Happy 2018 and welcome to our first reproducible finance post of the year! What better way to ring in a new beginning than pondering/calculating/visualizing returns distributions.

We ended 2017 by tackling skewness, and we will begin 2018 by tackling kurtosis.

Skewness is the degree to which returns are asymmetric around the mean. Since a normal distribution is symmetric around the mean, skewness can be taken as one measure of how returns are not distributed normally. Why does skewness matter? If portfolio returns are right, or positively, skewed, it implies numerous small negative returns and a few large positive returns. If portfolio returns are left, or negatively, skewed, it implies numerous small positive returns and few large negative returns. The phrase “large negative returns” should trigger Pavlovian sweating for investors, even if it’s preceded by a diminutive modifier like “just a few”. For a portfolio manager, a negatively skewed distribution of returns implies a portfolio at risk of rare but large losses. This makes us nervous and is a bit like saying, “I’m healthy, except for my occasional massive heart attack.”

Let’s get to it.

First, have a look at one equation for skewness:

Skew=n∑t=1(xi−¯¯¯x)3/n/(n∑t=1(xi−¯¯¯x)2/n)3/2Skew=∑t=1n(xi−x¯)3/n/(∑t=1n(xi−x¯)2/n)3/2

Skew has important substantive implications for risk, and is also a concept that lends itself to data visualization. In fact, I find the visualizations of skewness more illuminating than the numbers themselves (though the numbers are what matter in the end). In this section, we will cover how to calculate skewness using xts and tidyverse methods, how to calculate rolling skewness, and how to create several data visualizations as pedagogical aids. We will be working with our usual portfolio consisting of:

+ SPY (S&P500 fund) weighted 25%

+ EFA (a non-US equities fund) weighted 25%

+ IJS (a small-cap value fund) weighted 20%

+ EEM (an emerging-mkts fund) weighted 20%

+ AGG (a bond fund) weighted 10%

Building off that previous work, we will be working with two objects of portfolio returns:

+ portfolio\_returns\_xts\_rebalanced\_monthly (an xts of monthly returns)

+ portfolio\_returns\_tq\_rebalanced\_monthly (a tibble of monthly returns)

Let’s begin in the xts world and make use of the skewness() function from PerformanceAnalytics.

**library**(PerformanceAnalytics)

skew\_xts <- skewness(portfolio\_returns\_xts\_rebalanced\_monthly$returns)

skew\_xts

## [1] -0.1710568

Our portfolio is relatively balanced, and a slight negative skewness of -0.1710568 is unsurprising and unworrisome. However, that final number could be omitting important information and we will resist the temptation to stop there. For example, is that slight negative skew being caused by one very large negative monthly return? If so, what happened? Or is it caused by several medium-sized negative returns? What caused those? Were they consecutive? Are they seasonal? We need to investigate further.

Before doing so and having fun with data visualization, let’s explore the tidyverse methods and confirm consistent results.

We will make use of the same skewness() function, but because we are using a tibble, we use summarise() as well and call summarise(skew = skewness(returns). It’s not necessary, but we are also going to run this calculation by hand, the same as we have done with standard deviation. Feel free to delete the by-hand section from your code should this be ported to enterprise scripts, but keep in mind that there is a benefit to forcing ourselves and loved ones to write out equations: it emphasizes what those nice built-in functions are doing under the hood. If a client, customer or risk officer were ever to drill into our skewness calculations, it would be nice to have a super-firm grasp on the equation.

**library**(tidyverse)

**library**(tidyquant)

skew\_tidy <-

portfolio\_returns\_tq\_rebalanced\_monthly %>%

summarise(skew\_builtin = skewness(returns),

skew\_byhand =

(sum((returns - mean(returns))^3)/length(returns))/

((sum((returns - mean(returns))^2)/length(returns)))^(3/2)) %>%

select(skew\_builtin, skew\_byhand)

Let’s confirm that we have consistent calculations.

skew\_xts

## [1] -0.1710568

skew\_tidy$skew\_builtin

## [1] -0.1710568

skew\_tidy$skew\_byhand

## [1] -0.1710568

The results are consistent using xts and our tidyverse, by-hand methods. Again, though, that singular number -0.1710568 does not fully illuminate the riskiness or distribution of this portfolio. To dig deeper, let’s first visualize the density of returns with stat\_density from ggplot2.

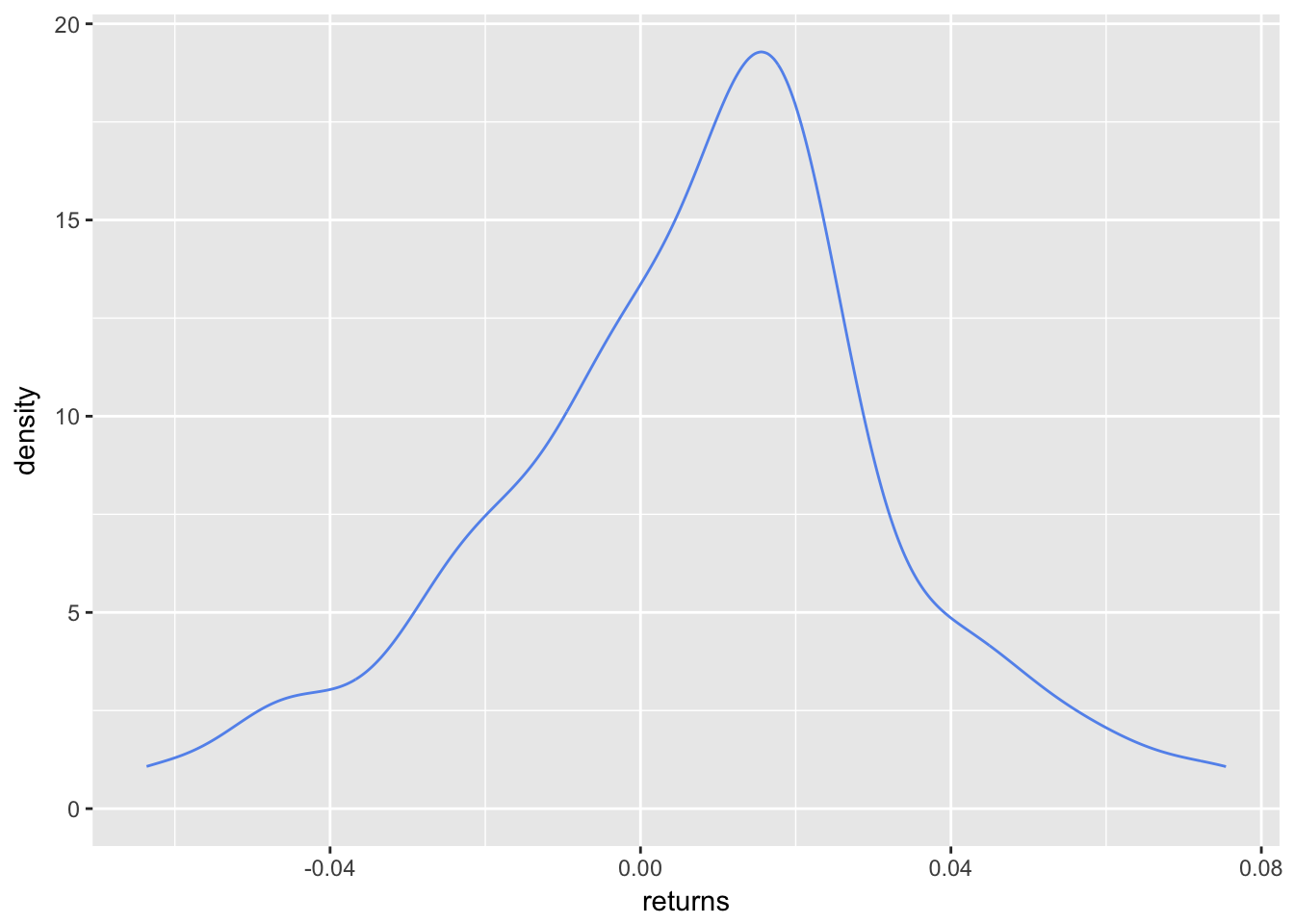
portfolio\_density\_plot <-

portfolio\_returns\_tq\_rebalanced\_monthly %>%

ggplot(aes(x = returns)) +

stat\_density(geom = "line", alpha = 1, colour = "cornflowerblue")

portfolio\_density\_plot

[](https://rviews.rstudio.com/post/2017-12-13-introduction-to-skewness_files/figure-html/unnamed-chunk-4-1.png)

The slight negative skew is a bit more evident here. It would be nice to shade the area that falls below some threshold again, and let’s go with the mean return. To do that, let’s create an object called shaded\_area using ggplot\_build(portfolio\_density\_plot)$data[[1]] %>% filter(x < mean(portfolio\_returns\_tq\_rebalanced\_monthly$returns)). That snippet will take our original ggplot object and create a new object filtered for x values less than mean return. Then we use geom\_area to add the shaded area to portfolio\_density\_plot.

shaded\_area\_data <-

ggplot\_build(portfolio\_density\_plot)$data[[1]] %>%

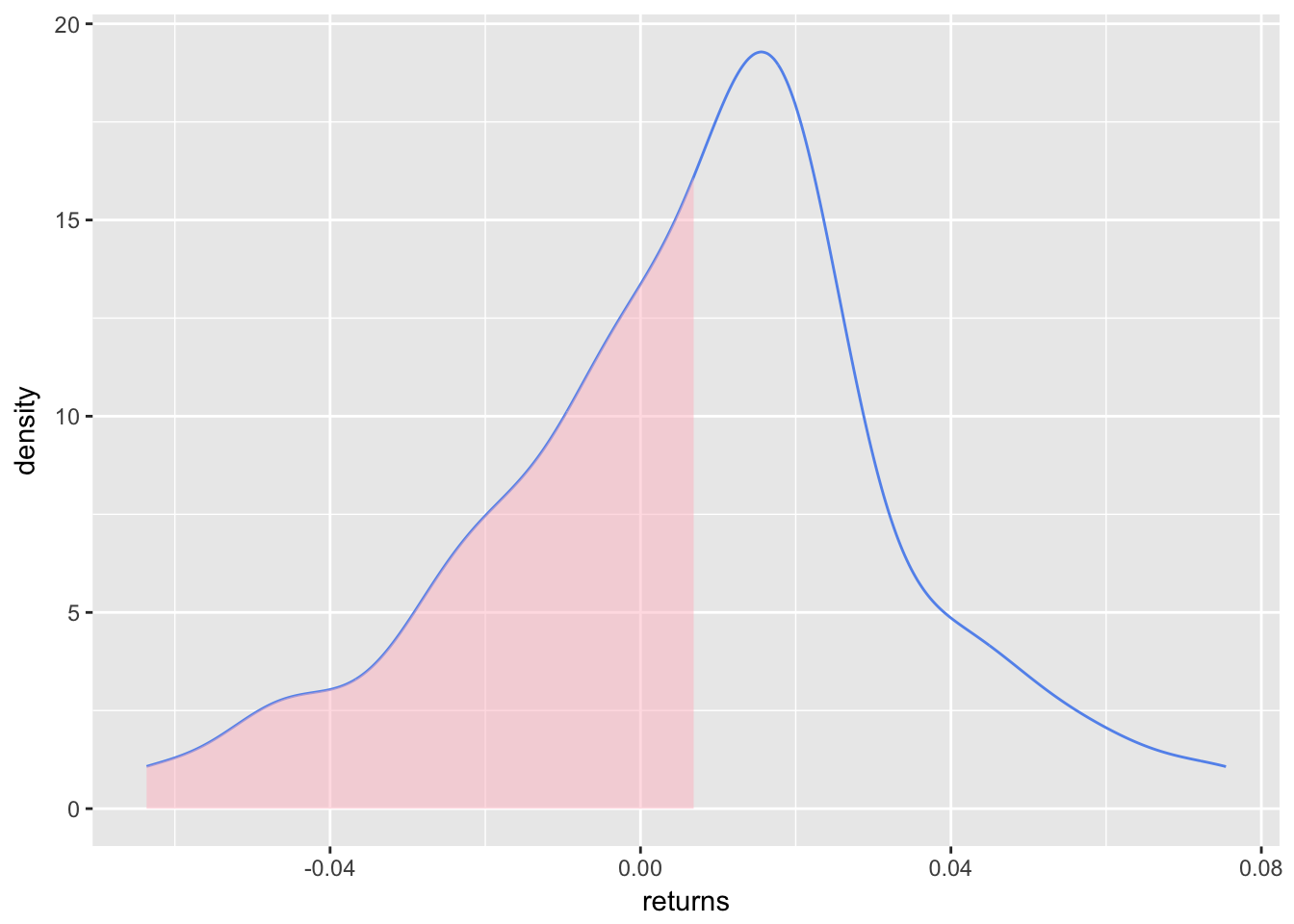
filter(x < mean(portfolio\_returns\_tq\_rebalanced\_monthly$returns))

portfolio\_density\_plot\_shaded <-

portfolio\_density\_plot +

geom\_area(data = shaded\_area\_data, aes(x = x, y = y), fill="pink", alpha = 0.5)

portfolio\_density\_plot\_shaded

[](https://rviews.rstudio.com/post/2017-12-13-introduction-to-skewness_files/figure-html/unnamed-chunk-5-1.png)

The shaded area highlights the mass of returns that fall below the mean. Let’s add a vertical line at the mean and median, and some explanatory labels. This will help to emphasize that negative skew indicates a mean less than the median.

First, create variables for mean and median so that we can add a vertical line.

median <- median(portfolio\_returns\_tq\_rebalanced\_monthly$returns)

mean <- mean(portfolio\_returns\_tq\_rebalanced\_monthly$returns)

We want the vertical lines to just touch the density plot so we once again use a call to ggplot\_build(portfolio\_density\_plot)$data[[1]].

median\_line\_data <-

ggplot\_build(portfolio\_density\_plot)$data[[1]] %>%

filter(x <= median)

Now we can start adding aesthetics to the latest iteration of our graph, which is stored in the object portfolio\_density\_plot\_shaded.

portfolio\_density\_plot\_shaded +

geom\_segment(aes(x = 0, y = 1.9, xend = -.045, yend = 1.9),

arrow = arrow(length = unit(0.5, "cm")), size = .05) +

annotate(geom = "text", x = -.02, y = .1, label = "returns <= mean",

fontface = "plain", alpha = .8, vjust = -1) +

geom\_segment(data = shaded\_area\_data, aes(x = mean, y = 0, xend = mean, yend = density),

color = "red", linetype = "dotted") +

annotate(geom = "text", x = mean, y = 5, label = "mean", color = "red",

fontface = "plain", angle = 90, alpha = .8, vjust = -1.75) +

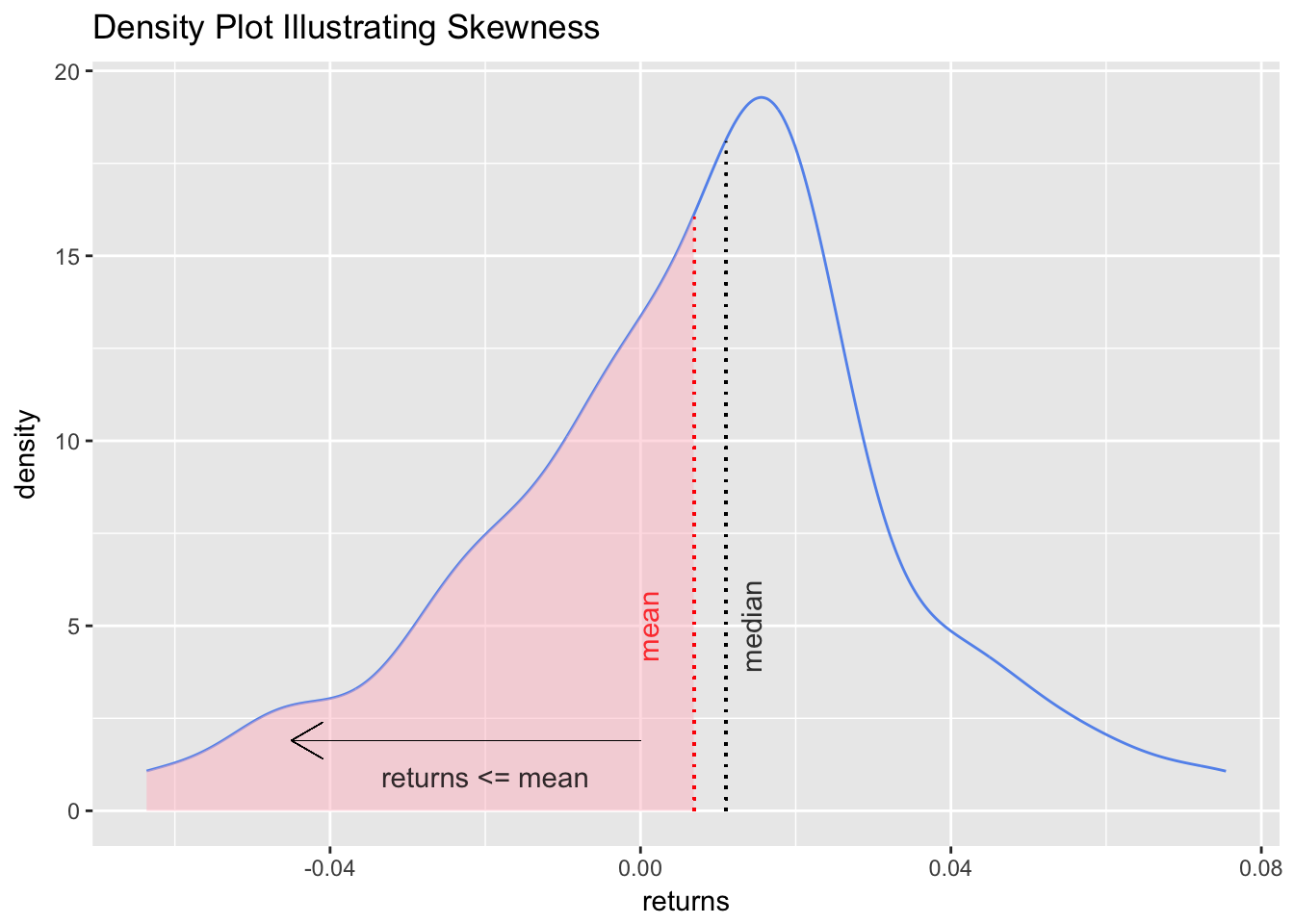
geom\_segment(data = median\_line\_data, aes(x = median, y = 0, xend = median, yend = density),

color = "black", linetype = "dotted") +

annotate(geom = "text", x = median, y = 5, label = "median",

fontface = "plain", angle = 90, alpha = .8, vjust = 1.75) +

ggtitle("Density Plot Illustrating Skewness")

[](https://rviews.rstudio.com/post/2017-12-13-introduction-to-skewness_files/figure-html/unnamed-chunk-8-1.png)

We added quite a bit to the chart, possibly too much, but it’s better to be over-inclusive now to test different variants. We can delete any of those features when using this chart later, or refer back to these lines of code should we ever want to reuse some of the aesthetics.

At this point, we have calculated the skewness of this portfolio throughout its history, and done so using three methods. We have also created an explanatory visualization.

Similar to the portfolio standard deviation, though, our work is not complete until we look at rolling skewness. Perhaps the first two years of the portfolio were positive skewed, and last two were negative skewed but the overall skewness is slightly negative. We would like to understand how the skewness has changed over time, and in different economic and market regimes. To do so, we calculate and visualize the rolling skewness over time.

In the xts world, calculating rolling skewness is almost identical to calculating rolling standard deviation, except we call the skewness() function instead of StdDev(). Since this is a rolling calculation, we need a window of time for each skewness; here, we will use a six-month window.

window <- 6

rolling\_skew\_xts <- na.omit(rollapply(portfolio\_returns\_xts\_rebalanced\_monthly, window,

**function**(x) skewness(x)))

Now we pop that xts object into highcharter for a visualization. Let’s make sure our y-axis range is large enough to capture the nature of the rolling skewness fluctuations by setting the range to between 3 and -3 with hc\_yAxis(..., max = 3, min = -3). I find that if we keep the range from 1 to -1, it makes most rolling skews look like a roller coaster.

**library**(highcharter)

highchart(type = "stock") %>%

hc\_title(text = "Rolling") %>%

hc\_add\_series(rolling\_skew\_xts, name = "Rolling skewness", color = "cornflowerblue") %>%

hc\_yAxis(title = list(text = "skewness"),

opposite = FALSE,

max = 3,

min = -3) %>%

hc\_navigator(enabled = FALSE) %>%

hc\_scrollbar(enabled = FALSE)

For completeness of methods, we can calculate rolling skewness in a tibble and then use ggplot.

We will make use of rollapply() from within tq\_mutate in tidyquant.

rolling\_skew\_tidy <-

portfolio\_returns\_tq\_rebalanced\_monthly %>%

tq\_mutate(select = returns,

mutate\_fun = rollapply,

width = window,

FUN = skewness,

col\_rename = "skew")

rolling\_skew\_tidy is ready for ggplot. ggplot is not purpose-built for time series plotting, but we can set aes(x = date, y = skew) to make the x-axis our date values.

**library**(scales)

theme\_update(plot.title = element\_text(hjust = 0.5))

rolling\_skew\_tidy %>%

ggplot(aes(x = date, y = skew)) +

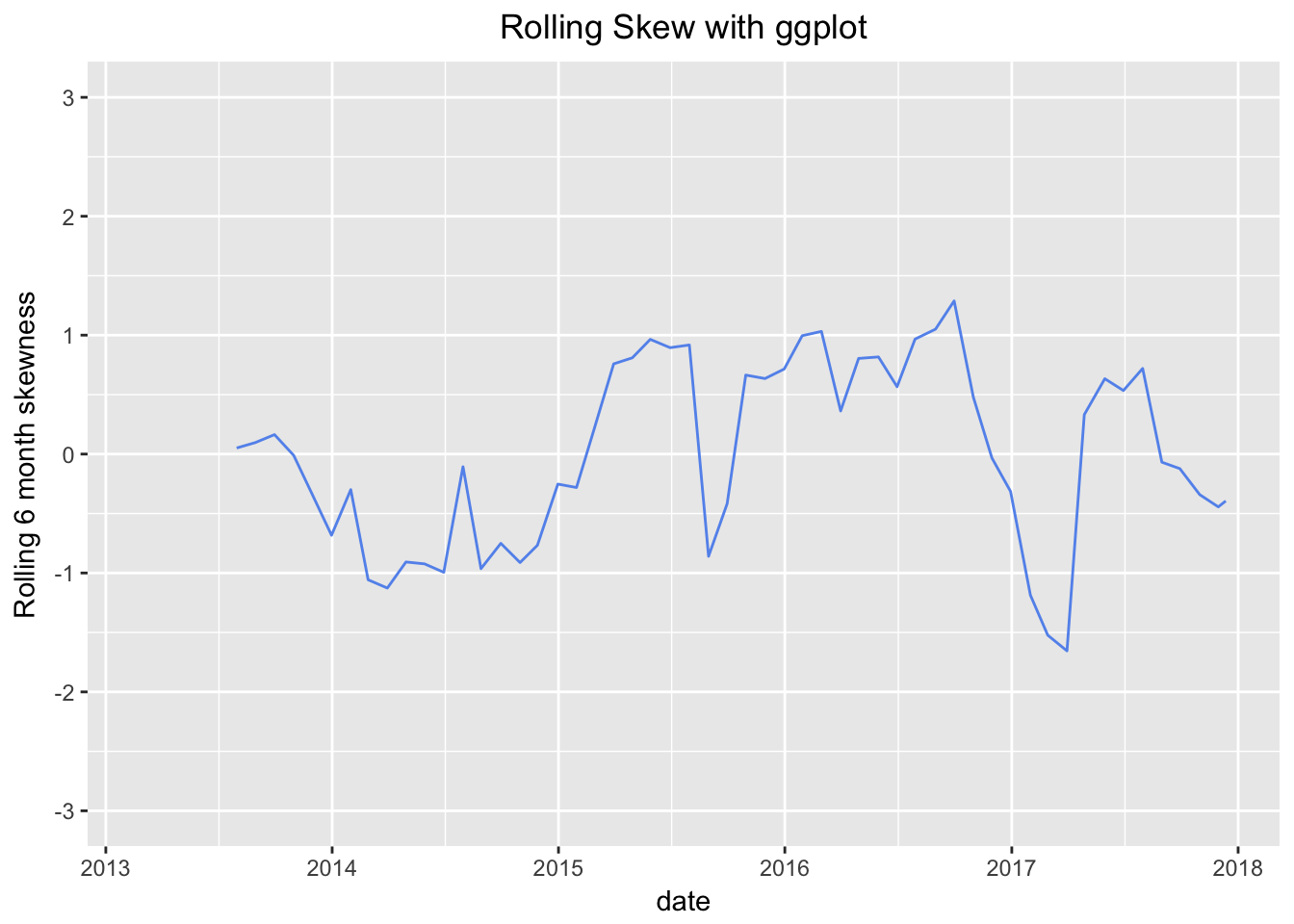
geom\_line(color = "cornflowerblue") +

ggtitle("Rolling Skew with ggplot") +

ylab(paste("Rolling", window, "month skewness", sep = " ")) +

scale\_y\_continuous(limits = c(-3, 3), breaks = pretty\_breaks(n = 8)) +

scale\_x\_date(breaks = pretty\_breaks(n = 8))

[](https://rviews.rstudio.com/post/2017-12-13-introduction-to-skewness_files/figure-html/unnamed-chunk-12-1.png)

The rolling charts are quite illuminating and show that the six-month-interval skewness has been positive for about half the lifetime of this portfolio. Today, the overall skewness is negative, but the rolling skewness in mid-2016 was positive and greater than 1. It took a huge plunge starting at the end of 2016, and the lowest reading was -1.65 in March of 2017, most likely caused by one or two very large negative returns when the market was worried about the US election. We can see those worries start to abate as the rolling skewness becomes more positive throughout 2017.

Kurtosis is a measure of the degree to which portfolio returns appear in the tails of our distribution. A normal distribution has a kurtosis of 3, which follows from the fact that a normal distribution does have some of its mass in its tails. A distribution with a kurtosis greater than 3 has more returns out in its tails than the normal, and one with kurtosis less than 3 has fewer returns in its tails than the normal. That matters to investors because more bad returns out in tails means that our portfolio might be at risk of a rare but huge downside. The terminology is a bit confusing. Negative kurtosis is considered less risky because it has fewer returns out in the tails. Negative == less risky? We’re not used to that in finance.

Kurtosis is often has the word ‘excess’ appended to its description, as in ‘negative excess kurtosis’ or ‘positive excess kurtosis’. That ‘excess’ is in comparison to a normal distribution kurtosis of 3. A distribution with negative excess kurtosis equal to -1 has an actual kurtosis of 2.

Enough with the faux investopedia entry, let’s get to the calculations, R code and visualizations.

Here’s the equation for excess kurtosis. Note that we subtract 3 at the end:

\[Kurtosis=\sum\_{t=1}^n (x\_i-\overline{x})^4/n \bigg/ (\sum\_{t=1}^n (x\_i-\overline{x})^2/n)^{2}-3 \]

By way of reminder, we will be working with our usual portfolio consisting of:

+ SPY (S&P500 fund) weighted 25%

+ EFA (a non-US equities fund) weighted 25%

+ IJS (a small-cap value fund) weighted 20%

+ EEM (an emerging-mkts fund) weighted 20%

+ AGG (a bond fund) weighted 10%

Before we can calculate kurtosis, we need to find portfolio monthly returns:

Code Chunks

Our five-asset portfolio will consist of the following securities and weights:

+ SPY (S&P500 fund) weighted 25%

+ EFA (a non-US equities fund) weighted 25%

+ IJS (a small-cap value fund) weighted 20%

+ EEM (an emerging-mkts fund) weighted 20%

+ AGG (a bond fund) weighted 10%

A brief interlude on portfolios: a portfolio is a weighted collection of assets (its etymology harkens back to the Latin for “to carry sheets of paper”, which I s’pose made its way to the world of investments because securities used to be sheets of paper). The theoretical reason that rational investors prefer a portfolio to a single asset is that a portfolio can offer a better risk/return trade-off due to low or negative covariance amongst portfolio components.

Back to the task at hand: transform a collection of daily ETF prices into an object of portfolio log returns.

Let’s load up our packages.

**library**(tidyverse)

**library**(tidyquant)

**library**(timetk)

First, we import daily prices for the five ETFs, using getSymbols to grab the data, map(~Ad(get(.))) to select adjusted prices only, and reduce(merge) to mash our five prices into one xts object.

*# The symbols vector holds our tickers.*

symbols <- c("SPY","EFA", "IJS", "EEM","AGG")

*# The prices object will hold our raw price data throughout this book.*

prices <-

getSymbols(symbols, src = 'yahoo', from = "2005-01-01",

auto.assign = TRUE, warnings = FALSE) %>%

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

reduce(merge) %>%

`colnames<-`(symbols)

Next, we convert those daily adjusted prices to monthly log returns using two methods. For the first method, we stay in the xts world.

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

asset\_returns\_xts <- na.omit(Return.calculate(prices\_monthly, method = "log"))

For the second method, we will head to the tidyverse/tidyquant world. We will convert from xts to tibble using a call to tk\_tbl(preserve\_index = TRUE, rename\_index = "date"). We will add a column for log returns using mutate(returns = (log(returns) - log(lag(returns)))).

*# Tidyverse method, to long, tidy format*

asset\_returns\_long <-

prices %>%

to.monthly(indexAt = "last", OHLC = FALSE) %>%

tk\_tbl(preserve\_index = TRUE, rename\_index = "date") %>%

gather(asset, returns, -date) %>%

group\_by(asset) %>%

mutate(returns = (log(returns) - log(lag(returns))))

Have a peek at both asset return objects.

head(asset\_returns\_xts)

## SPY EFA IJS EEM AGG

## 2005-02-28 0.020688126 0.037150948 0.02860922 0.09241734 -0.003721035

## 2005-03-31 -0.018461970 -0.026583514 -0.02388198 -0.08240681 -0.009790455

## 2005-04-29 -0.018913092 -0.016309073 -0.05255677 -0.01255390 0.017081857

## 2005-05-31 0.031716351 -0.008674664 0.05973603 0.03111818 0.008242118

## 2005-06-30 0.001514103 0.014225362 0.03840792 0.03892299 0.008724056

## 2005-07-29 0.037547542 0.029527397 0.05677107 0.07400825 -0.010408874

head(asset\_returns\_long)

## # A tibble: 6 x 3

## # Groups: asset [1]

## date asset returns

## <date> <chr> <dbl>

## 1 2005-01-31 SPY NA

## 2 2005-02-28 SPY 0.020688126

## 3 2005-03-31 SPY -0.018461970

## 4 2005-04-29 SPY -0.018913092

## 5 2005-05-31 SPY 0.031716351

## 6 2005-06-30 SPY 0.001514103

Do we notice any differences?

First, have a look at the left most part of asset\_returns\_xts, where the date is stored. The asset\_returns\_xts has a date index, not a column. It is accessed via index(asset\_returns\_xts). asset\_returns\_long has a column called “date”, accessed via the $date convention, i.e., asset\_returns\_long$date.

Second, notice the first date observation for January of 2005. asset\_returns\_long contains NA, and asset\_returns\_xts excludes the observation completely. Does it matter? It depends. In a few weeks when we get to the Sortino Ratio, we will see that it can matter quite a bit.

Third, asset\_returns\_xts is in wide format, which in this case means there is a column for each of our assets. This is the format that xts likes, and it’s the format that is easier to read as a human. However, asset\_returns\_long is in long, tidy format so that each variable has its own column. It’s a bit harder to read as human, but the tidyverse wants data in this format.

Now on to constructing a portfolio and calculating returns. To turn these five ETFs into a portfolio we need to assign them weights. Let’s first create a weights vector.

w <- c(0.25, 0.25, 0.20, 0.20, 0.10)

Before we use the weights in our calculations, we will run a quick sanity check in the next code chunk. This might not be necessary with five assets as we have today, but it is good practice because if we had 50 assets, it could save us a lot of grief to catch a mistake early.

*# Make sure the weights line up with assets.*

asset\_weights\_sanity\_check <- tibble(w, symbols)

asset\_weights\_sanity\_check

## # A tibble: 5 x 2

## w symbols

## <dbl> <chr>

## 1 0.25 SPY

## 2 0.25 EFA

## 3 0.20 IJS

## 4 0.20 EEM

## 5 0.10 AGG

Make sure that tibble match up with the portfolio we want to create.

Finally, make sure the weights sum to 100%, or 1. Again, we can eyeball this with five assets, but with 50 assets it would be easier to run the sanity check.

sum(asset\_weights\_sanity\_check$w)

## [1] 1

They sum to 1. Good to go, and on to portfolio returns.

We will start with the textbook equation for the return of a multi-asset portfolio which is:Returnportfolio=W1∗Returnasset1 + W2∗Returnasset2 + W3∗Returnasset3 + W4∗Returnasset4 + W5∗Returnasset5Returnportfolio=W1∗Returnasset1 + W2∗Returnasset2 + W3∗Returnasset3 + W4∗Returnasset4 + W5∗Returnasset5Here’s the LaTeX code for that equation.

*# $$Return\_{portfolio} = W\_{1}\*Return\_{asset1}~+~W\_{2}\*Return\_{asset2}~+~W\_{3}\*Return\_{asset3}~+~W\_{4}\*Return\_{asset4}~+~W\_{5}\*Return\_{asset5}$$*

We ground through the LaTeX; now let’s grind through the R calculation by hand instead of using built-in functions.

First, assign each weight from our w vector to a variable.

Next, assign each asset return stored in asset\_returns\_xts to a variable.

Last, we insert those new variables into the equation.

w\_1 <- w[1]

w\_2 <- w[2]

w\_3 <- w[3]

w\_4 <- w[4]

w\_5 <- w[5]

asset1 <- asset\_returns\_xts[,1]

asset2 <- asset\_returns\_xts[,2]

asset3 <- asset\_returns\_xts[,3]

asset4 <- asset\_returns\_xts[,4]

asset5 <- asset\_returns\_xts[,5]

portfolio\_returns\_byhand <-

(w\_1 \* asset1) +

(w\_2 \* asset2) +

(w\_3 \* asset3) +

(w\_4 \* asset4) +

(w\_5 \* asset5)

names(portfolio\_returns\_byhand) <- "returns"

Our first portfolio returns calculation is now complete and stored as portfolio\_returns\_byhand. From a substantive perspective, we are finished and could head to visualization.

We want to cover more methods, though, so let’s head to to the xts world and the PerformanceAnalytics package. We didn’t explicitly load that package in the setup, because tidyquant imports it for us.

We will use theReturn.portfolio function, which requires two arguments for a portfolio, an xts object of asset returns, and a vector of weights. We have those at hand: asset\_returns\_xts and w. It’s not necessary, but we will set rebalance\_on = "months" so we can confirm it matches our by-hand calculations. Remember, in the by-hand equation, the portfolio weights are fixed, meaning they never change on a month-to-month basis. That is equivalent to re-balancing every month, which in practice would be quite rare.

portfolio\_returns\_xts\_rebalanced\_monthly <-

Return.portfolio(asset\_returns\_xts, weights = w, rebalance\_on = "months") %>%

`colnames<-`("returns")

Next let’s change to a more realistic annual re-balancing and set rebalance\_on = "years". This will change our results so that they no longer match our by-hand calculation, which effectively re-balanced every month (since we hard-coded asset weights to be the same each month).

portfolio\_returns\_xts\_rebalanced\_yearly <-

Return.portfolio(asset\_returns\_xts, weights = w, rebalance\_on = "years") %>%

`colnames<-`("returns")

We can take a peek at our three portfolio objects and see how the annual re-balance made a small but important difference.

head(portfolio\_returns\_byhand)

## returns

## 2005-02-28 0.03829298

## 2005-03-31 -0.03349817

## 2005-04-29 -0.02011949

## 2005-05-31 0.02475548

## 2005-06-30 0.02027345

## 2005-07-29 0.04188371

head(portfolio\_returns\_xts\_rebalanced\_monthly)

## returns

## 2005-02-28 0.03829298

## 2005-03-31 -0.03349817

## 2005-04-29 -0.02011949

## 2005-05-31 0.02475548

## 2005-06-30 0.02027345

## 2005-07-29 0.04188371

head(portfolio\_returns\_xts\_rebalanced\_yearly)

## returns

## 2005-02-28 0.03829298

## 2005-03-31 -0.03418759

## 2005-04-29 -0.02018237

## 2005-05-31 0.02441794

## 2005-06-30 0.02032339

## 2005-07-29 0.04228070

Do you notice where the annual re-balancing starts to show a difference from monthly re-balancing?

As before, we could stop here and have accomplished our substantive task (twice already - by hand and using the built-in function from PerformanceAnalytics), but we want to explore alternate methods in the world of tidyverse/tidyquant. We will use our long, tidy-formatted asset\_returns\_long and convert to portfolio returns using the tq\_portfolio function from tidyquant.

The tq\_portfolio function takes a tibble and then asks for an assets column to group by, a returns column to find return data, and a weights column. It’s a wrapper for Return.portfolio, and thus also accepts the argument rebalance\_on = "months". Since we are re-balancing by months, we should again get a portfolio returns object that matches our two existing objects portfolio\_returns\_byhand and portfolio\_returns\_xts\_rebalanced\_monthly.

portfolio\_returns\_tq\_rebalanced\_monthly <-

asset\_returns\_long %>%

tq\_portfolio(assets\_col = asset,

returns\_col = returns,

weights = w,

col\_rename = "returns",

rebalance\_on = "months")

If we want to re-balance annually, it’s the same code as above, except we set rebalance\_on = "years".

portfolio\_returns\_tq\_rebalanced\_yearly <-

asset\_returns\_long %>%

tq\_portfolio(assets\_col = asset,

returns\_col = returns,

weights = w,

col\_rename = "returns",

rebalance\_on = "years")

We now have two more portfolio returns objects and they are both tidy tibbles. Let’s take a quick look and compare how a tidy tibble of portfolio returns compares to an xts object of portfolio returns.

head(portfolio\_returns\_tq\_rebalanced\_yearly)

## # A tibble: 6 x 2

## date returns

## <date> <dbl>

## 1 2005-01-31 0.00000000

## 2 2005-02-28 0.03829298

## 3 2005-03-31 -0.03418759

## 4 2005-04-29 -0.02018237

## 5 2005-05-31 0.02441794

## 6 2005-06-30 0.02032339

head(portfolio\_returns\_xts\_rebalanced\_yearly)

## returns

## 2005-02-28 0.03829298

## 2005-03-31 -0.03418759

## 2005-04-29 -0.02018237

## 2005-05-31 0.02441794

## 2005-06-30 0.02032339

## 2005-07-29 0.04228070

Building off that previous work, we will be working with two objects of portfolio returns:

+ portfolio\_returns\_xts\_rebalanced\_monthly (an xts of monthly returns)

+ portfolio\_returns\_tq\_rebalanced\_monthly (a tibble of monthly returns)

Now we are going to test our past self’s work on skewness, and reuse that code flow to expedite the kurtosis work. The logic will remain the same, but we will call different built-in functions and different by-hand calculations.

For the xts world, we use the kurtosis() function instead of the skewness() function.

kurt\_xts <- kurtosis(portfolio\_returns\_xts\_rebalanced\_monthly$returns)

kurt\_xts

## [1] 0.5267736

For tidy, we have the same piped flow and use the formula for kurtosis for our by-hand calculations. Our by-hand result is labeled with kurt\_byhand, and involves quite a few parentheticals to map it back to the kurtosis equation above.

kurt\_tidy <-

portfolio\_returns\_tq\_rebalanced\_monthly %>%

summarise(

kurt\_builtin = kurtosis(returns),

kurt\_byhand =

((sum((returns - mean(returns))^4)/length(returns))/

((sum((returns - mean(returns))^2)/length(returns))^2)) - 3) %>%

select(kurt\_builtin, kurt\_byhand)

Let’s confirm that we have consistent calculations.

kurt\_xts

## [1] 0.5267736

kurt\_tidy$kurt\_builtin

## [1] 0.5267736

kurt\_tidy$kurt\_byhand

## [1] 0.5267736

We have consistent results from xts and the tidy built-in/by-hand worlds, and we were able to reuse our code from above to shorten the development time here. dd

Let’s do the same with the visualizations and head straight for a density plot, starting with the same portfolio\_density\_plot.

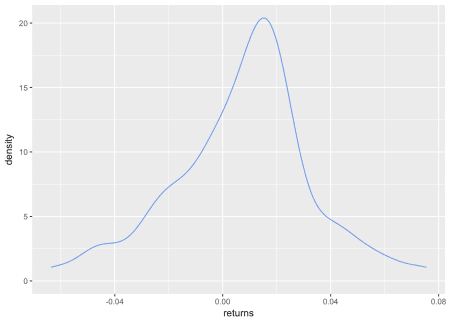
portfolio\_density\_plot <-

portfolio\_returns\_tq\_rebalanced\_monthly %>%

ggplot(aes(x = returns)) +

stat\_density(geom = "line", alpha = 1, colour = "cornflowerblue")

portfolio\_density\_plot



We are interested in *both* tails for kurtosis, so let’s shade at 2 standard deviations above and below the mean return (for our skewness work, we only shaded the negative tail).

mean <- mean(portfolio\_returns\_tq\_rebalanced\_monthly$returns)

sd\_pos <- mean + (2 \* sd(portfolio\_returns\_tq\_rebalanced\_monthly$returns))

sd\_neg <- mean - (2 \* sd(portfolio\_returns\_tq\_rebalanced\_monthly$returns))

sd\_pos\_shaded\_area <-

ggplot\_build(portfolio\_density\_plot)$data[[1]] %>%

filter(x > sd\_pos )

sd\_neg\_shaded\_area <-

ggplot\_build(portfolio\_density\_plot)$data[[1]] %>%

filter(x < sd\_neg)

portfolio\_density\_plot <-

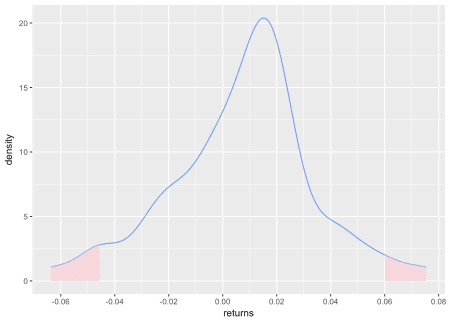
portfolio\_density\_plot +

geom\_area(data = sd\_pos\_shaded\_area, aes(x = x, y = y), fill="pink", alpha = 0.5) +

geom\_area(data = sd\_neg\_shaded\_area, aes(x = x, y = y), fill="pink", alpha = 0.5) +

scale\_x\_continuous(breaks = scales::pretty\_breaks(n = 10))

portfolio\_density\_plot



That density chart is a good look at the mass in both tails, where we have defined ‘tail’ as being two standard deviations away from the mean. We can add a line for the mean, as did in our skewness visualization, with the following.

mean <- mean(portfolio\_returns\_tq\_rebalanced\_monthly$returns)

mean\_line\_data <-

ggplot\_build(portfolio\_density\_plot)$data[[1]] %>%

filter(x <= mean)

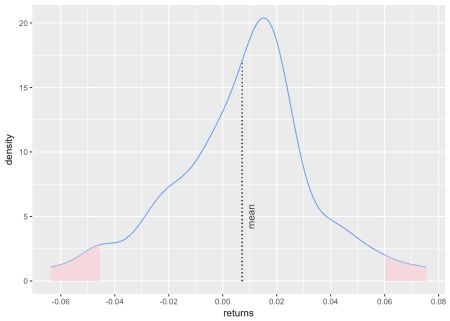
portfolio\_density\_plot +

geom\_segment(data = mean\_line\_data, aes(x = mean, y = 0, xend = mean, yend = density),

color = "black", linetype = "dotted") +

annotate(geom = "text", x = mean, y = 5, label = "mean",

fontface = "plain", angle = 90, alpha = .8, vjust = 1.75)



Finally, we can calculate and chart the rolling kurtosis with the same logic as we did for skewness. The only difference is that here we call fun = kurtosis instead of fun = skewness.

window <- 6

rolling\_kurt\_xts <- na.omit(apply.rolling(portfolio\_returns\_xts\_rebalanced\_monthly, window,

fun = kurtosis))

Now we pop that xts object into highcharter for a visualization.

highchart(type = "stock") %>%

hc\_title(text = "Rolling Kurt") %>%

hc\_add\_series(rolling\_kurt\_xts, name = "Rolling kurtosis", color = "cornflowerblue") %>%

hc\_yAxis(title = list(text = "kurtosis"),

opposite = FALSE,

max = .03,

min = -.03) %>%

hc\_navigator(enabled = FALSE) %>%

hc\_scrollbar(enabled = FALSE)

We didn’t cover this before, but what if we wanted to use ggplot for the rolling kurtosis? We could convert that xts object to a tibble with tk\_tbl() from the timetk package, and then pipe straight to ggplot.

rolling\_kurt\_xts %>%

tk\_tbl(preserve\_index = TRUE, rename\_index = "date") %>%

rename(rolling\_kurtosis = calcs) %>%

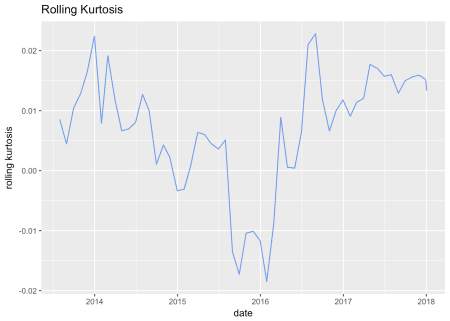
ggplot(aes(x = date, y = rolling\_kurtosis)) +

geom\_line(color = "cornflowerblue") +

xlab("date") +

ylab("rolling kurtosis") +

ggtitle("Rolling Kurtosis")



Interestingly, this portfolio has displayed slight positive rolling excess kurtosis for most of its life, except during the last half of 2015 through early 2016.

That’s all for today. Our work on kurtosis was made a lot more efficient by our work on skewness – so let’s thank our 2017 selves for constructing a reproducible and reusable code flow! See you next time.