# R Notebook

```r
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
```

```r
require(ggplot2)
```

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

```r
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
```

```r
require(rugarch)
```

```
## Loading required package: rugarch

## Loading required package: parallel

##
## Attaching package: ’rugarch’

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

```r
require(PerformanceAnalytics)
```

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

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

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

```r
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.
```

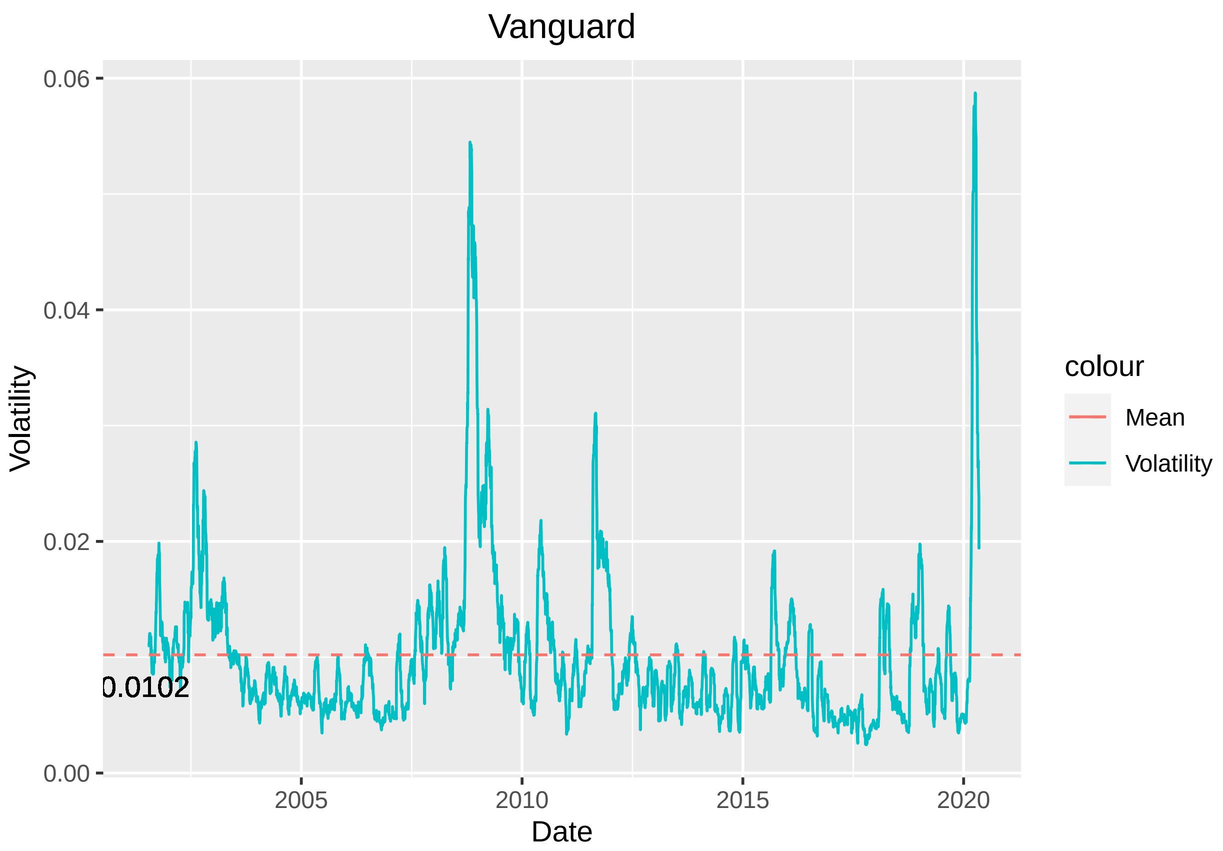
```r
library(skewt)
```

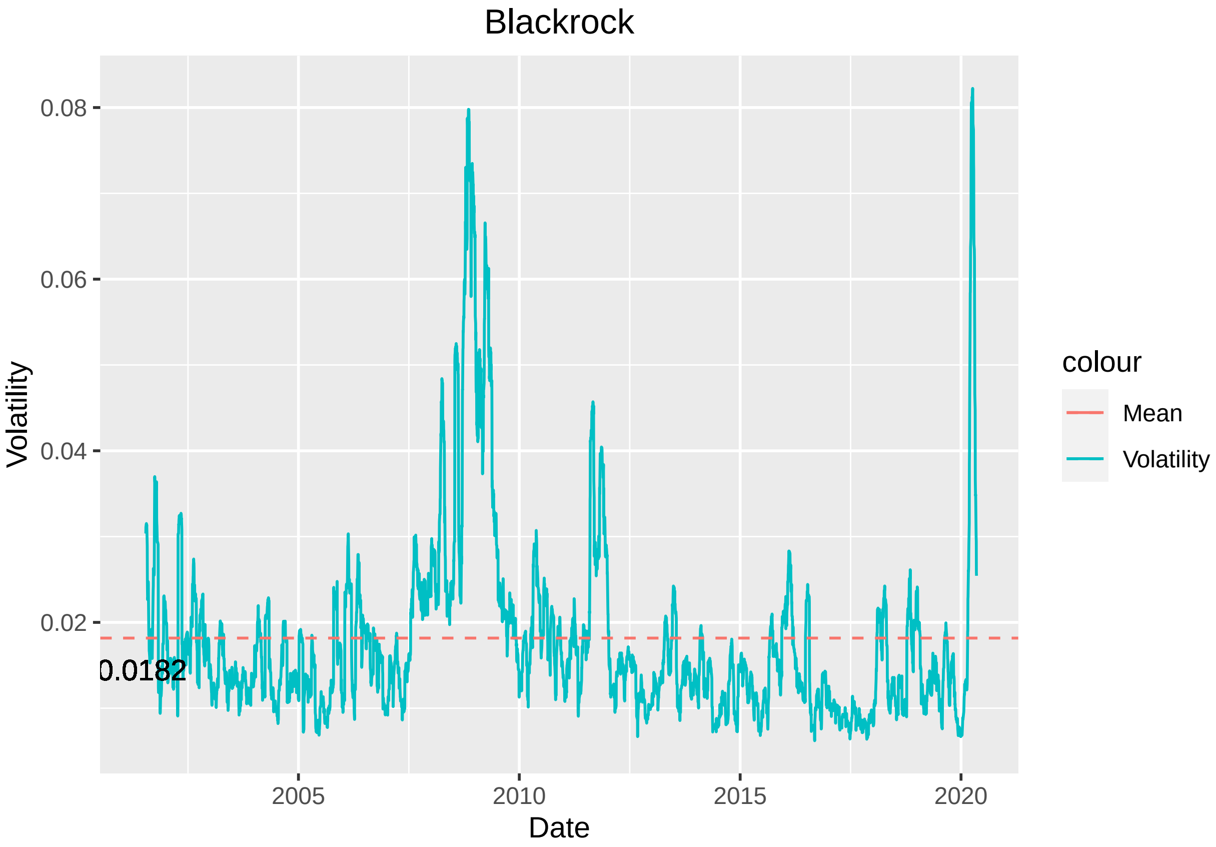
```r
read_funds <- function(lf) {
  dfs <- list()
 browser()
  for (f in 1:length(lf)) {
      cond <- substr(lf[f], nchar(lf[f])-3, nchar(lf[f])) == '.csv'
      if (cond) {
      temp <- data.frame(read.csv(lf[f], stringsAsFactors = FALSE))
      temp$Date <- as.Date(temp$Date)
      temp$Close <-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

  return(dfs)
}

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

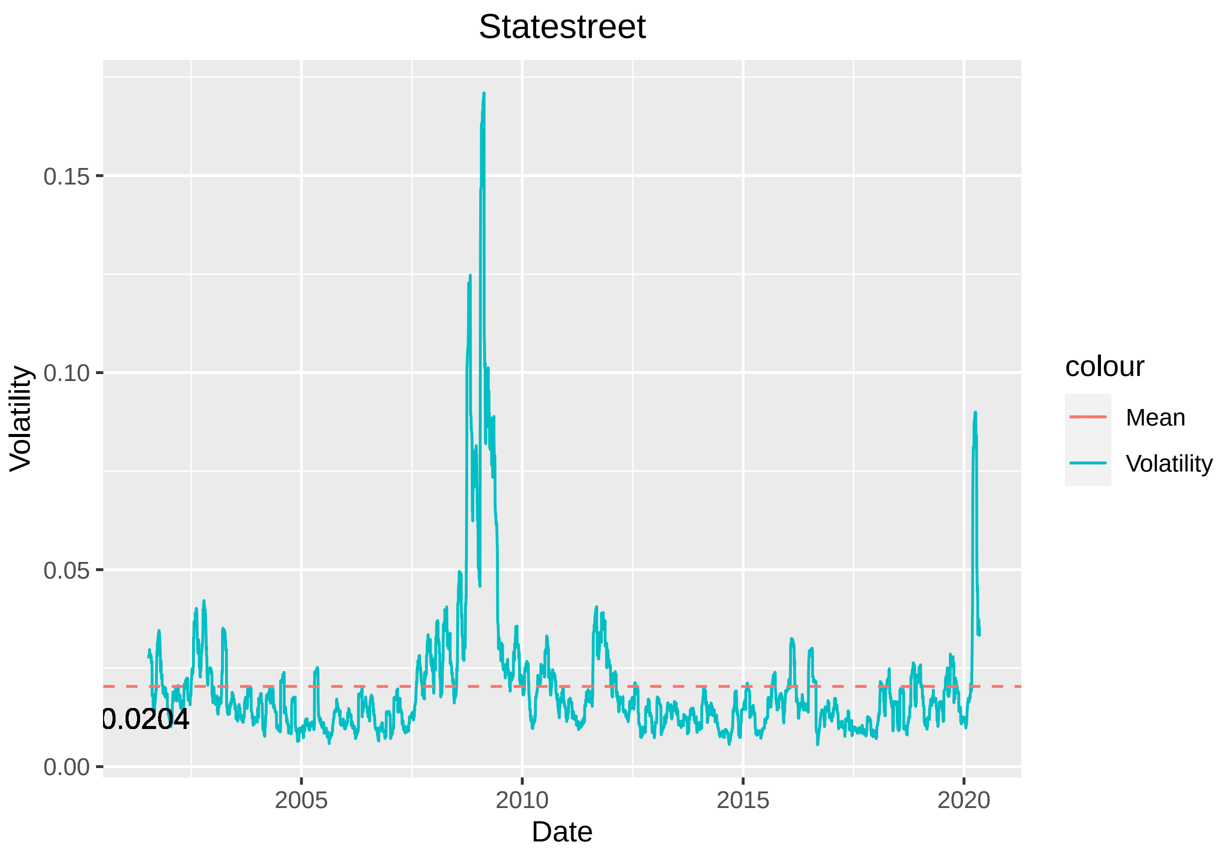
```r
# Data loading
df <- read.csv('Funds.csv', stringsAsFactors = F)
df <- df[,-1]
df$Date <- as.Date(df$Date)

# 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']
```

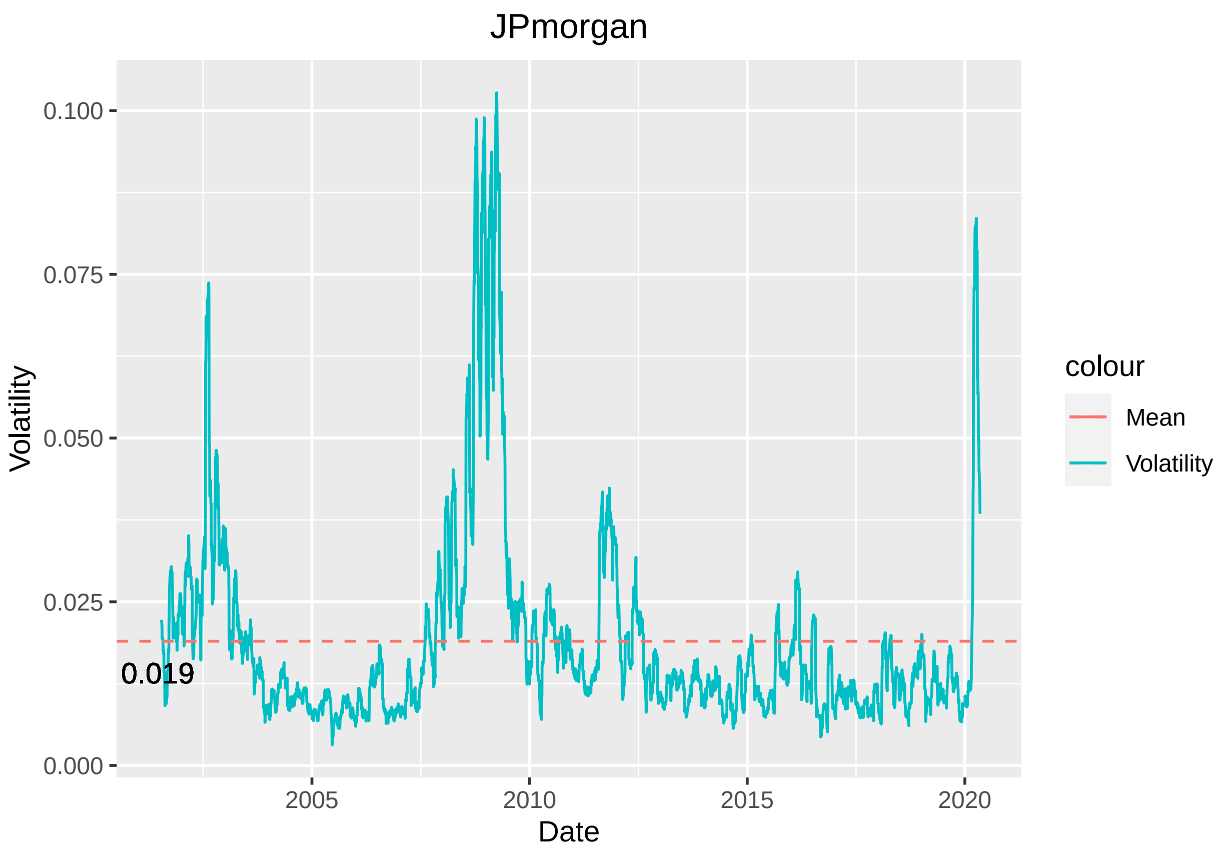
```r
get_volatiles <- function(funds, width = 22, time_scale = 1, funds_names) {
    vol_df <- data.frame()

    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,
                ethods = 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)
  }
}
```

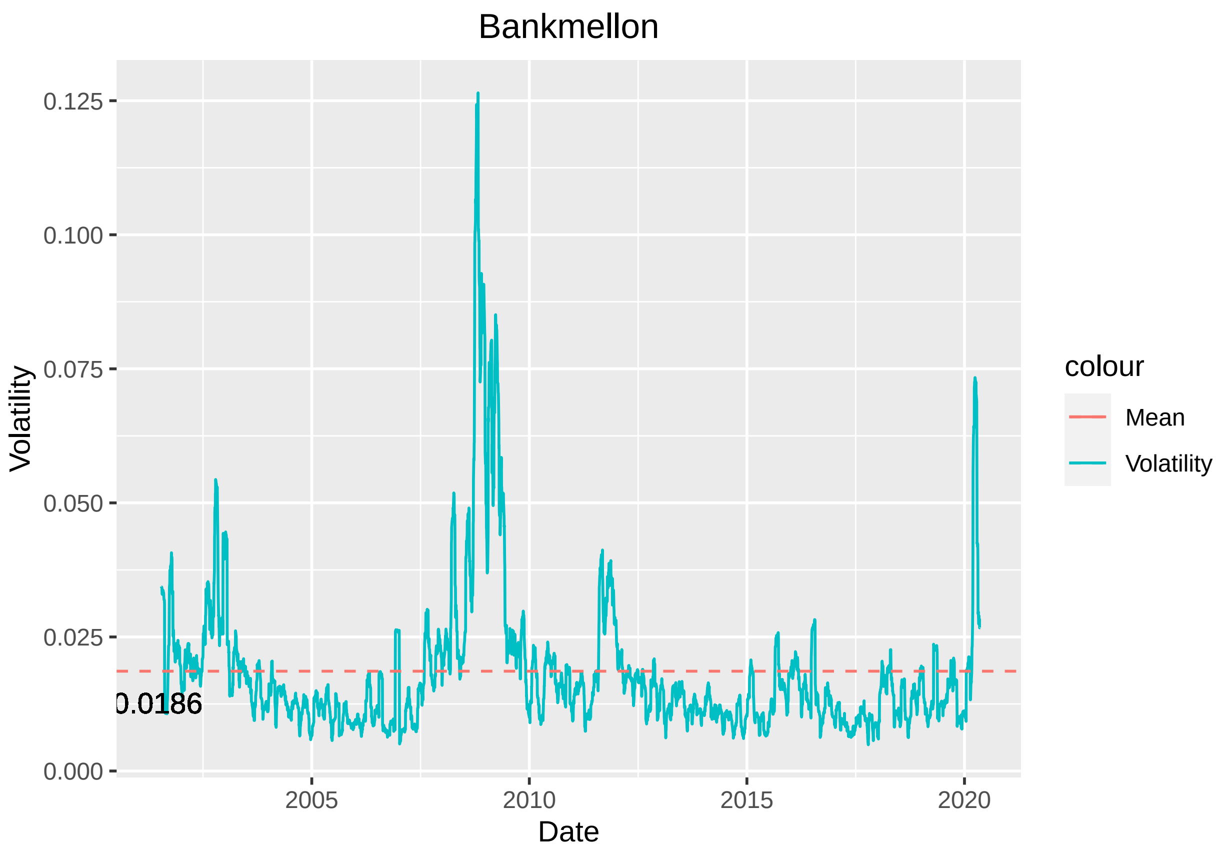
```
vol_df <- get_volatiles(funds, funds_names =  funds_names)

visualize_funds_lines(vol_df)
```

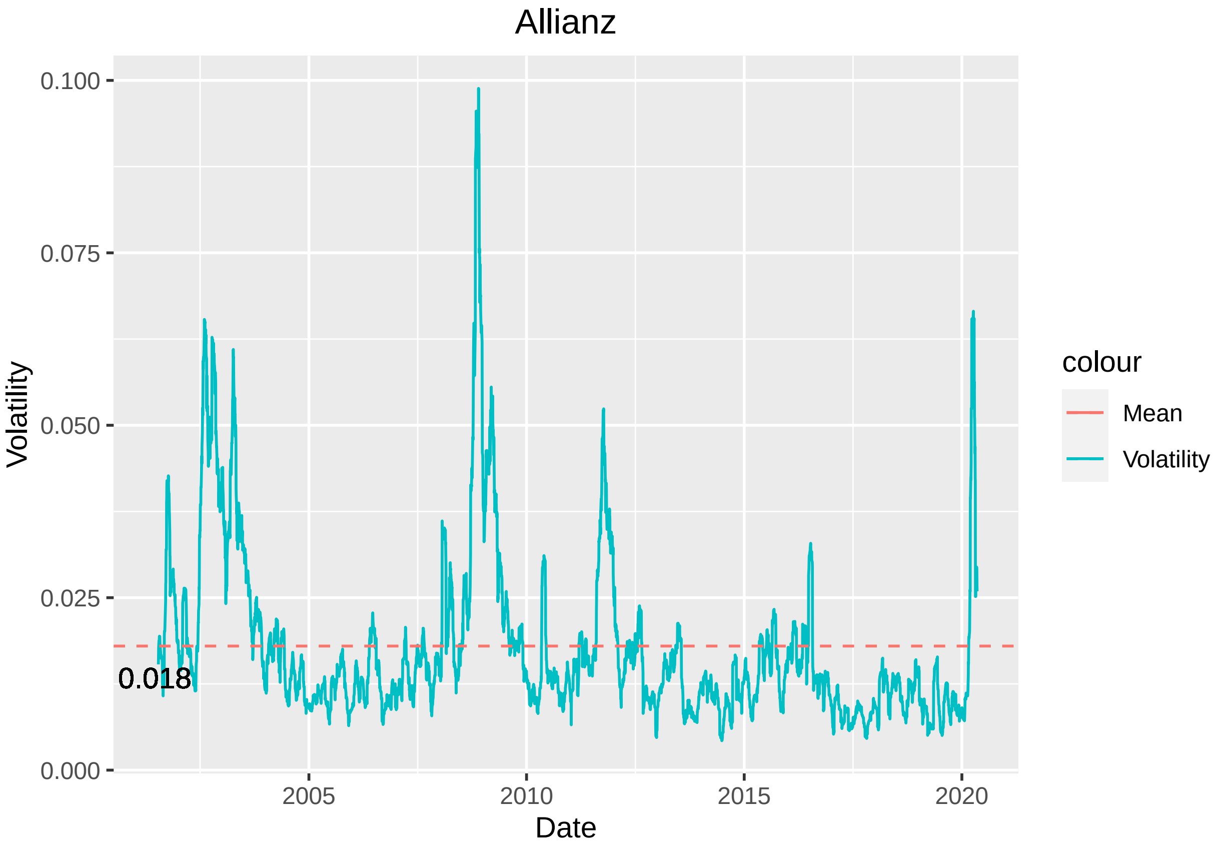

Vanguard
0.06
0.04
0.02
0.00
Volatility
0.0102
2005
2010
2015
2020
Date
colour
Mean
Volatility


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

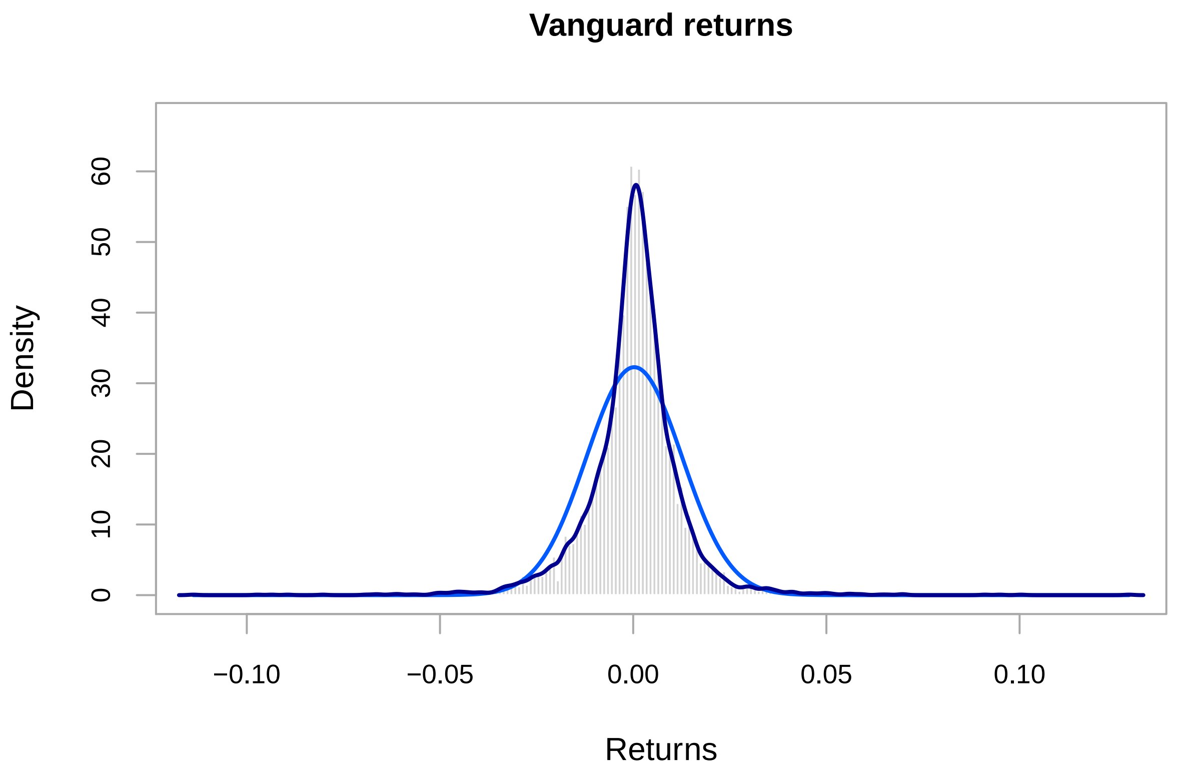

Statestreet
Volatility
0.15
0.10
0.05
0.00
0.0204
2005
2010
2015
2020
Date
colour
Mean
Volatility

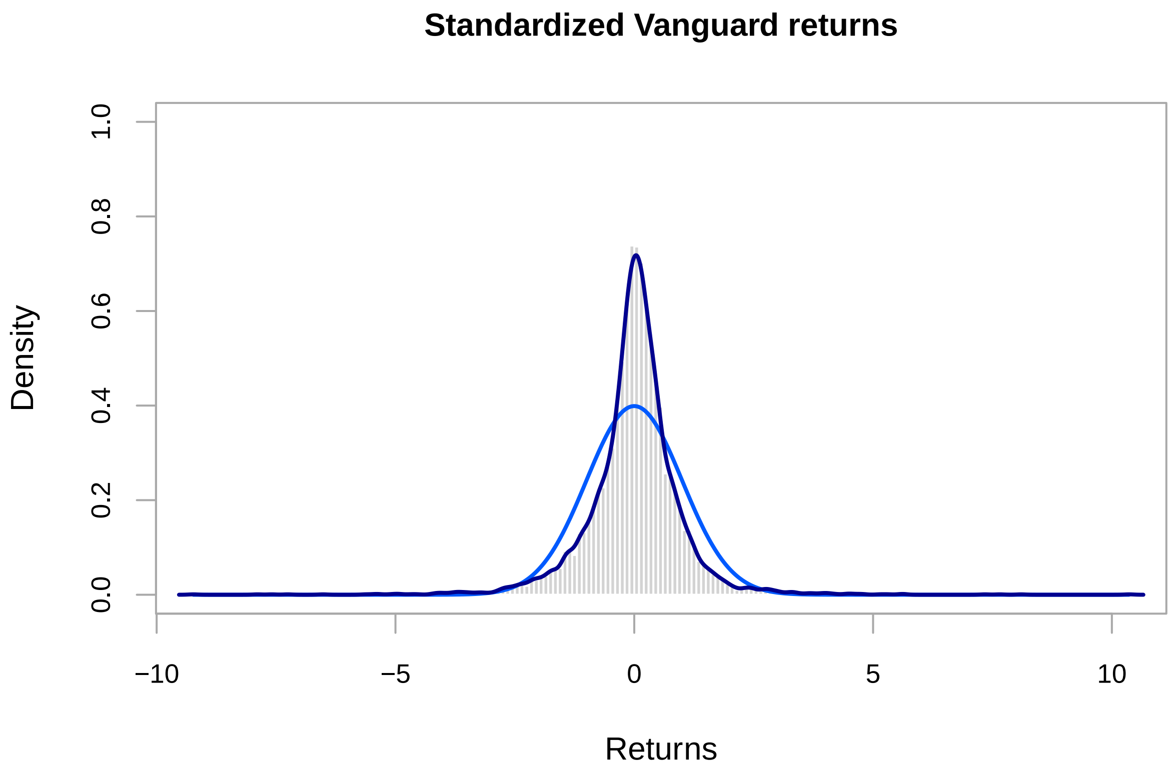

JPmorgan
Volatility
0.100
0.075
0.050
0.025
0.000
0.019
2005
2010
2015
2020
Date
colour
Mean
Volatility

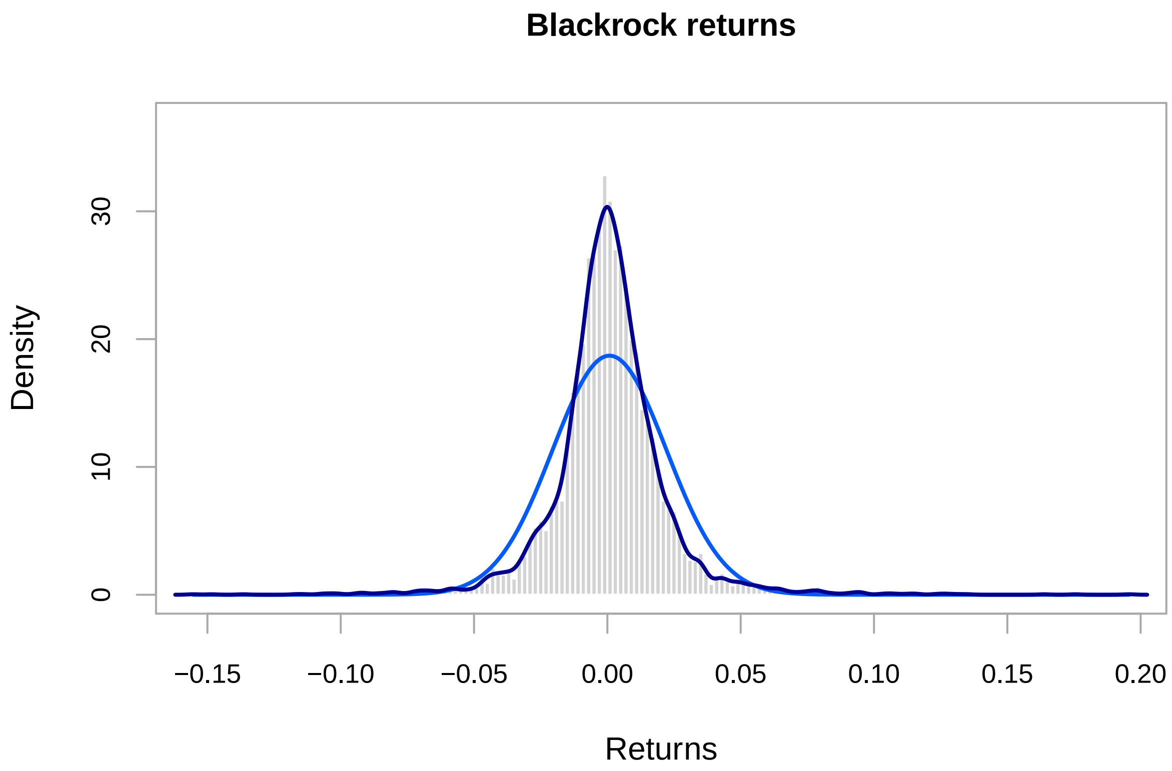

Bankmellon
Volatility
0.125
0.100
0.075
0.050
0.025
0.000
0.0186
2005
2010
2015
2020
Date
colour
Mean
Volatility

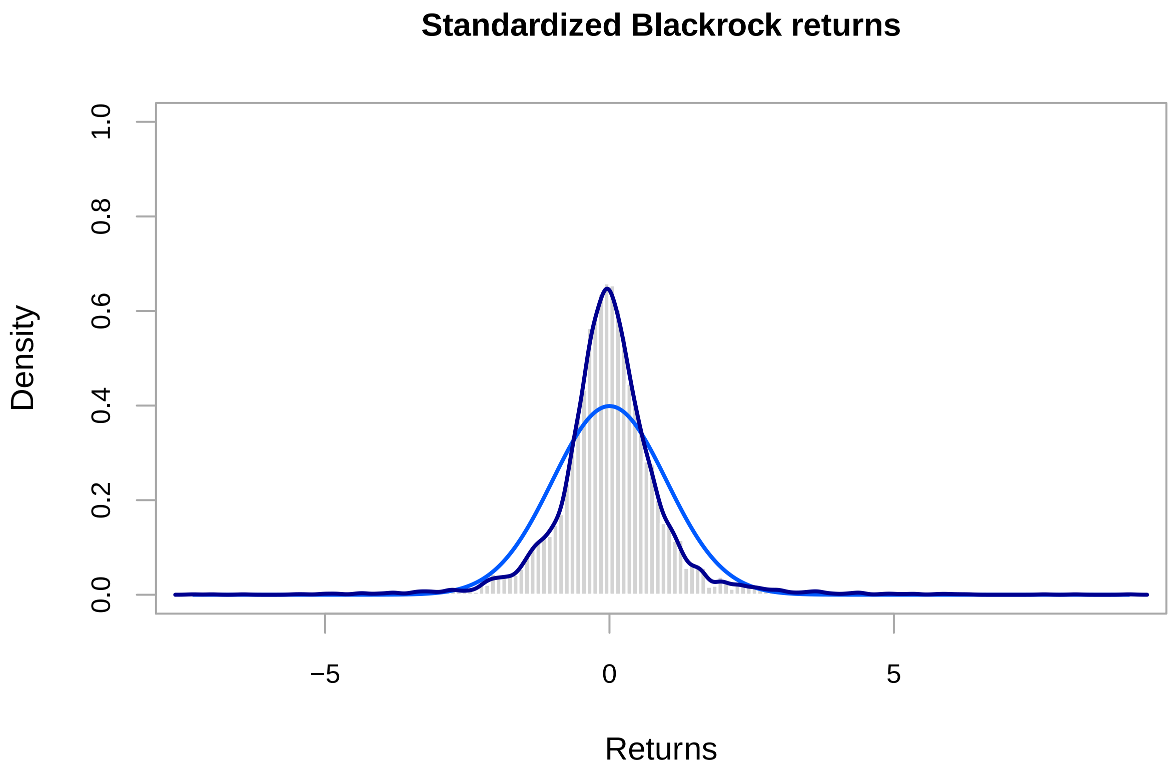


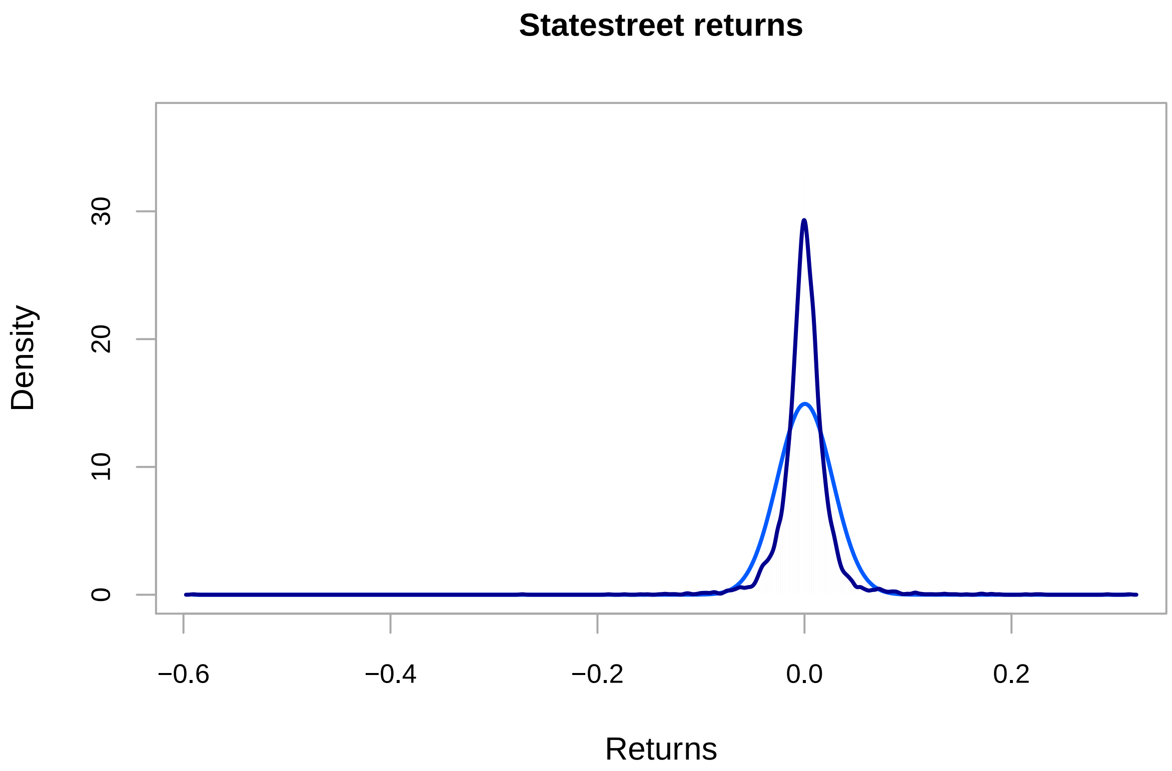
```
visualize_funds_hist(funds)
```

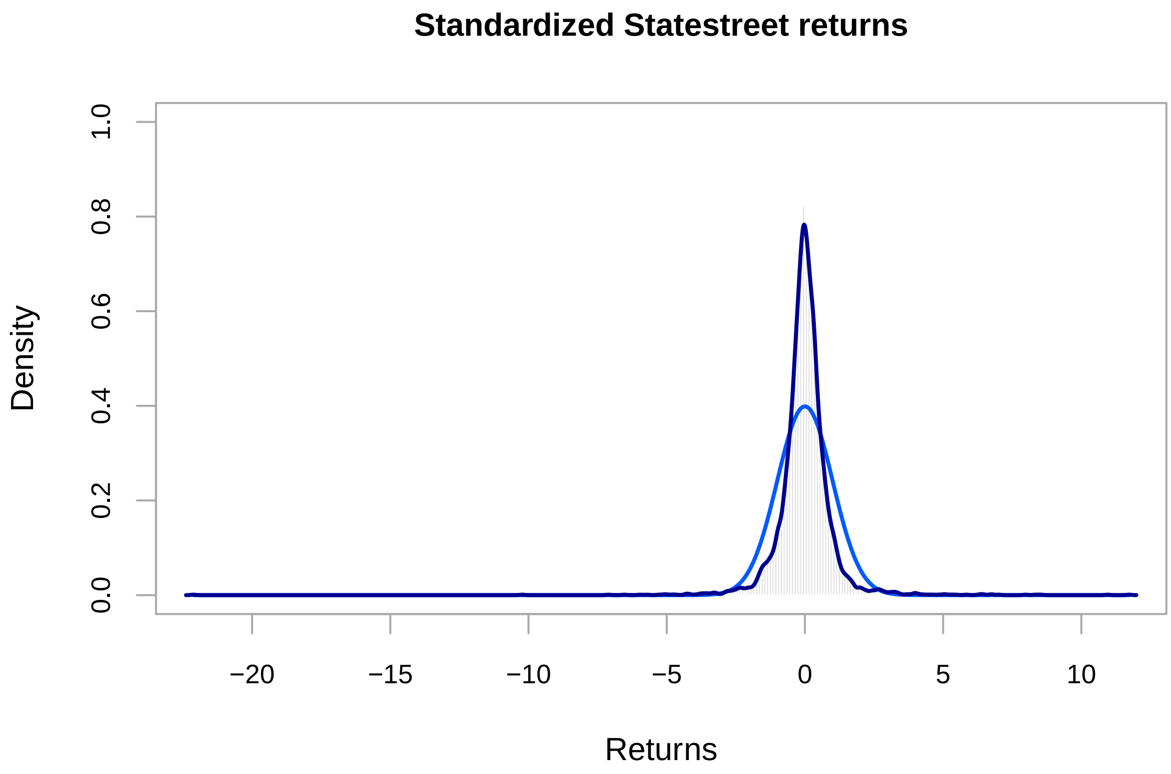

Vanguard returns
Density
0
10
20
30
40
50
60
−0.10
−0.05
0.00
0.05
0.10
Returns

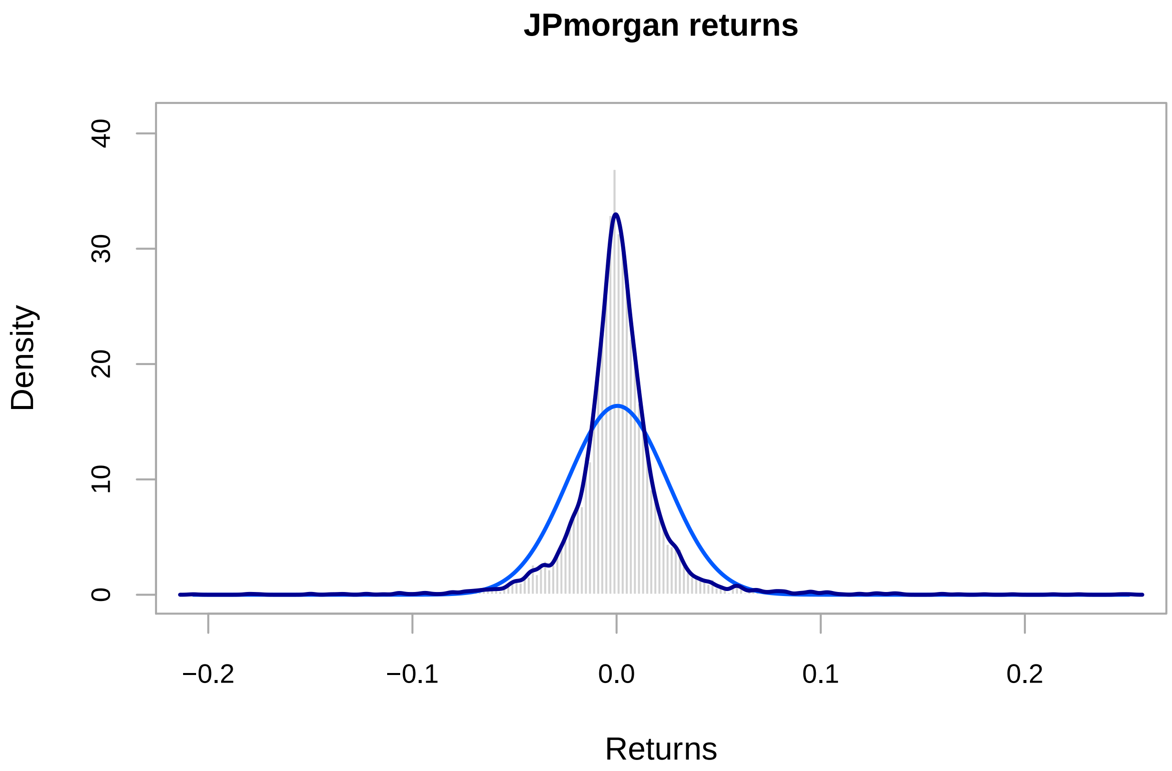

Standardized Vanguard returns
Density
0.0
0.2
0.4
0.6
0.8
1.0
−10
−5
0
5
10
Returns

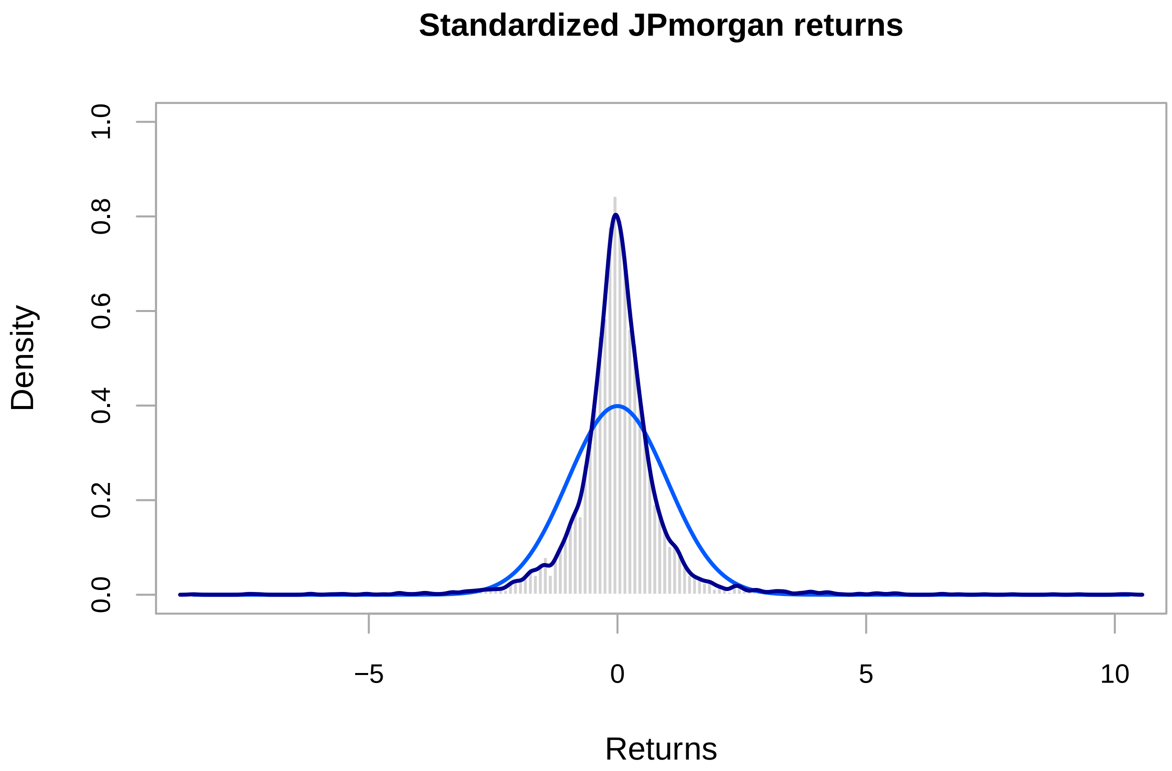

Blackrock returns
Density
0
10
20
30
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
0.20
Returns

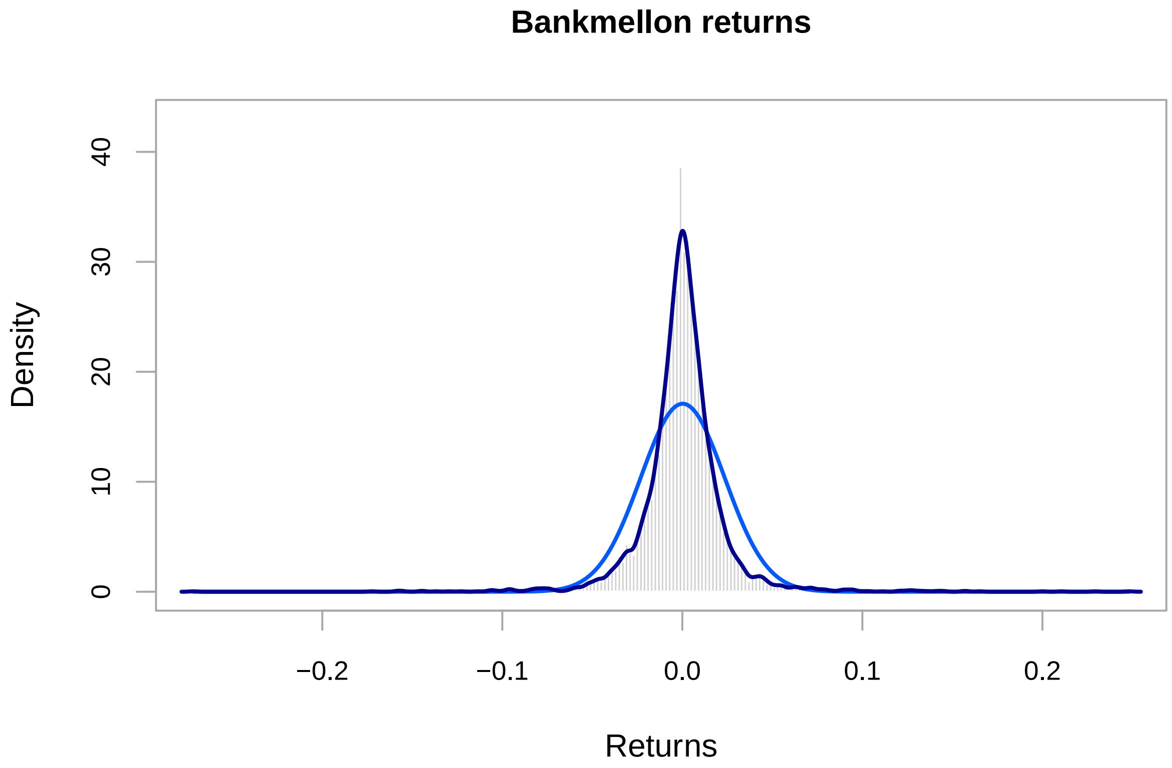

Standardized Blackrock returns
Density
0.0
0.2
0.4
0.6
0.8
1.0
−5
0
5
Returns

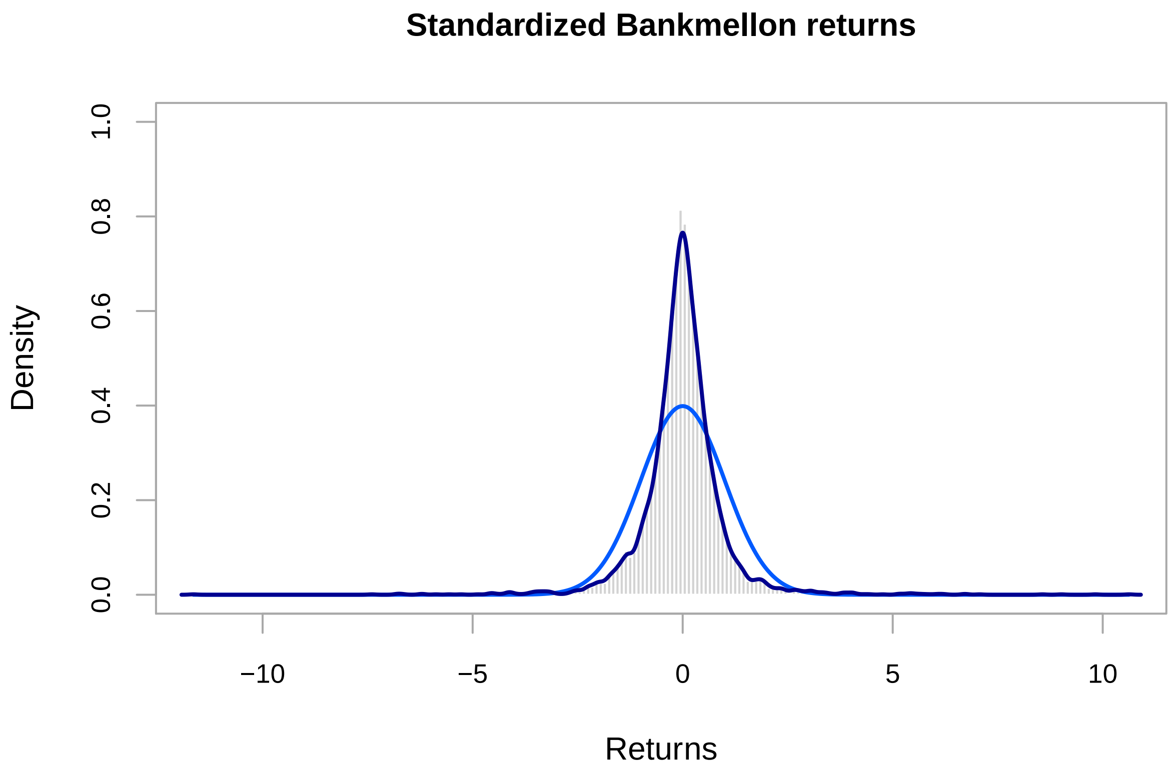

Statestreet returns
Density
0
10
20
30
−0.6
−0.4
−0.2
0.0
0.2
Returns

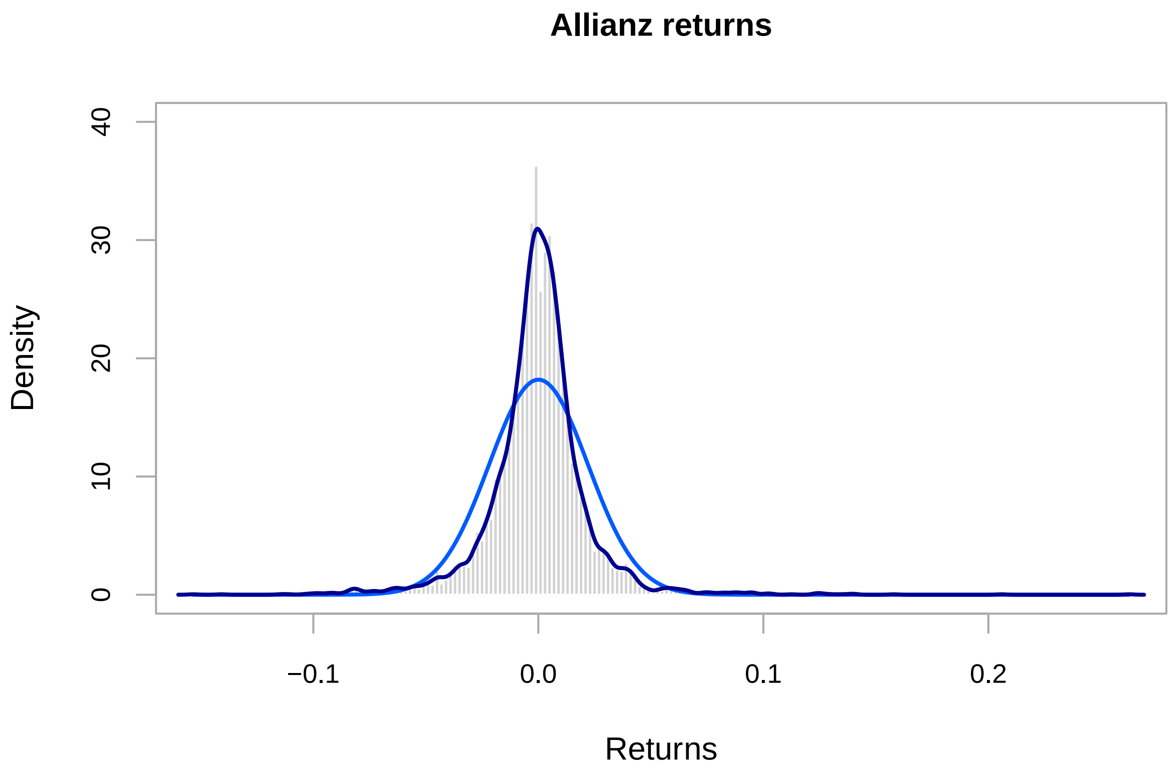

Standardized Statestreet returns
Density
0.0
0.2
0.4
0.6
0.8
1.0
−20
−15
−10
−5
0
5
10
Returns

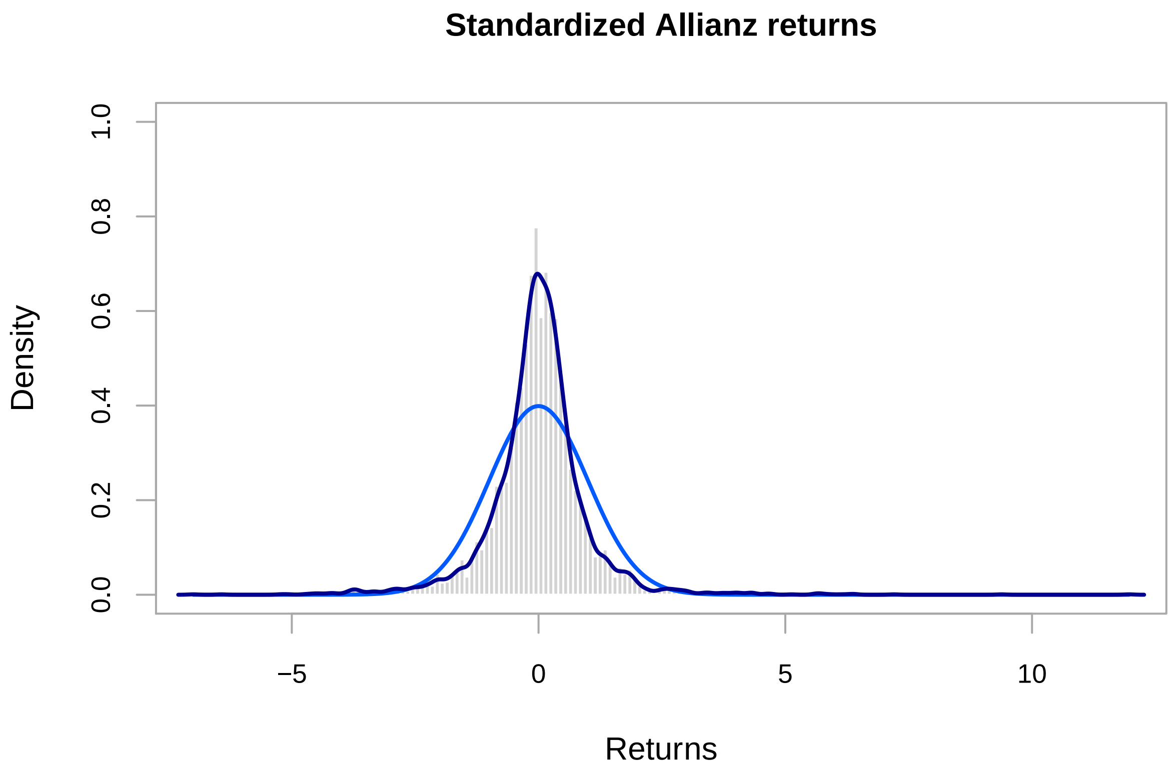

JPmorgan returns
Density
0
10
20
30
40
−0.2
−0.1
0.0
0.1
0.2
Returns


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



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

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

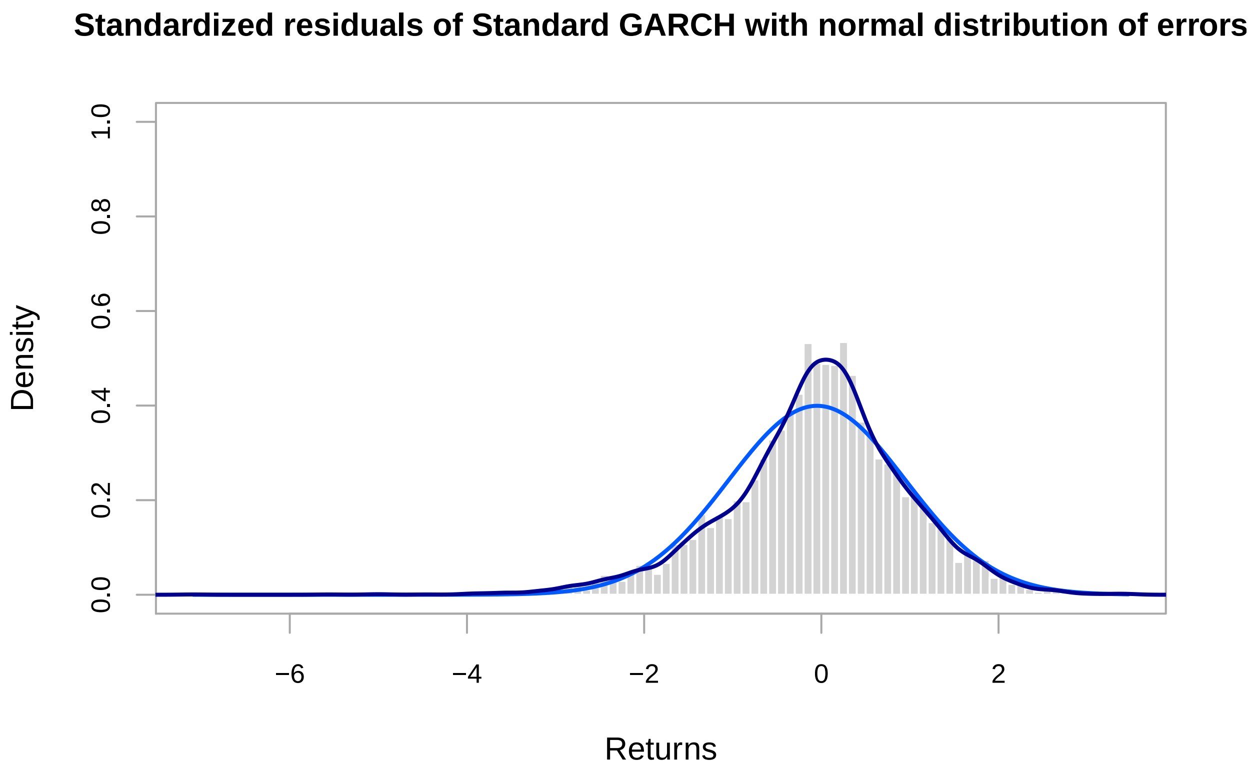
```
fund <- funds$vanguard_return


# 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')
```

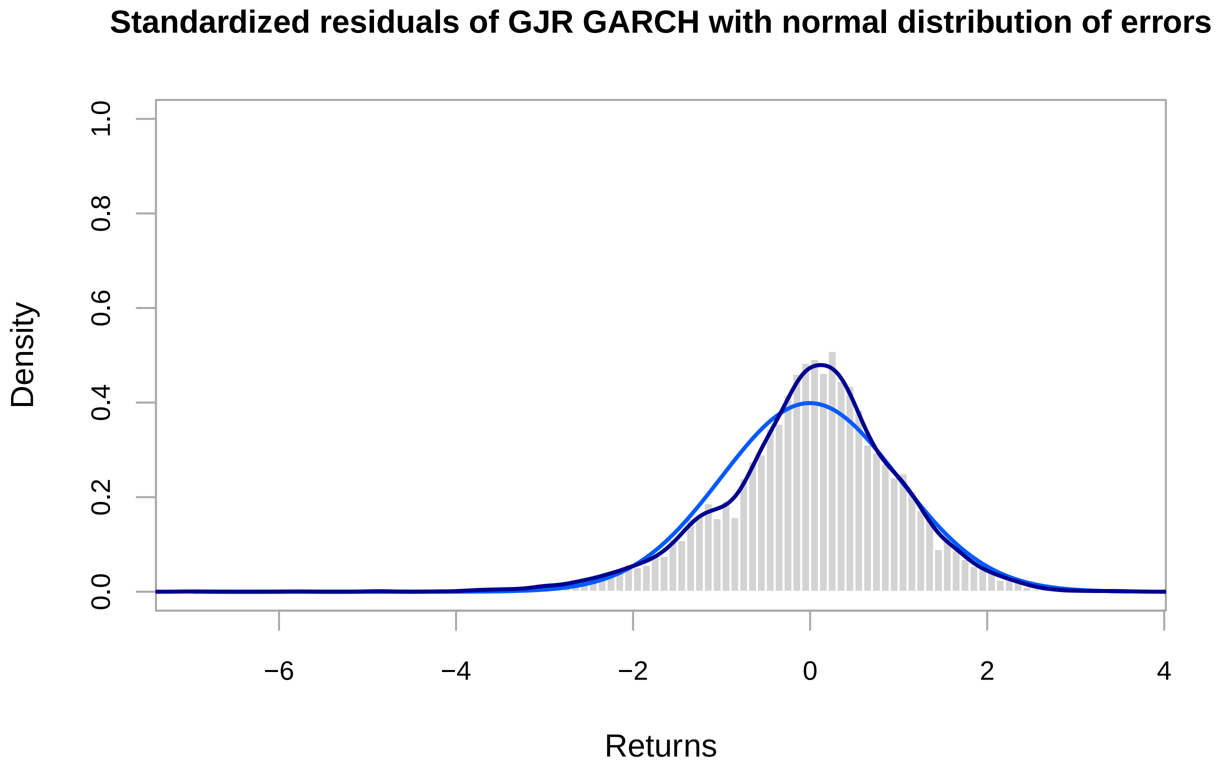
```r
# GJR GARCH with skewed Student t distribution of errors
sstd_gjr_spec <- ugarchspec(mean.model = list(armaOrder = c(0,0)),
                       variance.model = list(model = 'gjrGARCH'),
                       distribution.model = 'sstd')

garch_specs <- list(norm_garch_spec, norm_gjr_spec,
                     sstd_garch_spec, sstd_gjr_spec)

rm(norm_garch_spec, norm_gjr_spec, sstd_garch_spec, sstd_gjr_spec)
```

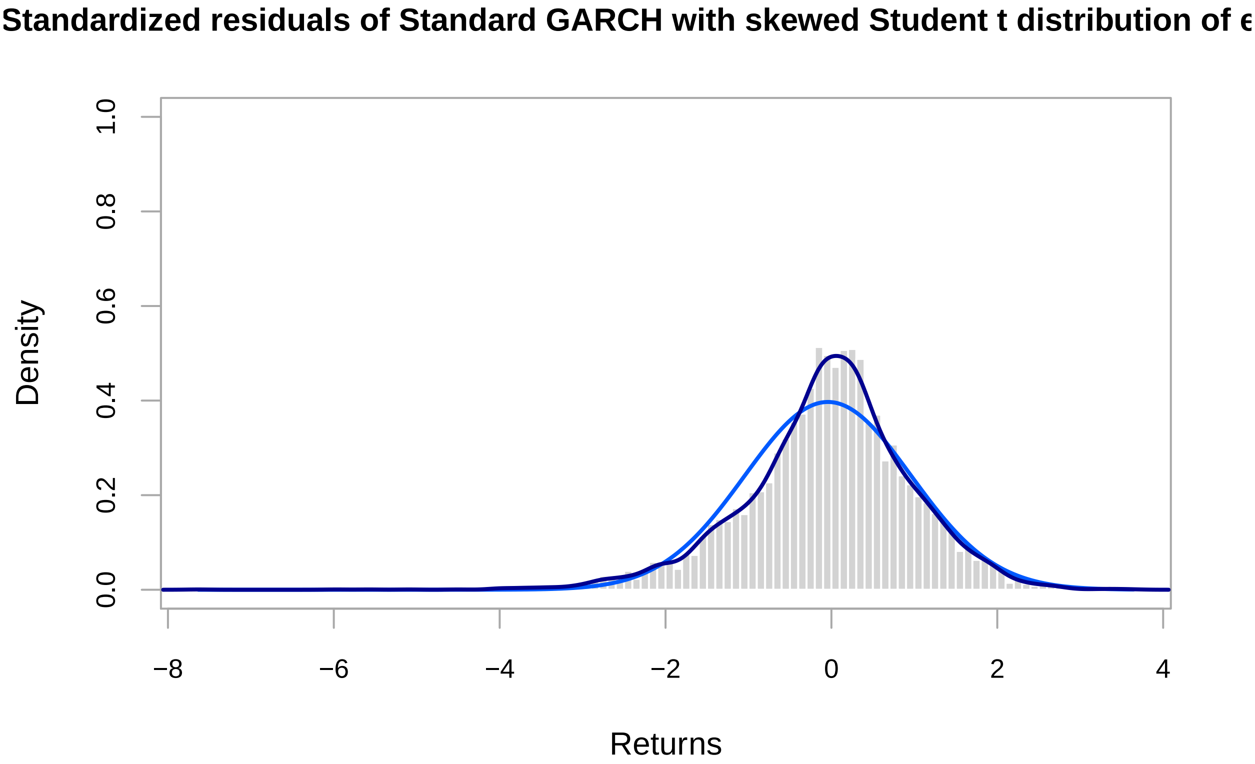
```r
# Models naming
model_names <- c('Standard GARCH with normal distribution of errors',
                         'GJR GARCH with normal distribution of errors',
                         'Standard GARCH with skewed Student t distribution of errors',
                         'GJR GARCH with skewed Student t distribution of errors')

short_model_names <- c('Normal GARCH', 'Normal GJR', 'Skewed t GARCH', 'Skewed t GJR')

names(garch_specs) <- model_names
```

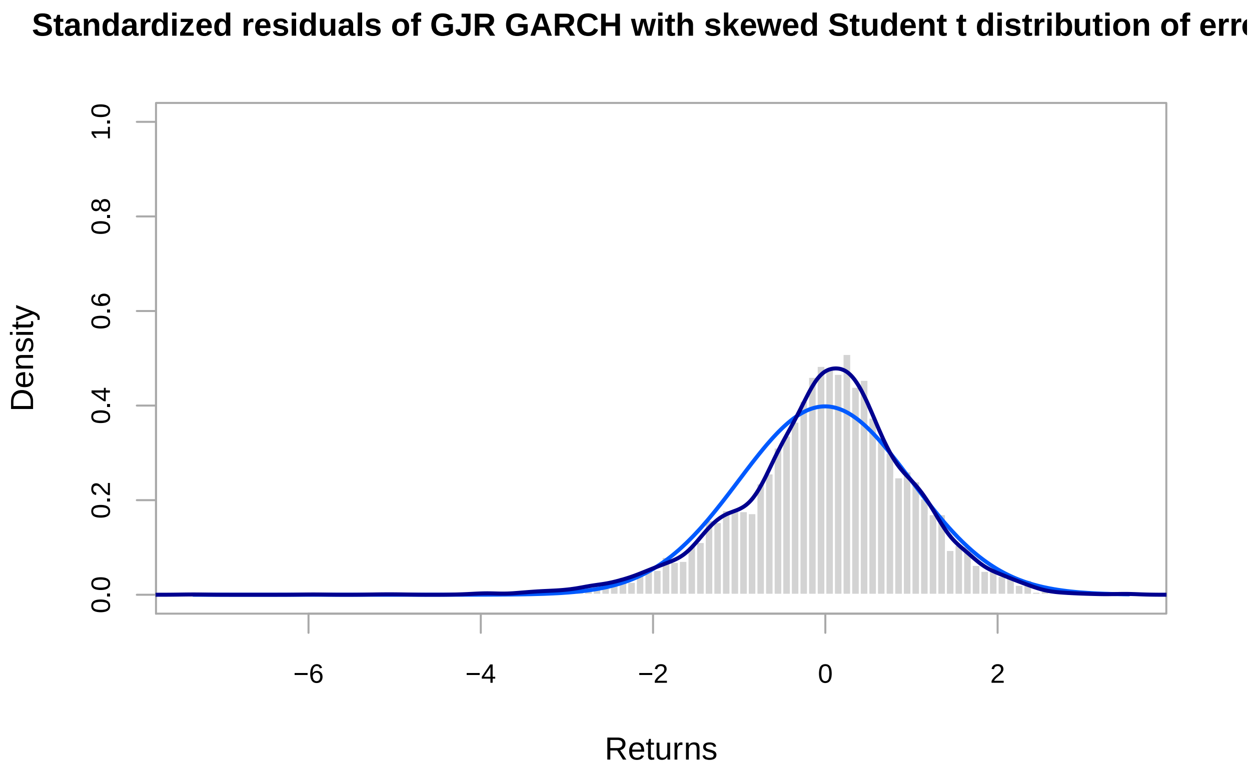
```r
# Apply GARCH model to our data
garch_fits <- list()

for (s in 1:length(garch_specs)) {
  suppressWarnings(garch_fits[[s]] <- ugarchfit(data = fund,
                                              spec = garch_specs[[s]]))
}
rm(s)
names(garch_fits) <- model_names
```

```r
# Visualizing standardized residuals for models
for (f in 1:length(garch_fits)) {
  chart.Histogram(residuals(garch_fits[[f]], standardize = T),
                methods = c('add.normal', 'add.density'),
                main = paste('Standardized residuals of',model_names[f]))
}
```

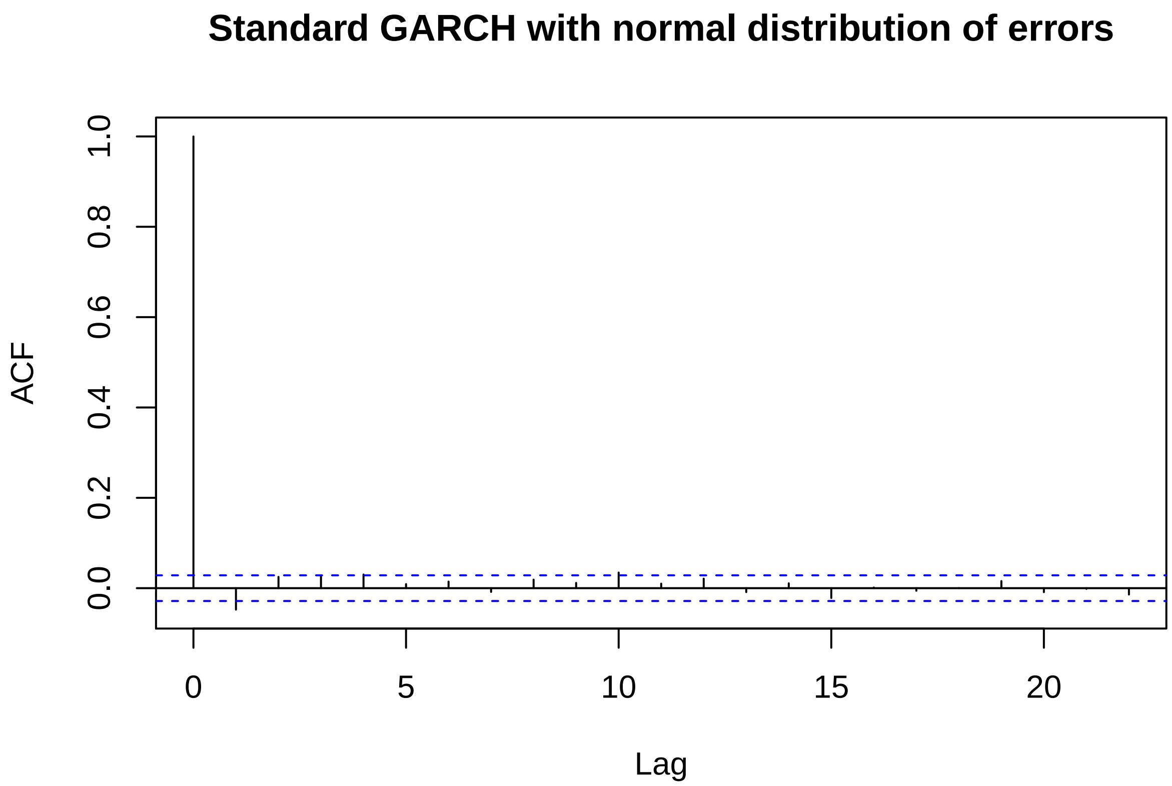

Standardized residuals of Standard GARCH with normal distribution of errors
Density
1.0
0.8
0.6
0.4
0.2
0.0
−6
−4
−2
0
2
Returns


Standardized residuals of GJR GARCH with normal distribution of errors
Density
1.0
0.8
0.6
0.4
0.2
0.0
−6
−4
−2
0
2
4
Returns

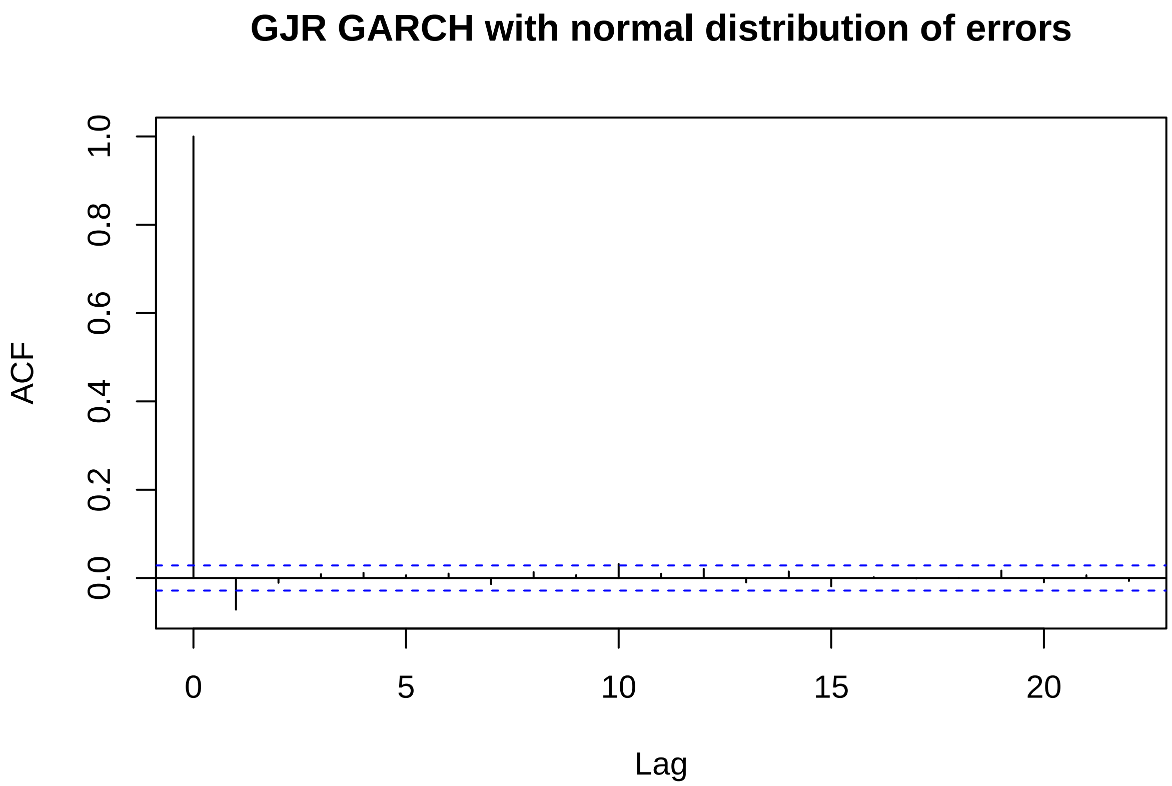

Standardized residuals of Standard GARCH with skewed Student t distribution of ϵ
Density
0.0
0.2
0.4
0.6
0.8
1.0
−8
−6
−4
−2
0
2
4
Returns



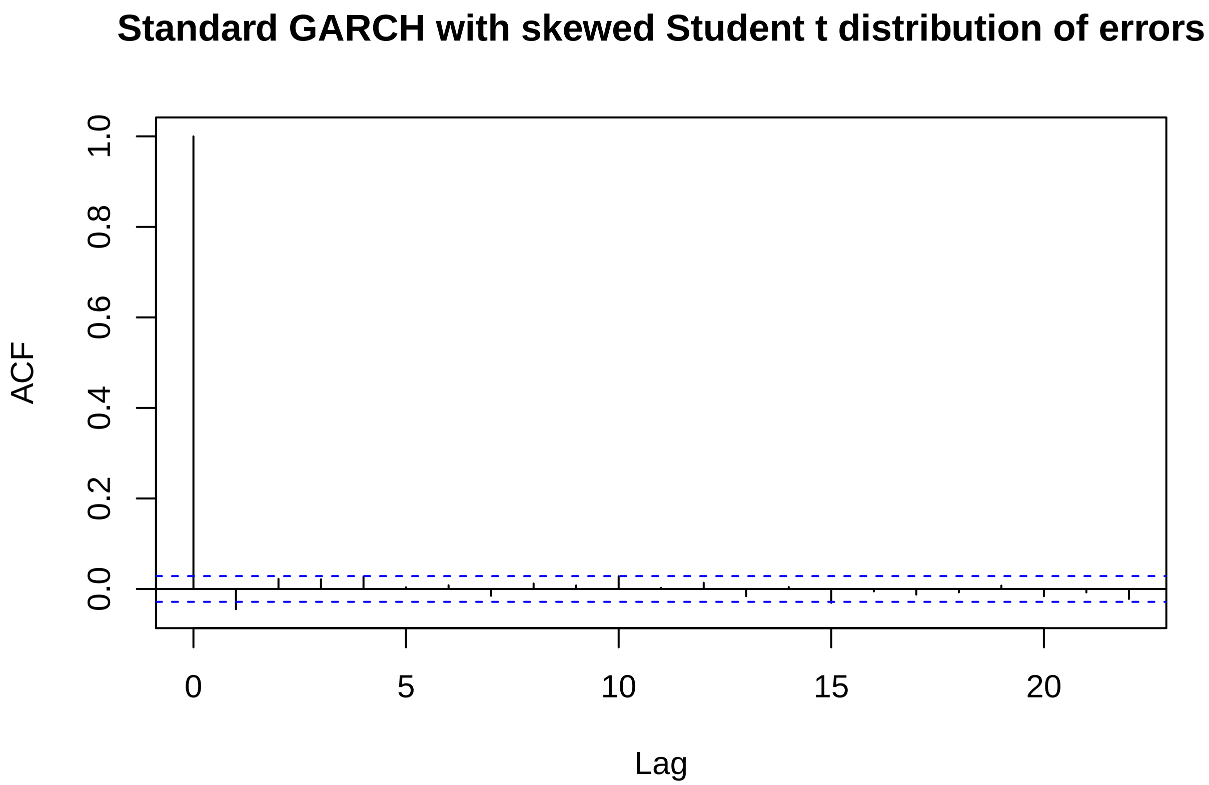
```
# Models validation
for (f in 1:length(garch_fits)) {
    standard_residuals <- residuals(garch_fits[[f]], standardize = T)
    p <- acf(abs(standard_residuals),22, plot = F)
    plot(p, main = names(garch_fits)[f])
    cat(’\n’, names(garch_fits)[f],’\n’)
    print(Box.test(abs(standard_residuals), 22, type = ’Ljung-Box’))
}
```



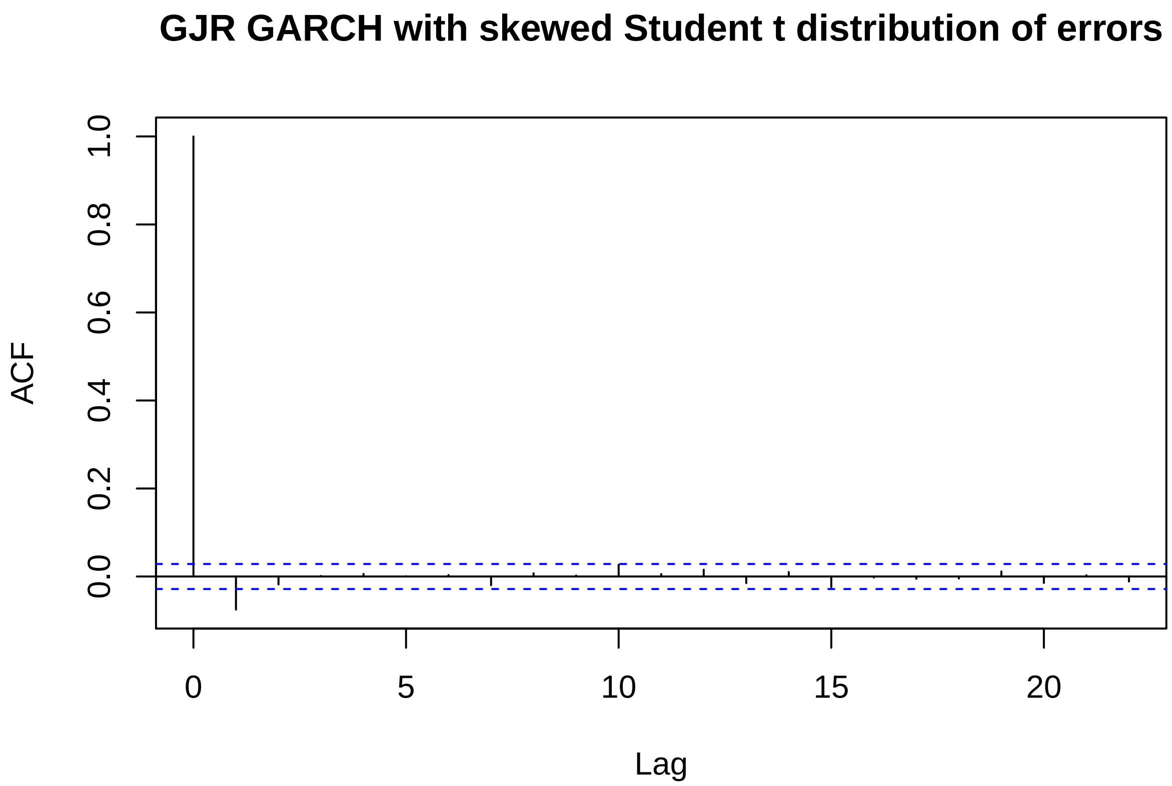
```
##
##  Standard GARCH with normal distribution of errors
##
##  Box-Ljung test
##
## data:  abs(standard_residuals)
## X-squared = 39.552, df = 22, p-value = 0.01219
```



```
##
##  GJR GARCH with normal distribution of errors
##
##  Box-Ljung test
##
## data:  abs(standard_residuals)
## X-squared = 41.362, df = 22, p-value = 0.00746
```



```
##
##  Standard GARCH with skewed Student t distribution of errors
##
##  Box-Ljung test
##
## data:  abs(standard_residuals)
## X-squared = 36.032, df = 22, p-value = 0.03013
```



```
##
##  GJR GARCH with skewed Student t distribution of errors
##
##  Box-Ljung test
##
## data:  abs(standard_residuals)
## X-squared = 42.602, df = 22, p-value = 0.00528
```

```
rm(standard_residuals, p)
```

```
#Coefficients
for (f in 1:length(garch_fits)) {
  cat('\nCoefficients of', names(garch_fits)[f], '\n')
  print(round(garch_fits[[f]]@fit$matcoef,10))

  cat('\nRobust coefficients of', names(garch_fits)[f], '\n')
  print(round(garch_fits[[f]]@fit$robust.matcoef,10))
}
```

```
##
## Coefficients of Standard GARCH with normal distribution of errors
##            Estimate    Std. Error   t value     Pr(>|t|)
## mu     0.0006678893 0.0001133821  5.890605 0.0000000038
## omega  0.0000025218 0.0000007821  3.224299 0.0012628155
## alpha1 0.1243765581 0.0097966632 12.695808 0.0000000000
## beta1  0.8552415698 0.0105548476 81.028320 0.0000000000
##
```

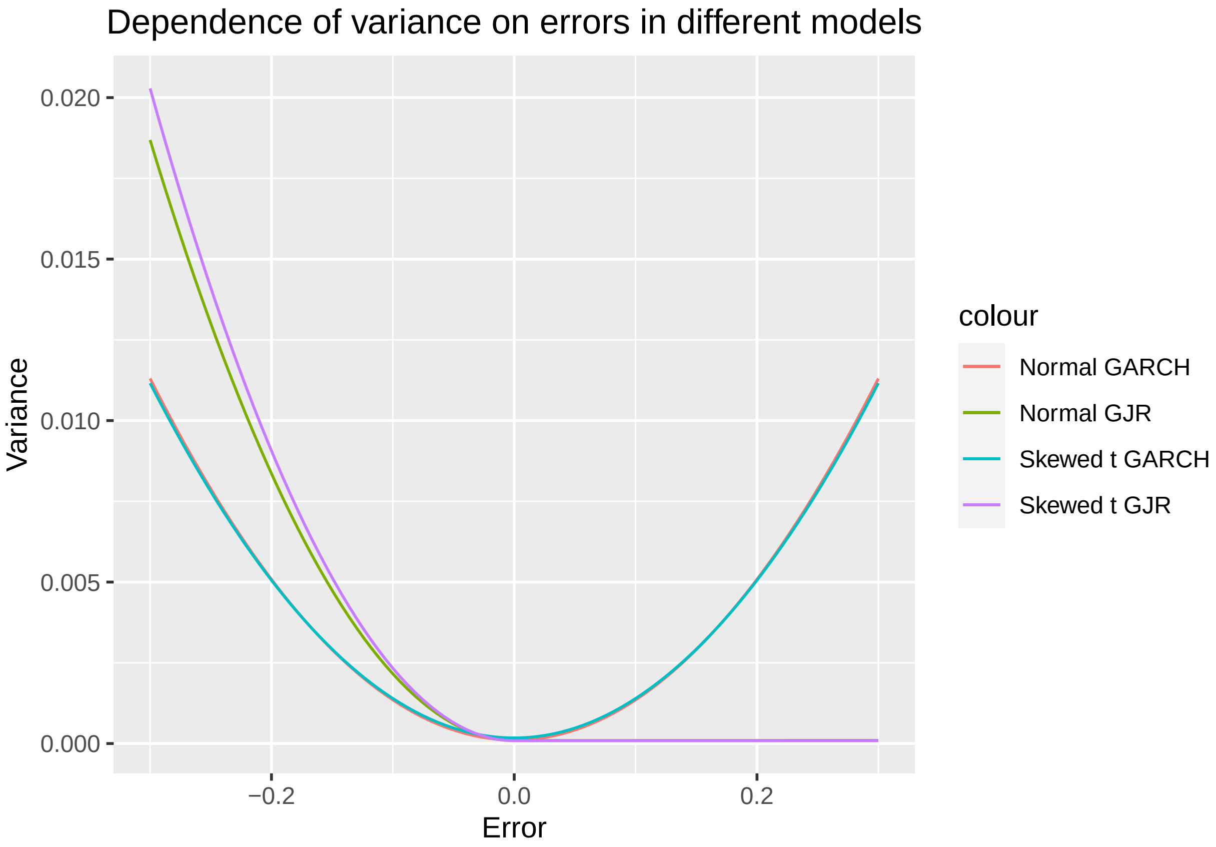
```
## Robust coefficients of Standard GARCH with normal distribution of errors
##            Estimate   Std. Error    t value     Pr(>|t|)
## mu     0.0006678893 0.0000947508  7.0489065 0.0000000000
## omega  0.0000025218 0.0000042624  0.5916353 0.5540948315
## alpha1 0.1243765581 0.0228482046  5.4436031 0.0000000522
## beta1  0.8552415698 0.0383271265 22.3142627 0.0000000000
##
## Coefficients of GJR GARCH with normal distribution of errors
##            Estimate   Std. Error      t value    Pr(>|t|)
## mu     0.0002606009 0.0000979826 2.659665e+00 0.007821837
## omega  0.0000024778 0.0000002397 1.033524e+01 0.000000000
## alpha1 0.0000209156 0.0033010953 6.335971e-03 0.994944661
## beta1  0.8729056647 0.0066451863 1.313591e+02 0.000000000
## gamma1 0.2065843904 0.0129819001 1.591326e+01 0.000000000
##
## Robust coefficients of GJR GARCH with normal distribution of errors
##            Estimate   Std. Error      t value     Pr(>|t|)
## mu     0.0002606009 0.0001316477 1.979533e+00 0.0477560492
## omega  0.0000024778 0.0000005923 4.183009e+00 0.0000287676
## alpha1 0.0000209156 0.0140679940 1.486754e-03 0.9988137426
## beta1  0.8729056647 0.0072181000 1.209329e+02 0.0000000000
## gamma1 0.2065843904 0.0314439994 6.569915e+00 0.0000000001
##
## Coefficients of Standard GARCH with skewed Student t distribution of errors
##            Estimate  Std. Error    t value     Pr(>|t|)
## mu     0.0006064283 0.000111164  5.4552561 0.0000000489
## omega  0.0000015080 0.000001644  0.9172266 0.3590239141
## alpha1 0.1220516502 0.027618551  4.4191910 0.0000099071
## beta1  0.8703537793 0.026023588 33.4448033 0.0000000000
## skew   0.8881626235 0.018172634 48.8736316 0.0000000000
## shape  7.1207987301 1.039608073  6.8495031 0.0000000000
##
## Robust coefficients of Standard GARCH with skewed Student t distribution of errors
##            Estimate   Std. Error    t value     Pr(>|t|)
## mu     0.0006064283 0.0001297724  4.6730161 0.0000029681
## omega  0.0000015080 0.0000119600  0.1260838 0.8996656146
## alpha1 0.1220516502 0.1884915577  0.6475179 0.5172968387
## beta1  0.8703537793 0.1798182388  4.8401863 0.0000012972
## skew   0.8881626235 0.0435211859 20.4075924 0.0000000000
## shape  7.1207987301 5.2846067667  1.3474605 0.1778319757
##
## Coefficients of GJR GARCH with skewed Student t distribution of errors
##            Estimate   Std. Error      t value   Pr(>|t|)
## mu     0.0002498529 0.0001176175 2.124284e+00 0.03364643
## omega  0.0000020128 0.0000010605 1.897992e+00 0.05769708
## alpha1 0.0000000821 0.0127379255 6.447100e-06 0.99999486
## beta1  0.8732435063 0.0108249601 8.066944e+01 0.00000000
## gamma1 0.2244263082 0.0306579443 7.320331e+00 0.00000000
## skew   0.8542197386 0.0168737583 5.062415e+01 0.00000000
## shape  8.2660071635 0.9890413211 8.357595e+00 0.00000000
##
## Robust coefficients of GJR GARCH with skewed Student t distribution of errors
##            Estimate   Std. Error      t value   Pr(>|t|)
## mu     0.0002498529 0.0002844967 8.782277e-01 0.37982015
## omega  0.0000020128 0.0000055599 3.620113e-01 0.71734359
## alpha1 0.0000000821 0.0442751785 1.854800e-06 0.99999852
## beta1  0.8732435063 0.0324759355 2.688894e+01 0.00000000
## gamma1 0.2244263082 0.1250738277 1.794351e+00 0.07275721
## skew   0.8542197386 0.0165412469 5.164180e+01 0.00000000
## shape  8.2660071635 1.0141193171 8.150922e+00 0.00000000
```

```
# Models comparing
model_comparison <- data.frame()

for (f in 1:length(garch_fits)) {
    temp <- data.frame()
    temp[1,1] <- likelihood(garch_fits[[f]])
    inf_criterion <- infocriteria(garch_fits[[f]])

    temp <- rbind(temp, inf_criterion)

    model_comparison <- c(model_comparison, temp)
}
model_comparison <- as.data.frame(model_comparison)

rownames(model_comparison) <- c(’Likelihood’,rownames(inf_criterion))
colnames(model_comparison) <- short_model_names
model_comparison
```

```
##                Normal GARCH   Normal GJR Skewed t GARCH Skewed t GJR
## Likelihood   15464.113912 15571.523249   15577.701243 15677.651493
## Akaike          -6.505413    -6.550189      -6.552367    -6.594004
## Bayes           -6.499971    -6.543386      -6.544204    -6.584481
## Shibata         -6.505414    -6.550191      -6.552371    -6.594009
## Hannan-Quinn    -6.503501    -6.547798      -6.549499    -6.590658
```

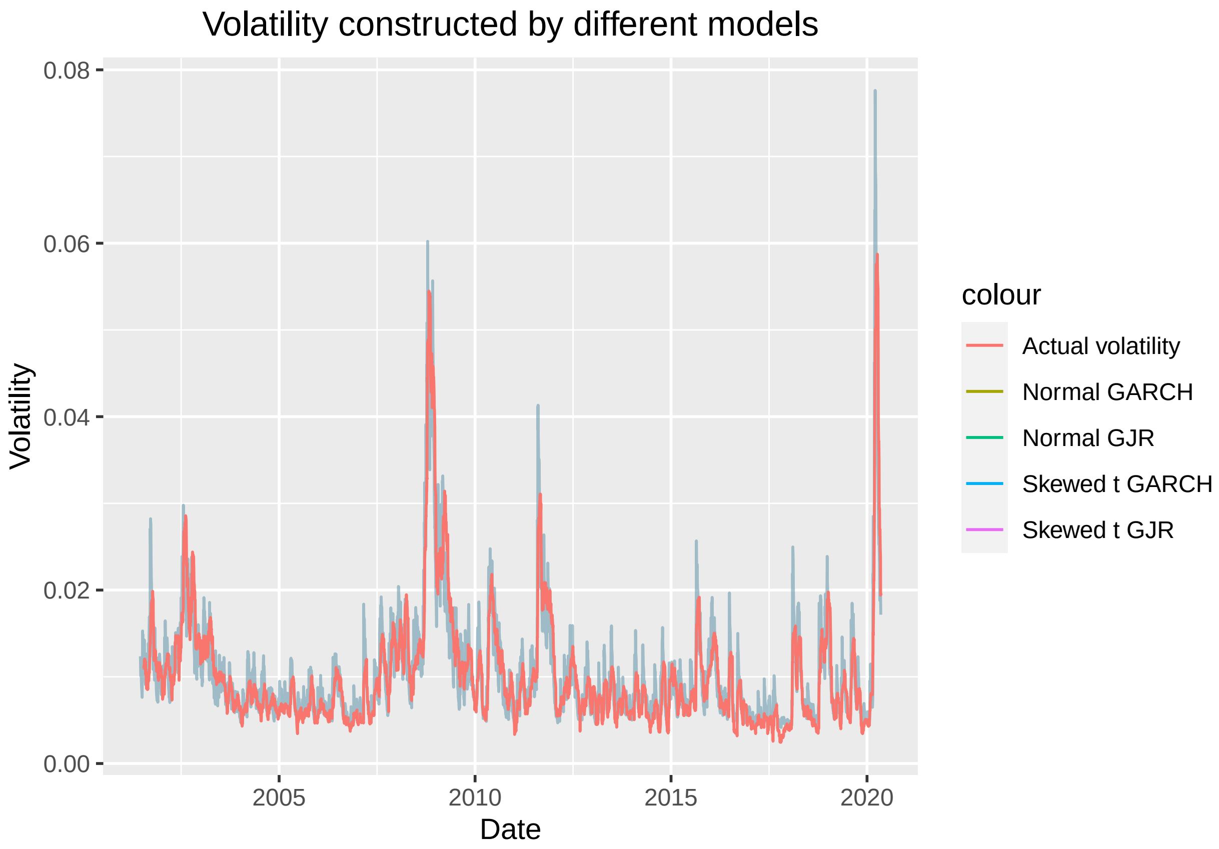
```
rm(f, temp, inf_criterion, model_comparison)
```

```
# Visualizing impact of negative previous return on variance
p <- ggplot()

for (f in 1:length(garch_fits)) {
  garch_news <- as.data.frame(newsimpact(garch_fits[[f]])[1:2])

  model_name <- short_model_names[f]
  model_name <- enquo(model_name)

  p <- p + geom_line(data = garch_news,
            aes(x = zx, y = zy, color = !!model_name))
}

p <- p + labs(x = ’Error’, y = ’Variance’,
       title = ’Dependence of variance on errors in different models’) +
  theme(plot.title = element_text(hjust = 0.5))

print(p)
```



```
rm(p, model_name, garch_news)

# 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')) +
```

```
# Visualizing volatility
p <- ggplot()
garch_vol <- list()
for (f in 1:length(garch_fits)) {
  garch_vol[[f]] <- sigma(garch_fits[[f]])

  model_name <- short_model_names[f]
  model_name <- enquo(model_name)
  p <- p + geom_line(data = garch_vol[[f]], aes(x = index(garch_vol[[f]][,1]),
                                       y = garch_vol[[f]][,1],
                                       color = !!model_name), alpha = 0.2)
}
names(garch_vol) <- short_model_names

p <- p + 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, f, model_name)
```

```
# Predicting volatility for n.ahead periods
predict_results <- data.frame(fund_tail_volatility)
garch_sst <- c()
for (f in 1:length(garch_fits)) {
  garch_forecast <- ugarchforecast(fitORspec = garch_fits[[f]],
                                      data = garch_vol[[f]], n.ahead = 10)

  predict_results <- cbind(predict_results, sigma(garch_forecast))

  garch_sst[[f]] <- sum(fund_tail_volatility - sigma(garch_forecast))
```

```r
  names(garch_sst)[f] <- paste('TES for',short_model_names[f])

}
names(predict_results)[2:ncol(predict_results)] <- short_model_names
```

```r
# Total error sum for models
garch_sst
```

```
##   TES for Normal GARCH     TES for Normal GJR TES for Skewed t GARCH
##             0.08050812             0.09724891             0.06188808
##   TES for Skewed t GJR
##             0.09188936
```

```r
#Comparing predicted volatility for models with actual one
predict_results
```

```
##              Vanguard Normal GARCH Normal GJR Skewed t GARCH Skewed t GJR
## 2020-04-27 0.02937419   0.01723897 0.01557547     0.01879428   0.01610753
## 2020-04-28 0.02737708   0.01713613 0.01546945     0.01876301   0.01600320
## 2020-04-29 0.02656118   0.01703477 0.01536525     0.01873192   0.01590036
## 2020-04-30 0.02637500   0.01693490 0.01526283     0.01870101   0.01579900
## 2020-05-01 0.02699270   0.01683649 0.01516219     0.01867029   0.01569910
## 2020-05-04 0.02449756   0.01673952 0.01506329     0.01863975   0.01560065
## 2020-05-05 0.02430946   0.01664398 0.01496611     0.01860940   0.01550362
## 2020-05-06 0.02389937   0.01654985 0.01487063     0.01857922   0.01540801
## 2020-05-07 0.01941791   0.01645712 0.01477682     0.01854923   0.01531380
## 2020-05-08 0.01964116   0.01636577 0.01468467     0.01851942   0.01522096
```

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

```r
# 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)
```