Skew Post Chapter

In the following chapter we will calculate and visualize skewness and kurtosis, both of which are important indicators about the distribution of portfolio returns. Let's get to skewness first

Skewness

Skewness is the degree to which returns are asymetric around the mean. Since a normal distribtuion 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 diminuitve modifier like "just a few". For a portfolio manager, or any investor, a negatively skewed distribution of returns implies a portolio 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.

Here's the equation for skew:

$$Skew = \sum_{t=1}^{n}(x_i - \overline{x})^3/n \Big/ (\sum_{t=1}^{n}(x_i - \overline{x})^2/n)^{3/2}$$

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 also think about skewness from a comparative perspective and review how to test the skewness of different portfolios or assets.

We will be working with two of our core data objects:

Let's begin in the `xts` world and make use of the `skewness()` function from `performanceAnalytics`.

```
skew_xts <-  skewness(portfolio_returns_xts_rebalanced_monthly$returns)

skew_xts
```

```
## [1] -0.1577161
```

Our portfolio is relatively balanced and a slight negative skewness of -0.1577161 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 montly return? If so, what happened? Or was several medium sized negative returns? What caused those? Were they consecutive? Are they seasonal? The skewness alerts us that there is something to investigate but we need to dig deeper.

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, same as we have done with standard deviation. Feel free to delete the by-hand 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 calculaations, it would be nice to have a super firm grasp on the equation.

```
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] -0.1577161
```

```
skew_tidy$skew_builtin
```

```
## [1] -0.1461588
```

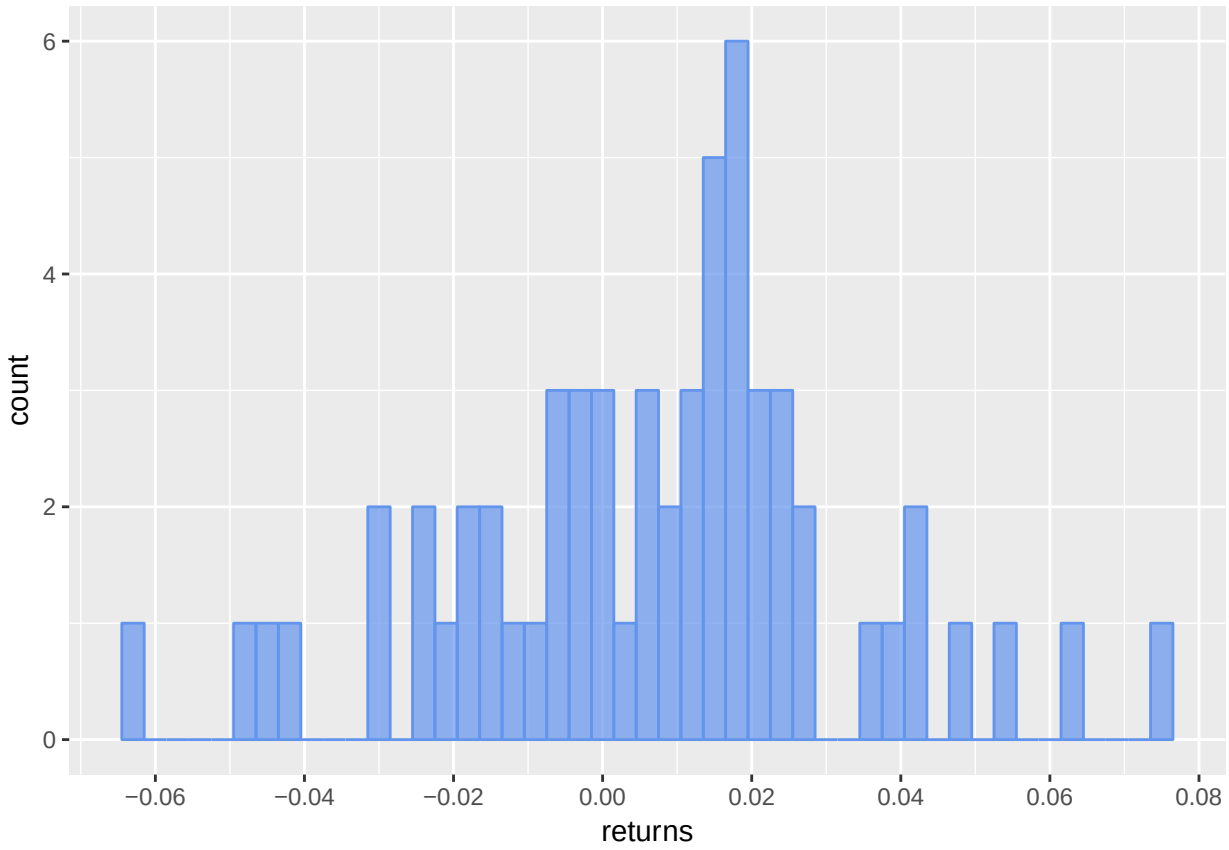
```
skew_tidy$skew_byhand
```

```
## [1] -0.1461588
```

The results are consistent using XTS and our tidyverse, by-hand methods. Again, though, that singular number -0.1577161 does not fully illuminate the riskiness or distribution of this portfolio.

We've already visualized the returns in a previous post and here they are again:

```
library(scales)
portfolio_returns_tq_rebalanced_monthly %>%
ggplot(aes(x = returns)) +
geom_histogram(alpha = .7,
               binwidth = .003,
               fill = "cornflowerblue",
               color = "cornflowerblue") +
scale_x_continuous(breaks = pretty_breaks(n = 10))
```



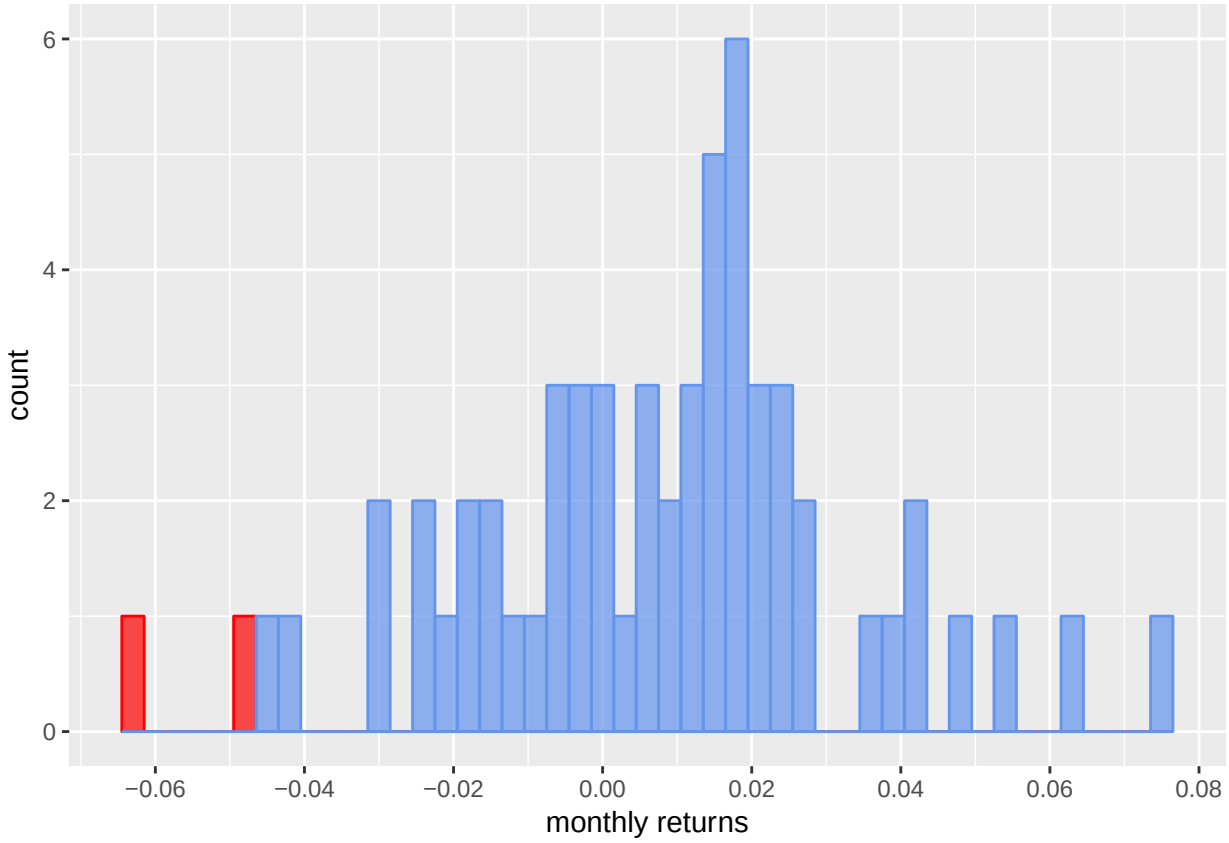
There seems to be one highly negative return (worse than -.06) and several between -.04 and -.08, plus a cluster of negative returns around -.02. These are small negative monthly returns, not cause for panic but worth investigating and we know that they resulted in an overall negative skewness. From the eyeball test, that cluster around -.02 seems to be the main driver.

Let's get more rigorous about which returns we want to highlight and investigate. For example, perhaps when our team thinks about skewness, we want to focus on monthly returns that fall 2 standard deviations below the mean, or maybe we focus on returns that small outside a threshold, less than -.03 and greater than +.03. `dplyr` and `ggplot` offer a nice way to visualize these.

In the case of highlighting returns that fall a certain number of standard deviations away, we will create three new columns: one for returns below a threshhold, one for returns above a threshhold, and one for returns within the two threshholds. For example, this code will create a column for each monthly return that is two standard deviations below the mean: `hist_col_red = ifelse(returns < (mean(returns) - 2*sd(returns)), returns, NA)`. I labeled the new column `hist_col_red` because we will shade these red to connotes that these are very negative returns.

```
portfolio_returns_tq_rebalanced_monthly %>%
  mutate(hist_col_red =
           ifelse(returns < (mean(returns) - 2*sd(returns)),
                  returns, NA),
         returns =
           ifelse(returns > (mean(returns) - 2*sd(returns)),
                  returns, NA)) %>%
ggplot() +
geom_histogram(aes(x = hist_col_red),
               alpha = .7,
               binwidth = .003,
```

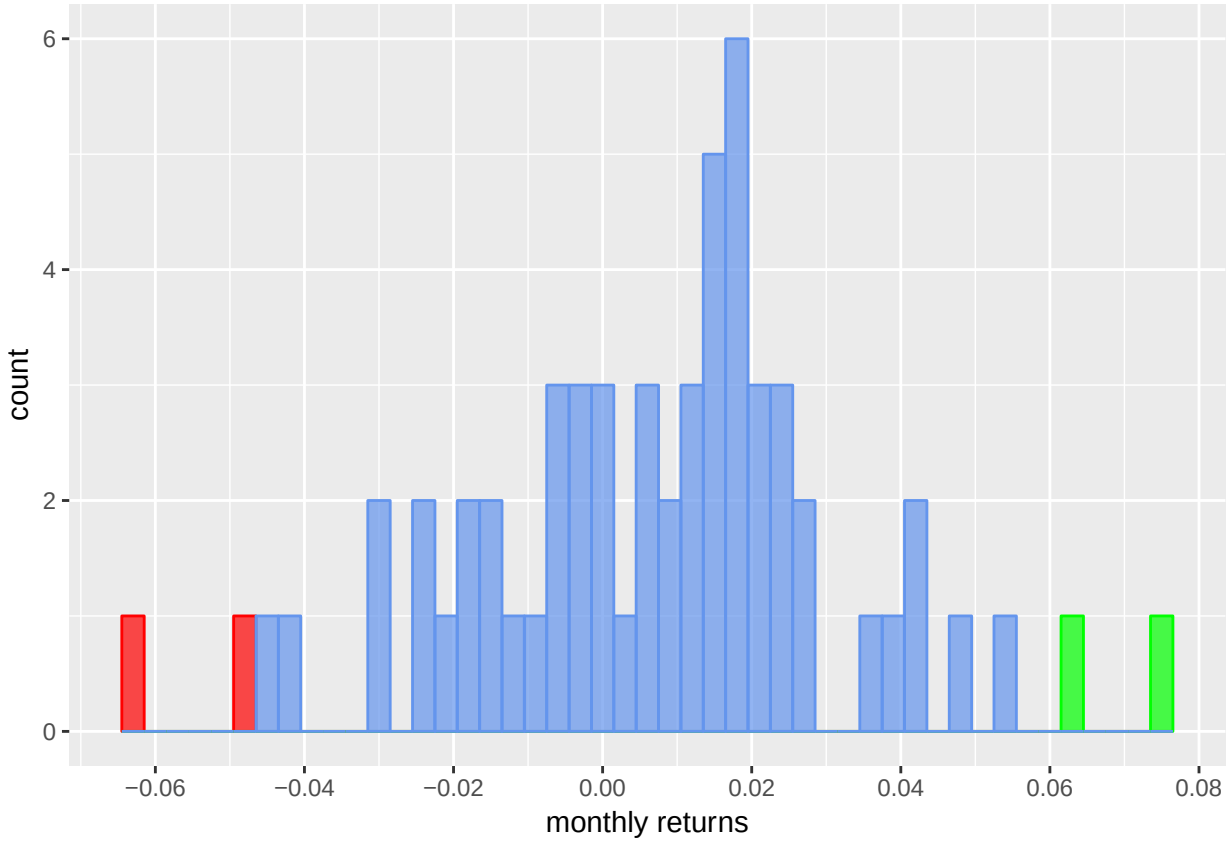
```
               fill = "red",
               color = "red") +
  geom_histogram(aes(x = returns),
               alpha = .7,
               binwidth = .003,
               fill = "cornflowerblue",
               color = "cornflowerblue") +
scale_x_continuous(breaks = pretty_breaks(n = 10)) +
xlab("monthly returns")
```



Now that negative skew makes a bit more sense - there are two obesrvations that more than two standard deviations away from the mean. Let's run the same aesthetic for positive returns.

```
portfolio_returns_tq_rebalanced_monthly %>%
  mutate(hist_col_red =
           ifelse(returns < (mean(returns) - 2*sd(returns)),
                  returns, NA),
         hist_col_green =
           ifelse(returns > (mean(returns) + 2*sd(returns)),
                  returns, NA),
         hist_col_blue =
           ifelse(returns > (mean(returns) - 2*sd(returns)) &
                  returns < (mean(returns) + 2*sd(returns)),
                  returns, NA)) %>%
  ggplot() +

  geom_histogram(aes(x = hist_col_red),
```

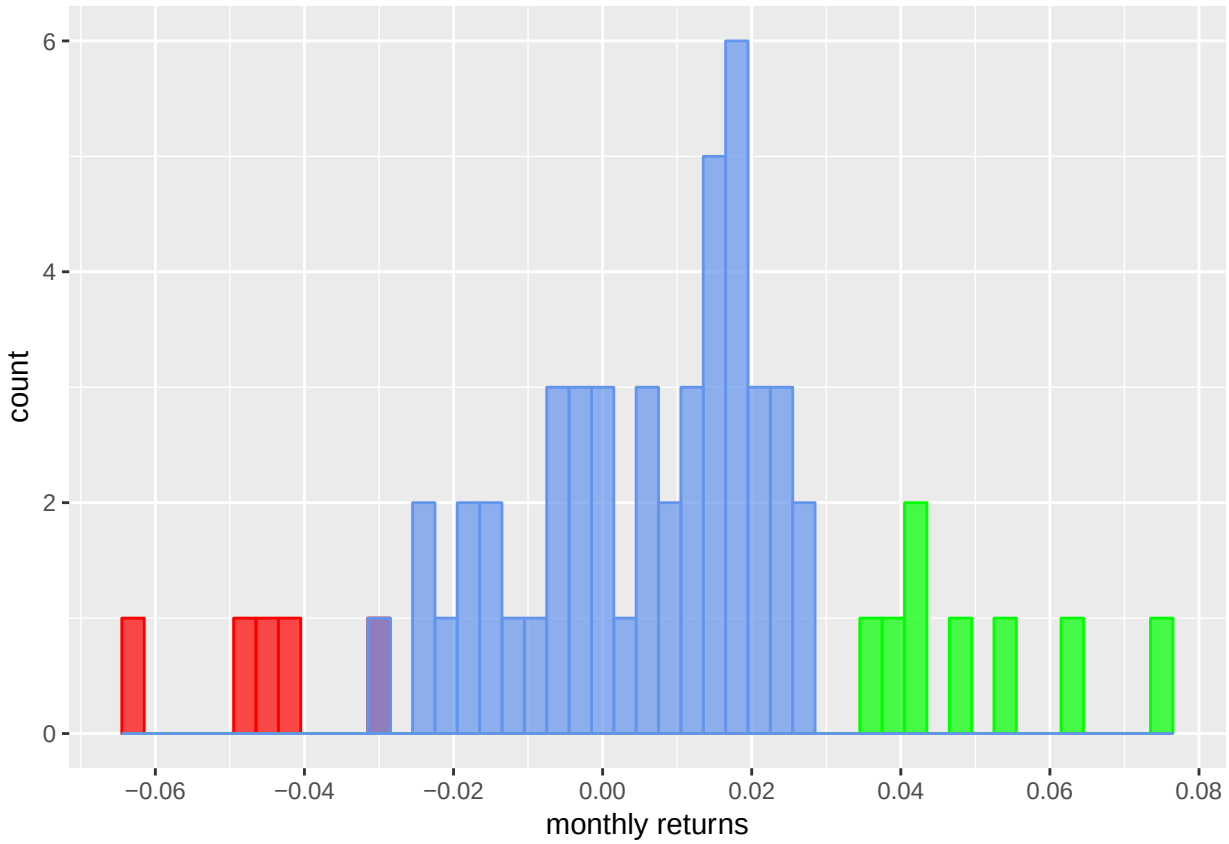
```
                alpha = .7,
                binwidth = .003,
                fill = "red",
                color = "red") +

  geom_histogram(aes(x = hist_col_green),
                alpha = .7,
                binwidth = .003,
                fill = "green",
                color = "green") +

  geom_histogram(aes(x = hist_col_blue),
                alpha = .7,
                binwidth = .003,
                fill = "cornflowerblue",
                color = "cornflowerblue") +

scale_x_continuous(breaks = pretty_breaks(n = 10)) +
xlab("monthly returns")
```



If we want to use a hard coded threshhold instead of a number of standard deviations, the code is very similar. Let's use a threshhol of -/+ .03.

```
portfolio_returns_tq_rebalanced_monthly %>%
  mutate(hist_col_red =
           ifelse(returns < -.03,
                  returns, NA),
```

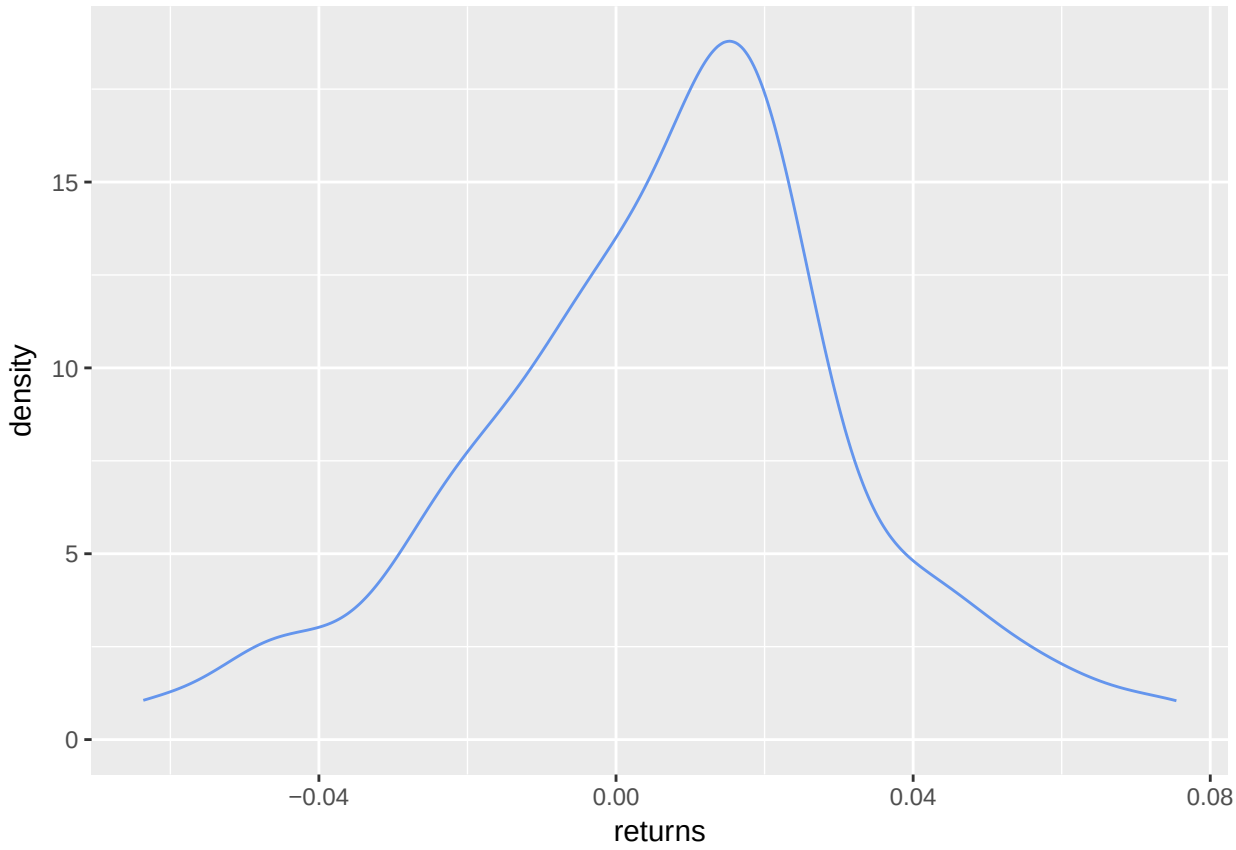
```
        hist_col_green =
          ifelse(returns > .03,
                 returns, NA),
        hist_col_blue =
          ifelse(returns > -.03 &
                 returns < .03,
                 returns, NA)) %>%
  ggplot() +

  geom_histogram(aes(x = hist_col_red),
                 alpha = .7,
                 binwidth = .003,
                 fill = "red",
                 color = "red") +

  geom_histogram(aes(x = hist_col_green),
                 alpha = .7,
                 binwidth = .003,
                 fill = "green",
                 color = "green") +

  geom_histogram(aes(x = hist_col_blue),
                 alpha = .7,
                 binwidth = .003,
                 fill = "cornflowerblue",
                 color = "cornflowerblue") +

scale_x_continuous(breaks = pretty_breaks(n = 10)) +
xlab("monthly returns")
```



There are more and higher green bins than red bins, meaning more positive returns above the threshhold. This indicates that the negative skewness is not being driven by the large negative returns, but rather by that strong cluster around -.02 and -.01.

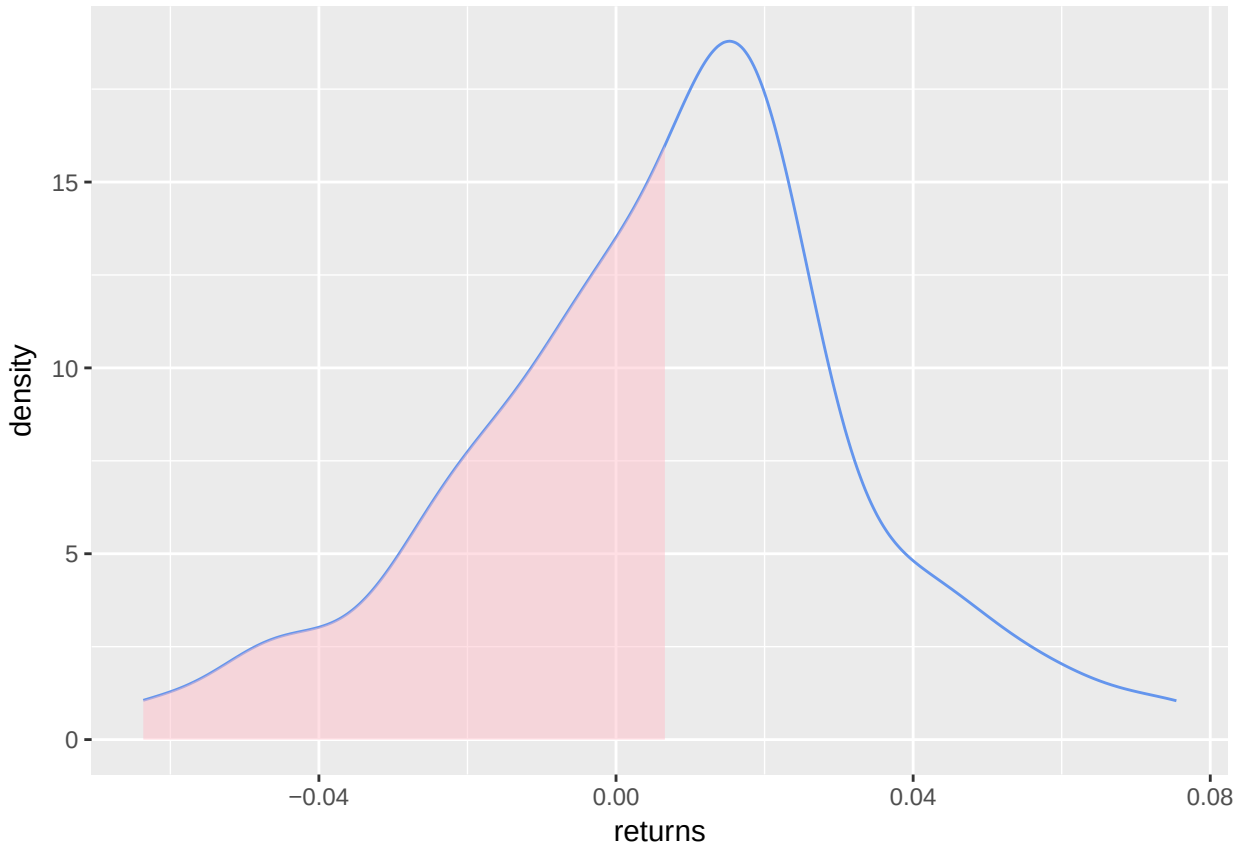
Those histograms help to see what is driving the skewness but skewness is traditionally visualized with a density plot so we will head to the `stat_density` call in `ggplot`.

```
portfolio_density_plot <-
  portfolio_returns_tq_rebalanced_monthly %>%
  ggplot(aes(x = returns)) +
  stat_density(geom = "line", alpha = 1, colour = "cornflowerblue")

portfolio_density_plot
```



The slight negative skew is a bit more evident here. It would be nice to shade the area that falls below some threshhold 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
```



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)
```

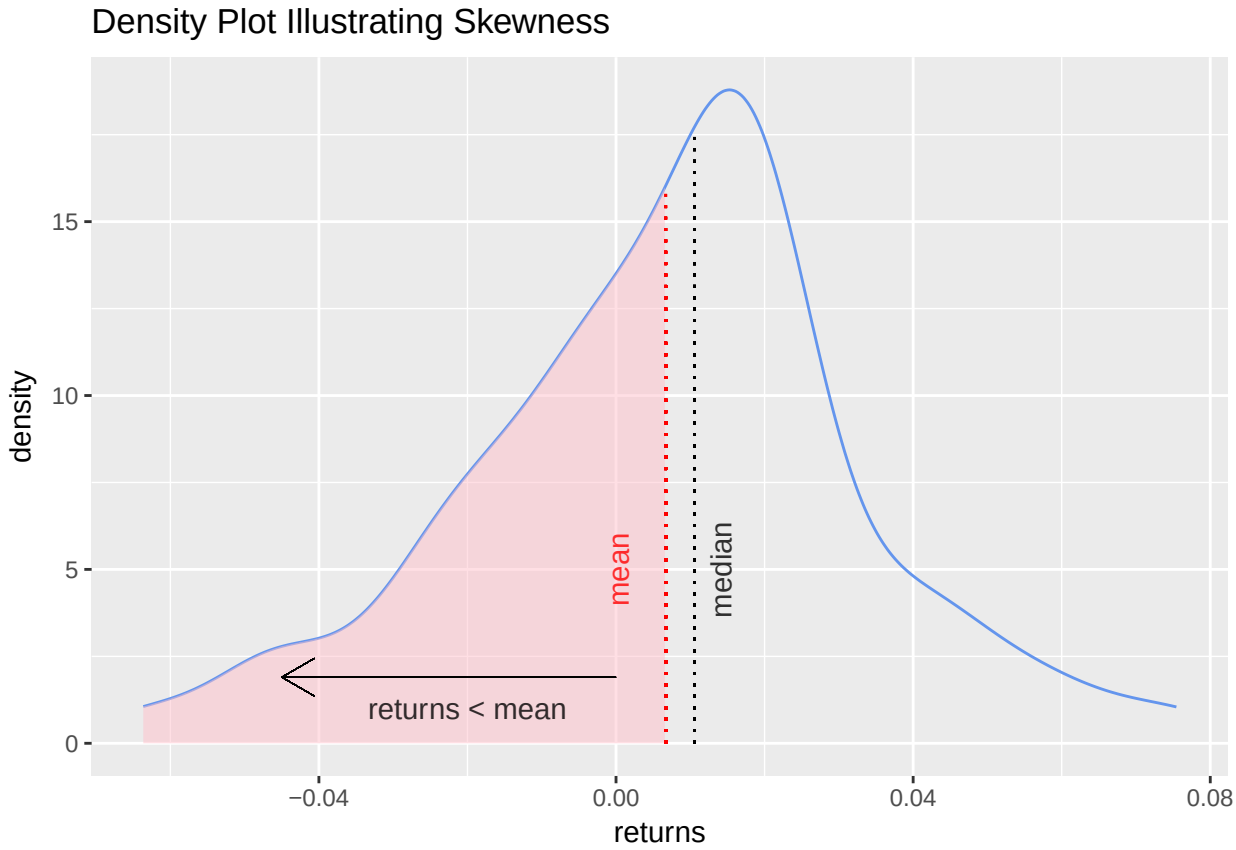
We want the vertical lines to just touch the density plot so we once again use a call to `ggplot_build(portfolio_density_plot)`

```
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")
```



We added quite a bit to the chart, possibly too much, but it's better to be overinclusive 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 some nice explanatory visualizations. But, thus far, our work has been focused on summarizing the skewness of the entire life of this portfolio.

Our work is not complete until we get to rolling skewness in the next section.

Rolling Skewness

Similar to the portfolio standard deviation, skewness is reported as one number for a portfolio and that one number can be misleading. Perhaps the first 2 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, we calculate and visuaize the

rolling skewness over time. We did something very similar with standard deviation for the same reasons and we will use very similar code.

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 and we will use a 6-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 rollercoaster.

```
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)
```

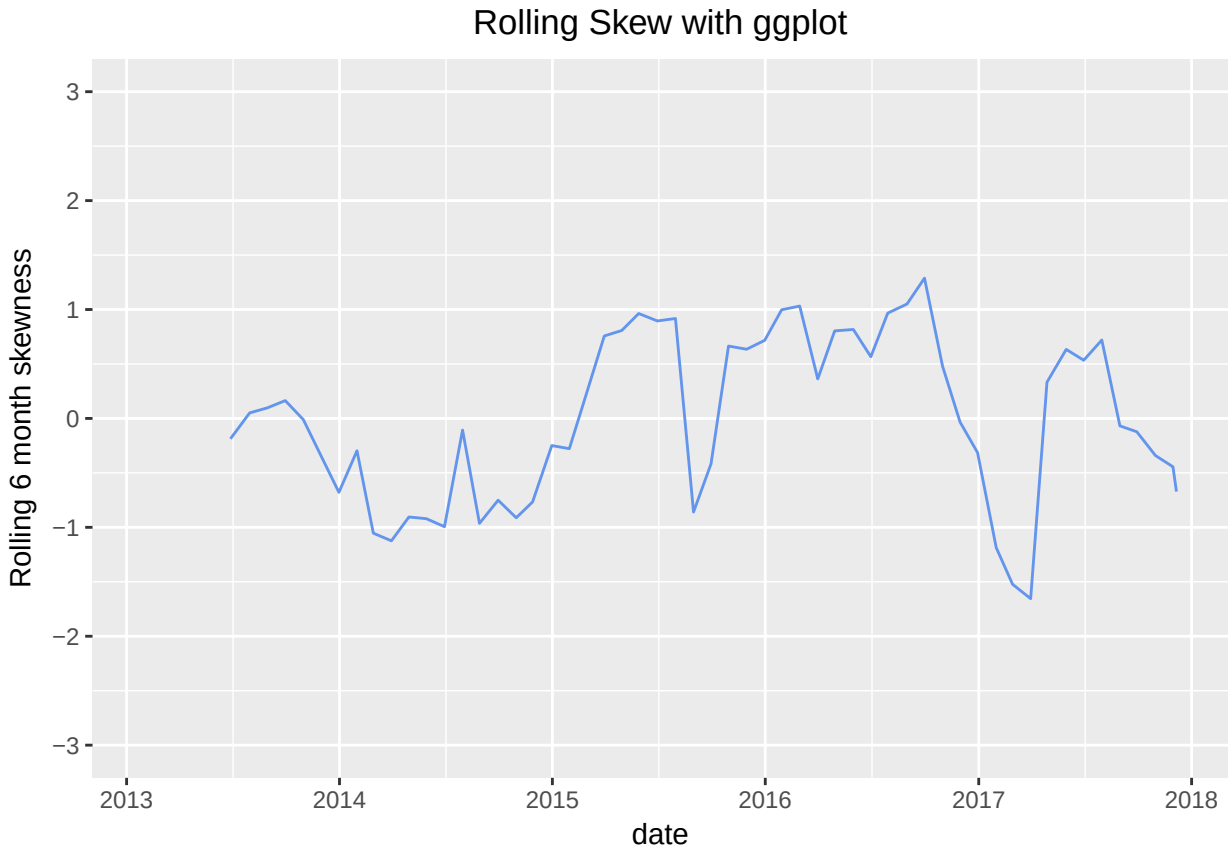
For completeness of methods, we can calculate rolling skewness in a `tibble` and then use `ggplot`.

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 build for time series plotting but we can set `aes(x = date, y = skew)` to make the x-axis our date values.

```
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))
```



The rolling charts are quite illuminating and show that the 6-month-interval skewness has been positive for about half the lifetime of this portfolio. Today, the 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.

In summary, the snapshot of the skewness for the life of the portfolio is informative but we need the rolling skewness to understand the whole story.

We painstakingly walked through the visualization process and that was intentional. When we head to Shiny and start sharing our work with end users, the goal is for those end users to love the data visualizations, which should lead the R team to create more visualizations. When the team starts to build more apps and projects, they will thank themselves for the painstaking work that makes the code readable and reusable, if a bit more verbose than necessary. And if team members should ever depart for greener pastures, and newbies need to reproduce an app, reproducibility is the only thing between the team and a complete start from scratch.

Let's cover kurtosis before building our Shiny app.

Kurtosis

Kurtosis is a measure of the degree to which our returns appear in the tails of our distribution. A normal distribution has a kurtosis of 3, which from a non-expert point of view means a normal distribution does have some of its mass in its tails as we expect. 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 because negative kurtosis actually is less risky because it has fewer returns out in the tails.

Kurtosis is often described as negative excess or positive excess, and that is in comparison to a kurtosis of 3. A distribution with negative excess kurtosis equal to -1 has an absolute kurtosis of 2, but we subtract 3 from 2 to get to -1. Remember though, the negative kurtosis means fewer returns in the tails, and probably less risky. Let's get to the calculations and visualizations.

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

$$Kurtosis = \sum_{t=1}^{n}(x_i - \overline{x})^4/n \Big/ (\sum_{t=1}^{n}(x_i - \overline{x})^2/n)^2 - 3$$

Now we are going to test ourselves and see if we can use the code above for our skewness calculations and visualizations to quickly do the same for kurtosis.

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] 0.4100983
```

For tidy, we again substitute functions, and need to use the formula for kurtosis for our by-hand caluclations.

```
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] 0.4100983
```

```
kurt_tidy$kurt_builtin
```

```
## [1] 0.4577633
```

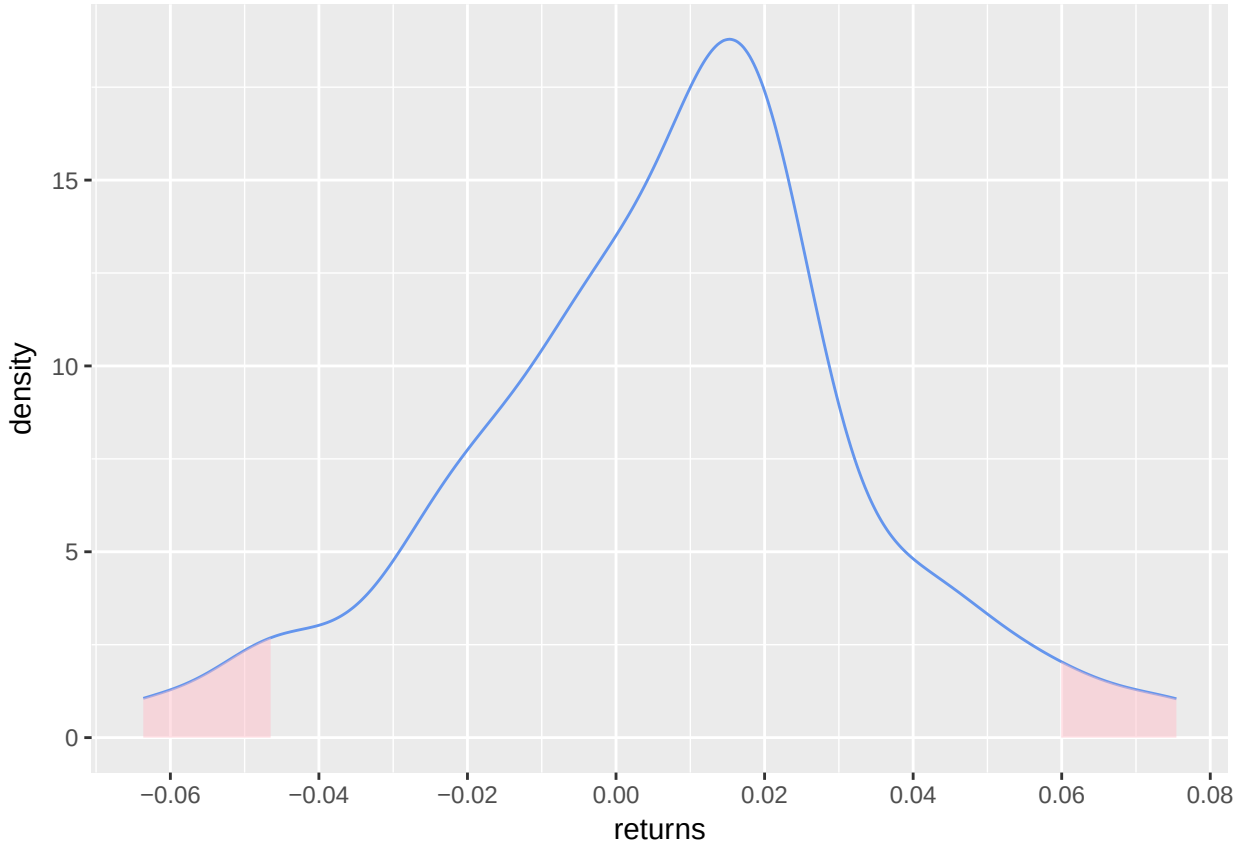
```
kurt_tidy$kurt_byhand
```

```
## [1] 0.4577633
```

We have consistent results from `xts` and the tidy world, and we were able to reuse our code from above to shorten the development time here.

Let's do the same with the visualizations and head straight for a density plot, starting with `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.

```
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 +
  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 = pretty_breaks(n = 10))
```



That density chart is a good look at how there seem to be a higher number of extreme negative returns, but the extreme positive returns are of a greater magnitude, mostly greater than .06.

And finally we can calculate and chart the rolling kurtosis as we did above.

```
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 = .2,
           min = -.2) %>%
  hc_navigator(enabled = FALSE) %>%
  hc_scrollbar(enabled = FALSE)
```

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

Our work on kurtosis was made a lot more efficient by our work on skewness. Now let's port this to a Shiny where we will again use our previous work to speed up development.

Shiny Skew and Kurtosis

We want to build a Shiny app that allows an end user to: 1) build a portfolio by choosing assets and weights 2) choose a start date for the portfolio 3) choose a rolling window 4) display rolling skewness 5) display rolling kurtosis 6) display returns distributions as visualized using `ggplot`

The structure and visualizations are very similar to what we built for calculating and visualizing the growth of a dollar. Have a quick look back at that Shiny app.

[www.reproduciblefinance.com/portfolio-growth-shiny]

The differences are that for skewness and kurtosis the end user chooses a rolling window and we display the results of rolling functions and not the dollar growth function.

Certain features are identical, such as how to build a portfolio and choose a start date. For those shared

features, we will use the exact same code, and we will add one line to allow the user to choose a window with this code: `numericInput("window", "Window", 6, min = 3, max = 20, step = 1))`.

```
fluidRow(
  column(6,
  textInput("stock1", "Stock 1", "SPY")),
  column(5,
  numericInput("w1", "Portf. %", 25, min = 1, max = 100))
)

fluidRow(
  column(6,
  textInput("stock2", "Stock 2", "EFA")),
  column(5,
  numericInput("w2", "Portf. %", 25, min = 1, max = 100))
)

fluidRow(
  column(6,
  textInput("stock3", "Stock 3", "IJS")),
  column(5,
  numericInput("w3", "Portf. %", 20, min = 1, max = 100))
)

fluidRow(
  column(6,
  textInput("stock4", "Stock 4", "EEM")),
  column(5,
  numericInput("w4", "Portf. %", 20, min = 1, max = 100))
)

fluidRow(
  column(6,
  textInput("stock5", "Stock 5", "AGG")),
  column(5,
  numericInput("w5", "Portf. %", 10, min = 1, max = 100))
)

fluidRow(
  column(6,
  dateInput("date", "Starting Date", "2013-01-01", format = "yyyy-mm-dd")),
  column(4,
  # The next line allows user to choose a window.
  numericInput("window", "Window", 6, min = 3, max = 20, step = 1)))

actionButton("go", "Submit")

# The prices object will hold our daily price data.
prices <- eventReactive(input$go, {

  symbols <- c(input$stock1, input$stock2, input$stock3, input$stock4, input$stock5)

  getSymbols(symbols, src = 'yahoo', from = input$date,
             auto.assign = TRUE, warnings = FALSE) %>%
```

```
  map(~Ad(get(.))) %>%
  reduce(merge) %>%
  `colnames<-`(symbols)
})
```

Next we want to put those prices to a portfolio returns object and since we'll be using `highcharter` to visualize the rolling statistics, we will stay in the `xts` world.

```
portfolio_returns_xts <- eventReactive(input$go, {
  prices <- prices()
  w <- c(input$w1/100, input$w2/100, input$w3/100, input$w4/100, input$w5/100)

  prices_monthly <- to.monthly(prices, indexAt = "last", OHLC = FALSE)
  asset_returns_xts <- na.omit(Return.calculate(prices_monthly, method = "log"))

  portfolio_returns_xts <-
  Return.portfolio(asset_returns_xts,
                   weights = w) %>%
    `colnames<-`("returns")
})
```

Let's calculate both rolling skewness and kurtosis and save those in objects that can be accessed in a later code chunk.

```
rolling_skew_xts <- eventReactive(input$go, {
  rolling_skew_xts <-
    na.omit(rollapply(portfolio_returns_xts(), input$window,
                      function(x) skewness(x)))

})

rolling_kurt_xts <- eventReactive(input$go, {
  rolling_kurt_xts <-
    na.omit(rollapply(portfolio_returns_xts(), input$window,
                      function(x) kurtosis(x)))

})
```

We now have two objects ready to be charted `rolling_skew_xts` and `rolling_kurt_xts` and we can pass them directly to `highcharter`. Note, though, that we do not display them side by side. Instead we put them in different tabs and let the end user toggle between them. We create those tabs with

"‘r Row {.tabset} ————————

Skewness

Kurtosis "‘

Then we place the following two code chunks in their respective rows:

```
### Skewness

renderHighchart({

  highchart(type = "stock") %>%
    hc_title(text = "Rolling Skew") %>%
    hc_add_series(rolling_skew_xts(), name = "rolling skew", color = "cornflowerblue") %>%
    hc_yAxis(title = list(text = "skewness"),
```

```
            opposite = FALSE,
            max = 3,
            min = -3) %>%
  hc_navigator(enabled = FALSE) %>%
  hc_scrollbar(enabled = FALSE)

})
```

```
### Kurtosis

renderHighchart({

  highchart(type = "stock") %>%
    hc_title(text = "Rolling Kurtosis") %>%
    hc_add_series(rolling_kurt_xts(), name = "rolling kurt", color = "cornflowerblue") %>%
    hc_yAxis(title = list(text = "kurtosis"),
            opposite = FALSE,
            max = 3,
            min = -3) %>%
  hc_navigator(enabled = FALSE) %>%
  hc_scrollbar(enabled = FALSE)

})
```

Lastly, we want to display different returns visualizations using `ggplot`, again very similar to how we did with the dollar growth Shiny app.

We first create a tidy portfolio returns object.

```
portfolio_byhand <- eventReactive(input$go, {

  prices <- prices()
  w <- c(input$w1/100, input$w2/100, input$w3/100, input$w4/100, input$w5/100)

  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))))

  portfolio_byhand <-
    asset_returns_long %>%
    tq_portfolio(assets_col = asset,
                 returns_col = returns,
                 weights = w,
                 col_rename = "returns")



})
```

Next we pass that object to `ggplot` for the different plots. Note we again use tabset so the user can toggle between different charts.

```
Row {.tabset .tabset-fade}
-------------------------------------
```

```
### Histogram with Number Thresh

renderPlot({

  portfolio_byhand <- portfolio_byhand()

  portfolio_byhand %>%
  mutate(hist_col_red =
           ifelse(returns < -.03,
                  returns, NA),
         hist_col_green =
           ifelse(returns > .03,
                  returns, NA),
         hist_col_blue =
           ifelse(returns > -.03 &
                  returns < .03,
                  returns, NA)) %>%
  ggplot() +

  geom_histogram(aes(x = hist_col_red),
                 alpha = .7,
                 binwidth = .003,
                 fill = "red",
                 color = "red") +

  geom_histogram(aes(x = hist_col_green),
                 alpha = .7,
                 binwidth = .003,
                 fill = "green",
                 color = "green") +

  geom_histogram(aes(x = hist_col_blue),
                 alpha = .7,
                 binwidth = .003,
                 fill = "cornflowerblue",
                 color = "cornflowerblue") +

scale_x_continuous(breaks = pretty_breaks(n = 10)) +
xlab("monthly returns")

})
```

```
### Histogram with Std Dev Thresh

renderPlot({

  portfolio_byhand <- portfolio_byhand()

  portfolio_byhand %>%
  mutate(hist_col_red =
           ifelse(returns < (mean(returns) - 2*sd(returns)),
```

```
                  returns, NA),
         hist_col_green =
           ifelse(returns > (mean(returns) + 2*sd(returns)),
                  returns, NA),
         hist_col_blue =
           ifelse(returns > (mean(returns) - 2*sd(returns)) &
                  returns < (mean(returns) + 2*sd(returns)),
                  returns, NA)) %>%
  ggplot() +

  geom_histogram(aes(x = hist_col_red),
                 alpha = .7,
                 binwidth = .003,
                 fill = "red",
                 color = "red") +

  geom_histogram(aes(x = hist_col_green),
                 alpha = .7,
                 binwidth = .003,
                 fill = "green",
                 color = "green") +

  geom_histogram(aes(x = hist_col_blue),
                 alpha = .7,
                 binwidth = .003,
                 fill = "cornflowerblue",
                 color = "cornflowerblue") +

scale_x_continuous(breaks = pretty_breaks(n = 10)) +
xlab("monthly returns")
})
```

```
### Density showing less than mean
renderPlot({

  portfolio_byhand <- portfolio_byhand()
  mean <- mean(portfolio_byhand$returns)
  median <- median(portfolio_byhand$returns)

  skew_density_plot <- portfolio_byhand %>%
    ggplot(aes(x = returns)) +
    stat_density(geom = "line", size = 1, color = "cornflowerblue")

  shaded_area_data <-
    ggplot_build(skew_density_plot)$data[[1]] %>%
    filter(x < mean)

  skew_density_plot_shaded <-
    skew_density_plot +
    geom_area(data = shaded_area_data, aes(x = x, y = y), fill="pink", alpha = 0.5)

  median_line_data <-
    ggplot_build(skew_density_plot)$data[[1]] %>%
    filter(x <= median)
```

```
skew_density_plot_shaded +

  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) +

  annotate(geom = "text", x = (mean - .03), y = .1, label = "returns < mean",
           fontface = "plain", color = "red", alpha = .8, vjust =  -1) +
  ggtitle("Density Plot Illustrating Skewness")

})
```

```
### Density showing tails

renderPlot({

  portfolio_byhand <- portfolio_byhand()

  sd_pos <- mean(portfolio_byhand$returns) + (2* sd(portfolio_byhand$returns))
  sd_neg <- mean(portfolio_byhand$returns) - (2* sd(portfolio_byhand$returns))

  kurt_density_plot <- portfolio_byhand %>%
    ggplot(aes(x = returns)) +
    stat_density(geom = "line", size = 1, color = "cornflowerblue")

sd_pos_shaded_area <-
  ggplot_build(kurt_density_plot)$data[[1]] %>%
  filter(x > sd_pos )

sd_neg_shaded_area <-
  ggplot_build(kurt_density_plot)$data[[1]] %>%
  filter(x < sd_neg)

  kurt_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 = pretty_breaks(n = 10))
})
```