)
abline(v=funds[funds$Ticker=="VFIAX",]$CAGR,col="blue")

# Standard Deviation
hist(funds$stdDev,main="Standard Deviation of Returns (all funds)",xlab="",breaks=20)
abline(v=funds[funds$Ticker=="VFIAX",]$stdDev,col="blue")

# Sharpe Ratio
hist(funds$Sharpe,main="Sharpe Ratio (all funds)",xlab="",breaks=20)
abline(v=funds[funds$Ticker=="VFIAX",]$Sharpe,col="blue")
#
# Compare claimed Expense Ratios vs. Actual Expense Ratios
#
getSymbols("SPY", src="yahoo", from=as.Date(fromDate),to=toDate,adjust=T)
p0 = as.numeric(Ad(SPY[1]))
p1 = as.numeric(Ad(last(SPY[])))
tot = (p1 - p0)/p0

# Compound expenses over an expected 16 year period
expectedVFIAX = round((1.0005 ^ 16 - 1) * 100, 2)
expectedVFINX = round((1.0015 ^ 16 - 1) * 100, 2)
# Determine the percentage difference in total return
spyOverVFIAX = round((tot - funds[funds$Ticker=="VFIAX",]$totalRet)/ funds[funds$Ticker=="VFIAX",]$tota
spyOverVFINX = round((tot - funds[funds$Ticker=="VFINX",]$totalRet)/ funds[funds$Ticker=="VFINX",]$tota
pctOverVFIAX = round((expectedVFIAX - spyOverVFIAX) / expectedVFIAX * 100,1)
pctOverVFINX = round((expectedVFINX - spyOverVFINX) / expectedVFINX * 100,1)

years =  as.numeric((as.Date(funds[funds$Ticker=="VFIAX",]$endDate)) -
                     as.Date(funds[funds$Ticker=="VFIAX",]$startDate  ))/365
spy.CAGR =  (p1 / p0)^(1/years)  - 1

# Plot Difference in Adjusted Price based on Expense Ratio
plot(Ad(SPY), main="S&P 500 vs. VFIAX, VFINX",ylab="Adjusted Price")
```

```r
lines(Ad(fundData$VFIAX),col="blue")
lines(Ad(fundData$VFINX),col="red")
legend("bottomright",c("SPY","VFIAX","VFINX"),fill=c("black","blue","red"))
##
##
```

6.2 Caching Stock Data

Stock data was downloaded and cached with the following code:

```r
# Harvard Stats 107, Fall 2016, Dr. Parzen
# by David Wihl
# Download and cache stock symbol data
# Inspired from http://gekkoquant.com/2012/06/01/stock-data-download-saving-r/
library(quantmod)
fromDate = "2000-11-13"
toDate   = "2016-11-13"
funds    = read.csv(
  "data/vanguard.csv",
  header = T,
  colClasses = c("character", "character", "numeric")
)
stocksLst = funds$Ticker
savefilename <- "data/stockdata.RData" #The file to save the data in
startDate = as.Date(fromDate) #Specify what date to get the prices from
maxretryattempts <- 5 #If there is an error downloading a price how many times to retry

#Load the list of ticker symbols from a csv, each row contains a ticker
stockData <- new.env() #Make a new environment for quantmod to store data in
nrstocks = length(stocksLst) #The number of stocks to download

#Download all the stock data
for (i in 1:nrstocks) {
  for (t in 1:maxretryattempts) {
    tryCatch({
      #This is the statement to Try
      #Check to see if the variables exists
      #NEAT TRICK ON HOW TO TURN A STRING INTO A VARIABLE
      #SEE  http://www.r-bloggers.com/converting-a-string-to-a-variable-name-on-the-fly-and-vice-versa-
      if (!is.null(eval(parse(
        text = paste("stockData$", stocksLst[i], sep = "")
      )))) {
        #The variable exists so dont need to download data for this stock
        #So lets break out of the retry loop and process the next stock
        #cat("No need to retry")
        break
      }

      #The stock wasnt previously downloaded so lets attempt to download it
      cat("(",i,":",nrstocks,") ","Downloading ", stocksLst[i],"\t\t Attempt: ",t,"/",maxretryattempts,
      getSymbols(
        stocksLst[i],
        env = stockData,
        src = "yahoo",
        from = startDate,
        to = toDate,
        adjust = T
      )
    }
    #Specify the catch function, and the finally function
    , error = function(e)
      print(e))
  }
}

#Lets save the stock data to a data file
tryCatch({
  save(stockData, file = savefilename)
  cat("Sucessfully saved the stock data to %s", savefilename)
}
, error = function(e)
  print(e))
```