Portfolio Analytics

```
data.file <- file.path(data.dir, "Reproducible Finance.csv")

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

# Yahoo! Finance

prices <- getSymbols(symbols,
             from = "2012-12-31",
             to = "2017-12-31",
             auto.assign = T,
             warnings = F) %>%
          map(~Ad(get(.))) %>%
  reduce(merge) %>%
  `colnames<-`(symbols)
```

```
'getSymbols' currently uses auto.assign=TRUE by default, but will
use auto.assign=FALSE in 0.5-0. You will still be able to use
'loadSymbols' to automatically load data. getOption("getSymbols.env")
and getOption("getSymbols.auto.assign") will still be checked for
alternate defaults.

This message is shown once per session and may be disabled by setting
options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
# CSV

prices <- read_csv(data.file,
                   col_types =
                     cols(date =
                            col_date(format = "%Y-%m-%d"))) %>%
  tk_xts(date_var = date)
```

```
Warning in tk_xts_.data.frame(data = data, select = select, date_var =
date_var, : Non-numeric columns being dropped: date
```

```
# Excel

prices <-
  read_excel(file.path(data.dir, "Reproducible Finance.xlsx"),
             col_types = c("text", "numeric",
                           "numeric", "numeric",
                           "numeric", "numeric")) %>%
  mutate(date = ymd(date)) %>%
  tk_xts(date_var = date)
```

```
Warning in tk_xts_.data.frame(data = data, select = select, date_var =
date_var, : Non-numeric columns being dropped: date
```

```
# Verfiy Import
head(prices, 3)
```

```
                SPY      EFA      IJS      EEM      AGG
2012-12-31 128.3092 49.16410 75.06590 39.89233 98.19626
2013-01-02 131.5977 49.92501 77.12553 40.67489 98.08131
2013-01-03 131.3004 49.44080 77.02349 40.38705 97.83374
```

```
# convert to monthly prices.

prices_monthly <- to.monthly(prices,
                             indexAt = "lastof",
                             OHLC = F)

head(prices_monthly)
```

```
                SPY      EFA      IJS      EEM      AGG
2012-12-31 128.3092 49.16410 75.06590 39.89233 98.19626
2013-01-31 134.8773 50.99717 79.08315 39.77539 97.58625
2013-02-28 136.5982 50.34004 80.37274 38.86691 98.16285
2013-03-31 141.7850 50.99717 83.67441 38.47113 98.25957
2013-04-30 144.5090 53.55654 83.77677 38.93888 99.21130
2013-05-31 147.9209 51.93964 87.36826 37.05894 97.22598
```

```
# Convert to monthly returns, xts.

asset_returns_xts <-
  Return.calculate(prices_monthly,
                   method = "log") %>%
  na.omit()

head(asset_returns_xts, 3)
```

```
                  SPY         EFA        IJS          EEM          AGG
2013-01-31 0.04992297  0.03660636 0.05213343 -0.002935495 -0.006231517
2013-02-28 0.01267831 -0.01296938 0.01617522 -0.023105260  0.005891222
2013-03-31 0.03726793  0.01296938 0.04025808 -0.010235026  0.000984796
```

```
# Convert to monthly returns, dplyr.

asset_returns_dplyr_byhand <-
  prices %>%
  to.monthly(indexAt = "lastof", OHLC = F) %>%
  # convert the index to a date
  data.frame(date = index(.)) %>%
```

```
  # now remove the index because it got converted to row names
  remove_rownames() %>%
  gather(asset, prices, - date) %>%
  group_by(asset) %>%
  mutate(returns = (log(prices) - log(lag(prices)))) %>%
  select(-prices) %>%
  spread(asset, returns) %>%
  select(date, symbols)

head(asset_returns_dplyr_byhand)
```

```
# A tibble: 6 x 6
  date            SPY     EFA      IJS      EEM       AGG
  <date>        <dbl>   <dbl>    <dbl>    <dbl>     <dbl>
1 2012-12-31 NA      NA       NA       NA        NA
2 2013-01-31  0.0499  0.0366  0.0521  -0.00294 -0.00623
3 2013-02-28  0.0127 -0.0130  0.0162  -0.0231   0.00589
4 2013-03-31  0.0373  0.0130  0.0403  -0.0102   0.000985
5 2013-04-30  0.0190  0.0490  0.00122  0.0121   0.00964
6 2013-05-31  0.0233 -0.0307  0.0420  -0.0495  -0.0202
```

```
asset_returns_dplyr_byhand <- asset_returns_dplyr_byhand %>%
  na.omit()

head(asset_returns_dplyr_byhand)
```

```
# A tibble: 6 x 6
  date            SPY     EFA      IJS      EEM       AGG
  <date>        <dbl>   <dbl>    <dbl>    <dbl>     <dbl>
1 2013-01-31  0.0499  0.0366  0.0521  -0.00294 -0.00623
2 2013-02-28  0.0127 -0.0130  0.0162  -0.0231   0.00589
3 2013-03-31  0.0373  0.0130  0.0403  -0.0102   0.000985
4 2013-04-30  0.0190  0.0490  0.00122  0.0121   0.00964
5 2013-05-31  0.0233 -0.0307  0.0420  -0.0495  -0.0202
6 2013-06-30 -0.0134 -0.0272 -0.00140 -0.0547  -0.0158
```

```
# convert to monthly returns, tidyquant.

asset_returns_tq_builtin <-
  prices %>%
  tk_tbl(preserve_index = T,
         rename_index = "date") %>%
  gather(asset, prices, -date) %>%
  group_by(asset) %>%
  tq_transmute(mutate_fun = periodReturn,
               period = "monthly",
```

```
               type = "log") %>%
  spread(asset, monthly.returns) %>%
  select(date, symbols) %>%
  slice(-1)

head(asset_returns_tq_builtin)
```

```
# A tibble: 6 x 6
  date            SPY     EFA      IJS      EEM       AGG
  <date>        <dbl>   <dbl>    <dbl>    <dbl>     <dbl>
1 2013-01-31  0.0499  0.0366  0.0521  -0.00294 -0.00623
2 2013-02-28  0.0127 -0.0130  0.0162  -0.0231   0.00589
3 2013-03-28  0.0373  0.0130  0.0403  -0.0102   0.000985
4 2013-04-30  0.0190  0.0490  0.00122  0.0121   0.00964
5 2013-05-31  0.0233 -0.0307  0.0420  -0.0495  -0.0202
6 2013-06-28 -0.0134 -0.0272 -0.00140 -0.0547  -0.0158
```

```
# convert to monthly returns, tibbletime.

asset_returns_tbltime <-
  prices %>%
  tk_tbl(preserve_index = T,
         rename_index = "date") %>%
  # this is the tibbletime function
  as_tbl_time(index = date) %>%
  as_period(period = "monthly",
            side = "end") %>%
  gather(asset, returns, - date) %>%
  group_by(asset) %>%
  tq_transmute(mutate_fun = periodReturn,
               type = "log") %>%
  spread(asset, monthly.returns) %>%
  select(date, symbols) %>%
  slice(-1)

head(asset_returns_tbltime)
```

```
# A time tibble: 6 x 6
# Index: date
  date            SPY     EFA      IJS      EEM       AGG
  <date>        <dbl>   <dbl>    <dbl>    <dbl>     <dbl>
1 2013-01-31  0.0499  0.0366  0.0521  -0.00294 -0.00623
2 2013-02-28  0.0127 -0.0130  0.0162  -0.0231   0.00589
3 2013-03-28  0.0373  0.0130  0.0403  -0.0102   0.000985
4 2013-04-30  0.0190  0.0490  0.00122  0.0121   0.00964
```

```
5 2013-05-31  0.0233 -0.0307  0.0420  -0.0495  -0.0202
6 2013-06-28 -0.0134 -0.0272 -0.00140 -0.0547  -0.0158
```

```
asset_returns_long <-
  asset_returns_dplyr_byhand %>%
  gather(asset, returns, - date) %>%
  group_by(asset)

head(asset_returns_long)
```

```
# A tibble: 6 x 3
# Groups:   asset [1]
  date       asset returns
  <date>     <chr>   <dbl>
1 2013-01-31 SPY    0.0499
2 2013-02-28 SPY    0.0127
3 2013-03-31 SPY    0.0373
4 2013-04-30 SPY    0.0190
5 2013-05-31 SPY    0.0233
6 2013-06-30 SPY   -0.0134
```

```
# Asset Weights
w <- c(0.25,
       0.25,
       0.20,
       0.20,
       0.10)

stopifnot(sum(w) == 1)

tibble(w, symbols) %>%
  summarise(total_weights = sum(w))
```

```
# A tibble: 1 x 1
  total_weights
          <dbl>
1             1
```

```
# Portfolio returns, dplyr.
portfolio_returns_dplyr_byhand <-
  asset_returns_long %>%
  group_by(asset) %>%
  mutate(weights = case_when(asset == symbols[1] ~ w[1],
                             asset == symbols[2] ~ w[2],
                             asset == symbols[3] ~ w[3],
                             asset == symbols[4] ~ w[4],
                             asset == symbols[5] ~ w[5]),
```

```
        weighted_returns = returns * weights) %>%
  group_by(date) %>%
  summarize(returns = sum(weighted_returns))

head(portfolio_returns_dplyr_byhand)
```

```
# A tibble: 6 x 2
  date         returns
  <date>         <dbl>
1 2013-01-31  0.0308
2 2013-02-28 -0.000870
3 2013-03-31  0.0187
4 2013-04-30  0.0206
5 2013-05-31 -0.00535
6 2013-06-30 -0.0230
```

```
# Portfolio returns, tidyquant.
portfolio_returns_tq_rebalanced_monthly <-
  asset_returns_long %>%
  tq_portfolio(assets_col = asset,
               returns_col = returns,
               weights = w,
               col_rename = "returns",
               rebalance_on = "months")

head(portfolio_returns_tq_rebalanced_monthly, 3)
```

```
# A tibble: 3 x 2
  date         returns
  <date>         <dbl>
1 2013-01-31  0.0308
2 2013-02-28 -0.000870
3 2013-03-31  0.0187
```

```
# Calculate Covarariance, by hand

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

```
         SPY     EFA      IJS     EEM      AGG
SPY  0.00074 0.00070  0.00083 0.00068 -0.00001
EFA  0.00070 0.00106  0.00065 0.00104  0.00004
IJS  0.00083 0.00065  0.00156 0.00065 -0.00008
EEM  0.00068 0.00104  0.00065 0.00175  0.00011
AGG -0.00001 0.00004 -0.00008 0.00011  0.00007
```

```
# Standard Deviation, by hand
sd_matrix_algebra <- sqrt(t(w) %*% covariance_matrix %*% w)

sd_matrix_algebra_percent <-
  round(sd_matrix_algebra * 100, 2) %>%
  `colnames<-`("standard deviation")

sd_matrix_algebra_percent[1,]
```

```
standard deviation
              2.66
```

```
# SD, xts

portfolio_sd_xts_builtin <-
  StdDev(asset_returns_xts, weights = w)

portfolio_sd_xts_builtin_percent <-
  round(portfolio_sd_xts_builtin * 100, 2)

portfolio_sd_xts_builtin_percent[1,]
```

```
[1] 2.66
```

```
# SD, tidyverse

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)
```

```
# A tibble: 1 x 2
  dplyr dplyr_byhand
  <dbl>        <dbl>
1  2.66         2.66
```

```
# SD, tidyquant

portfolio_sd_tidyquant_builtin_percent <-
  portfolio_returns_tq_rebalanced_monthly %>%
```

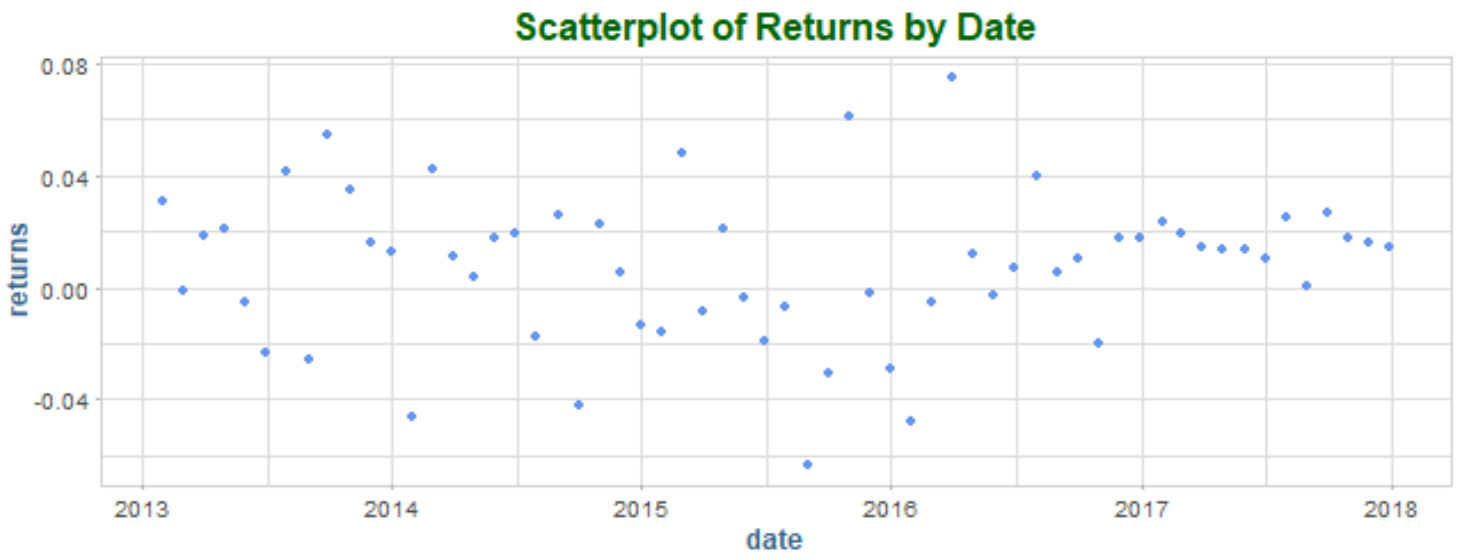
```
  tq_performance(Ra = returns,
                 Rb = NULL,
                 performance_fun = table.Stats) %>%
  select(Stdev) %>%
  mutate(tq_sd = round(Stdev, 4) * 100)

head(portfolio_sd_tidyquant_builtin_percent)
```

```
# A tibble: 1 x 2
   Stdev tq_sd
   <dbl> <dbl>
1 0.0266  2.66
```

```
# SD, PerformanceAnalytics

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)
```

```
# A tibble: 1 x 5
  dplyr dplyr_byhand xts_builtin[,1] matrix[,"standard deviation"]    tq
  <dbl>        <dbl>           <dbl>                         <dbl> <dbl>
1  2.66         2.66            2.66                          2.66  2.66
```

```
# Portfolio returns

portfolio_returns_dplyr_byhand %>%
  ggplot(aes(x = date, 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))
```



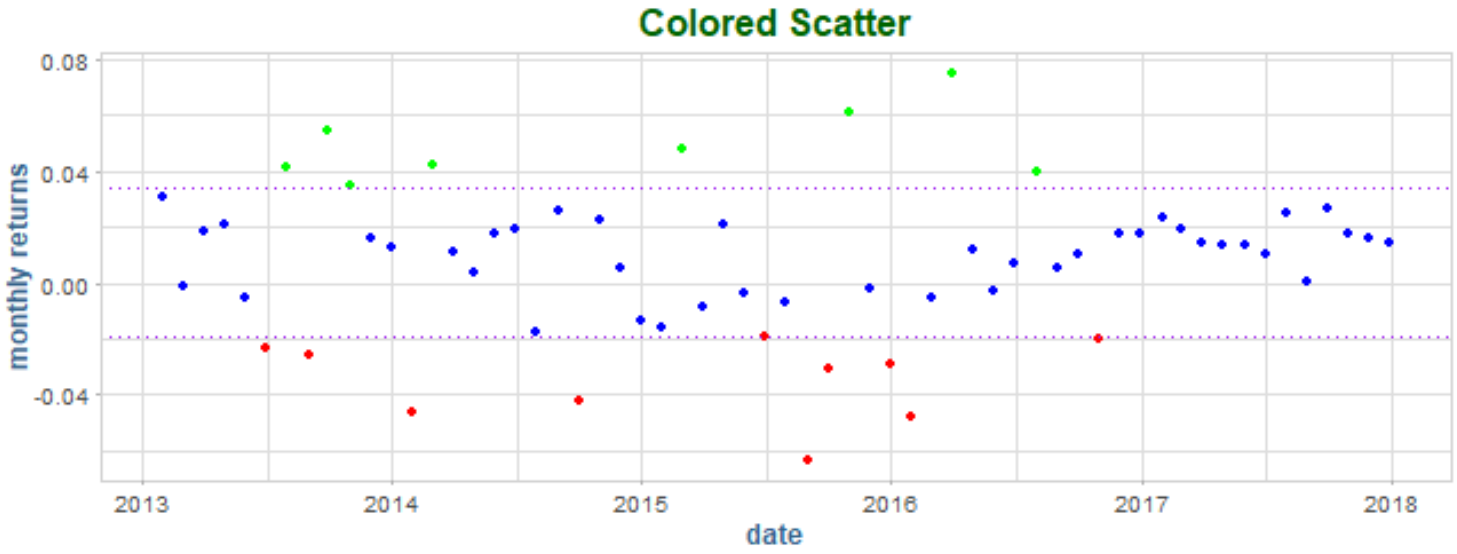

```
sd_plot <-
  sd(portfolio_returns_tq_rebalanced_monthly$returns)

mean_plot <-
  mean(portfolio_returns_tq_rebalanced_monthly$returns)

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 = date)) +

  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", y = "monthly returns") +
  scale_x_date(breaks = pretty_breaks(n = 8)) +
  theme(plot.title = element_text(hjust = 0.5))
```

```
Warning: Removed 50 rows containing missing values (geom_point).
```

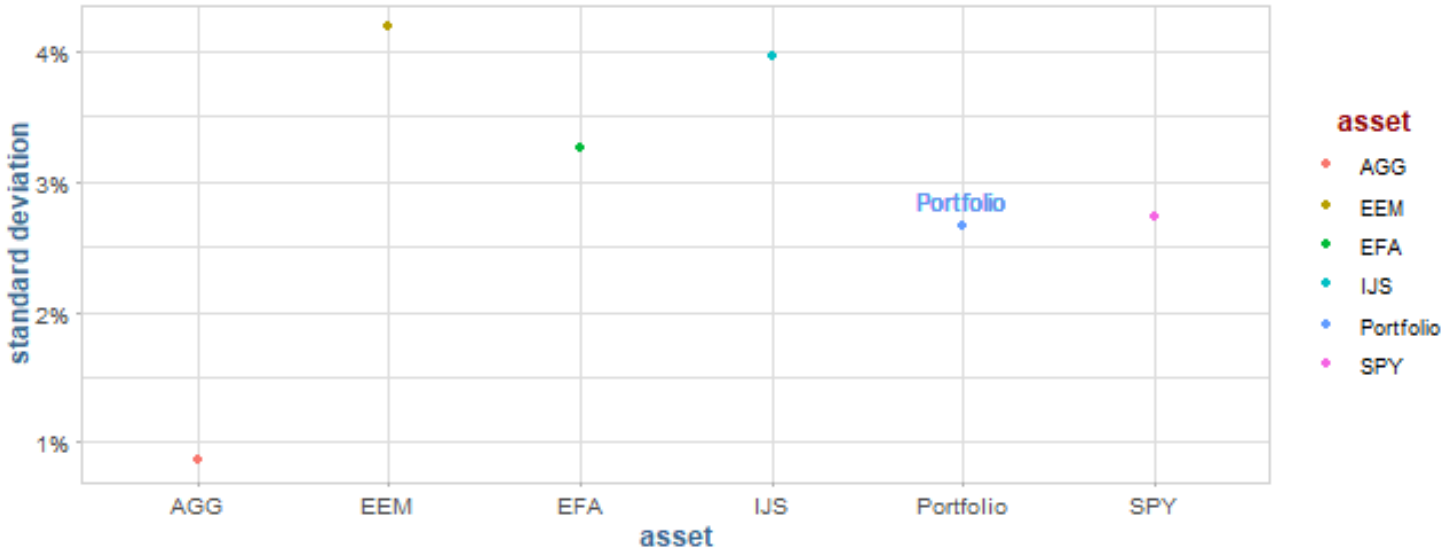
```
Warning: Removed 52 rows containing missing values (geom_point).
```

```
Warning: Removed 18 rows containing missing values (geom_point).
```



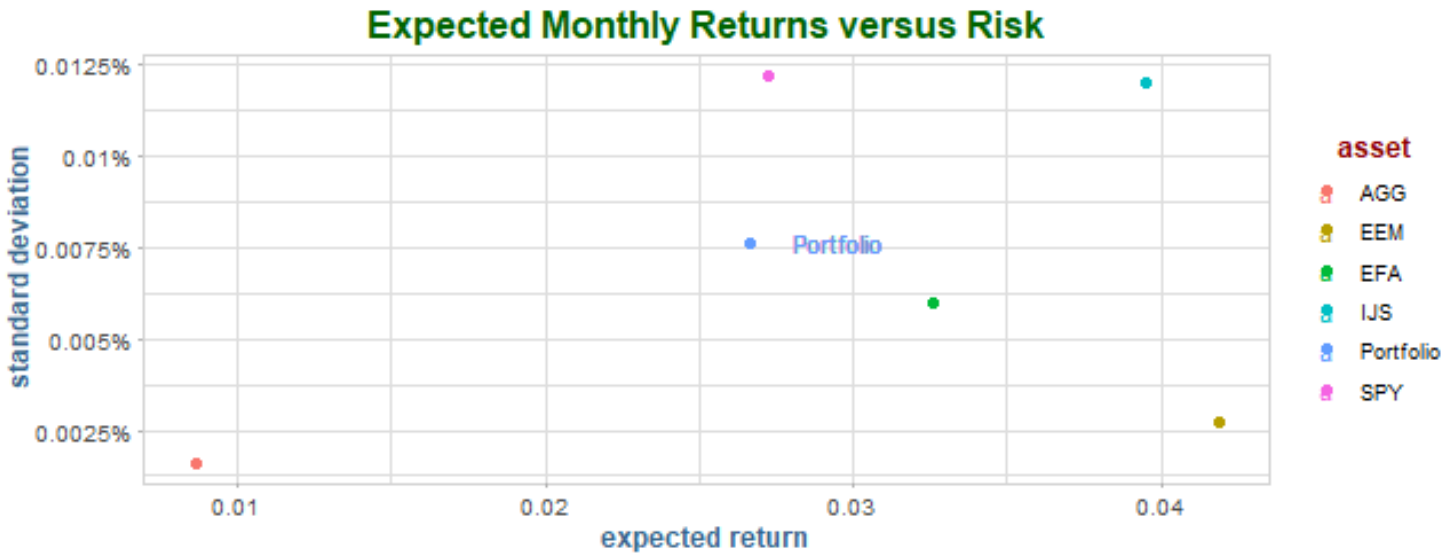

```
# By asset

asset_returns_long %>%
  group_by(asset) %>%
  summarize(sd = 100 * sd(returns)) %>%
  add_row(asset = "Portfolio",
          sd = portfolio_sd_tidy_builtin_percent$dplyr) %>%
  ggplot(aes(x = asset,
             y = sd,
             colour = asset)) +
  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")
```




```
asset_returns_long %>%
  group_by(asset) %>%
  summarize(expected_return = mean(returns),
            stand_dev = sd(returns)) %>%
  add_row(asset = "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,
             colour = asset)) +
  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")
  ) +
  xlab("expected return") +
```

```
  ylab("standard deviation") +
  ggtitle("Expected Monthly Returns versus Risk") +
  scale_y_continuous(labels = function(x) paste0(x, "%")) +
  theme_update(plot.title = element_text(hjust = 0.5))
```




```
# Portfolio returns, xts.

portfolio_returns_xts_rebalanced_monthly <-
  Return.portfolio(asset_returns_xts,
                   weights = w,
                   rebalance_on = "months") %>%
  `colnames<-`("returns")

head(portfolio_returns_xts_rebalanced_monthly, 3)
```

```
                returns
2013-01-31  0.0308487678
2013-02-28 -0.0008696533
2013-03-31  0.0186624177
```

```
# Rolling StdDev

window <- 24

# rolling sd, xts
port_rolling_sd_xts <-
  rollapply(portfolio_returns_xts_rebalanced_monthly,
            FUN = sd,
            width = window) %>%
  # omit the 23 months for which there is no rolling 24 month sd
```

```
  na.omit() %>%
  `colnames<-`("rolling_sd")

tail(port_rolling_sd_xts, 3)
```

```
           rolling_sd
2017-10-31 0.02339125
2017-11-30 0.02328079
2017-12-31 0.02169450
```

```
# rolling sd, tidyverse

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

tail(port_rolling_sd_tidy_does_not_work, 3)
```

```
# A tibble: 3 x 2
  date       rolling_sd
  <date>          <dbl>
1 2016-10-31     0.0234
2 2016-11-30     0.0233
3 2016-12-31     0.0217
```

```
# rolling sd, tibbletime

sd_roll_24 <- rollify(sd, window = window)

port_rolling_sd_tidy_tibbletime <-
  portfolio_returns_tq_rebalanced_monthly %>%
  as_tbl_time(index = date) %>%
  mutate(sd = sd_roll_24(returns)) %>%
  select(-returns) %>%
  na.omit()

tail(port_rolling_sd_tidy_tibbletime, 3)
```

```
# A time tibble: 3 x 2
# Index: date
  date           sd
```

```
  <date>      <dbl>
1 2017-10-31 0.0234
2 2017-11-30 0.0233
3 2017-12-31 0.0217
```

```
# rolling sd, tidyquant

port_rolling_sd_tq <-
  portfolio_returns_tq_rebalanced_monthly %>%
  tq_mutate(mutate_fun = rollapply,
            width = window,
            FUN = sd,
            col_rename = "rolling_sd") %>%
  select(date, rolling_sd) %>%
  na.omit()

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
# Index: date
  date           sd  sd_tq sd_xts[,"rolling_sd"]
  <date>      <dbl>  <dbl> <xts>
1 2017-10-31 0.0234 0.0234 0.0234
2 2017-11-30 0.0233 0.0233 0.0233
3 2017-12-31 0.0217 0.0217 0.0217
```

```
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 = F) %>%
  hc_navigator(enabled = F) %>%
  hc_scrollbar(enabled = F) %>%
  hc_exporting(enabled = T) %>%
  hc_legend(enabled = T)
```

```
PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed,
```

```
sd_roll_24 <- rollify(sd, window = window)

port_rolling_sd_tidy_tibbletime <-
  portfolio_returns_tq_rebalanced_monthly %>%
  as_tbl_time(index = date) %>%
  mutate(sd = sd_roll_24(returns)) %>%
  select(-returns) %>%
  na.omit()

tail(port_rolling_sd_tidy_tibbletime, 3)
```

```
# A time tibble: 3 x 2
# Index: date
  date           sd
  <date>      <dbl>
1 2017-10-31 0.0234
2 2017-11-30 0.0233
3 2017-12-31 0.0217
```

```
# rolling sd, tidyquant

port_rolling_sd_tq <-
  portfolio_returns_tq_rebalanced_monthly %>%
  tq_mutate(mutate_fun = rollapply,
            width = window,
            FUN = sd,
            col_rename = "rolling_sd") %>%
  select(date, rolling_sd) %>%
  na.omit()

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
# Index: date
  date           sd  sd_tq sd_xts[,"rolling_sd"]
  <date>      <dbl>  <dbl> <xts>
1 2017-10-31 0.0234 0.0234 0.0234
2 2017-11-30 0.0233 0.0233 0.0233
3 2017-12-31 0.0217 0.0217 0.0217
```

```
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 = F) %>%
  hc_navigator(enabled = F) %>%
  hc_scrollbar(enabled = F) %>%
  hc_exporting(enabled = T) %>%
  hc_legend(enabled = T)
```

```
# rolling sd vis, ggplot

port_rolling_sd_tq %>%
  ggplot(aes(x = date)) +
  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))
```