Standard Deviation

Code

More than a long list of good stocks and bonds, a good portfolio is a balanced whole providing the investor with protections and opportunities with respect to a wide range of contingencies. In the next, we will analyze the portfolio from the aspect of the return and the standard deviation.

In this article, we calculate the monthly return and standard deviation of four funds.

We constructed a portfolio based on the four funds given a weights vector [0.25, 0.3, 0.25, 0.2]. And we calculate the portfolio's return and standard deviation using xts, PerformanceAnalytics, tidyquant and tidyverse and visualizing them with ggplot.

In order to show the time-dependent volatility of the portfolio and the four funds, we also calculated the rolling standard deviation and visualized them with ggplot and highcharter. The details of our code are above

Import the packages

Hide

```
library(highcharter)
library(tibbletime)
library(timetk)
library(shiny)
library(rmarkdown)
library(ggplot2)
library(xts)
library(DBI)
library(scales)
library(RMySQL)
library(purrr)
library(tidyverse)
library(tidyquant)
library(lubridate)
```

The assets we selected are:
华夏成长证券投资基金 000001
中海可转换债券债券型证券投资基金A类 000003
国联安中证医药100指数证券投资基金 000059
广发美国房地产指数证券投资基金（美元）000180

Import the daily net value data of the four funds from the database

Hide

```
mydb= dbConnect(MySQL(),user='ktruc002', password='35442fed', dbname='cn_stock_quote', host=
'172.19.3.250')
SQL_statement<- "SELECT  `end_date`,  `fund_code`, `acc_net_value`
FROM `cn_fund`.`net_value`
WHERE fund_code IN (000001,000003,000059,000180)
ORDER BY `end_date` DESC"
funds <- dbGetQuery(mydb,SQL_statement)
```

```
Decimal MySQL column 2 imported as numeric
```

Hide

```
funds <- reshape(funds, idvar = "end_date", timevar = "fund_code", direction = "wide")
funds <- zoo::na.locf(funds)
time <- funds$end_date %>% as.Date()
funds <- xts(funds[2:5],time)
tail(funds,5)
```

```
           acc_net_value.000001 acc_net_value.000003 acc_net_value.000059 acc_net_value.00018
0
2019-09-27                3.524                0.966               1.4758                0.242
6
2019-09-30                3.512                0.959               1.4733                0.242
6
2019-10-08                3.511                0.955               1.4796                0.241
6
2019-10-09                3.515                0.959               1.4801                0.241
9
2019-10-10                3.528                0.965               1.5018                0.241
9
```

Calculate the assets' monthly return series

Hide

```
funds <- to.period(funds,period = "months")
asset_returns_xts <-
  Return.calculate(funds,
                   method = "log") %>%
  na.omit()

asset_returns_xts <- asset_returns_xts["201309/201909"]
names(asset_returns_xts) <- c("funds.000001","funds.000003","funds.000059","funds.000180")
tail(asset_returns_xts,5)
```

```
           funds.000001 funds.000003 funds.000059  funds.000180
2019-05-31 -0.025747638 -0.086207195  -0.04032833 -0.0004344992
2019-06-30 -0.004884365 -0.004296462  -0.03453873 -0.0056657375
2019-07-31  0.017699577  0.011771136   0.03866685  0.0199054603
2019-08-30 -0.007384299  0.022094534  -0.01344318  0.0203569371
2019-09-30  0.004834362  0.036776361   0.05762623  0.0183033396
```

4.0 Compute portfolio covariance matrix

Hide

```
covariance_matrix <- cov(asset_returns_xts)
round(covariance_matrix,5)
```

```
             funds.000001 funds.000003 funds.000059 funds.000180
funds.000001      0.00060      0.00111      0.00091     -0.00015
funds.000003      0.00111      0.00390      0.00232     -0.00021
funds.000059      0.00091      0.00232      0.00360      0.00009
funds.000180     -0.00015     -0.00021      0.00009      0.00122
```

Genertate weights and calculate portfolio standard deviation

Hide

```
w = c(0.25,0.3,0.25,0.2)
# calculate standard deviation of portfolio
sd_matrix_algebra <- sqrt(t(w) %*% covariance_matrix %*% w)
#calculate std in percent
sd_matrix_algebra_percent <-
  round(sd_matrix_algebra * 100, 2) %>%
  `colnames<-`("standard deviation")
sd_matrix_algebra_percent
```

```
     standard deviation
[1,]               3.55
```

4.1 Calculate standard deviation using Performance Analytics

Hide

```
portfolio_sd_xts_builtin <-
  PerformanceAnalytics::StdDev(asset_returns_xts, weights = w)
#std in percent
portfolio_sd_xts_builtin_percent <-
  round(portfolio_sd_xts_builtin * 100, 2)
portfolio_sd_xts_builtin_percent
```

```
     [,1]
[1,] 3.55
```

4.2 Calculate standard deviation using tidyverse

Hide

```
assets_returns_dplyr <- tk_tbl(asset_returns_xts)
portfolio_returns_dplyr_byhand <- assets_returns_dplyr[1]
portfolio_returns_dplyr_byhand <- add_column(portfolio_returns_dplyr_byhand,returns = 0)
for (i in c(1,2,3,4)){
  portfolio_returns_dplyr_byhand["returns"] = portfolio_returns_dplyr_byhand["returns"] + w
[i]*assets_returns_dplyr[i+1]
}

portfolio_sd_tidy_builtin_percent <-
  portfolio_returns_dplyr_byhand %>%
  summarise(
    sd = sd(returns),
    sd_byhand =
      sqrt(sum((returns - mean(returns))^2)/(nrow(.)-1))) %>%
  mutate(dplyr = round(sd, 4) * 100,
         dplyr_byhand = round(sd_byhand, 4) * 100)

portfolio_sd_tidy_builtin_percent %>%
  select(dplyr, dplyr_byhand)
```

dplyr <dbl>	dplyr_byhand <dbl>
3.55	3.55

1 row

Hide

```
portfolio_sd_tidy_builtin_percent
```

sd <dbl>	sd_byhand <dbl>	dplyr <dbl>	dplyr_byhand <dbl>
0.03549137	0.03549137	3.55	3.55

1 row

4.3 Calculate standard deviation using tidyquant

Hide

```
asset_returns_long <- assets_returns_dplyr %>% gather (assets,returns,-index) %>% group_by(as
sets)

portfolio_returns_tq_rebalanced_monthly  <- asset_returns_long %>%
      tq_portfolio(assets_col = assets,
                   returns_col = returns,
                   weights = w,
                   col_rename = "returns",
                   rebalance_on = "months")

portfolio_sd_tidyquant_builtin_percent <- tq_performance(portfolio_returns_tq_rebalanced_mont
hly,
                                          Ra = returns,
                                          Rb = NULL,
                                          performance_fun = table.Stats) %>%
                                          select(Stdev) %>%
                                          mutate(tq_sd = round(Stdev,4)*100)
portfolio_sd_tidyquant_builtin_percent
```

Stdev <dbl>	tq_sd <dbl>
0.0355	3.55

1 row

Demonstrate the Stds calculated by different methods

Hide

```
portfolio_sd_tidy_builtin_percent %>%
  select(dplyr, dplyr_byhand) %>%
  mutate(xts_builtin = portfolio_sd_xts_builtin_percent,
         matrix = sd_matrix_algebra_percent,
         tq = portfolio_sd_tidyquant_builtin_percent$tq_sd)
```

Hide

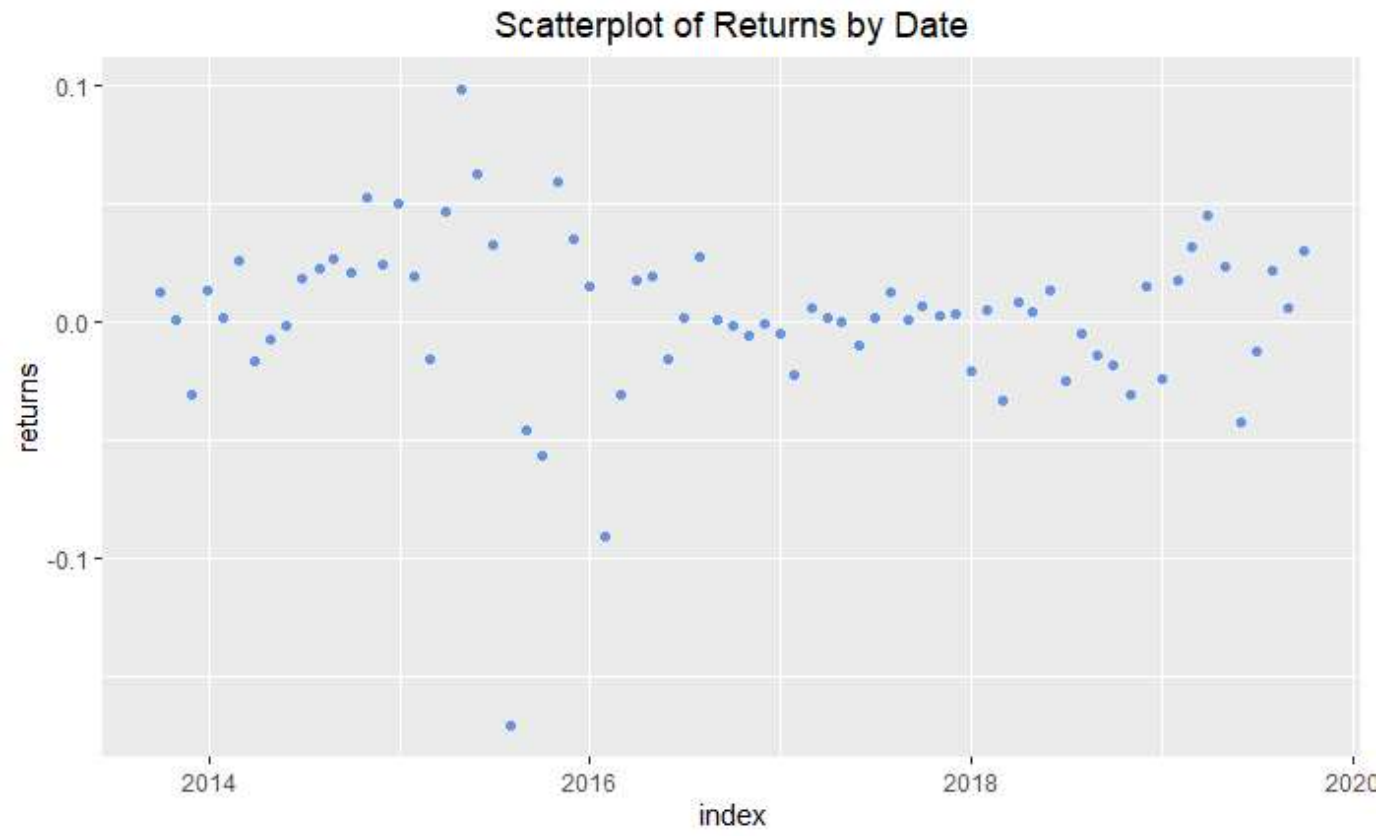
```
portfolio_sd_tidy_builtin_percent
```

4.4 Visualizing Standard Deviation

Show the scatter plot of monthly returns

Hide

```
portfolio_returns_dplyr_byhand %>%
  ggplot(aes(x = index, y = returns)) +
  geom_point(color = "cornflowerblue") +
  scale_x_date(breaks = pretty_breaks(n = 6)) +
  ggtitle("Scatterplot of Returns by Date") +
  theme(plot.title = element_text(hjust = 0.5))
```



Create indicators sd_plot/mean_plot

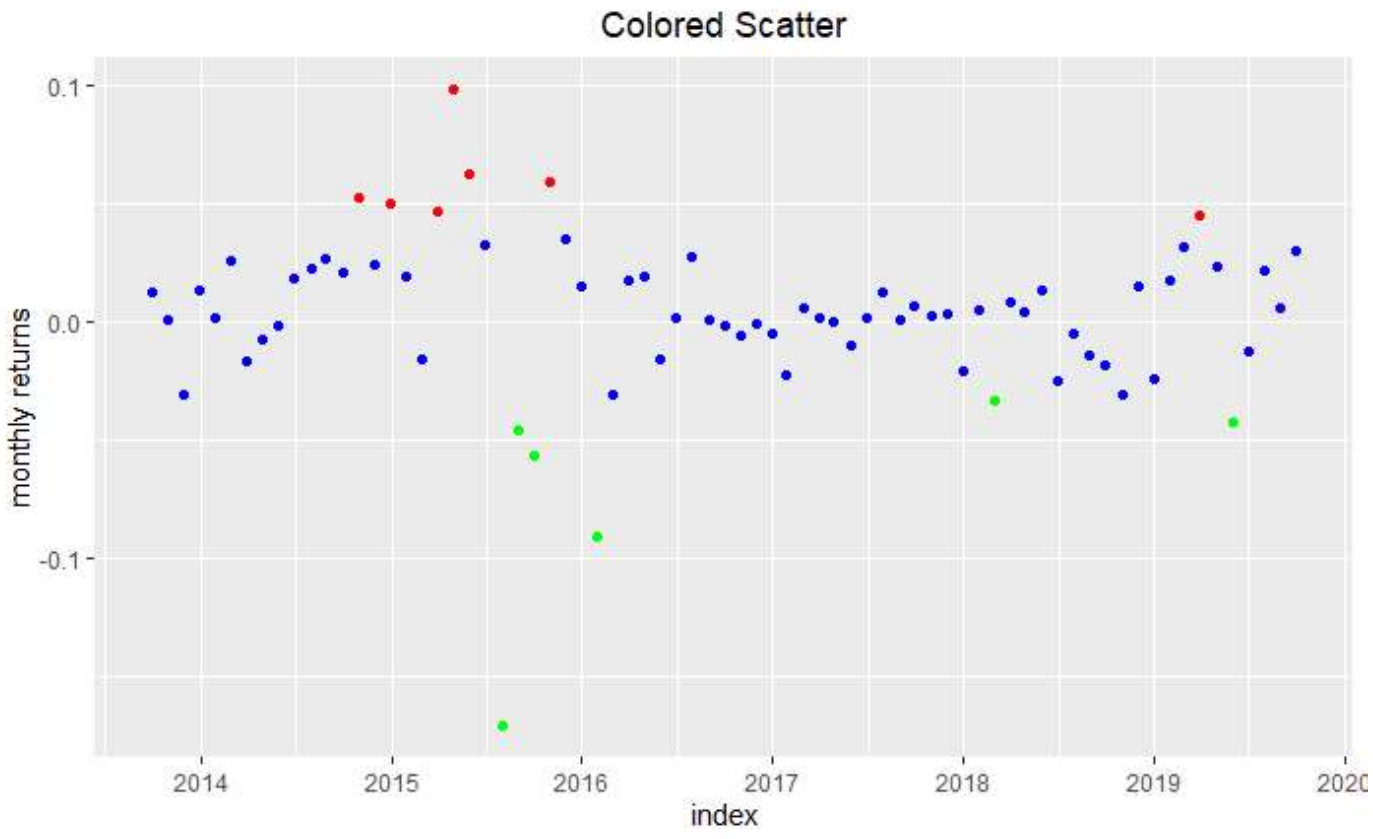
Hide

```
sd_plot <-
  sd(portfolio_returns_tq_rebalanced_monthly$returns)
mean_plot <-
  mean(portfolio_returns_tq_rebalanced_monthly$returns)
```

Set the returns that deviate from the mean too much to red/green

Hide

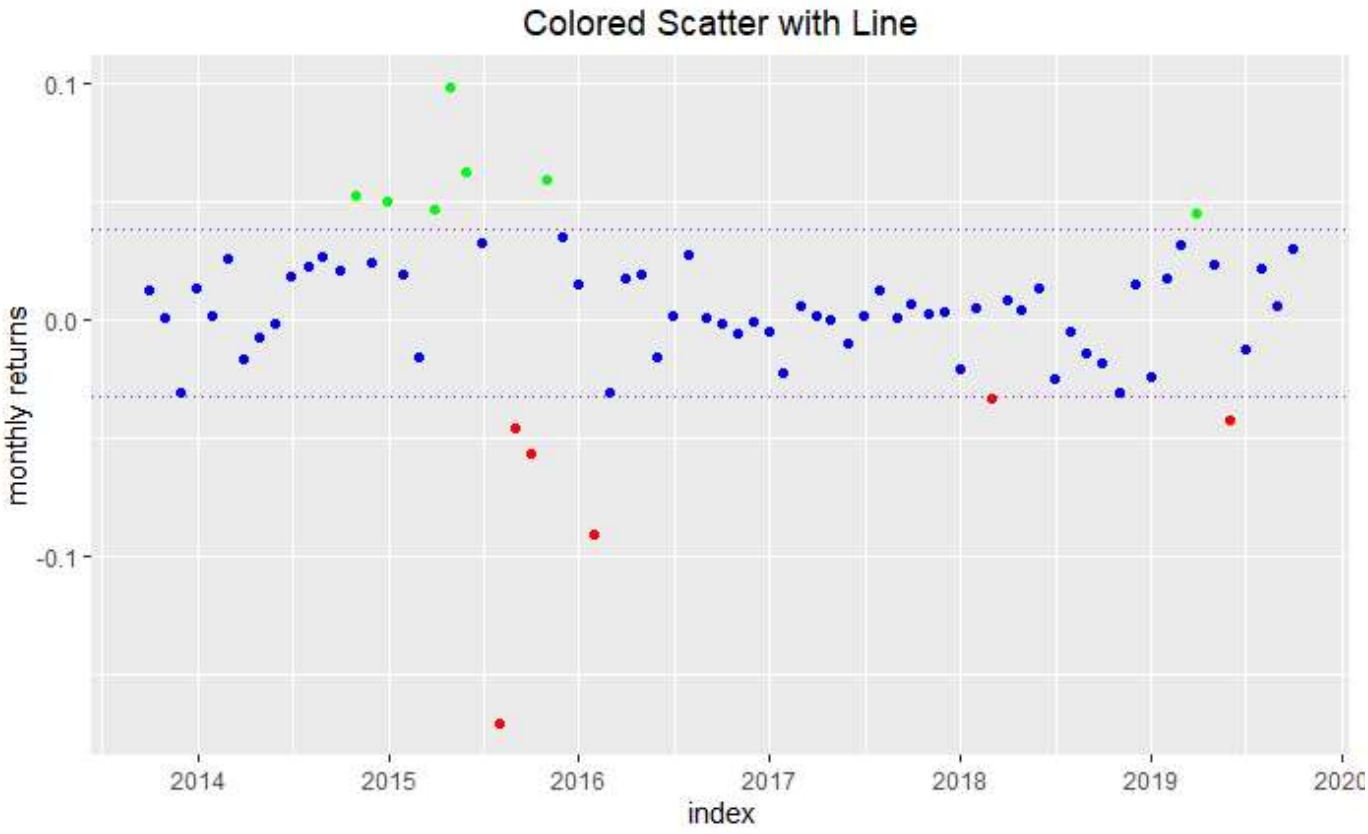
```
portfolio_returns_tq_rebalanced_monthly %>%
  mutate(hist_col_red =
           if_else(returns < (mean_plot - sd_plot),
                   returns, as.numeric(NA)),
         hist_col_green =
           if_else(returns > (mean_plot + sd_plot),
                   returns, as.numeric(NA)),
         hist_col_blue =
           if_else(returns > (mean_plot - sd_plot) &
                     returns < (mean_plot + sd_plot),
                   returns, as.numeric(NA))) %>%
  ggplot(aes(x = index)) +
  geom_point(aes(y = hist_col_red),
             color = "green") +
  geom_point(aes(y = hist_col_green),
             color = "red") +
  geom_point(aes(y = hist_col_blue),
             color = "blue") +
  labs(title = "Colored Scatter", y = "monthly returns") +
  scale_x_date(breaks = pretty_breaks(n = 8)) +
  theme(plot.title = element_text(hjust = 0.5))
```



Add a line for the value that is one standard deviation above and below the mean

Hide

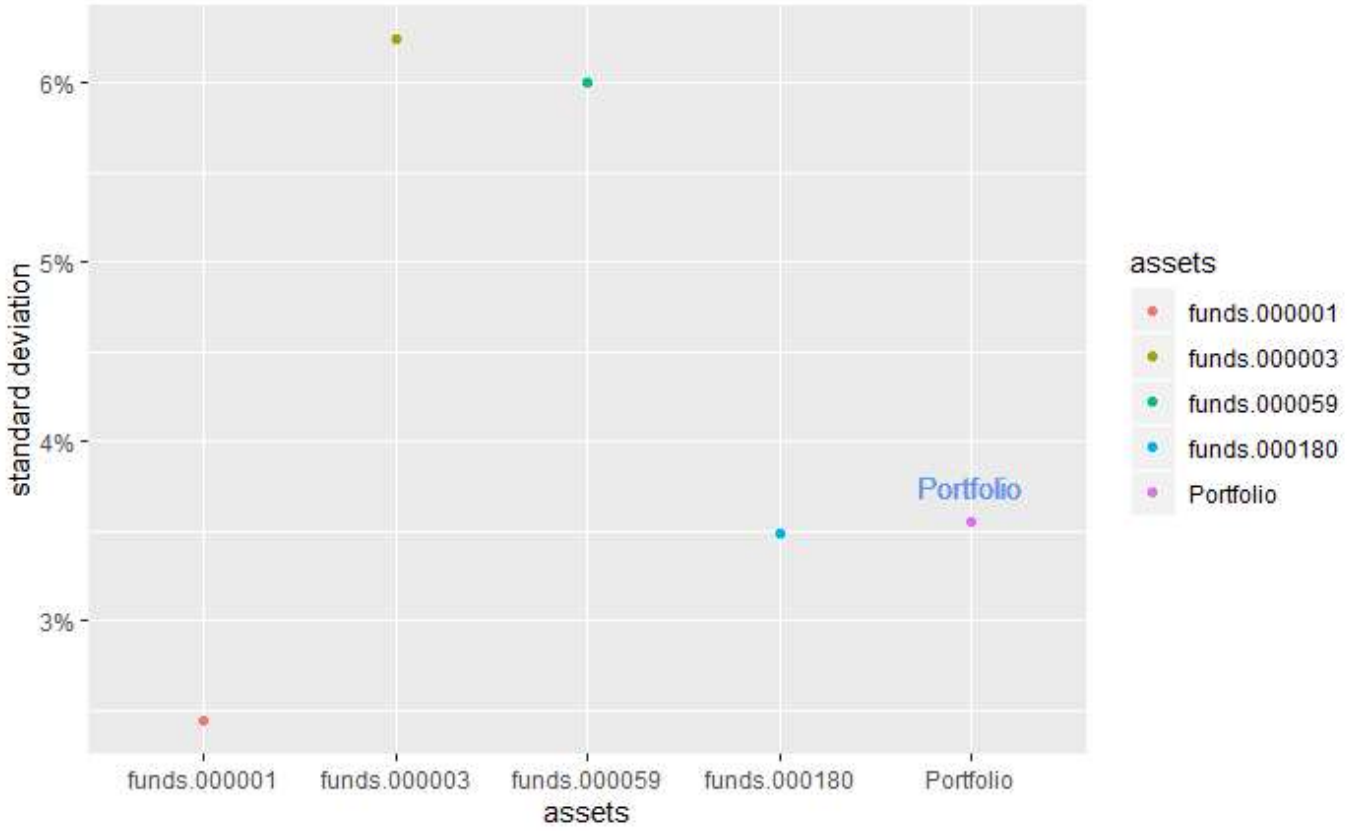
```
portfolio_returns_tq_rebalanced_monthly %>%
  mutate(hist_col_red =
           if_else(returns < (mean_plot - sd_plot),
                   returns, as.numeric(NA)),
         hist_col_green =
           if_else(returns > (mean_plot + sd_plot),
                   returns, as.numeric(NA)),
         hist_col_blue =
           if_else(returns > (mean_plot - sd_plot) &
                     returns < (mean_plot + sd_plot),
                   returns, as.numeric(NA))) %>%
  ggplot(aes(x = index)) +
  geom_point(aes(y = hist_col_red),
             color = "red") +
  geom_point(aes(y = hist_col_green),
             color = "green") +
  geom_point(aes(y = hist_col_blue),
             color = "blue") +
  geom_hline(yintercept = (mean_plot + sd_plot),
             color = "purple",
             linetype = "dotted") +
  geom_hline(yintercept = (mean_plot-sd_plot),
             color = "purple",
             linetype = "dotted") +
  labs(title = "Colored Scatter with Line", y = "monthly returns") +
  scale_x_date(breaks = pretty_breaks(n = 8)) +
  theme(plot.title = element_text(hjust = 0.5))
```



Visualize the actual standard deviation of our portfolio

Hide

```
asset_returns_long %>%
  group_by(assets) %>%
  summarize(sd = 100 *sd(returns)) %>%
  add_row(assets = "Portfolio",
          sd = portfolio_sd_tidy_builtin_percent$dplyr) %>%
  ggplot(aes(x = assets,
             y = sd,
             colour = assets)) +
  geom_point() +
  scale_y_continuous(labels = function(x) paste0(x, "%")) +
  geom_text(
    aes(x = "Portfolio",
        y =
          portfolio_sd_tidy_builtin_percent$dplyr + .2),
    label = "Portfolio",
    color = "cornflowerblue") +
  labs(y = "standard deviation")
```



Visualizing expected monthly returns

Hide

```
asset_returns_long %>%
  group_by(assets) %>%
  summarise(expected_return = mean(returns),
            stand_dev = sd(returns)) %>%
  add_row(assets = "Portfolio",
          stand_dev =
            sd(portfolio_returns_tq_rebalanced_monthly$returns),
          expected_return =
            mean(portfolio_returns_tq_rebalanced_monthly$returns)) %>%
  ggplot(aes(x = stand_dev,
             y = expected_return,
             color = assets)) +
  geom_point(size = 2) +
  geom_text(
    aes(x =
          sd(portfolio_returns_tq_rebalanced_monthly$returns) * 1.11,
        y =
          mean(portfolio_returns_tq_rebalanced_monthly$returns),
        label = "Portfolio")) +
  ylab("expected return") +
  xlab("standard deviation") +
  ggtitle("Expected Monthly Returns versus Risk") +
  scale_y_continuous(labels = function(x){ paste0(x, "%")}) +
  # The next line centers the title
  theme_update(plot.title = element_text(hjust = 0.5))
```

4.6 Rolling Standard Deviation in the xts world

Hide

```
window<-24
portfolio_returns_xts_rebalanced_monthly <-
  Return.portfolio(asset_returns_xts,
                   weights = w,
                   rebalance_on = "months") %>%
  `colnames<-`("returns")
# calculate rolling standard deviation based on xts
port_rolling_sd_xts <-
  rollapply(portfolio_returns_xts_rebalanced_monthly,
            FUN = sd,
            width = window) %>%
            na.omit() %>%
            `colnames<-`("rolling_sd")
tail(port_rolling_sd_xts,5)
```

```
           rolling_sd
2019-05-31 0.02160900
2019-06-30 0.02172044
2019-07-31 0.02205498
2019-08-30 0.02209345
2019-09-30 0.02295307
```

4.7 Rolling standard deviation based on tidyverse

Hide

```
port_rolling_sd_tidy_does_not_work <-
  portfolio_returns_dplyr_byhand %>%
  mutate(rolling_sd = rollapply(returns,
                                FUN = sd,
                                width = window,
                                fill = NA)) %>%
  select(index, rolling_sd) %>%
  na.omit()
tail(port_rolling_sd_tidy_does_not_work,3)
```

4.8 Rolling Standard Deviation with the tidyverse

Hide

```
# define a rolling sd function based on rollify of tibbletime package
sd_roll_24 <-
  rollify(sd, window = window)
# calculate the rolling standard deviation using tibbletime
port_rolling_sd_tidy_tibbletime <-
  portfolio_returns_tq_rebalanced_monthly %>%
  as_tbl_time(index = index) %>%
  mutate(sd = sd_roll_24(returns)) %>%
  select(-returns) %>%
  na.omit()
tail(port_rolling_sd_tidy_tibbletime, 3)
```

```
# A time tibble: 3 x 2
[38;5;246m# Index: index[39m
  index          sd
  [3m[38;5;246m<date>[39m[23m      [3m[38;5;246m<dbl>[39m[23m
[38;5;250m1[39m 2019-07-31 0.022[4m1[24m
[38;5;250m2[39m 2019-08-30 0.022[4m1[24m
[38;5;250m3[39m 2019-09-30 0.023[4m0[24m
```

4.9 Rolling Standard Deviation in the tidyquant world

Hide

```
## calculate std using tidyquant
port_rolling_sd_tq <-
  portfolio_returns_tq_rebalanced_monthly %>%
  tq_mutate(mutate_fun = rollapply,
            width = window,
            FUN = sd,
            col_rename = "rolling_sd") %>%
  select(index, rolling_sd) %>%
  na.omit()
## confirm three different methods
port_rolling_sd_tidy_tibbletime %>%
  mutate(sd_tq = port_rolling_sd_tq$rolling_sd,
         sd_xts = round(port_rolling_sd_xts$rolling_sd, 4)) %>%
  tail(3)
```

```
# A time tibble: 3 x 4
[38;5;246m# Index: index[39m
  index          sd  sd_tq sd_xts[,"rolling_sd"]
  [3m[38;5;246m<date>[39m[23m      [3m[38;5;246m<dbl>[39m[23m  [3m[38;5;246m<dbl>[39m[23m [3
m[38;5;246m<xts>[39m[23m
[38;5;250m1[39m 2019-07-31 0.022[4m1[24m 0.022[4m1[24m 0.0221
[38;5;250m2[39m 2019-08-30 0.022[4m1[24m 0.022[4m1[24m 0.0221
[38;5;250m3[39m 2019-09-30 0.023[4m0[24m 0.023[4m0[24m 0.0230
```

4.10 Visualizing Rolling Standard Deviation in the xts world

Hide

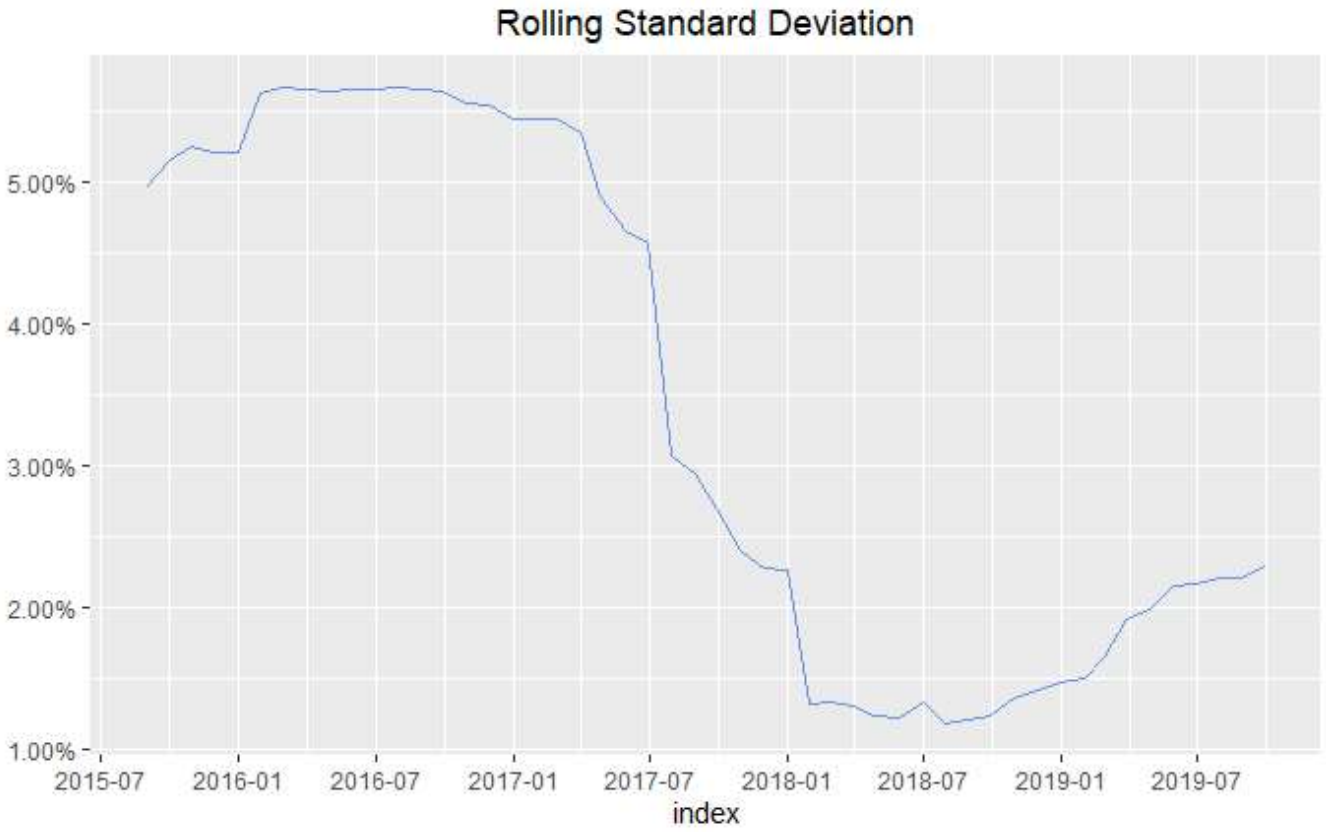
```
port_rolling_sd_xts_hc <-
  round(port_rolling_sd_xts, 4) * 100

highchart(type = "stock") %>%
  hc_title(text = "24-Month Rolling Volatility") %>%
  hc_add_series(port_rolling_sd_xts_hc,
                color = "cornflowerblue") %>%
  hc_add_theme(hc_theme_flat()) %>%
  hc_yAxis(
    labels = list(format = "{value}%"),
    opposite = FALSE) %>%
  hc_navigator(enabled = FALSE) %>%
  hc_scrollbar(enabled = FALSE) %>%
  hc_exporting(enabled= TRUE) %>%
  hc_legend(enabled = TRUE)
```

4.11 Visualizing Rolling Standard Deviation in the tidyverse

Hide

```
port_rolling_sd_tq %>%
  ggplot(aes(x = index)) +
  geom_line(aes(y = rolling_sd), color = "cornflowerblue") +
  scale_y_continuous(labels = scales::percent) +
  scale_x_date(breaks = pretty_breaks(n = 8)) +
  labs(title = "Rolling Standard Deviation", y = "") +
  theme(plot.title = element_text(hjust = 0.5))
```



4.12 The Shiny is written in the app.Rmd in the Rmarkdown format