The Buffet Bet - Basket of Funds vs. All Passive

Stats 107 Final Project

David Wihl

December 6, 2016

Abstract

Type: Simulating a trading strategy. Eight years ago, Warren Buffett made a famous bet: the ultimate passive investment, the Vanguard S&P 500 Index Admiral Mutual Fund (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 the 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`. Any fund with less than five years of history was eliminated (VCOBX, VCORX, VDADX, VDVIX).

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

A list was obtained of the 168 currently offered Vanguard mutual funds including published expense ratios. The list was converted into a CSV for easy import into R.

Once the list of funds was imported, `quantmod` was used to read in historical daily adjusted data for the entire date range available from Yahoo. Since the earliest available date was November 13, 2000, an even 16 year end period was chosen, ending November 13, 2016. While more data would have been preferable, it was 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), the data was cached 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, it was determined that `quantmod` does not use adjusted price to calculate returns. Without explicitly using adjusted pricing, the default calculation is wrong. The was first discovered by going through the quantmod sources and then confirming at http://stackoverflow.com/questions/34772616/r-function-periodreturn-not-computing-returns-using-adjusted-closing-prices As an extra precaution, values were adjusted upon retrieval 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
VMMSX	Vanguard Emerging Mkts Select Stock Inv	0.93	-0.01	0
VTAPX	Vanguard Shrt-Term Infl-Prot Sec Idx Adm	0.08	0.00	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

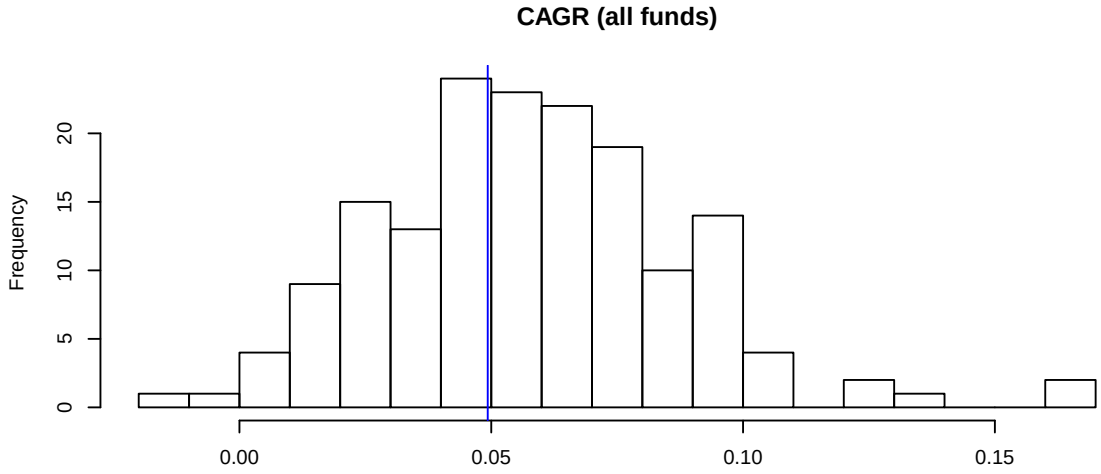
Table 4: Worst CAGR

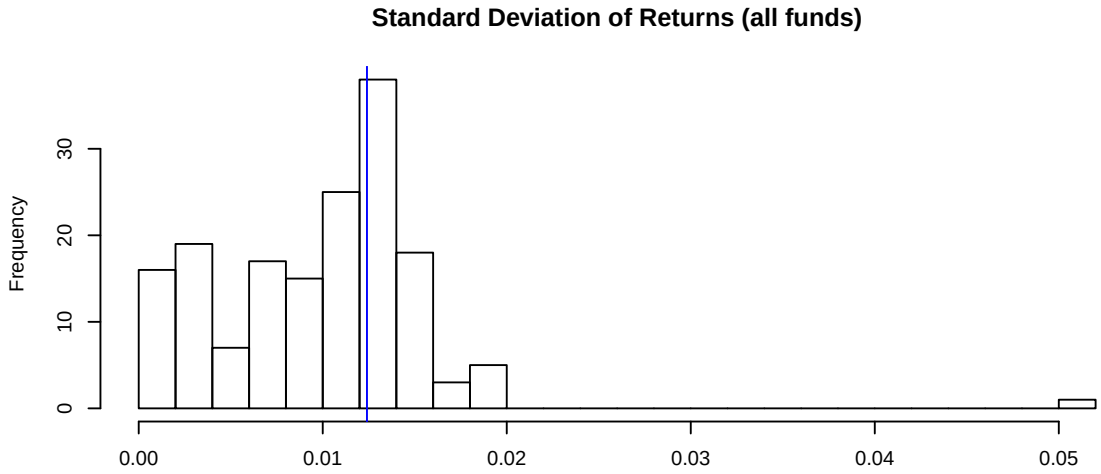
Ticker	Fund.Name	Expenses	CAGR	Sharpe
VMMSX	Vanguard Emerging Mkts Select Stock Inv	0.93	-0.01	0
VTIPX	Vanguard Shrt-Term Infl-Prot Sec Idx Inv	0.17	0.00	0
VTAPX	Vanguard Shrt-Term Infl-Prot Sec Idx Adm	0.08	0.00	0

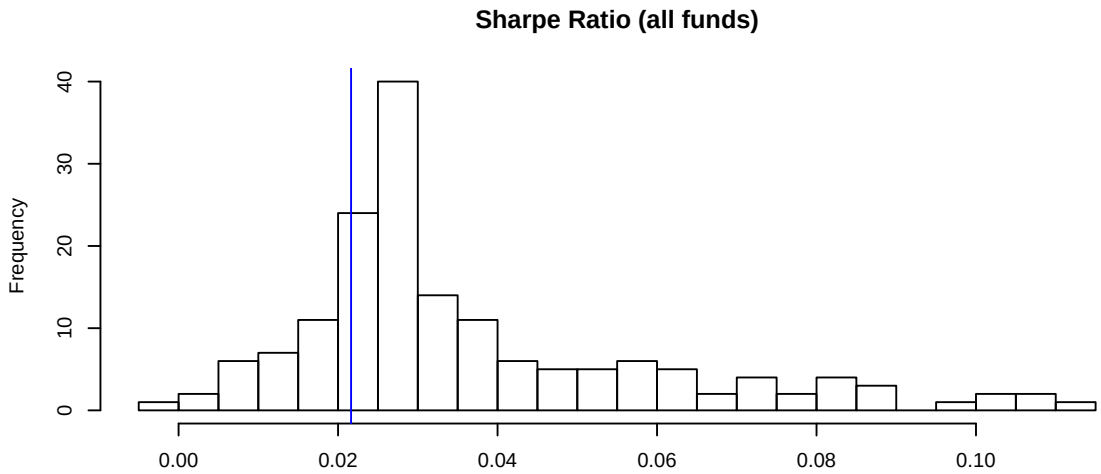
VMMSX is particularly bad: high expenses, low returns and low Sharpe Ratio.

2.3 Evaluation of VFIAX

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 than VFIAX as a combination of higher CAGR and lower volatility.

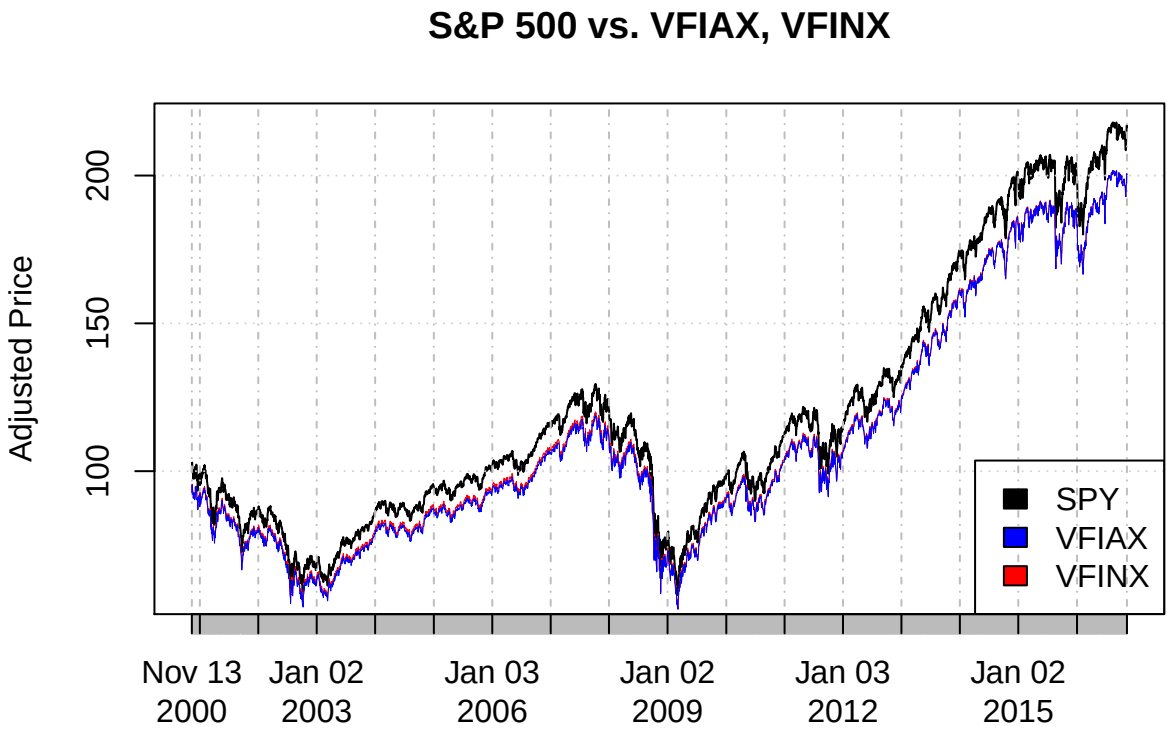
2.4 Expenses

Every mutual fund has expense ratios which is the principle way that a company like Vanguard makes revenue. 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 and will likely continue to do so. 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
- may not be an optimal choice. We should be able to do better.

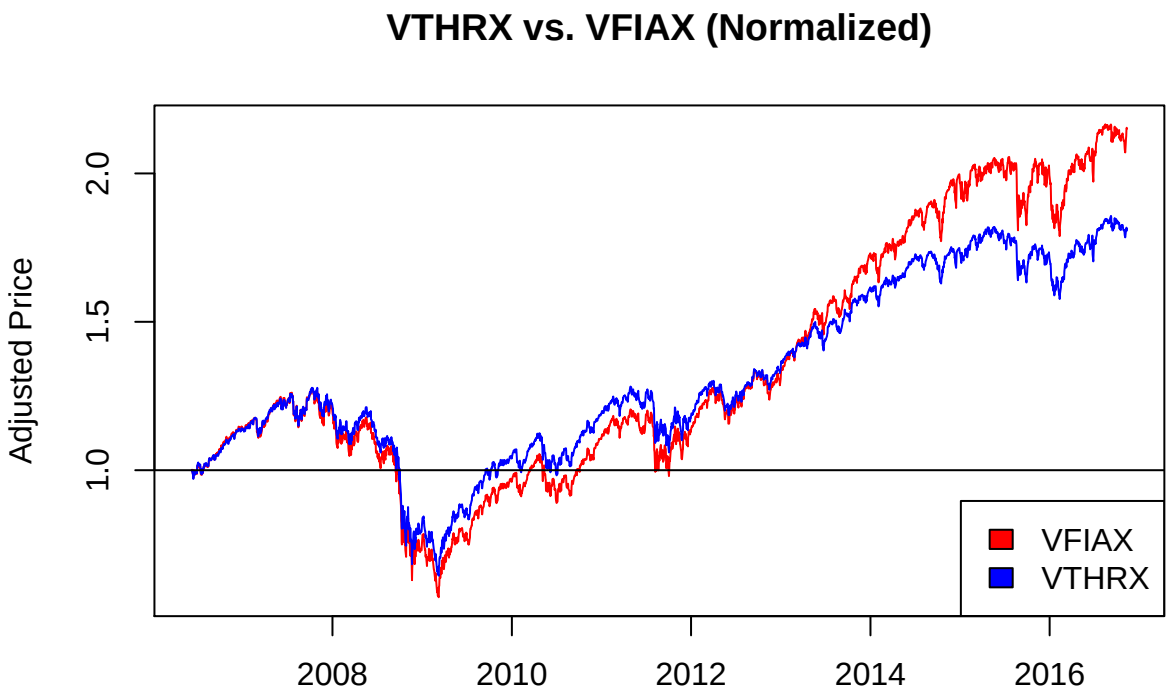
We will now iterate through a number of trading strategies to determine which trading models can best achieve this potential.

3.1 Age Based Balanced Fund

The default mutual fund for many Vanguard-based retirement plans is age adjusted balanced fund, such as VTHRX. Since VTHRX is a newer fund, having started on 2006-06-07, we will calculate VFIAX performance over the same time period.

Metric	VFIAX	VTHRX
Expense Ratio	0.05%	0.15%
CAGR	7.61%	5.83%

Not looking so good: lower CAGR and higher expenses.



The age adjusted balanced fund slightly outperforms the market during downturns, but over a longer time period significantly underperforms the market.

3.2 Better Single Fund

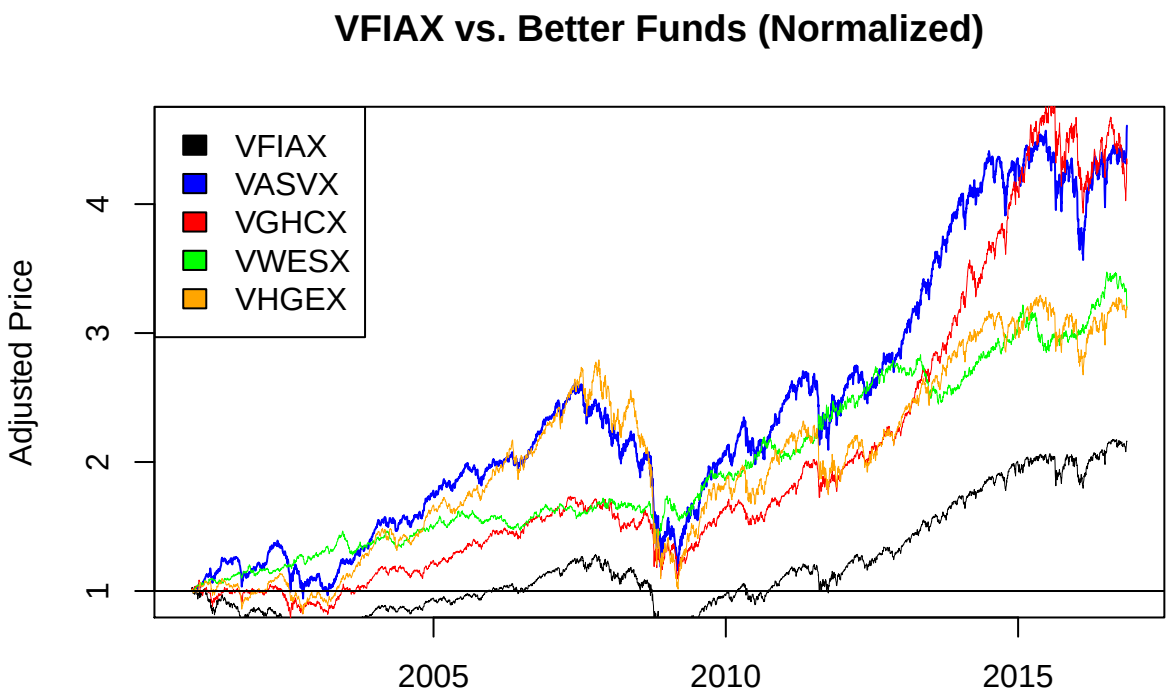
From the selection of funds, is there a fund that outperforms the market by 20%, has been in existance at least 10 years and has lower volatility?

It turns out that there are 26 out of 164 that meet these criteria. Here are the top ten:

Table 7: Top 10 Overall better funds than VFIAX

Ticker	Fund.Name	startDate	Expenses	CAGR	stdDev
VASVX	Vanguard Selected Value Inv	2000-11-13	0.39	0.100	0.012
VCSAX	Vanguard Consumer Staples Index Adm	2004-01-30	0.10	0.097	0.009
VGHAX	Vanguard Health Care Adm	2001-11-12	0.31	0.096	0.010
VGHCX	Vanguard Health Care Inv	2000-11-13	0.36	0.096	0.010
VUIAX	Vanguard Utilities Index Adm	2004-02-02	0.10	0.092	0.011
VHCIX	Vanguard Health Care Index Adm	2004-02-02	0.10	0.091	0.011
VWESX	Vanguard Long-Term Investment Grade Inv	2000-11-13	0.21	0.075	0.007
VHGEX	Vanguard Global Equity Inv	2000-11-13	0.57	0.075	0.012
VLACX	Vanguard Large-Cap Index Inv	2004-01-30	0.20	0.075	0.012
VBLTX	Vanguard Long-Term Bond Index Inv	2000-11-13	0.16	0.073	0.006
VFIAX	Vanguard 500 Index Adm	2000-11-13	0.05	0.049	0.012

Let's plot a selection:



We've discovered that approximately 15% of funds consistently beat the market over a 16 year period across a large number of different asset classes: Health Care Sector, Consumer Staples Sector, even several bond funds.

Should the entire portfolio be put into one of these such as VASVX? Even though the volatility is lower, this would not be wise. Let's now examine more complex strategies that involve diversification to see if risks can be lowered even further at while potentially increasing in CAGR.

3.3 Diversified Static Set of Funds

Vanguard and many other consumer general advice experts recommend a diversified set of static funds. Using Vanguard supplied tools on their website, the following simulation was run using a mid-career individual saving for retirement:

- Time to invest before withdrawl: $\geq$ 15 years
- Money will be withdrawn: $\geq$ 15 years
- Medium risk tolerance
- Medium likelihood of selling riskier assets during market declines
- Medium volatility
- Current and future income sources will be somewhat stable (3 on a scale of 5)
- \$100,000 to invest

The Vanguard calculator came back with the following recommendations:

Weight	Ticker
36%	VTSMX
24%	VGTSX
28%	VBMFX
12%	VTIBX

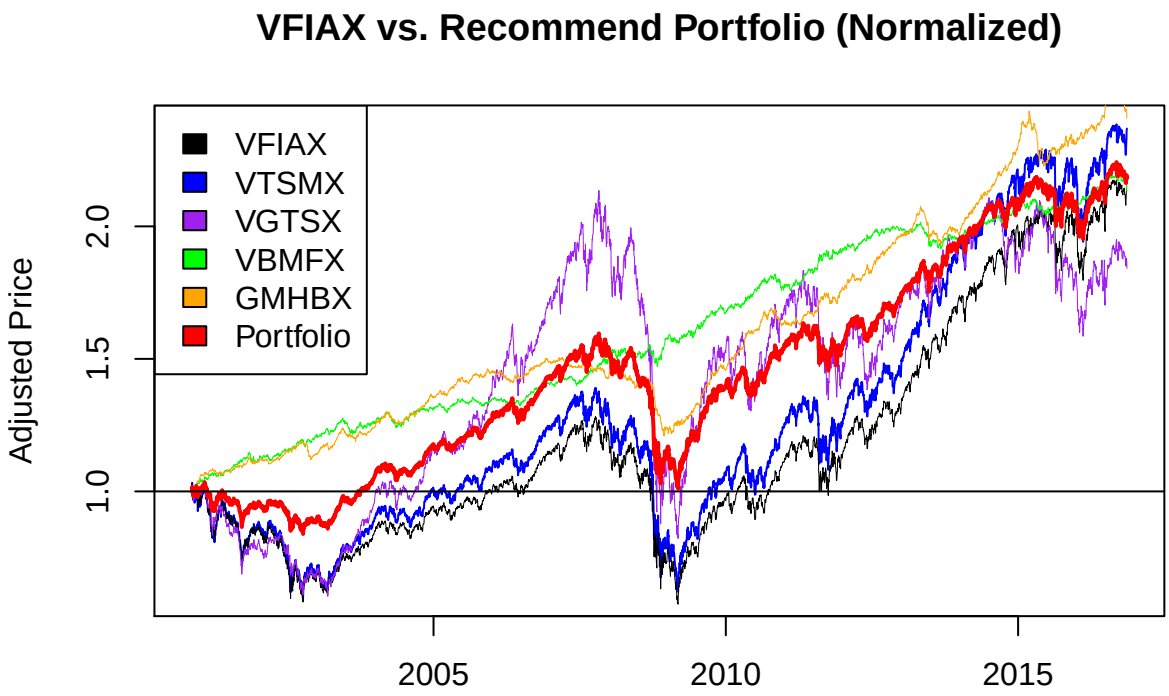
Unfortunately VTIBX, the Total International Bond Index Fund Investor shares has only been available since June 2013. Using Morningstar, an equivalent was found that had a sixteen year history: GMHBX, GMO Currency Hedged International Bond Fund Class III (GMO funds are unlikely to be available to a small investor so this is a best case scenario for Vanguard suggestions.)

The portfolio risk is 0.0071 which is certainly better than VFIAX's 0.0124

Table 9: Vanguard Recommended Portfolio Growth

	Start Value	End Value	CAGR
VTSMX	36000	85286.20	5.54
VGTSX	24000	44202.76	3.89
VBMFX	28000	59709.69	4.85
GMHBX	12000	28895.54	5.64
Total Portfolio	100000	218094.20	4.99
VFIAX	100000	216024.87	4.93

The recommended portfolio provide effectively the same returns as the market, albeit at lower volatility. Note from the plot below that the portfolio had a better return than the market for the majority of the period.



3.4 Efficient Frontier - Analysis of Vanguard Suggestions

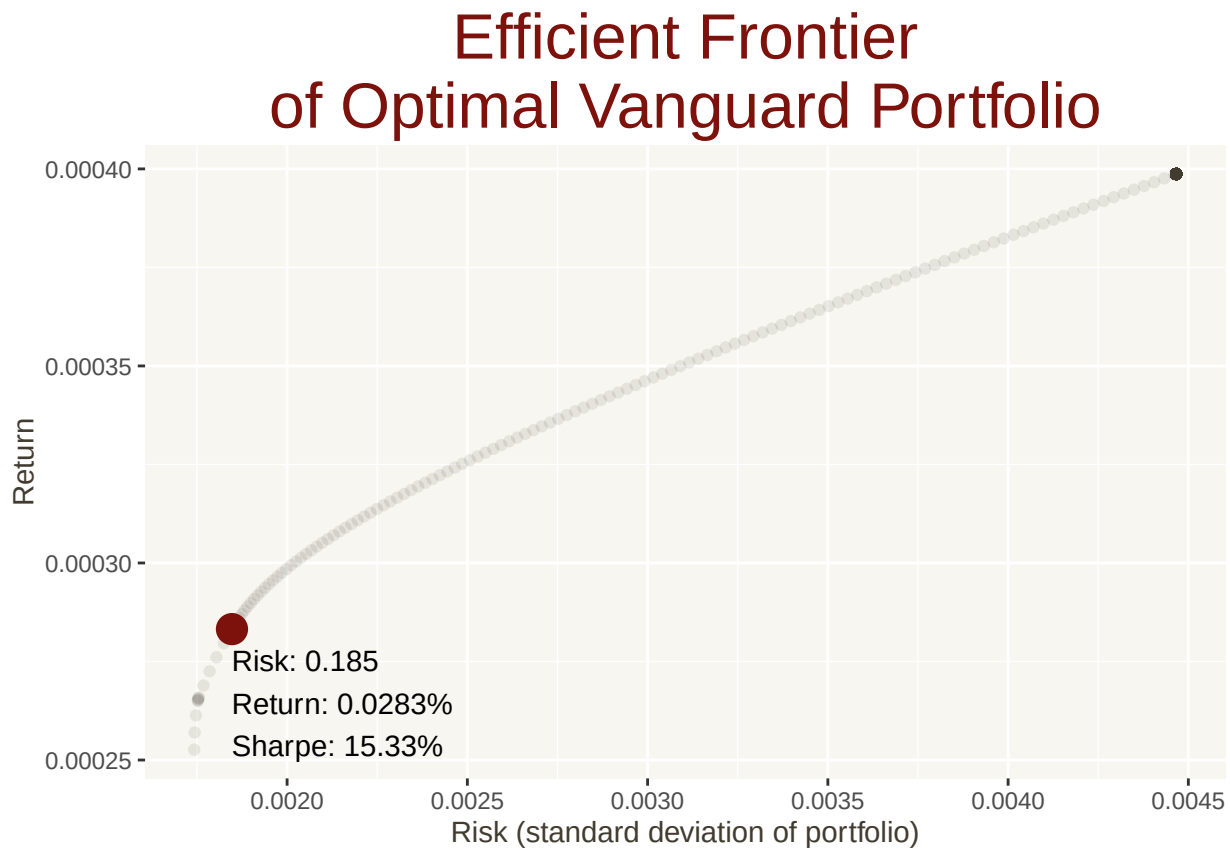
We now perform style analysis of a sort. By using an efficient frontier calculation, we determine if the weights suggested by Vanguard's calculator are optimal by backtesting on three years of daily returns. The maximum weight per security is capped at 50%.

The resulting weights are fairly different than Vanguard:

Ticker	Vanguard Suggested	EF Optimal Point	EF Min Variance
VTSMX	36%	11%	6%
VGTSX	24%	0%	4%
VBMFX	28%	40%	50%
GMHBX	12%	49%	40%
Overall Portfolio Risk	0.0071	0.0018	0.0017
Expected Return	4.99%	10.89%	9.66%

By using the efficient frontier calculations, at the optimal point annual returns have doubled and risk has decreased significantly. Vanguard does not specify how their weights are calculated.

Backtesting is an example of overfitting, as the optimal weights are determined for the period to be examined. The conclusion is not that the efficient frontier is vastly superior to Vanguard's recommendations, but rather to rebalance on a regular basis, rather than choosing a fixed set of funds and weights over the long term.



3.5 CAPM / Sharpe Ratio / Efficient Frontier

As highlighted earlier in this paper, there are funds such as VSIAX which offer CAGR three times the market albeit at higher risk (See Table 3). By using CAPM, a selection of funds will be made to maximize returns while minimizing risk.

4 Future Ideas

This paper has illustrated several techniques to beating the market over the long term using well known trading techniques. It would be interesting to explore more complex novel techniques that apply modern Machine Learning methods. One area particularly worth exploring would be Reinforcement Learning which is well suited to problems such as maximizing returns while minimizing risks. Furthermore, additional factors like restrictions on number of trades, and cost of trades are relatively easy to model.

5 Conclusions and Discussions

In summary: VFIAX vs:

Alternative	Result	Comment
Age Balanced	Loser	Lower CAGR, higher expenses
Single Alternative Fund	Winner	Higher return, but possibly higher risk

Alternative	Result	Comment
Diversified Static	Winner	Same return, significantly lower risk

This illustrates that there are a number of different techniques of besting the market over the long term, either in terms of higher returns, lower risk or both.

6 References

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

Ang, Clifford S., *Analyzing Financial Data and Implementing Financial Models Using R*, Springer 2015, ISBN 978-3-319-14075-9 (eBook)

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

Efficient Frontier calculation: http://economistatlarge.com/portfolio-theory/r-optimized-portfolio/r-code-graph-efficient-frontier

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

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

7 Appendix A - Code

7.1 Preparing Report

The following is the code used to prepare this report:

```
# Load libraries
library(ggplot2)
library(quantmod)
library(knitr)
library(stockPortfolio) # Alternative to quantmod
library(reshape2)
library(quadprog) # Quadratic programming library


# dot product
"%.%" <- function(x,y) sum(x*y)
# Calculate percent ready for display
percent <- function(x, digits = 2, format = "f", ...) {
  paste0(formatC(100 * x, format = format, digits = digits, ...), "%")
}

# Constants
fromDate = "2000-11-13"
toDate   = "2016-11-13"
startCash = 100000
#
# 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"])
}
funds    = read.csv("data/vanguard.csv",header = T, stringsAsFactors = F)
# Load from previously cached download
stockDataEnv= loadRData("data/stockData.RData")
fundData = mget(funds$Ticker,stockDataEnv)
# Clean some specific missing or extra values
fundData$VBMFX = fundData$VBMFX[index(fundData$VBMFX) != '2001-11-22']

#
# 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[])))
    funds$startPrice[i] = as.numeric(Ad(data[1]))
    funds$endPrice[i] = as.numeric(Ad(last(data[])))
    funds$totalRet[i] = (funds$endPrice[i] - funds$startPrice[i]) / funds$startPrice[i]
    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] = ((funds$endPrice[i] / funds$startPrice[i])^(1/years)) - 1

  }
}
# Store some VFIAX data in separate variables for convenience
VFIAX.CAGR = funds[funds$Ticker=="VFIAX","CAGR"]
VFIAX.stdDev = funds[funds$Ticker=="VFIAX","stdDev"]
VFIAX.startPrice = funds[funds$Ticker=="VFIAX","startPrice"]
VFIAX.endPrice = funds[funds$Ticker=="VFIAX","endPrice"]
VFIAX.Sharpe = funds[funds$Ticker=="VFIAX","Sharpe"]
VFIAX.totReturn =funds[funds$Ticker=="VFIAX","totalRet"]

#
# 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)

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

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

# Sharpe Ratio
hist(funds$Sharpe,main="Sharpe Ratio (all funds)",xlab="",breaks=20)
abline(v=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")
lines(Ad(fundData$VFINX),col="red",lwd=.1)
lines(Ad(fundData$VFIAX),col="blue",lwd=.5)
legend("bottomright",c("SPY","VFIAX","VFINX"),fill=c("black","blue","red"))
#
# Compare VFIAX to VTHRX over equivalent time periods.
#
```

```r
# Since VTHRX is younger than VFIAX, recalculate CAGR on matching dates
sDate = as.Date(funds[funds$Ticker=="VTHRX",]$startDate)
startPrice = as.numeric(Ad(fundData$VFIAX[sDate]))
endPrice = as.numeric(Ad(last(fundData$VFIAX[])))
totRet = (endPrice - startPrice) / startPrice
years = as.numeric((as.Date(funds[funds$Ticker=="VTHRX",]$endDate) -
                          as.Date(funds[funds$Ticker=="VTHRX",]$startDate)))/365
CAGR = ((endPrice / startPrice)^(1/years)) - 1
f1 = as.numeric(Ad(fundData$VFIAX[paste0(sDate,"::"),])) / startPrice
f2 = as.numeric(Ad(fundData$VTHRX)) / as.numeric(Ad(fundData$VTHRX[1]))
plot(index(fundData$VTHRX),f1,type="l",main="VTHRX vs. VFIAX (Normalized)",xlab="",ylab="Adjusted Price
lines(index(fundData$VTHRX),f2,col="blue",lwd=1)
abline(h=1)
legend("bottomright",c("VFIAX","VTHRX"),fill=c("red","blue"))
#
# Find a better single fund that beats the market with higher CAGR and lower volatility
#
betterFunds = c("VFIAX")
for (i in 1:nrow(funds)) {
  if (funds$CAGR[i] > (1.2 * VFIAX.CAGR)) {
    if (funds$stdDev[i] < VFIAX.stdDev) {
      if (funds$startDate[i] < as.Date("2006-01-01")) {
        #cat(funds$Ticker[i]," ",funds$Fund.Name[i],"\n")
        betterFunds = c(betterFunds, funds$Ticker[i])
      }
    }
  }
}

colsToDisplay = c("Ticker","Fund.Name","startDate","Expenses","CAGR","stdDev")
bf = funds[funds$Ticker %in% betterFunds,colsToDisplay]
bf$startDate = as.Date(bf$startDate)
kable(rbind(head(bf[order(-bf$CAGR),],n=10), tail(bf[order(-bf$CAGR),],n=1)),
        caption="Top 10 Overall better funds than VFIAX",
      row.names = F,digits=3)

f1 = as.numeric(Ad(fundData$VFIAX)) / as.numeric(Ad(fundData$VFIAX[1]))
f2 = as.numeric(Ad(fundData$VASVX)) / as.numeric(Ad(fundData$VASVX[1]))
f3 = as.numeric(Ad(fundData$VGHCX)) / as.numeric(Ad(fundData$VGHCX[1]))
f4 = as.numeric(Ad(fundData$VWESX)) / as.numeric(Ad(fundData$VWESX[1]))
f5 = as.numeric(Ad(fundData$VHGEX)) / as.numeric(Ad(fundData$VHGEX[1]))

plot(index(fundData$VFIAX),f2,type="l",main="VFIAX vs. Better Funds (Normalized)",xlab="",ylab="Adjuste
lines(index(fundData$VFIAX),f1,col="black",lwd=0.5)
lines(index(fundData$VFIAX),f3,col="red",lwd=0.5)
lines(index(fundData$VFIAX),f4,col="green",lwd=0.5)
lines(index(fundData$VFIAX),f5,col="orange",lwd=0.5)
abline(h=1)
legend("topleft",c("VFIAX","VASVX","VGHCX","VWESX","VHGEX"),fill=c("black","blue","red","green","orange
#
# Compare against a diversified set of funds recommended by Vanguard
#
suggestions = c("VTSMX","VGTSX","VBMFX","GMHBX")
```

```
getSymbols("GMHBX", src="yahoo", from=as.Date(fromDate),to=toDate,adjust=T)
weights = c(.36,.24,.28,.12)
# Build covariance matrix
returns = cbind(dailyReturn(fundData$VTSMX),dailyReturn(fundData$VGTSX),dailyReturn(fundData$VBMFX),dai
# delete one extra row from VBMFX
returns = returns[index(returns) !='2001-11-22']
names(returns) = suggestions
cov.mat = cov(returns)  # annualized
risk.p = sqrt(t(weights) %*% cov.mat %*% weights)
shares = rep(0,length(suggestions))
startValue = rep(0,length(suggestions))
endValue = rep(0,length(suggestions))
CAGR = rep(0,length(suggestions))
# Find total return on three known funds
for (i in 1:length(suggestions)-1) {
  startValue[i] = startCash * weights[i]
  shares[i] = startValue[i] / funds[funds$Ticker==suggestions[i],"startPrice"]
  endValue[i] = shares[i] * funds[funds$Ticker==suggestions[i],"endPrice"]
}
# special case for GMHBX
startValue[4] = startCash * weights[4]
shares[4] = startCash * weights[4] / as.numeric(Ad(first(GMHBX)))
endValue[4] = shares[4] * Ad(last(GMHBX))
years =  as.numeric((as.Date(funds[funds$Ticker=="VFIAX","endDate"]) -
                          as.Date(funds[funds$Ticker=="VFIAX","startDate"])))/365
for (i in 1:length(suggestions)) {
  CAGR[i] = 100*(((endValue[i] / startValue[i])^(1/years)) - 1)
}
port = cbind(startValue, endValue,CAGR)
port.CAGR = 100*((sum(endValue) / startCash)^(1/years) - 1)
port = rbind(port, c(sum(startValue), sum(endValue), port.CAGR ))
port = rbind(port, c(startCash,startCash * (1+VFIAX.totReturn), VFIAX.CAGR*100))
rownames(port) = c(suggestions,"Total Portfolio","VFIAX")
colnames(port) = c("Start Value","End Value","CAGR")
port.prices = cbind(as.numeric(Ad(fundData$VTSMX)), as.numeric(Ad(fundData$VGTSX)),
                    as.numeric(Ad(fundData$VBMFX)), as.numeric(Ad(GMHBX)))

port.value = sweep(port.prices, MARGIN=2, shares,'*')
kable(port, caption="Vanguard Recommended Portfolio Growth", digits=2)

f1 = as.numeric(Ad(fundData$VFIAX)) / as.numeric(Ad(fundData$VFIAX[1]))
f2 = port.prices[,1] / port.prices[1,1]
f3 = port.prices[,2] / port.prices[1,2]
f4 = port.prices[,3] / port.prices[1,3]
f5 = port.prices[,4] / port.prices[1,4]
f6 = rowSums(port.value) / startCash

plot(index(fundData$VFIAX),f2,type="l",main="VFIAX vs. Recommend Portfolio (Normalized)",xlab="",ylab="
lines(index(fundData$VFIAX),f1,col="black",lwd=0.5)
lines(index(fundData$VFIAX),f3,col="purple",lwd=0.5)
lines(index(fundData$VFIAX),f4,col="green",lwd=0.5)
lines(index(fundData$VFIAX),f5,col="orange",lwd=0.5)
lines(index(fundData$VFIAX),f6,col="red",lwd=2.0)
```

```r
abline(h=1)
legend("topleft",c("VFIAX","VTSMX","VGTSX","VBMFX","GMHBX","Portfolio"),fill=c("black","blue","purple",

ef.startdate = "2013-09-01"
ef.enddate = "2016-09-01"

ef.stocks <-c(
  "VTSMX" = .36,
  "VGTSX" = .24,
  "VBMFX" = .28,
  "GMHBX" = .12
  )

ef.stockReturns <- stockPortfolio::getReturns(names(ef.stocks), freq ="day",get="overlapOnly", start = 


eff.frontier <- function (returns, short="no", max.allocation=NULL,
 risk.premium.up=.5, risk.increment=.005){
 # return argument should be a m x n matrix with one column per security
 # short argument is whether short-selling is allowed; default is no (short
 # selling prohibited)max.allocation is the maximum % allowed for any one
 # security (reduces concentration) risk.premium.up is the upper limit of the
 # risk premium modeled (see for loop below) and risk.increment is the
 # increment (by) value used in the for loop

 covariance <- cov(returns)
 print(covariance)
 n <- ncol(covariance)

 # Create initial Amat and bvec assuming only equality constraint
 # (short-selling is allowed, no allocation constraints)
 Amat <- matrix (1, nrow=n)
 bvec <- 1
 meq <- 1

 # Then modify the Amat and bvec if short-selling is prohibited
 if(short=="no"){
 Amat <- cbind(1, diag(n))
 bvec <- c(bvec, rep(0, n))
 }

 # And modify Amat and bvec if a max allocation (concentration) is specified
 if(!is.null(max.allocation)){
 if(max.allocation > 1 | max.allocation <0){
 stop("max.allocation must be greater than 0 and less than 1")
 }
 if(max.allocation * n < 1){
 stop("Need to set max.allocation higher; not enough assets to add to 1")
 }
 Amat <- cbind(Amat, -diag(n))
 bvec <- c(bvec, rep(-max.allocation, n))
 }
```

```
 # Calculate the number of loops
 loops <- risk.premium.up / risk.increment + 1
 loop <- 1

 # Initialize a matrix to contain allocation and statistics
 # This is not necessary, but speeds up processing and uses less memory
 eff <- matrix(nrow=loops, ncol=n+3)
 # Now I need to give the matrix column names
 colnames(eff) <- c(colnames(returns), "Std.Dev", "Exp.Return", "sharpe")

 # Loop through the quadratic program solver
 for (i in seq(from=0, to=risk.premium.up, by=risk.increment)){
 dvec <- colMeans(returns) * i # This moves the solution along the EF
 sol <- solve.QP(covariance, dvec=dvec, Amat=Amat, bvec=bvec, meq=meq)
 eff[loop,"Std.Dev"] <- sqrt(sum(sol$solution*colSums((covariance*sol$solution))))
 eff[loop,"Exp.Return"] <- as.numeric(sol$solution %*% colMeans(returns))
 eff[loop,"sharpe"] <- eff[loop,"Exp.Return"] / eff[loop,"Std.Dev"]
 eff[loop,1:n] <- sol$solution
 loop <- loop+1
 }

 return(as.data.frame(eff))
}

# Run the eff.frontier function based on no short and 50% alloc. restrictions
eff <- eff.frontier(returns=ef.stockReturns$R, short="no", max.allocation=.50,
 risk.premium.up=1, risk.increment=.001)

# Find the optimal portfolio
eff.optimal.point <- eff[eff$sharpe==max(eff$sharpe),]
eff.min.variance = eff[eff$Std.Dev==min(eff$Std.Dev),]

# graph efficient frontier
# Start with color scheme
ealred <- "#7D110C"
ealtan <- "#CDC4B6"
eallighttan <- "#F7F6F0"
ealdark <- "#423C30"

ggplot(eff, aes(x=Std.Dev, y=Exp.Return)) + geom_point(alpha=.1, color=ealdark) +
 geom_point(data=eff.optimal.point, aes(x=Std.Dev, y=Exp.Return, label=sharpe),
 color=ealred, size=5) +
 annotate(geom="text", x=eff.optimal.point$Std.Dev,
 y=eff.optimal.point$Exp.Return,
 label=paste("Risk: ",
 round(eff.optimal.point$Std.Dev*100, digits=3),"\nReturn: ",
 round(eff.optimal.point$Exp.Return*100, digits=4),"%\nSharpe: ",
 round(eff.optimal.point$sharpe*100, digits=2), "%", sep=""),
 hjust=0, vjust=1.2) +
 ggtitle("Efficient Frontier\nof Optimal Vanguard Portfolio") +
 labs(x="Risk (standard deviation of portfolio)", y="Return") +
 theme(panel.background=element_rect(fill=eallighttan),
 text=element_text(color=ealdark),
```

```
 plot.title=element_text(size=24, color=ealred))
## Minimum Variance
# The original Buffet Bet was the market vs. a hedge fund consisting of a basket of funds. In retrospec
# it is not surprising that the market is beating the hedge fund. The primary purpose of the hedge fund
# is the eponymous hedging - that is reducing risk. Hedge funds outperform the market during downtimes
# but underperform over the long term. Their primary raison d'être is captial preservation. This strate
# will be mimicked by selecting a minimum variance portfolio. The portfolio will be compared against
# market downturn periods vs. long term.

#source('zivot_code.R')
#
# Minimum Variance portfolio
#
# We are now going to use the Zivot code to also build the efficient frontier
#
# nFunds = nrow(funds)
# cov.mat = matrix(0,nrow=nFunds, ncol=nFunds)
# colnames(cov.mat) = funds$Ticker
# rownames(cov.mat) = funds$Ticker
#
# # We have to build the variance-covariance matrix manually rather than using
# # the cov() function because the date lengths are not the same.
# # This code is not efficient as each covariance is calculated twice. It still takes
# # just a few seconds to run.
# # TODO: move this into the caching R script
# for (i in 1:nFunds) { # nrow(funds)
#   for (j in 1:nFunds) {
#     # start with later date of either
#     sDate = as.Date(max(funds$startDate[i],funds$startDate[j]))
#     r.i = eval(parse(text = paste("fundData$", funds$Ticker[i], sep = "")))
#     r.i = dailyReturn(Ad(r.i[paste0(sDate,"::"),]))
#     r.j = eval(parse(text = paste("fundData$", funds$Ticker[j], sep = "")))
#     r.j = dailyReturn(Ad(r.j[paste0(sDate,"::"),]))
#     # http://r.789695.n4.nabble.com/Finding-the-correlation-coefficient-of-two-stocks-td3246992.html
#     m = merge(r.i,r.j)
#     cov.mat[i,j] = cov(m[,1],m[,2],use="pairwise.complete.obs")
#   }
# }
#
# TODO. Should we use Zivot? Is there a better way? How can we find optimal weights?
#returns = cbind(dailyReturn(fundData$VTSMX),dailyReturn(fundData$VGTSX),dailyReturn(fundData$VBMFX),da
#names(returns) = tickers
#risk.p = sqrt(t(weights) %*% cov.mat %*% weights)
# returns = c()
# for (i in 1:nFunds) { # nrow(funds)
#   r = eval(parse(text = paste("fundData$", funds$Ticker[i], sep = "")))
#   returns = cbind(returns, dailyReturn(Ad(r)))
# }
# colnames(returns) = funds$Ticker
# cov.mat2 = cov(returns,use="pairwise.complete.obs")
# colnames(cov.mat2) = funds$Ticker
# rownames(cov.mat2) = funds$Ticker
#
```

```
# cor.mat = cor(returns,use="pairwise.complete.obs")
# colnames(cor.mat) = funds$Ticker
# rownames(cor.mat) = funds$Ticker
# source('zivot_code.R')
# rk.free = 0.005
# er = funds$CAGR
# gmin.port = globalMin.portfolio(er,cov.mat)
# summary(gmin.port, risk.free = rk.free)
##
##
```

7.2 Caching Stock Data

Stock data was downloaded and cached with the following code:

```
# 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))
```