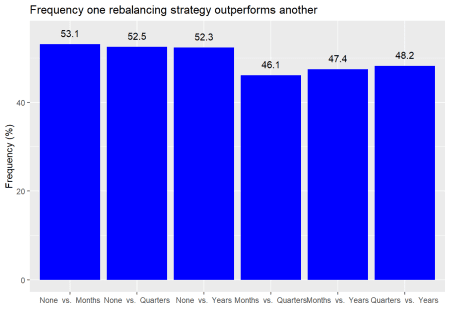
Back in the rebalancing saddle! ,we analyzed whether rebalancing over different periods would have any effect on mean or risk-adjusted returns for our three (equal, naive, and risky) portfolios. We found little evidence that returns were much different whether we rebalanced monthly, quarterly, yearly, 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.

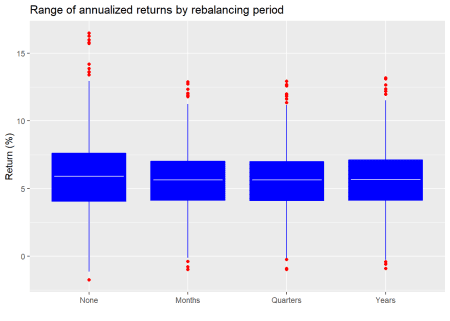
But this lack of evidence, doesn’t mean there’s evidence of a lack of better returns from rebalancing. We believe a broader test is required. To that end, we’ll run many simulations of those historical returns to identify the likelihood that rebalancing is worthwhile. Recall that since we’ve been using 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. 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.

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.



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.com/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 |

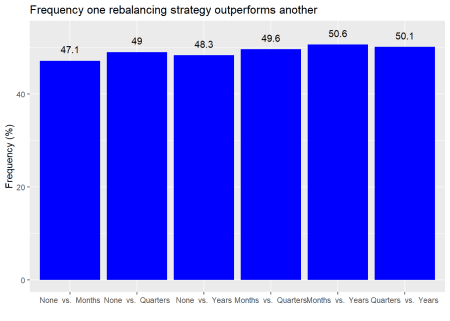
No rebalancing produces a slightly worse risk-adjusted return than rebalancing. But is the difference significant? Our guess is probably not. Performing a robust test would require some more complicated math that would be beyond the scope of this post to explain. Instead, we’ll look at the number of 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.



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.com/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.

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](http://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 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 sharpe\_wt[,j])

count <- count + 1

}

}

}

# Table

sharpe\_wt\_t %>%

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