# Portfolio starter kit

 January 25, 2020

[R](https://osm.netlify.app/categories/r), [Stocks](https://osm.netlify.app/categories/stocks)

Say you’ve built a little nest egg thanks to some discipline and frugality. And now you realize that you should probably invest that money so that you’ve got something to live off of in retirement. Or perhaps you simply want to earn a better return than stashing your cash underneath your bed, I mean your savings account. How do you choose the assets? What amount of money should you put into each asset? Should you keep the same relative weighting or should you change it? And if you do change it, how frequently?

These questions and more are part of a larger project that academics, institutional investors, investment advisers, endowments, and financial planners grapple with daily: the portfolio construction process. There are no laws to this process—despite Nobel prize winning literature, scores of research, and thousands of practitioners analyzing the best way to build, optimize, and allocate portfolios—only theory and rules-of-thumb. That’s to be expected. Humans construct portfolios based on the flow of capital into, out of, and across markets driven by what humans are doing in response, or oblivious to, that flow of capital. If there were immutable laws of human nature, then there might be immutable laws of portfolio construction.

In this post, we head back to basics; so if you’re an experienced investor, you might want to stop here. We’re headed in your direction. How we get there might be a bit different since data science will guide the way. Indeed, our plan is to build the intuition behind constructing satisfactory portfolios, so that as we move forward, to discuss more complicated issues, we have a solid foundation that can address the difficulties in that process. As usual, we’ll use R to facilitate and to inform the analysis, hoping to build a model portfolio that melds theory and behavior.[**1**](https://osm.netlify.app/post/portfolio-starter-kit/#fn1)

## Starting assumptions

First, we need to start with a few assumptions. Don’t put all of your eggs in one basket. And if you don’t have any insights on a good asset, might as well own everything. The first one is diversification at its simplest. The second assumption is more nuanced, but works as follows. If you’re not Warren Buffett or George Soros, begin by accepting that you don’t know what the market, or any asset will do tomorrow let alone next year. If you don’t know how assets will perform in the future, then might as well own all of them. Hopefully, you’ll be lucky enough to ride the winners and the losers won’t hurt too much.

But, owning one unit of every single asset outright is impossible. Nor is there a way to own a fractional share of all risky assets globally.[**2**](https://osm.netlify.app/post/portfolio-starter-kit/#fn2) What will have to suffice is owning reasonable proxies of the big asset classes. Those classes are stocks, bonds, and real assets.

Most folks know what stocks and bonds are. Real assets are bit more nebulous. Commodities, real estate, and timber are all included. If relative homogeneity is somewhat necessary to define an asset class, real assets seem to be the exception to the rule. Real estate is the odd one out. But let’s not get too nit-picky.

Let’s assume you decide to own a reasonable proxy for each class. In this case, we’ll use liquid ETFs with a relatively long history: SPY (for the S&P 500), SHY (for short-term US treasuries) and GLD (for gold). Having no idea how much to allocate to each asset you simply buy equal amounts. And away you go.

But then you speak to your cousin, who tells you that you should have most of your assets in stocks and only a little bit, maybe 10% in bonds. Meanwhile, your risk averse co-worker, tells you that commodities are way too risky and that he owns a portfolio of 50/50 stocks and bonds.

So which one is right? Maybe data science can provide some answers. First, we gather a full data set as far back in history as is available (2005 in this case). Then we construct the portfolios with the target allocations mentioned above at inception and with no rebalancing. Don’t worry about rebalancing now. We’ll discuss that in a later post. The graph below shows your portfolio (Equal), along with your cousin’s (Risky) and your co-worker’s (Naive). Note that the naive portfolio is not called so to cast aspersions. Rather, it’s a portfolio that some pundits use as a base or comparison case. It’s naive because it implicitly takes no view on asset attractiveness. Why it doesn’t usually include real assets is beyond the scope of this post. In any event, here’s the graph.

Chart, histogram

Description automatically generated

So your cousin looks like a genius. But, for a while, you were crushing it, and, at least you’ve performed better than your co-worker. Water-cooler bragging rights are not something to scoff at! Let’s look at the average returns, risk (i.e., volatility), and risk-adjusted returns for each portfolio.

| Table 1: Annualized performance metrics | | | | |
| --- | --- | --- | --- | --- |
| **Asset** | **Mean (%)** | **Volatility (%)** | **Risk-adjusted (%)** | **Cumulative (%)** |
| Equal | 6.0 | 9.2 | 66.5 | 234.5 |
| Naive | 6.0 | 6.9 | 80.6 | 222.0 |
| Risky | 8.4 | 12.4 | 63.8 | 291.6 |

Even though your overall performance was better than your co-worker, average annualized performance was about the same. However, your volatility was much higher leading to worse risk-adjusted performance. Here risk-adjusted performance is simply the mean return divided by the volatility. Think of it as how much you earn relative to how much you risk. Higher is obviously better. Your cousin’s portfolio performed much better on a cumulative and average basis. But his risk-adjusted performance was even worse than yours.

Notice some interesting takeaways. Even though the Equal portfolio had the same average return as the Naive one, but worse risk-adjusted performance, it still enjoyed a better cumulative return. Which return should you focus on—average, risk-adjusted, or cumulative? When you retire what matters most is your cumulative return. However, you can’t have a high cumulative return without also having a high average return. Still, cumulative return can be less than another portfolio with virtually the same average return due to volatility and path dependency. One way to show these two together is to calculate what’s called semi-deviation. In other words, we only look at the deviation above or below the mean. We show that in the table below.

| Table 2: Annualized semi-deviation | | |
| --- | --- | --- |
| **Asset** | **Down volatility (%)** | **Up volatility (%)** |
| Equal | 9.6 | 8.8 |
| Naive | 8.1 | 5.7 |
| Risky | 15.0 | 10.1 |

From the preceding table we can see why the Equal portfolio outperformed the Naive portfolio. It’s upside volatility was much higher. The Risky portfolio’s volatility was too. We won’t want spend any more time on semi-deviation now; but we will come back to it.

Now that you’ve seen how the portfolios have performed, would you keep your current allocation or would you switch? If you’re conservative you might re-allocate to the Naive portfolio. But if you like a Vegas weekend every chance you get, you might favor the Risky portfolio instead. Or you could toss a coin. How can we answer that question by applying some data science? Simulation! Note that rigorous finance would recommend optimization. But we want to build the intuition first before launching into the math.

Let’s simulate a thousand random weights for the three assets and graph a scatter plot to visualize the range of outcomes. Here we assume the same return, volatility, and correlation as occurred historically. While correlation is important, we don’t want to get into the nuances of that just yet. For a more detailed discussion see our previous post [**Detour: correlation**](https://osm.netlify.com/post/diversification-2/).

When we graph the range of possible portfolios we see a wide spread of outcomes. The points at the top are said to “dominate” the points below because they offer a higher return for the same level of risk. The shape might seem a bit odd for folks used to looking at the “efficient frontier”. See the footnote for a more detailed discussion on this .[**3**](https://osm.netlify.app/post/portfolio-starter-kit/#fn3) Don’t worry if you don’t know what an efficient frontier is. Just appreciate the spread of outcomes.

Chart, scatter chart

Description automatically generated

The different colored dots represent the original three portfolios. We can see that none of the portfolios “dominate” each other. That is, none is directly above the other. But each portfolio is dominated by other simulated portfolios. For example, for the Naive portfolio (black dot), the maximum return for that level of risk is about 0.38 percentage points higher than the portfolio. For the Equal (red dot) and Risky (purple dot) portfolios, the maximum returns are 1.2 and 0.5 percentage points higher. A half a percentage point greater return on average might seem small, but over a ten-year period leads to slightly over a 5% higher cumulative return. Still, most people are not likely to worry about half a percent. A full percent, as in the Equal portfolio, might be worth considering.

What the graph shows is that there is a trade-off between risk and return. As returns go up, risk does too, as delineated by the smooth grey line, but the dispersion of returns also widens. If you accept more risk, you should generally enjoy a higher return, but the chance that you won’t also increases. Additionally, the graph allows you to dial in the range of outcomes for different slices of risk or return. If you want a certain level of return, you can find a portfolio that offers that for the least amount of risk. Or you can find a range of portfolios that get you close for a range of risk levels. Alternatively, if you want to accept only a certain level of risk, you can find the portfolio that offers you the greatest return. Of course, you’d have to chug through some calculations to get to the appropriate weights, but there’s nothing mysterious about the process.

For argument’s sake, suppose you’re happy with the current riskiness of your portfolio, but you’d clearly like to get a better return, what would you do? You find the maximum return for that level of risk, and then back into the implied weight. What would that be? We show the results in the graph below.

Chart, bar chart

Description automatically generated

The dominant portfolio for the same level of risk as the Equal portfolio has a 56% weight to the SPY ETF, a 16% weight in SHY and a 28% weight in GLD. To achieve a higher return you swap out of bonds and into stocks and add a little to gold.

Let’s think about this for a moment. Even if you don’t know anything about where stocks or bonds or other assets are going to go in the next year, you can probably assume that stocks are riskier than bonds, and gold is riskier than both. Moreover, even if you don’t know the direction of stocks, bonds, or gold, you probably could guess that it’s easier to understand why stocks might increase in price than why gold might. Stock price direction might be a coin flip, but gold? That’s probably an unfair coin.

Given that, would you be comfortable increasing your allocation to stocks or commodities? Think about raising your exposure to stocks by 26 percentage points for a modest one-to-two percentage point increase in average returns? Seems like a lot for a little. This brings us to what we call the first problem of portfolio construction: sometimes what appears to be the optimal weighting is not the most palatable.

Now this was an artificial case in the sense that we had only three assets, one of which was gold. And while these assets were meant to be reasonable proxies for their asset classes, it wouldn’t be hard to argue there are better proxies. And what is the right proxy for real assets? Nonetheless, the point is that even with some simplistic, and superficially reasonable assumptions, it is possible to produce recommendations that would be uncomfortable to implement or not particularly intuitive. Portfolio construction isn’t simply about complex models and fancy code, it’s also about common sense.

We’ll leave you with that thought. In successive posts, we’ll look at picking portfolios out of a range of inputs, which will hopefully be more intuitive. Additionally, we’ll look at constructing portfolios using semi-deviation or accounting for rebalancing, two topics we mentioned above. After that, we’ll focus on setting asset class expectations. If you’d like to see us touch on another topic or disagree with our analysis, send us an email at the address after the code. Speaking of which, here’s the code underlying all of the analysis, graphs, and tables.

# Load package

library(tidyquant)

library(tidyverse)

# Get data

symbols <- c("SPY", "EEM", "SHY", "IYR", "GLD")

symbols\_low <- tolower(symbols)

prices <- getSymbols(symbols, src = "yahoo",

from = "1990-01-01",

auto.assign = TRUE) %>%

map(~Ad(get(.))) %>%

reduce(merge) %>%

`colnames<-`(symbols\_low)

prices\_monthly <- to.monthly(prices, indexAt = "last", OHLC = FALSE)

ret <- ROC(prices\_monthly)["2005/2019"]

naive <- ret[,c("spy", "shy")]

basic <- ret[,c("spy", "shy", "gld")]

# Create different weights and portflios

wt1 <- rep(1/(ncol(basic)), ncol(basic))

port1 <- Return.portfolio(basic, wt1) %>%

`colnames<-`("ret")

wt2 <- c(0.9, 0.10, 0)

port2 <- Return.portfolio(basic, weights = wt2) %>%

`colnames<-`("ret")

wtn <- c(0.5, 0.5)

portn <- Return.portfolio(naive, wtn)

port\_comp <- data.frame(date = index(port1), equal = as.numeric(port1),

wtd = as.numeric(port2),

naive = as.numeric(portn))

port\_comp %>%

gather(key,value, -date) %>%

group\_by(key) %>%

mutate(value = cumprod(value+1)) %>%

ggplot(aes(date, value\*100, color = key)) +

geom\_line() +

scale\_color\_manual("", labels = c("Equal", "Naive", "Risky"),

values = c("blue", "black", "red")) +

labs(x = "",

y = "Index",

title = "Three portfolios, which is best?",

caption = "Source: Yahoo, OSM estimates") +

theme(legend.position = "top",

plot.caption = element\_text(hjust = 0))

# Portfolio summary table

port\_comp %>%

rename("Equal" = equal,

"Naive" = naive,

"Risky" = wtd) %>%

gather(Asset, value, -date) %>%

group\_by(Asset) %>%

summarise(`Mean (%)` = round(mean(value, na.rm = TRUE),3)\*1200,

`Volatility (%)` = round(sd(value, na.rm = TRUE)\*sqrt(12),3)\*100,

`Risk-adjusted (%)` = round(mean(value, na.rm = TRUE)/sd(value, na.rm=TRUE)\*sqrt(12),3)\*100,

`Cumulative (%)` = round(prod(1+value, na.rm = TRUE),3)\*100) %>%

knitr::kable(caption = "Annualized performance metrics")

# Semi-deviation functions

down\_dev <- function(vec){

mean\_vec <- mean(vec, na.rm = TRUE)

down\_vec <- vec[vec < mean\_vec]

dev <- sqrt(mean((down\_vec - mean\_vec)^2))

dev

}

up\_dev <- function(vec){

mean\_vec <- mean(vec, na.rm = TRUE)

up\_vec <- vec[vec > mean\_vec]

dev <- sqrt(mean((up\_vec - mean\_vec)^2))

dev

}

# Semi-deviation table

port\_comp %>%

rename("Equal" = equal,

"Naive" = naive,

"Risky" = wtd) %>%

gather(Asset, value, -date) %>%

group\_by(Asset) %>%

summarise(`Down volatility (%)` = round(down\_dev(value)\*sqrt(12),3)\*100,

`Up volatility(%)` = round(up\_dev(value)\*sqrt(12),3)\*100) %>%

knitr::kable(caption = "Annualized performance metrics")

## Portfolio simulations

# Portfolio

mean\_ret <- apply(ret[,c("spy", "shy", "gld")],2,mean)

cov\_port <- cov(ret[,c("spy", "shy", "gld")])

port\_exam <- data.frame(ports = colnames(port\_comp)[-1],

ret = as.numeric(apply(port\_comp[,-1],2, mean)),

vol = as.numeric(apply(port\_comp[,-1], 2, sd)))

# Weighting that ensures more variation and random weighthing to stocks

set.seed(123)

wts <- matrix(nrow = 1000, ncol = 3)

for(i in 1:1000){

a <- runif(1,0,1)

b <- c()

for(j in 1:2){

b[j] <- runif(1,0,1-sum(a,b))

}

if(sum(a,b) < 1){

inc <- (1-sum(a,b))/3

vec <- c(a+inc, b+inc)

}else{

vec <- c(a,b)

}

wts[i,] <- sample(vec,replace = FALSE)

}

# Calculate random portfolios

port <- matrix(nrow = 1000, ncol = 2)

for(i in 1:1000){

port[i,1] <- as.numeric(sum(wts[i,] \* mean\_ret))

port[i,2] <- as.numeric(sqrt(t(wts[i,] %\*% cov\_port %\*% wts[i,])))

}

colnames(port) <- c("returns", "risk")

port <- as.data.frame(port)

# Graph with points

port %>%

ggplot(aes(risk\*sqrt(12)\*100, returns\*1200)) +

geom\_point(color = "blue", size = 1.2, alpha = 0.4) +

geom\_smooth(method = "loess", formula = y ~ log(x), se = FALSE, color = "slategrey") +

geom\_point(data = port\_exam, aes(port\_exam[1,3]\*sqrt(12)\*100,

port\_exam[1,2]\*1200),

color = "red", size = 6) +

geom\_point(data = port\_exam, aes(port\_exam[2,3]\*sqrt(12)\*100,

port\_exam[2,2]\*1200),

color = "purple", size = 7) +

geom\_point(data = port\_exam, aes(port\_exam[3,3]\*sqrt(12)\*100,

port\_exam[3,2]\*1200),

color = "black", size = 5) +

scale\_x\_continuous(limits = c(0,14)) +

labs(x = "Risk (%)",

y = "Return (%)",

title = "Simulated portfolios")

# Dominated portfolios

naive\_dom <- port %>%

filter(risk < port\_exam[3,3]+0.0005,

risk > port\_exam[3,3]-0.0005) %>%

summarise(round(max(returns) - port\_exam[3,2],4)\*1200+.02) %>%

as.numeric()

equal\_dom <- port %>%

filter(risk < port\_exam[1,3]+0.0005,

risk > port\_exam[1,3]-0.0005) %>%

summarise(round(max(returns) - port\_exam[1,2],3)\*1200) %>%

as.numeric()

risky\_dom <- port %>%

filter(risk < port\_exam[2,3]+0.0005,

risk > port\_exam[2,3]-0.0005) %>%

summarise(round(max(returns) - port\_exam[2,2],4)\*1200+.02) %>%

as.numeric()

# Finad max and equivalent risk for Equal risk slice

equal\_max <- port %>%

filter(risk < port\_exam[1,3]+0.0005,

risk > port\_exam[1,3]-0.0005) %>%

mutate(returns = returns\*1200,

risk = risk \* sqrt(12)\*100) %>%

arrange(desc(returns)) %>%

slice(1)

# Find wieghts for dominant portfolio

eq\_wt <- port %>%

mutate(spy\_wt = wts[,1],

shy\_wt = wts[,2],

gld\_wt = wts[,3],

returns = returns \* 1200,

risk = risk \* sqrt(12) \*100) %>%

filter(returns == equal\_max$returns,

risk == equal\_max$risk) %>%

select(spy\_wt, shy\_wt, gld\_wt)

# Graph weights

eq\_wt %>%

rename("SPY" = spy\_wt,

"SHY" = shy\_wt,

"GLD" = gld\_wt) %>%

gather(key,value) %>%

ggplot(aes(factor(key, level = c("SPY", "SHY", "GLD")), value\*100)) +

geom\_bar(stat = 'identity', fill = "blue") +

geom\_text(aes(label = round(value,2)\*100), nudge\_y = 5) +

labs(x = "Assets",

y = "Weights (%)",

title = "Derived weighting to improve returns")

1. Ultimately, our goal will be to yield a framework for building [**satisficing**](https://en.wikipedia.org/wiki/Satisficing), not opitmal portfolios. There’s no guarantee we’ll succeed![**↩**](https://osm.netlify.app/post/portfolio-starter-kit/#fnref1)
2. If there were, then we’d be able to conduct a complete test of the Capital Asset Pricing Model![**↩**](https://osm.netlify.app/post/portfolio-starter-kit/#fnref2)
3. Many simulations that we’ve seen generate a vector of random uniform numbers between 0 and 1 and then force them to sum to one by dividing each element by the sum of the vector. But this does two things: forces the randomness to cluster closer to 1/n weights, removing extreme values; and prevents extreme values from occurring randomly across the choice of assets. Our random weight algorithm attempts to adjust for that, but probably isn’t perfect. Let us know if you think it can be improved![**↩**](https://osm.netlify.app/post/portfolio-starter-kit/#fnref3)

**SHARPEn your portfolio**

 February 07, 2020

[R](https://osm.netlify.app/categories/r), [Stocks](https://osm.netlify.app/categories/stocks)

In our last [**post**](https://osm.netlify.com/post/portfolio-starter-kit/), we started building the intuition around constructing a reasonable portfolio to achieve an acceptable return. The hero of our story had built up a small nest egg and then decided to invest it equally across the three major asset classes: stocks, bonds, and real assets. For that we used three liquid ETFs (SPY, SHY, and GLD) as proxies. But our protagonist was faced with some alternative scenarios offered by his cousin and his co-worker; a Risky portfolio of almost all stocks and a Naive portfolio of 50/50 stocks and bonds.

Chart, histogram

Description automatically generated

After seeing the outcomes of the different portfolios, our hero wondered if there were a better alternative. To accommodate, we simulated the range of outcomes one could potentially expect based on the risk, return, and correlation profiles of the three ETFs. We did this by creating a 1000 randomly weighted portfolios. When we graphed the results of the simulation, our hero could see how his portfolio (red dot) compared with the risky (purple dot), naive (black dot), and many other portfolios, as shown below. Additionally, the scatter plot showed our hero that for a given level of risk, he could find a portfolio that offered the best possible return, or, for a given level of return, he could decide how much risk he wanted to take. The portfolio with the highest return for a given level of risk “dominated” the other portfolios at that level of risk.

Chart, scatter chart

Description automatically generated

However, we showed that some of these dominant portfolios might not be intuitively acceptable even if mathematically optimal. For example, when our hero thought that he was fine with the current riskiness of his portfolio, but wanted to eke out a bit more return, the solution was to increase his exposure to stocks by over 20 points and increase his exposure to gold by four points, all at the expense of bonds. But this only resulted in a one-to-two point improvement in returns. If he wanted to improve returns more than that, he would have to alter how much risk he would be willing to accept.

Chart, bar chart

Description automatically generated

That begged the question of whether there was an alternative solution. Let’s resume where we left off…

Now that we’ve seen that a major change in the portfolio weights doesn’t yield that much improvement in returns, should we find a different metric? Maybe we should be looking for the best risk-adjusted return. Let’s graph the same random portfolios, but color them according to their risk-adjusted returns—in this case, simply return over risk—and we’ll call this the Sharpe ratio after the Nobel Prize winner William F. Sharpe who developed the concept.[**1**](https://osm.netlify.app/post/sharpen-your-portfolio/#fn1) The higher the Sharpe ratio the greener the point, the lower the redder.

Chart, scatter chart

Description automatically generated

Interestingly, the highest risk-adjusted returns appear to be at the low end of the graph. In fact, the highest risk-adjusted return also happens to be the portfolio with the lowest risk. That isn’t exactly counter-intuitive. But it raises the question of how much additional return you’re getting for taking on more risk. To see this we add a line whose slope matches a one-to-one correspondence between change in risk and change in return. This is shown in the graph below.

Chart, scatter chart

Description automatically generated

What’s interesting about this line is that it tells you which portfolios generate more than one unit of return per unit of risk and which ones generate less. Let’s spend a few moments on the graph.

The points that lie above the purple line represent portfolios where you’re return per unit of risk is greater than one for one. The portfolios below that are, obviously, the opposite. It’s important to remember that in this case, a unit of risk is not the same as a unit of return. Volatility (or the standard deviation of returns) is used as a proxy for risk. Hence, risk is a standardized range, while return is a point. So it’s not the same as risk a dollar to make a dollar. You’re risking a likely range of dollars to make a dollar. If one of the portfolios has a 5% average return and a 10% risk, that means the returns of the portfolio could be -5% to 15% close to 70% of the time. Hence, when risk increases by one unit, the range of possible outcomes widens by two units. In the previous example, the range of values (-5% to 15%) based on risk was about 20 percentage points. If that risk increased by one unit to 11%, then the range would be 22 percentage points (-6% to 16%).

For those not indoctrinated by portfolio theory this isn’t the most intuitive concept on first blush. But think about it this way: embedded in that range of potential values is a risk of loss. By bearing that potential loss, you’re expecting a potential gain. So the purple line cuts the portfolios between those for which the expected upside potential is greater than overall potential. In other words, the upside is greater than a reasonable expectation of the downside and vice versa. Most folks prefer the upside to be greater than the down. In future posts, we’ll re-arrange this to look at risk only as expected loss. But we need to walk before we can run.

Let’s go back and see where the three portfolios are in relation to the purple line.

Chart, scatter chart

Description automatically generated

None of the portfolios enjoy a one-to-one relationship between return and risk. That doesn’t mean they’re “bad” portfolios. If you’re required return is greater than 5%—roughly the point above which return starts to lag risk—then to achieve that you’ll need to accept a poorer risk-adjusted return profile. That begs the question of whether this trade-off of accepting incrementally more downside potential for incrementally less upside potential is worth it.

Answering the “worth it” question takes us out of the realm of numbers and into the realm of preferences, psychology, and behavior. We won’t dwell too long on this because it’s hard to generalize individual preferences. Behavioral finance attempts to identify and explain the motivation and effect of such preferences. But that is way beyond the scope of this post.

Let’s move on to look at what the average weights for those greater than one-for-one return-to-risk portfolios actually look like.

Chart, bar chart, histogram

Description automatically generated

On average we see a very high allocation to bonds and not much to stocks or gold. For people that don’t require a high return, this would probably be a good portfolio mix. But let’s assume our protagonist needs more than that, yet he doesn’t want to stray too far from a relatively evenly balanced weighting. So we’ll keep close to the same volatility and see what types of returns we can generate along with the implied portfolio weights. Here’s the original table of returns and risk.

| Table 1: Annualized performance metrics | | | |
| --- | --- | --- | --- |
| **Asset** | **Return (%)** | **Risk (%)** | **Sharpe ratio** |
| Equal | 6.0 | 9.2 | 0.66 |
| Naive | 6.0 | 6.9 | 0.81 |
| Risky | 8.4 | 12.4 | 0.64 |

Let’s look at the portfolios between the two bands that represent one percentage point more or less risk than the equal-weighted portfolio in the graph below.

Chart, scatter chart

Description automatically generated

Now we’ll see what the average returns and risk are for those risk bands. One thing should stand out: while both average returns and risk are higher, so is the Sharpe ratio. In general, then, our hero can achieve better returns and risk-adjusted returns by widening his risk parameters.

| Table 2: Average returns and risk for risk bands (%) | | |
| --- | --- | --- |
| **Returns** | **Risk** | **Sharpe** |
| 7.4 | 9.7 | 0.76 |

That doesn’t seem too bad. Let’s graph the average weights.

Chart, bar chart, histogram

Description automatically generated

Seems reasonable. But you’ll note that this change in allocation isn’t too different from switching to the dominant portfolio we calculated earlier. So our hero thinks that maybe the gold allocation is too high. He wants to see if there are any portfolios that would afford him a similar risk and return, but allocate no more than 20% to gold. Indeed, there is as shown below.

| Table 3: Average returns and risk for risk bands with gold constraint (%) | | |
| --- | --- | --- |
| **Returns** | **Risk** | **Sharpe** |
| 7.3 | 9.7 | 0.75 |

However, to get gold below 20%, we need to raise the allocation to stocks to over 60%. Our hero’s not sure if this is the type of allocation he wants, so he asks if it’s possible to lower the exposure to stocks a little. Unfortunately, no luck there. So what does the average weighting look like?

Chart, bar chart, histogram

Description automatically generated

Almost two-thirds of the portfolio is allocated to stocks and the remainder is relatively evenly divided among bonds and commodities. Is this acceptable? The weighting to stocks more than doubles, the weighting to gold is almost chopped in half, and the Sharpe ratio improves by over 10%. Our hero may not like the higher weighting to stocks, but at least his risk-adjusted return is much better. Only our hero can tell if he’s comfortable with the new portfolio. Whatever the case, we’re far off from an “optimal” portfolio. Where does the “average” portfolio lie on the scatter plot? The yellow dot is that portfolio, we’ll call it the “sufficient portfolio.”

Chart, scatter chart

Description automatically generated

What does this tell us? While the sufficient portfolio doesn’t offer the highest return for the given level of risk, it does offer a higher return for only a moderate increase of risk and with an allocation our hero may prefer relative to his cousin’s, co-worker’s, or the remainder of options. But then again it might not, in which case, we’d have to re-run the calculations with different weight constraints. Let’s at least look at how the sufficient would have performed historically, as shown in the graph below with the wider line in purple, before we summarize.

Chart, histogram

Description automatically generated

What’s the key takeaway? Portfolios that offer the highest return for a given level of risk may not be an allocation that many investors would be comfortable with. And the highest risk-adjusted return portfolio may not offer the required return. But it is possible to find a portfolio that offers most of the necessary requirements and improves risk-adjusted returns if the range of acceptable outcomes is broadened and the constraints aren’t overly stringent. Finding these portfolios becomes more of an iterative process than a closed form solution. Would the new portfolio be satisfactory? At the very least, that depends on the cost of adherence. If the psychological cost to maintain the portfolio is low—that is, it doesn’t keep you up at night—then, provided the portfolio satisfies the other requirements, it is a more “satisfactory” portfolio. This obviously touches on behavioral elements that would require separate posts, but the goal is to view most of these concepts through the lens of what has an intuitive appeal. Eventually, we might find a portfolio that [**satisfices**](https://en.wikipedia.org/wiki/Satisficing) our hero’s risk and return requirements. That is, it satisfies his needs based on sufficient thresholds. We won’t delve into that concept more in this post, but it will underlie the rest of this series on portfolio construction.

And speaking of this series, there’s still more ground to cover. Over the next few posts we’ll examine benchmarking, rebalancing, semi-deviation, capital market expectations, and time dependence. Stick with us and let us know if there’s something you’d like to see. Until then, here’s the code:

# Load package

library(tidyquant)

# Get data

symbols <- c("SPY", "EEM", "SHY", "IYR", "GLD")

symbols\_low <- tolower(symbols)

prices <- getSymbols(symbols, src = "yahoo",

from = "1990-01-01",

auto.assign = TRUE) %>%

map(~Ad(get(.))) %>%

reduce(merge) %>%

`colnames<-`(symbols\_low)

prices\_monthly <- to.monthly(prices, indexAt = "last", OHLC = FALSE)

ret <- ROC(prices\_monthly)["2005/2019"]

naive <- ret[,c("spy", "shy")]

basic <- ret[,c("spy", "shy", "gld")]

# Create different weights and portflios

wt1 <- rep(1/(ncol(basic)), ncol(basic))

port1 <- Return.portfolio(basic, wt1) %>%

`colnames<-`("ret")

wt2 <- c(0.9, 0.10, 0)

port2 <- Return.portfolio(basic, weights = wt2) %>%

`colnames<-`("ret")

wtn <- c(0.5, 0.5)

portn <- Return.portfolio(naive, wtn)

port\_comp <- data.frame(date = index(port1), equal = as.numeric(port1),

wtd = as.numeric(port2),

naive = as.numeric(portn))

port\_comp %>%

gather(key,value, -date) %>%

group\_by(key) %>%

mutate(value = cumprod(value+1)) %>%

ggplot(aes(date, value\*100, color = key)) +

geom\_line() +

scale\_color\_manual("", labels = c("Equal", "Naive", "Risky"),

values = c("blue", "black", "red")) +

labs(x = "",

y = "Index",

title = "Three portfolios, which is best?",

caption = "Source: Yahoo, OSM estimates") +

theme(legend.position = "top",

plot.caption = element\_text(hjust = 0))

# Portfolio

mean\_ret <- apply(ret[,c("spy", "shy", "gld")],2,mean)

cov\_port <- cov(ret[,c("spy", "shy", "gld")])

port\_exam <- data.frame(ports = colnames(port\_comp)[-1],

ret = as.numeric(apply(port\_comp[,-1],2, mean)),

vol = as.numeric(apply(port\_comp[,-1], 2, sd)))

# Weighting that ensures more variation and random weighthing to stocks

set.seed(123)

wts <- matrix(nrow = 1000, ncol = 3)

for(i in 1:1000){

a <- runif(1,0,1)

b <- c()

for(j in 1:2){

b[j] <- runif(1,0,1-sum(a,b))

}

if(sum(a,b) < 1){

inc <- (1-sum(a,b))/3

vec <- c(a+inc, b+inc)

}else{

vec <- c(a,b)

}

wts[i,] <- sample(vec,replace = FALSE)

}

# Calculate random portfolios

port <- matrix(nrow = 1000, ncol = 2)

for(i in 1:1000){

port[i,1] <- as.numeric(sum(wts[i,] \* mean\_ret))

port[i,2] <- as.numeric(sqrt(t(wts[i,] %\*% cov\_port %\*% wts[i,])))

}

colnames(port) <- c("returns", "risk")

port <- as.data.frame(port)

# Graph with points

port %>%

ggplot(aes(risk\*sqrt(12)\*100, returns\*1200)) +

geom\_point(color = "blue", size = 1.2, alpha = 0.4) +

geom\_smooth(method = "loess", formula = y ~ log(x), se = FALSE, color = "slategrey") +

geom\_point(data = port\_exam, aes(port\_exam[1,3]\*sqrt(12)\*100,

port\_exam[1,2]\*1200),

color = "red", size = 6) +

geom\_point(data = port\_exam, aes(port\_exam[2,3]\*sqrt(12)\*100,

port\_exam[2,2]\*1200),

color = "purple", size = 7) +

geom\_point(data = port\_exam, aes(port\_exam[3,3]\*sqrt(12)\*100,

port\_exam[3,2]\*1200),

color = "black", size = 5) +

scale\_x\_continuous(limits = c(0,14)) +

labs(x = "Risk (%)",

y = "Return (%)",

title = "Simulated portfolios")

# Finad max and equivalent risk for Equal risk slice

equal\_max <- port %>%

filter(risk < port\_exam[1,3]+0.0005,

risk > port\_exam[1,3]-0.0005) %>%

mutate(returns = returns\*1200,

risk = risk \* sqrt(12)\*100) %>%

arrange(desc(returns)) %>%

slice(1)

# Find wieghts for dominant portfolio

eq\_wt <- port %>%

mutate(spy\_wt = wts[,1],

shy\_wt = wts[,2],

gld\_wt = wts[,3],

returns = returns \* 1200,

risk = risk \* sqrt(12) \*100) %>%

filter(returns == equal\_max$returns,

risk == equal\_max$risk) %>%

select(spy\_wt, shy\_wt, gld\_wt)

# Graph weights

eq\_wt %>%

rename("SPY" = spy\_wt,

"SHY" = shy\_wt,

"GLD" = gld\_wt) %>%

gather(key,value) %>%

ggplot(aes(factor(key, level = c("SPY", "SHY", "GLD")), value\*100)) +

geom\_bar(stat = 'identity', fill = "blue") +

geom\_text(aes(label = round(value,2)\*100), nudge\_y = 5) +

labs(x = "Assets",

y = "Weights (%)",

title = "Derived weighting to improve returns")

# Portfolio with Sharpe ratio

port %>%

ggplot(aes(risk\*sqrt(12)\*100, returns\*1200, color = sharpe)) +

geom\_point(size = 1.2, alpha = 0.4) +

geom\_point(data = port\_exam, aes(port\_exam[1,3]\*sqrt(12)\*100,

port\_exam[1,2]\*1200),

color = "red", size = 6) +

geom\_point(data = port\_exam, aes(port\_exam[2,3]\*sqrt(12)\*100,

port\_exam[2,2]\*1200),

color = "purple", size = 7) +

geom\_point(data = port\_exam, aes(port\_exam[3,3]\*sqrt(12)\*100,

port\_exam[3,2]\*1200),

color = "black", size = 5) +

scale\_x\_continuous(limits = c(0,14)) +

labs(x = "Risk (%)",

y = "Return (%)",

title = "Simulated portfolios",

color = "Sharpe ratio") +

scale\_color\_gradient(low = "red", high = "green") +

theme(legend.position = "top", legend.key.size = unit(.5, "cm"))

# Portfolio with sharpe line

max\_sharpe <- max(port$sharpe)\*sqrt(12)

port %>%

ggplot(aes(risk\*sqrt(12)\*100, returns\*1200, color = sharpe)) +

geom\_point(size = 1.2, alpha = 0.4) +

geom\_abline(intercept = 0, slope = max\_sharpe, color = "blue") +

labs(x = "Risk (%)",

y = "Return (%)",

title = "Simulated portfolios",

color = "Sharpe ratio") +

scale\_color\_gradient(low = "red", high = "green") +

theme(legend.position = "top", legend.key.size = unit(.5, "cm"))

# Graph with one-to-one

port %>%

ggplot(aes(risk\*sqrt(12)\*100, returns\*1200, color = sharpe)) +

geom\_point(size = 1.2, alpha = 0.4) +

geom\_abline(intercept = 0, slope = max\_sharpe, color = "blue") +

geom\_abline(color = "purple", lwd = 1.25)+

labs(x = "Risk (%)",

y = "Return (%)",

title = "Simulated portfolios",

color = "Sharpe ratio") +

scale\_color\_gradient(low = "red", high = "green") +

geom\_text(aes(x = 5, y = 7),

label = "Purple line is \none-to-one \nreturn-to-risk.",

color = "purple")

# Three portfolios with purple line

port %>%

ggplot(aes(risk\*sqrt(12)\*100, returns\*1200, color = sharpe)) +

geom\_point(size = 1.2, alpha = 0.4) +

geom\_point(data = port\_exam, aes(port\_exam[1,3]\*sqrt(12)\*100,

port\_exam[1,2]\*1200),

color = "red", size = 6) +

geom\_point(data = port\_exam, aes(port\_exam[2,3]\*sqrt(12)\*100,

port\_exam[2,2]\*1200),

color = "purple", size = 7) +

geom\_point(data = port\_exam, aes(port\_exam[3,3]\*sqrt(12)\*100,

port\_exam[3,2]\*1200),

color = "black", size = 5) +

geom\_abline(color = "purple", size = 1.1) +

scale\_x\_continuous(limits = c(0,14)) +

labs(x = "Risk (%)",

y = "Return (%)",

title = "Simulated portfolios",

color = "Sharpe ratio") +

scale\_color\_gradient(low = "red", high = "green")

# High return to risk

port %>%

mutate(SPY = wts[,1],

SHY = wts[,2],

GLD = wts[,3],

returns = returns \* 1200,

risk = risk \* sqrt(12) \*100,

sharpe = sharpe\*sqrt(12)) %>%

filter(sharpe >= 1) %>%

summarise\_all(mean) %>%

gather(key, value) %>%

filter(!key %in% c("returns", "risk", "sharpe")) %>%

ggplot(aes(factor(key, labels = c("SPY", "SHY", "GLD")), value \*100)) +

geom\_bar(stat = "identity", fill = "blue") +

labs(x = "",

y = "Weight (%)",

title = "Average weights for high risk-adjusted return portfolios") +

geom\_text(aes(label = round(value,2)\*100), nudge\_y = 4)

# Table

port\_comp %>%

rename("Equal" = equal,

"Naive" = naive,

"Risky" = wtd) %>%

gather(Asset, value, -date) %>%

group\_by(Asset) %>%

summarise(`Mean (%)` = round(mean(value, na.rm = TRUE),3)\*1200,

`Volatility (%)` = round(sd(value, na.rm = TRUE)\*sqrt(12),3)\*100,

`Risk-adjusted (%)` = round(mean(value, na.rm = TRUE)/sd(value, na.rm=TRUE)\*sqrt(12),3)\*100,

`Cumulative (%)` = round(prod(1+value, na.rm = TRUE),3)\*100) %>%

knitr::kable(caption = "Annualized performance metrics")

# Graph with risk bands

port %>%

ggplot(aes(risk\*sqrt(12)\*100, returns\*1200)) +

geom\_point(color = "blue", size = 1.2, alpha = 0.4) +

geom\_point(data = port\_exam, aes(port\_exam[1,3]\*sqrt(12)\*100,

port\_exam[1,2]\*1200),

color = "red", size = 6) +

geom\_vline(xintercept = up\_band, color = "slateblue") +

geom\_vline(xintercept = down\_band, color = "slateblue") +

labs(x = "Risk (%)",

y = "Return (%)",

title = "Simulated portfolios")

# Portfoilio band output for blog

port %>%

mutate(spy\_wt = wts[,1],

shy\_wt = wts[,2],

gld\_wt = wts[,3],

returns = returns \* 1200,

risk = risk \* sqrt(12) \*100,

sharpe = returns/risk) %>%

filter(returns > port\_exam[1,2]\*1200 +1,

risk >= down\_band,

risk < up\_band) %>%

summarise\_all(function(x) round(mean(x),1)) %>%

select(returns, risk, sharpe) %>%

rename("Returns" = returns,

"Risk" = risk,

"Sharpe" = sharpe) %>%

knitr::kable(caption = "Average returns and risk for risk bands (%)")

port %>%

mutate(SPY = wts[,1],

SHY = wts[,2],

GLD = wts[,3],

returns = returns \* 1200,

risk = risk \* sqrt(12) \*100,

sharpe = returns/risk) %>%

filter(returns > port\_exam[1,2]\*1200 +1,

risk >= down\_band,

risk < up\_band) %>%

summarise\_all(mean) %>%

gather(key, value) %>%

filter(key %in% c("SPY", "SHY", "GLD")) %>%

ggplot(aes(factor(key, levels = c("SPY", "SHY", "GLD")) ,value\*100)) +

geom\_bar(stat = "identity", fill = "blue") +

labs(x = "",

y = "Weight (%)",

title = "Average weights for high risk-adjusted return portfolios") +

geom\_text(aes(label = round(value,2)\*100), nudge\_y = 5)

# Portfoilio band output for blog

port %>%

mutate(spy\_wt = wts[,1],

shy\_wt = wts[,2],

gld\_wt = wts[,3],

returns = returns \* 1200,

risk = risk \* sqrt(12) \*100,

sharpe = returns/risk) %>%

filter(returns > port\_exam[1,2]\*1200 +1,

risk >= down\_band,

risk < up\_band,

gld\_wt <= 0.2) %>%

summarise\_all(function(x) round(mean(x),2)) %>%

select(returns, risk, sharpe) %>%

rename("Returns" = returns,

"Risk" = risk,

"Sharpe" = sharpe) %>%

knitr::kable(caption = "Average returns and risk for risk bands (%)")

# Bar chart of weights

port %>%

mutate(SPY = wts[,1],

SHY = wts[,2],

GLD = wts[,3],

returns = returns \* 1200,

risk = risk \* sqrt(12) \*100,

sharpe = returns/risk) %>%

filter(returns > port\_exam[1,2]\*1200 +1,

risk >= down\_band,

risk < up\_band,

GLD <= 0.2) %>%

summarise\_all(mean) %>%

gather(key, value) %>%

filter(key %in% c("SPY", "SHY", "GLD")) %>%

ggplot(aes(factor(key, levels = c("SPY", "SHY", "GLD")) ,value\*100)) +

geom\_bar(stat = "identity", fill = "blue") +

labs(x = "",

y = "Weight (%)",

title = "Average weights for high risk-adjusted return portfolios") +

geom\_text(aes(label = round(value,2)\*100), nudge\_y = 5)

port %>%

ggplot(aes(risk\*sqrt(12)\*100, returns\*1200)) +

geom\_point(color = "blue", size = 1.2, alpha = 0.4) +

geom\_point(data = port\_exam, aes(port\_exam[1,3]\*sqrt(12)\*100,

port\_exam[1,2]\*1200),

color = "red", size = 6) +

geom\_point(data = port\_exam, aes(port\_exam[2,3]\*sqrt(12)\*100,

port\_exam[2,2]\*1200),

color = "purple", size = 7) +

geom\_point(data = port\_exam, aes(port\_exam[3,3]\*sqrt(12)\*100,

port\_exam[3,2]\*1200),

color = "black", size = 5) +

geom\_point(data = suff\_port, aes(risk,returns),

color = "yellow", size = 8) +

geom\_vline(xintercept = up\_band, color = "slateblue") +

geom\_vline(xintercept = down\_band, color = "slateblue") +

scale\_x\_continuous(limits = c(0,14)) +

labs(x = "Risk (%)",

y = "Return (%)",

title = "Simulated portfolios with sufficient allocation")

# Add portfolio

port\_suff <- Return.portfolio(basic,suff\_port\_wts) %>%

`colnames<-`("suff")

# Graph

port\_comp %>%

mutate(suff = as.numeric(port\_suff)) %>%

gather(key,value, -date) %>%

group\_by(key) %>%

mutate(value = cumprod(value+1)) %>%

ggplot(aes(date, value\*100, color = key)) +

geom\_line(aes(size = key)) +

scale\_color\_manual("", labels = c("Equal", "Naive", "Sufficient", "Risky"),

values = c("blue", "black", "purple","red")) +

scale\_size\_manual(values = c(1,1,2,1), guide = 'none') +

labs(x = "",

y = "Index",

title = "Adding the sufficient portfolio",

caption = "Source: Yahoo, OSM estimates") +

theme(legend.position = "top",

plot.caption = element\_text(hjust = 0))

1. The [**Sharpe ratio**](https://en.wikipedia.org/wiki/Sharpe_ratio) was developed by [**William Sharpe**](https://en.wikipedia.org/wiki/William_F._Sharpe) to measure the excess return of an asset over risk-free rates adjusted for volatility.

**Benchmarking the portfolio**

 February 14, 2020

[R](https://osm.netlify.app/categories/r), [Stocks](https://osm.netlify.app/categories/stocks)

In our last [**post**](https://osm.netlify.com/post/sharpen-your-portfolio/), we looked at one measure of risk-adjusted returns, the Sharpe ratio, to help our hero decide whether he wanted to alter his portfolio allocations. Then, as opposed to finding the maximum return for our hero’s initial level of risk, we broadened the risk parameters and searched for portfolios that would at least offer the same return or better as his current portfolio and would also allow him to find a “comfortable” asset allocation. While we couldn’t confirm that we ended up with a portfolio our hero found satisfactory (this is fictional after all!), we did show how we might come close. If the first iteration failed, further iterations were possible.

While this held out hope that we might find a portfolio that could potentially [**satisfice**](https://en.wikipedia.org/wiki/Satisficing) the risk/return question, there was a problem with this process: trying to choose appropriate levels of risk-adjusted returns lacked a basis for comparison. True, if one were working directly with our hero to develop a full-fledged financial plan, the first part would be to establish the investment goals and then from there establish acceptable risk/return parameters to achieve those goals. For example, perhaps our hero wants to retire at age “x” and believes he can lead a reasonably comfortable life by spending “y” per year. Then, based on his current age, and his life expectancy, it is relatively straightforward to calculate the required return to fulfill his goal. Figuring out his risk tolerance (i.e., the amount of volatility he’s willing to sustain) is more difficult and is a combination of art and science. However, assuming you can establish a range, which is more reasonable that a single risk level, it is possible to find a combination of assets that are likely to yield a reasonably good ex ante expectation of achieving our hero’s goals.

But what if it’s not possible to define such goals succinctly or what if our hero only cares about achieving “good” returns. In the first post, he was happy because his total return beat his co-worker’s but not ecstatic because he underperformed his cousin. Were either of these results reasonable yardsticks? Not really. What’s the alternative? Establish a benchmark against which to judge results.

What is an appropriate benchmark? At the most general, it would be one that includes all risk assets. But, as we noted in our first [**post**](https://osm.netlify.com/post/portfolio-starter-kit/), it’s next to impossible to find such a beast. And that is why we chose a very simple portfolio of three ETFs: SPY, SHY, and GLD. However, there are ETFs that encompass more assets than these three. Indeed, there are total stock and bond market ETFs. Total real asset ETFs are relatively rare, however, and the ones that do exist come with a hefty expense ratio or track an index even many professionals have never heard of. Of course, there are some commodity ETFs, but they too track indices that only a professional could analyze.

Even if we could find a good ETF for real assets, it is unclear how much we should allocate to them as a percent of the total. Good luck finding a rigorous answer too! While we don’t want to punt on this issue, we also don’t want to dive down a rabbit hole just yet. So we’ll shelve the real asset for now.

Most “recommended” allocation strategies assume some sort of stock/bond mix. So we’ll go with that, not because we want to appeal to the lowest common denominator, but because that data is more readily available. What should be a good benchmark then? To our mind, one that includes all stocks and all bonds. Four ETFs that encapsulate the total US and ex-US stock and bond markets are VTI, VXUS, BND, BNDX. We’ll start with those.

Next we should figure out the allocations. According to one [**source**](https://www.statista.com/statistics/710680/global-stock-markets-by-country/), the US stock market makes up about 53% of the global stock market; the US bond market is about 39% of global bonds, according to this [**site**](https://en.wikipedia.org/wiki/Bond_market). The global stock market relative to the global bond market? A quick Google search reveals it’s about 45/55. While this is a bit rough, let’s construct a benchmark that is 24% US and 21% ex-US stocks, and 22% and 33% US and ex-US bonds. This should be relatively close to a benchmark that tracks the returns to the majority of risk assets.

The only problem with this data is that we don’t have a full set until the beginning of 2014. Whatever the case, we’ll build the benchmark and graph it along with the other portfolios.

Chart, histogram

Description automatically generated

Our hero will be disappointed. His portfolio underperforms all others based on cumulative returns since 2014. Let’s look at the standard performance metrics including the benchmark.

| Table 1: Annualized performance metrics | | | | |
| --- | --- | --- | --- | --- |
| **Asset** | **Mean (%)** | **Volatility (%)** | **Sharpe** | **Cumulative (%)** |
| Bench | 6.0 | 5.1 | 1.10 | 139.3 |
| Equal | 6.0 | 6.7 | 0.82 | 137.1 |
| Naive | 7.2 | 7.1 | 1.03 | 152.9 |
| Risky | 10.8 | 10.7 | 0.99 | 181.8 |

We see that even though our hero’s portfolio enjoyed the same average return as the benchmark, its volatility was higher, and, presumably, suffered greater downside deviation than the benchmark based on the slightly worse cumulative return. While we won’t delve into it now, recall that over the 2005-2019 period our hero’s portfolio outperformed the naive portfolio. Given the time slice we’re looking at now, that suggests that the 2005-2014 period was when the major outperformance occurred. That highlights the impact of starting point, or, as we’ll call it “time dependence”, on portfolio returns. We’ll come back to this topic later.

Now we’ll run the simulations for a 1000 portfolios. First, we’ll show the simulations for the full period data to refresh the reader’s memory. The dots represent the equal-weighted (blue for our hero), naive (black), and risky (purple) portfolios. We color the points by Sharpe ratio with green being the higher and red being the lower.

Chart, scatter chart

Description automatically generated

Now let’s run the simulations again but based on data since 2014.

Chart, scatter chart

Description automatically generated

The resulting scatter plot has an unusual shape for those accustomed to looking these sorts of things. We won’t spend a lot of time on it, but the main reason for this is that the points at the bottom end of the plot have a majority of their weighting in gold, which generated only about 3.5% annualized returns vs. over 11% for the S&P in the period. Because we’ve only included three assets, and SHY’s returns were modest, there wasn’t a mid-level return that would fill in the empty space. In any event, we see our hero’s portfolio underperforms the naive and risky portfolios in the context of all the simulated portfolios. The highest Sharpe ratios remain at the low end of the plot.

Next we’ll graph the simulated portfolios in relation to the benchmark. Here we use an adjusted Sharpe ratio, where we calculate how much the portfolio exceeds the benchmark return divided by the portfolio risk (i.e., volatility of returns).

Chart, scatter chart

Description automatically generated

This is an interesting graph in contrast to the previous one. Here, many of the portfolios that showed up with a lower Sharpe ratio are the exact opposite relative to the benchmark. While it’s a pretty graph, all the color makes it tough to distinguish under vs. outperformance. We’ll segment ratios according to whether they’re positive (green for outperformance) or negative (red for underperformance).

Chart, scatter chart

Description automatically generated

Now we see that most of the portfolios failed to exceed the benchmark’s return. In fact, only 39% of the portfolios outperformed. Whether the ratio is scaled by volatility or not does not affect the sign.

Does this mean that those portfolios that underperform are “bad”? Not necessarily. It ultimately depends on the risk/return parameters. If you don’t want to expose yourself to the same level of risk as the benchmark, then you’ll have to accept lower returns. If we scale the return over the benchmark by the sensitivity to the benchmark the results don’t differ that much when viewed graphically.[**1**](https://osm.netlify.app/post/benchmarking-the-portfolio/#fn1) Hence, we’ll save some space and not graph that raio.

One might be wondering if there were a quick to way to discover the source of portfolio performance relative to the benchmark. One way is to calculate how much the portfolios exceed the benchmark relative to how much they deviate from it. This is called the information ratio, which is the average excess return over the benchmark divided by the standard deviation of that excess return, often referred to as tracking error. While it won’t tell you what were the sources of relative performance, it will tell you how much deviation from the benchmark played a role. We show a graph of the simulations colored by the information ratio.

Chart, scatter chart

Description automatically generated

This graph doesn’t appear too much different from the adjusted Sharpe graph. However, there does appear to be a greater number of bright green dots. What does that actually tell you? In the context of different portfolios, the brighter green represents a higher information ratio, which means it generates higher returns for its incremental deviation from the benchmark. While the information ratio has often been used to analyze investment managers’ performance, it can offer some insights into whether deviating from the benchmark is worth it. The graph shows our hero that the equal-weighted portfolio is not getting compensated for it’s deviation since it falls in the zero range area. Hence, a reasonable conclusion is that our hero would do better if he moved his allocation closer to the benchmark or simply switched out his current portfolio into the benchmark, given how broad-based it is. Of course, these conclusions are based on historical data. If we had benchmark data from 2005, we might very well make a different conclusion.

The problem with the information ratio is that it isn’t as informative as you might like! If a portfolio beats the index, but also strays a lot from it too, the information ratio might be worse than a portfolio that deviates very little. However, if the outperformance is sufficient, it will generally outweigh the tracking error. Alternatively a portfolio could have a higher ratio than a closet benchmark hugger, but have a lower cumulative return due to time dependence. Another problem is that the ratio focuses exclusively on benchmark risk. If the portfolio includes assets not in the benchmark, then you’d have to examine returns from those excluded risk exposures to get a more accurate picture. Even in our broad-based example, our hero’s information ratio didn’t look that great due to his gold exposure. But what if we had included commodities in the benchmark?

Let’s summarise? Looking at different portfolio allocations to achieve a desired risk/return parameter is all well good if you have well defined constraints. If you don’t, then it is helpful to incorporate a benchmark which is broad enough to encompass as many of the investable risk assets as possible. Such a yardstick helps frame and measure performance. Using an adjusted Sharpe ratio where you look at the excess return over the benchmark was useful, but was not able to distinguish performance strongly without segmenting the data rather crudely. The information ratio revealed important insights on whether a portfolio that contains mostly similar assets is getting compensated for deviating from benchmark allocations. In this case, our hero’s portfolio wasn’t, but that was probably due to the gold exposure.

In our upcoming posts, we’ll follow our hero as he investigates rebalancing, semi-deviation, capital market expectations, and time dependence. If you think we’re missing something, let us know. We’re also thinking of instituting some shorter, semi-regular posts that incorporate more of the the “machines” component of OSM. Let us know if there’s a model or signal you’d like us to investigage. Stay tuned. Until then, here’s the code that underpins all the previous analysis and graphics.

# Load package

library(tidyquant)

library(broom)

# Load data for portfolios

symbols <- c("SPY", "SHY", "GLD")

symbols\_low <- tolower(symbols)

prices <- getSymbols(symbols, src = "yahoo",

from = "1990-01-01",

auto.assign = TRUE) %>%

map(~Ad(get(.))) %>%

reduce(merge) %>%

`colnames<-`(symbols\_low)

prices\_monthly <- to.monthly(prices, indexAt = "last", OHLC = FALSE)

ret <- ROC(prices\_monthly)["2005/2019"]

# Load benchmark data

bench\_sym <- c("VTI", "VXUS", "BND", "BNDX")

bench <- getSymbols(bench\_sym, src = "yahoo",

from = "1990-01-01",

auto.assign = TRUE) %>%

map(~Ad(get(.))) %>%

reduce(merge) %>%

`colnames<-`(tolower(bench\_sym))

bench <- to.monthly(bench, indexAt = "last", OHLC = FALSE)

bench\_ret <- ROC(bench)["2014/2019"]

# Create different weights and portflios

# Equal weigthed

wt1 <- rep(1/(ncol(ret)), ncol(ret))

port1 <- Return.portfolio(ret, wt1) %>%

`colnames<-`("ret")

# Risk portfolio

wt2 <- c(0.9, 0.1, 0)

port2 <- Return.portfolio(ret, weights = wt2) %>%

`colnames<-`("ret")

# Naive portfolio

wtn <- c(0.5, 0.5, 0)

portn <- Return.portfolio(ret, wtn)

# Data frame of portfolios

port\_comp <- data.frame(date = index(port1), equal = as.numeric(port1),

risky = as.numeric(port2),

naive = as.numeric(portn))

# Benchmark portfolio

wtb <- c(0.24, 0.21, 0.22, 0.33)

portb <- Return.portfolio(bench\_ret, wtb, rebalance\_on = "quarters") %>%

`colnames<-`("bench")

# Graph of portfolios vs. benchmark

port\_comp %>%

filter(date >= "2014-01-01") %>%

mutate(bench = portb) %>%

gather(key,value, -date) %>%

group\_by(key) %>%

mutate(value = cumprod(value+1)) %>%

ggplot(aes(date, value\*100, color = key)) +

geom\_line() +

scale\_color\_manual("", labels = c("Bench", "Equal", "Naive", "Risky"),

values = c("purple", "blue", "black", "red")) +

labs(x = "",

y = "Index",

title = "The three portfolios with a benchmark",

caption = "Source: Yahoo, OSM estimates") +

theme(legend.position = "top",

plot.caption = element\_text(hjust = 0))

# summary

port\_comp %>%

filter(date >= "2014-01-01") %>%

mutate(bench = as.numeric(portb)) %>%

rename("Equal" = equal,

"Naive" = naive,

"Risky" = risky,

"Bench" = bench) %>%

gather(Asset, value, -date) %>%

group\_by(Asset) %>%

summarise(`Mean (%)` = round(mean(value, na.rm = TRUE),3)\*1200,

`Volatility (%)` = round(sd(value, na.rm = TRUE)\*sqrt(12),3)\*100,

`Sharpe` = round(mean(value, na.rm = TRUE)/sd(value, na.rm=TRUE)\*sqrt(12),2),

`Cumulative (%)` = round(prod(1+value, na.rm = TRUE),3)\*100) %>%

knitr::kable(caption = "Annualized performance metrics")

# Portfolio

mean\_ret <- apply(ret[,c("spy", "shy", "gld")],2,mean)

cov\_port <- cov(ret[,c("spy", "shy", "gld")])

port\_exam <- data.frame(ports = colnames(port\_comp)[-1],

ret = as.numeric(apply(port\_comp[,-1],2, mean)),

vol = as.numeric(apply(port\_comp[,-1], 2, sd)))

bench\_exam <- data.frame(ports = "bench",

ret = mean(bench\_ret),

vol = sd(bench\_ret))

bench\_spy <- data.frame(ports = "sp",

ret = mean(ret$spy),

vol = sd(ret$spy))

bench\_spy\_14 <- data.frame(ports = "sp",

ret = mean(ret$spy["2014/2019"]),

vol = sd(ret$spy["2014/2019"]))

mean\_ret\_14 <- apply(ret[,c("spy", "shy", "gld")]["2014/2019"],2,mean)

cov\_port\_14 <- cov(ret[,c("spy", "shy", "gld")]["2014/2019"])

port\_exam\_14 <- port\_comp %>%

filter(date >= "2014-01-01") %>%

select(-date) %>%

gather(ports, value) %>%

group\_by(ports) %>%

summarise\_all(list(ret = mean, vol = sd)) %>%

data.frame()

### Random weighting

# wts for full period

wts <- matrix(nrow = 1000, ncol = 3)

set.seed(123)

for(i in 1:1000){

a <- runif(1,0,1)

b <- c()

for(j in 1:2){

b[j] <- runif(1,0,1-sum(a,b))

}

if(sum(a,b) < 1){

inc <- (1-sum(a,b))/3

vec <- c(a+inc, b+inc)

}else{

vec <- c(a,b)

}

wts[i,] <- sample(vec,replace = FALSE)

}

# wts for 2014

wts1 <- matrix(nrow = 1000, ncol = 3)

set.seed(123)

for(i in 1:1000){

a <- runif(1,0,1)

b <- c()

for(j in 1:2){

if(j == 2){

b[j] <- 1 - sum(a,b)

}

else {

b[j] <- runif(1,0,1-sum(a,b))

}

vec <- c(a,b)

}

wts1[i,] <- sample(vec,replace = FALSE)

}

# Calculate random portfolios

# Weighting: wts

port <- matrix(nrow = 1000, ncol = 2)

for(i in 1:1000){

port[i,1] <- as.numeric(sum(wts[i,] \* mean\_ret))

port[i,2] <- as.numeric(sqrt(t(wts[i,] %\*% cov\_port %\*% wts[i,])))

}

colnames(port) <- c("returns", "risk")

port <- as.data.frame(port)

port <- port %>%

mutate(sharpe = returns/risk)

# Calculate random portfolios since 2014

# Weighting: wts1

port\_14 <- matrix(nrow = 1000, ncol = 2)

for(i in 1:1000){

port\_14[i,1] <- as.numeric(sum(wts1[i,] \* mean\_ret\_14))

port\_14[i,2] <- as.numeric(sqrt(t(wts1[i,] %\*% cov\_port\_14 %\*% wts1[i,])))

}

colnames(port\_14) <- c("returns", "risk")

port\_14 <- as.data.frame(port\_14)

port\_14 <- port\_14 %>%

mutate(sharpe = returns/risk)

# Grraph with Sharpe ratio

port %>%

ggplot(aes(risk\*sqrt(12)\*100, returns\*1200, color = sharpe)) +

geom\_point(size = 1.2, alpha = 0.4) +

geom\_point(data = port\_exam, aes(port\_exam[1,3]\*sqrt(12)\*100,

port\_exam[1,2]\*1200),

color = "red", size = 6) +

geom\_point(data = port\_exam, aes(port\_exam[2,3]\*sqrt(12)\*100,

port\_exam[2,2]\*1200),

color = "purple", size = 7) +

geom\_point(data = port\_exam, aes(port\_exam[3,3]\*sqrt(12)\*100,

port\_exam[3,2]\*1200),

color = "black", size = 5) +

scale\_x\_continuous(limits = c(0,14)) +

labs(x = "Risk (%)",

y = "Return (%)",

title = "Simulated portfolios",

color = "Sharpe ratio") +

scale\_color\_gradient(low = "red", high = "green") +

theme(legend.position = c(0.075,.8),

legend.key.size = unit(.5, "cm"),

legend.background = element\_rect(fill = NA))

# Graph since 2014

port\_14 %>%

ggplot(aes(risk\*sqrt(12)\*100, returns\*1200, color = sharpe)) +

geom\_point(size = 1.2, alpha = 0.4) +

geom\_point(data = port\_exam\_14, aes(port\_exam\_14[1,3]\*sqrt(12)\*100,

port\_exam\_14[1,2]\*1200),

color = "blue", size = 6) +

geom\_point(data = port\_exam\_14, aes(port\_exam\_14[3,3]\*sqrt(12)\*100,

port\_exam\_14[3, 2]\*1200),

color = "purple", size = 7) +

geom\_point(data = port\_exam\_14, aes(port\_exam\_14[2,3]\*sqrt(12)\*100,

port\_exam\_14[2,2]\*1200),

color = "black", size = 5) +

scale\_x\_continuous(limits = c(0,14)) +

labs(x = "Risk (%)",

y = "Return (%)",

title = "Simulated portfolios since 2014",

color = "Sharpe ratio") +

scale\_color\_gradient(low = "red", high = "green") +

theme(legend.position = c(0.075,0.8),

legend.background = element\_rect(fill = NA),

legend.key.size = unit(.5, "cm"))

# Portfolios benchmarked vs Vanguard

port\_14 %>%

mutate(Bench = returns - bench\_exam$ret) %>%

# mutate(Bench = ifelse(Bench > 0, 1, 0)) %>%

ggplot(aes(risk\*sqrt(12)\*100, returns\*1200, color = Bench)) +

geom\_point(size = 1.2, alpha = 0.4) +

scale\_color\_gradient(low = "red", high = "green") +

geom\_point(data = port\_exam\_14, aes(port\_exam\_14[1,3]\*sqrt(12)\*100,

port\_exam\_14[1,2]\*1200),

color = "blue", size = 6) +

geom\_point(data = port\_exam\_14, aes(port\_exam\_14[3,3]\*sqrt(12)\*100,

port\_exam\_14[3,2]\*1200),

color = "purple", size = 7) +

geom\_point(data = port\_exam\_14, aes(port\_exam\_14[2,3]\*sqrt(12)\*100,

port\_exam\_14[2,2]\*1200),

color = "black", size = 5) +

labs(x = "Risk (%)",

y = "Return (%)",

title = "Simulated portfolios since 2014") +

theme(legend.position = c(0.06,0.8),

legend.background = element\_rect(fill = NA),

legend.key.size = unit(.5, "cm"))

# Portfolios benchmarked vs Vanguard

port\_14 %>%

mutate(Bench = returns - bench\_exam$ret) %>%

mutate(Bench = ifelse(Bench > 0, 1, 0)) %>%

ggplot(aes(risk\*sqrt(12)\*100, returns\*1200, color = Bench)) +

geom\_point(size = 1.2, alpha = 0.4) +

scale\_color\_gradient(low = "red", high = "green") +

geom\_point(data = port\_exam\_14, aes(port\_exam\_14[1,3]\*sqrt(12)\*100,

port\_exam\_14[1,2]\*1200),

color = "blue", size = 6) +

geom\_point(data = port\_exam\_14, aes(port\_exam\_14[3,3]\*sqrt(12)\*100,

port\_exam\_14[3,2]\*1200),

color = "purple", size = 7) +

geom\_point(data = port\_exam\_14, aes(port\_exam\_14[2,3]\*sqrt(12)\*100,

port\_exam\_14[2,2]\*1200),

color = "black", size = 5) +

labs(x = "Risk (%)",

y = "Return (%)",

title = "Simulated portfolios") +

theme(legend.position = c(0.05,0.8),

legend.background = element\_rect(fill = NA),

legend.key.size = unit(.5, "cm"))

# Count how many portfolios are negative

pos\_b <- port\_14 %>%

mutate(Bench = returns - bench\_exam$ret) %>%

mutate(Bench = ifelse(Bench > 0, 1, 0)) %>%

summarise(bench = round(mean(Bench),2)\*100) %>%

as.numeric()

port\_list\_14 <- list()

for(i in 1:1000){

port\_list\_14[[i]] <- Return.portfolio(ret["2014/2019"], wts[i,]) %>%

data.frame() %>%

summarise(returns = mean(portfolio.returns),

excess\_ret = mean(portfolio.returns) - mean(portb$bench),

track\_err = sd(portfolio.returns - portb$bench),

risk = sd(portfolio.returns))

}

port\_info <- port\_list\_14 %>% bind\_rows

rfr <- mean(ret$shy)

# Graph info

port\_info %>%

mutate(info\_ratio = excess\_ret/track\_err) %>%

ggplot(aes(risk\*sqrt(12)\*100, returns\*1200, color = info\_ratio)) +

geom\_point(size = 1.2, alpha = 0.4) +

geom\_point(data = port\_exam\_14, aes(port\_exam\_14[1,3]\*sqrt(12)\*100,

port\_exam\_14[1,2]\*1200),

color = "blue", size = 6) +

geom\_point(data = port\_exam\_14, aes(port\_exam\_14[3,3]\*sqrt(12)\*100,

port\_exam\_14[3,2]\*1200),

color = "purple", size = 7) +

geom\_point(data = port\_exam\_14, aes(port\_exam\_14[2,3]\*sqrt(12)\*100,

port\_exam\_14[2,2]\*1200),

color = "black", size = 5) +

labs(x = "Risk (%)",

y = "Return (%)",

title = "Simulated portfolios") +

theme(legend.position = c(0.075,0.8),

legend.background = element\_rect(fill = NA),

legend.key.size = unit(.5, "cm")) +

scale\_color\_gradient("Information ratio", low = "red", high = "green")

1. This is an adjusted Treynor ratio in which the mean excess return of the portfolio over the benchmark is divided by the portfolio’s beta with the benchmark.[**↩**](https://osm.netlify.app/post/benchmarking-the-portfolio/#fnref1)

**Rebalancing! Really?**

 February 21, 2020

[R](https://osm.netlify.app/categories/r), [Stocks](https://osm.netlify.app/categories/stocks)

In our last [**post**](https://osm.netlify.com/post/benchmarking-the-portfolio/), we introduced benchmarking as a way to analyze our hero’s investment results apart from comparing it to alternate weightings or Sharpe ratios. In this case, the benchmark was meant to capture the returns available to a global aggregate of investable risk assets. If you could own almost every stock and bond globally and in the same proportion as their global contribution, what would your returns look like? We then used this benchmark as a way to judge our hero’s portfolio. We looked first at returns in excess of the benchmark scaled by volatility. Then we looked at excess returns scaled by deviation from the benchmark. We found that our hero’s portfolio was not being compensated for its deviation from the benchmark. The reason: gold was a drag on performance.

In this post, we move away from portfolio selection to rebalancing. Our goal will be to set up the intuition behind rebalancing, which will launch us into a more detailed discussion in further posts.

Why rebalance? Assume you establish some reasonably well defined risk and return parameters. Then you select the assets that offer the best approximation of achieving those parameters.[**1**](https://osm.netlify.app/post/rebalancing-really/#fn1) As those assets gain or lose value, the exposures change, leaving you with a different level of potential risk and return than originally envisioned.

A simple example. You invested at the bottom of the global financial crisis with a 50/50 weighting to SPY and SHY (the assets we’ve been using to proxy stocks and bonds). A year later, that weighting would be closer to 54/46. Moreover, you’re annualized risk has gone up by almost half a percentage point due to the higher exposure to stocks. While that might not seem like a lot, it may be more than you want. Indeed, with more assets and more variance in the returns, the weights and ending risk exposure could change significantly. So you rebalance to return to the original risk-return parameters.

Another reason to rebalance is if external factors have changed your risk or return parameters The most obvious example for an individual is the approach of, and then transition to, retirement. As one approaches retirement, risk tolerance generally declines since you don’t want to risk losing what you’ve worked so hard to gain. Hence, there’s a typical shift to assets with lower volatility. But other factors can change the parameters as well—changes in inflation, mortality, and health expectations are a few. For professional investors, reasons to rebalance include changes in funding mandates, sources, and scope. But we won’t go into those.

A third reason to rebalance is if the expectations for future risk and return profiles of the assets in the portfolio have changed. One sees this most ofthen in actively managed portfolios; that is, portfolios attempting to beat a benchmark. But even if you’re not trying to beat an index, if your expectations for asset returns and risk changes dramatically, then there’s a very good reason to rebalance. While this seems logical, getting it right is really tough. Unless you’ve got a crystal ball or there’s some obvious secular change, the risk of being wrong is great. In any event, we won’t say much more about that here, but will touch on this in depth when we get to the post on capital market expectations.

Apart from imminent retirement or some exogenous factor, the rationale behind rebalancing suggests that it should lead to better risk-adjusted returns. You sell winners to buy losers. And since “trees don’t grow to the sky” and nothing “stays down forever”[**2**](https://osm.netlify.app/post/rebalancing-really/#fn2) you’re dampening volatility by selling at the peak and buying near the trough. Or so the logic goes. If this is true, our hero should probably think about what type of rebalancing regime he might want to employ. Let’s see if rebalancing produces better results.

First off, we should note that we did not rebalance the various portfolios we looked at; that is, the equal-weighted (our hero’s), the naive, or the risky. We started them at their respective weights in 2005 and did not change. For the benchmark, however, we did rebalance every quarter. We did this to approximate the way many benchmarks are constructed. What would returns look like if we had rebalanced? We’ll look at results for different rebalancing periods: none, monthly, quarterly, and yearly. The table below shows our hero’s equal-weighted portfolio.

| Table 1: Equal-weighted portfolio performance for different rebalance periods (%) | | | | |
| --- | --- | --- | --- | --- |
| **Period** | **Return** | **Volatility** | **Sharpe** | **Total return** |
| None | 6.1 | 7.6 | 0.82 | 140.0 |
| Months | 6.1 | 9.2 | 0.67 | 134.5 |
| Quarters | 6.1 | 7.6 | 0.81 | 138.8 |
| Years | 6.2 | 7.6 | 0.83 | 142.8 |

Interestingly, we see very little difference in the average annual return. And the only difference in the remaining metrics is for the monthly rebalancing, which suffers higher volatility and thus a lower Sharpe ratio and total return. What’s causing the performance drag for monthly rebalancing? It could be that volatility whipsaws returns, which gets captured by the shorter reallocation time frame. Or the monthly rebalancing could simply be unlucky. Instead of buying losers that become winners, you just buy losers. We’d need to look at this in more detail, but we’ll save that for a later post. Next up the naive portfolio.

| Table 2: Naive portfolio performance for different rebalance periods (%) | | | | |
| --- | --- | --- | --- | --- |
| **Period** | **Return** | **Volatility** | **Sharpe** | **Total return** |
| None | 5.6 | 6.9 | 0.82 | 122.0 |
| Months | 5.3 | 6.8 | 0.79 | 112.6 |
| Quarters | 5.3 | 6.8 | 0.80 | 114.4 |
| Years | 5.5 | 6.6 | 0.84 | 119.6 |

At first glance, it’s unclear why monthly and quarterly rebalancing exhibit worse performance than the others. But it is probably not due to whipsawing, since SHY is mainly short-term US Treasuries. Hence, there’s not a lot of volatility. It is more likely due to performance drag since volatility is relatively the same across difference rebalancing periods. Whatever the case, let’s move on to the risky portfolio.

| Table 3: Risky portfolio performance for different rebalance periods (%) | | | | |
| --- | --- | --- | --- | --- |
| **Period** | **Return** | **Volatility** | **Sharpe** | **Total return** |
| None | 7.9 | 12.4 | 0.65 | 191.6 |
| Months | 7.9 | 12.5 | 0.64 | 188.7 |
| Quarters | 7.9 | 12.5 | 0.64 | 189.6 |
| Years | 8.0 | 12.4 | 0.65 | 192.5 |

Not surprisingly, a portfolio that has a 90% weighting to one asset, won’t see a lot of difference in performance due to different rebalancing periods. That’s because, unless the asset soared or crashed, the weighting is likely to remain relatively stable, so rebalancing would have a minimal effect. Finally, let’s look at different rebalancing periods for the benchmark portfolio

| Table 4: Benchmark portfolio performance for different rebalance periods (%) | | | | |
| --- | --- | --- | --- | --- |
| **Period** | **Return** | **Volatility** | **Sharpe** | **Total return** |
| None | 5.7 | 5.1 | 1.11 | 39.3 |
| Months | 5.7 | 5.2 | 1.11 | 39.4 |
| Quarters | 5.7 | 5.1 | 1.11 | 39.3 |
| Years | 5.7 | 5.2 | 1.10 | 39.4 |

Hmm. Nothing dramatic here either. So what gives? We presented all these rebalancing strategies and the surprise was that there wasn’t much of a surprise. Despite what appears to be sound logic, rebalancing did not produce better risk-adjusted returns in most cases. This confirms what some pundits argue: that rebalancing is useless. On the other hand, there are some who claim that it generates not just better risk-adjusted returns, but may even offer consistent outperformance.

We can’t run a full test on those competing claims here, we’ll do that in the next post along with providing some links to the different views, For now, let’s at least get a flavor of the kind of test we might use.

We simulate a portfolio of 10 years of monthly returns whose assets match the real returns and risk of US and global stocks and bonds and the S&P GSCI Commodity Total Return Index. We apply the same rebalancing periods as before. We present the table of performance metrics below.

| Table 5: Simulated portfolio performance for different rebalance periods (%) | | | | |
| --- | --- | --- | --- | --- |
| **Period** | **Return** | **Volatility** | **Sharpe** | **Total return** |
| None | 5.5 | 7.6 | 0.73 | 68.1 |
| Months | 6.2 | 7.2 | 0.87 | 81.3 |
| Quarters | 6.1 | 7.2 | 0.86 | 79.3 |
| Years | 5.8 | 7.2 | 0.81 | 74.1 |

We see that monthly and quarterly rebalancing produce modestly better risk-adjusted and total returns. Are these results significant? When we run t-tests on the returns, we find little to suggest the differences are anything more than noise. This is based on the p-values we show in the table below. For those who never took statistics or who find the mention of stats causes an immediate gag reflex, just note that the p-values are no where near 5%, which means the differences are likely due to randomness. We won’t test the significance of the Sharpe ratio for now simply because it involves some more sophisticated techniques.

| Table 6: Simulated portfolio p-values from t-test for differences in returns | |
| --- | --- |
| **Periods** | **P-values** |
| None vs. Months | 0.83 |
| None vs. Quarters | 0.85 |
| None vs. Years | 0.92 |
| Months vs. Quarters | 0.97 |
| Months vs. Years | 0.90 |
| Quarters vs. Years | 0.93 |

What have we learned thus far? The logic behing rebalancing appears straightforward: when circumstances (internal or external) change, then the portfolio should be rebalanced to weights that match the prior or new risk-return parameters. The main reason for this was to maintain or improve risk-adjusted returns. Yet, when we ran different rebalancing scenarios, we found little evidence to suggest returns were any different. And one simulation using historical returns of major assets, also showed little evidence of significant differences in performance.

Nonetheless, we’re not yet ready to say rebalancing is hogwash. There’s still more to do including running thousands of simulations to test for significant differences between rebalancing periods; examining whether rebalancing due to life changes does what it says on the tin; and testing rebalancing on a larger set of assets. But we’ll save those for the next couple posts. Until then, here’s the code:

# Load package

library(tidyquant)

library(tidyverse)

# Get data

symbols <- c("SPY", "SHY", "GLD")

symbols\_low <- tolower(symbols)

prices <- getSymbols(symbols, src = "yahoo",

from = "1990-01-01",

auto.assign = TRUE) %>%

map(~Ad(get(.))) %>%

reduce(merge) %>%

`colnames<-`(symbols\_low)

prices\_monthly <- to.monthly(prices, indexAt = "last", OHLC = FALSE)

ret <- ROC(prices\_monthly)["2005/2019"]

bench\_sym <- c("VTI", "VXUS", "BND", "BNDX")

bench <- getSymbols(bench\_sym, src = "yahoo",

from = "1990-01-01",

auto.assign = TRUE) %>%

map(~Ad(get(.))) %>%

reduce(merge) %>%

`colnames<-`(tolower(bench\_sym))

bench <- to.monthly(bench, indexAt = "last", OHLC = FALSE)

bench\_ret <- ROC(bench)["2014/2019"]

# Create different weights and portfolios

wt1 <- rep(1/(ncol(ret)), ncol(ret))

port1 <- Return.portfolio(ret, wt1) %>%

`colnames<-`("ret")

wt2 <- c(0.9, 0.1, 0)

port2 <- Return.portfolio(ret, weights = wt2) %>%

`colnames<-`("ret")

wtn <- c(0.5, 0.5, 0)

portn <- Return.portfolio(ret, wtn)

port\_comp <- data.frame(date = index(port1), equal = as.numeric(port1),

risky = as.numeric(port2),

naive = as.numeric(portn))

## Rebalancing for equal

# Create list

rebal = c("months", "quarters", "years")

equal\_list <- list()

for(pd in rebal){

equal\_list[[pd]] <- Return.portfolio(ret, wt1, rebalance\_on = pd) %>%

`colnames<-`(pd)

}

# Create data frame

equal <- equal\_list %>% bind\_cols() %>%

data.frame() %>%

mutate\_all(as.numeric) %>%

mutate(date = index(easy\_list[["months"]]),

none = as.numeric(port1)) %>%

select(date, none, everything())

equal %>%

rename("Months" = months,

"Quarters" = quarters,

"Years" = years,

"None" = none) %>%

gather(Period,value, -date) %>%

mutate(Period = factor(Period, labels = c("None", "Months", "Quarters", "Years"))) %>%

group\_by(Period) %>%

summarise(Return = round(mean(value)\*12,3)\*100,

Volatility = round(sd(value)\*sqrt(12),3)\*100,

Sharpe = round(mean(value)/sd(value)\*sqrt(12),2)+.01,

`Total return` = round(prod(1+value)-1,3)\*100) %>%

knitr::kable(caption = "Equal-weighted portfolio performance (%) for different rebalance periods")

# Rebalance for naive

naive\_list <- list()

for(pd in rebal){

naive\_list[[pd]] <- Return.portfolio(ret, wtn, rebalance\_on = pd) %>%

`colnames<-`(pd)

}

naive <- naive\_list %>%

bind\_cols() %>%

data.frame() %>%

mutate\_all(as.numeric) %>%

mutate(date = index(naive\_list[["months"]]),

none = as.numeric(portn)) %>%

select(date, none, everything())

naive %>%

rename("None" = none,

"Months" = months,

"Quarters" = quarters,

"Years" = years) %>%

gather(Period,value, -date) %>%

mutate(Period = factor(Period, levels = c("None", "Months", "Quarters", "Years"))) %>%

group\_by(Period) %>%

summarise(Return = round(mean(value),3)\*1200,

Volatility = round(sd(value)\*sqrt(12),3)\*100,

Sharpe = round(mean(value)/sd(value)\*sqrt(12),2)+.01,

`Total return` = round(prod(1+value)-1,3)\*100) %>%

knitr::kable(caption = "Risk and returns for different rebalance periods (%)")

# Rebalance for risky

risky\_list <- list()

for(pd in rebal){

risky\_list[[pd]] <- Return.portfolio(ret, wt2, rebalance\_on = pd) %>%

`colnames<-`(pd)

}

risky <- risky\_list %>% bind\_cols() %>%

data.frame() %>%

mutate\_all(as.numeric) %>%

mutate(date = index(risky\_list[["months"]]),

none = as.numeric(port2)) %>%

select(date, none, everything())

risky %>%

rename("None" = none,

"Months" = months,

"Quarters" = quarters,

"Years" = years) %>%

gather(Period,value, -date) %>%

mutate(Period = factor(Period, levels = c("None", "Months", "Quarters", "Years"))) %>%

group\_by(Period) %>%

summarise(Return = round(mean(value),3)\*1200,

Volatility = round(sd(value)\*sqrt(12),3)\*100,

Sharpe = round(mean(value)/sd(value)\*sqrt(12),2)+.01,

`Total return` = round(prod(1+value)-1,3)\*100) %>%

knitr::kable(caption = "Risk and returns for different rebalance periods (%)")

# Rebalance for benchmark

bench\_list <- list()

for(pd in rebal){

bench\_list[[pd]] <- Return.portfolio(bench\_ret, wtb, rebalance\_on = pd) %>%

`colnames<-`(pd)

}

bench\_rb <- bench\_list %>% bind\_cols() %>%

data.frame() %>%

mutate\_all(as.numeric) %>%

mutate(date = index(bench\_list[["months"]]),

none = as.numeric(portb)) %>%

select(date, none, everything())

bench\_rb %>%

rename("None" = none,

"Months" = months,

"Quarters" = quarters,

"Years" = years) %>%

gather(Period,value, -date) %>%

mutate(Period = factor(Period, levels = c("None", "Months", "Quarters", "Years"))) %>%

group\_by(Period) %>%

summarise(Return = round(mean(value)\*12+.0001,3)\*100,

Volatility = round(sd(value)\*sqrt(12),3)\*100,

Sharpe = round(mean(value)/sd(value)\*sqrt(12),2)+.01,

`Total return` = round(prod(1+value)-1,3)\*100) %>%

knitr::kable(caption = "Risk and returns for different rebalance periods (%)")

# Simulate

set.seed(123)

stock\_us <- rnorm(120, 0.08/12, 0.2/sqrt(12))

stock\_world <- rnorm(120, 0.065/12, 0.17/sqrt(12))

bond\_us <- rnorm(120, 0.024/12, 0.1/sqrt(12))

bond\_world <- rnorm(120, 0.025/12, 0.14/sqrt(12))

commod <- rnorm(120, 0.007, 0.057)

wt <- c(0.25, 0.25, 0.2, 0.2, 0.1)

date <- seq(as.Date("2010-02-01"), length = 120, by = "months")-1

port <- as.xts(cbind(stock\_us, stock\_world, bond\_us, bond\_world, commod),

order.by = date)

port\_list <- list()

rebals = c("months", "quarters", "years")

for(pd in rebals){

port\_list[[pd]] <- Return.portfolio(port, wt, rebalance\_on = pd)

}

port\_r <- port\_list %>%

bind\_cols() %>%

data.frame() %>%

mutate\_all(as.numeric) %>%

mutate(date = date,

none = as.numeric(none)) %>%

select(date, none, everything())

port\_r %>%

rename("None" = none,

"Months" = months,

"Quarters" = quarters,

"Years" = years) %>%

gather(Period,value, -date) %>%

mutate(Period = factor(Period, levels = c("None", "Months", "Quarters", "Years"))) %>%

group\_by(Period) %>%

summarise(Return = round(mean(value)\*12+.0001,3)\*100,

Volatility = round(sd(value)\*sqrt(12),3)\*100,

Sharpe = round(mean(value)/sd(value)\*sqrt(12),2)+.01,

`Total return` = round(prod(1+value)-1,3)\*100) %>%

knitr::kable(caption = "Risk and returns for different rebalance periods (%)")

count <- 0

port\_names <- c("None", "Months", "Quarters", "Years")

# paste(toupper(substr(colnames(port\_1)[-1], 1, 1)),

# substr(colnames(port\_1)[-1], 2, nchar(colnames(port\_1)[-1])), sep="")

t\_tests <- c()

for(i in 1:4){

for(j in 2:4){

if(i != j & count != 6){

t\_tests[paste(port\_names[i]," vs. ", port\_names[j])] <- t.test(port\_1[,i+1],

port\_1[,j+1])$p.value

count = count +1

}

}

}

t\_tests

data.frame(pds = names(t\_tests), p\_val = round(as.numeric(t\_tests),2)) %>%

rename("Periods" = pds,

"P-values" = p\_val) %>%

knitr::kable(caption = "T-test for means")

1. We’ve covered this in the previous posts, but only insofar as we limited the number of assets we could choose. We’ll come back to this concept in a slightly different way once we’ve covered capital market expectations. So assume there’s some violent hand-waving going on here![**↩**](https://osm.netlify.app/post/rebalancing-really/#fnref1)
2. Unless it goes bankrupt![**↩**](https://osm.netlify.app/post/rebalancing-really/#fnref2)

**Rebalancing ruminations**

 March 13, 2020

[R](https://osm.netlify.app/categories/r), [Stocks](https://osm.netlify.app/categories/stocks)

Back in the rebalancing saddle! In our last [**post**](https://osm.netlify.com/post/rebalancing-really/) on rebalancing, we analyzed whether rebalancing over different periods would have any effect on mean or risk-adjusted returns for our three (equal, naive, and risky) portfolios. We found little evidence that returns were much different whether we rebalanced monthly, quarterly, yearly, or not at all. Critically, as an astute reader pointed out, if these had been taxable accounts, the rebalancing would likely have been a drag on performance. This is clearly an important point that we will address in later posts.

Even though our tests suggested rebalancing wasn’t worth the effort, we decided to proceed. In truth, while our three ETF portfolios were meant to approximate the major asset classes, they didn’t mimic the benchmark portfolio we constructed all that well. However, that portfolio suffered from limited data. So we simulated a 10-year period for a portfolio whose components matched the historical return and risk profiles of the major asset classes. Here again we found little evidence that rebalancing did much.

But this lack of evidence, doesn’t mean there’s evidence of a lack of better returns from rebalancing. We believe a broader test is required. To that end, we’ll run many simulations of those historical returns to identify the likelihood that rebalancing is worthwhile. Recall that since we’ve been using historical data, we’re really only looking at one observation because historical events are necessarily independent of one another. If Caesar hadn’t crossed the Rubicon, the French Revolution might never have happened! The best way to test rebalancing would be to aggregate data from a thousand alternate realities. Sadly access to such data only shows up in science fiction. We’ll have to make do with the next best thing: the power of R programming. Strap in, we’re headed down the wormhole of simulations!

As before, we’ll run a simulation using the historical mean and standard deviation of return for US and non-US stocks and bonds as well as the S&P GSCI Commodity Total Return index. But this time we’ll run a 1,000 of those simulations and then aggregate them.[**1**](https://osm.netlify.app/post/rebalancing-ruminations/#fn1) Additionally, we’ll equal weight the assets as a starting point. Our first test will be to see how often the mean return of one rebalancing strategy exceeds that of another.

Chart, bar chart

Description automatically generated

Not exactly a resounding result. But we see that no rebalancing tends to outperform other rebalancing strategies a little bit better than a coin flip. Comparing rebalancing strategies against one another shows that rebalancing more frequently tends to underperform rebalancing less frequently. But here it’s only slightly less than coin flip too.

Let’s look at the mean returns and range of outcomes to get a sense of the overall performance.

Chart, box and whisker chart

Description automatically generated

The white horizontal lines represent the average of all the simulations’ annualized returns. The bottom and top edge of the boxes represent the middle 50% of all observations. There doesn’t seem to be much of difference, though it’s clear that no rebalancing results in a wider range of outcomes. Eyeballing this suggests that there really shouldn’t be much of a difference in the mean return. However, to check the significance we need to be careful how we run the test. If we test the mean returns from no rebalancing for all simulations against the other rebalancing periods we’ll be comparing results from different simulations. This is different from comparing returns within each simulation using t-tests and then calculating mean p-value from all the simulations. The latter seems more accurate.[**2**](https://osm.netlify.app/post/rebalancing-ruminations/#fn2)

| Table 1: Aggregate p-values for simulation | |
| --- | --- |
| **Comparison** | **P-value** |
| None vs. Months | 0.84 |
| None vs. Quarters | 0.84 |
| None vs. Years | 0.85 |
| Months vs. Quarters | 0.98 |
| Months vs. Years | 0.95 |
| Quarters vs. Years | 0.96 |

Another strike against rebalancing. We see that on average, the p-values are quite high, suggesting any difference in mean returns is likely due to chance. Now let’s check risk-adjusted returns using the Sharpe ratio.

| Table 2: Sharpe ratios by rebalancing period | |
| --- | --- |
| **Period** | **Ratio** |
| None | 0.72 |
| Months | 0.76 |
| Quarters | 0.76 |
| Years | 0.76 |

No rebalancing produces a slightly worse risk-adjusted return than rebalancing. But is the difference significant? Our guess is probably not. Performing a robust test would require some more complicated math that would be beyond the scope of this post to explain. Instead, we’ll look at the number of occurrences in which one rebalancing regime produced a better Sharpe ratio than another. The assumption is that if it’s greater than 90%, the differences are probably significant.

| Table 3: Frequency of getting a better Sharpe ratio (%) | |
| --- | --- |
| **Periods** | **Occurence** |
| None vs. Months | 24.7 |
| None vs. Quarters | 24.4 |
| None vs. Years | 25.6 |
| Months vs. Quarters | 51.0 |
| Months vs. Years | 56.6 |
| Quarters vs. Years | 56.5 |

As the table shows even though rebalancing did produce better Sharpe ratios than no rebalancing a majority of the time, it didn’t do so more than 90% of the time.

Is this the end of our analysis? No because we looked at an equal-weighted portfolio. Perhaps a different weighting scheme would produce different relative results. The prior weighting scheme effectively had a 40% weighting to both stocks and bonds, and a 20% weighting to commodities. Our new weighting scheme will have a 60% weighting to stocks, a 35% weighting to bonds, and a 5% weighting to commodities. This is close to the empirical recommendation of a 60/40 stock/bond portfolio many financial advisors recommend. We’ll run 1,000 simulations and output the same analyses as above.

First, the frequency of outperformance.

Chart, bar chart

Description automatically generated

On a different weighting scheme, rebalancing appears to outperform not rebalancing, but only slightly better than a coin toss. Interestingly, there appears to be almost no difference in performance among different rebalancing periods. Are these results significant?

| Table 4: Aggregate p-values for simulation | |
| --- | --- |
| **Comparison** | **P-value** |
| None vs. Months | 0.87 |
| None vs. Quarters | 0.87 |
| None vs. Years | 0.88 |
| Months vs. Quarters | 0.98 |
| Months vs. Years | 0.96 |
| Quarters vs. Years | 0.96 |

Again, not much to see here. Even if the occurrence is somewhat better than 50/50, the differences aren’t significant. We’ll move along to at the Sharpe ratio for the different strategies.

| Table 5: Sharpe ratios by rebalancing period | |
| --- | --- |
| **Period** | **Ratio** |
| None | 0.63 |
| Months | 0.67 |
| Quarters | 0.67 |
| Years | 0.66 |

Like the previous simulation, no rebalancing returned a Sharpe ratio that was 30-40bps lower than the rebalancing strategies. To see if that’s significant we calculate the frequency in which one strategy had a better Sharpe ratio than another.

| Table 6: Frequency of getting a better Sharpe ratio (%) | |
| --- | --- |
| **Periods** | **Occurence** |
| None vs. Months | 23.3 |
| None vs. Quarters | 23.7 |
| None vs. Years | 26.2 |
| Months vs. Quarters | 53.8 |
| Months vs. Years | 60.7 |
| Quarters vs. Years | 59.5 |

Here too the frequency of a better Sharpe ratio was quite similar to the previous weighting scheme. And there was no occurrence better than 90%.

Where does this leave us? Rebalancing does not appear to produce better returns, risk-adjusted or otherwise. Are there problems with this study? Yes. First, we only tested rebalancing based on timing. In our mind, timing alone is a sort of silly reason to rebalance. What difference does it make to deciding whether to rebalance or not whether a month or a year has passed? Rebalancing based on a threshold seems more logical. Of course, one can rebalance at the end of certain periods if a threshold has been crossed to prevent over trading. Thus, we’d need to conduct a different rebalancing simulation based on some threshold.

The second problem with our tests is that our simulations were based on random samples pulled from a normal distribution. Asset returns have been shown to be non-normal; that is, (without getting too deep into stats) they don’t have a nice bell-shaped curve, are often skewed to one side, and have a lot of outliers. That means our simulated results could be much different than what is likely to occur in reality, perhaps significantly.

A third problem has to do with correlations: they change and they’re often serial. On the first issue, correlations change over time. In periods of high positive correlation, there’s probably no reason to rebalance, as the original weighting likely remains stable. But in periods of low correlation, rebalancing should help because when one is selling upwardly trending assets to buy downward ones, and those trends reverse, the rebalanced portfolio theoretically would exploit that trend reversal.

But there’s a nuance. Rebalancing implicitly assumes correlations don’t change much over time. They certainly aren’t supposed to flip signs. Assets that are negatively correlated aren’t supposed to become positive. Yet this happens and might happen right after one rebalances, potentially nullifying the intended purpose of rebalancing.

We did not explicitly simulate different correlation scenarios, as we assumed the randomness of simulating each asset’s return implicitly introduced different correlations. That assumption might not be correct. Do the simulations produce enough diverse correlation scenarios to do justice to rebalancing? To answer that would require an entire post on its own, which will have to wait.

Then there’s serial correlation, or the phenomenon where current period results are related to any number of past periods. This is important because time series data, especially asset returns, exhibit modest to high amounts of serial correlation. If our simulations don’t account for that phenomenon, then we’re not really approximating real data.

Many of these problems can be resolved by sampling from historical data. Since historical data is non-normal, skewed, and features more outliers, then that would be expressed in the simulations. The same goes for changing and serial correlations, though adjusting for that requires a bit more art.[**3**](https://osm.netlify.app/post/rebalancing-ruminations/#fn3)

Clearly there’s a lot more wood to chop to figure out if rebalancing produces better risk-adjusted returns given the issues we’ve highlighted. From a reproducibility standpoint, it’s tough to find the last 50 years of global stock and bond returns from a freely available source. A bigger question is whether the effort is worth it. If rebalancing really were a source of better returns, wouldn’t it be obvious even from a somewhat flawed simulation? We’d love to know what our readers think. Want more on rebalancing or is it time to move on? Please send us a message at the email after the code with your view.

For those interested in reading more about rebalancing, we’ve included a (hopefully!) balanced representation of the argument in the links below.

Links:

* [**Rebalancing best practices**](https://www.vanguard.com/pdf/ISGPORE.pdf)
* [**Smart rebalancing guide**](https://personal.vanguard.com/pdf/ISGGBOT.pdf)
* [**Rebalancing bonus**](http://www.efficientfrontier.com/ef/996/rebal.htm)
* [**Rebalancing alpha**](https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2202736)

Until next time, here’s the code behind our analyses.

# Load packages

library(tidyquant)

library(tidyverse)

## Create rebalancing simulation function

rebal\_sim <- function(wt,...){

stock\_us <- rnorm(120, 0.08/12, 0.2/sqrt(12))

stock\_world <- rnorm(120, 0.065/12, 0.17/sqrt(12))

bond\_us <- rnorm(120, 0.024/12, 0.1/sqrt(12))

bond\_world <- rnorm(120, 0.025/12, 0.14/sqrt(12))

commod <- rnorm(120, 0.007, 0.057)

if(missing(wt)){

wt <- rep(.2, 5)

}else{

wt <- wt

}

date <- seq(as.Date("2010-02-01"), length = 120, by = "months")-1

port <- as.xts(cbind(stock\_us, stock\_world, bond\_us, bond\_world, commod),

order.by = date)

port\_list <- list()

rebals = c("none","months", "quarters", "years")

for(pd in rebals){

if(pd == "none"){

port\_list[[pd]] <- Return.portfolio(port, wt)

}else{

port\_list[[pd]] <- Return.portfolio(port, wt, rebalance\_on = pd)

}

}

port\_r <- port\_list %>%

bind\_cols() %>%

data.frame() %>%

mutate\_all(as.numeric) %>%

mutate(date = date) %>%

select(date, everything())

corr <- cor(cbind(stock\_us, stock\_world, bond\_us, bond\_world, commod))

results <- list(port\_r = port\_r, corr = corr)

results

}

## Rum simulation

set.seed(123)

rebal\_test <- list()

for(i in 1:1000){

rebal\_test[[i]] <- rebal\_sim()

}

## Find percentage of time one rebalancing period generates higher returns than another

# Create means comparison function

freq\_comp <- function(df){

count <- 1

opf <- data.frame(comp = rep(0,6), prob = rep(0,6))

port\_names <- c("None", "Months", "Quarters", "Years")

for(i in 1:4){

for(j in 2:4){

if(i < j & count < 7){

opf[count,1] <- paste(port\_names[i], " vs. ", port\_names[j])

opf[count,2] <- mean(df[,i]) > mean(df[,j])

count <- count + 1

}

}

}

opf

}

# Aggregate function across simulations

prop\_df <- matrix(rep(0,6000), nrow = 1000)

for(i in 1:1000){

prop\_df[i,] <- freq\_comp(rebal\_test[[i]][,2:5])[,2]

}

short\_names <- c("n", "m", "q", "y")

df\_names <- c()

count <- 1

for(i in 1:4){

for(j in 2:4){

if(i < j & count < 7){

df\_names[count] <- paste(short\_names[i], " vs. ", short\_names[j])

count <- count+1

}

}

}

rebal\_names <- c("None", "Months", "Quarters", "Years")

long\_names <- c()

count <- 1

for(i in 1:4){

for(j in 2:4){

if(i < j & count < 7){

long\_names[count] <- paste(rebal\_names[i], " vs. ", rebal\_names[j])

count <- count + 1

}

}

}

prop\_df <- prop\_df %>% data.frame %>%

`colnames<-`(df\_names)

prop\_df %>%

summarize\_all(mean) %>%

`colnames<-`(long\_names) %>%

gather(key, value) %>%

mutate(key = factor(key, levels = long\_names)) %>%

ggplot(aes(key,value\*100)) +

geom\_bar(stat = "identity", fill = "blue")+

labs(x= "",

y = "Frequency (%)",

title = "Number of times one rebalancing strategy outperforms another") +

geom\_text(aes(label = value\*100), nudge\_y = 2.5)

## Average results

rebal\_mean\_df <- data.frame(none = rep(0,1000),

monthly = rep(0,1000),

quarterly = rep(0,1000),

yearly = rep(0,1000))

for(i in 1:1000){

rebal\_mean\_df[i,] <- colMeans(rebal\_test[[i]][,2:5]) %>% as.vector()

}

# Boxplot of reults

rebal\_mean\_df %>%

`colnames<-`(port\_names) %>%

gather(key,value) %>%

mutate(key = factor(key, levels = port\_names)) %>%

ggplot(aes(key,value\*1200)) +

geom\_boxplot(fill = "blue", color = "blue", outlier.colour = "red") +

stat\_summary(geom = "crossbar", width=0.7, fatten=0, color="white",

fun.data = function(x){ return(c(y=mean(x), ymin=mean(x), ymax=mean(x))) })+

labs(x = "",

y = "Return (%)",

title = "Range of mean annualized returns by rebalancing period")

# Create function

t\_test\_func <- function(df){

count <- 1

t\_tests <- c()

for(i in 1:4){

for(j in 2:4){

if(i < j & count < 7){

t\_tests[count] <- t.test(df[,i],df[,j])$p.value

count <- count +1

}

}

}

t\_tests

}

t\_test\_func(rebal\_test[[995]][,2:5])

t\_tests <- matrix(rep(0,6000), ncol = 6)

for(i in 1:1000){

t\_tests[i,] <- t\_test\_func(rebal\_test[[i]][,2:5])

}

t\_tests <- t\_tests %>%

data.frame() %>%

`colnames<-`(long\_names)

t\_tests %>%

summarise\_all(function(x) round(mean(x),2)) %>%

gather(Comparison, `P-value`) %>%

knitr::kable(caption = "Aggregate p-values for simulation")

## Sharpe ratios

sharpe <- matrix(rep(0,4000), ncol = 4)

for(i in 1:1000){

sharpe[i,] <- apply(rebal\_test[[i]][,2:5], 2, mean)/apply(rebal\_test[[i]][,2:5],2, sd) \* sqrt(12)

}

sharpe <- sharpe %>%

data.frame() %>%

`colnames<-`(port\_names)

# Table

sharpe %>%

summarise\_all(mean) %>%

gather(Period, Ratio) %>%

mutate(Ratio = round(Ratio,3)) %>%

knitr::kable(caption = "Sharpe ratios by rebalancing period")

sharpe\_t <- data.frame(Periods = names(t\_tests), Occurence = rep(0,6))

count <- 1

for(i in 1:4){

for(j in 2:4){

if(i <j & count < 7){

sharpe\_t[count,2] <- mean(sharpe[,i] > sharpe[,j])

count <- count + 1

}

}

}

sharpe\_t %>%

knitr::kable(caption = "Frequency of better Sharpe ratio")

# Load data

wt1 <- (0.3, 0.3, 0.2, 0.15, 0.05)

rebal\_wt <- rebal\_sim(wt=wt1)

# Aggregate function across simulations

means\_wt\_df <- matrix(rep(0,6000), nrow = 1000)

for(i in 1:1000){

means\_wt\_df[i,] <- means\_comp(rebal\_wt[[i]][,2:5])[,2]

}

# Graph

means\_wt\_df %>%

summarize\_all(mean) %>%

`colnames<-`(long\_names) %>%

gather(key, value) %>%

mutate(key = factor(key, levels = long\_names)) %>%

ggplot(aes(key,value\*100)) +

geom\_bar(stat = "identity", fill = "blue")+

labs(x= "",

y = "Frequency (%)",

title = "Number of times one rebalancing strategy outperforms another") +

geom\_text(aes(label = value\*100), nudge\_y = 2.5)

## Run t-test

t\_tests\_wt <- matrix(rep(0,6000), ncol = 6)

for(i in 1:1000){

t\_tests\_wt[i,] <- t\_test\_func(rebal\_wt[[i]][,2:5])

}

t\_tests\_wt <- t\_tests\_wt %>%

data.frame() %>%

`colnames<-`(long\_names)

t\_tests\_wt %>%

summarise\_all(function(x) round(mean(x),2)) %>%

gather(Comparison, `P-value`) %>%

knitr::kable(caption = "Aggregate p-values for simulation")

## Sharpe ratios0

sharpe\_wt <- matrix(rep(0,4000), ncol = 4)

for(i in 1:1000){

sharpe\_wt[i,] <- apply(rebal\_wt[[i]][,2:5], 2, mean)/apply(rebal\_wt[[i]][,2:5],2, sd) \* sqrt(12)

}

sharpe\_wt <- sharpe\_wt %>%

data.frame() %>%

`colnames<-`(port\_names)

# table

sharpe\_wt %>%

summarise\_all(mean) %>%

gather(Period, Ratio) %>%

mutate(Ratio = round(Ratio,2)) %>%

knitr::kable(caption = "Sharpe ratios by rebalancing period")

# Permutation test for sharpe

sharpe\_wt\_t <- data.frame(Periods = names(t\_tests\_wt), Occurence = rep(0,6))

count <- 1

for(i in 1:4){

for(j in 2:4){

if(i <j & count < 7){

sharpe\_wt\_t[count,2] <- mean(sharpe\_wt[,i] > sharpe\_wt[,j])

count <- count + 1

}

}

}

# Table

sharpe\_wt\_t %>%

knitr::kable(caption = "Frequency of getting a better Sharpe ratio")

1. If you try to reproduce the simulation note that it may take a while to run. If anyone has a better idea on how to write better code for this simulation, please send us an email.[**↩**](https://osm.netlify.app/post/rebalancing-ruminations/#fnref1)
2. This method is perhaps not academically correct, but we hope it is sufficient to reveal some insights.[**↩**](https://osm.netlify.app/post/rebalancing-ruminations/#fnref2)
3. Clearly, beyond the scope of this post, but we’d need to sample returns in a block, That is, we’d choose a representative length and then sample blocks of returns based on that length. The length would approximate the highest serial correlation period of that particular asset. But if you’re simulating more than one asset, each with a different order of serial correlation, which lag do you use?[**↩**](https://osm.netlify.app/post/rebalancing-ruminations/#fnref3)

**Rebalancing history**

 March 20, 2020

[R](https://osm.netlify.app/categories/r), [Stocks](https://osm.netlify.app/categories/stocks)

Our last [**post**](https://osm.netlify.com/post/rebalancing-ruminations/) on rebalancing struck an equivocal note. We ran a thousand simulations using historical averages across different rebalancing regimes to test whether rebalancing produced better absolute or risk-adjusted returns. The results suggested it did not. But we noted many problems with the tests—namely, unrealistic return distributions and correlation scenarios. We argued that if we used actual historical data and sampled from it, we might resolve many of these issues. But we also asked our readers whether it was worthwhile to test further. Based on the responses and page views, we believe the interest is there, so we’ll proceed!

As we mentioned, historical data more closely approximates the fat-tailed, skewed distribution common to asset returns. But only if you have a long enough time series. While we weren’t able to find 50 years worth of major asset class returns, we were able to compile a 20-year series that includes two market downturns. The data isn’t all from the same source, unfortunately. But it is fairly reputable—Vanguard’s stock and US bond index funds, emerging market bond indices from the St. Louis Fed, and the S&P GSCI commodity index. The code will show how we aggregated it for those interested. Using this data series we should be able to test rebalancing more robustly.

Before we proceed, a brief word on methodology. To run the simulation, we need to sample (with replacement) from our twenty year period and combine each sample into an entire series. To capture the non-normal distribution and serial correlation of asset returns, we can’t just sample one return, however. We need to sample a block of returns. This allows us to approximate the serial correlation of individual assets as well as the correlation between assets. But how long should the block be? Trying to answer that can get pretty complicated, pretty quickly.[**1**](https://osm.netlify.app/post/rebalancing-history/#fn1) We decided to take a shortcut and use a simple block of 6 periods. This equates to six months, since our series is monthly returns. There’s nothing magical about this number but it does feature as a period used in academic studies on momentum, a topic beyond the scope of this post.[**2**](https://osm.netlify.app/post/rebalancing-history/#fn2)

We sample six months of returns at at time. Repeat 42 times to get over 20 years of data. Repeat to create 1000 portfolios. From there we apply the different rebalancing regimes on each of the portfolios and then aggregate the data. As before, we first use an equal-weighting, and then a 60/35/5 weighting for stocks, bonds, and commodities. Let’s see what we get.

First, we look at the average return for equal-weighted portfolios by rebalancing regime along with the range of outcomes.

Chart, box and whisker chart

Description automatically generated

Recall, the white line is the mean and the top and bottom of the boxes represent the middle 50% of outcomes, Interestingly, no rebalancing had far more positive and far less negative outliers (the red dots) than any of the rebalancing regimes.

Given where the averages line up, it doesn’t look like there are significant differences. Let’s run some t-tests for completeness.

| Table 1: Aggregate p-values for simulation | |
| --- | --- |
| **Comparison** | **P-value** |
| None vs. Months | 0.87 |
| None vs. Quarters | 0.87 |
| None vs. Years | 0.88 |
| Months vs. Quarters | 0.97 |
| Months vs. Years | 0.95 |
| Quarters vs. Years | 0.97 |

As expected, the p-values are quite high, meaning that any differences in mean returns are likely due to chance.

Now we’ll check on the number of times each rebalancing strategy outperforms the others.

Chart, bar chart

Description automatically generated

A dramatic result! No rebalancing beat the other strategies a majority of the time and less rebalancing outperformed more most of the time too. Now for the crux. Does rebalancing lead to better risk-adjusted returns as calculated by the Sharpe ratio.

| Table 2: Sharpe ratios by rebalancing period | |
| --- | --- |
| **Period** | **Ratio** |
| None | 0.76 |
| Months | 0.74 |
| Quarters | 0.75 |
| Years | 0.76 |

Not much difference. Recall that from our previous simulations, no rebalancing actually generated a slightly worse Sharpe ratio by about 30-40 bps. But that result occurred less than 90% of the time, so it could be due to randomness. Let’s check the Sharpe ratios for the present simulation.

| Table 3: Frequency of a better Sharpe ratio (%) | |
| --- | --- |
| **Periods** | **Occurence** |
| None vs. Months | 60.2 |
| None vs. Quarters | 53.4 |
| None vs. Years | 48.3 |
| Months vs. Quarters | 3.7 |
| Months vs. Years | 8.2 |
| Quarters vs. Years | 22.1 |

No rebalancing generates a better Sharpe ratio a majority of the time, but not enough to conclude it isn’t due to chance. Interestingly, the frequency with which quarterly and yearly rebalancing produce better Sharpe ratios than monthly rebalancing looks significant. In both cases the frequency is greater than 90% of the time. That the lower frequency rebalancing outperforms the higher frequency likely plays a role in the significance of the Sharpe ratios, but is an area of investigation we’ll shelve for now.

Let’s move to the next simulation where we weight the portfolios 60/35/5 for stocks, bonds, and commodities. First, we show the boxplot of mean returns and range of outcomes.

Chart, box and whisker chart

Description automatically generated

Like the equal-weighted simulations, the means don’t look that dissimilar and no rebalancing generates more positive and less negative outliers than other rebalancing regimes. We can say almost undoubtedly that the differences in average returns, if there are any, is likely due to chance. The p-values from the t-tests we show below should prove that.

| Table 4: Aggregate p-values for simulation | |
| --- | --- |
| **Comparison** | **P-value** |
| None vs. Months | 0.91 |
| None vs. Quarters | 0.92 |
| None vs. Years | 0.92 |
| Months vs. Quarters | 0.98 |
| Months vs. Years | 0.97 |
| Quarters vs. Years | 0.98 |

Now let’s calculate and present the frequency of outperformance by rebalancing strategy.

Chart, histogram

Description automatically generated

No rebalancing outperforms again! Less frequent rebalancing outperforms more frequent. And risk-adjusted returns?

| Table 5: Sharpe ratios by rebalancing period | |
| --- | --- |
| **Period** | **Ratio** |
| None | 0.66 |
| Months | 0.66 |
| Quarters | 0.67 |
| Years | 0.68 |

Here, no rebalancing performed slightly worse than rebalancing quarterly or yearly. How likely should we believe these results to be significant?

| Table 6: Frequency of a better Sharpe ratio (%) | |
| --- | --- |
| **Periods** | **Occurence** |
| None vs. Months | 47.8 |
| None vs. Quarters | 40.9 |
| None vs. Years | 35.5 |
| Months vs. Quarters | 2.2 |
| Months vs. Years | 6.4 |
| Quarters vs. Years | 21.3 |

Slightly worse than 50/50 for no rebalancing. But less frequent rebalancing appears to have the potential to produce higher risk-adjusted returns that more frequent rebalancing.

Let’s briefly sum up what we’ve discovered thus far. Rebalancing does not seem to produce better risk-adjusted returns. If we threw in taxation and slippage, we think rebalancing might likely be a significant underperformer most of the time.

Does this mean you should never rebalance? No. You should definitely rebalance if you’ve got a crystal ball. Baring that, if your risk-return parameters change, then you should rebalance. But it would not be bringing the weights back to their original targets; rather, it would be new targets. A entirely separate case.

What do we see as the biggest criticism of the foregoing analysis? That it was a straw man argument. In practice, few professional investors rebalance because it’s July 29th or October 1st. Of course, there is quarter-end and year-end rebalancing, but those dates are usually coupled with a threshold. That is, only rebalance if the weights have exceeded some threshold, say five or ten percentage points from target. Analyzing the effects of only rebalancing based on thresholds would require more involved code on our part.[**3**](https://osm.netlify.app/post/rebalancing-history/#fn3) Given the results thus far, we’re not convinced that rebalancing based on the thresholds would produce meaningfully better risk-adjusted returns.

However, rebalancing based on changes in risk-return constraints might do so. Modeling that would be difficult since we’d also have to model (or assume) new risk-return forecasts. But we could model the traditional shift recommended by financial advisors to clients as they age; that is, slowly shifting from high-risk to low-risk assets. In other words, lower the exposure to stocks and increase the exposure to bonds.

As a toy example, we use our data set to compare a no rebalancing strategy with an initial 60/35/5 split between stocks, bonds, and commodities to a yearly rebalancing strategy that starts at a 90/5/5 split and changes to a 40/60/0 split over the period. The Sharpe ratio for the rebalanced portfolio is actually a bit worse than the no rebalancing one. Mean returns are very close. Here’s the graph of the cumulative return.

Chart, line chart

Description automatically generated

This is clearly one example and highly time-dependent. But we see that rebalancing wasn’t altogether different than not, and we’re not including tax and slippage effects. To test this notion we’d have to run some more simulations, but that will be for another post.

We’ll end this post with a question for our readers. Are you convinced rebalancing doesn’t improve returns or do you think more analysis is required? Please send us your answer to nbw dot osm at gmail dot com. Until next time, here’s all the code behind the simulations, analyses, and charts.

## Load packages

library(tidyquant)

library(tidyverse)

### Load data

## Stocks

symbols <- c("VTSMX", "VGTSX", "VBMFX", "VTIBX")

prices <- getSymbols(symbols, src = "yahoo",

from = "1990-01-01",

auto.assign = TRUE) %>%

map(~Ad(get(.))) %>%

reduce(merge) %>%

`colnames<-`(tolower(symbols))

## Bonds

# Source for bond indices:

# https://fred.stlouisfed.org/categories/32413

em\_hg <- getSymbols("BAMLEMIBHGCRPITRIV",

src = "FRED",

from = "1990-01-01",

auto.assign = FALSE)

em\_hg <- em\_hg %>% na.locf()

em\_hy <- getSymbols("BAMLEMHBHYCRPITRIV",

src = "FRED",

from = "1990-01-01",

auto.assign = FALSE)

em\_hy <- em\_hy %>% na.locf()

# Commodity data

# Source for commodity data

# https://www.investing.com/indices/sp-gsci-commodity-total-return-historical-data

# Unfortunately, the data doesn't open into a separate link so you'll need to download it into a

# csv file unless you're good a web scraping. We're not. Note too, the dates get a little funky

# when being transferred into the csv, so you'll need to clean that up. Finally, the dates are

# give as beginning of the month. But when we spot checked a few, they were actually end of the

# month, which lines up with the other data

cmdty <- read\_csv("sp\_gsci.csv")

cmdty$Date <- as.Date(cmdty$Date,"%m/%d/%Y")

cmd\_price <- cmdty %>%

filter(Date >="1998-12-01", Date <="2019-12-31")

## Merged

merged <- merge(prices[,1:3], em\_hg, em\_hy)

colnames(merged) <- c("us\_stock", "intl\_stock", "us\_bond", "em\_hg", "em\_hy")

merged <- merged["1998-12-31/2019-12-31"] %>% na.locf()

merge\_mon <- to.monthly(merged, indexAt = "lastof", OHLC = FALSE)

merge\_mon$cmdty <- cmd\_price$Price

merge\_yr <- to.yearly(merge\_mon, indexAt = "lastof", OHLC = FALSE)

merge\_ret <- ROC(merge\_mon, type = "discrete") %>% na.omit()

merge\_ret\_yr <- ROC(merge\_yr, type = "discrete") %>% na.omit()

## Data frame

df <- data.frame(date = index(merge\_ret), coredata(merge\_ret))

df\_yr <- data.frame(date = index(merge\_ret\_yr), coredata(merge\_ret\_yr))

### Block sampling

## Create function

block\_samp <- function(dframe,block,cycles){

idx <- seq(1,block\*cycles,block)

assets <- ncol(dframe)

size <- block\*cycles

mat <- matrix(rep(0,assets\*size), ncol = assets)

for(i in 1:cycles){

start <-sample(size,1)

if(start <= (size - block + 1)){

end <- start + block -1

len <- start:end

}else if(start > (size - block + 1) & start < size){

end <- size

step <- block - (end - start) - 1

if(step == 1){

adder <- 1

}else{

adder <- 1:step

}

len <- c(start:end, adder)

}else{

adder <- 1:(block - 1)

len <- c(start, adder)

}

mat[idx[i]:(idx[i]+block-1),] <- data.matrix(df[len,2:7])

}

mat

}

# Create 1000 samples

set.seed(123)

block\_list <- list()

for(i in 1:1000){

block\_list[[i]] <- block\_samp(df[,2:7], 6, 42)

}

### Rebalancing on simulation

## Create function

rebal\_func <- function(port, wt, ...){

if(missing(wt)){

wt <- rep(1/ncol(port), ncol(port))

}else{

wt <- wt

}

port <- ts(port, start = c(1999,1), frequency = 12)

port\_list <- list()

rebals = c("none","months", "quarters", "years")

for(pd in rebals){

if(pd == "none"){

port\_list[[pd]] <- Return.portfolio(port, wt) %>%

`colnames<-`(pd)

}else{

port\_list[[pd]] <- Return.portfolio(port, wt, rebalance\_on = pd)%>%

`colnames<-`(pd)

}

}

port\_r <- port\_list %>%

bind\_cols() %>%

data.frame()

port\_r

}

## Run function on simulations

# Note this may take 10 minutes to run. We hope to figure out a way to speed this up in later

# versions.

rebal\_test <- list()

for(i in 1:1000){

rebal\_test[[i]] <- rebal\_func(block\_list[[i]])

}

### Analyze results

## Average results

rebal\_mean\_df <- data.frame(none = rep(0,1000),

monthly = rep(0,1000),

quarterly = rep(0,1000),

yearly = rep(0,1000))

for(i in 1:1000){

rebal\_mean\_df[i,] <- colMeans(rebal\_test[[i]]) %>% as.vector()

}

port\_names <- c("None", "Months", "Quarters", "Years")

# Boxplot of reults

rebal\_mean\_df %>%

`colnames<-`(port\_names) %>%

gather(key,value) %>%

mutate(key = factor(key, levels = port\_names)) %>%

ggplot(aes(key,value\*1200)) +

geom\_boxplot(fill = "blue", color = "blue", outlier.colour = "red") +

stat\_summary(geom = "crossbar", width=0.7, fatten=0, color="white",

fun.data = function(x){ return(c(y=mean(x), ymin=mean(x), ymax=mean(x))) })+

labs(x = "",

y = "Return (%)",

title = "Range of mean annualized returns by rebalancing period")

## Find percentage of time one rebalancing period generates higher returns than another

# Create means comparison function

freq\_comp <- function(df){

count <- 1

opf <- data.frame(comp = rep(0,6), prob = rep(0,6))

port\_names <- c("None", "Months", "Quarters", "Years")

for(i in 1:4){

for(j in 2:4){

if(i < j & count < 7){

opf[count,1] <- paste(port\_names[i], " vs. ", port\_names[j])

opf[count,2] <- mean(df[,i]) > mean(df[,j])

count <- count + 1

}

}

}

opf

}

# Aggregate function across simulations

prop\_df <- matrix(rep(0,6000), nrow = 1000)

for(i in 1:1000){

prop\_df[i,] <- freq\_comp(rebal\_test[[i]])[,2]

}

long\_names <- c()

count <- 1

for(i in 1:4){

for(j in 2:4){

if(i < j & count < 7){

long\_names[count] <- paste(port\_names[i], " vs. ", port\_names[j])

count <- count + 1

}

}

}

prop\_df %>%

data.frame() %>%

summarize\_all(mean) %>%

`colnames<-`(long\_names) %>%

gather(key, value) %>%

mutate(key = factor(key, levels = long\_names)) %>%

ggplot(aes(key,value\*100)) +

geom\_bar(stat = "identity", fill = "blue")+

labs(x= "",

y = "Frequency (%)",

title = "Number of times one rebalancing strategy outperforms another") +

geom\_text(aes(label = value\*100), nudge\_y = 2.5)

## Run t-test

# Create function

t\_test\_func <- function(df){

count <- 1

t\_tests <- c()

for(i in 1:4){

for(j in 2:4){

if(i < j & count < 7){

t\_tests[count] <- t.test(df[,i],df[,j])$p.value

count <- count +1

}

}

}

t\_tests

}

t\_tests <- matrix(rep(0,6000), ncol = 6)

for(i in 1:1000){

t\_tests[i,] <- t\_test\_func(rebal\_test[[i]])

}

t\_tests <- t\_tests %>%

data.frame() %>%

`colnames<-`(long\_names)

t\_tests %>%

summarise\_all(function(x) round(mean(x),2)) %>%

gather(Comparison, `P-value`) %>%

knitr::kable(caption = "Aggregate p-values for simulation")

## Sharpe ratios

sharpe <- matrix(rep(0,4000), ncol = 4)

for(i in 1:1000){

sharpe[i,] <- apply(rebal\_test[[i]], 2, mean)/apply(rebal\_test[[i]], 2, sd) \* sqrt(12)

}

sharpe <- sharpe %>%

data.frame() %>%

`colnames<-`(port\_names)

# Table

sharpe %>%

summarise\_all(mean) %>%

gather(Period, Ratio) %>%

mutate(Ratio = round(Ratio,2)) %>%

knitr::kable(caption = "Sharpe ratios by rebalancing period")

# Permutation test for sharpe

sharpe\_t <- data.frame(Periods = names(t\_tests), Occurence = rep(0,6))

count <- 1

for(i in 1:4){

for(j in 2:4){

if(i <j & count < 7){

sharpe\_t[count,2] <- mean(sharpe[,i] > sharpe[,j])

count <- count + 1

}

}

}

# table

sharpe\_t %>%

knitr::kable(caption = "Frequency of better Sharpe ratio")

## Rum simulation pt 2

# This may take 10 minutes or so to run.

wt1 <- c(0.30, 0.30, 0.2, 0.075, 0.075, 0.05)

rebal\_wt <- list()

for(i in 1:1000){

rebal\_wt[[i]] <- rebal\_func(block\_list[[i]], wt1)

}

## Average results

rebal\_wt\_mean\_df <- data.frame(none = rep(0,1000),

monthly = rep(0,1000),

quarterly = rep(0,1000),

yearly = rep(0,1000))

for(i in 1:1000){

rebal\_wt\_mean\_df[i,] <- colMeans(rebal\_test[[i]]) %>% as.vector()

}

# Boxplot

rebal\_wt\_mean\_df %>%

`colnames<-`(port\_names) %>%

gather(key,value) %>%

mutate(key = factor(key, levels = port\_names)) %>%

ggplot(aes(key,value\*1200)) +

geom\_boxplot(fill = "blue", color = "blue", outlier.colour = "red") +

stat\_summary(geom = "crossbar", width=0.7, fatten=0, color="white",

fun.data = function(x){ return(c(y=mean(x), ymin=mean(x), ymax=mean(x))) })+

labs(x = "",

y = "Return (%)",

title = "Range of mean annualized returns by rebalancing period")

## Find percentage of time one rebalancing period generates higher returns than another

# Aggregate function across simulations

prop\_wt\_df <- matrix(rep(0,6000), nrow = 1000)

for(i in 1:1000){

prop\_wt\_df[i,] <- freq\_comp(rebal\_wt[[i]])[,2]

}

prop\_wt\_df %>%

data.frame() %>%

summarize\_all(mean) %>%

`colnames<-`(long\_names) %>%

gather(key, value) %>%

mutate(key = factor(key, levels = long\_names)) %>%

ggplot(aes(key,value\*100)) +

geom\_bar(stat = "identity", fill = "blue")+

labs(x= "",

y = "Frequency (%)",

title = "Number of times one rebalancing strategy outperforms another") +

geom\_text(aes(label = value\*100), nudge\_y = 2.5)

## Run t-test

t\_tests\_wt <- matrix(rep(0,6000), ncol = 6)

for(i in 1:1000){

t\_tests\_wt[i,] <- t\_test\_func(rebal\_wt[[i]])

}

t\_tests\_wt <- t\_tests\_wt %>%

data.frame() %>%

`colnames<-`(long\_names)

t\_tests\_wt %>%

summarise\_all(function(x) round(mean(x),2)) %>%

gather(Comparison, `P-value`) %>%

knitr::kable(caption = "Aggregate p-values for simulation")

## Sharpe ratios

sharpe\_wt <- matrix(rep(0,4000), ncol = 4)

for(i in 1:1000){

sharpe\_wt[i,] <- apply(rebal\_wt[[i]], 2, mean)/apply(rebal\_wt[[i]],2, sd) \* sqrt(12)

}

sharpe\_wt <- sharpe\_wt %>%

data.frame() %>%

`colnames<-`(port\_names)

# table

sharpe\_wt %>%

summarise\_all(mean) %>%

gather(Period, Ratio) %>%

mutate(Ratio = round(Ratio,2)) %>%

knitr::kable(caption = "Sharpe ratios by rebalancing period")

# Permutation test for sharpe

sharpe\_wt\_t <- data.frame(Periods = names(t\_tests\_wt), Occurence = rep(0,6))

count <- 1

for(i in 1:4){

for(j in 2:4){

if(i <j & count < 7){

sharpe\_wt\_t[count,2] <- mean(sharpe\_wt[,i] > sharpe\_wt[,j])

count <- count + 1

}

}

}

sharpe\_wt\_t %>%

mutate(Occurence = round(Occurence,3)\*100) %>%

knitr::kable(caption = "Frequency of better Sharpe ratio (%)")

# Create weight change data frame

weights <- data.frame(us\_stock = seq(.45,.2, -0.0125),

intl\_stock =seq(.45,.2, -0.0125),

us\_bond = seq(.025, .3, .01375),

em\_hg = seq(0.0125, .15, .006875),

em\_hy = seq(0.0125, .15, .006875),

cmdty = seq(.05, 0, -0.0025))

# Change in ts objects

yr\_ts <- ts(df\_yr[,2:7], start = c(1999,1), frequency = 1)

wts\_ts <- ts(weights, start=c(1998,1), frequency = 1)

# Run portfolio rebalancing

no\_rebal <- Return.portfolio(yr\_ts,wt1)

rebal <- Return.portfolio(yr\_ts,wts\_ts)

# Convert into data frame

rebal\_yr <- data.frame(date = index(no\_rebal), no\_rebal = as.numeric(no\_rebal),

rebal = as.numeric(rebal))

# Graph

rebal\_yr %>%

gather(key, value, -date) %>%

group\_by(key) %>%

mutate(value = (cumprod(1+value)-1)\*100) %>%

ggplot(aes(date, value, color = key)) +

geom\_line() +

scale\_color\_manual("", labels = c("No rebalancing", "Rebalancing"),

values = c("black", "blue"))+

labs(x="",

y="Return(%)",

title = "Rebalancing or not?")+

theme(legend.position = "top")

1. We started experimenting with the cross-correlation function to see which lag had a higher or more significant correlation across the assets. But it became clear, quite quickly, that choosing a reasonable lag would take longer than the time we had allotted for this post. So we opted for the easy way out. Make a simplifying assumption! If anyone can point us to cross-correlation function studies that might apply to this scenario, please let us know.[**↩**](https://osm.netlify.app/post/rebalancing-history/#fnref1)
2. Please see this [**article**](http://www.business.unr.edu/faculty/liuc/files/BADM742/Jegadeesh_Titman_1993.pdf) for more detail.[**↩**](https://osm.netlify.app/post/rebalancing-history/#fnref2)
3. The Performance Analytics package in R doesn’t offer a rebalancing algorithm based on thresholds unless we missed it.[**↩**](https://osm.netlify.app/post/rebalancing-history/#fnref3)