The Buffet Bet - Basket of Funds vs. All Passive

Stats 107 Final Project

David Wihl

December 6, 2016

Abstract

Ten years ago, Warren Buffett made a famous : 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.

Type: Simulating a trading strategy.

Introduction

TODO

Methods

Importing List of Vanguard Mutual Funds

From Vanguard, I obtain a list of the 168 currently offered mutual funds including 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 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.

Summary Statistics

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

Interestingly, in performing these summary statistics, I determined that `quantmod` does not use adjusted price to calculate returns. So, basically, it 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

Ticker	Fund.Name	Expenses	CAGR	Sharpe
VTABX	Vanguard Total Intl Bond Index Adm	0.14	0.04	0.11
VSGBX	Vanguard Short-Term Federal Inv	0.20	0.04	0.10
VTIBX	Vanguard Total Intl Bond Index Inv	0.17	0.04	0.10
VFISX	Vanguard Short-Term Treasury Inv	0.20	0.03	0.10

Table 2: Worst Sharpe Ratio

Ticker	Fund.Name	Expenses	CAGR	Sharpe
VTIPX	Vanguard Shrt-Term Infl-Prot Sec Idx Inv	0.17	0.00	0.00
VDVIX	Vanguard Developed Mkts Index Inv	0.20	-0.01	0.00
VMMSX	Vanguard Emerging Mkts Select Stock Inv	0.93	-0.01	0.00
VTAPX	Vanguard Shrt-Term Infl-Prot Sec Idx Adm	0.08	0.00	0.00
VFWIX	Vanguard FTSE All-World ex-US Index Inv	0.26	0.00	0.01
VIRSX	Vanguard Inst'l Target Retirement 2040	0.10	0.01	0.01

Table 3: Best CAGR

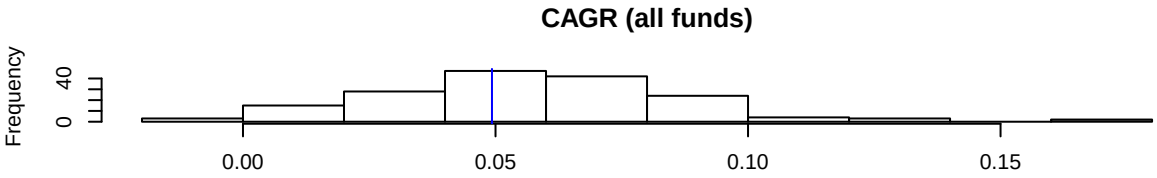
Ticker	Fund.Name	Expenses	CAGR	Sharpe
VSIAX	Vanguard Small-Cap Value Index Adm	0.08	0.16	0.03
VMVAX	Vanguard Mid-Cap Value Index Adm	0.08	0.16	0.07
VSGAX	Vanguard Small-Cap Growth Index Adm	0.08	0.13	0.05
VMGMX	Vanguard Mid-Cap Growth Index Adm	0.08	0.13	0.05
VEVFX	Vanguard Explorer Value Inv	0.65	0.12	0.04
VGSIX	Vanguard REIT Index Inv	0.26	0.11	0.03

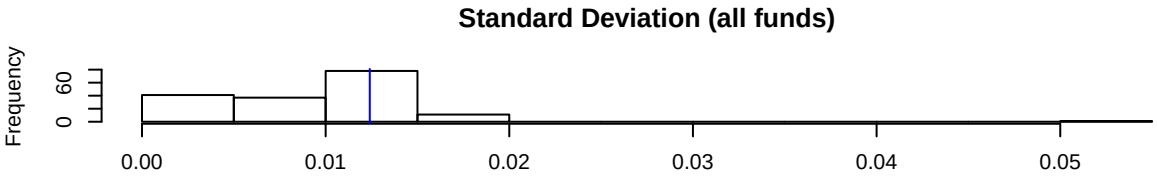
Table 4: Worst CAGR

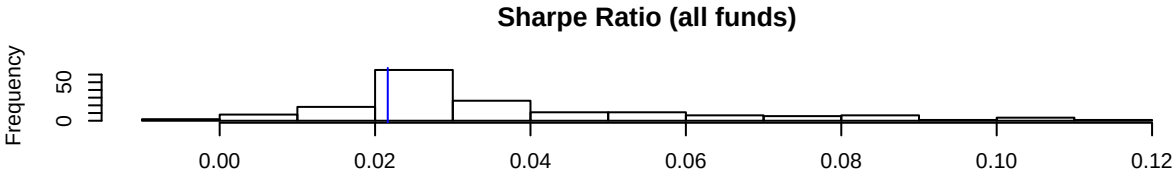
Ticker	Fund.Name	Expenses	CAGR	Sharpe
VMMSX	Vanguard Emerging Mkts Select Stock Inv	0.93	-0.01	0.00
VDVIX	Vanguard Developed Mkts Index Inv	0.20	-0.01	0.00
VTIPX	Vanguard Shrt-Term Infl-Prot Sec Idx Inv	0.17	0.00	0.00
VTAPX	Vanguard Shrt-Term Infl-Prot Sec Idx Adm	0.08	0.00	0.00
VFWIX	Vanguard FTSE All-World ex-US Index Inv	0.26	0.00	0.01
VUBFX	Vanguard Ultra-Short-Term Bond Inv	0.20	0.01	0.08

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

```
## Warning in par(mfrow = c(3, 1), par = rep(4, 2)): "par" is not a graphical
## parameter
```







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.

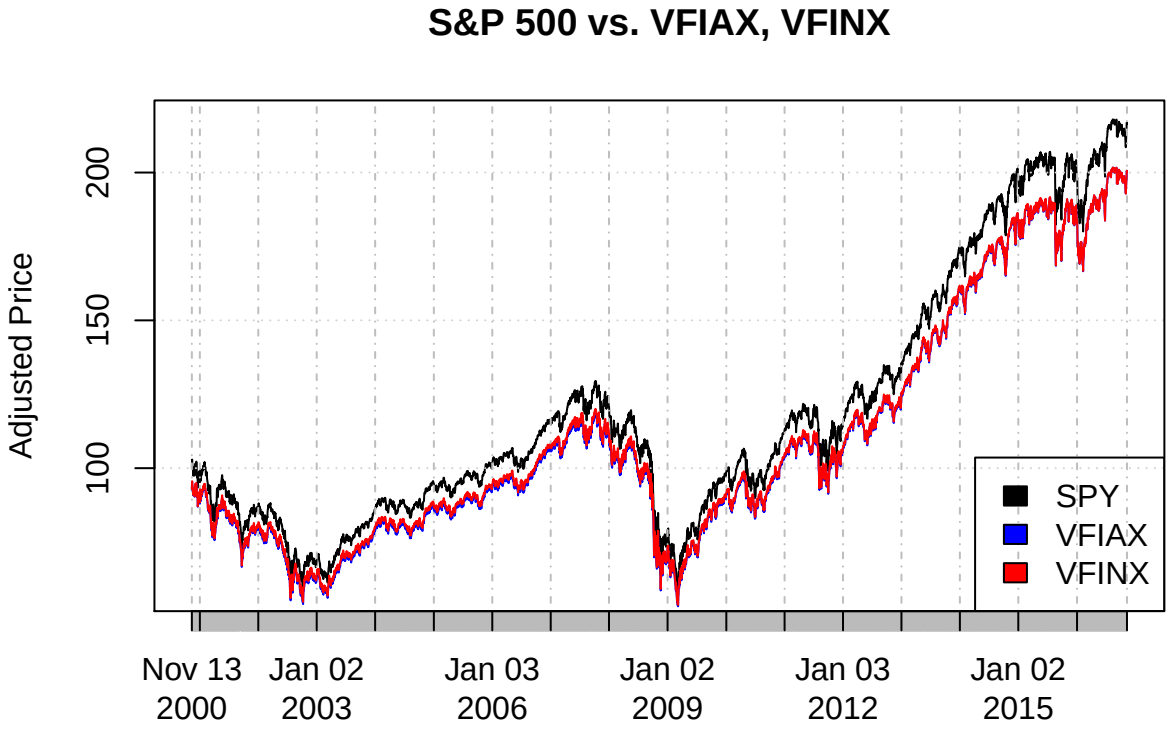
For VFIAX, Vanguard claims a 0.05% expense ratio. These are considered "admiral" class shares. For smaller investors, there are "investor" class shares (VFINX) with a 0.16% expense ratio. This expense has gone down over time. We shall now examine if this matches reality and how returns deviate from the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

```
## [1] "SPY"
```

TODO: Fix expense ratio calculation

Metric	VFINX (Investor)	VFIAX (Admiral)
Claimed Expense	0.15%	0.05%
Cum. Exp. 16 Yrs	243%	80%
Actual Expense	0.0079953	0.0041855
Difference	0.3%	0.5%
CAGR	0.049166	0.0492992

SPY CAGR: 0.0494464



In sum, expenses add up over time. Admiral shares do not offer significant benefit over Investor shares even though the expenses are purported to be 1/3. This illustrates how even low expenses add drag to a portfolio.

Results

Conclusions and Discussions

References

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

Appendix A - Code

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]), caption="Best Sharpe Ratio",
      row.names = F,digits=2)

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

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

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

#
# Show histograms with base VFIAX case
#
```

```r
par(mfrow=c(3,1), par=rep(4,2))

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

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

# Sharpe Ratio
hist(funds$Sharpe,main="Sharpe Ratio (all funds)",xlab="")
abline(v=funds[funds$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 = (tot - funds[funds$Ticker=="VFIAX",]$totalRet)/ funds[funds$Ticker=="VFIAX",]$totalRet
spyOverVFINX = (tot - funds[funds$Ticker=="VFINX",]$totalRet)/ funds[funds$Ticker=="VFINX",]$totalRet
pctOverVFIAX = round(spyOverVFIAX / expectedVFIAX * 100,1)
pctOverVFINX = round(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")
lines(Ad(fundData$VFIAX),col="blue")
lines(Ad(fundData$VFINX),col="red")
legend("bottomright",c("SPY","VFIAX","VFINX"),fill=c("black","blue","red"))
##
##
```

Caching Stock Data

Stock data was downloaded and cached with the following code:

```r
# 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-vers
        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 , "/", maxretrya
        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))
```