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)
```

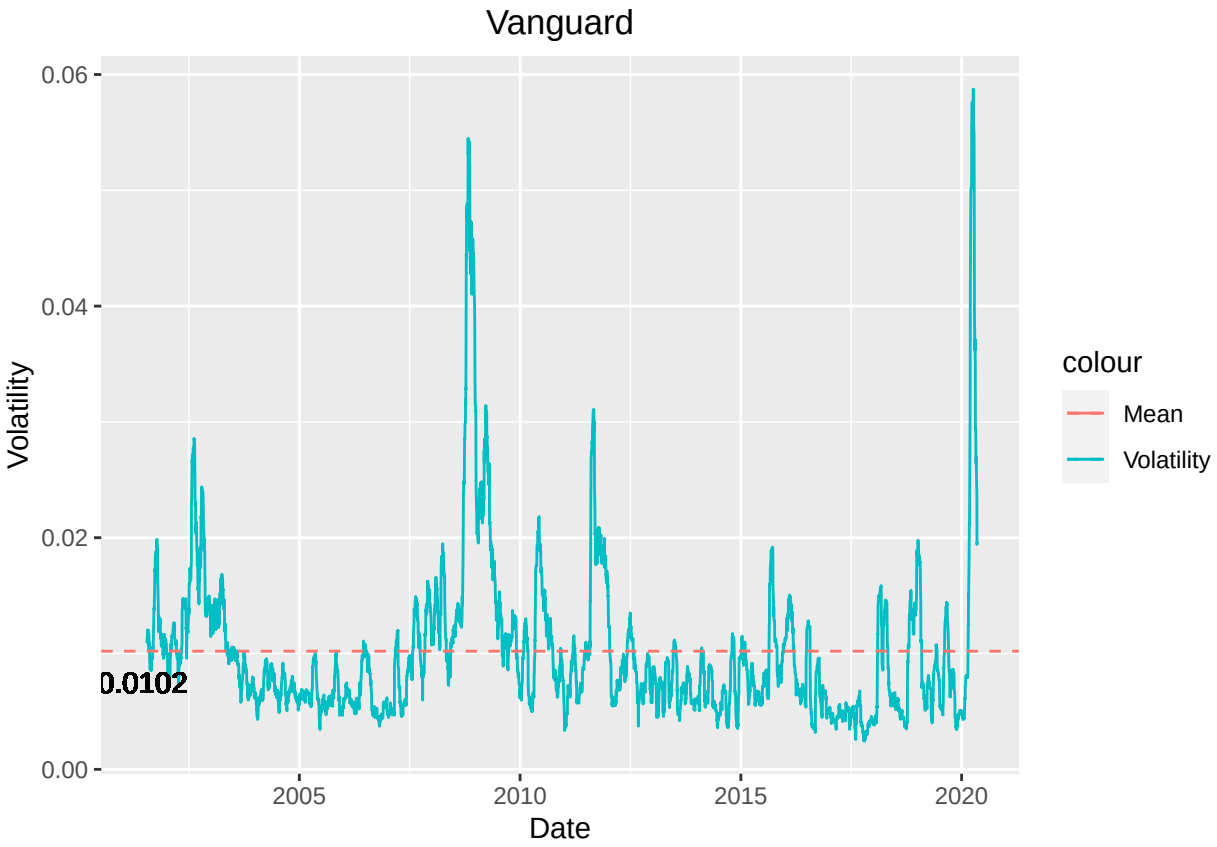
```
read_funds <- function(lf) {
  dfs <- list()
 browser()
  for (f in 1:length(lf)) {
      cond <- substr(lf[f], nchar(lf[f])-3, nchar(lf[f])) == '.csv'
      if (cond) {
      temp <- data.frame(read.csv(lf[f], stringsAsFactors = FALSE))
      temp$Date <- as.Date(temp$Date)
      temp$Close <- as.numeric(temp$Close)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

  return(dfs)
}

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

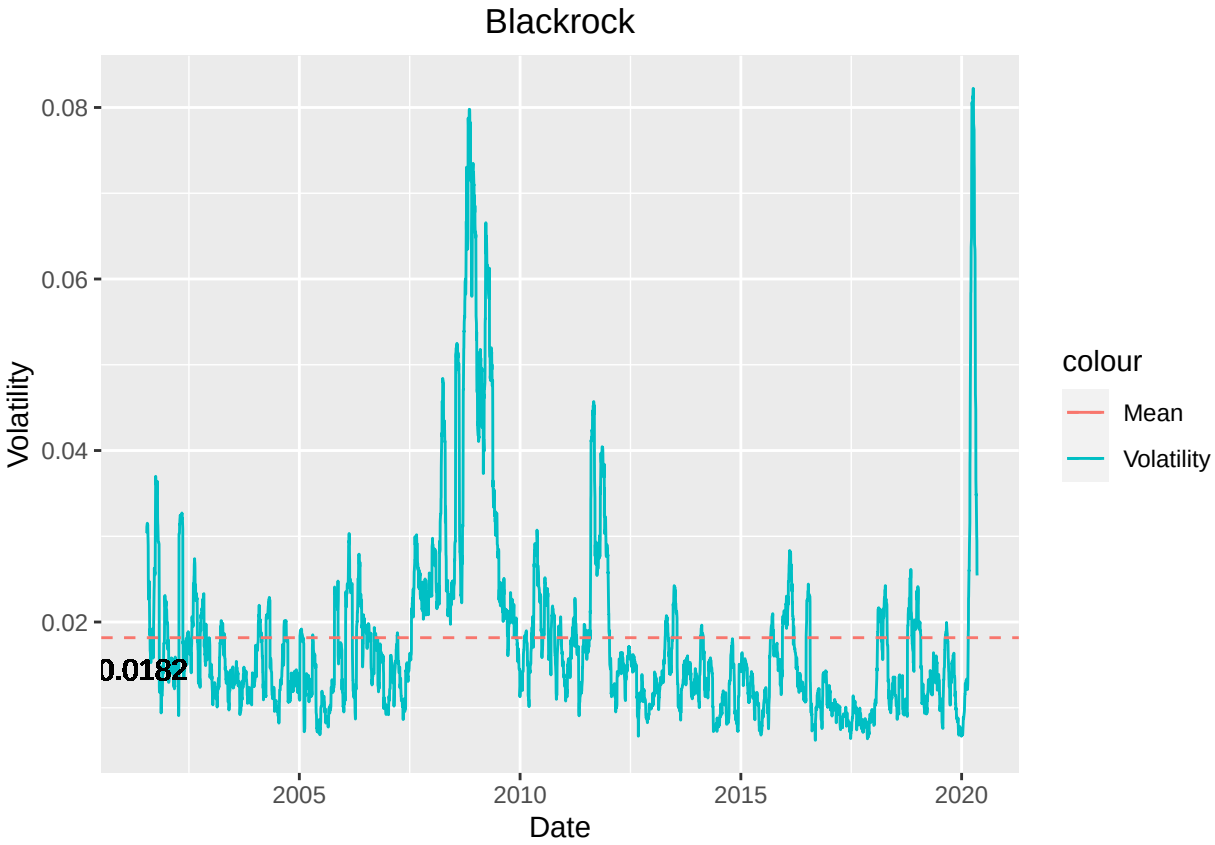
```
# Data loading
df <- read.csv('Funds.csv', stringsAsFactors = F)
df <- df[,-1]
df$Date <- as.Date(df$Date)

# 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']
```

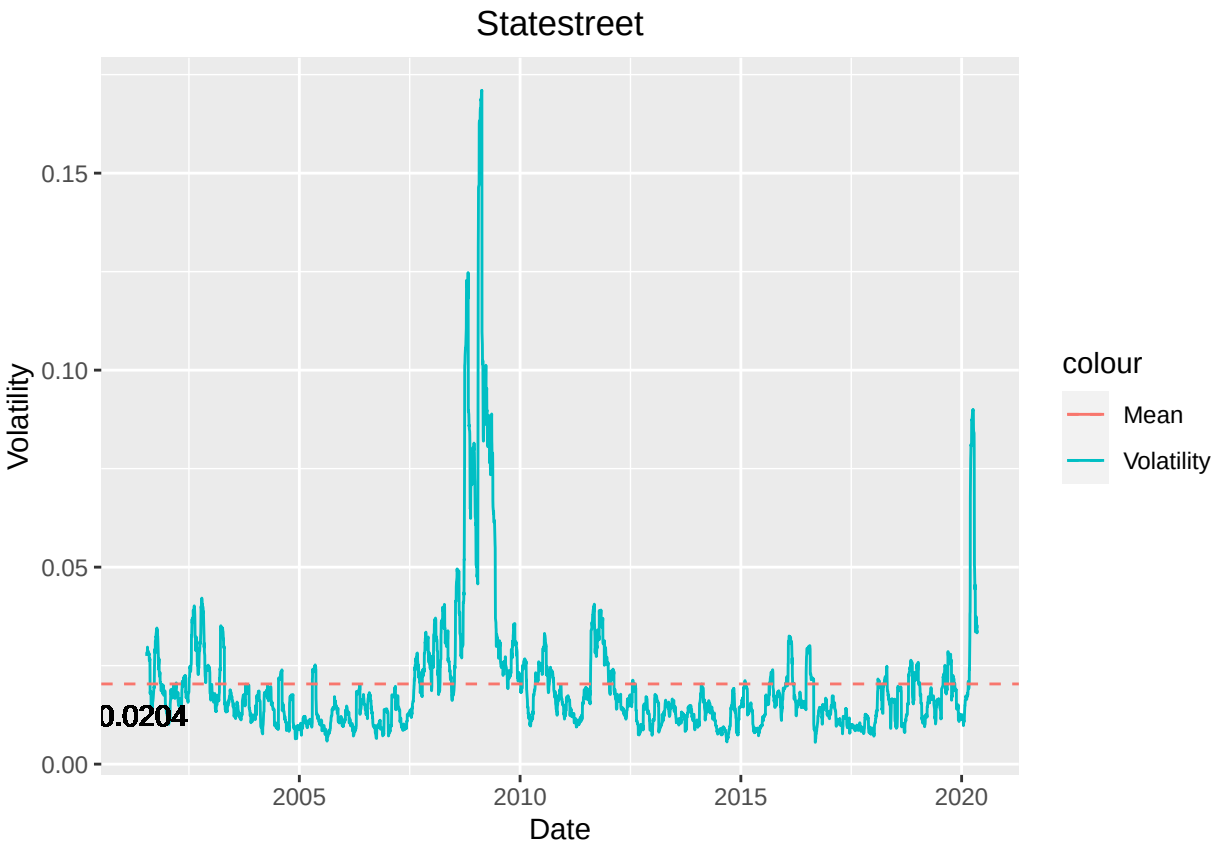
```
allianz <- funds[,1]


allianz_ret_pa <- CalculateReturns(allianz)[-1]
```

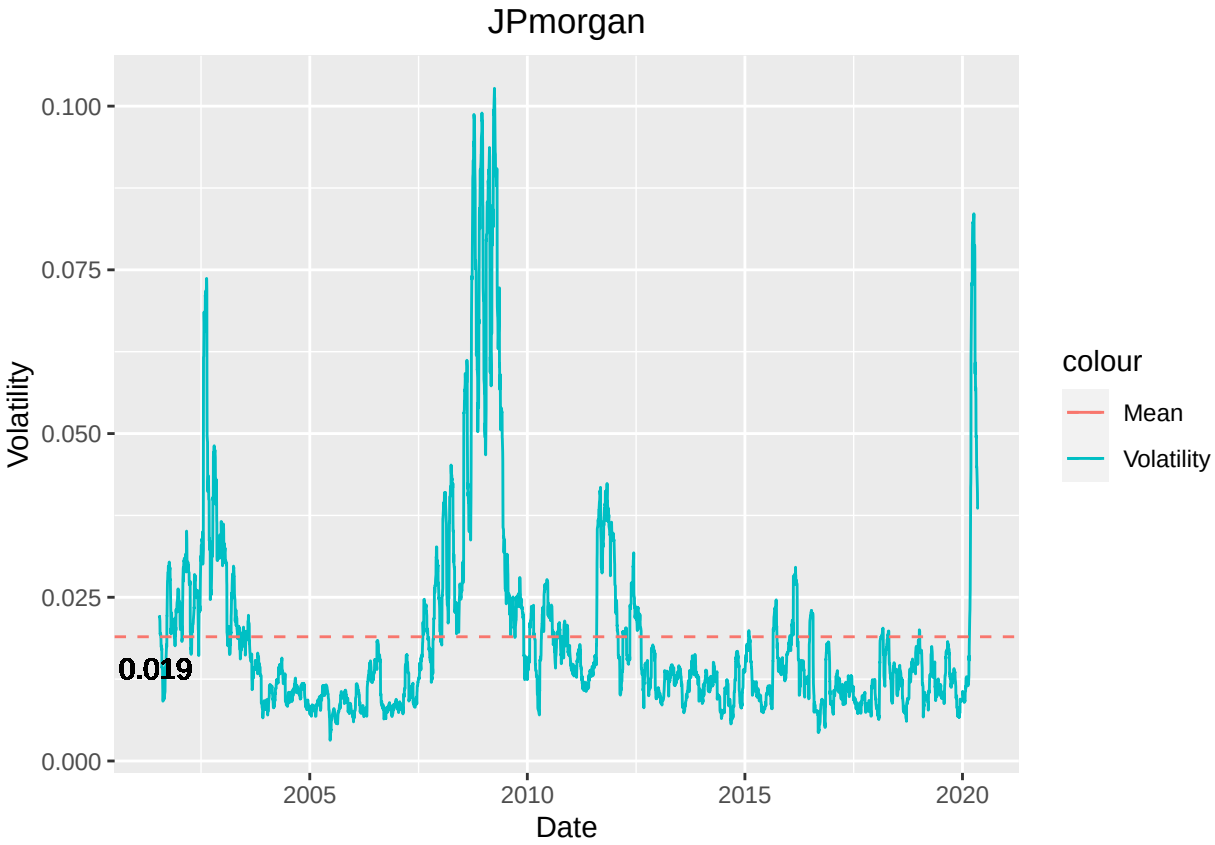
```
get_volatiles <- function(funds, width = 22, time_scale = 1, funds_names) {
    vol_df <- data.frame()

    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)
  }
}
```

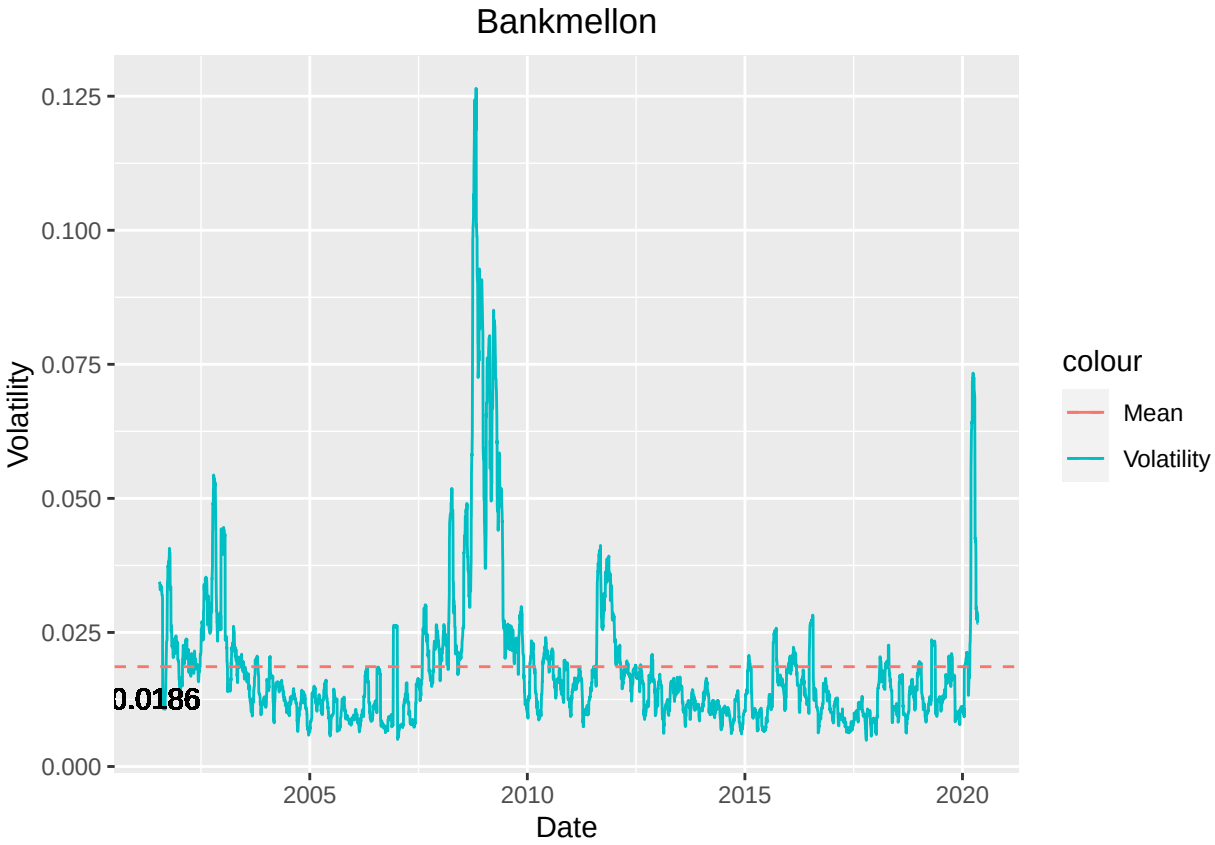
```
vol_df <- get_volatiles(funds, funds_names =  funds_names)

visualize_funds_lines(vol_df)
```

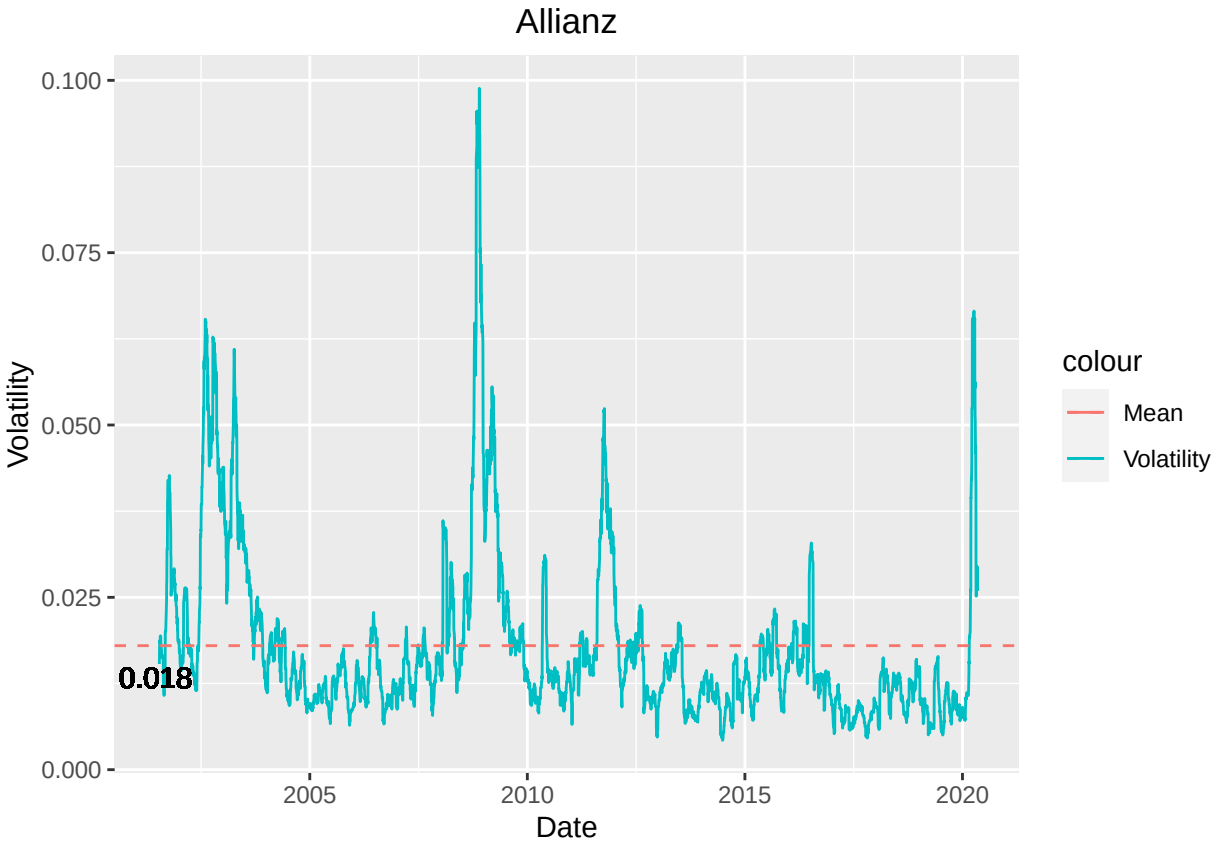



Blackrock
Volatility
0.08
0.06
0.04
0.02
0.0182
2005
2010
2015
2020
Date
colour
Mean
Volatility

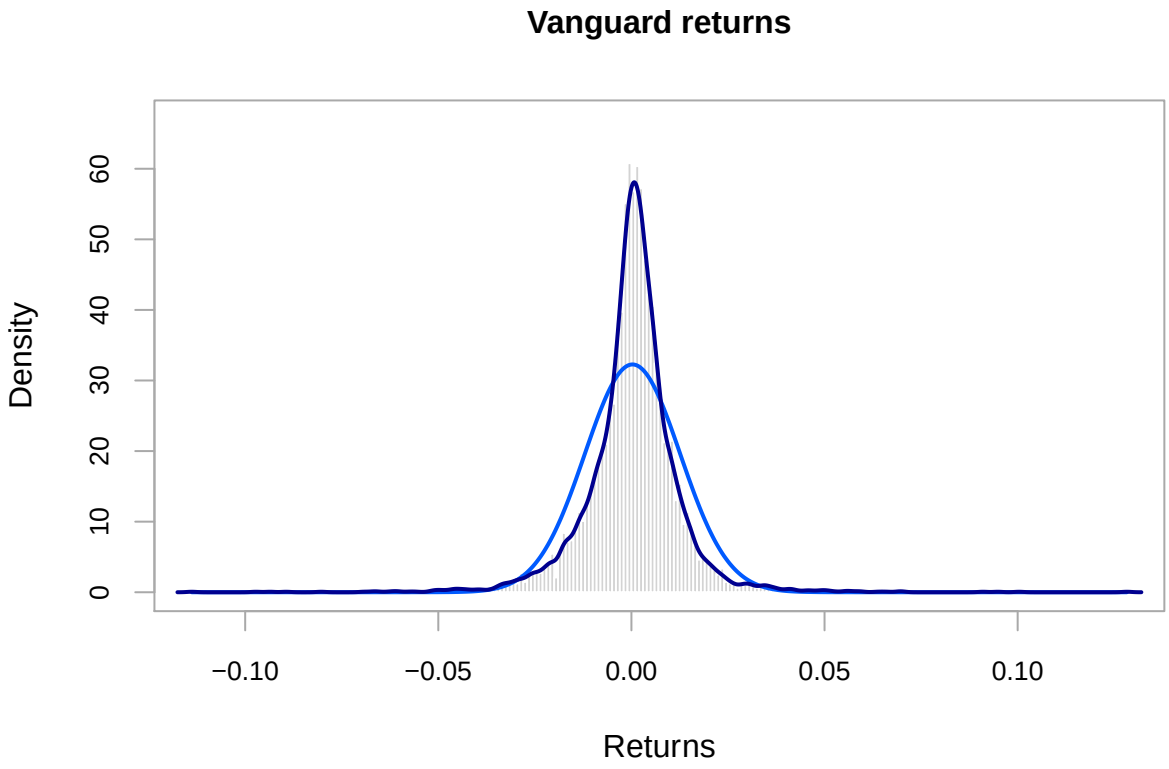

Statestreet
Volatility
0.15
0.10
0.05
0.00
0.0204
2005
2010
2015
2020
Date
colour
Mean
Volatility

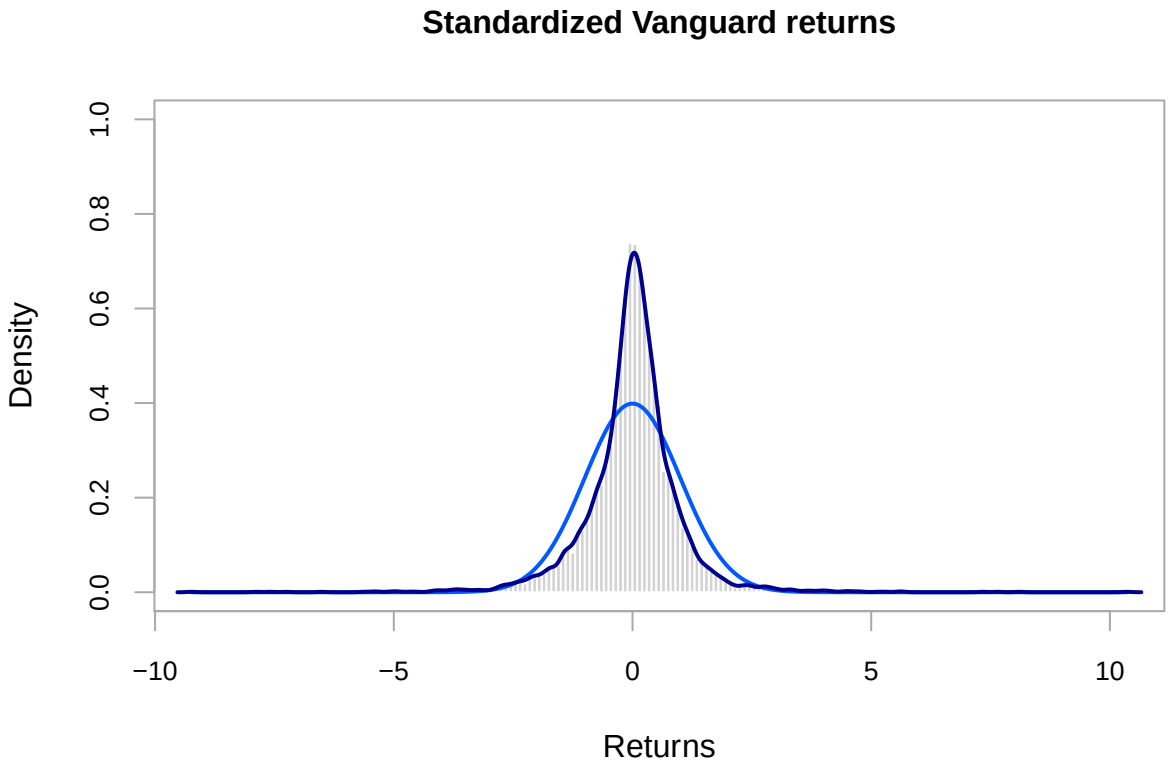

JPmorgan
0.100
0.075
0.050
0.025
0.000
Volatility
0.019
2005
2010
2015
2020
Date
colour
Mean
Volatility

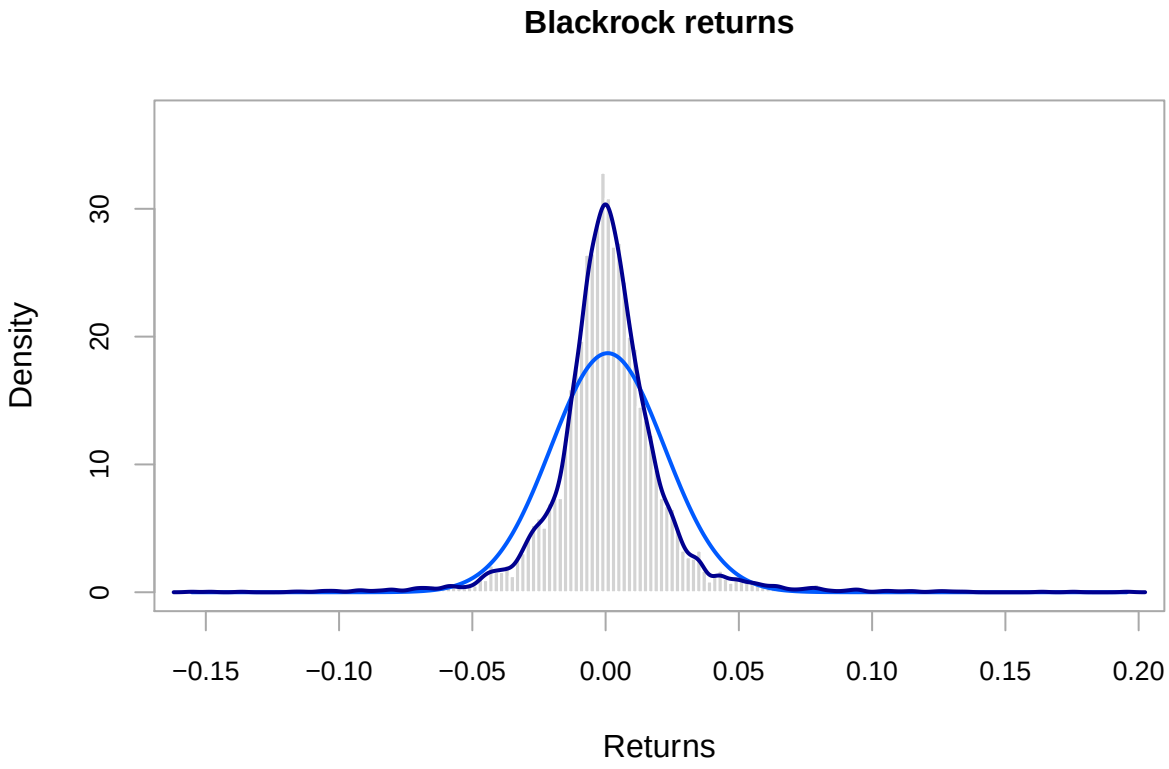

Bankmellon
Volatility
0.125
0.100
0.075
0.050
0.025
0.000
0.0186
2005
2010
2015
2020
Date
colour
Mean
Volatility

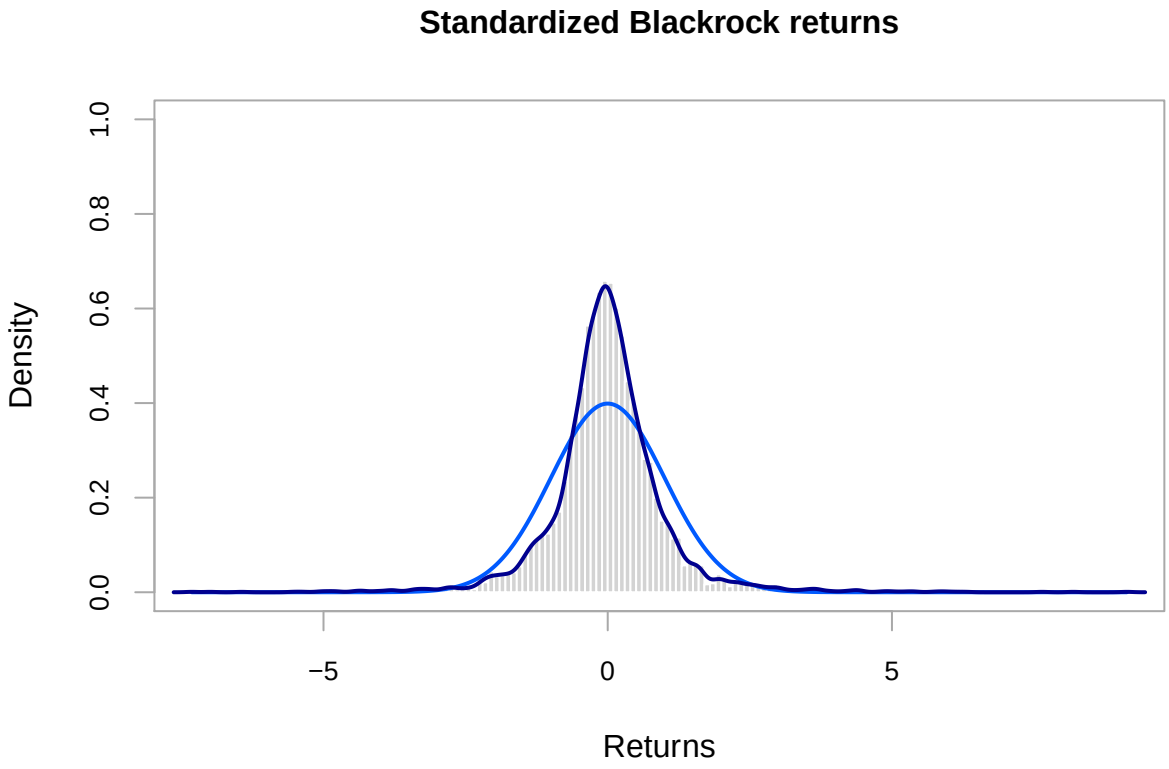


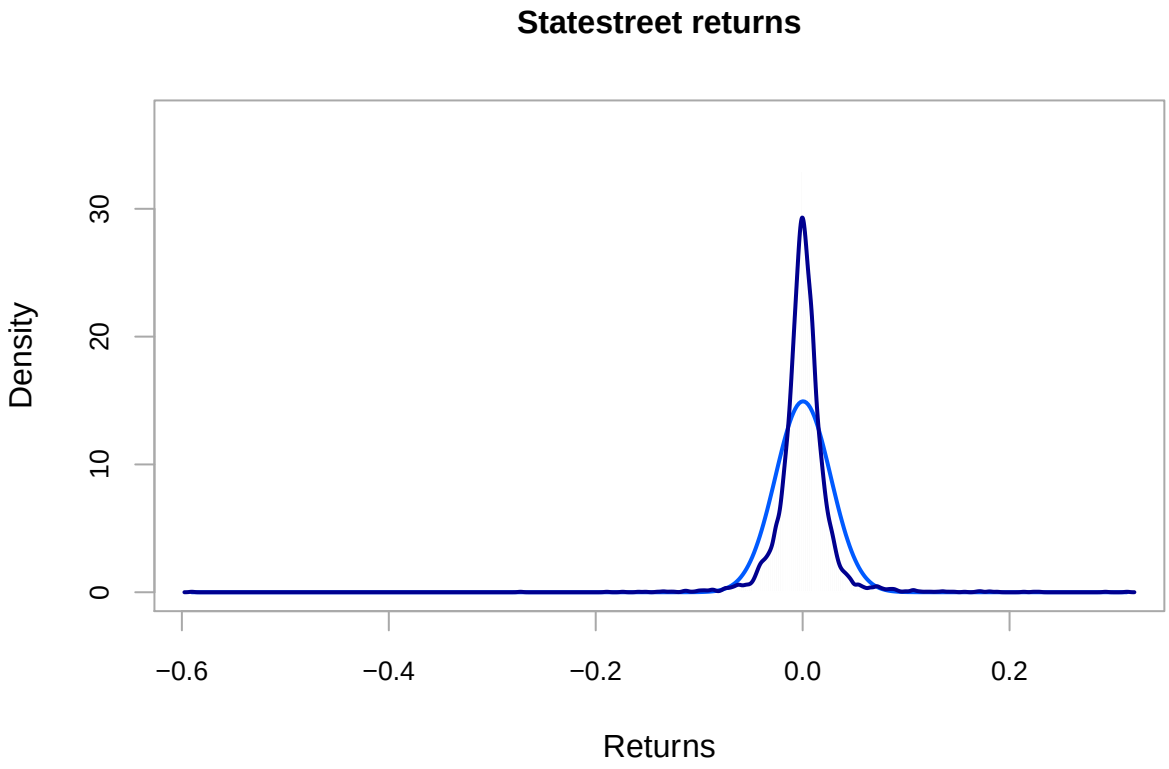

```
visualize_funds_hist(funds)
```

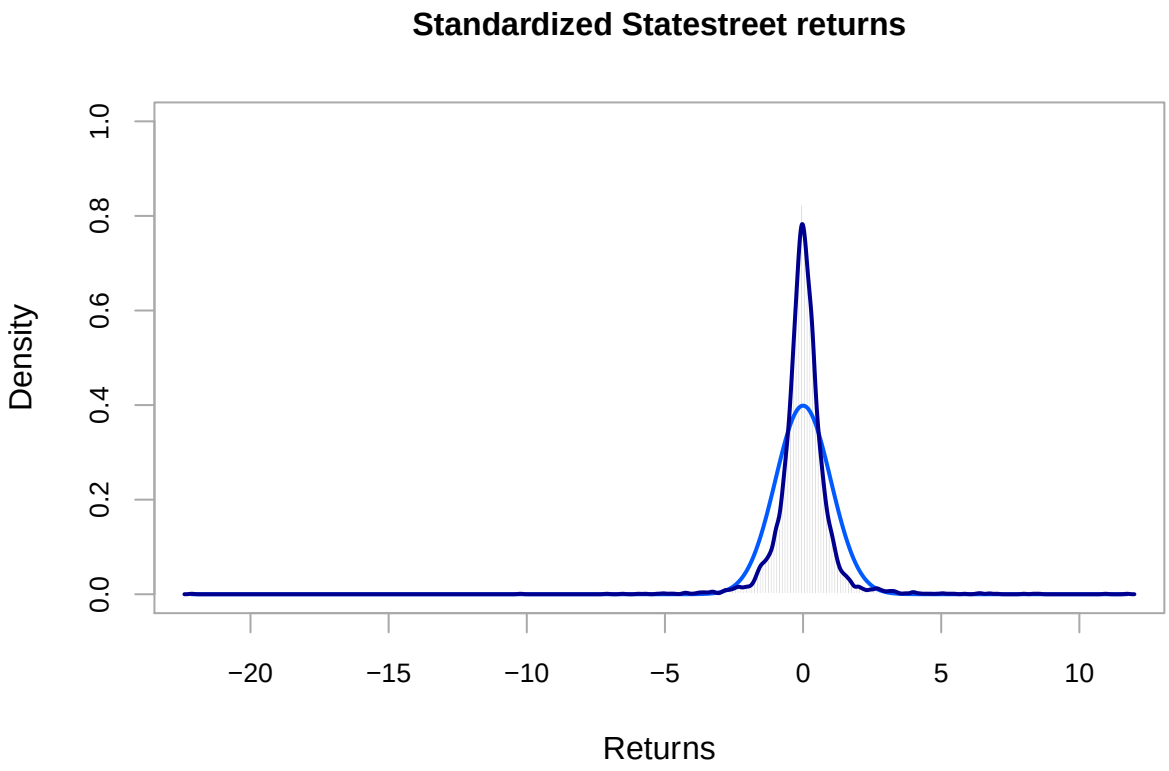

Vanguard returns
Density
0
10
20
30
40
50
60
−0.10
−0.05
0.00
0.05
0.10
Returns

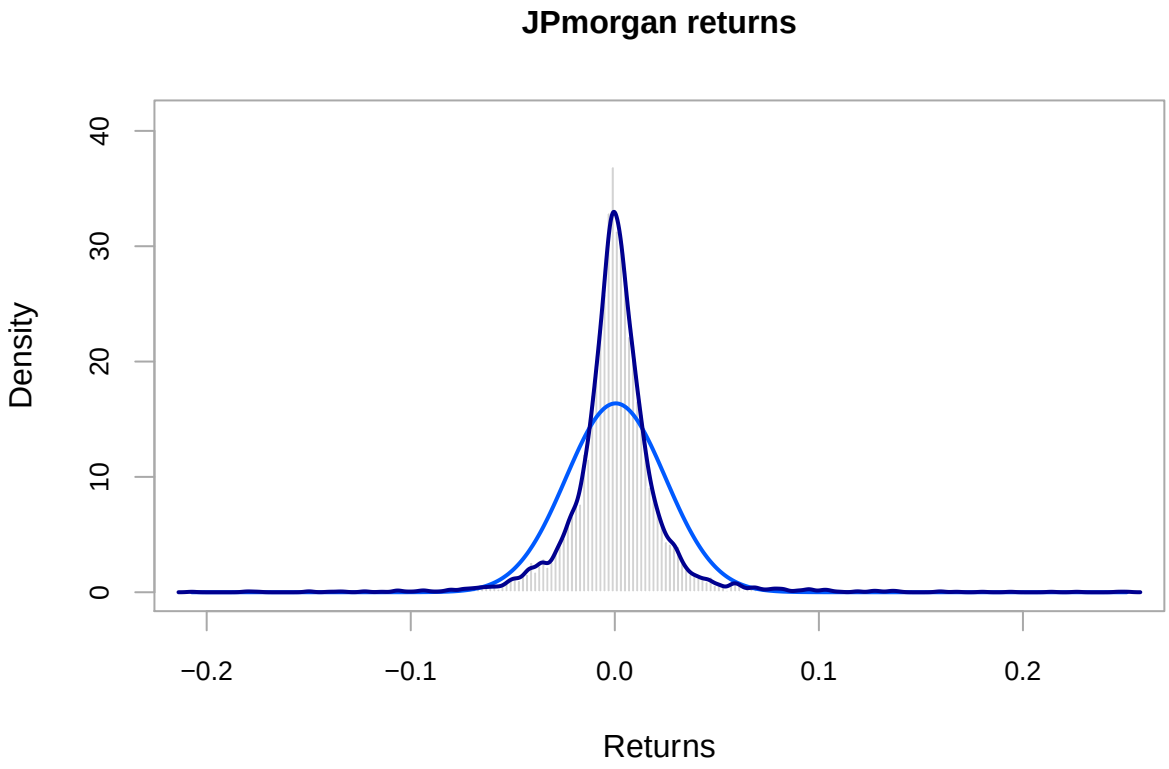

Standardized Vanguard returns
Density
0.0
0.2
0.4
0.6
0.8
1.0
−10
−5
0
5
10
Returns

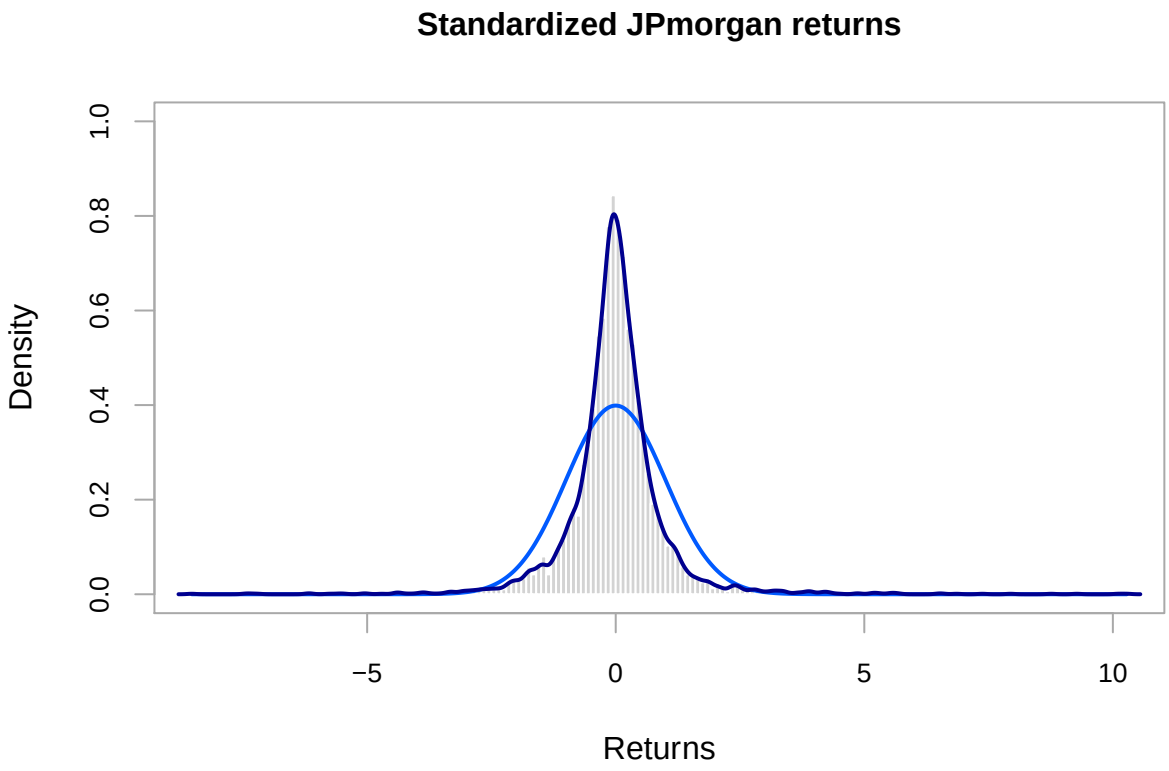

Blackrock returns
Density
0
10
20
30
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
0.20
Returns

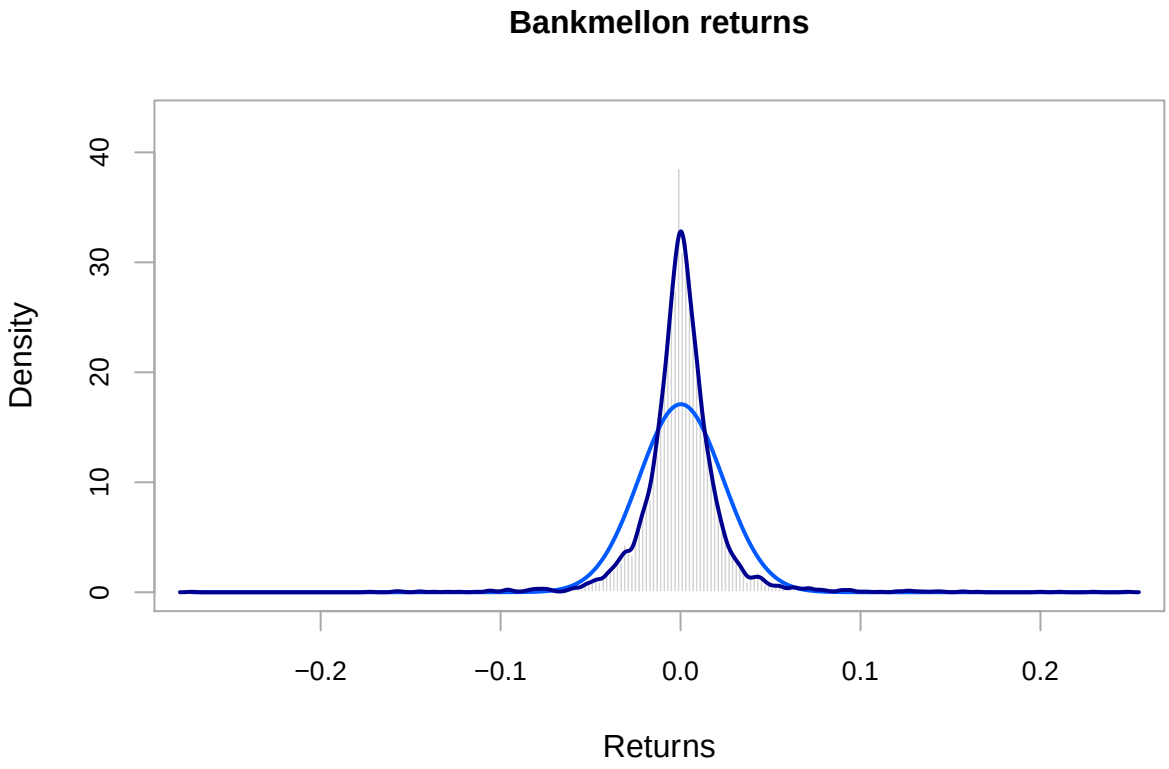

Standardized Blackrock returns
Density
0.0
0.2
0.4
0.6
0.8
1.0
−5
0
5
Returns

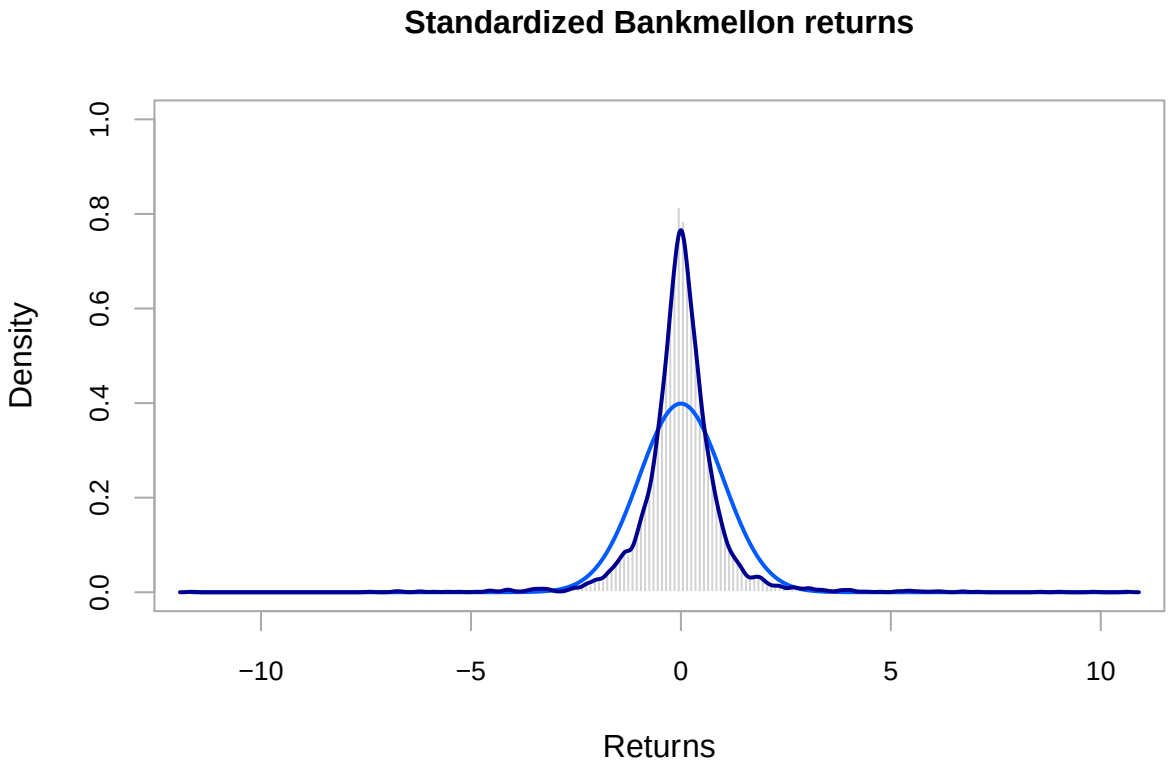

Statestreet returns
Density
0
10
20
30
−0.6
−0.4
−0.2
0.0
0.2
Returns

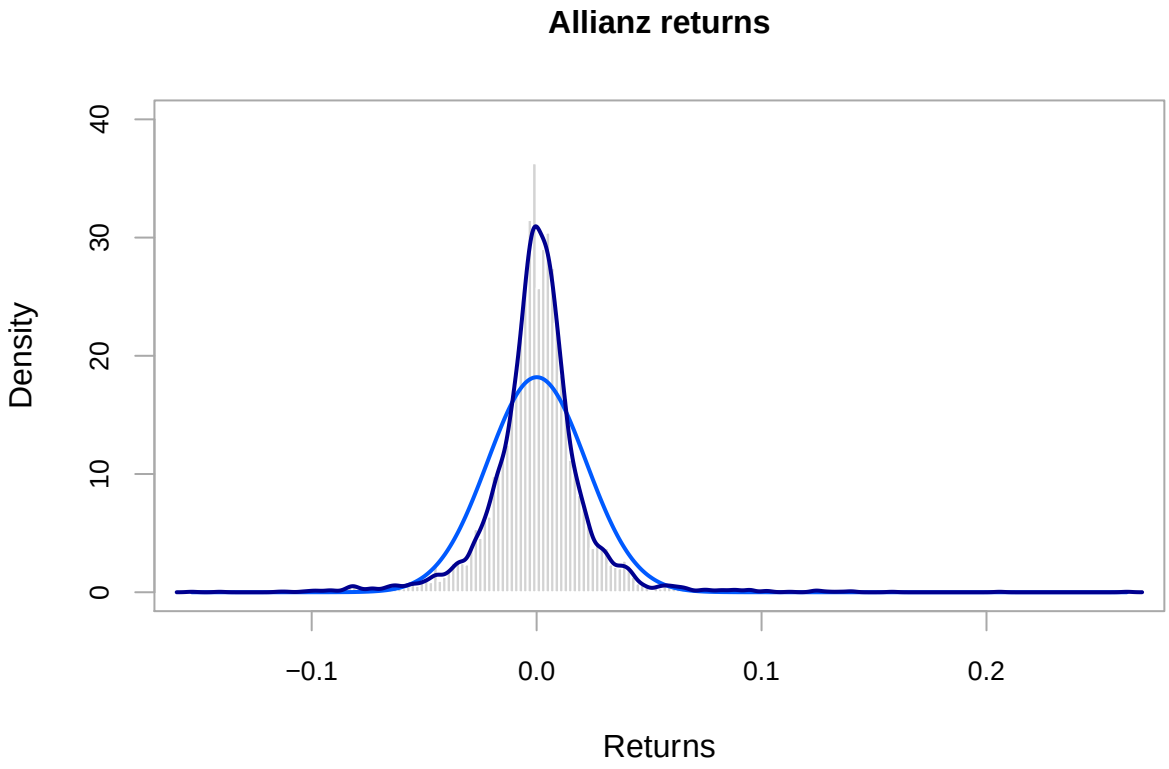

Standardized Statestreet returns
Density
0.0
0.2
0.4
0.6
0.8
1.0
−20
−15
−10
−5
0
5
10
Returns

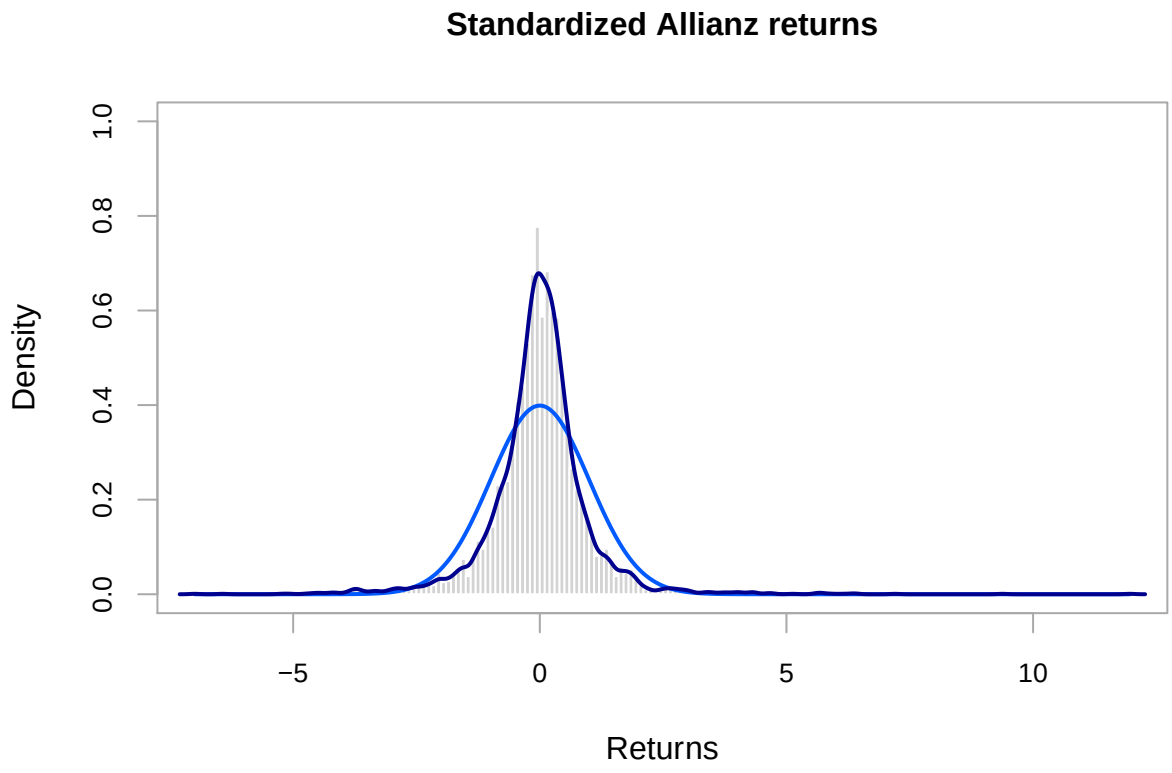

JPmorgan returns
Density
0
10
20
30
40
−0.2
−0.1
0.0
0.1
0.2
Returns


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 GARCH model
# The most important is distribution.model - we are specifying type of distribution
norm_garch_spec <- ugarchspec(mean.model = list(armaOrder = c(0,0)),
                      variance.model = list(model = 'sGARCH'),
                      distribution.model = 'norm')

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

# Apply GARCH model to our data
```

```r
van_fit <- ugarchfit(data = funds_red$vanguard_return,
                     spec = norm_garch_spec)

van_fit
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000666    0.000113   5.8727 0.000000
## omega   0.000002    0.000001   3.1680 0.001535
## alpha1  0.123656    0.009874  12.5235 0.000000
## beta1   0.856401    0.010647  80.4390 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000666    0.000095  7.04176   0.0000
## omega   0.000002    0.000004  0.57278   0.5668
## alpha1  0.123656    0.024258  5.09747   0.0000
## beta1   0.856401    0.039710 21.56629   0.0000
##
## LogLikelihood : 15447.87
##
## Information Criteria
## ------------------------------------
##
## Akaike       -6.5068
## Bayes        -6.5013
## Shibata      -6.5068
## Hannan-Quinn -6.5049
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      2.005  0.1568
## Lag[2*(p+q)+(p+q)-1][2]     2.484  0.1937
## Lag[4*(p+q)+(p+q)-1][5]     4.050  0.2479
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      1.630  0.2017
```

```
## Lag[2*(p+q)+(p+q)-1][5]     5.063  0.1481
## Lag[4*(p+q)+(p+q)-1][9]     6.377  0.2571
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.9874 0.500 2.000  0.3204
## ARCH Lag[5]    2.7887 1.440 1.667  0.3219
## ARCH Lag[7]    3.2192 2.315 1.543  0.4732
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  25.5724
## Individual Statistics:
## mu     0.1622
## omega  1.3709
## alpha1 0.3002
## beta1  0.9171
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.07 1.24 1.6
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value      prob sig
## Sign Bias           3.4799 5.061e-04 ***
## Negative Sign Bias  0.4792 6.318e-01
## Positive Sign Bias  2.1302 3.321e-02  **
## Joint Effect       35.3189 1.043e-07 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     163.1    6.219e-25
## 2    30     192.3    5.102e-26
## 3    40     222.2    1.683e-27
## 4    50     239.6    6.404e-27
##
##
## Elapsed time : 0.3860896
```

```
van_vol <- sigma(van_fit)

# Predicting volatility for n.ahead periods
van_forecast <- ugarchforecast(fitORspec = van_fit,
                               data = van_vol, n.ahead = 10)
van_forecast
```

```
##
## *----------------------------------*
## *       GARCH Model Forecast       *
## *----------------------------------*
```

```
## Model: sGARCH
## Horizon: 10
## Roll Steps: 0
## Out of Sample: 0
##
## 0-roll forecast [T0=2020-04-30]:
##         Series   Sigma
## T+1  0.0006661 0.02117
## T+2  0.0006661 0.02102
## T+3  0.0006661 0.02087
## T+4  0.0006661 0.02072
## T+5  0.0006661 0.02057
## T+6  0.0006661 0.02043
## T+7  0.0006661 0.02028
## T+8  0.0006661 0.02014
## T+9  0.0006661 0.02000
## T+10 0.0006661 0.01986
```

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

```
##      2020-04-30
## T+1  0.02117248
## T+2  0.02101937
## T+3  0.02086822
## T+4  0.02071902
## T+5  0.02057174
## T+6  0.02042636
## T+7  0.02028288
## T+8  0.02014126
## T+9  0.02000150
## T+10 0.01986357
```

```
# distribution.model: sstd - Skewed Student t Distribution
van_sstd_spec <- ugarchspec(mean.model = list(armaOrder = c(0,0)),
                            variance.model = list(model = 'sGARCH'),
                            distribution.model = 'sstd')

van_sstd_fit <- ugarchfit(data = funds_red$vanguard_return,
                          spec = van_sstd_spec)

van_sstd_fit
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : sstd
##
## Optimal Parameters
```

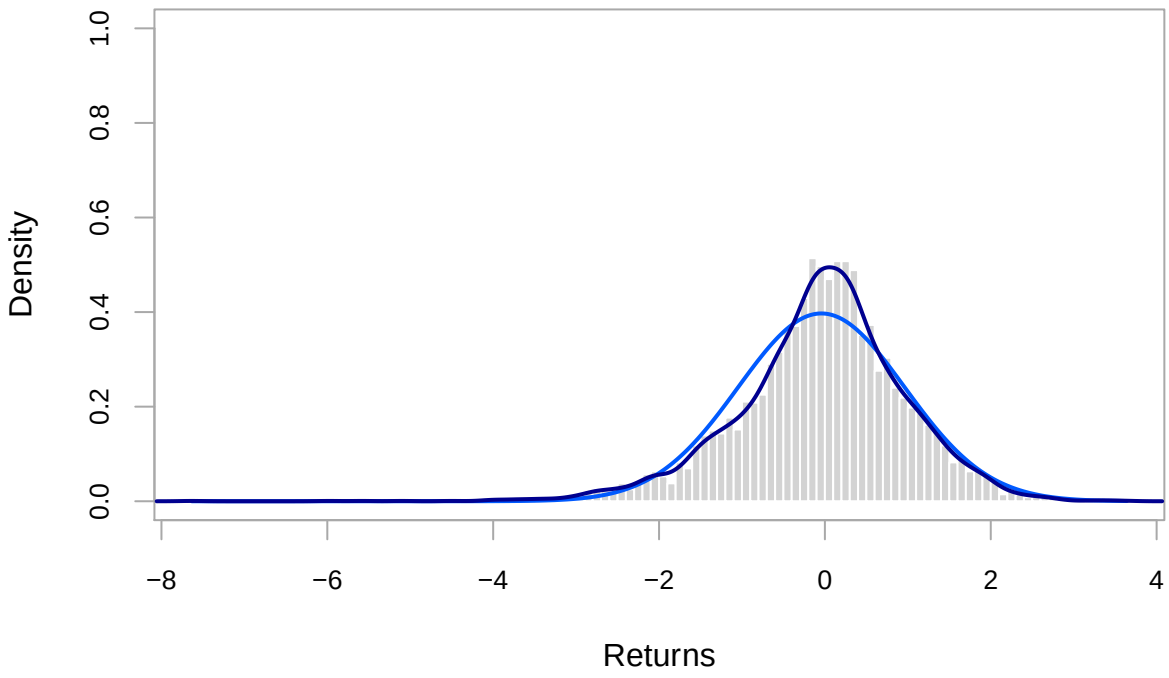
```
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000606    0.000112  5.42481 0.000000
## omega   0.000001    0.000002  0.79416 0.427104
## alpha1  0.120933    0.031350  3.85748 0.000115
## beta1   0.871791    0.029543 29.50895 0.000000
## skew    0.888821    0.018548 47.91960 0.000000
## shape   7.108462    1.118777  6.35378 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.000606    0.000164  3.691612 0.000223
## omega   0.000001    0.000016  0.095045 0.924279
## alpha1  0.120933    0.249618  0.484472 0.628051
## beta1   0.871791    0.237367  3.672751 0.000240
## skew    0.888821    0.057521 15.452063 0.000000
## shape   7.108462    6.988717  1.017134 0.309090
##
## LogLikelihood : 15561.41
##
## Information Criteria
## ------------------------------------
##
## Akaike       -6.5538
## Bayes        -6.5456
## Shibata      -6.5538
## Hannan-Quinn -6.5509
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      1.916  0.1663
## Lag[2*(p+q)+(p+q)-1][2]     2.398  0.2045
## Lag[4*(p+q)+(p+q)-1][5]     3.978  0.2568
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.7925  0.3733
## Lag[2*(p+q)+(p+q)-1][5]    3.3154  0.3525
## Lag[4*(p+q)+(p+q)-1][9]    4.7396  0.4686
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.4842 0.500 2.000  0.4865
## ARCH Lag[5]    2.0708 1.440 1.667  0.4558
## ARCH Lag[7]    2.9682 2.315 1.543  0.5196
##
## Nyblom stability test
## ------------------------------------
```

```
## Joint Statistic:  159.4704
## Individual Statistics:
## mu      0.08331
## omega  25.62030
## alpha1  0.58024
## beta1   1.44046
## skew    0.28835
## shape   1.34956
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.49 1.68 2.12
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value      prob sig
## Sign Bias           3.4681 5.287e-04 ***
## Negative Sign Bias  0.9704 3.319e-01
## Positive Sign Bias  2.4059 1.617e-02  **
## Joint Effect       35.6071 9.067e-08 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     58.70    6.201e-06
## 2    30     68.94    4.222e-05
## 3    40     75.25    4.348e-04
## 4    50     92.51    1.714e-04
##
##
## Elapsed time : 1.383584
```

```
van_sstd_vol <- sigma(van_sstd_fit)
tail(van_sstd_vol, 10)
```

```
##                  [,1]
## 2020-04-17 0.03301709
## 2020-04-20 0.03232061
## 2020-04-21 0.03076295
## 2020-04-22 0.03078248
## 2020-04-23 0.02976416
## 2020-04-24 0.02781776
## 2020-04-27 0.02647300
## 2020-04-28 0.02545337
## 2020-04-29 0.02383211
## 2020-04-30 0.02428053
```

```
van_sstd_forecast <- ugarchforecast(fitORspec = van_sstd_fit,
                                    data = van_sstd_vol, n.ahead = 10)

van_sstd_res <- sigma(van_sstd_forecast) #* sqrt(252)

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




```
cat('SST for norm\n')
```

```
## SST for norm
```

```
sum(vanguard_tail_volatility - van_norm_res)
```

```
## [1] 0.0433792
```

```
cat('\nSST for sstd\n')
```

```
##
## SST for sstd
```

```
sum(vanguard_tail_volatility - van_sstd_res)
```

```
## [1] 0.01887305
```

```
cbind(vanguard_tail_volatility, van_norm_res, van_sstd_res)
```

```
##              Vanguard X2020.04.30 X2020.04.30.1
## 2020-04-27 0.02937419  0.02117248    0.02319064
## 2020-04-28 0.02737708  0.02101937    0.02313808
## 2020-04-29 0.02656118  0.02086822    0.02308578
## 2020-04-30 0.02637500  0.02071902    0.02303374
## 2020-05-01 0.02699270  0.02057174    0.02298197
## 2020-05-04 0.02449756  0.02042636    0.02293046
## 2020-05-05 0.02430946  0.02028288    0.02287921
## 2020-05-06 0.02389937  0.02014126    0.02282821
## 2020-05-07 0.01941791  0.02000150    0.02277748
## 2020-05-08 0.01964116  0.01986357    0.02272700
```

```
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)
```

```
## Warning: `mapping` is not used by stat_function()

## Warning: `mapping` is not used by stat_function()
```