R Notebook

```
require(dplyr)
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
require(xts)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
##
## Attaching package: 'xts'
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, last
```

```
require(rugarch)
```

```
## Loading required package: rugarch
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'rugarch'
```

```
## The following object is masked from 'package:stats':
##
##     sigma
```

```
require(PerformanceAnalytics)
```

```
## Loading required package: PerformanceAnalytics
```

```
##
## Attaching package: 'PerformanceAnalytics'

## The following object is masked from 'package:graphics':
##
##     legend
```

```
require(quantmod)
```

```
## Loading required package: quantmod

## Loading required package: TTR

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```
library(skewt)
```

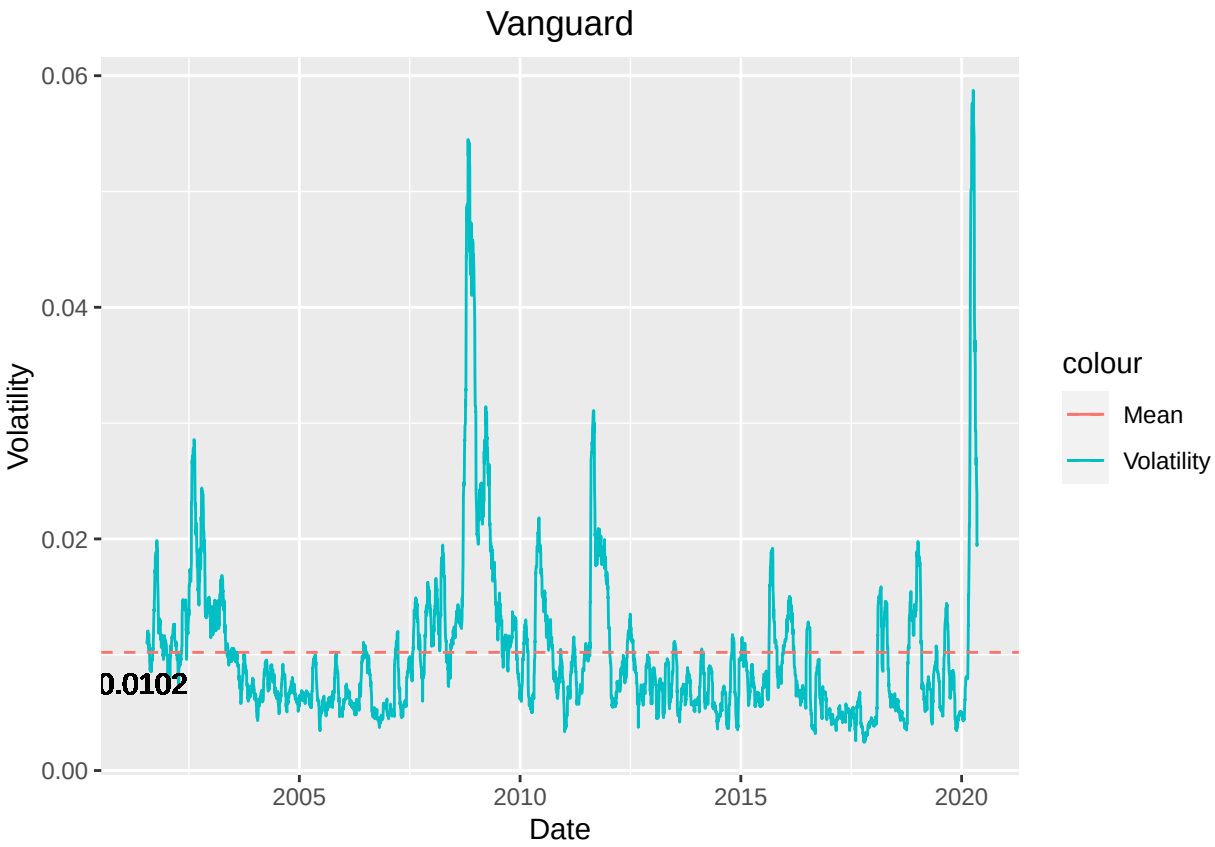
```
read_funds <- function(lf) {
  dfs <- list()
 browser()
  for (f in 1:length(lf)) {
      cond <- substr(lf[f], nchar(lf[f])-3, nchar(lf[f])) == '.csv'
      if (cond) {
      temp <- data.frame(read.csv(lf[f], stringsAsFactors = FALSE))
      temp$Date <- as.Date(temp$Date)
      temp$Close <- as.numeric(temp$Close)
      dfs[[f]] <- temp[,c(1,5)]
    }
  }

  return(dfs)
}

funds_names <- c("Vanguard", "Blackrock", "Statestreet",
                     "JPmorgan", "Bankmellon", "Allianz")
```

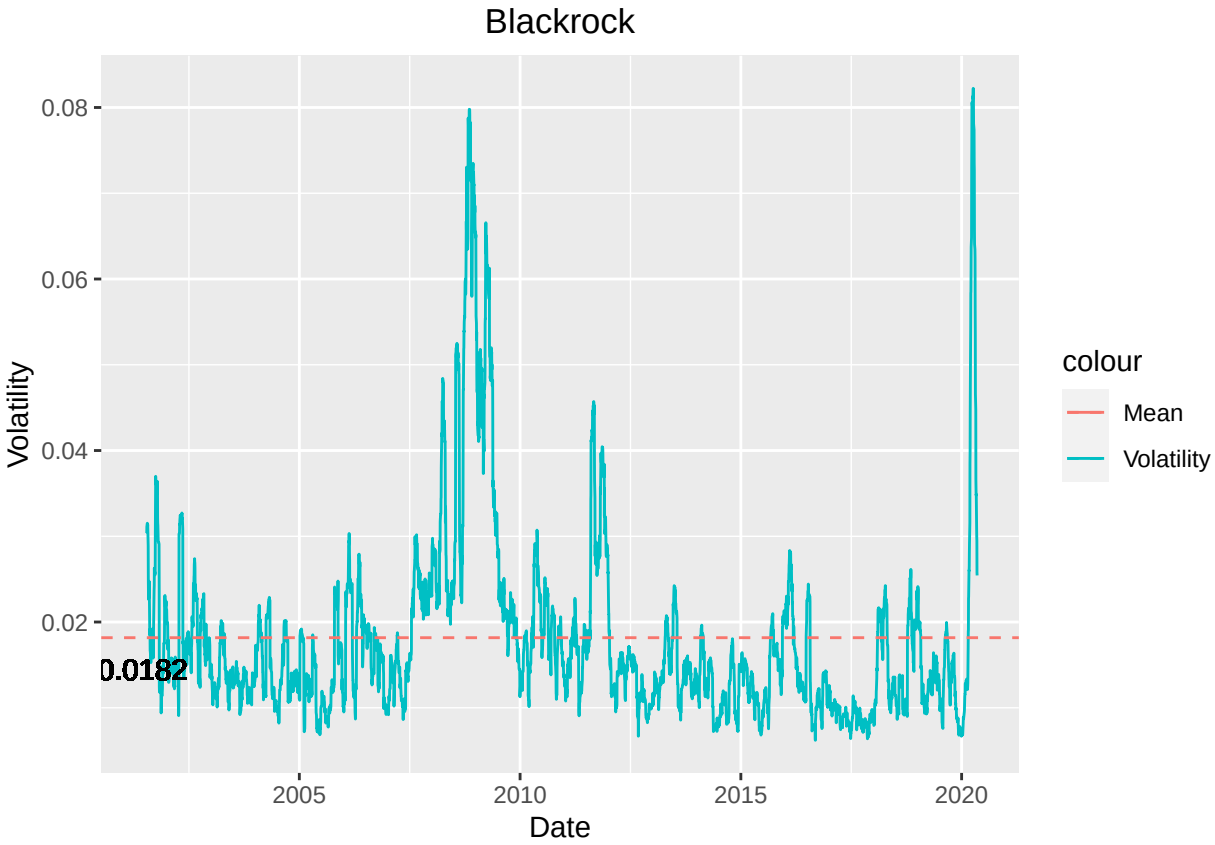
```
# Data loading
df <- read.csv('Funds.csv', stringsAsFactors = F)
df <- df[,-1]
df$Date <- as.Date(df$Date)

# Transforming data from df to xts
funds <- xts(df[,2:ncol(df)], order.by = df$Date)

rm(df)

# Subseting original data for April of 2020 year
funds_red <- funds['/202004']
```

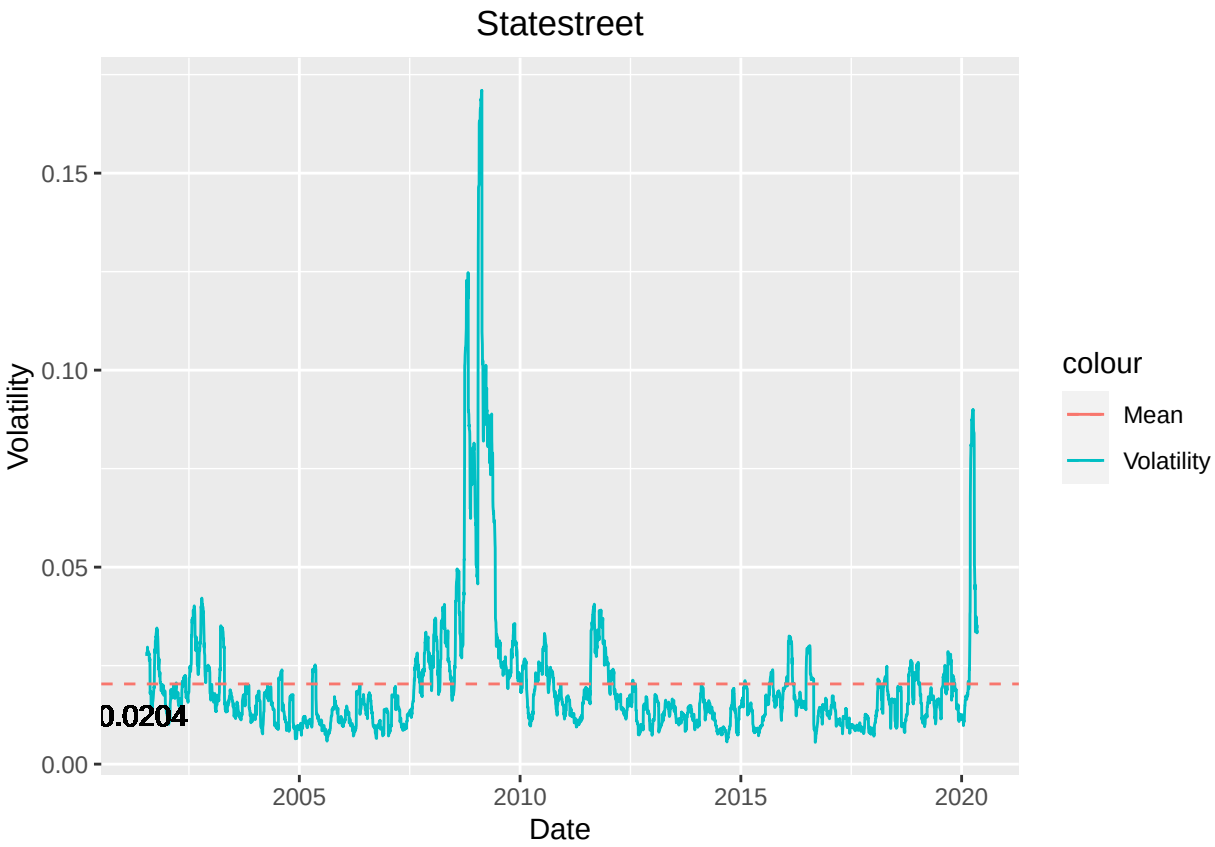
```
get_volatiles <- function(funds, width = 22, time_scale = 1, funds_names) {
    vol_df <- data.frame()

    for (i in 1:ncol(funds)) {
      fund <- funds[,i]
```

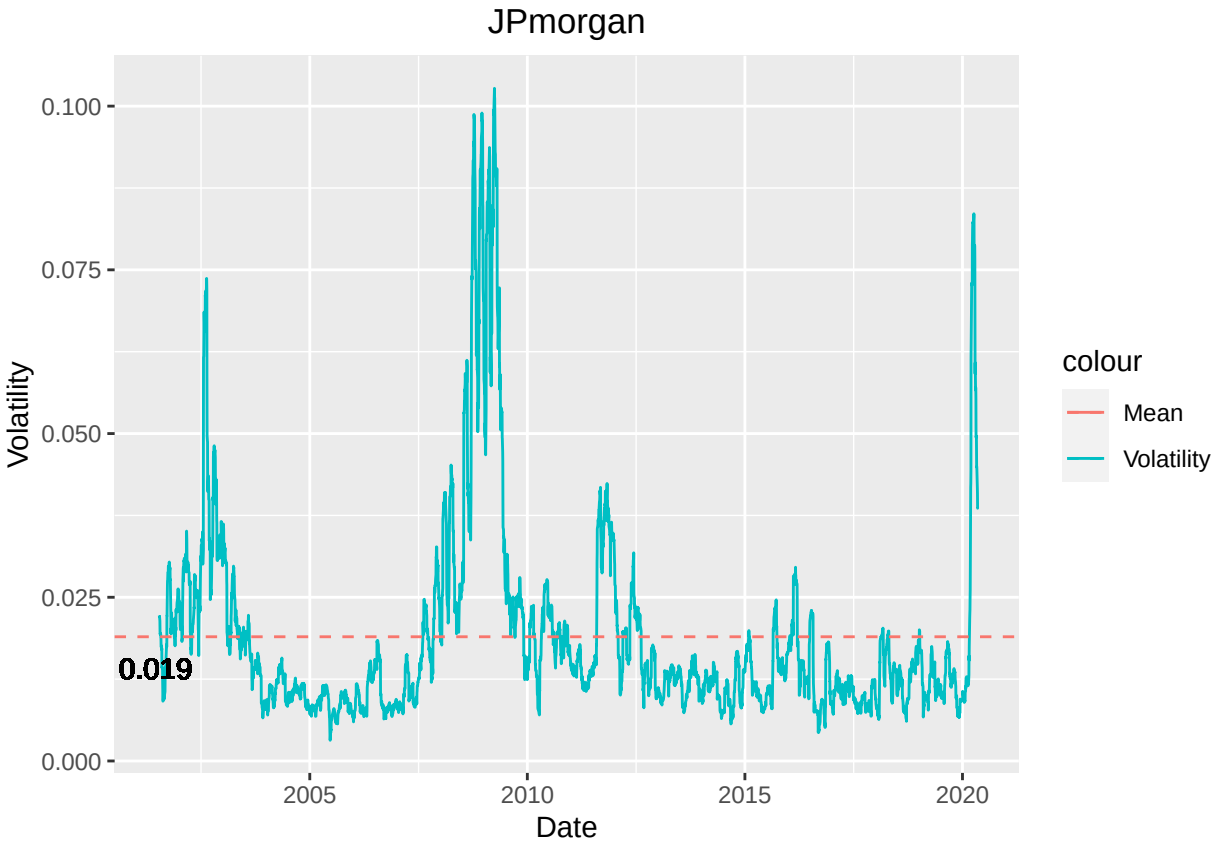
```
      temp <- rollapply(data = fund, width = 22, FUN = 'sd.annualized', scale = time_scale)

      if (nrow(vol_df) == 0) {
        vol_df <- temp
      } else {
        vol_df <- cbind(vol_df, temp)
      }

    }
    names(vol_df) <- funds_names
    return(vol_df)
}

visualize_funds_lines <- function(funds, y_axis_label = 'Volatility') {

  for (i in 1:ncol(funds)) {
    temp <- funds[,i]
    temp_mean <- mean(temp, na.rm = TRUE)

    p <- ggplot(temp, aes(x = index(temp), y = temp)) +
    geom_line(aes(color = 'Volatility')) +
    geom_hline(aes(yintercept = temp_mean, color = 'Mean'),
               size=.5, linetype='dashed') +
    geom_text( aes( min(index(temp)) , temp_mean, label = round(temp_mean, 4), vjust = 2)) +
    labs(x = 'Date', y = y_axis_label, title = names(funds)[i]) +
    theme(plot.title = element_text(hjust = 0.5))

    suppressMessages(suppressWarnings(print(p)))
  }
}

visualize_funds_hist <- function(funds, x_axis_label = 'Return') {
  for (i in 1:ncol(funds)) {
      temp <- funds[,i]
      title <- paste(funds_names[i], 'returns')

      chart.Histogram(temp,
                methods = c('add.normal', 'add.density'),
                main = title)

      temp <- (temp - mean(temp, na.rm = T))/sd(temp, na.rm = T)
      title <- paste('Standardized', title)

      chart.Histogram(temp,
                      methods = c('add.normal', 'add.density'),
                      main = title)
  }
}
```

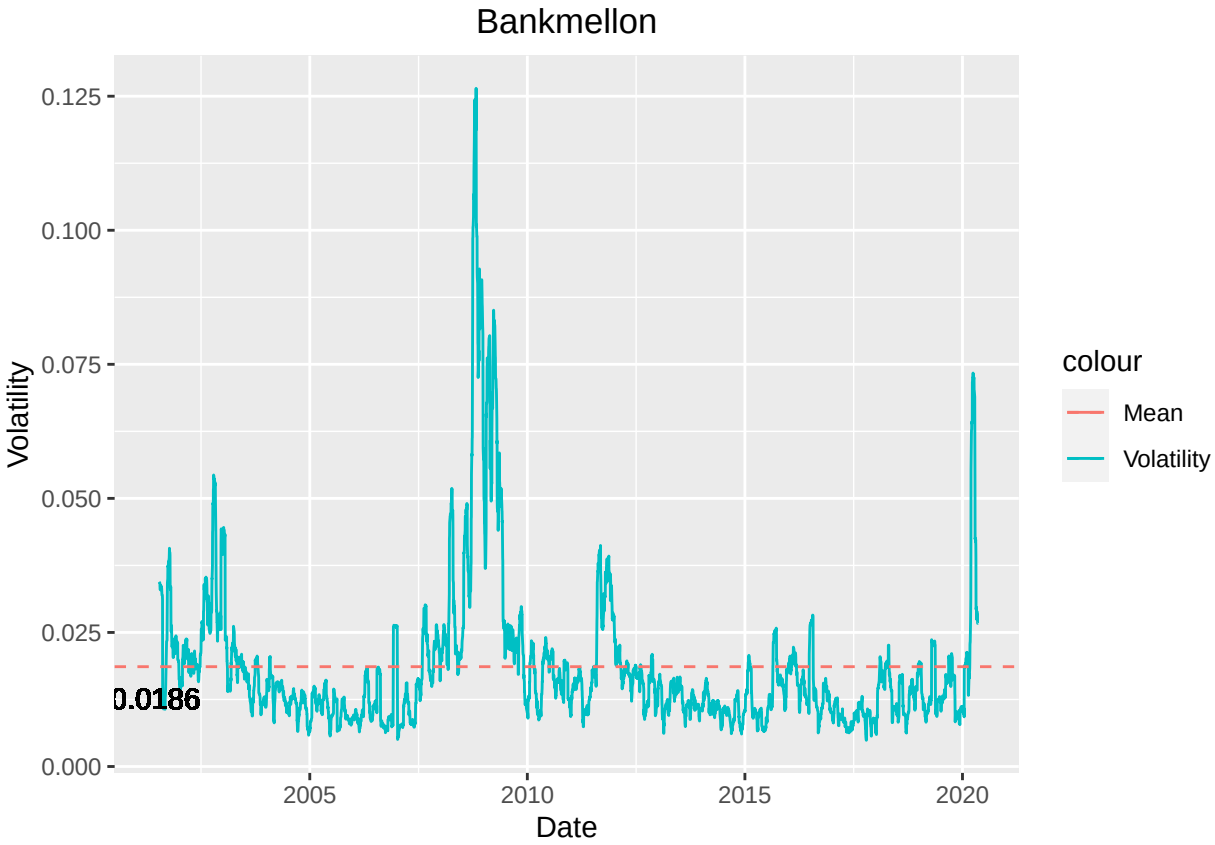
```
vol_df <- get_volatiles(funds, funds_names =  funds_names)

visualize_funds_lines(vol_df)
```

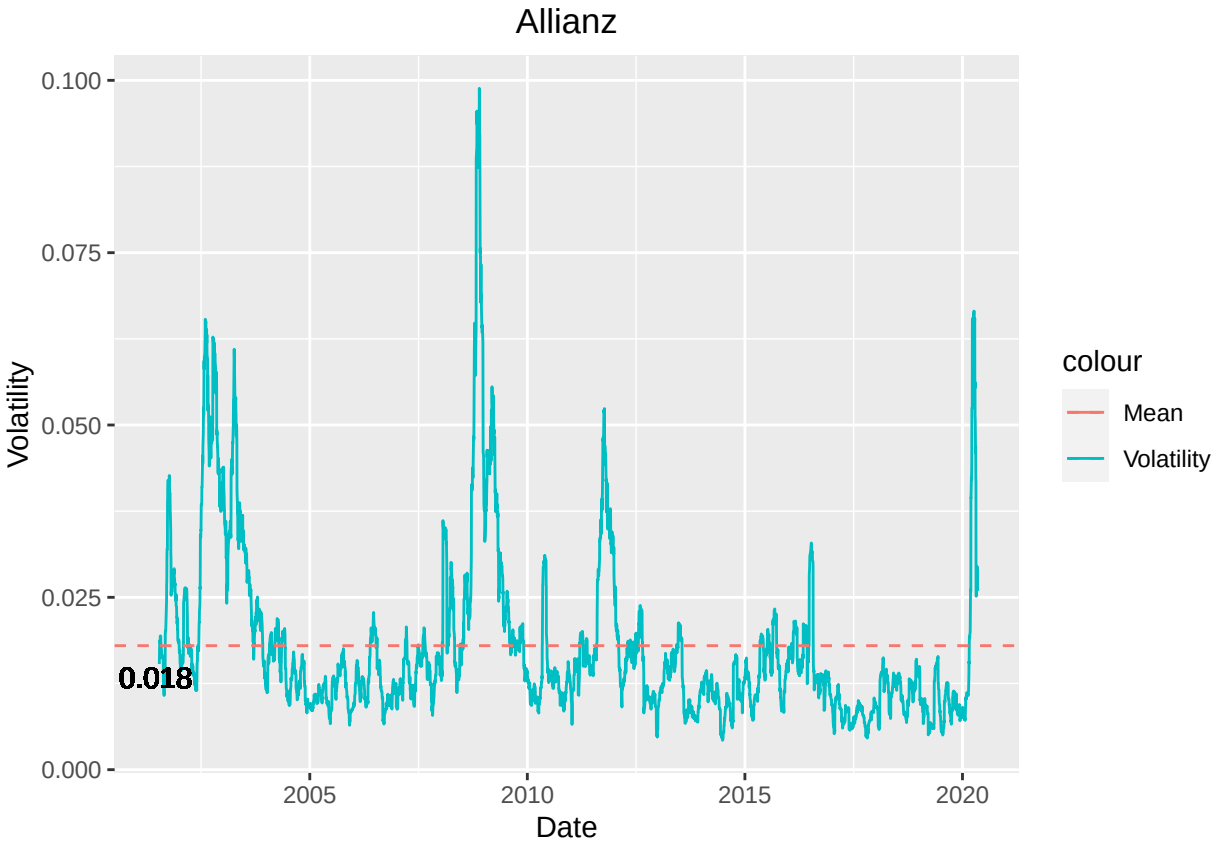

Vanguard
0.06
0.04
0.02
0.00
Volatility
0.0102
2005
2010
2015
2020
Date
colour
Mean
Volatility


Blackrock
Volatility
0.08
0.06
0.04
0.02
0.0182
2005
2010
2015
2020
Date
colour
Mean
Volatility

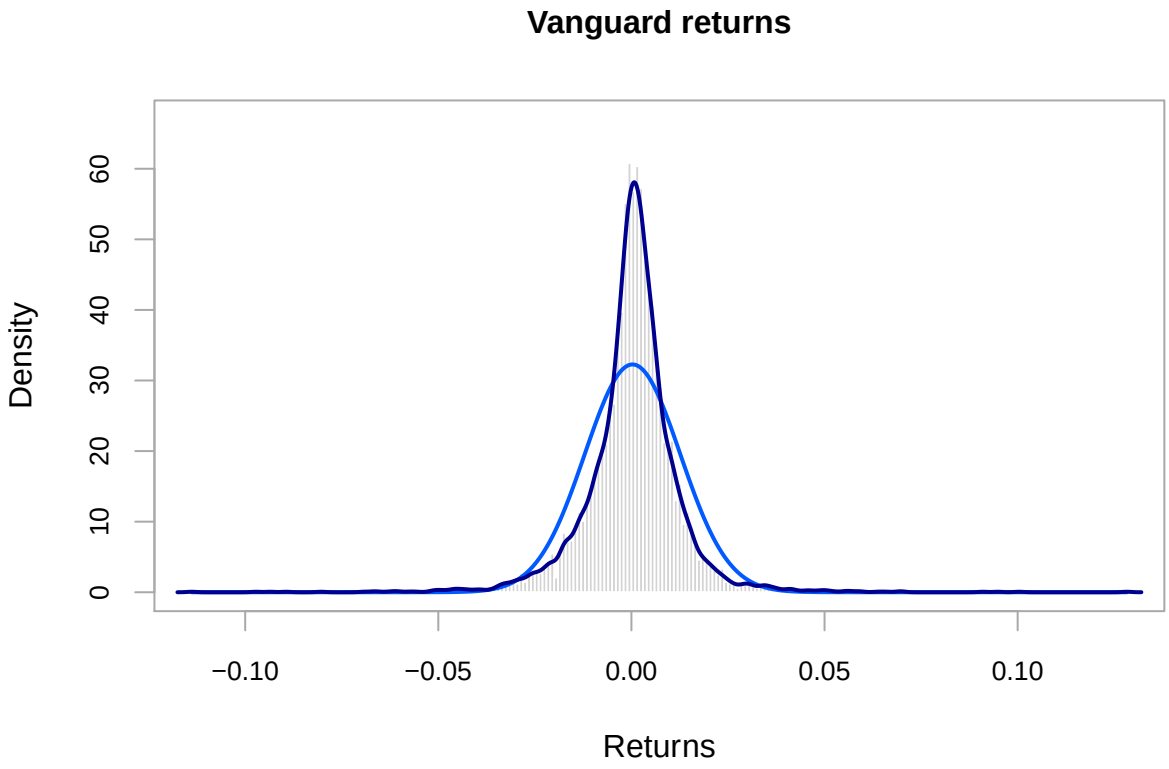

Statestreet
Volatility
0.15
0.10
0.05
0.00
0.0204
2005
2010
2015
2020
Date
colour
Mean
Volatility

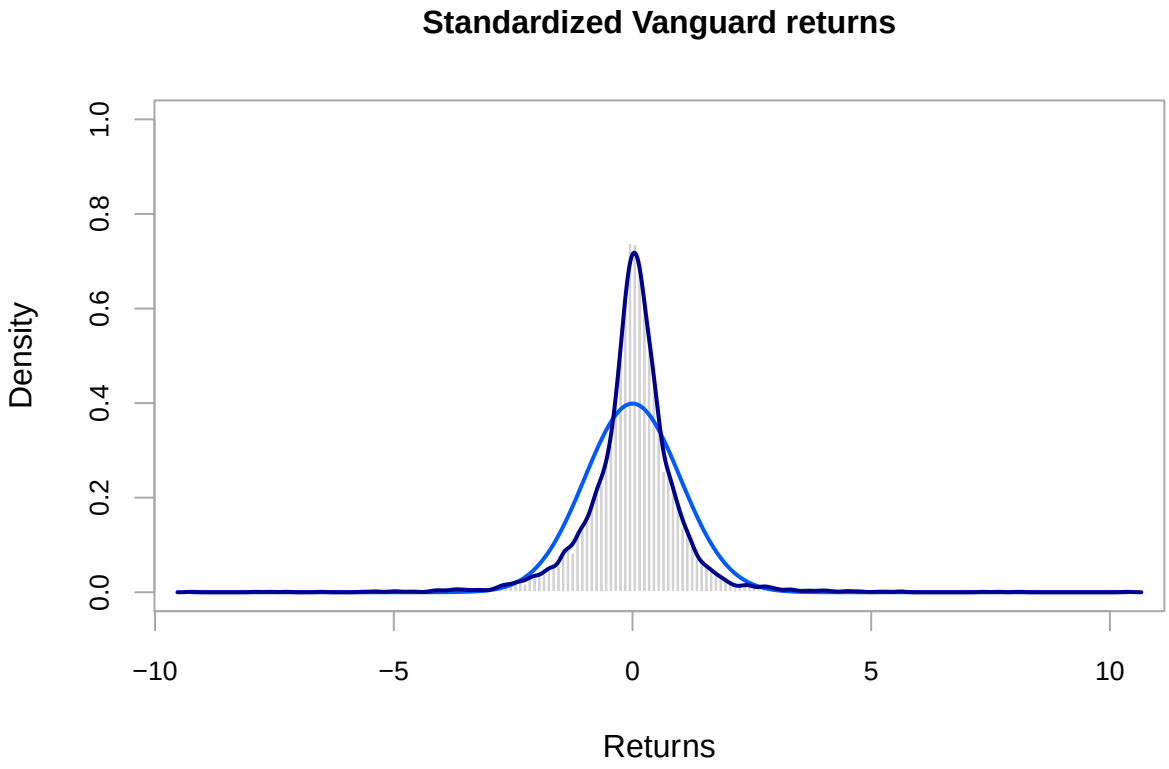

JPmorgan
0.100
0.075
0.050
0.025
0.000
Volatility
0.019
2005
2010
2015
2020
Date
colour
Mean
Volatility

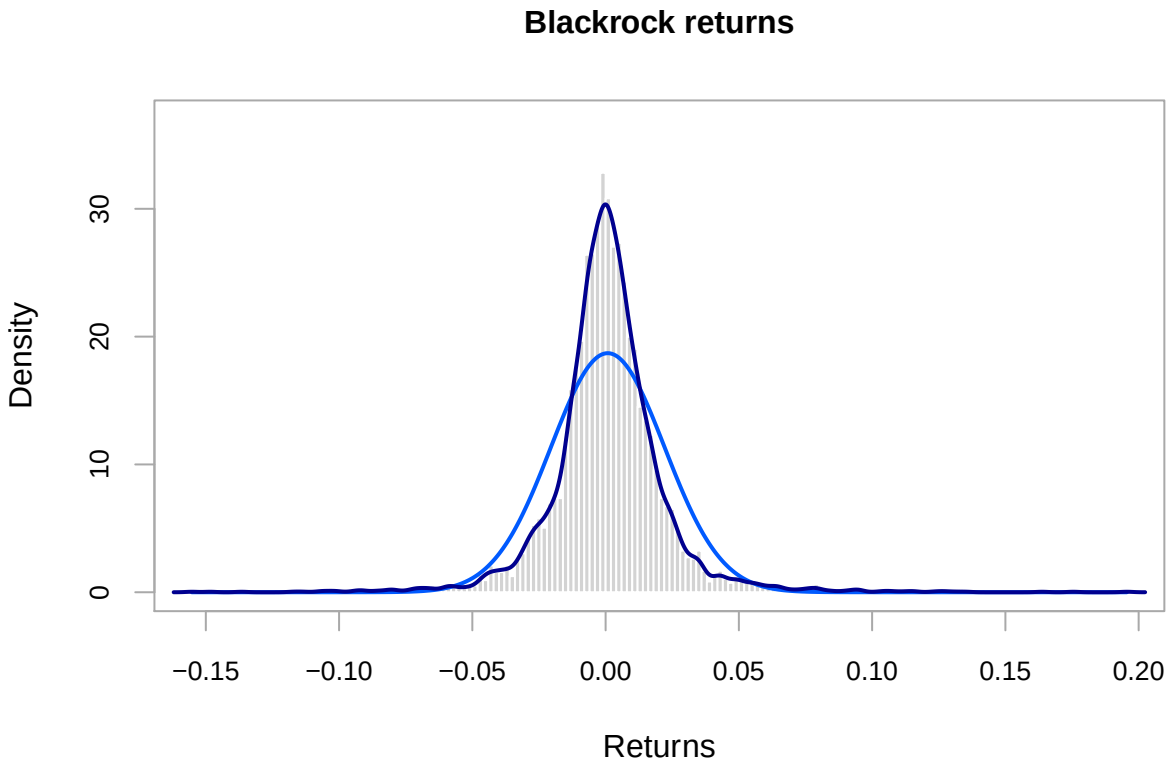

Bankmellon
Volatility
0.125
0.100
0.075
0.050
0.025
0.000
0.0186
2005
2010
2015
2020
Date
colour
Mean
Volatility

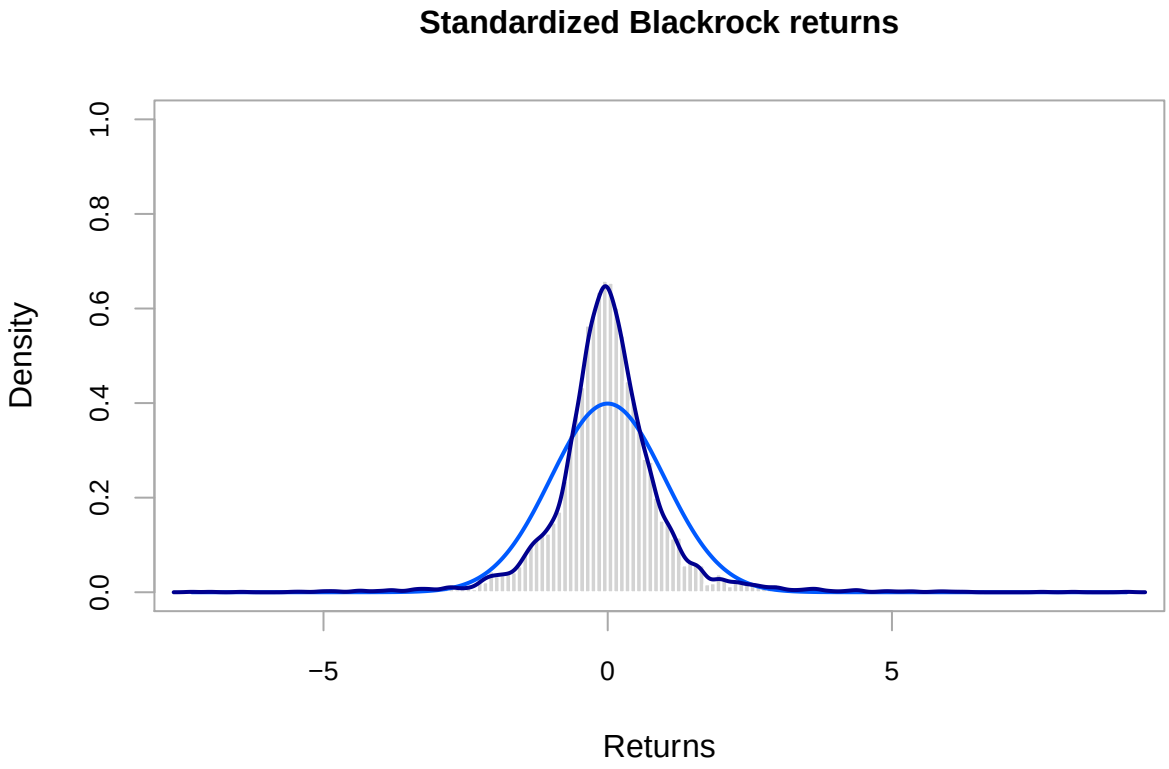


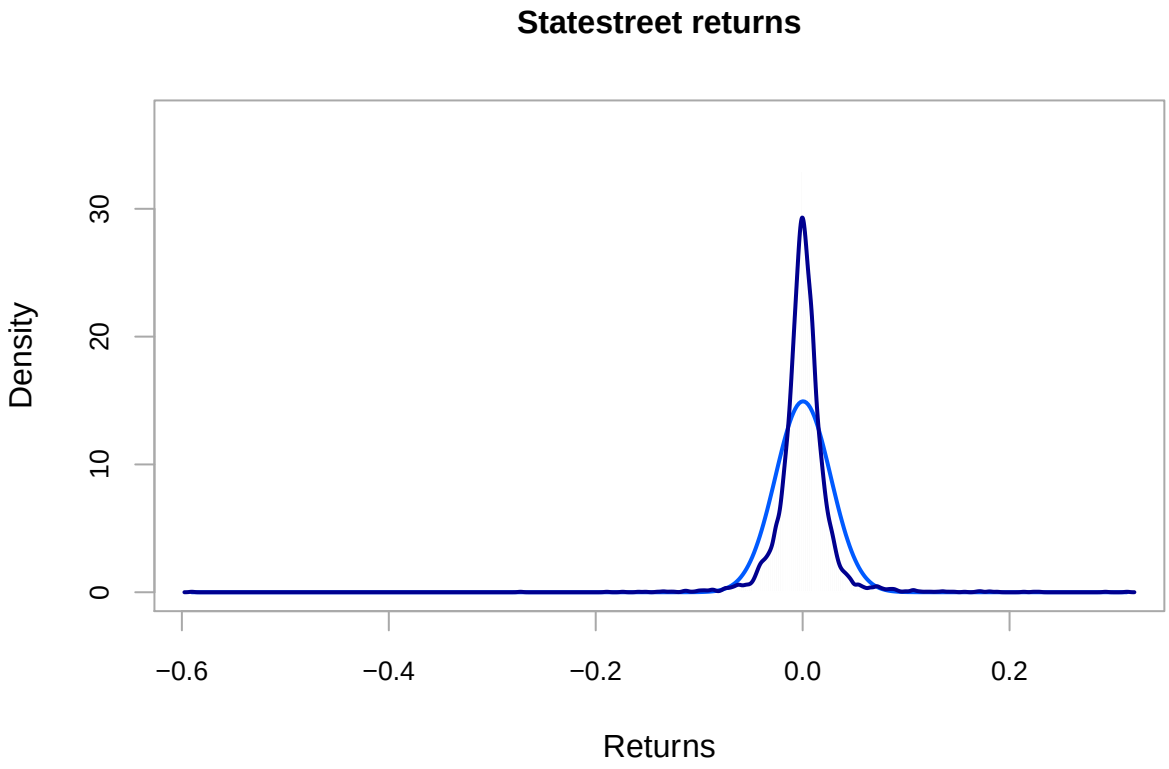

```
visualize_funds_hist(funds)
```

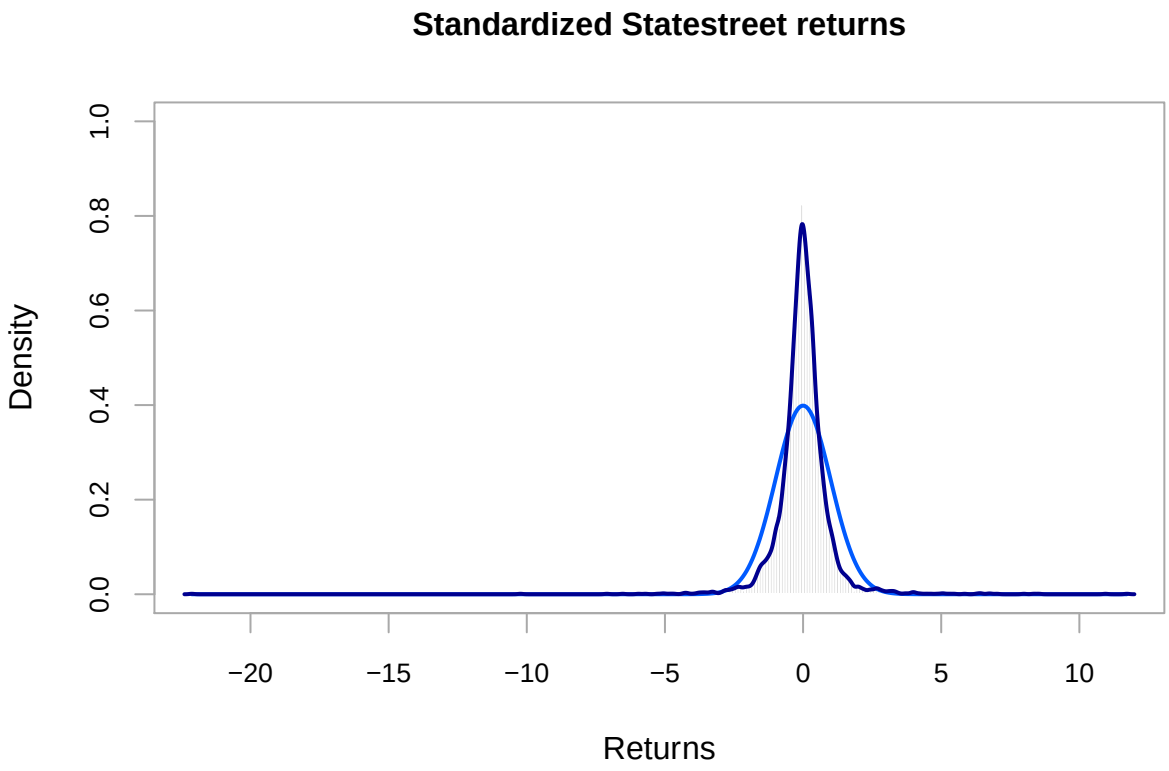

Vanguard returns
Density
0
10
20
30
40
50
60
−0.10
−0.05
0.00
0.05
0.10
Returns

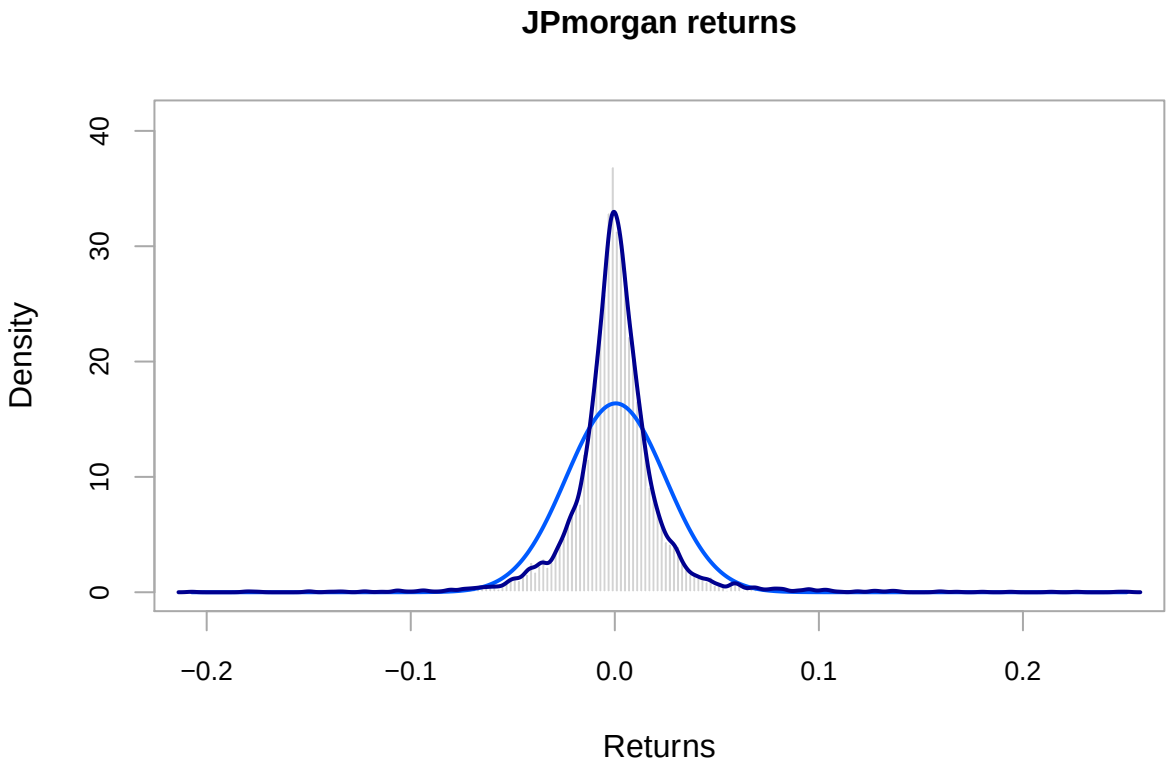

Standardized Vanguard returns
Density
0.0
0.2
0.4
0.6
0.8
1.0
−10
−5
0
5
10
Returns

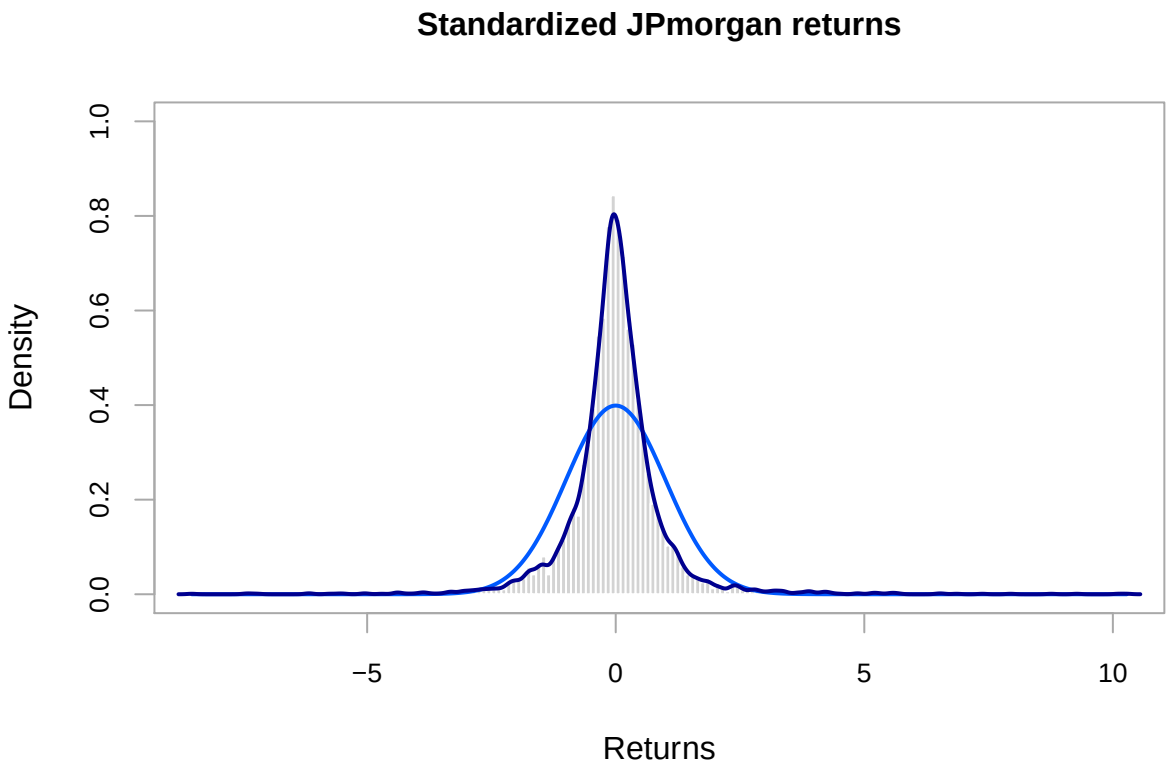

Blackrock returns
Density
0
10
20
30
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
0.20
Returns

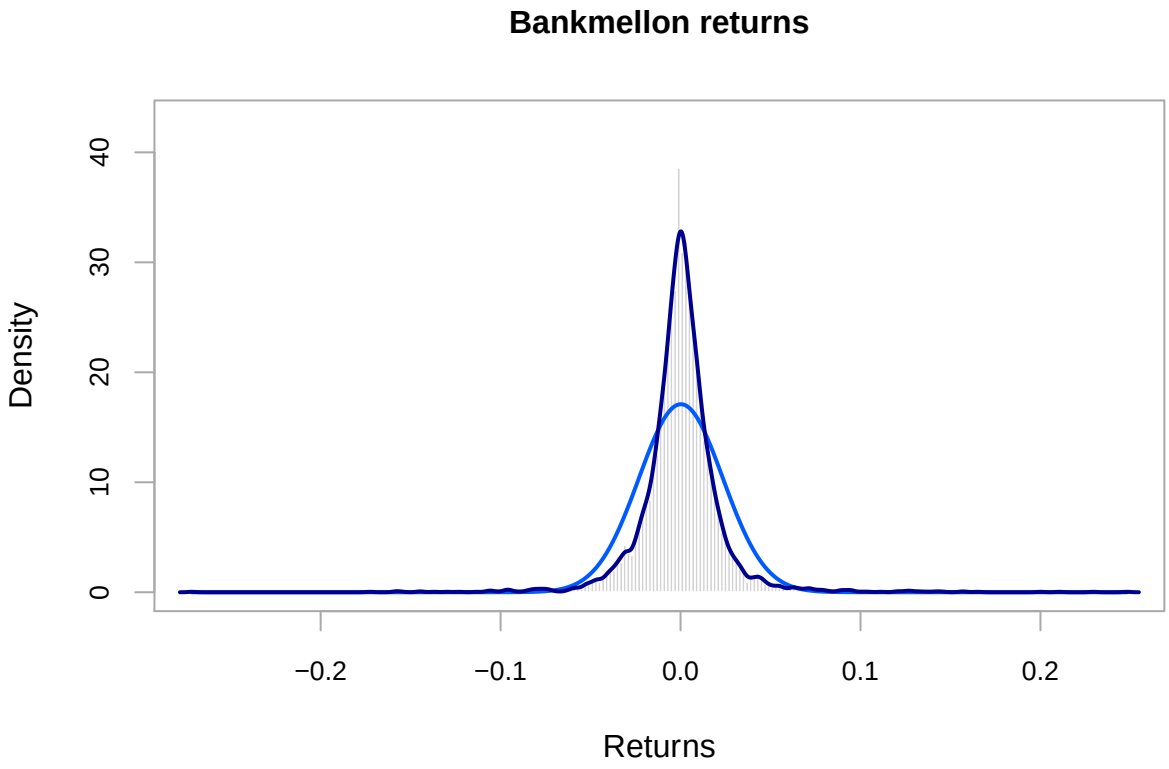

Standardized Blackrock returns
Density
0.0
0.2
0.4
0.6
0.8
1.0
−5
0
5
Returns

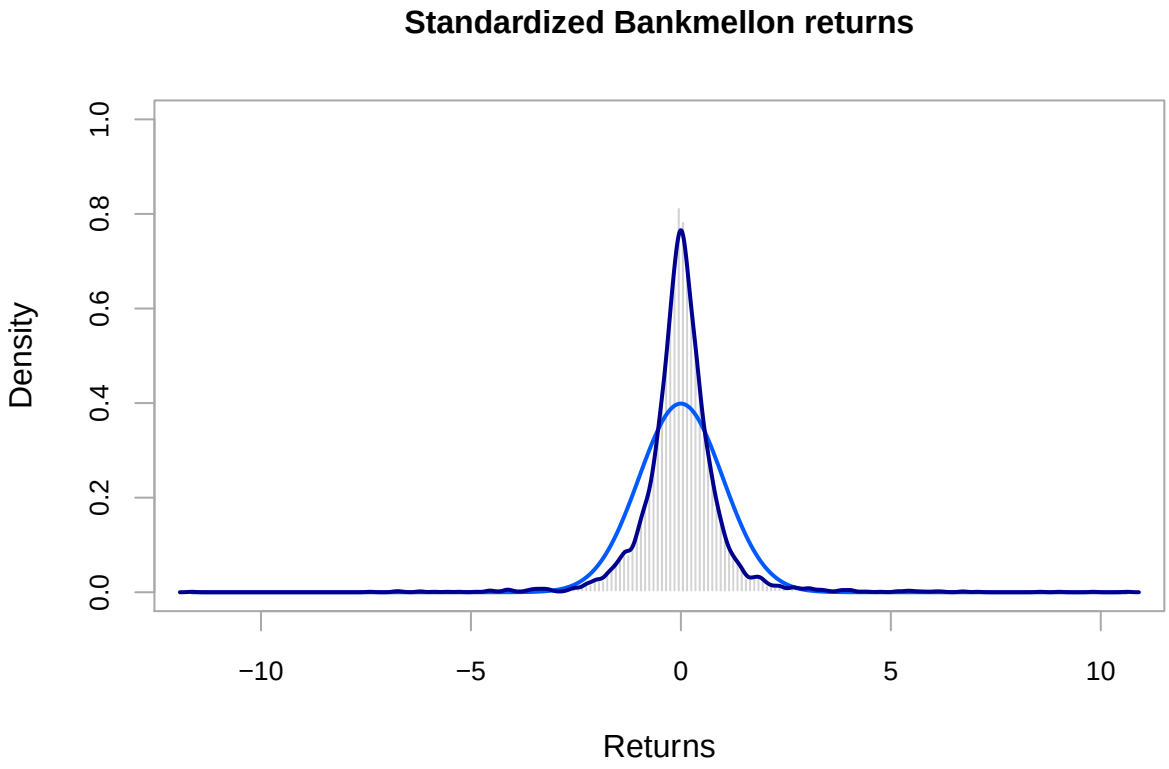

Statestreet returns
Density
0
10
20
30
−0.6
−0.4
−0.2
0.0
0.2
Returns

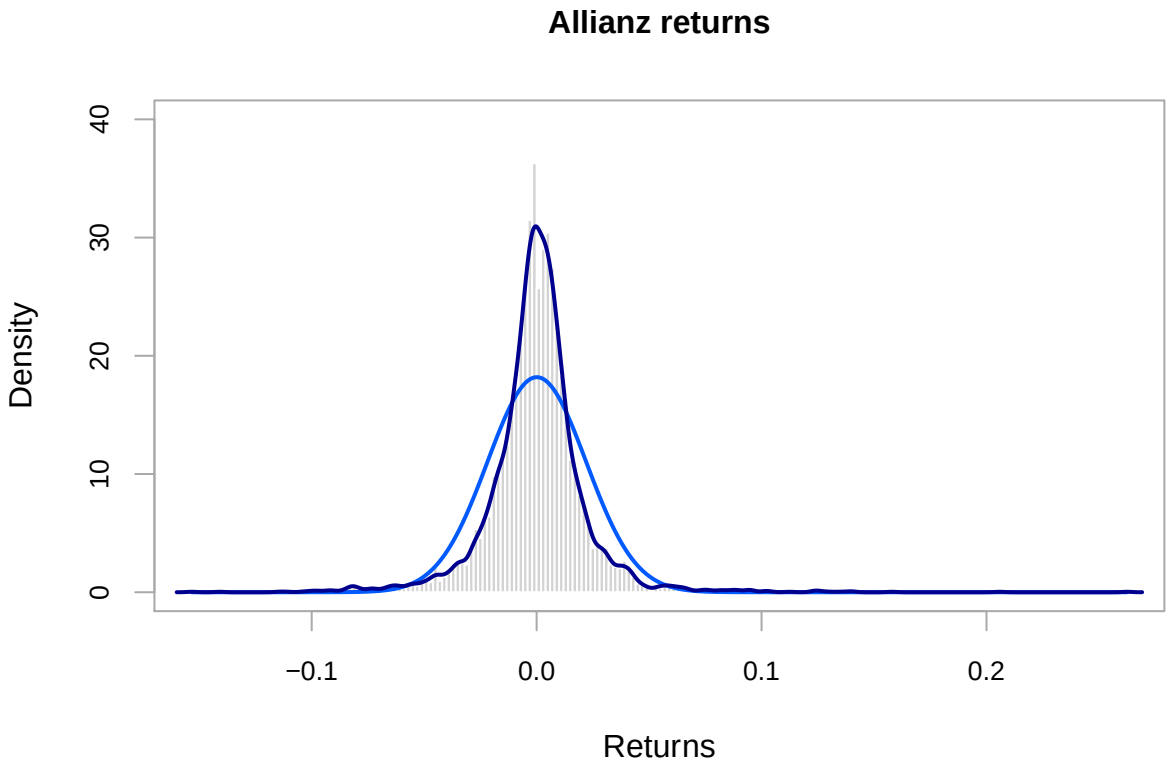

Standardized Statestreet returns
Density
0.0
0.2
0.4
0.6
0.8
1.0
−20
−15
−10
−5
0
5
10
Returns

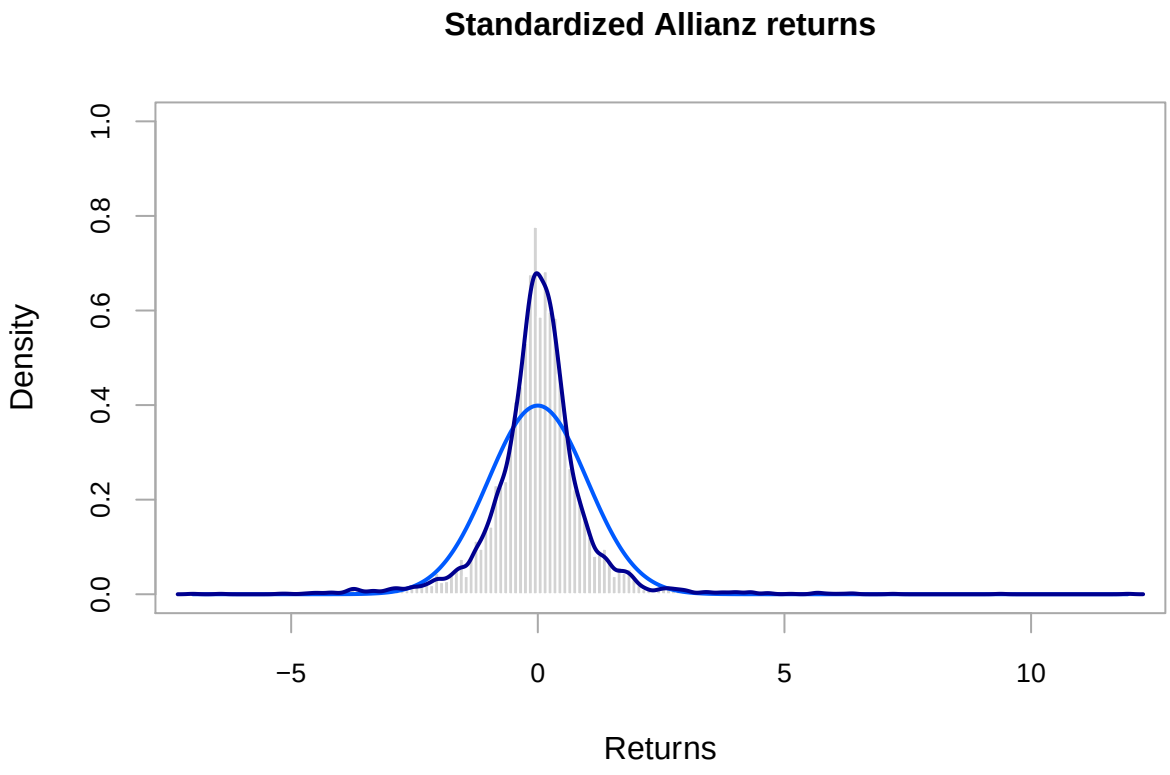

JPmorgan returns
Density
0
10
20
30
40
−0.2
−0.1
0.0
0.1
0.2
Returns


Standardized JPmorgan returns
Density
0.0
0.2
0.4
0.6
0.8
1.0
−5
0
5
10
Returns


Bankmellon returns
Density
0
10
20
30
40
−0.2
−0.1
0.0
0.1
0.2
Returns


Standardized Bankmellon returns
Density
0.0
0.2
0.4
0.6
0.8
1.0
−10
−5
0
5
10
Returns


Allianz returns
Density
0
10
20
30
40
−0.1
0.0
0.1
0.2
Returns




```
vanguard_tail_volatility <- tail(vol_df$Vanguard, 10)
```

```
plot_acf <- function(fund) {
  f_mean <- mean(fund)
  acf(abs(f_mean))
}
```

```
# In this chunk, prefix van_ - is for Vanguard company
fund <- funds$vanguard_return


# Forming set of parameters for different GARCH model
# The most important is distribution.model - we are specifying type of distribution
# Standard GARCH with normal distribution of errors
norm_garch_spec <- ugarchspec(mean.model = list(armaOrder = c(0,0)),
                        variance.model = list(model = 'sGARCH'),
                        distribution.model = 'norm')

# GJR GARCH with normal distribution of errors
norm_gjr_spec <- ugarchspec(mean.model = list(armaOrder = c(0,0)),
                        variance.model = list(model = 'gjrGARCH'),
                        distribution.model = 'norm')

# Standard GARCH with skewed Student t distribution of errors
sstd_garch_spec <- ugarchspec(mean.model = list(armaOrder = c(0,0)),
                        variance.model = list(model = 'sGARCH'),
                        distribution.model = 'sstd')
```

```
# GJR GARCH with skewed Student t distribution of errors
sstd_gjr_spec <- ugarchspec(mean.model = list(armaOrder = c(0,0)),
                        variance.model = list(model = 'gjrGARCH'),
                        distribution.model = 'sstd')
```

```
# Apply GARCH model to our data
norm_garch_fit <- ugarchfit(data = funds_red$vanguard_return,
                      spec = norm_garch_spec)

norm_gjr_fit <- ugarchfit(data = funds_red$vanguard_return,
                      spec = norm_gjr_spec)

sstd_garch_fit <- ugarchfit(data = funds_red$vanguard_return,
                      spec = sstd_garch_spec)

sstd_gjr_fit <- ugarchfit(data = funds_red$vanguard_return,
                          spec = sstd_gjr_spec)
```

```
#Coefficients
cat('\nCoefficients of', 'Standard GARCH with normal distribution of errors','\n')
```

```
##
## Coefficients of Standard GARCH with normal distribution of errors
```

```
print(round(norm_garch_fit@fit$matcoef,10))
```

```
##            Estimate   Std. Error   t value     Pr(>|t|)
## mu     0.0006660551 0.0001134153  5.872709 0.0000000043
## omega  0.0000024796 0.0000007827  3.167999 0.0015349183
## alpha1 0.1236562746 0.0098739025 12.523546 0.0000000000
## beta1  0.8564012304 0.0106465927 80.438996 0.0000000000
```

```
cat('\nRobust coefficients of', 'Standard GARCH with normal distribution of errors','\n')
```

```
##
## Robust coefficients of Standard GARCH with normal distribution of errors
```

```
print(round(norm_garch_fit@fit$robust.matcoef,10))
```

```
##            Estimate   Std. Error    t value     Pr(>|t|)
## mu     0.0006660551 0.0000945864  7.0417614 0.0000000000
## omega  0.0000024796 0.0000043291  0.5727754 0.5667967733
## alpha1 0.1236562746 0.0242583805  5.0974662 0.0000003442
## beta1  0.8564012304 0.0397101847 21.5662868 0.0000000000
```

```
cat('\nCoefficients of', 'GJR GARCH with normal distribution of errors','\n')
```

```
##
## Coefficients of GJR GARCH with normal distribution of errors
```

```
print(round(norm_gjr_fit@fit$matcoef,10))
```

```
##            Estimate   Std. Error      t value    Pr(>|t|)
## mu     0.0002599302 0.0000981960 2.647054e+00 0.008119637
## omega  0.0000024736 0.0000002385 1.037265e+01 0.000000000
## alpha1 0.0000001769 0.0033173146 5.333410e-05 0.999957445
## beta1  0.8729915992 0.0066767202 1.307516e+02 0.000000000
## gamma1 0.2067021938 0.0129895853 1.591292e+01 0.000000000
```

```
cat('\nRobust coefficients of', 'Standard GARCH with normal distribution of errors','\n')
```

```
##
## Robust coefficients of Standard GARCH with normal distribution of errors
```

```
print(round(norm_gjr_fit@fit$robust.matcoef,10))
```

```
##            Estimate   Std. Error      t value     Pr(>|t|)
## mu     0.0002599302 0.0001309420 1.985079e+00 0.0471356725
## omega  0.0000024736 0.0000005885 4.203159e+00 0.0000263216
## alpha1 0.0000001769 0.0140922669 1.255480e-05 0.9999899827
## beta1  0.8729915992 0.0072802743 1.199119e+02 0.0000000000
## gamma1 0.2067021938 0.0315575960 6.549998e+00 0.0000000001
```

```
cat('\nCoefficients of', 'Standard GARCH with skewed Student t distribution of errors','\n')
```

```
##
## Coefficients of Standard GARCH with skewed Student t distribution of errors
```

```
print(round(sstd_garch_fit@fit$matcoef,10))
```

```
##            Estimate   Std. Error    t value     Pr(>|t|)
## mu     0.0006062493 0.0001117549  5.4248140 0.0000000580
## omega  0.0000014778 0.0000018608  0.7941576 0.4271037050
## alpha1 0.1209330654 0.0313502725  3.8574805 0.0001145618
## beta1  0.8717911859 0.0295432800 29.5089504 0.0000000000
## skew   0.8888211922 0.0185481764 47.9195998 0.0000000000
## shape  7.1084616197 1.1187769377  6.3537792 0.0000000002
```

```
cat('\nRobust coefficients of', 'Standard GARCH with normal distribution of errors','\n')
```

```
##
## Robust coefficients of Standard GARCH with normal distribution of errors
```

```
print(round(sstd_garch_fit@fit$robust.matcoef,10))
```

```
##            Estimate   Std. Error    t value     Pr(>|t|)
## mu     0.0006062493 0.0001642235  3.6916119 0.0002228373
## omega  0.0000014778 0.0000155483  0.0950453 0.9242788484
## alpha1 0.1209330654 0.2496181390  0.4844723 0.6280507427
## beta1  0.8717911859 0.2373673347  3.6727513 0.0002399530
## skew   0.8888211922 0.0575212007 15.4520626 0.0000000000
## shape  7.1084616197 6.9887168178  1.0171340 0.3090896792
```

```
cat('\nCoefficients of', 'GJR GARCH with skewed Student t distribution of errors','\n')
```

```
##
## Coefficients of GJR GARCH with skewed Student t distribution of errors
```

```
print(round(sstd_gjr_fit@fit$matcoef,10))
```

```
##            Estimate   Std. Error      t value   Pr(>|t|)
## mu     0.0002495716 0.0001175544 2.123031e+00 0.03375129
## omega  0.0000020107 0.0000010712 1.876978e+00 0.06052114
## alpha1 0.0000000450 0.0127386126 3.535900e-06 0.99999718
## beta1  0.8732336384 0.0109951208 7.942010e+01 0.00000000
## gamma1 0.2246589215 0.0309933212 7.248624e+00 0.00000000
## skew   0.8546648715 0.0168886550 5.060586e+01 0.00000000
## shape  8.2593642262 0.9887244708 8.353555e+00 0.00000000
```

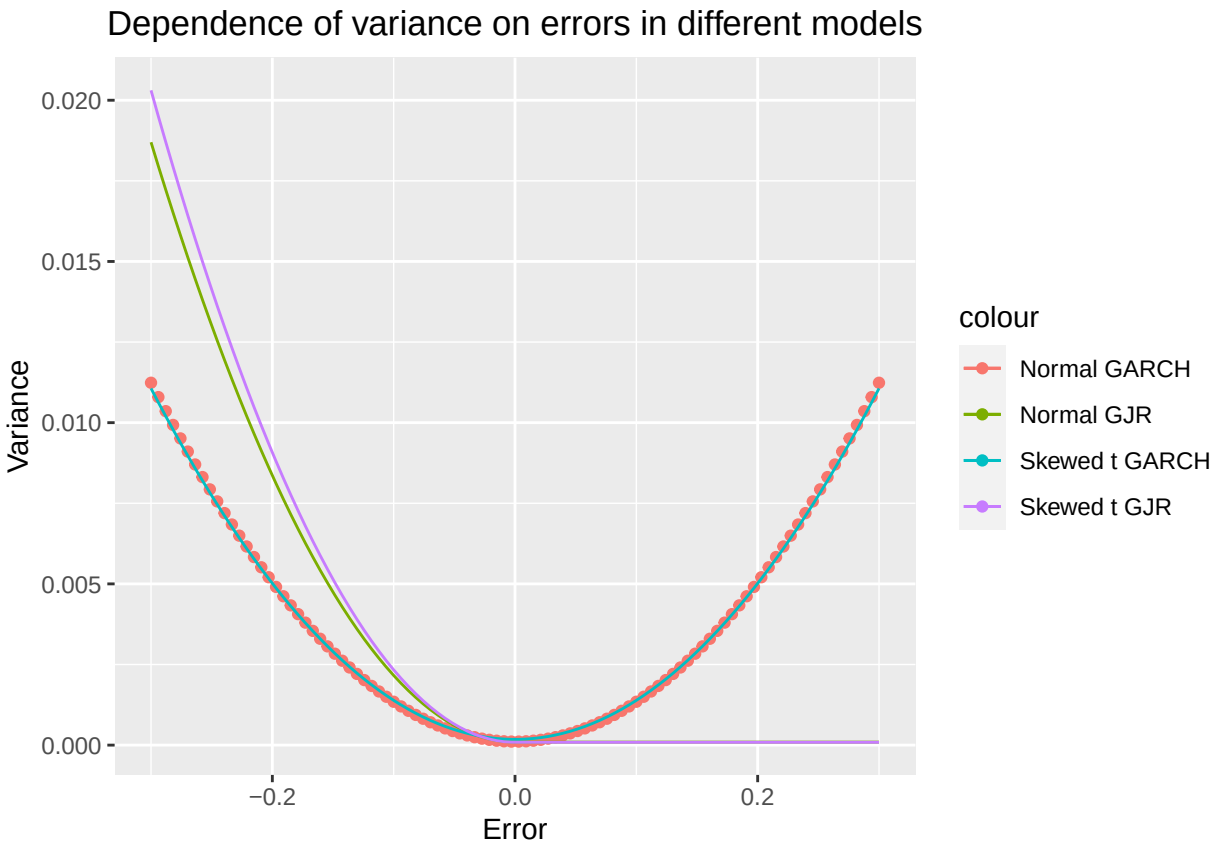
```
cat('\nRobust coefficients of', 'GJR GARCH with normal distribution of errors','\n')
```

```
##
## Robust coefficients of GJR GARCH with normal distribution of errors
```

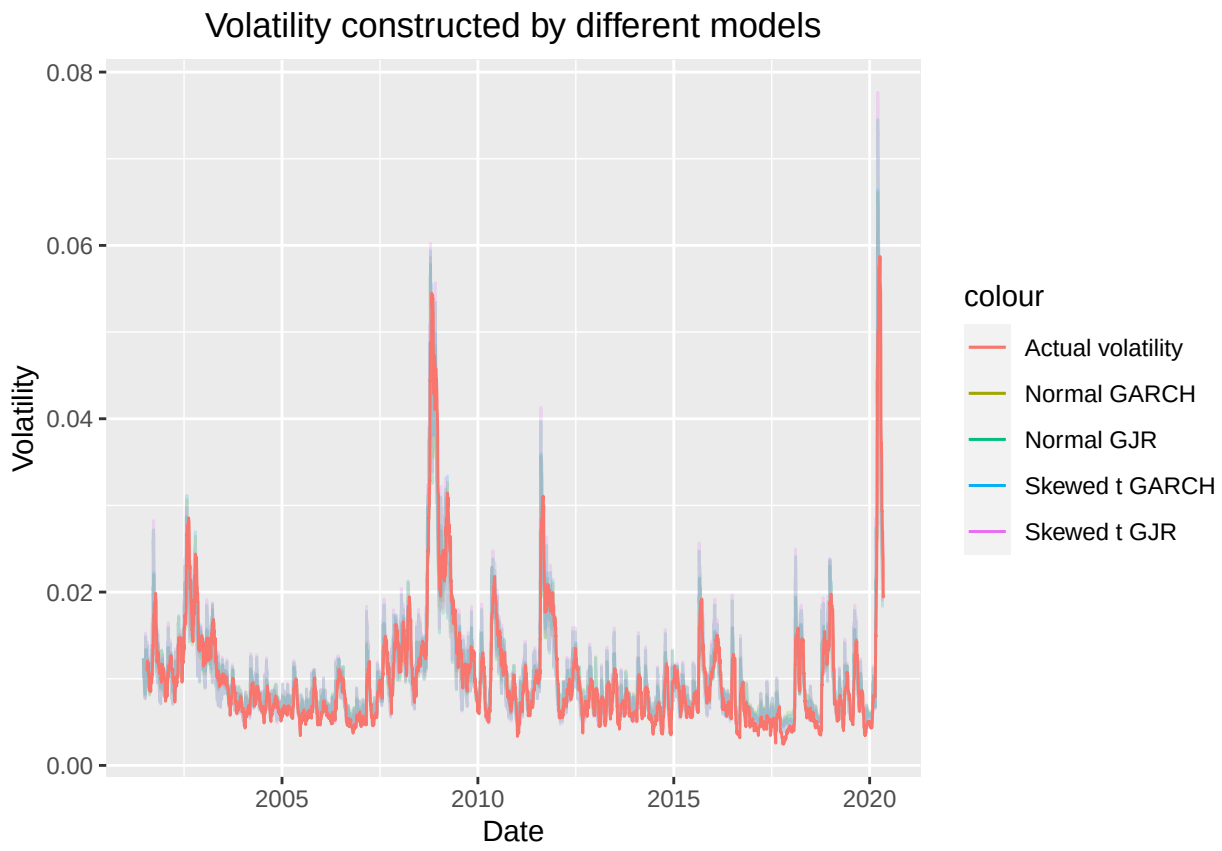
```
print(round(sstd_gjr_fit@fit$robust.matcoef,10))
```

```
##            Estimate   Std. Error      t value   Pr(>|t|)
## mu     0.0002495716 0.0002846399 8.767975e-01 0.38059664
## omega  0.0000020107 0.0000056600 3.552445e-01 0.72240643
## alpha1 0.0000000450 0.0445173275 1.011800e-06 0.99999919
## beta1  0.8732336384 0.0338930298 2.576440e+01 0.00000000
## gamma1 0.2246589215 0.1280897966 1.753917e+00 0.07944466
## skew   0.8546648715 0.0164777668 5.186776e+01 0.00000000
## shape  8.2593642262 1.0175550758 8.116872e+00 0.00000000
```

```
# Visualizing impact of negative previous return on variance
norm_garch_news <- as.data.frame(newsimpact(norm_garch_fit)[1:2])
norm_gjr_news <- as.data.frame(newsimpact(norm_gjr_fit) [1:2])
sstd_garch_news <- as.data.frame(newsimpact(sstd_garch_fit)[1:2])
sstd_gjr_news <- as.data.frame(newsimpact(sstd_gjr_fit)[1:2])

ggplot() +
  geom_point(data = norm_garch_news,
             aes(x = zx, y =zy, color = 'Normal GARCH')) +
  geom_line(data = norm_gjr_news,
            aes(x = zx, y = zy, color = 'Normal GJR')) +
  geom_line(data = sstd_garch_news,
            aes(x = zx, y = zy, color = 'Skewed t GARCH')) +
  geom_line(data = sstd_gjr_news,
            aes(x = zx, y = zy, color = 'Skewed t GJR')) +
  labs(x = 'Error', y = 'Variance',
       title = 'Dependence of variance on errors in different models') +
  theme(plot.title = element_text(hjust = 0.5))
```




```
# Visualizing volatility
norm_garch_vol <- sigma(norm_garch_fit)
norm_gjr_vol <- sigma(norm_gjr_fit)
sstd_garch_vol <- sigma(sstd_garch_fit)
sstd_gjr_vol <- sigma(sstd_gjr_fit)

p <- ggplot() +
  geom_line(data = norm_garch_vol, aes(x = index(norm_garch_vol[,1]),
                                       y = norm_garch_vol[,1],
                                       color = 'Normal GARCH'), alpha = 0.2)+
  geom_line(data = norm_gjr_vol, aes(x = index(norm_gjr_vol[,1]),
                                     y = norm_gjr_vol[,1],
                                     color = 'Normal GJR'), alpha = 0.2) +
  geom_line(data = sstd_garch_vol, aes(x = index(sstd_garch_vol[,1]),
                                       y = sstd_garch_vol[,1],
                                       color = 'Skewed t GARCH'), alpha = 0.2) +
  geom_line(data = sstd_gjr_vol, aes(x = index(sstd_gjr_vol[,1]),
                                     y = sstd_gjr_vol[,1],
                                     color = 'Skewed t GJR'), alpha = 0.2) +
  geom_line(data = vol_df[,1], aes(y = vol_df[,1], x = index(vol_df[,1]),
                                   color = 'Actual volatility')) +
  labs(x = 'Date', y = 'Volatility',
       title = 'Volatility constructed by different models') +
  theme(plot.title = element_text(hjust = 0.5))

suppressMessages(suppressWarnings(print(p)))
```




```
rm(p)
```

```
# Predicting volatility for n.ahead periods
norm_garch_forecast <- ugarchforecast(fitORspec = norm_garch_fit,
                                      data = norm_garch_vol, n.ahead = 10)
norm_gjr_forecast <- ugarchforecast(fitORspec = norm_gjr_fit,
                                    data = norm_gjr_vol, n.ahead = 10)
sstd_garch_forecast <- ugarchforecast(fitORspec = sstd_garch_fit,
                                      data = sstd_garch_vol, n.ahead = 10)
sstd_gjr_forecast <- ugarchforecast(fitORspec = sstd_gjr_fit,
                                    data = sstd_gjr_vol, n.ahead = 10)
```

```
# 252 - number of working days in year
#van_norm_res <- sigma(van_forecast) #* sqrt(252)

# garch_for_funds <- function(funds) {
#   for (i in 1:ncol(funds)) {
#     return()
#   }
# }
```

```
# chart.Histogram(residuals(van_sstd_fit, standardize = T),
#                 methods = c('add.normal', 'add.density', 'add.student'))
```

```
# cat('SST for norm\n')
# sum(vanguard_tail_volatility - van_norm_res)
#
```

```
# cat('\nSST for sstd\n')
# sum(vanguard_tail_volatility - van_sstd_res)
#
# cbind(vanguard_tail_volatility, van_norm_res, van_sstd_res)
```

```
# vanguard_std <- (funds$vanguard_return - mean(funds$vanguard_return))/sd(funds$vanguard_return)
#
# ggplot(data = data.frame(x = c(-10, 10)), aes(x)) +
#   stat_function(fun = dnorm, n = 4800, args = list(mean = 0, sd = 3)
#                 , aes(x = x,colour = 'Normal')) + ylab("") +
#   scale_y_continuous(breaks = NULL) +
#   stat_function(fun = dskt,  n = 4800, args = list(df = 100, gamma = 1.2),
#                 aes(colour = 'Skewed Student t')) +
#   geom_histogram(data = vanguard_std, aes(x = vanguard_std,y = ..density..), bins = 100, alpha = 0.3)
```