Quantitative Risk Management

Code

Jeff Gross

Libraries

Hide

```
library(qrmdata)
library(xts)
library(moments)
library(QRM)
library(readr)
library(qrmtools)
```

```
Attaching package: <U+393C><U+3E31>qrmtools<U+393C><U+3E32>

The following objects are masked from <U+393C><U+3E31>package:QRM<U+393C><U+3E32>:

    dGEV, dGPD, pGEV, pGPD, qGEV, qGPD, rGEV, rGPD

The following object is masked from <U+393C><U+3E31>package:timeSeries<U+393C><U+3E32>:

    returns
```

Introduction

In quantitative risk management, QRM, the risk of a portfolio is quantified. Measuring risk is the first step towawrd managing risk.

Ways to manage risk: Selling risky assets, diversifying portfolios, implementing hedging with derivatives

Goal for banks: Maintain sufficient capital to withstand losses. An example is Value-at-Risk

The value of a portfolio depends on risk factor, i.e. equity indexes/prices, FX rates, interest rates, commodity prices.

Exploring risk-factor time series: equity indexes

Hide

```
# Load DJ index
data("DJ")
# Show head() and tail() of DJ index
head(DJ)
```

```
             ^DJI
1985-01-29 1292.62
1985-01-30 1287.88
1985-01-31 1286.77
1985-02-01 1277.72
1985-02-04 1290.08
1985-02-05 1285.23
```

Hide

```
tail(DJ)
```

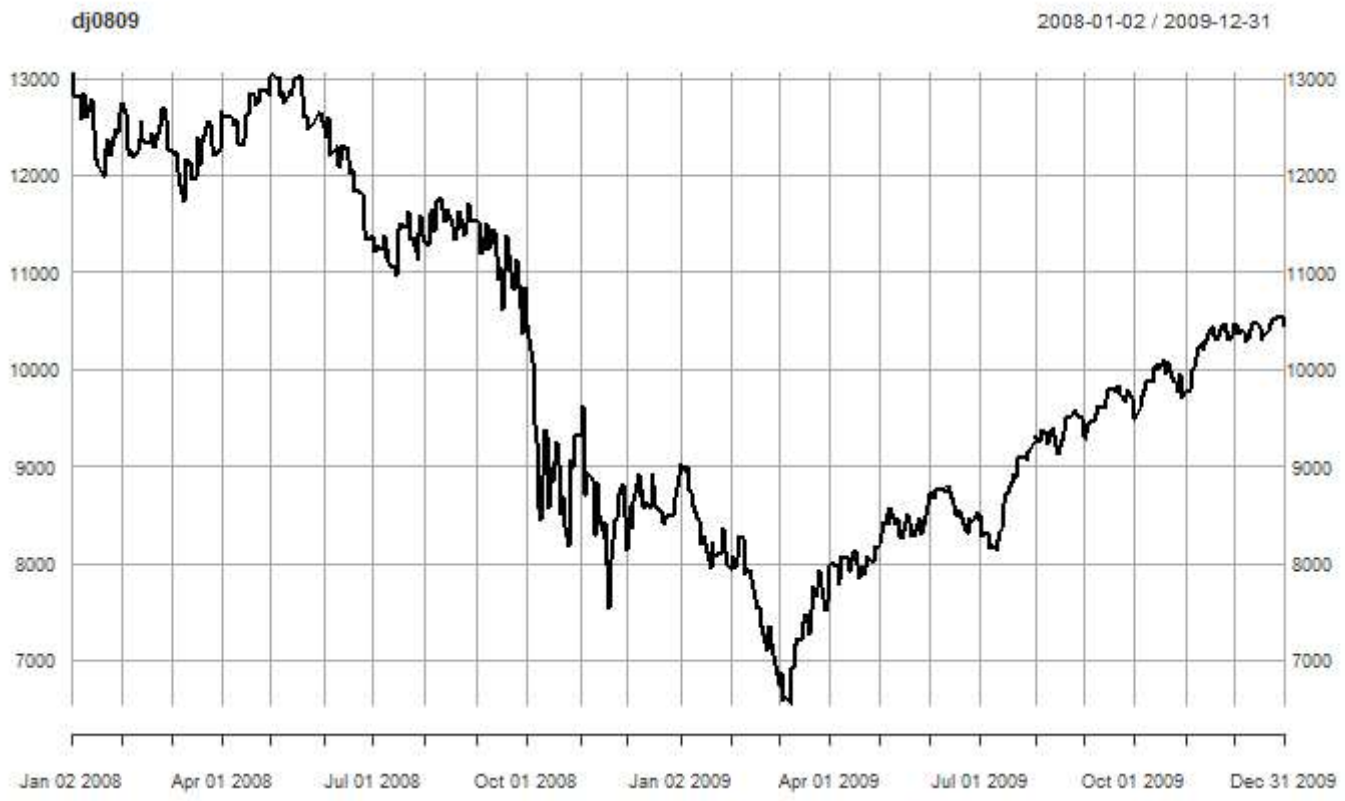
```
              ^DJI
2015-12-23 17602.61
2015-12-24 17552.17
2015-12-28 17528.27
2015-12-29 17720.98
2015-12-30 17603.87
2015-12-31 17425.03
```

Hide

```
# Plot DJ index
plot(DJ)
```

Hide

```
# Extract 2008-2009 and assign to dj0809
dj0809 <- DJ["2008/2009"]
# Plot dj0809
plot(dj0809)
```



Exploring risk-factor time series: individual equities

Hide

```
# Load DJ constituents data
data("DJ_const")
# Apply names() and head() to DJ_const
names(DJ_const)
```

```
 [1] "AAPL" "AXP"  "BA"   "CAT"  "CSCO" "CVX"  "DD"   "DIS"  "GE"   "GS"   "HD"   "IBM"  "INTC"
"JNJ"  "JPM"
[16] "KO"   "MCD"  "MMM"  "MRK"  "MSFT" "NKE"  "PFE"  "PG"   "TRV"  "UNH"  "UTX"  "V"    "VZ"
"WMT"  "XOM"
```

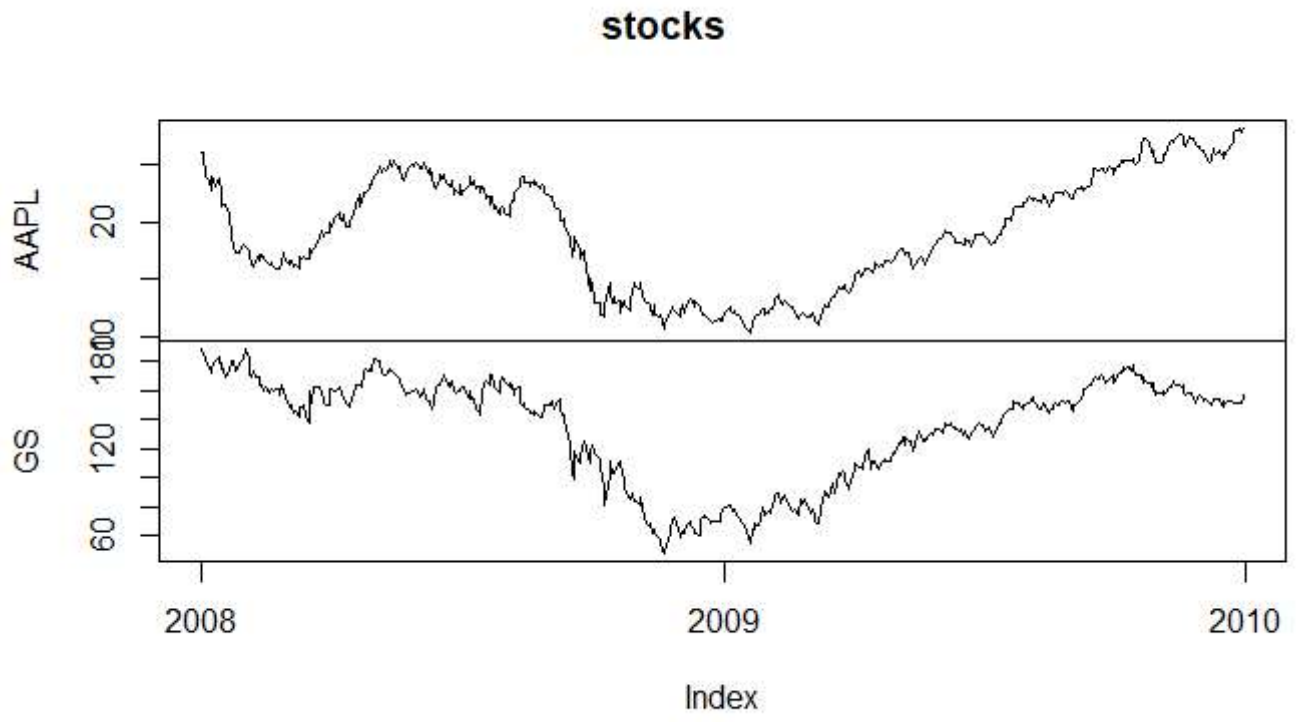
Hide

```
head(DJ_const)
```

```
           AAPL AXP       BA      CAT CSCO CVX       DD      DIS       GE GS HD      IBM INTC JN
J JPM       KO
1962-01-02   NA  NA 0.212905 0.593184   NA  NA 1.227958 0.061014 0.145967 NA NA 2.346625   NA  N
A  NA 0.031031
1962-01-03   NA  NA 0.217163 0.598964   NA  NA 1.229230 0.061832 0.144501 NA NA 2.367136   NA  N
A  NA 0.030339
1962-01-04   NA  NA 0.215034 0.614372   NA  NA 1.220331 0.061832 0.142794 NA NA 2.343548   NA  N
A  NA 0.030571
1962-01-05   NA  NA 0.210773 0.620152   NA  NA 1.187280 0.062039 0.139132 NA NA 2.297395   NA  N
A  NA 0.029879
1962-01-08   NA  NA 0.211306 0.624001   NA  NA 1.169484 0.061832 0.138889 NA NA 2.254319   NA  N
A  NA 0.029570
1962-01-09   NA  NA 0.211839 0.629777   NA  NA 1.173297 0.063060 0.139620 NA NA 2.280983   NA  N
A  NA 0.030110
           MCD MMM MRK MSFT NKE PFE PG TRV UNH UTX  V VZ WMT XOM
1962-01-02  NA  NA  NA   NA  NA  NA NA  NA  NA  NA NA NA  NA  NA
1962-01-03  NA  NA  NA   NA  NA  NA NA  NA  NA  NA NA NA  NA  NA
1962-01-04  NA  NA  NA   NA  NA  NA NA  NA  NA  NA NA NA  NA  NA
1962-01-05  NA  NA  NA   NA  NA  NA NA  NA  NA  NA NA NA  NA  NA
1962-01-08  NA  NA  NA   NA  NA  NA NA  NA  NA  NA NA NA  NA  NA
1962-01-09  NA  NA  NA   NA  NA  NA NA  NA  NA  NA NA NA  NA  NA
```

Hide

```
# Extract AAPL and GS in 2008-09 and assign to stocks
stocks <- DJ_const["2008/2009", c("AAPL","GS")]
# Plot stocks with plot.zoo()
plot.zoo(stocks)
```



Exploring risk-factor data: exchange rates

Hide

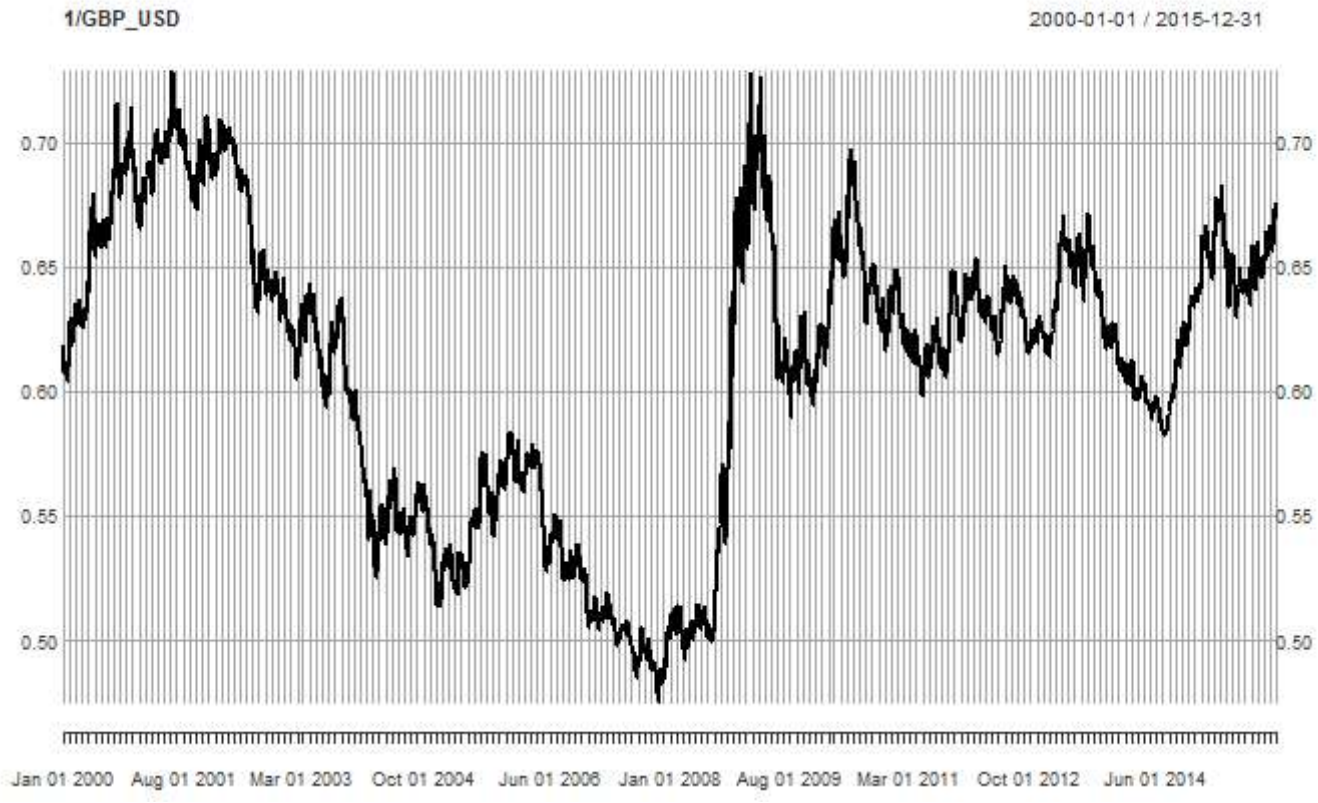
```
# Load exchange rate data
data("GBP_USD")
data("EUR_USD")
# Plot the two exchange rates
plot(GBP_USD)
```

Hide

```
plot(EUR_USD)
```

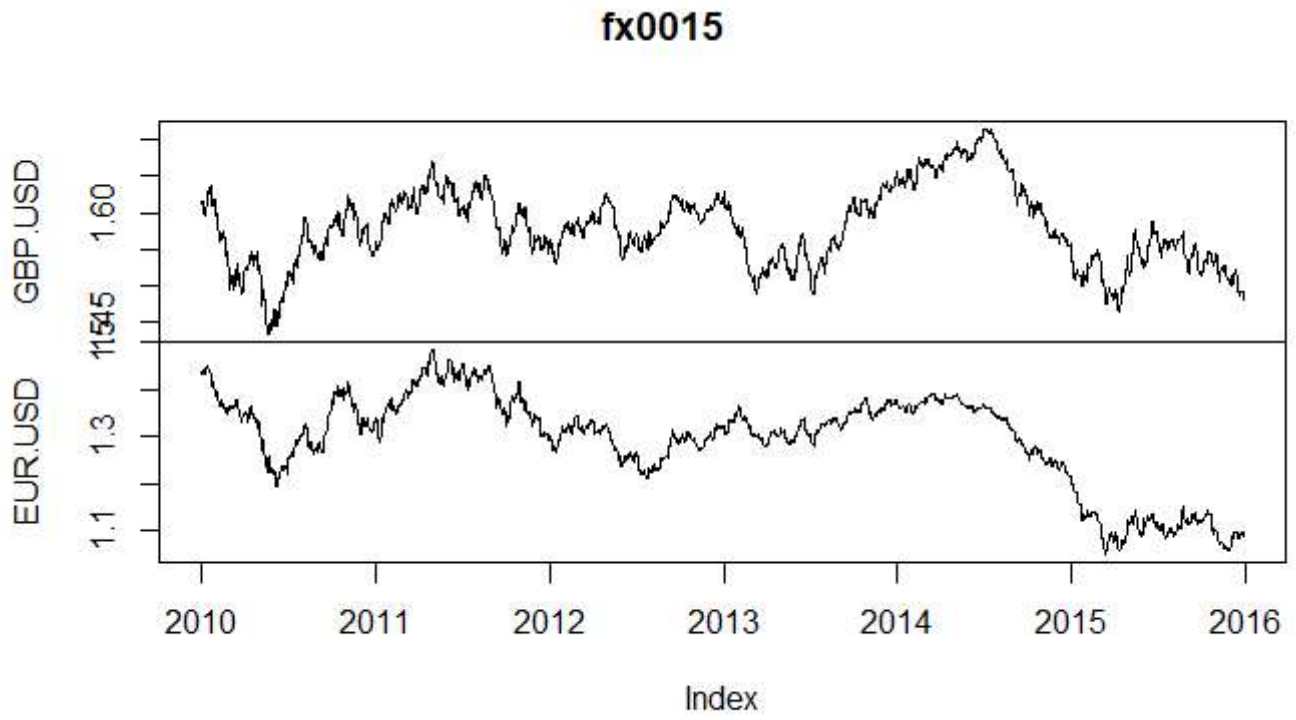
Hide

```
# Plot a USD_GBP exchange rate
plot(1/GBP_USD)
```



Hide

```
# Merge the two exchange rates GBP_USD and EUR_USD
fx <- merge(GBP_USD, EUR_USD, all = TRUE)
# Extract 2010-15 data from fx and assign to fx0015
fx0015 <- fx["2010/2015"]
# Plot the exchange rates in fx0015
plot.zoo(fx0015)
```



Risk-factor returns

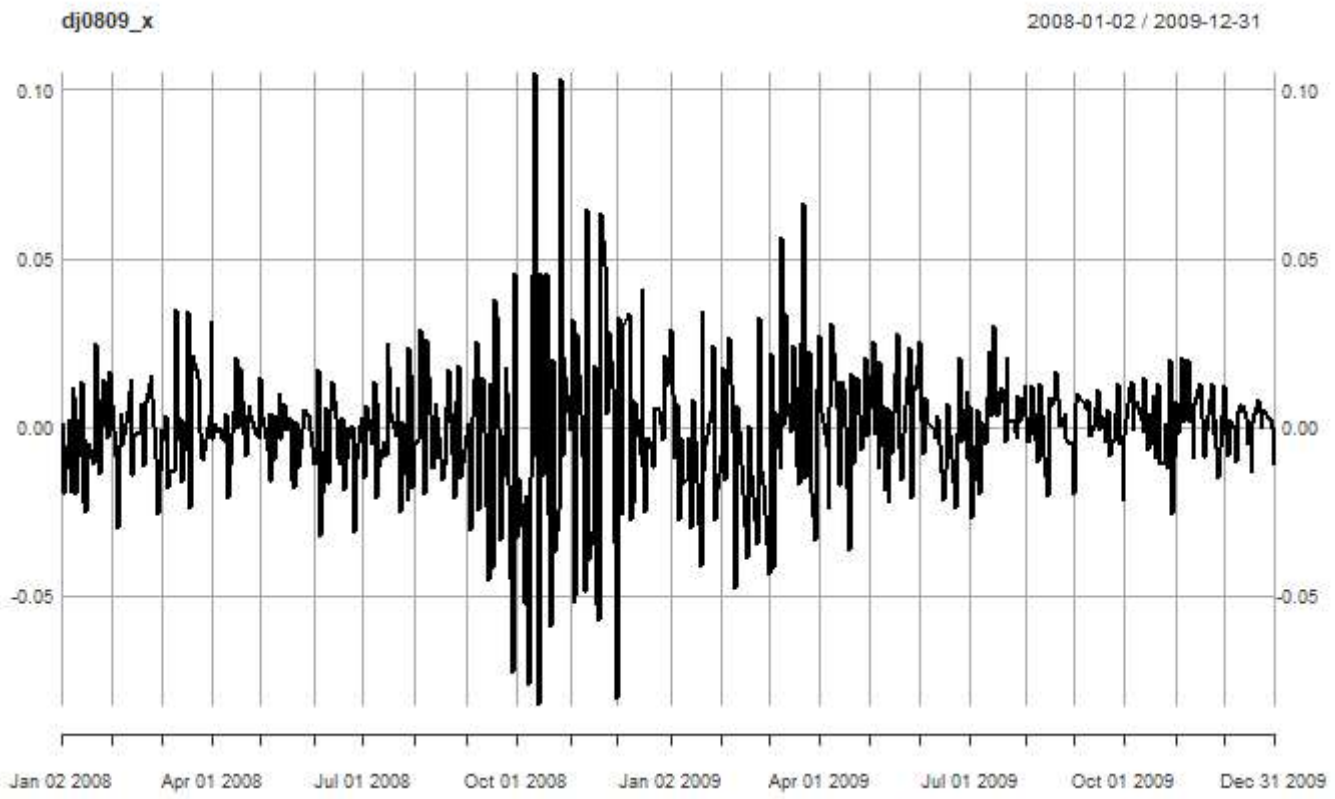
Properties of log-returns:

Properties of log-returns . Resulting risk factors cannot become negative . Very close to relative returns for small changes . Easy to aggregate by summation to obtain longer- interval log-returns . If a price series follows GBM, then the log returns will be normally distributed

Exploring return series

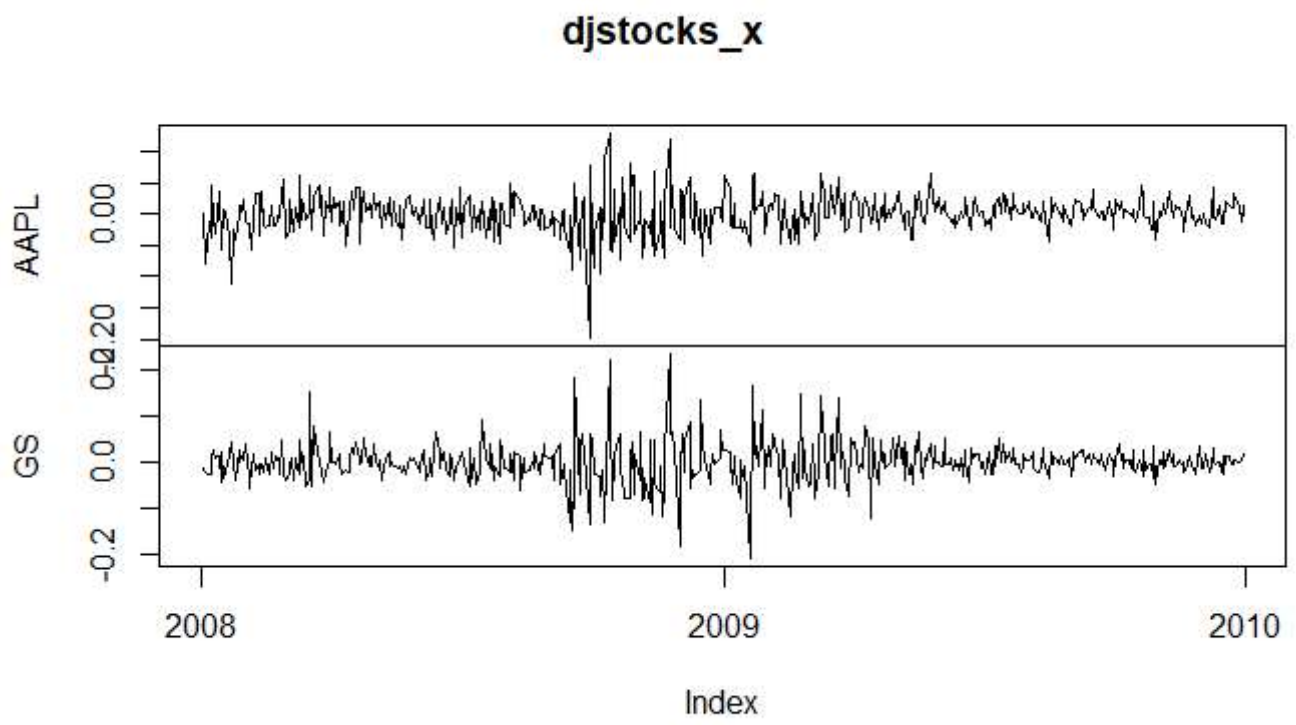
Hide

```
djstocks <- read_csv("C:/Users/Y/Documents/1.R_scripts/ML/Risk_Modeling/Quantitative Risk Manage
ment in R/djstocks.csv",
    col_types = cols(AAPL = col_number(),
        GS = col_number(), date = col_date(format = "%m/%d/%Y")))
djstocks_xts <- as.xts(djstocks[,-1], order.by = djstocks$date, "%Y-%m-%d")
# Compute the log-returns of dj0809 and assign to dj0809_x
dj0809_x <- diff(log(dj0809))
# Plot the log-returns
plot(dj0809_x)
```



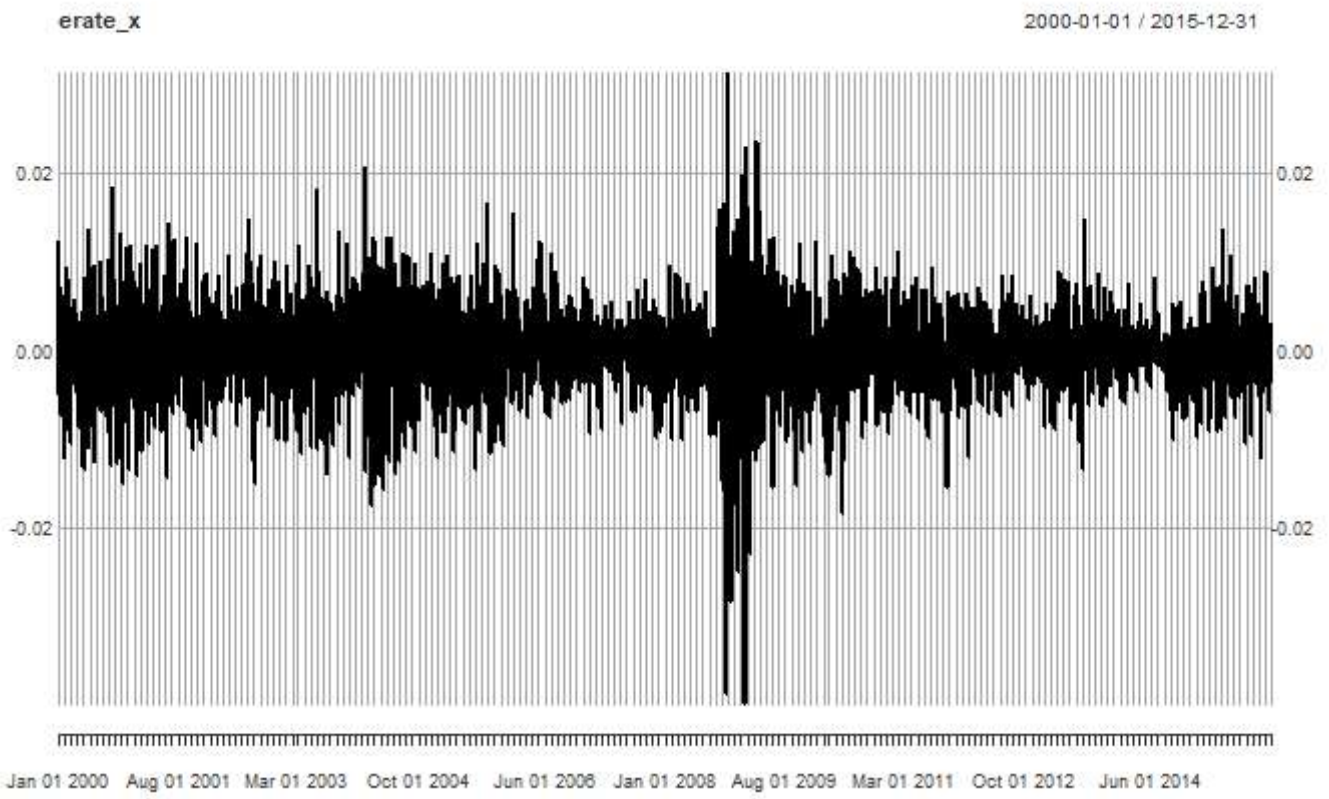
Hide

```
# Compute the log-returns of djstocks and assign to djstocks_x
djstocks_x <- diff(log(djstocks_xts))
# Plot the two share returns
plot.zoo(djstocks_x)
```



Hide

```
# Compute the log-returns of GBP_USD and assign to erate_x
erate_x <- diff(log(GBP_USD))
# Plot the log-returns
plot(erate_x)
```



Different ways of plotting risk-factor and return series

Note how in late 2008 there were large returns for all series. That was the height of the financial crisis.

Hide

```
djstocks_1 <- read_csv("C:/Users/Y/Documents/1.R_scripts/ML/Risk_Modeling/Quantitative Risk Mana
gement in R/djstocks_1.csv",
    col_types = cols(AAPL = col_number(),
        GS = col_number(), date = col_date(format = "%m/%d/%Y")))
```

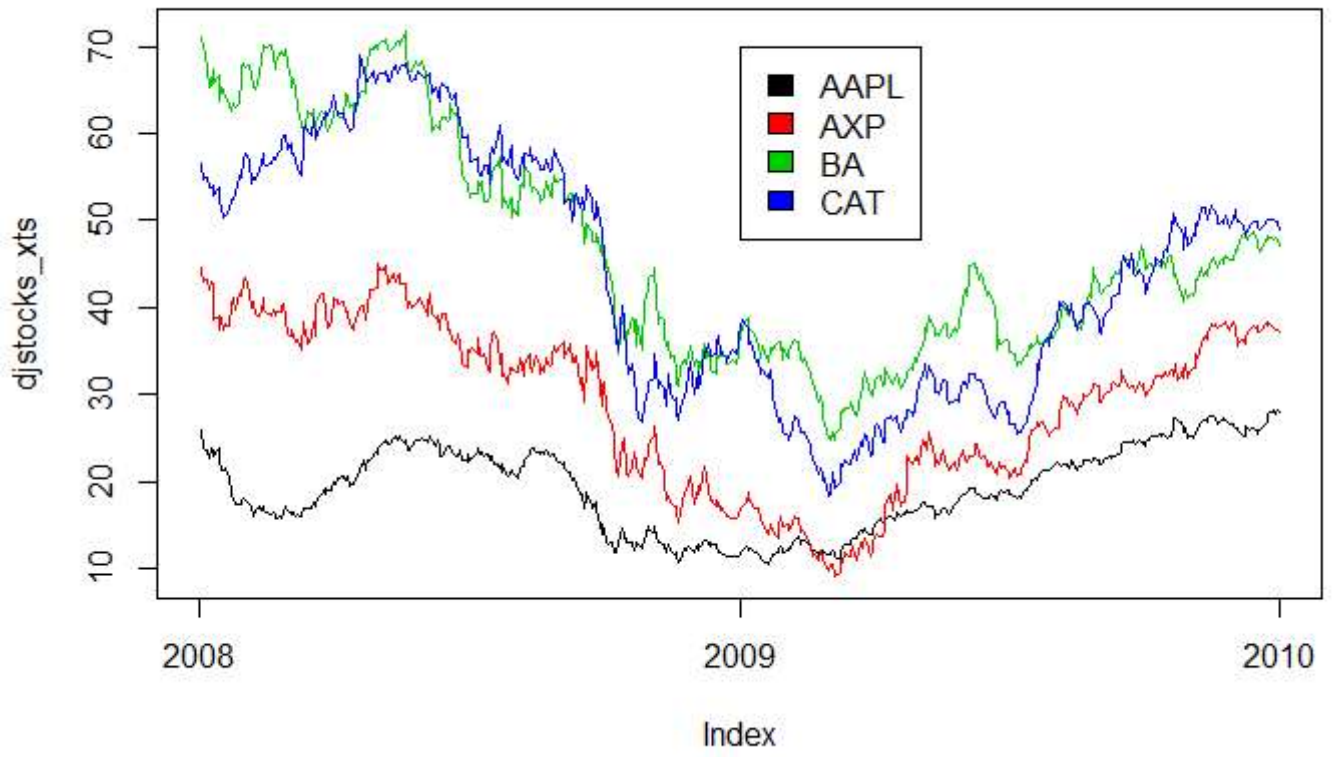
```
The following named parsers don't match the column names: GS
```

Hide

```
djstocks_xts <- as.xts(djstocks_1[,-1], order.by = djstocks_1$date, "%Y-%m-%d")
# Plot djstocks in four separate plots
plot.zoo(djstocks_xts, plot.type="single")
```

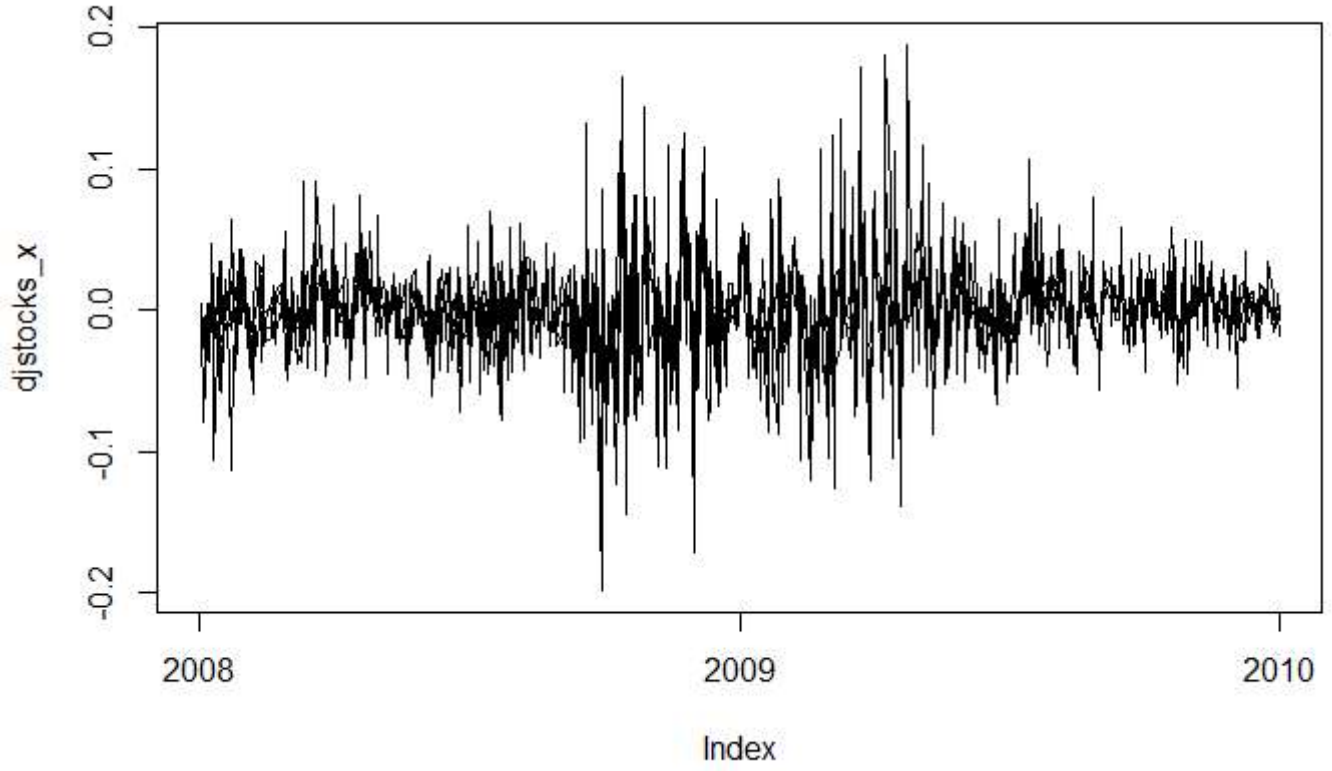
Hide

```
# Plot djstocks in one plot and add legend
plot.zoo(djstocks_xts, plot.type="single", col = 1:4)
legend(julian(x = as.Date("2009-01-01")), y = 70, legend = names(DJ_const)[1:4], fill = 1:4)
```



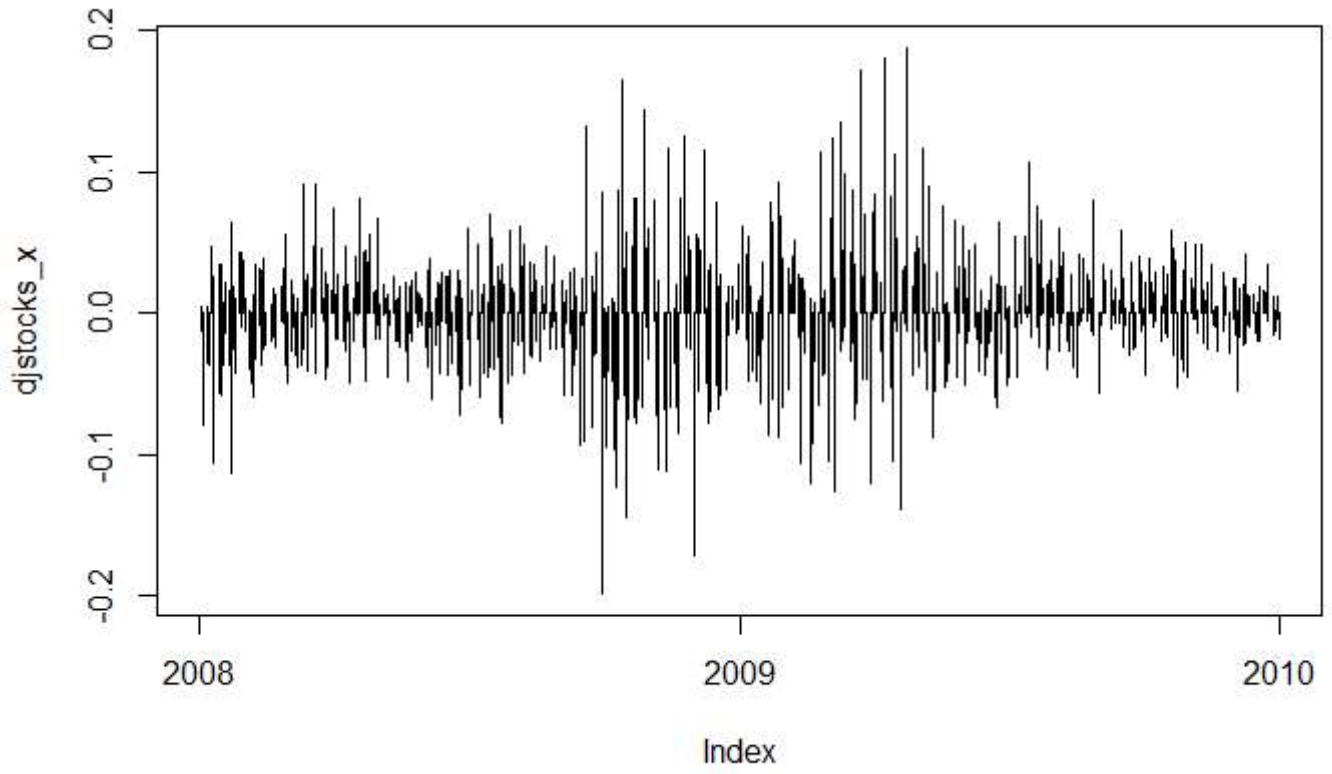
Hide

```
# Compute log-returns and assign to djstocks_x
djstocks_x <- diff(log(djstocks_xts))
# Plot djstocks_x in four separate plots
plot.zoo(djstocks_x, plot.type="single")
```



Hide

```
# Plot djstocks_x with vertical bars
plot.zoo(djstocks_x, plot.type="single", type = "h")
```



Aggregating log-return series

These aggregation functions are extremely useful as for analyzing risk over longer time horizons.

Hide

```
djx <- read_csv("C:/Users/Y/Documents/1.R_scripts/ML/Risk_Modeling/Quantitative Risk Management
 in R/djx.csv",
    col_types = cols(`^DJI` = col_number(),
        date = col_date(format = "%m/%d/%Y")))
djx_xts <- as.xts(djx[,-1], order.by = djx$date, "%Y-%m-%d")
djreturns <- read_csv("C:/Users/Y/Documents/1.R_scripts/ML/Risk_Modeling/Quantitative Risk Manag
ement in R/djreturns.csv",
    col_types = cols(AAPL = col_number(),
        AXP = col_number(), BA = col_number(),
        CAT = col_number(), date = col_date(format = "%m/%d/%Y")))
djreturns_xts <- as.xts(djreturns[,-1], order.by = djreturns$date, "%Y-%m-%d")
# Calculate skewness and kurtosis of djx
skewness(djx)
```

```
Error in skewness(djx) : could not find function "skewness"
```

A test on aggregation of log-returns

To three decimal places, what is the average quarterly log-return for the S&P 500 from 1990-2010?

Hide

```
sp <- read_csv("C:/Users/Y/Documents/1.R_scripts/ML/Risk_Modeling/Quantitative Risk Management i
n R/sp.csv",
    col_types = cols(`^GSPC` = col_number(),
        date = col_date(format = "%m/%d/%Y")))
sp_xts <- as.xts(sp[,-1], order.by = sp$date, "%m-%d-%Y")
#sp_xts <- xts(sp[,-1], order.by = as.POSIXct(sp$date, format = "%m.%d.%Y %H:%M"))
#head(sp_xts)
mean(apply.quarterly(sp_xts["1990/2010", ], sum))
```

```
[1] 0.01511181
```

Exploring other kinds of risk factors

??? Let p(t, T) denote the price at time small t of a zerocoupon bond paying one unit at maturity T

The **yield** y(t, T) is defined by the equation:

$$y(t,T) = \frac{-\ln p(t,T)}{T-t}$$

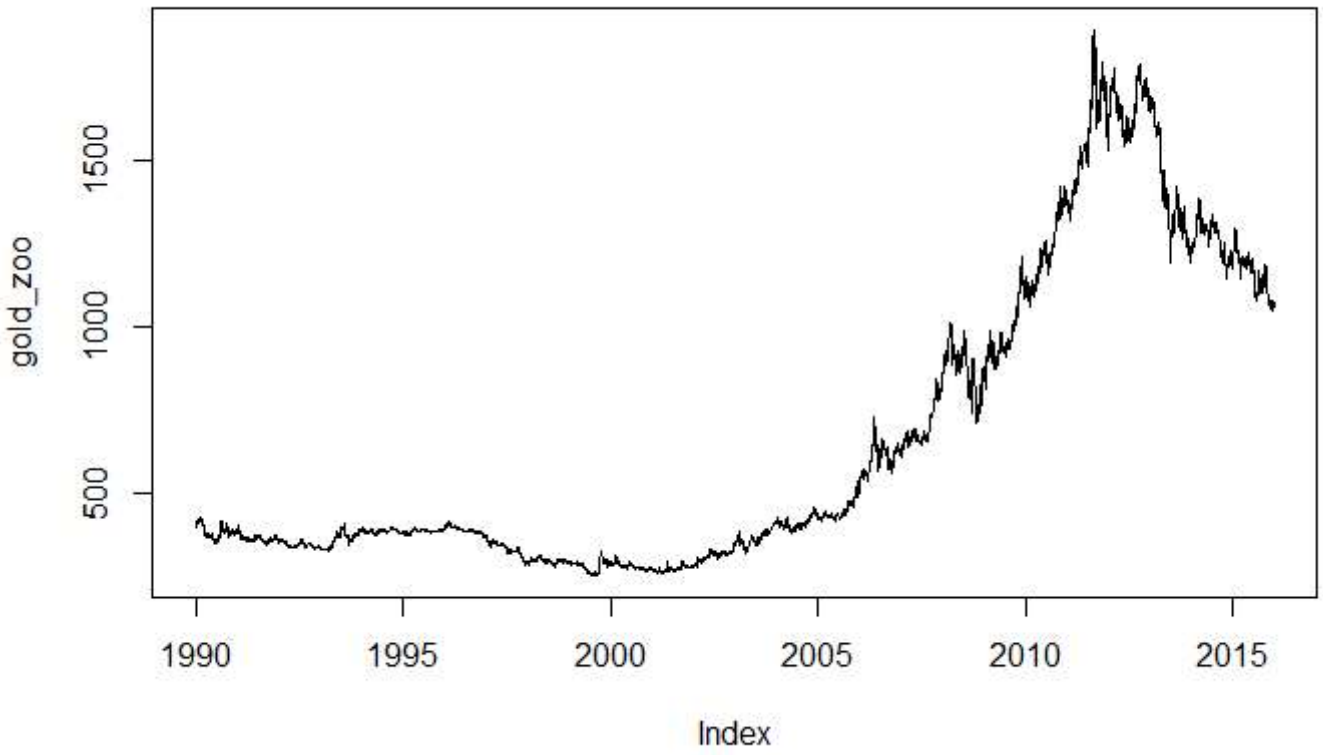
Bond Yield

??? Advantage of yields: comparable across maturities T ??? The mapping T to y(t, T) is yield curve at time t ??? In low rate environment and when rates can go negative, there are arguments for using simple returns.

Commodities data

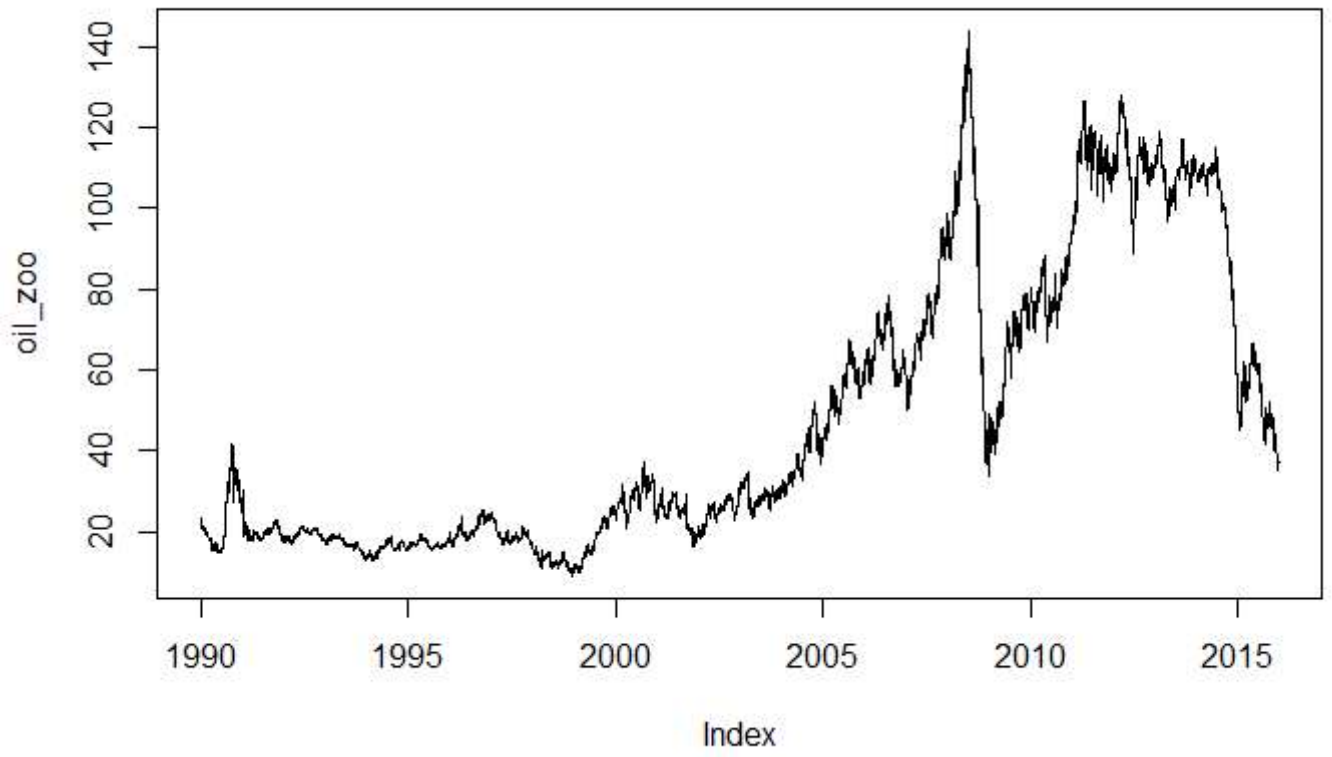
Hide

```
gold_zoo <- read.zoo('C:/Users/Y/Documents/1.R_scripts/ML/Risk_Modeling/Quantitative Risk Manage
ment in R/gold.csv', header=TRUE, sep = ",", format = "%m/%d/%Y")
oil_zoo <- read.zoo('C:/Users/Y/Documents/1.R_scripts/ML/Risk_Modeling/Quantitative Risk Managem
ent in R/oil.csv', header=TRUE, sep = ",", format = "%m/%d/%Y")
# Plot gold and oil prices
plot(gold_zoo)
```



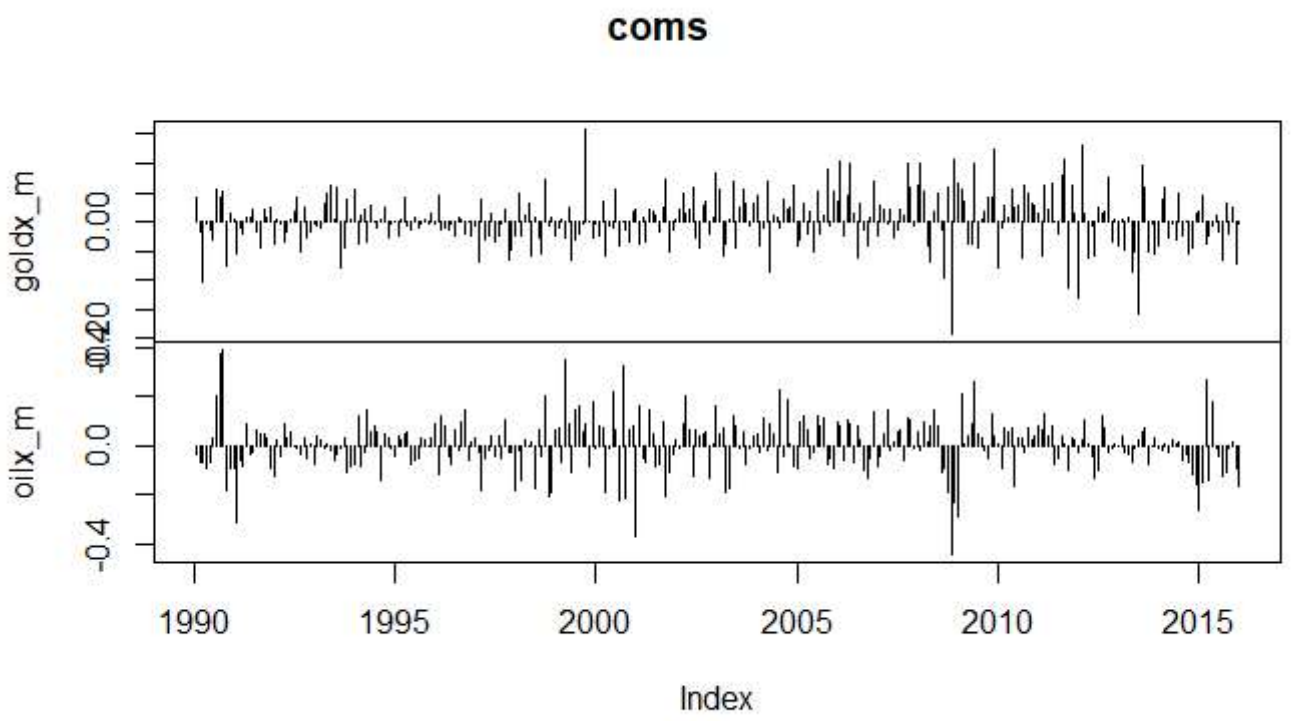
Hide

```
plot(oil_zoo)
```



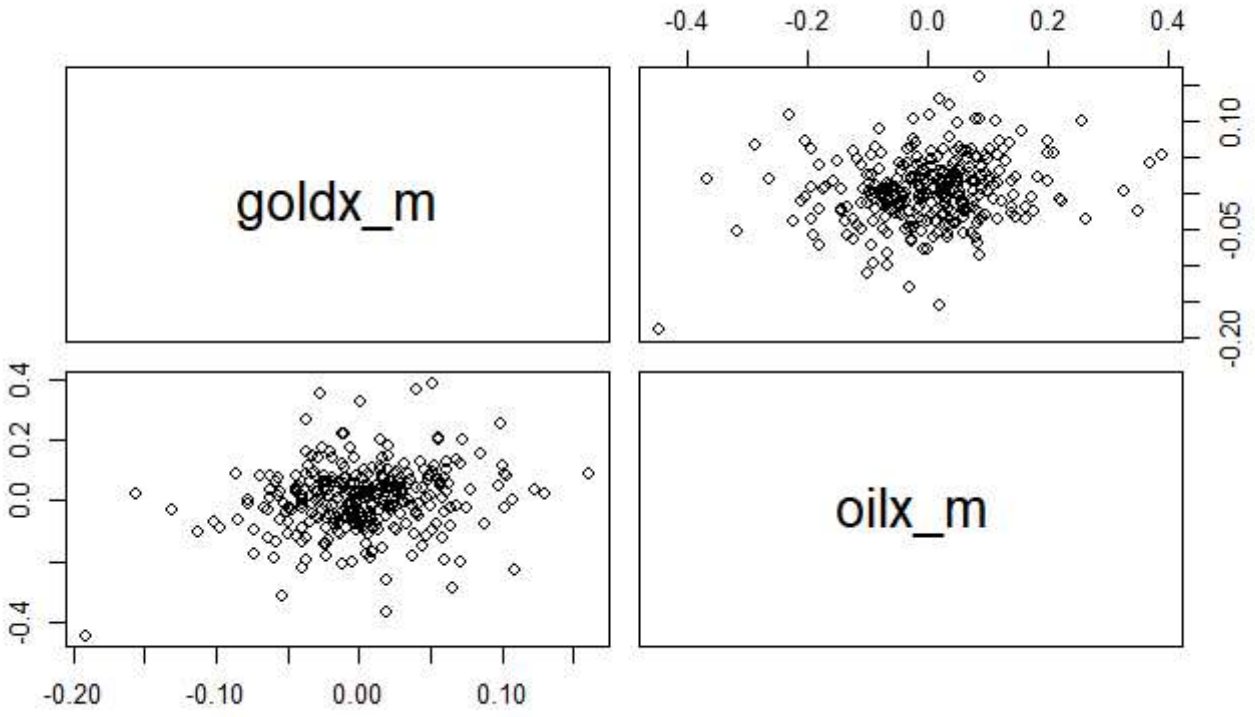
Hide

```
# Calculate daily log-returns
goldx <- diff(log(gold_zoo))
oilx <- diff(log(oil_zoo))
# Calculate monthly log-returns
goldx_m <- apply.monthly(goldx,sum)
oilx_m <- apply.monthly(oilx,sum)
# Merge goldx_m and oilx_m into coms
coms <- merge(goldx_m, oilx_m)
# Plot coms with vertical bars
plot.zoo(coms, type = "h")
```



Hide

```
# Make a pairwise scatterplot of coms
pairs(as.zoo(coms))
```



Interest-rate data

Hide

```
zcb_x <- diff(log(zcb_1))
```

```
Error in r[i1] - r[-length(r):-(length(r) - lag + 1L)] :
  non-numeric argument to binary operator
```

The normal distribution

Black Scholes Geometric Brownian Motion model for assett prices implies tht log returns over different time periods are normally distributed and independent.

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{\sigma^2}}$$

Density of a Normal Distirbution

Graphical methods for assessing normality

Task: Use graphical methods to assess normality.

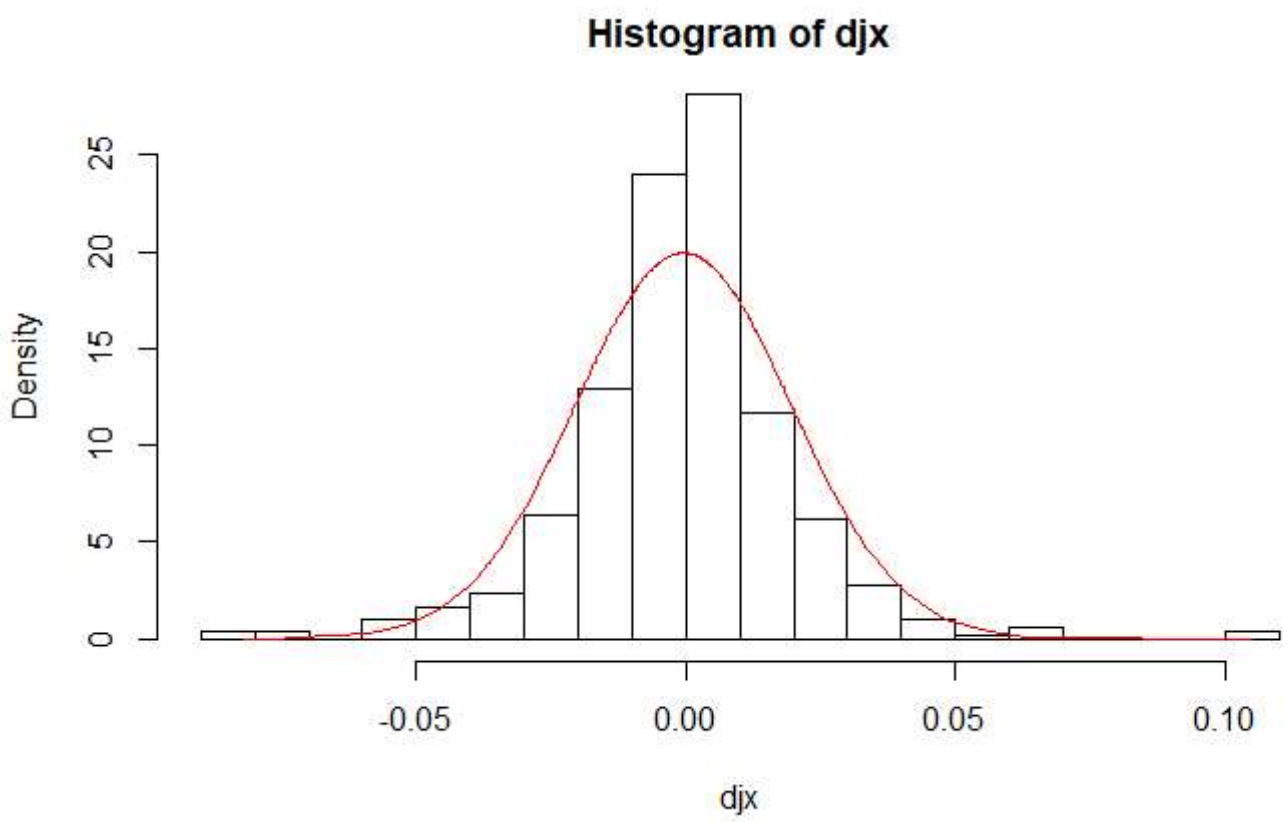
Conclusion: The data don’t look very normal. Compare in particular the center and the tails of the histogram and density plot with the red normal curve.

Hide

```
djx <- c(-0.08200514,   -0.08014005,    -0.07615925,    -0.07234497,    -0.05863401,
-0.05725062,    -0.05241625,    -0.05207235,    -0.05181254,    -0.04974146,
-0.04846434,    -0.04728556,    -0.04517256,    -0.04335054,    -0.04178693,
-0.04148142,    -0.04093283,    -0.03899981,    -0.03867389,    -0.03659414,
-0.03647977,    -0.03626504,    -0.03465573,    -0.03327818,    -0.03323052,
-0.03267833,    -0.03181025,    -0.03081321,    -0.0303397, -0.02984478,
-0.02972315,    -0.02759829,    -0.02755835,    -0.02740841,    -0.02668228,
-0.02661128,    -0.02541853,    -0.02539864,    -0.02539635,    -0.02533262,
-0.0251715, -0.02493088,    -0.02463825,    -0.02462672,    -0.02454773,
-0.02392701,    -0.02378497,    -0.02363386,    -0.02266336,    -0.02199214,
-0.02191924,    -0.02149594,    -0.02129783,    -0.0211229, -0.02101739,
-0.02101462,    -0.02068463,    -0.02060188,    -0.02021011,    -0.02016248,
-0.02010715,    -0.01984371,    -0.01974668,    -0.01956217,    -0.01946055,
-0.01938756,    -0.01890157,    -0.01876154,    -0.01843953,    -0.0183474,
-0.01834654,    -0.01791473,    -0.01789198,    -0.01766642,    -0.01722786,
-0.01690319,    -0.01672904,    -0.01667978,    -0.0165248, -0.01615604,
-0.01598478,    -0.01560242,    -0.01554523,    -0.01542992,    -0.01517953,
-0.01513562,    -0.01503224,    -0.01501595,    -0.01487253,    -0.01477133,
-0.01475836,    -0.01472306,    -0.01466773,    -0.0142492, -0.01406082,
-0.01394653,    -0.0135227, -0.01299342,    -0.01288018,    -0.01280637,
-0.01255625,    -0.01228968,    -0.01225876,    -0.01216409,    -0.01212861,
-0.01210627,    -0.01194302,    -0.01194026,    -0.01183072,    -0.01163039,
-0.01157042,    -0.01152402,    -0.0115077, -0.01148532,    -0.01146326,
-0.01124613,    -0.01094946,    -0.01088399,    -0.01085118,    -0.01069927,
-0.01064462,    -0.01050604,    -0.010468,  -0.01043521,    -0.01038794,
-0.01007362,    -0.009739113,   -0.009705312,   -0.009450576,   -0.009440744,
-0.009378182,   -0.009216298,   -0.009155345,   -0.009095314,   -0.009043876,
-0.008905751,   -0.008870017,   -0.008794866,   -0.008513629,   -0.00841603,
-0.008312647,   -0.008307185,   -0.008225947,   -0.008214471,   -0.008204289,
-0.008204238,   -0.008195324,   -0.008107969,   -0.008045147,   -0.007578101,
-0.007551941,   -0.007536722,   -0.007019896,   -0.006950214,   -0.00681277,
-0.006683386,   -0.006370944,   -0.006250824,   -0.005585044,   -0.005525874,
-0.005316151,   -0.005310895,   -0.00521962,    -0.00503734,    -0.005033489,
-0.005024773,   -0.004939321,   -0.004829133,   -0.004681886,   -0.004554134,
-0.004488752,   -0.004361831,   -0.00422589,    -0.00421856,    -0.004217023,
-0.004093719,   -0.004022377,   -0.003918205,   -0.003897322,   -0.003838076,
-0.003809679,   -0.003730128,   -0.003720132,   -0.003685792,   -0.003597354,
-0.00344315,    -0.003433837,   -0.003433101,   -0.00321976,    -0.003168273,
-0.00311433,    -0.00311101,    -0.003097579,   -0.003075893,   -0.003012075,
-0.003006826,   -0.003003118,   -0.002953996,   -0.002862266,   -0.002857554,
-0.002799683,   -0.002773282,   -0.002747108,   -0.00267536,    -0.002665984,
-0.002545776,   -0.002345101,   -0.002327223,   -0.00229506,    -0.00227514,
-0.002153605,   -0.001932504,   -0.001911204,   -0.001897097,   -0.001896118,
-0.001856547,   -0.001807037,   -0.001806553,   -0.001792338,   -0.001787576,
-0.001650995,   -0.001561144,   -0.001492164,   -0.001383037,   -0.00131643,
-0.001295237,   -0.001279047,   -0.001174486,   -0.00106504,    -0.00104148,
-0.001025541,   -0.001003594,   -0.0009977797,  -0.0009708178,  -0.0009208261,
-0.0008904228,  -0.0008811082,  -0.0008530615,  -0.0007958916,  -0.0006248415,
-0.0006107375,  -0.0005828208,  -0.0005556163,  -0.0004511526,  -0.0001632475,
-0.0001583428,  -0.00002787226, 0.00009672858,  0.0001164599,   0.0001443519,
0.0001552237,   0.0002386634,   0.0002938866,   0.0003084173,   0.0003492264,
0.0003726103,   0.0004010774,   0.0004432777,   0.0005642923,   0.0005818223,
0.0007258399,   0.0007376396,   0.0007683699,   0.0009777345,   0.001118739,
```

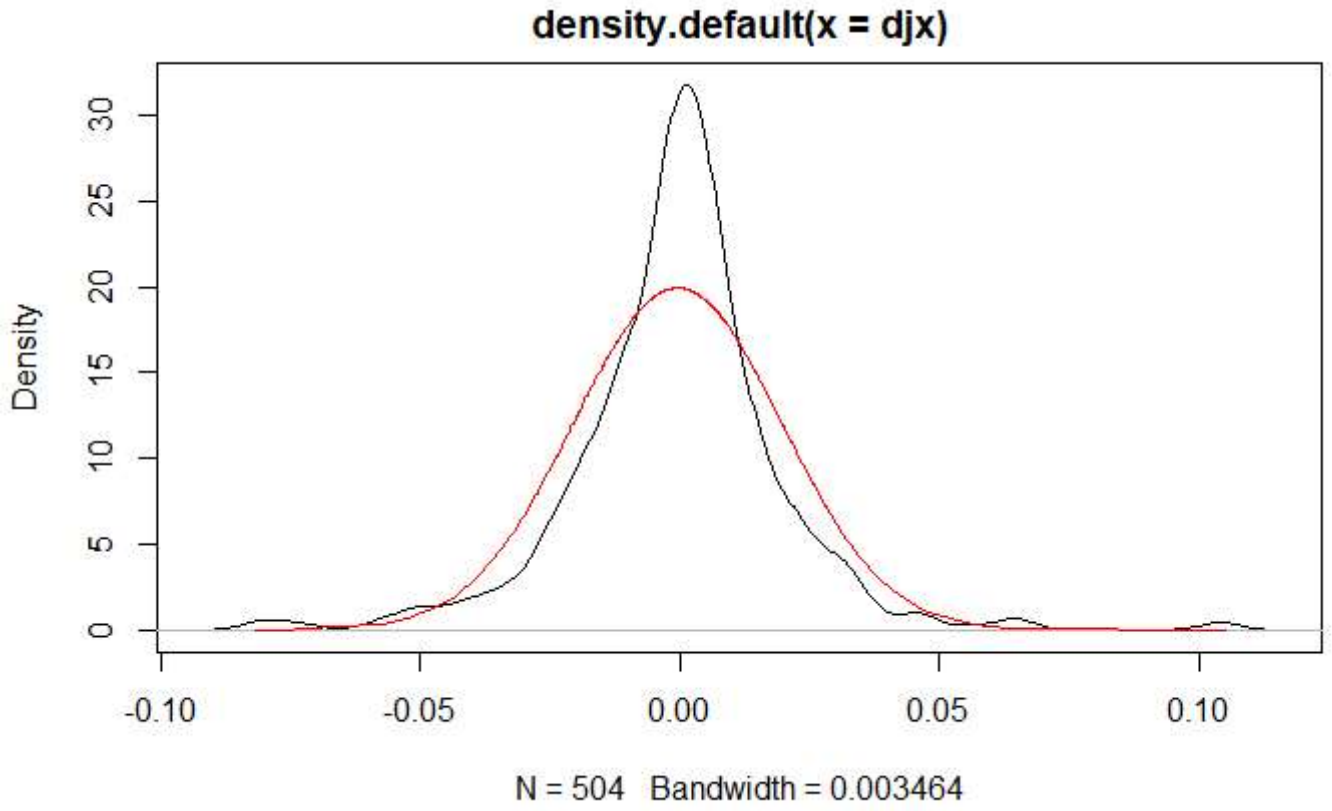
```
0.001356084,    0.001384824,    0.001438974,    0.001471978,    0.00150501,
0.00167781, 0.001743435,    0.001768997,    0.001812514,    0.001871694,
0.0018847,  0.001936804,    0.001956568,    0.001999296,    0.002112263,
0.002131334,    0.002192238,    0.002224358,    0.002225329,    0.002335189,
0.002562304,    0.002571987,    0.002637321,    0.002817952,    0.002825011,
0.002837378,    0.00292108, 0.002922609,    0.002927115,    0.002937151,
0.003088734,    0.003150871,    0.003179739,    0.003225905,    0.003332369,
0.003334021,    0.003366776,    0.003373899,    0.003381299,    0.003394674,
0.003463484,    0.003614914,    0.003633762,    0.003643694,    0.003680178,
0.003698011,    0.003701295,    0.003778233,    0.003798257,    0.003836893,
0.003880999,    0.003899841,    0.003946855,    0.004083211,    0.004084708,
0.004135421,    0.004264071,    0.004312943,    0.004350759,    0.004358618,
0.004689539,    0.004739961,    0.004751238,    0.004772741,    0.004865173,
0.004898555,    0.004916289,    0.004940068,    0.004953705,    0.005112759,
0.005145659,    0.005202772,    0.005238241,    0.00531507, 0.005407657,
0.005491446,    0.00554279, 0.005588613,    0.005758315,    0.005801807,
0.005863273,    0.005921288,    0.005961787,    0.006051143,    0.006085743,
0.006282168,    0.006291048,    0.006399894,    0.006490434,    0.00661941,
0.006631724,    0.006694535,    0.006732297,    0.006851687,    0.00686601,
0.00692456, 0.007133137,    0.007168385,    0.007206594,    0.007271741,
0.007282882,    0.007291543,    0.007311344,    0.007492014,    0.007512767,
0.007610628,    0.007632292,    0.00775498, 0.007823576,    0.007842609,
0.007866853,    0.007945398,    0.008031997,    0.008219675,    0.008334073,
0.008371574,    0.008608137,    0.008798904,    0.008898302,    0.00900124,
0.009083378,    0.009189575,    0.00920582, 0.009585875,    0.01018108,
0.01029071, 0.01072517, 0.01083975, 0.01092447, 0.01103545,
0.01112198, 0.01142103, 0.01148651, 0.01154941, 0.01166024,
0.01166927, 0.01171761, 0.011744,   0.0118475,  0.01186145,
0.01201105, 0.0121771,  0.01221332, 0.01222055, 0.01238209,
0.01242609, 0.01245527, 0.0127653,  0.01278744, 0.01287844,
0.0129185,  0.01317491, 0.01320205, 0.01328502, 0.01346402,
0.01354004, 0.01356068, 0.0136053,  0.01372326, 0.01434933,
0.01437293, 0.01451752, 0.01456267, 0.01470171, 0.01487302,
0.01504,    0.01516589, 0.0155976,  0.01616982, 0.01619885,
0.0165373,  0.01654114, 0.01712147, 0.01716126, 0.01768499,
0.01797233, 0.01802471, 0.01810649, 0.01828319, 0.01832016,
0.01940663, 0.01986728, 0.01999316, 0.02010111, 0.02026806,
0.02029378, 0.02055973, 0.02057518, 0.02058022, 0.02083432,
0.02088279, 0.02095057, 0.0213949,  0.02150929, 0.02203021,
0.02229847, 0.02247423, 0.02342321, 0.02365806, 0.02425714,
0.02446355, 0.02466814, 0.0249308,  0.02549708, 0.02567942,
0.02576357, 0.02615136, 0.02661985, 0.02750928, 0.02807605,
0.02812262, 0.0287291,  0.0289645,  0.02900654, 0.03024786,
0.03047319, 0.03093996, 0.03142425, 0.0322486,  0.03259548,
0.03265391, 0.03291513, 0.03399651, 0.03401199, 0.03449764,
0.0345129,  0.03487485, 0.03791879, 0.04113064, 0.04562199,
0.04572714, 0.04574774, 0.04815658, 0.05633869, 0.06337643,
0.06458521, 0.06611575, 0.1032592,  0.1050835)
# Calculate average and standard deviation of djx
mu <- mean(djx)
sigma <- sd(djx)
# Plot histogram of djx
hist(djx, nclass = 20, probability = TRUE)
```

```
# Add the normal density as a red line to histogram
lines(djx, dnorm(djx, mean=mu, sd=sigma), col = "red")
```



Hide

```
# Plot non-parametric KDE of djx
plot(density(djx))
# Add the normal density as red line to KDE
lines(djx, dnorm(djx, mean=mu, sd=sigma), col = "red")
```



Testing for normality

Interpreting the Q-Q plot . Data with heavier tails than normal: inverted S shape . Data with lighter tails than normal: S shape . Data from a very skewed distribution: curved shape

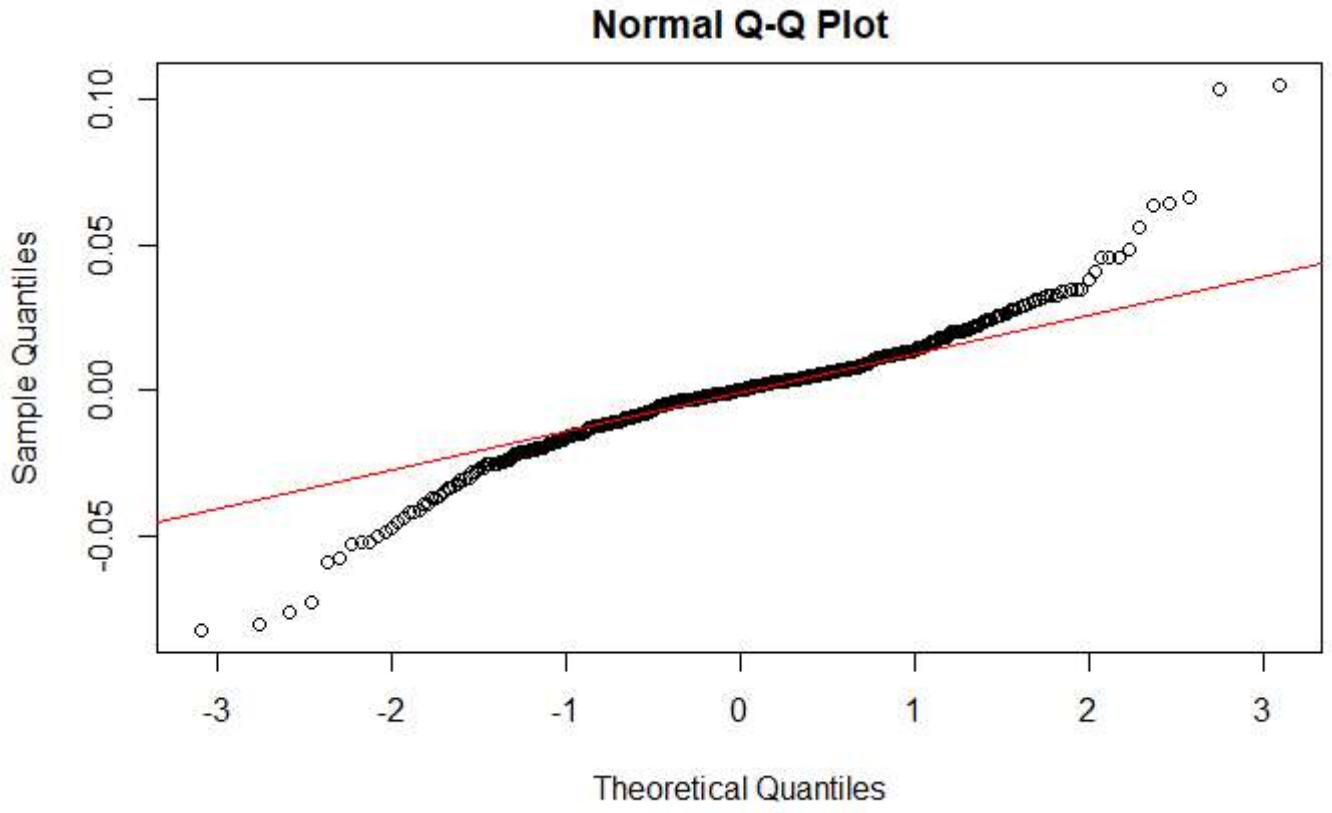
Q-Q plots for assessing normality

Task: Compare the plot with simulated datasets from normal, Student t and uniform distributions generated with the rnorm(), rt() and runif() functions.

Result: The data are from a normal distribution since the dots are close to the red line.The deviation at the very end is acceptable.

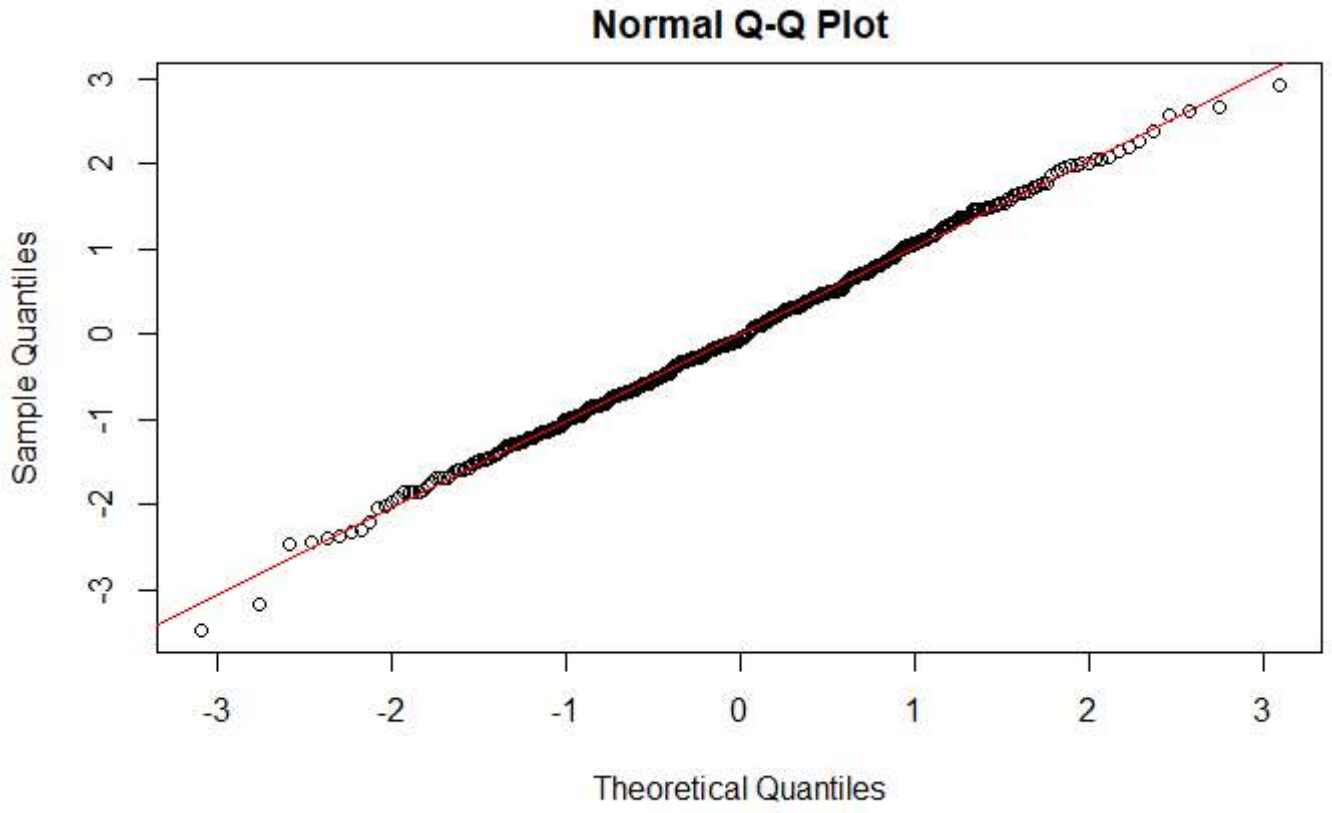
Hide

```
# Make a Q-Q plot of djx and add a red line
qqnorm(djx)
qqline(djx, col = "red")
```



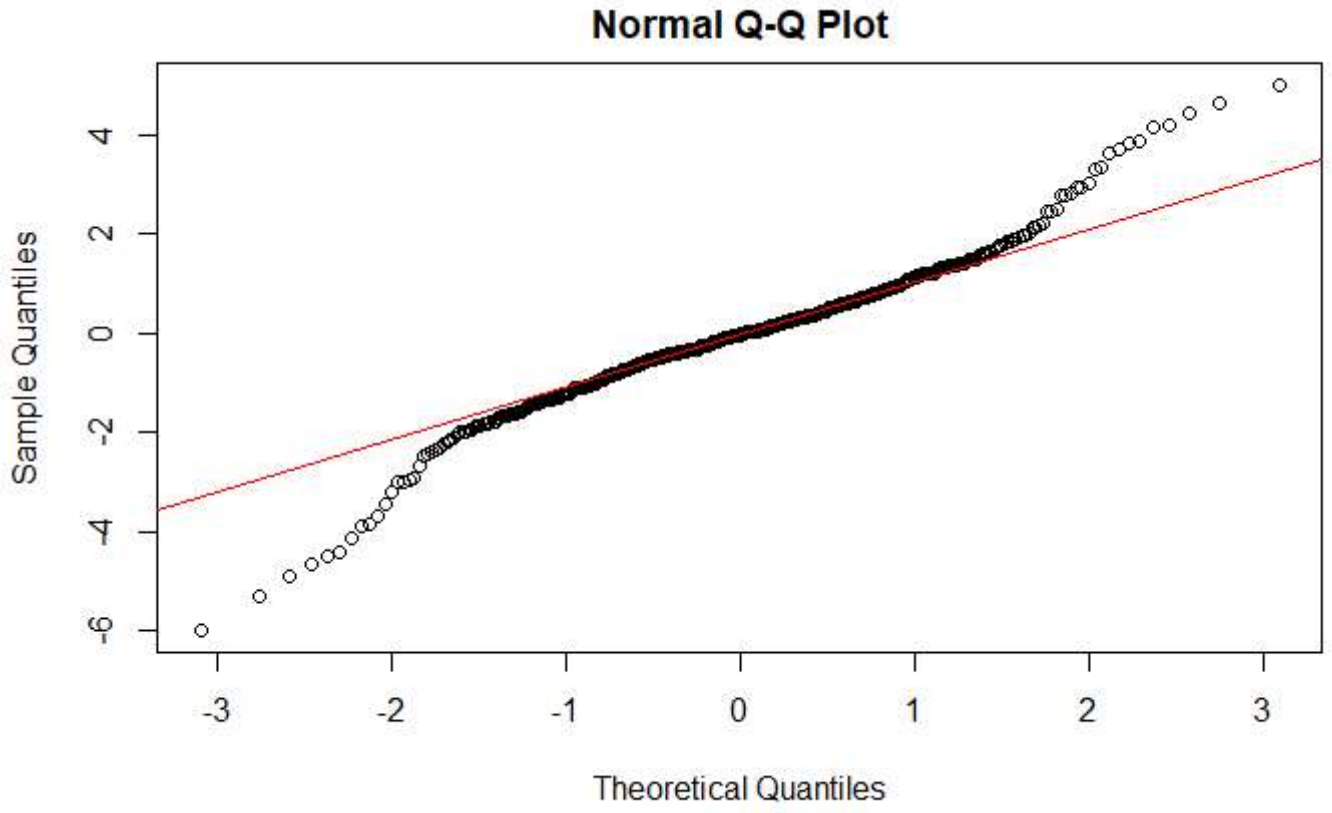
Hide

```
# Calculate the length of djx as n
n <- length(djx)
# Generate n standard normal variables, make a Q-Q plot, add a red line
x1 <- rnorm(n)
qqnorm(x1)
qqline(x1, col = "red")
```



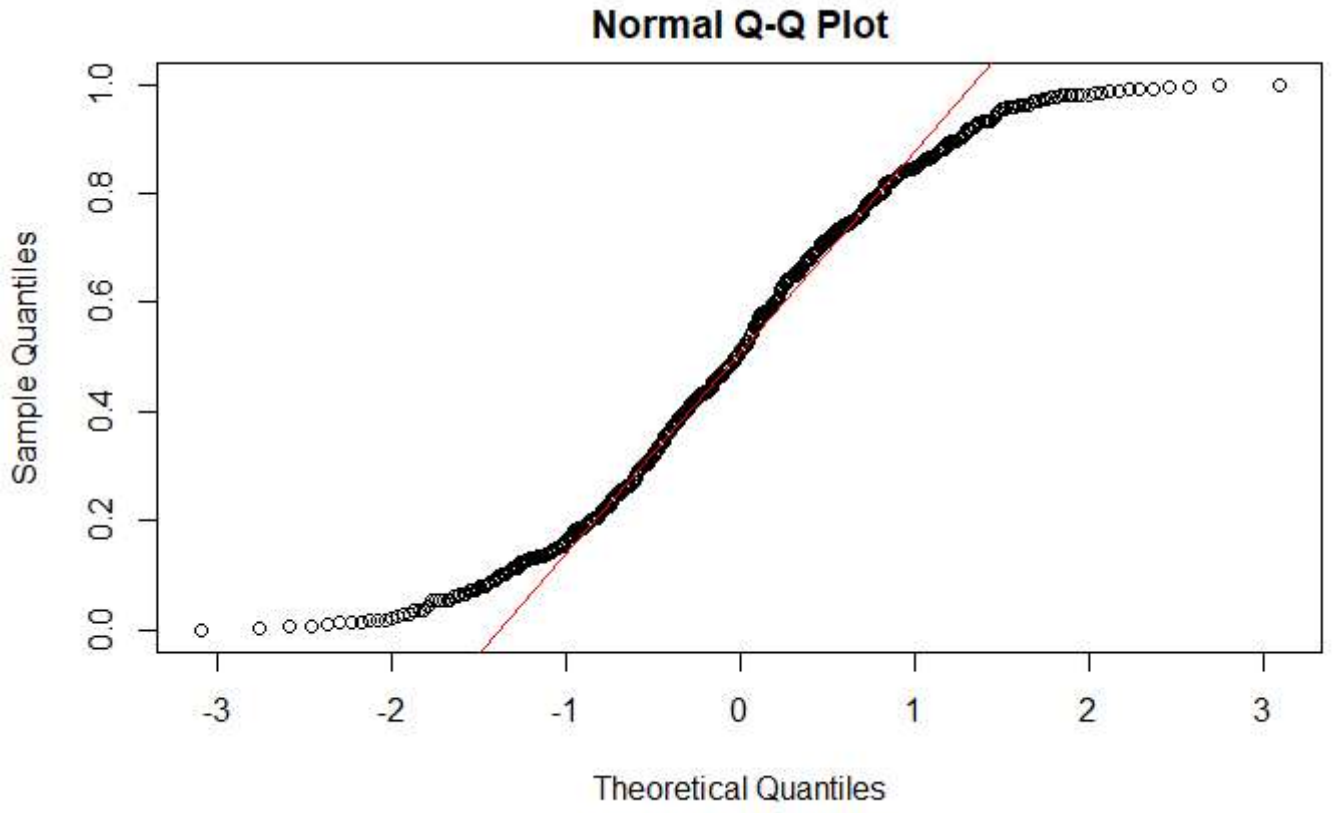
Hide

```
# Generate n Student t variables, make a Q-Q plot, add a red line
x2 <- rt(n, df = 4)
qqnorm(x2)
qqline(x2, col = "red")
```



Hide

```
# Generate n standard uniform variables, make a Q-Q plot, add red line
x3 <- runif(n)
qqnorm(x3)
qqline(x3, col = "red")
```



Skewness, kurtosis and the Jarque-Bera test

. Skewness (b) is a measure of asymmetry . Kurtosis (k) is a measure of heavy-tailedness . Skewness and kurtosis of normal are o and 3, respectively

The Jarque-Bera test . Compares skewness and kurtosis of data with theoretical normal values (o and 3) . Detects skewness, heavy tails, or both

$$T = \frac{1}{6}n\left(b^2 + \frac{1}{4}(k-3)^2\right)$$

Jarque-Bera Test

Code to use with vector data:

jarque.test(goldx_q) tfit <- fit.st(goldx_q) tpars <- tfit$par.ests tpars[1]

Numerical tests of normality

Task:

Result: The return distributions of the Dow Jones stocks all have high kurtosis and some of them are quite skewed. Based on the result of the Jarque-Bera Test, the hypothesis of normality can be

rejected as the test on all of the columns resulted in a p-value < .01. Skewness appear normal ~0 and kurtosis appears leptojurtic as it is >3.

Hide

```
djreturns_1 <- read_csv("C:/Users/Y/Documents/1.R_scripts/ML/Risk_Modeling/Quantitative Risk Man
agement in R/djreturns_1.csv",
    col_types = cols(date = col_date(format = "%m/%d/%Y")))
djreturns_1_xts <- xts(djreturns_1[,-1], order.by = as.POSIXct(djreturns_1$date, format = "%m.%
d.%Y %H:%M"))
# Calculate skewness and kurtosis of djx
skewness(djx)
```

```
[1] 0.1291381
```

Hide

```
kurtosis(djx)
```

```
[1] 7.561201
```

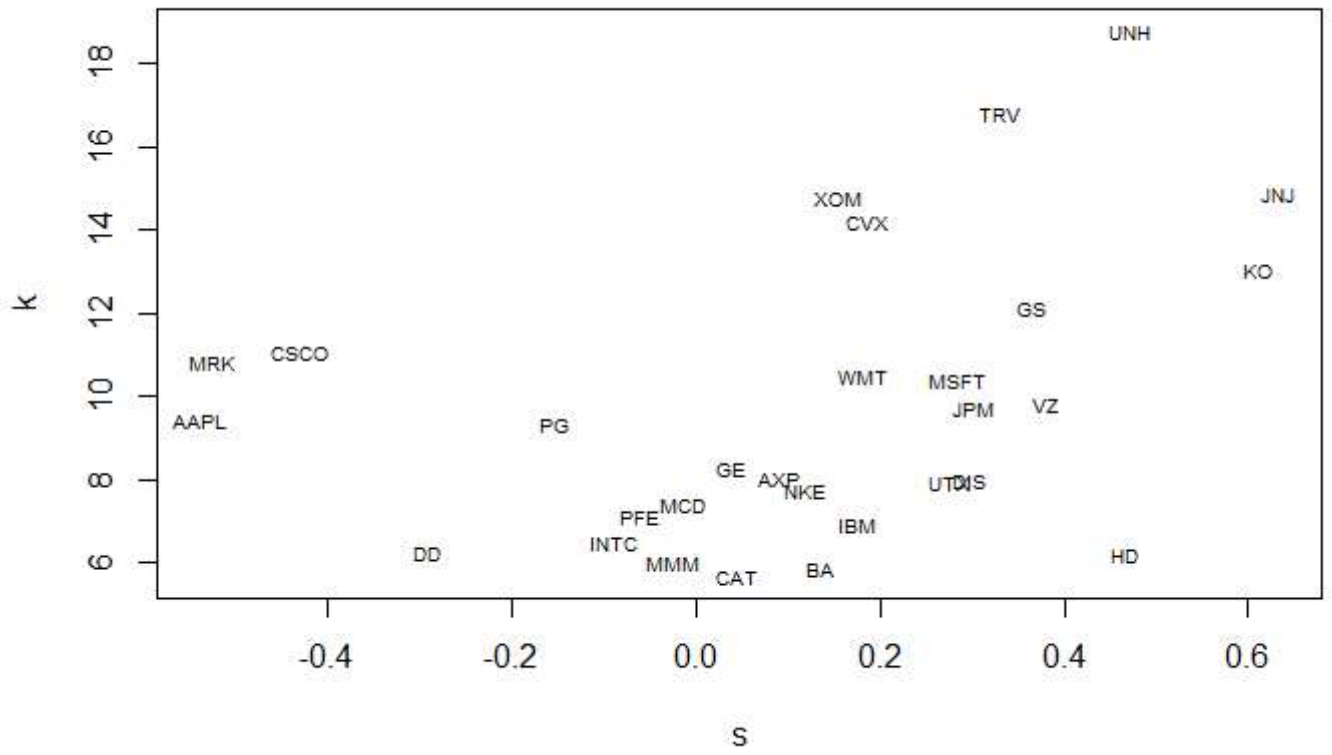
Hide

```
# Carry out a Jarque-Bera test for djx
jarque.test(djx)
```

```

    Jarque-Bera Normality Test

data:  djx
JB = 438.3, p-value < 2.2e-16
alternative hypothesis: greater
```

Hide

```
# Calculate skewness and kurtosis of djreturns
s <- apply(djreturns_1_xts, 2, skewness) #rows=1, columns=2
k <- apply(djreturns_1_xts, 2, kurtosis)
# Plot k against s and add text labels to identify stocks
plot(s, k, type = "n")
text(s, k, names(s), cex = 0.6)
```



Hide

```
# Carry out Jarque-Bera tests for each constituent in djreturns
apply(djreturns_1_xts, 2, jarque.test)
```

```
$AAPL

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 1799.7, p-value < 2.2e-16
alternative hypothesis: greater


$AXP

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 1066.1, p-value < 2.2e-16
alternative hypothesis: greater


$BA

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 347.08, p-value < 2.2e-16
alternative hypothesis: greater


$CAT

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 304.64, p-value < 2.2e-16
alternative hypothesis: greater


$CSCO

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 2758.9, p-value < 2.2e-16
alternative hypothesis: greater


$CVX

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 5268.2, p-value < 2.2e-16
alternative hypothesis: greater
```

```
$DD

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 459.61, p-value < 2.2e-16
alternative hypothesis: greater


$DIS

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 1053.4, p-value < 2.2e-16
alternative hypothesis: greater


$GE

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 1176.9, p-value < 2.2e-16
alternative hypothesis: greater


$GS

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 3511.3, p-value < 2.2e-16
alternative hypothesis: greater


$HD

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 470.66, p-value < 2.2e-16
alternative hypothesis: greater


$IBM

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 660.82, p-value < 2.2e-16
alternative hypothesis: greater
```

```
$INTC

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 510.51, p-value < 2.2e-16
alternative hypothesis: greater


$JNJ

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 5987.3, p-value < 2.2e-16
alternative hypothesis: greater


$JPM

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 1906.3, p-value < 2.2e-16
alternative hypothesis: greater


$KO

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 4310.6, p-value < 2.2e-16
alternative hypothesis: greater


$MCD

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 809.57, p-value < 2.2e-16
alternative hypothesis: greater


$MMM

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 379.55, p-value < 2.2e-16
alternative hypothesis: greater
```

```
$MRK

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 2635.7, p-value < 2.2e-16
alternative hypothesis: greater


$MSFT

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 2307.9, p-value < 2.2e-16
alternative hypothesis: greater


$NKE

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 944.42, p-value < 2.2e-16
alternative hypothesis: greater


$PFE

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 717.42, p-value < 2.2e-16
alternative hypothesis: greater


$PG

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 1690.9, p-value < 2.2e-16
alternative hypothesis: greater


$TRV

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 8039.2, p-value < 2.2e-16
alternative hypothesis: greater
```

```
$UNH

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 10514, p-value < 2.2e-16
alternative hypothesis: greater


$UTX

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 1044, p-value < 2.2e-16
alternative hypothesis: greater


$VZ

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 1969.1, p-value < 2.2e-16
alternative hypothesis: greater


$WMT

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 2351.9, p-value < 2.2e-16
alternative hypothesis: greater


$XOM

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 5812.4, p-value < 2.2e-16
alternative hypothesis: greater
```

Testing normality for longer time horizons

Longer-intervals returns are expected to be more normal than shorter-interval returns.

Task: Test daily, weekly, and monthly log-returns for 29 of the Dow Jones stocks for the period 2008-2011 for normality.

Result: Although the p-values get larger, all monthly returns other than for Chevron (CVX), 3M (MMM), and Pfizer (PFE) still fail the normality test since only the p-value for those three is > .05.

Hide

```
djx_d <- read_csv("C:/Users/Y/Documents/1.R_scripts/ML/Risk_Modeling/Quantitative Risk Managemen
t in R/djx_d.csv",
    col_types = cols(date = col_date(format = "%m/%d/%Y")))
djx_d_xts <- xts(djx_d[,-1], order.by = as.POSIXct(djx_d$date, format = "%m.%d.%Y %H:%M"))
#x_1_xts <- as.xts(x_1[,-1], order.by = x_1$date, "%Y-%m-%d")
# Calculate weekly and monthly log-returns from djx_d
djx_w <- apply.weekly(djx_d_xts, colSums)
djx_m <- apply.monthly(djx_d_xts, colSums)
# Calculate the p-value for each series in djx_d
apply(djx_d_xts, 2, function(v){jarque.test(v)$p.value})
```

```
AAPL  AXP   BA  CAT CSCO  CVX   DD  DIS   GE   GS   HD  IBM INTC  JNJ  JPM   KO  MCD  MMM  MRK M
SFT  NKE  PFE
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0    0
  PG  TRV  UNH  UTX   VZ  WMT  XOM
   0    0    0    0    0    0    0
```

Hide

```
# Calculate the p-value for each series in djx_w
apply(djx_w, 2, function(v){jarque.test(v)$p.value})
```

```
AAPL  AXP   BA  CAT CSCO  CVX   DD  DIS   GE   GS   HD  IBM INTC  JNJ  JPM   KO  MCD  MMM  MRK M
SFT  NKE  PFE
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  0    0    0
  PG  TRV  UNH  UTX   VZ  WMT  XOM
   0    0    0    0    0    0    0
```

Hide

```
# Calculate the p-value for each series in djx_m
apply(djx_m, 2, function(v){jarque.test(v)$p.value})
```

```
       AAPL          AXP           BA          CAT         CSCO          CVX           DD
    DIS
0.000000e+00 0.000000e+00 0.000000e+00 6.750975e-09 0.000000e+00 1.488565e-03 1.798506e-11 0.000
000e+00
         GE           GS           HD          IBM         INTC          JNJ          JPM
     KO
7.144151e-10 8.472104e-08 1.446676e-01 5.554213e-11 0.000000e+00 5.029466e-08 0.000000e+00 3.550
985e-05
        MCD          MMM          MRK         MSFT          NKE          PFE           PG
    TRV
7.971729e-10 3.330375e-01 8.828102e-08 2.588949e-10 0.000000e+00 6.212808e-01 0.000000e+00 0.000
000e+00
        UNH          UTX           VZ          WMT          XOM
0.000000e+00 0.000000e+00 5.954856e-05 1.154207e-03 4.488559e-03
```

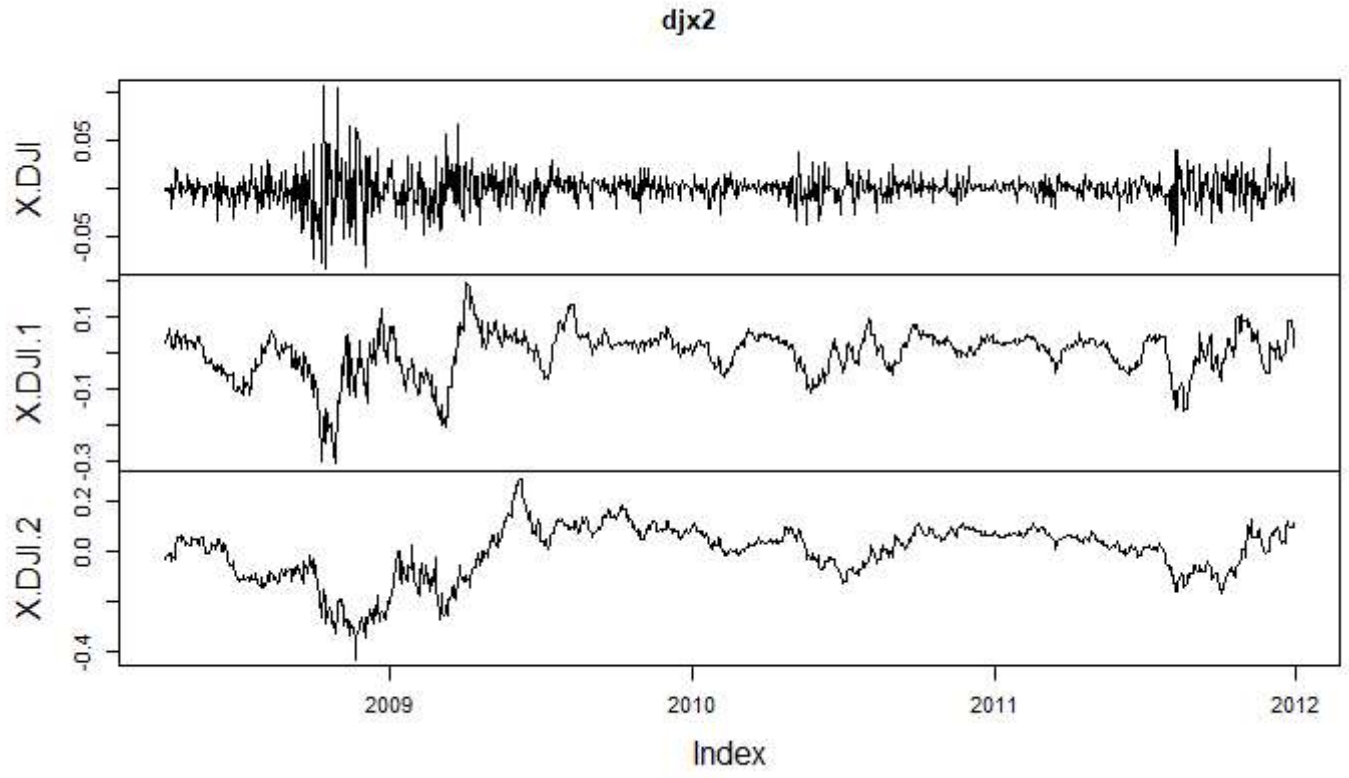
Overlapping returns

Task: Calculate moving sums of different intervals from the Dow Jones data set. Then find the skewness and kurtosis of the resulting data and conduct the Jarque-Bera test to test for normality.

Result: These overlapping returns are highly correlated. The hypothesis of normality can be rejected as the test on all of the columns resulted in a p-value < .01.

Hide

```
djx_1 <- read_csv("C:/Users/Y/Documents/1.R_scripts/ML/Risk_Modeling/Quantitative Risk Managemen
t in R/djx_1.csv",
    col_types = cols(date = col_date(format = "%m/%d/%Y")))
djx_1_xts <- as.xts(djx_1[,-1], order.by = djx_1$date, "%Y-%m-%d")
# Calculate a 21-day moving sum of djx
djx21 <- rollapplyr(djx_1_xts, width=21, FUN=sum)[-(1:20)]
# Calculate a 63-day moving sum of djx
djx63 <- rollapplyr(djx_1_xts, width=63, FUN=sum)[-(1:62)]
# Merge the three series and plot
djx2 <- merge(djx_1_xts, djx21, djx63, all=FALSE)
plot.zoo(djx2)
```



Hide

```
# Compute the skewness and kurtosis for each series in djx2
apply(djx2, 2, skewness)
```

```
     X.DJI     X.DJI.1     X.DJI.2
-0.01572983 -1.15987077 -0.87599211
```

Hide

```
apply(djx2, 2, kurtosis)
```

```
   X.DJI  X.DJI.1  X.DJI.2
9.247017 5.845877 3.851254
```

Hide

```
# Conduct the Jarque-Bera test to each series in djx2
apply(djx2, 2, jarque.test)
```

```
$X.DJI

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 1538.3, p-value < 2.2e-16
alternative hypothesis: greater


$X.DJI.1

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 531.35, p-value < 2.2e-16
alternative hypothesis: greater


$X.DJI.2

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 149.55, p-value < 2.2e-16
alternative hypothesis: greater
```

The Student t distribution

$$f_X(x) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sigma\sqrt{\nu\pi}\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{(x-\mu)^2}{\nu\sigma^2}\right)^{-\frac{\nu+1}{2}}$$

Student T Equation

. This distribution has three parameters: 11, a, v . Small values of v give heavier tails . As v gets larger the distribution tends to normal

Fit the Student t distibution with the Method of Maximum Likelihood

Fitting t distribution to data

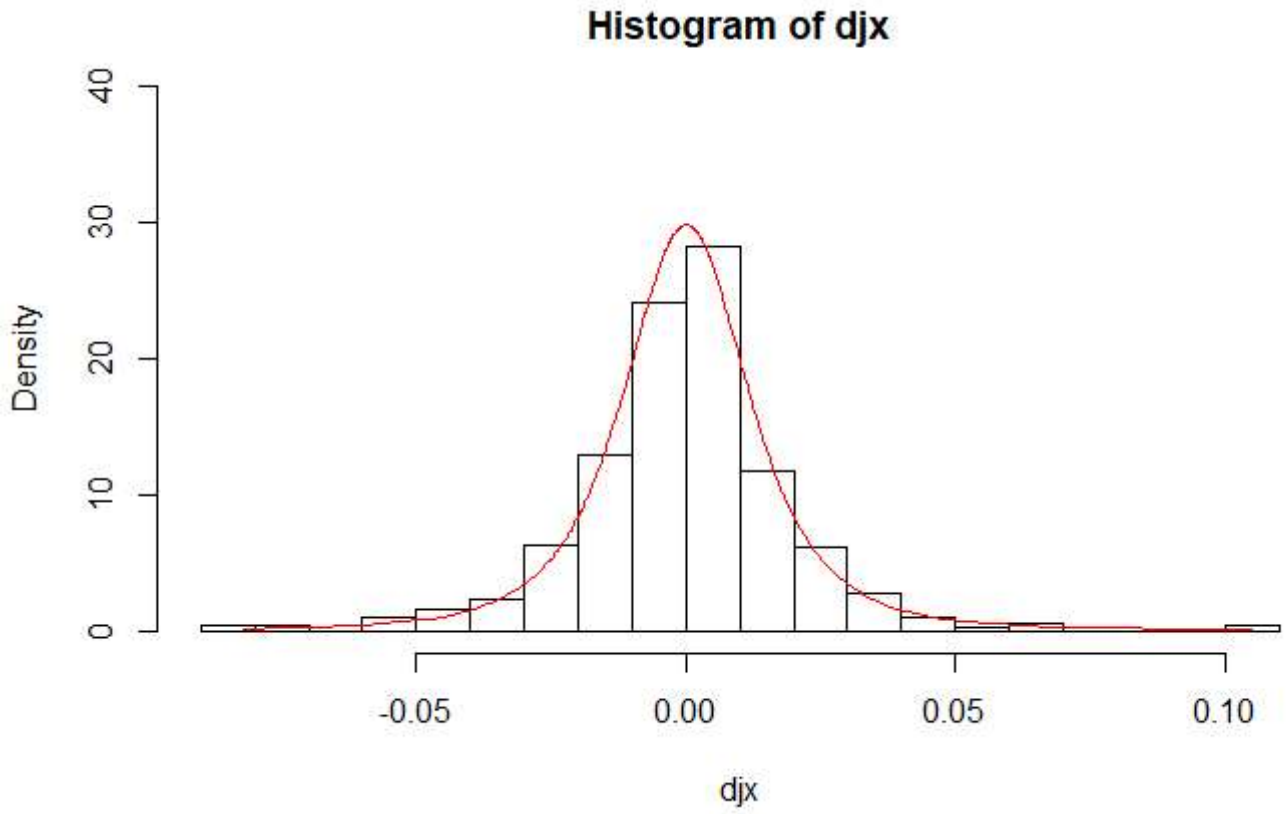
A Student t distribution is generally a much better fit to daily, weekly, and monthly returns than a normal distribution

Task: Fit a Student t distribution to the daily log-returns of the Dow Jones index from 2008-2011 contained in djx. Then, plot a histogram of the data and superimpose a red line to the plot showing the fitted t density.

Result: The fitted Student t distribution looks a lot better than the normal did.

Hide

```
# Fit a Student t distribution to djx
tfit <- fit.st(djx)
# Define tpars, nu, mu, and sigma
tpars <- tfit$par.ests
mu <- tpars[2]
nu <- tpars[1]
sigma <- tpars[3]
# Plot a histogram of djx
hist(djx, nclass = 20, probability = TRUE, ylim = range(0, 40))
# Compute the fitted t density at the values djx
yvals <- dt((djx - mu)/sigma, df = nu)/sigma
# Superimpose a red line to show the fitted t density
lines(djx, yvals, col = "red")
```



Testing FX returns for normality

Task: The dataset fx_d contains daily log-returns of the EUR/USD, GBP/USD and JPY/USD exchange rates for the period 2001-2015, and the dataset fx_m contains the corresponding monthly log-

returns. Both are multivariate. Which of the monthly log-return series appears the most normal?

Result: The JPY/USD exchange rate log-returns appear to be the most normal. Larger value of nu, the closer to a normal distribution. Nu for USD.JPY is 18.4. Nu is the estimated degree of freedom.

Hide

```
# Fit a Student t distribution to each of the series in fx_m
apply(fx_m_xts[,1:2], 2, function(v){fit.st(v)$par.ests})
```

```
           GBP.USD     EUR.USD
nu    6.0444926719 4.912878252
mu    0.0008655583 0.001947311
sigma 0.0204357970 0.023764956
```

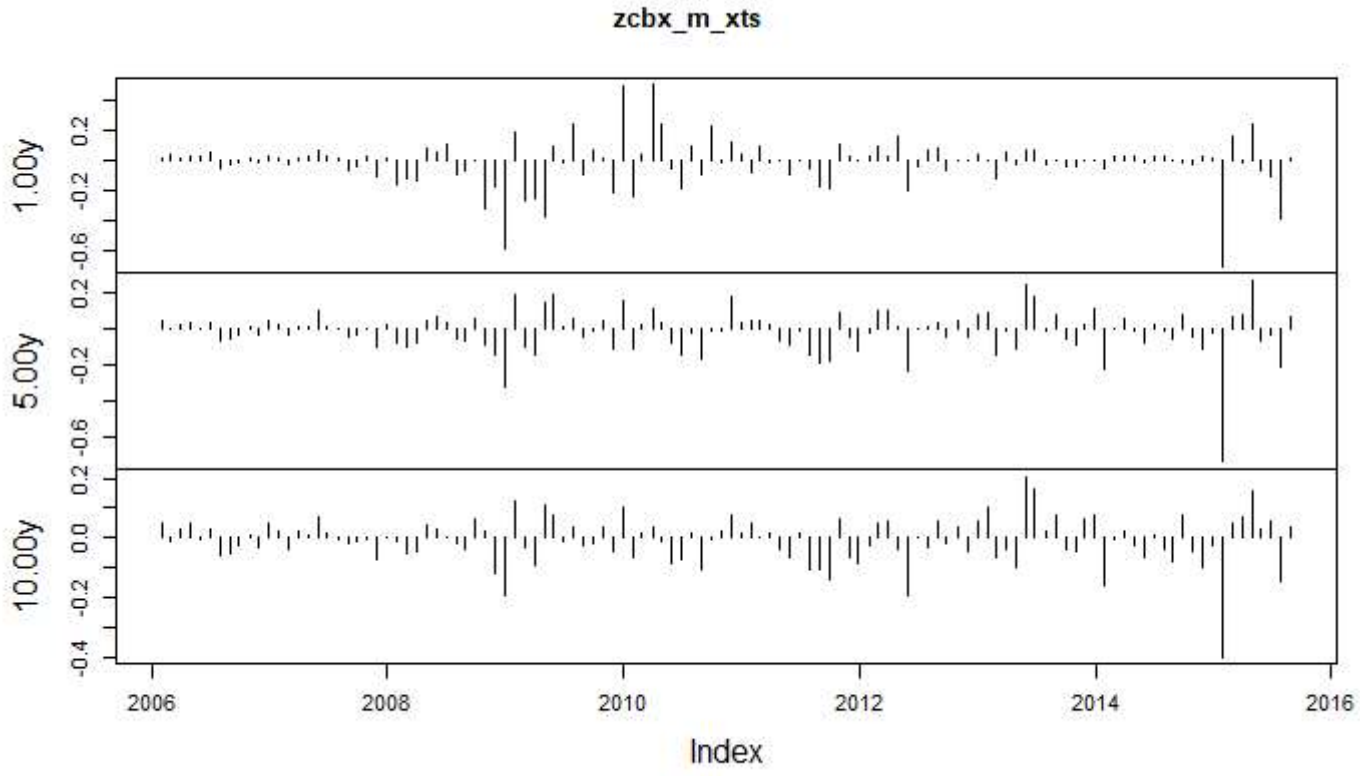
Testing interest-rate returns for normality

Task: The object zcbx_m contains monthly log-return series for the 1-year, 5-year and 10-year Canadian zero-coupon bond yields. The object zcbx2_m contains the corresponding simple returns. Both are multivariate. Plot these interest rate return series and then examine their normality with Q-Q plots and Jarque-Bera tests.

Result: The log-returns show clearer evidence of non-normality than the simple returns in this case. The simple monthly returns for the 5 and 10 year yields did not fail the normality test.

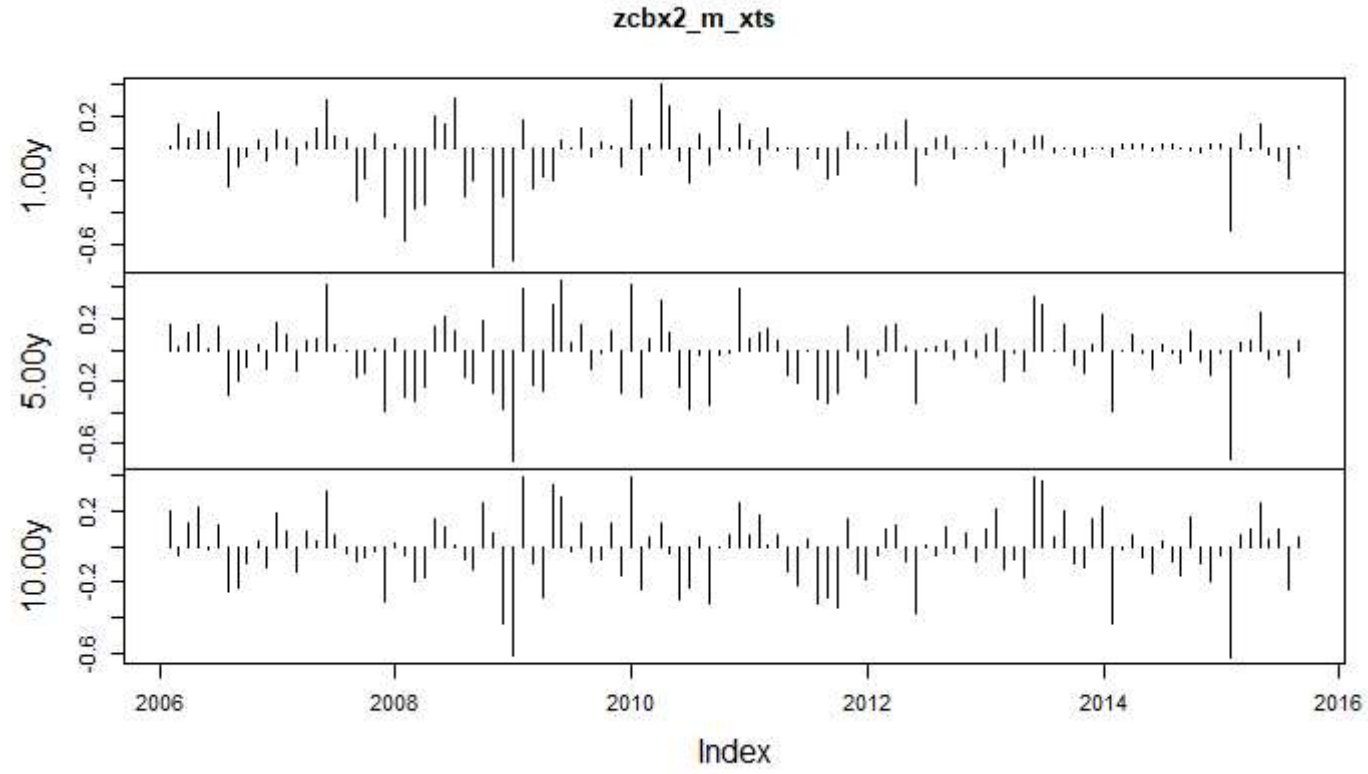
Hide

```
zcbx_m <- read_csv("C:/Users/Y/Documents/1.R_scripts/ML/Risk_Modeling/Quantitative Risk Manageme
nt in R/zcbx_m.csv",
    col_types = cols(`1.00y` = col_number(),
        `10.00y` = col_number(), `5.00y` = col_number(),
        date = col_date(format = "%m/%d/%Y")))
zcbx_m_xts <- as.xts(zcbx_m[,-1], order.by = zcbx_m$date, "%Y-%m-%d")
zcbx2_m <- read_csv("C:/Users/Y/Documents/1.R_scripts/ML/Risk_Modeling/Quantitative Risk Managem
ent in R/zcbx2_m.csv",
    col_types = cols(`1.00y` = col_number(),
        `10.00y` = col_number(), `5.00y` = col_number(),
        date = col_date(format = "%m/%d/%Y")))
zcbx2_m_xts <- as.xts(zcbx2_m[,-1], order.by = zcbx2_m$date, "%Y-%m-%d")
# Plot the interest-rate return series zcbx_m and zcbx2_m
plot.zoo(zcbx_m_xts, type = "h")
```



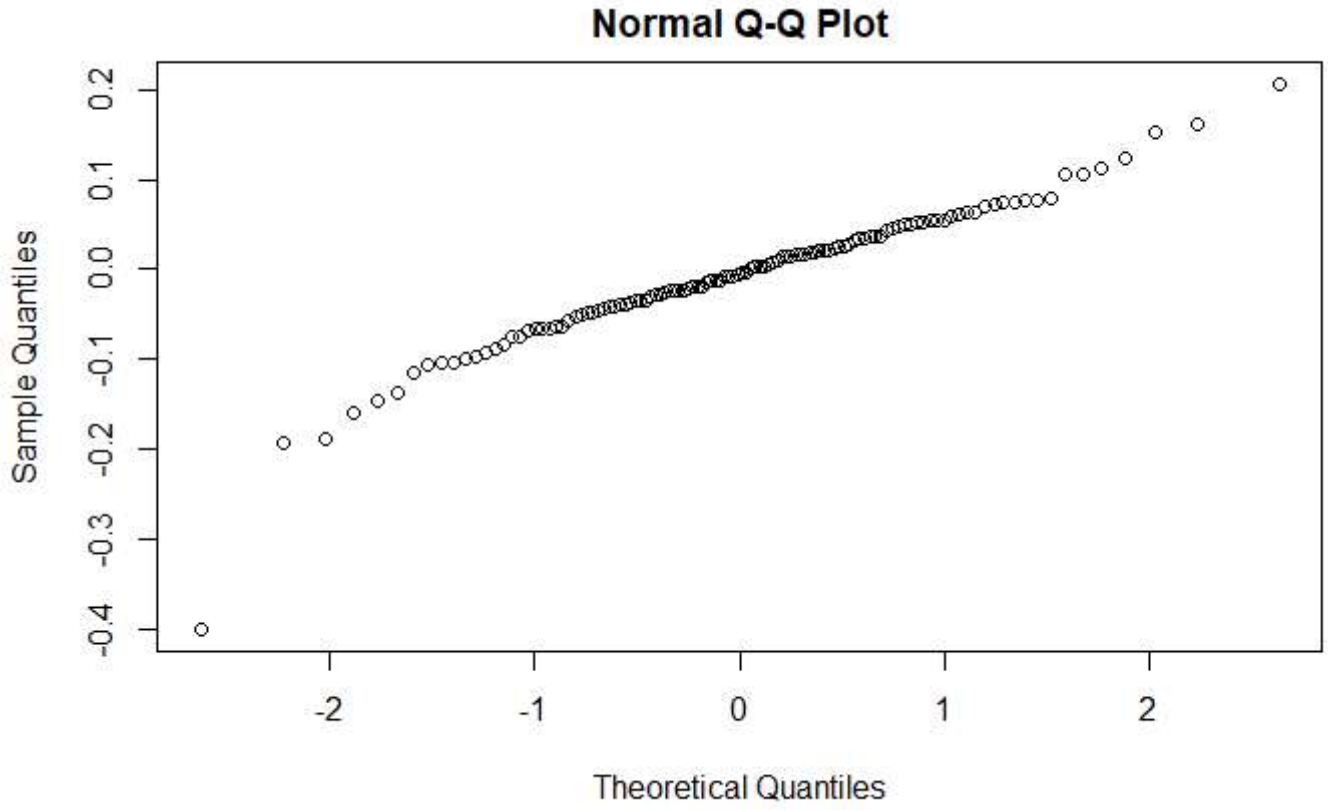
Hide

```
plot.zoo(zcbx2_m_xts, type = "h")
```



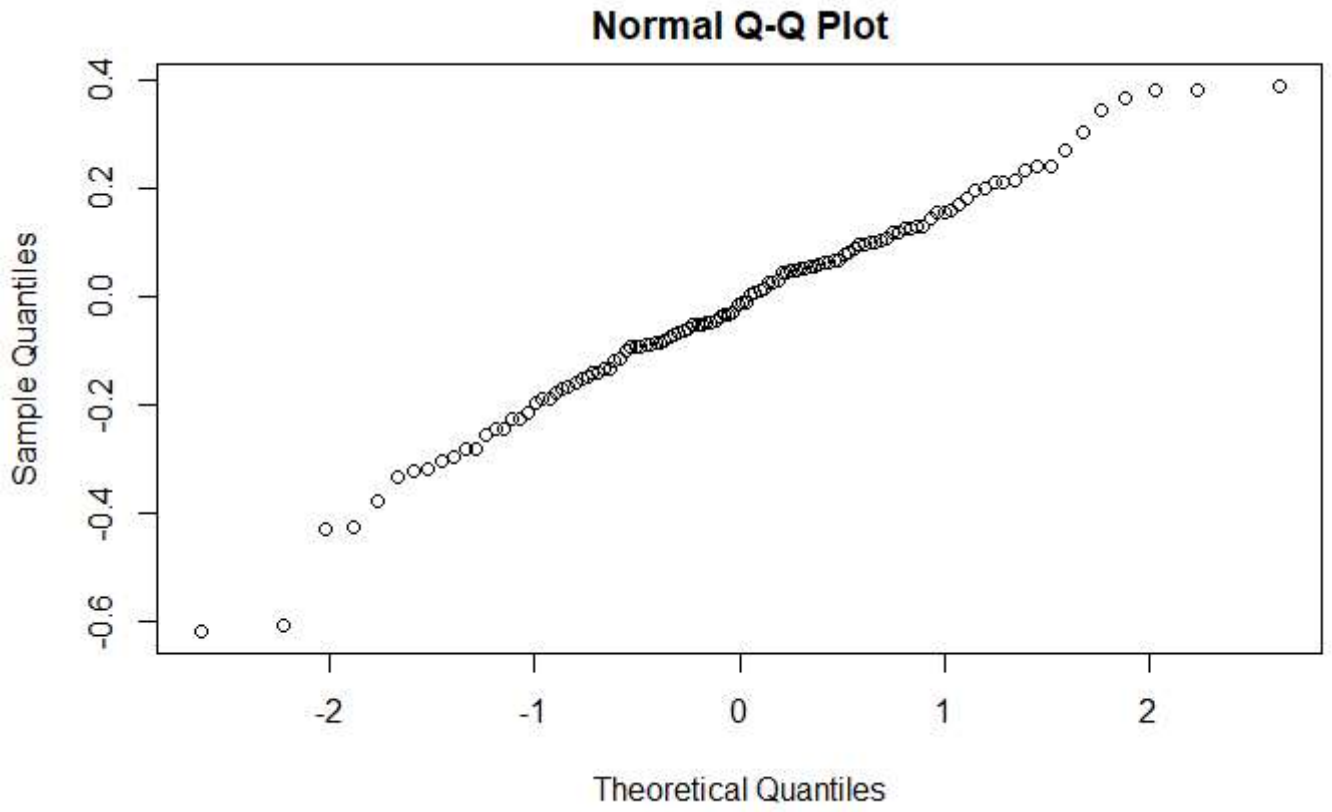
Hide

```
# Make Q-Q plots of the 3rd component series of zcbx_m_xts and zcbx2_m
qqnorm(zcbx_m_xts[,3])
```



Hide

```
qqnorm(zcbx2_m_xts[,3])
```



Hide

```
# Compute the kurtosis of each series in zcbx_m_xts and zcbx2_m
apply(zcbx_m_xts, 2, kurtosis)
```

```
    1.00y     5.00y    10.00y
 5.449968 10.653175  4.854350
```

Hide

```
apply(zcbx2_m_xts, 2, kurtosis)
```

```
    1.00y     5.00y    10.00y
3.0023876 0.6646095 0.4515469
```

Hide

```
# Conduct the Jarque-Bera test on each series in zcbx_m_xts and zcbx2_m
apply(zcbx_m_xts, 2, jarque.test)
```

```
$`1.00y`

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 164.96, p-value < 2.2e-16
alternative hypothesis: greater


$`5.00y`

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 644.21, p-value < 2.2e-16
alternative hypothesis: greater


$`10.00y`

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 142.27, p-value < 2.2e-16
alternative hypothesis: greater
```

Hide

```
apply(zcbx2_m_xts, 2, jarque.test)
```

```
$`1.00y`

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 77.596, p-value < 2.2e-16
alternative hypothesis: greater


$`5.00y`

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 5.7135, p-value = 0.05746
alternative hypothesis: greater


$`10.00y`

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 4.0954, p-value = 0.129
alternative hypothesis: greater
```

Characteristics of volatile return series

Real returns often show volatility clustering

Spotting a volatile time series

Task: Plot the Dow Jones log-returns for 2008-2011 alongside independent and identically distributed (iid) normal data and iid Student t data.

Result:

Hide

```
ndatax <- xts(ndata, time(djx_xts))
```

```
Error in xts(ndata, time(djx_xts)) : NROW(x) must match length(order.by)
```

Estimating serial correlations

If serial dependencies are present, they can be exploited for prediction. For example, when there is volatility clustering, and you know your in the middle of a particularly volatile period, then you know that the probability of another extreme market move is increased.

.Sample autocorrelations Sample autocorrelation function (acf) measures correlation between variables separated by lag (k)

Stationarity is implicitly assumed: . Expected return constant over time Variance of return distribution always the same . Correlation between returns k apart always the same Notation for sample autocorrelation: P(k)

The sample acf plot or correlogram . Because of the frequent sign changes, the strong serial dependence is hidden when you look at the ACF plot. But it is revealed when you remove the sign changes with the absolute function.

Using acf plots to reveal volatility

Task: For the Dow Jones returns from 2008-11 in djx and the simulated normal and t-distributed data in ndata and tdata, respectively, you will calculate and plot the sample autocorrelation functions (acf) using the command acf().

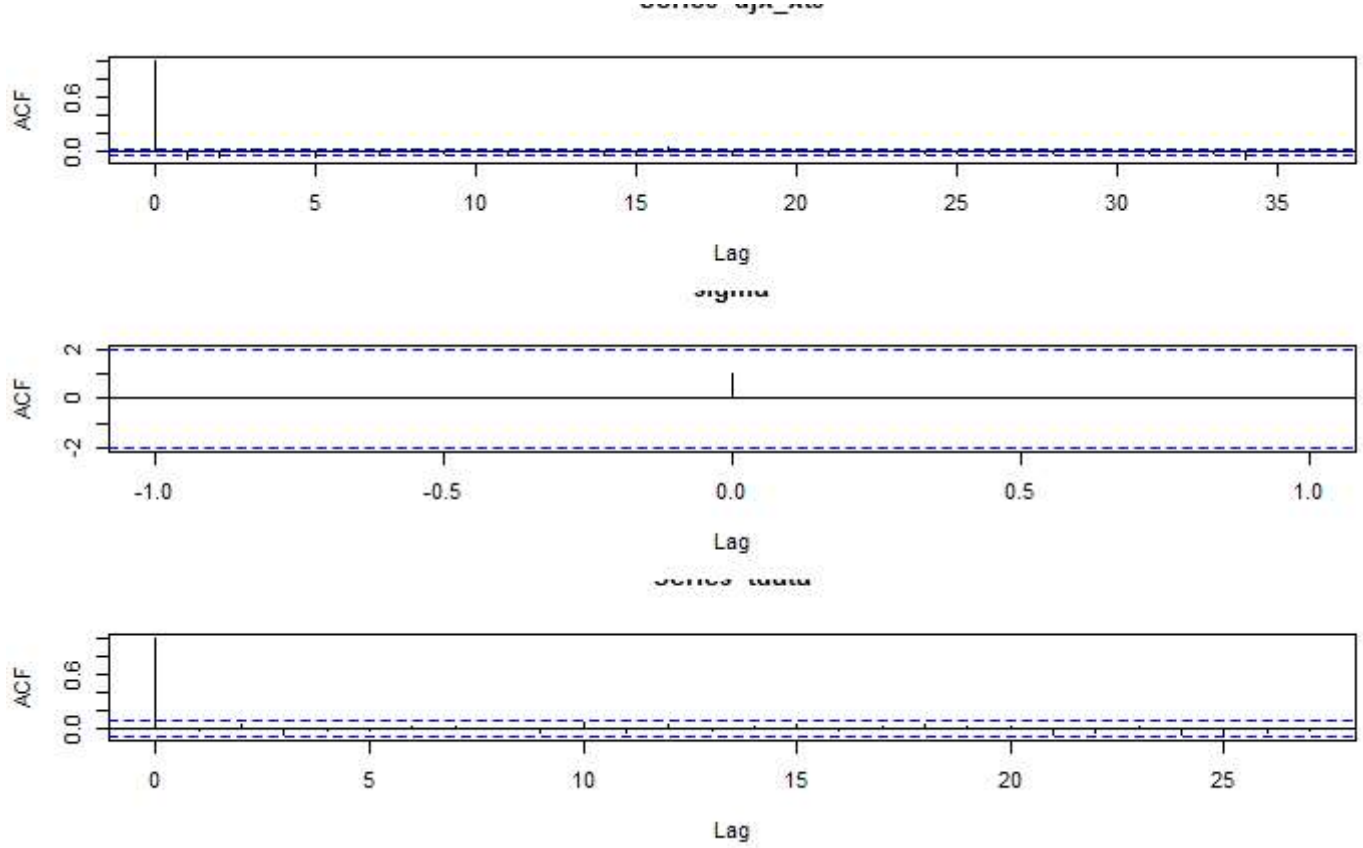
Result: It is very common to see strong serial correlation in absolute and squared return series.

Hide

```
# Set up a plot region to show 3 plots at a time
par(mfrow = c(3, 1))
# Plot the acfs of djx, ndata and tdata
acf(djx_xts)
acf(ndata)
```

Hide

```
acf(tdata)
```

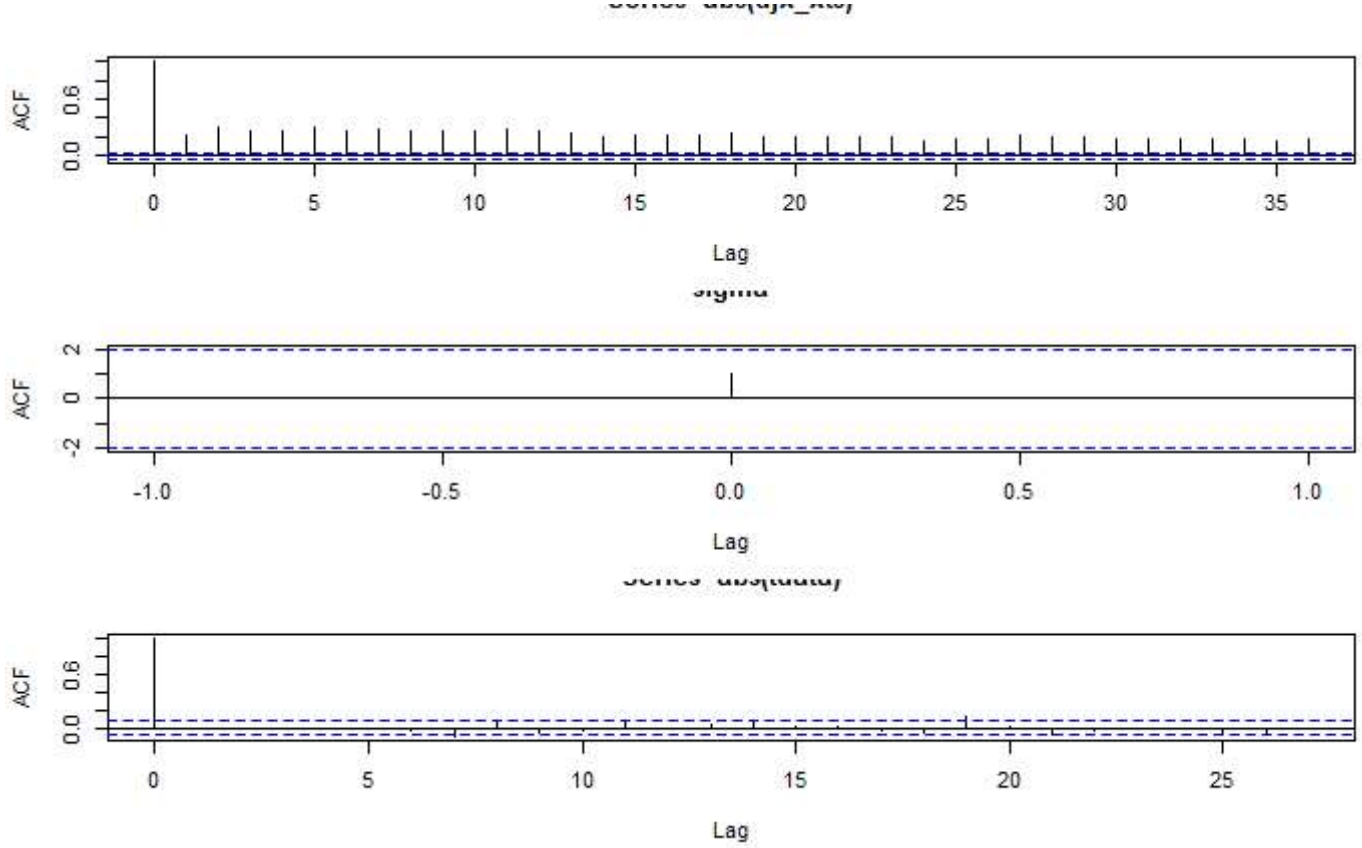


Hide

```
# Plot the acfs of the absolute values
acf(abs(djx_xts))
acf(abs(ndata))
```

Hide

```
acf(abs(tdata))
```

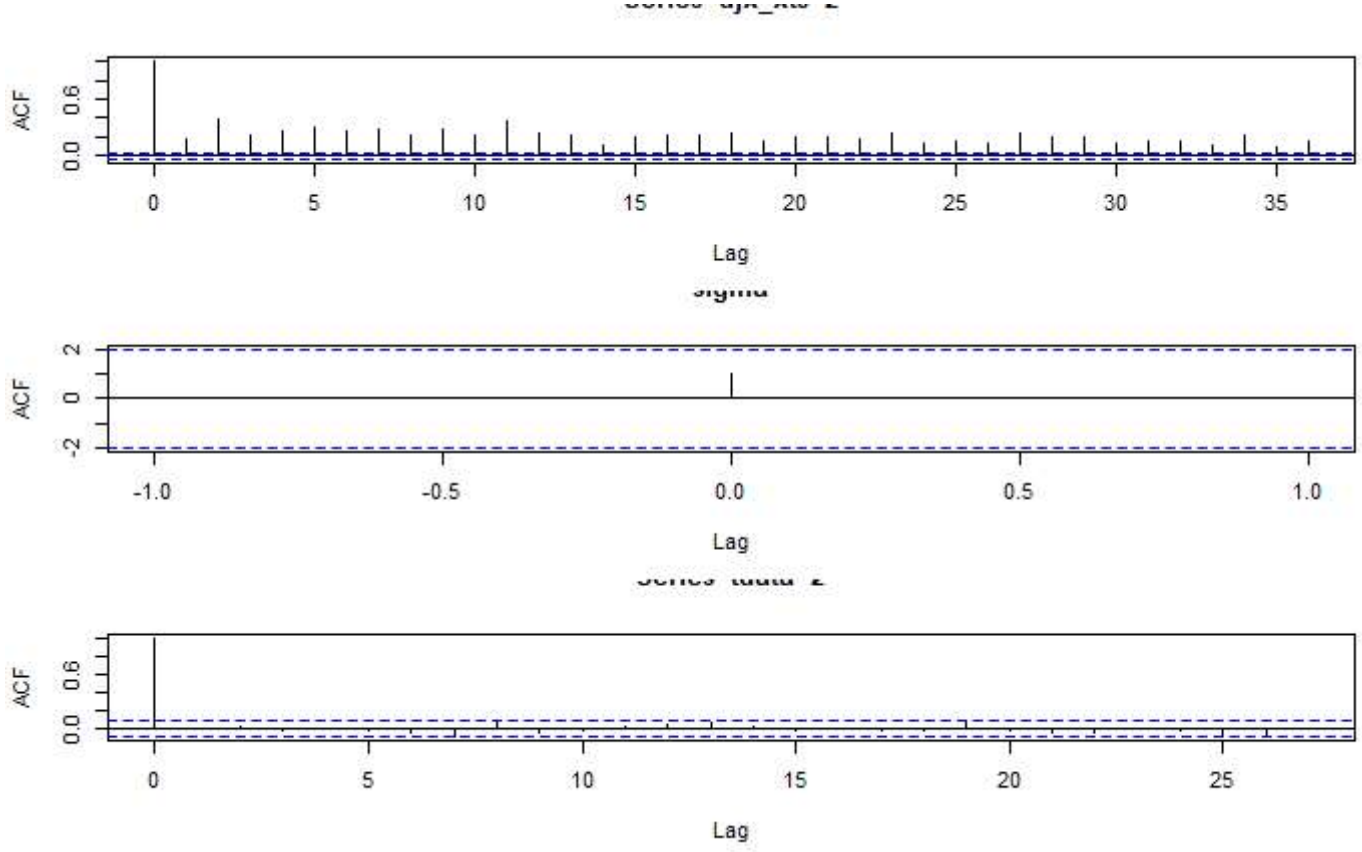


Hide

```
# Plot the acfs of the squares of the values
acf(djx_xts^2)
acf(ndata^2)
```

Hide

```
acf(tdata^2)
```



The Ljung-Box test

Based on the sum of squared values of the sample autocorrelations. If the data are IID, the sample autocorrelations should mostly be close to zero. So the larger the value of the Ljung-Box statistic, the greater the evidence that there is serial dependence in the series.

Applying Ljung-Box tests to return data

Task: Carry out a Ljung-Box test for serial correlation on the time series djx which contains the Dow Jones daily index returns for 2008-2011, as well as on all the individual equity return series in djall which contains the Dow Jones data for 2006-2015. Implement this test on both the raw return series and the absolute values of the series.

Conclusion: the hypothesis of no serial correlation is rejected for many of the raw return series, it is rejected overwhelmingly for all of the absolute value series.

Hide

```
djall <- read_csv("C:/Users/Y/Documents/1.R_scripts/ML/Risk_Modeling/Quantitative Risk Managemen
t in R/djall.csv",
    col_types = cols(date = col_date(format = "%m/%d/%Y")))
djall_xts <- as.xts(djall[,-1], order.by = djall$date, "%d-%m-%Y")
# Apply the Ljung-Box test to djx
Box.test(djx_1_xts, lag = 10, type = "Ljung")
```

```
    Box-Ljung test

data:  djx_1_xts
X-squared = 34.499, df = 10, p-value = 0.000152
```

Hide

```
# Apply the Ljung-Box test to absolute values of djx
Box.test(abs(djx_1_xts), lag = 10, type = "Ljung")
```

```
    Box-Ljung test

data:  abs(djx_1_xts)
X-squared = 1082.6, df = 10, p-value < 2.2e-16
```

Hide

```
# Apply the Ljung-Box test to all return series in djall
apply(djall_xts, 2, Box.test, lag = 10, type = "Ljung")
```

```
$AAPL

    Box-Ljung test

data:  newX[, i]
X-squared = 14.77, df = 10, p-value = 0.1407


$AXP

    Box-Ljung test

data:  newX[, i]
X-squared = 45.464, df = 10, p-value = 1.793e-06


$BA

    Box-Ljung test

data:  newX[, i]
X-squared = 19.986, df = 10, p-value = 0.02939


$CAT

    Box-Ljung test

data:  newX[, i]
X-squared = 16.596, df = 10, p-value = 0.08379


$CSCO

    Box-Ljung test

data:  newX[, i]
X-squared = 15.651, df = 10, p-value = 0.1101


$CVX

    Box-Ljung test

data:  newX[, i]
X-squared = 63.895, df = 10, p-value = 6.591e-10


$DD

    Box-Ljung test

data:  newX[, i]
```

```
X-squared = 28.255, df = 10, p-value = 0.001643


$DIS

    Box-Ljung test

data:  newX[, i]
X-squared = 31.368, df = 10, p-value = 0.00051


$GE

    Box-Ljung test

data:  newX[, i]
X-squared = 39.436, df = 10, p-value = 2.129e-05


$GS

    Box-Ljung test

data:  newX[, i]
X-squared = 34.993, df = 10, p-value = 0.0001252


$HD

    Box-Ljung test

data:  newX[, i]
X-squared = 12.583, df = 10, p-value = 0.2479


$IBM

    Box-Ljung test

data:  newX[, i]
X-squared = 27.629, df = 10, p-value = 0.002069


$INTC

    Box-Ljung test

data:  newX[, i]
X-squared = 56.775, df = 10, p-value = 1.469e-08


$JNJ

    Box-Ljung test
```

```
data:  newX[, i]
X-squared = 56.686, df = 10, p-value = 1.526e-08


$JPM

    Box-Ljung test

data:  newX[, i]
X-squared = 50.205, df = 10, p-value = 2.447e-07


$KO

    Box-Ljung test

data:  newX[, i]
X-squared = 37.302, df = 10, p-value = 5.018e-05


$MCD

    Box-Ljung test

data:  newX[, i]
X-squared = 39.949, df = 10, p-value = 1.73e-05


$MMM

    Box-Ljung test

data:  newX[, i]
X-squared = 28.284, df = 10, p-value = 0.001626


$MRK

    Box-Ljung test

data:  newX[, i]
X-squared = 19.781, df = 10, p-value = 0.03139


$MSFT

    Box-Ljung test

data:  newX[, i]
X-squared = 40.724, df = 10, p-value = 1.263e-05


$NKE
```

```
    Box-Ljung test

data:  newX[, i]
X-squared = 34.342, df = 10, p-value = 0.0001616


$PFE

    Box-Ljung test

data:  newX[, i]
X-squared = 49.204, df = 10, p-value = 3.737e-07


$PG

    Box-Ljung test

data:  newX[, i]
X-squared = 48.135, df = 10, p-value = 5.865e-07


$TRV

    Box-Ljung test

data:  newX[, i]
X-squared = 131.92, df = 10, p-value < 2.2e-16


$UNH

    Box-Ljung test

data:  newX[, i]
X-squared = 46.841, df = 10, p-value = 1.009e-06


$UTX

    Box-Ljung test

data:  newX[, i]
X-squared = 38.002, df = 10, p-value = 3.793e-05


$VZ

    Box-Ljung test

data:  newX[, i]
X-squared = 37.538, df = 10, p-value = 4.567e-05
```

```
$WMT

    Box-Ljung test

data:  newX[, i]
X-squared = 44.5, df = 10, p-value = 2.676e-06


$XOM

    Box-Ljung test

data:  newX[, i]
X-squared = 117.44, df = 10, p-value < 2.2e-16
```

Hide

```
# Apply the Ljung-Box test to absolute values of all returns in djall
apply(abs(djall_xts), 2, Box.test, lag = 10, type = "Ljung")
```

```
$AAPL

    Box-Ljung test

data:  newX[, i]
X-squared = 910.1, df = 10, p-value < 2.2e-16


$AXP

    Box-Ljung test

data:  newX[, i]
X-squared = 3114.7, df = 10, p-value < 2.2e-16


$BA

    Box-Ljung test

data:  newX[, i]
X-squared = 1078.8, df = 10, p-value < 2.2e-16


$CAT

    Box-Ljung test

data:  newX[, i]
X-squared = 1569.7, df = 10, p-value < 2.2e-16


$CSCO

    Box-Ljung test

data:  newX[, i]
X-squared = 646.82, df = 10, p-value < 2.2e-16


$CVX

    Box-Ljung test

data:  newX[, i]
X-squared = 2197.7, df = 10, p-value < 2.2e-16


$DD

    Box-Ljung test

data:  newX[, i]
```

```
X-squared = 1434, df = 10, p-value < 2.2e-16


$DIS

    Box-Ljung test

data:  newX[, i]
X-squared = 1822.3, df = 10, p-value < 2.2e-16


$GE

    Box-Ljung test

data:  newX[, i]
X-squared = 2335.5, df = 10, p-value < 2.2e-16


$GS

    Box-Ljung test

data:  newX[, i]
X-squared = 2081.1, df = 10, p-value < 2.2e-16


$HD

    Box-Ljung test

data:  newX[, i]
X-squared = 1842.3, df = 10, p-value < 2.2e-16


$IBM

    Box-Ljung test

data:  newX[, i]
X-squared = 1025.3, df = 10, p-value < 2.2e-16


$INTC

    Box-Ljung test

data:  newX[, i]
X-squared = 1063, df = 10, p-value < 2.2e-16


$JNJ

    Box-Ljung test
```

```
data:  newX[, i]
X-squared = 1497.3, df = 10, p-value < 2.2e-16


$JPM

    Box-Ljung test

data:  newX[, i]
X-squared = 3041.6, df = 10, p-value < 2.2e-16


$KO

    Box-Ljung test

data:  newX[, i]
X-squared = 1526.7, df = 10, p-value < 2.2e-16


$MCD

    Box-Ljung test

data:  newX[, i]
X-squared = 1197.4, df = 10, p-value < 2.2e-16


$MMM

    Box-Ljung test

data:  newX[, i]
X-squared = 1274.7, df = 10, p-value < 2.2e-16


$MRK

    Box-Ljung test

data:  newX[, i]
X-squared = 1404.4, df = 10, p-value < 2.2e-16


$MSFT

    Box-Ljung test

data:  newX[, i]
X-squared = 824.76, df = 10, p-value < 2.2e-16


$NKE
```

```
    Box-Ljung test

data:  newX[, i]
X-squared = 1175.8, df = 10, p-value < 2.2e-16


$PFE

    Box-Ljung test

data:  newX[, i]
X-squared = 1187.2, df = 10, p-value < 2.2e-16


$PG

    Box-Ljung test

data:  newX[, i]
X-squared = 1116, df = 10, p-value < 2.2e-16


$TRV

    Box-Ljung test

data:  newX[, i]
X-squared = 2874.7, df = 10, p-value < 2.2e-16


$UNH

    Box-Ljung test

data:  newX[, i]
X-squared = 1408, df = 10, p-value < 2.2e-16


$UTX

    Box-Ljung test

data:  newX[, i]
X-squared = 1464.3, df = 10, p-value < 2.2e-16


$VZ

    Box-Ljung test

data:  newX[, i]
X-squared = 1916, df = 10, p-value < 2.2e-16
```

```
$WMT

    Box-Ljung test

data:  newX[, i]
X-squared = 681.11, df = 10, p-value < 2.2e-16


$XOM

    Box-Ljung test

data:  newX[, i]
X-squared = 1913.9, df = 10, p-value < 2.2e-16
```

Applying Ljung-Box tests to longer-interval returns

Task: Determine what happens What when you apply the Box-Ljung test to the monthly returns rather than the daily returns? Does the amount of serial dependence in the data appear to increase or decrease

Result: The amount of serial dependence appears to decrease when we move from daily to monthly returns.

Hide

```
# Create monthly log-returns from djx
djx_m <- apply.monthly(djx_1_xts, FUN = sum)
# Apply Ljung-Box tests to raw and absolute values of djx_m
Box.test(djx_m, lag = 10, type = "Ljung")
```

```
    Box-Ljung test

data:  djx_m
X-squared = 22.014, df = 10, p-value = 0.01503
```

Hide

```
Box.test(abs(djx_m), lag = 10, type = "Ljung")
```

```
    Box-Ljung test

data:  abs(djx_m)
X-squared = 12.874, df = 10, p-value = 0.2308
```

Hide

```
# Create monthly log-returns from djall
djall_m <- apply.monthly(djall_xts, FUN = colSums)
# Apply Ljung-Box tests to raw and absolute values of djall_m
apply(djall_m, 2, Box.test, lag = 10, type = "Ljung")
```

```
$AAPL

    Box-Ljung test

data:  newX[, i]
X-squared = 8.8986, df = 10, p-value = 0.5418


$AXP

    Box-Ljung test

data:  newX[, i]
X-squared = 24.309, df = 10, p-value = 0.006821


$BA

    Box-Ljung test

data:  newX[, i]
X-squared = 15.07, df = 10, p-value = 0.1295


$CAT

    Box-Ljung test

data:  newX[, i]
X-squared = 24.798, df = 10, p-value = 0.005741


$CSCO

    Box-Ljung test

data:  newX[, i]
X-squared = 11.516, df = 10, p-value = 0.3187


$CVX

    Box-Ljung test

data:  newX[, i]
X-squared = 8.2543, df = 10, p-value = 0.604


$DD

    Box-Ljung test

data:  newX[, i]
```

```
X-squared = 16.337, df = 10, p-value = 0.09038


$DIS

    Box-Ljung test

data:  newX[, i]
X-squared = 25.76, df = 10, p-value = 0.004077


$GE

    Box-Ljung test

data:  newX[, i]
X-squared = 14.035, df = 10, p-value = 0.1714


$GS

    Box-Ljung test

data:  newX[, i]
X-squared = 8.23, df = 10, p-value = 0.6064


$HD

    Box-Ljung test

data:  newX[, i]
X-squared = 15.347, df = 10, p-value = 0.1199


$IBM

    Box-Ljung test

data:  newX[, i]
X-squared = 7.6073, df = 10, p-value = 0.6671


$INTC

    Box-Ljung test

data:  newX[, i]
X-squared = 7.766, df = 10, p-value = 0.6517


$JNJ

    Box-Ljung test
```

```
data:  newX[, i]
X-squared = 12.506, df = 10, p-value = 0.2526


$JPM

    Box-Ljung test

data:  newX[, i]
X-squared = 8.1311, df = 10, p-value = 0.616


$KO

    Box-Ljung test

data:  newX[, i]
X-squared = 23.427, df = 10, p-value = 0.009276


$MCD

    Box-Ljung test

data:  newX[, i]
X-squared = 16.032, df = 10, p-value = 0.09873


$MMM

    Box-Ljung test

data:  newX[, i]
X-squared = 10.903, df = 10, p-value = 0.3651


$MRK

    Box-Ljung test

data:  newX[, i]
X-squared = 9.8863, df = 10, p-value = 0.4505


$MSFT

    Box-Ljung test

data:  newX[, i]
X-squared = 6.1313, df = 10, p-value = 0.8041


$NKE
```

```
    Box-Ljung test

data:  newX[, i]
X-squared = 6.2576, df = 10, p-value = 0.7932


$PFE

    Box-Ljung test

data:  newX[, i]
X-squared = 11.553, df = 10, p-value = 0.3161


$PG

    Box-Ljung test

data:  newX[, i]
X-squared = 6.401, df = 10, p-value = 0.7805


$TRV

    Box-Ljung test

data:  newX[, i]
X-squared = 11.746, df = 10, p-value = 0.3024


$UNH

    Box-Ljung test

data:  newX[, i]
X-squared = 19.733, df = 10, p-value = 0.03188


$UTX

    Box-Ljung test

data:  newX[, i]
X-squared = 20.668, df = 10, p-value = 0.02353


$VZ

    Box-Ljung test

data:  newX[, i]
X-squared = 16.024, df = 10, p-value = 0.09896
```

```
$WMT

    Box-Ljung test

data:  newX[, i]
X-squared = 5.8276, df = 10, p-value = 0.8295


$XOM

    Box-Ljung test

data:  newX[, i]
X-squared = 8.3251, df = 10, p-value = 0.5971
```

Hide

```
apply(abs(djall_m), 2, Box.test, lag = 10, type = "Ljung")
```

```
$AAPL

    Box-Ljung test

data:  newX[, i]
X-squared = 25.402, df = 10, p-value = 0.004633


$AXP

    Box-Ljung test

data:  newX[, i]
X-squared = 63.498, df = 10, p-value = 7.847e-10


$BA

    Box-Ljung test

data:  newX[, i]
X-squared = 30.372, df = 10, p-value = 0.0007445


$CAT

    Box-Ljung test

data:  newX[, i]
X-squared = 40.463, df = 10, p-value = 1.404e-05


$CSCO

    Box-Ljung test

data:  newX[, i]
X-squared = 14.818, df = 10, p-value = 0.1388


$CVX

    Box-Ljung test

data:  newX[, i]
X-squared = 11.761, df = 10, p-value = 0.3013


$DD

    Box-Ljung test

data:  newX[, i]
```

```
X-squared = 21.963, df = 10, p-value = 0.0153


$DIS

    Box-Ljung test

data:  newX[, i]
X-squared = 19.134, df = 10, p-value = 0.0386


$GE

    Box-Ljung test

data:  newX[, i]
X-squared = 94.468, df = 10, p-value = 6.661e-16


$GS

    Box-Ljung test

data:  newX[, i]
X-squared = 27.767, df = 10, p-value = 0.001967


$HD

    Box-Ljung test

data:  newX[, i]
X-squared = 10.251, df = 10, p-value = 0.4188


$IBM

    Box-Ljung test

data:  newX[, i]
X-squared = 7.8383, df = 10, p-value = 0.6446


$INTC

    Box-Ljung test

data:  newX[, i]
X-squared = 4.9157, df = 10, p-value = 0.8967


$JNJ

    Box-Ljung test
```

```
data:  newX[, i]
X-squared = 11.859, df = 10, p-value = 0.2946


$JPM

    Box-Ljung test

data:  newX[, i]
X-squared = 50.243, df = 10, p-value = 2.408e-07


$KO

    Box-Ljung test

data:  newX[, i]
X-squared = 13.112, df = 10, p-value = 0.2175


$MCD

    Box-Ljung test

data:  newX[, i]
X-squared = 8.2577, df = 10, p-value = 0.6037


$MMM

    Box-Ljung test

data:  newX[, i]
X-squared = 9.2315, df = 10, p-value = 0.5103


$MRK

    Box-Ljung test

data:  newX[, i]
X-squared = 22.123, df = 10, p-value = 0.01449


$MSFT

    Box-Ljung test

data:  newX[, i]
X-squared = 4.829, df = 10, p-value = 0.9023


$NKE
```

```
    Box-Ljung test

data:  newX[, i]
X-squared = 6.9332, df = 10, p-value = 0.7317


$PFE

    Box-Ljung test

data:  newX[, i]
X-squared = 18.012, df = 10, p-value = 0.05477


$PG

    Box-Ljung test

data:  newX[, i]
X-squared = 19.049, df = 10, p-value = 0.03964


$TRV

    Box-Ljung test

data:  newX[, i]
X-squared = 6.0938, df = 10, p-value = 0.8073


$UNH

    Box-Ljung test

data:  newX[, i]
X-squared = 58.742, df = 10, p-value = 6.264e-09


$UTX

    Box-Ljung test

data:  newX[, i]
X-squared = 16.733, df = 10, p-value = 0.08049


$VZ

    Box-Ljung test

data:  newX[, i]
X-squared = 10.169, df = 10, p-value = 0.4258
```

```
$WMT

    Box-Ljung test

data:  newX[, i]
X-squared = 5.4399, df = 10, p-value = 0.8599


$XOM

    Box-Ljung test

data:  newX[, i]
X-squared = 10.476, df = 10, p-value = 0.3998
```

Looking at the extremes in volatile return series

Extreme values in volatile time series

When you take a long series of iid (independent & identically distributed) data, such as several thousand observations, and select a small subset of the most extreme observations, like less than 100, then these extremes appear at random and the spaces or gaps between the extremes follow a distribution that is very close to exponential. When we carry out the same exercise for a volatile financial log-return series then the extremes appear in clusters during periods of high volatility.

Task: Investigate the irregular time series djx_extremes which contains the 100 most extreme negative log-returns of the Dow Jones index between 1985 and 2015. Compare it with iid_extremes which contains the 100 most extreme values in an iid series of the same length as djx_extremes. To do this, use the object exp_quantiles, which contains 100 theoretical quantiles of the standard exponential distribution.

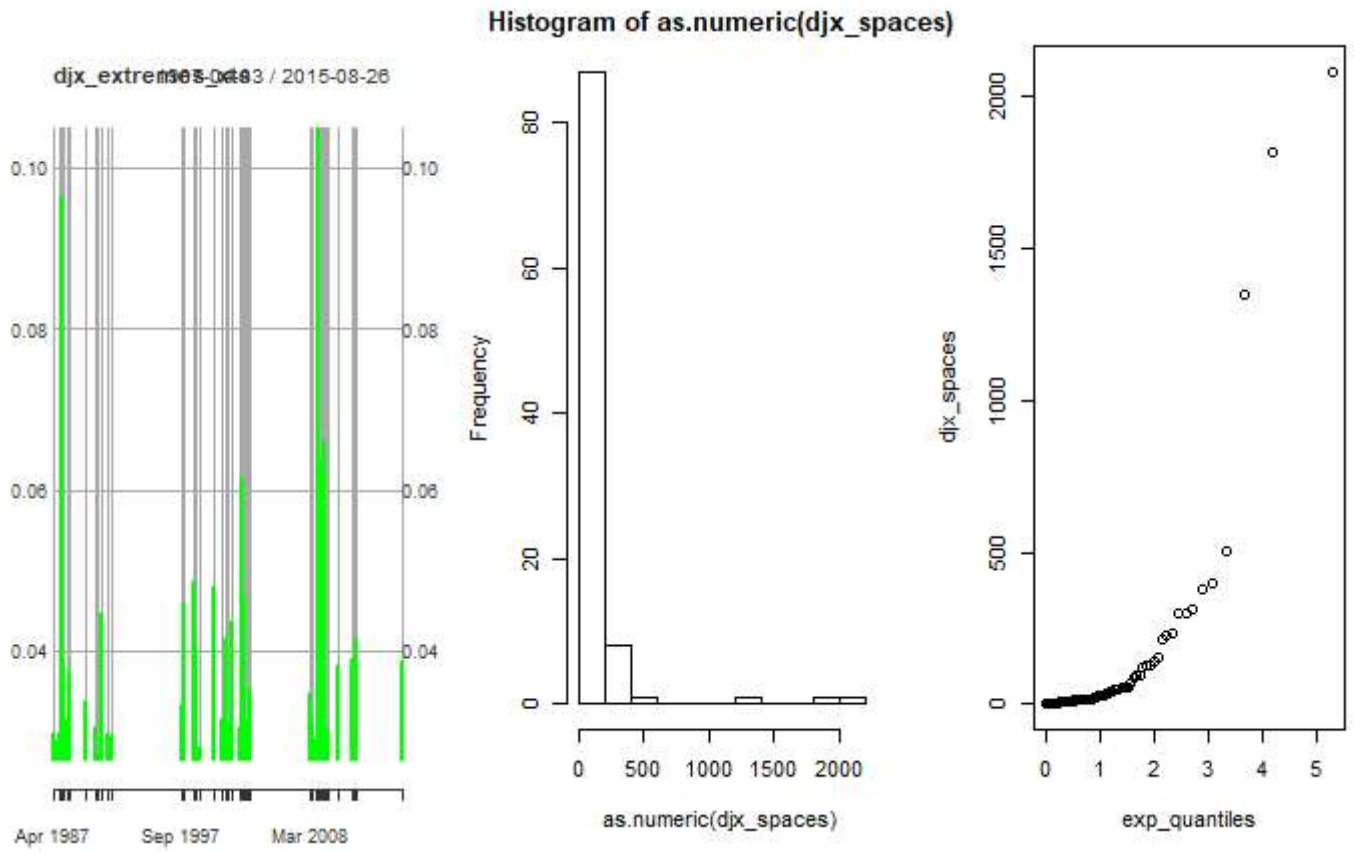
Result: Note how strongly clustered the extreme returns are compared with the extremes in the iid series.

Hide

```
exp_quantiles <- c(0.005012542, 0.015113638,    0.025317808,    0.035627178,    0.046043939,
0.056570351,
0.06720875, 0.077961541,    0.088831214,    0.099820335,    0.110931561,    0.122167634,
0.133531393,    0.145025772,    0.15665381, 0.168418652,    0.180323554,    0.192371893,
0.204567166,    0.216913002,    0.229413164,    0.242071561,    0.25489225, 0.267879445,
0.28103753, 0.294371061,    0.30788478, 0.321583624,    0.335472736,    0.349557476,
0.363843433,    0.378336441,    0.393042588,    0.407968238,    0.423120043,    0.438504962,
0.45413028, 0.470003629,    0.486133011,    0.502526821,    0.519193873,    0.536143432,
0.553385238,    0.570929548,    0.588787165,    0.606969484,    0.625488532,    0.644357016,
0.663588378,    0.68319685, 0.703197516,    0.723606388,    0.744440475,    0.765717873,
0.78745786, 0.809680997,    0.832409248,    0.85566611, 0.879476759,    0.903868212,
0.928869514,    0.954511945,    0.980829253,    1.007857925,    1.03563749, 1.064210862,
1.093624747,    1.123930097,    1.15518264, 1.187443502,    1.220779923,    1.255266099,
1.290984181,    1.328025453,    1.366491734,    1.406497068,    1.448169765,    1.491654877,
1.537117251,    1.5847453,  1.63475572, 1.687399454,    1.742969305,    1.801809805,
1.864330162,    1.931021537,    2.002480501,    2.079441542,    2.162823151,    2.253794929,
2.353878387,    2.465104022,    2.590267165,    2.733368009,    2.900422094,    3.101092789,
3.352407217,    3.688879454,    4.199705078,    5.298317367)
djx_extremes <- read_csv("C:/Users/Y/Documents/1.R_scripts/ML/Risk_Modeling/Quantitative Risk Ma
nagement in R/djx_extremes.csv",
    col_types = cols(date = col_date(format = "%m/%d/%Y")))
djx_extremes_xts <- as.xts(djx_extremes[,-1], order.by = djx_extremes$date, "%d-%m-%Y")
iid_extremes <- read_csv("C:/Users/Y/Documents/1.R_scripts/ML/Risk_Modeling/Quantitative Risk Ma
nagement in R/iid_extremes.csv",
    col_types = cols(date = col_date(format = "%m/%d/%Y")))
iid_extremes_xts <- as.xts(iid_extremes[,-1], order.by = iid_extremes$date, "%d-%m-%Y")
# Partition plotting area into 3 pieces
par(mfrow = c(1, 3))
# Plot djx_extremes
plot(djx_extremes_xts, type = "h")
# Compute the spaces between the times of the extremes
djx_spaces <- diff(time(djx_extremes_xts))
# Make a histogram of these spaces
hist(as.numeric(djx_spaces))
```

Hide

```
# Make a Q-Q plot of djx_spaces against exp_quantiles
qqplot(exp_quantiles, djx_spaces)
```

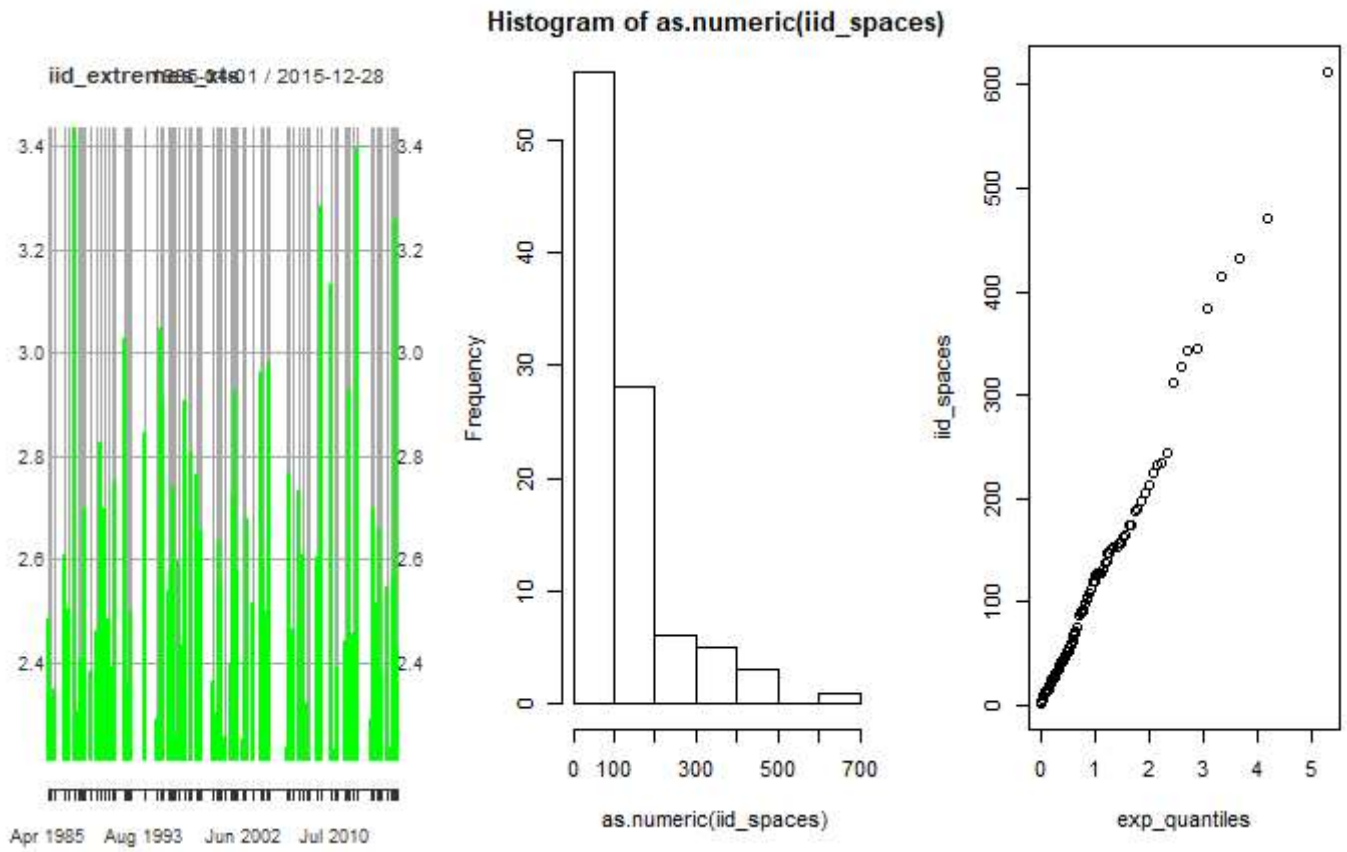


Hide

```
# Carry out the previous 4 steps for iid_extremes
plot(iid_extremes_xts, type = "h")
iid_spaces <- diff(time(iid_extremes_xts))
hist(as.numeric(iid_spaces))
```

Hide

```
qqplot(exp_quantiles, iid_spaces)
```



Cross correlations between risk-factor return series

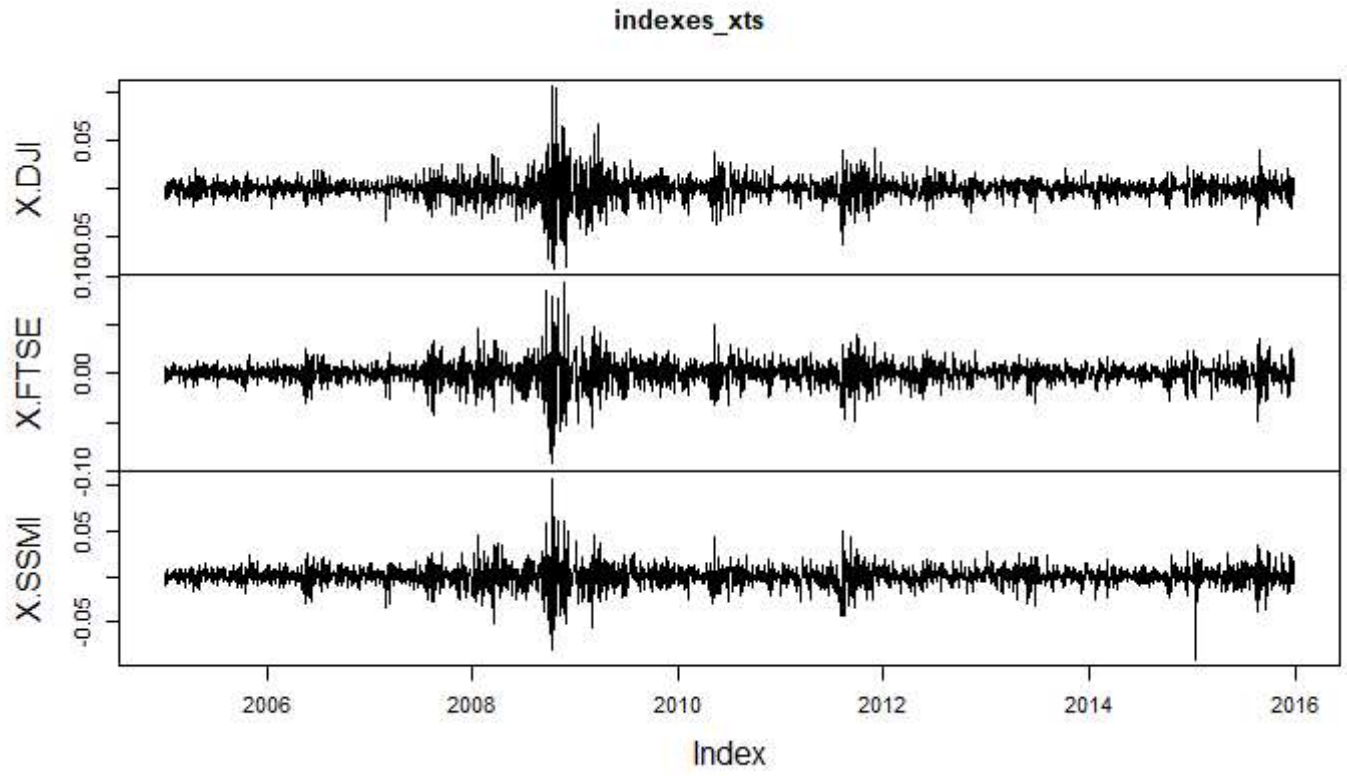
In the same way that there tends to be only weak serial correlation within series, there tends to be only weak cross correlation between series in different time periods. This picture changes dramatically when we look at the absolute values, which are often strongly correlated both within and between series.

Task: Investigate cross correlations between the daily log-returns of the Dow Jones, FTSE100 and SMI equity indexes.

Result: Evidence that the US market leads the European markets in the third of the four plots.

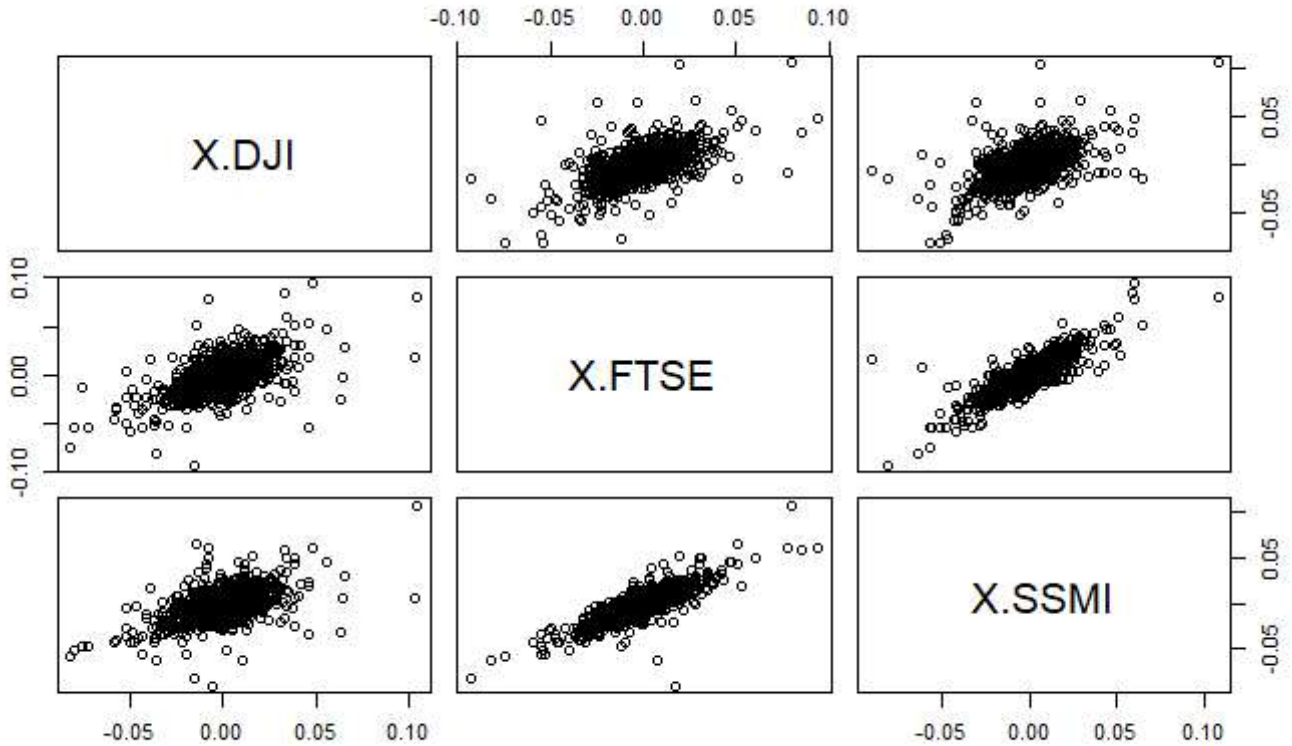
Hide

```
indexes <- read_csv("C:/Users/Y/Documents/1.R_scripts/ML/Risk_Modeling/Quantitative Risk Managem
ent in R/indexes.csv",
    col_types = cols(date = col_date(format = "%m/%d/%Y")))
indexes_xts <- as.xts(indexes[,-1], order.by = indexes$date, "%d-%m-%Y")
# Make a time series plot of indexes with plot.zoo and a pairwise scatterplot with pairs
plot.zoo(indexes_xts)
```



Hide

```
pairs(as.zoo(indexes_xts))
```



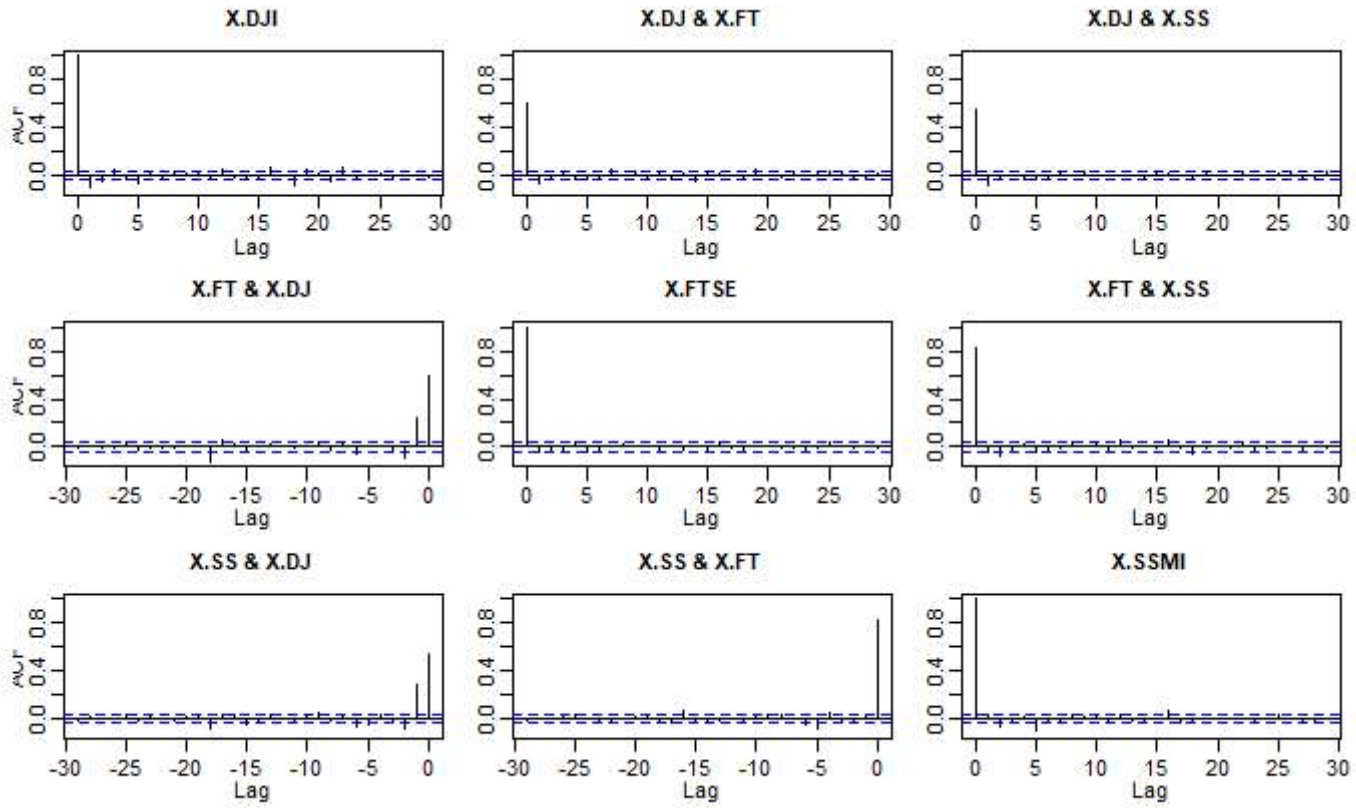
Hide

```
# Calculate the sample correlation matrix of indexes
cor(indexes_xts)
```

```
          X.DJI    X.FTSE    X.SSMI
X.DJI  1.0000000 0.5928777 0.5430227
X.FTSE 0.5928777 1.0000000 0.8274246
X.SSMI 0.5430227 0.8274246 1.0000000
```

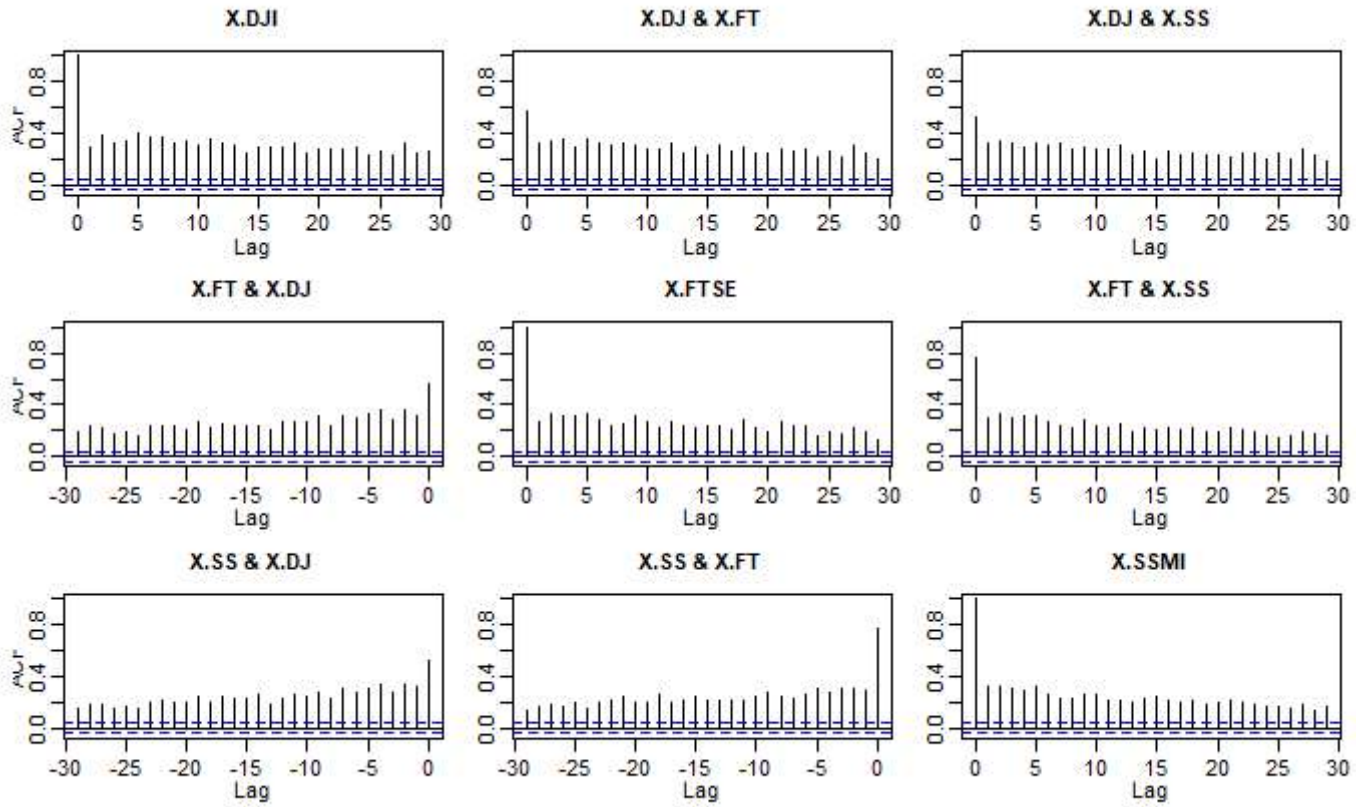
Hide

```
# Plot the sample acfs and cross-correlation functions for the returns in indexes
acf(indexes_xts)
```



Hide

```
# Plots the sample acfs and cross-correlations functions for the absolute values of indexes
acf(abs(indexes_xts))
```



The stylised facts of return series

The facts: .Return series are heavier-tailed than normal, or leptokurtic .The volatility of return series appears to vary over time .Return series show relatively little serial correlation .Series of absolute returns show profound serial correlation .Extreme returns appear in clusters .Returns aggregated over longer periods tend to become more normal and less serially dependent

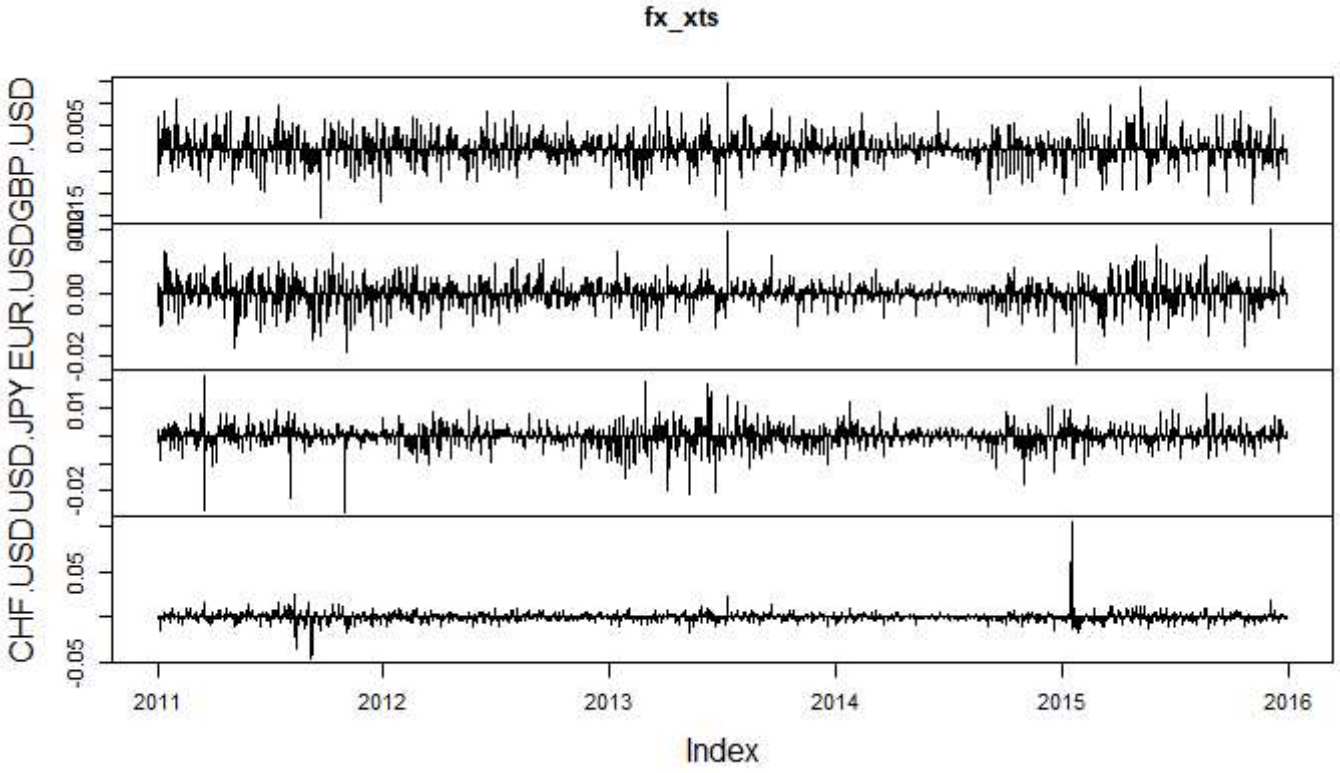
Volatility and correlation of FX returns

Task: Discover evidence of volatility and serial dependence in daily and weekly exchange-rate log-returns. The dataset fx contains daily log-returns for the “EUR_USD”, “GBP_USD”, “JPY_USD” and “CHF_USD” exchange rates while fx_w contains the corresponding weekly log-returns.

Result:There is strong evidence of serial dependence in the absolute values of the weekly log-returns.

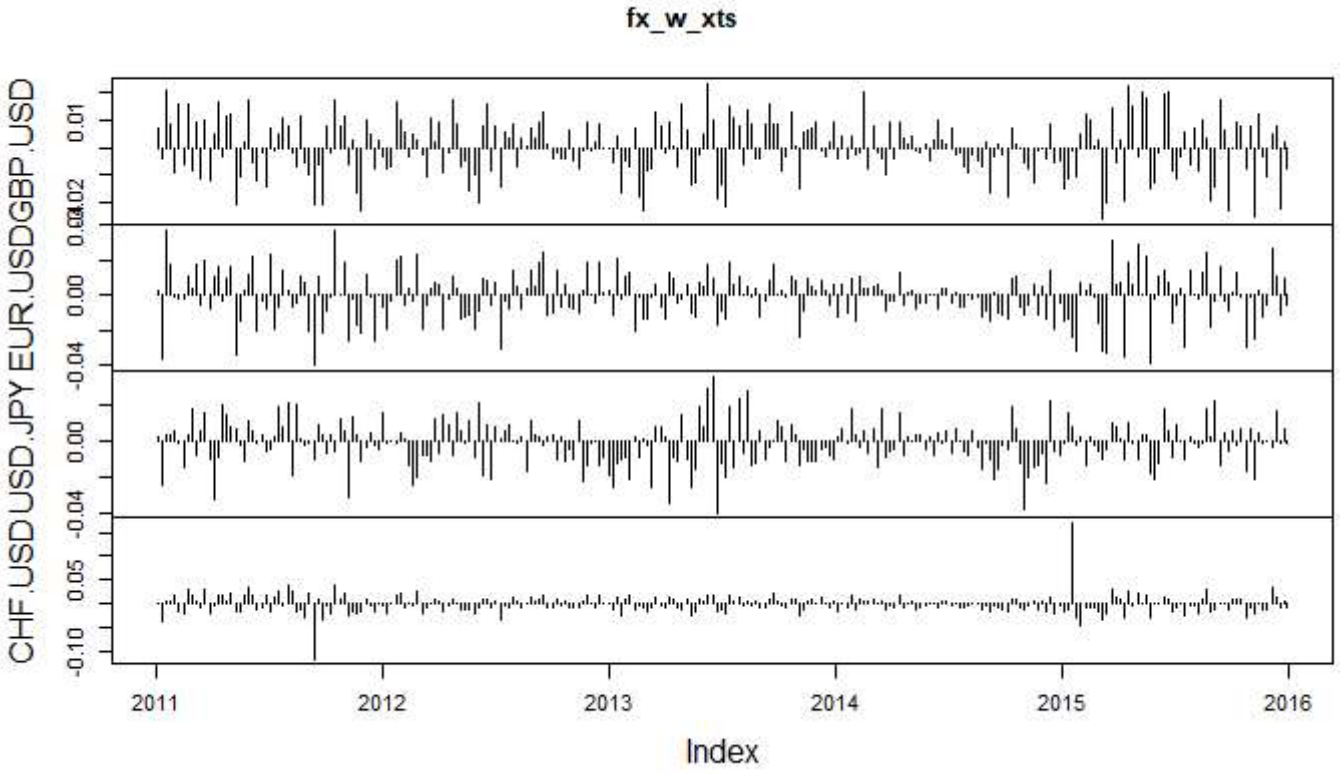
Hide

```
fx <- read_csv("C:/Users/Y/Documents/1.R_scripts/ML/Risk_Modeling/Quantitative Risk Management i
n R/fx.csv",
    col_types = cols(date = col_date(format = "%m/%d/%Y")))
fx_xts <- as.xts(fx[,-1], order.by = fx$date, "%d-%m-%Y")
fx_w <- read_csv("C:/Users/Y/Documents/1.R_scripts/ML/Risk_Modeling/Quantitative Risk Management
 in R/fx_w.csv",
    col_types = cols(date = col_date(format = "%m/%d/%Y")))
fx_w_xts <- as.xts(fx_w[,-1], order.by = fx_w$date, "%d-%m-%Y")
# Plot fx and fx_w
plot.zoo(fx_xts, type = "h")
```



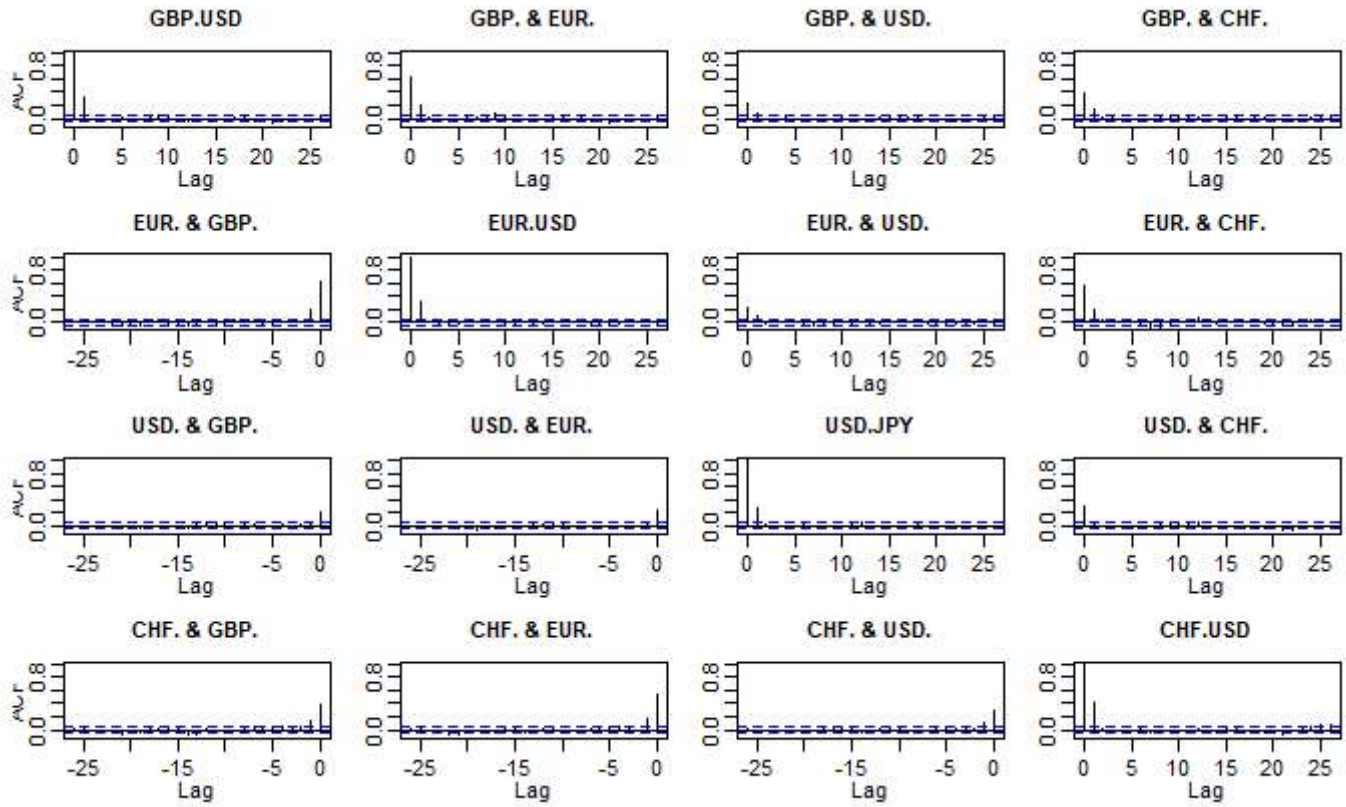
Hide

```
plot.zoo(fx_w_xts, type = "h")
```



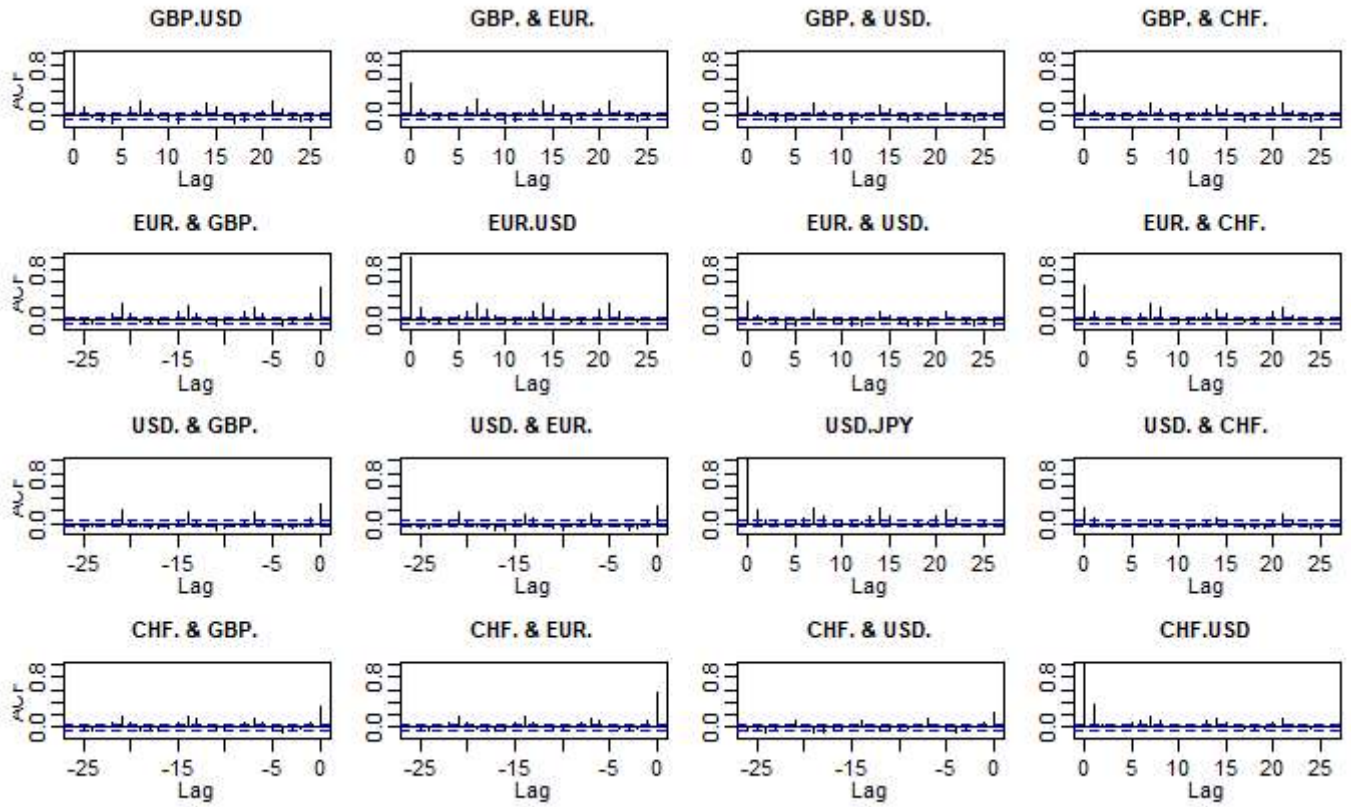
Hide

```
# Make acf plots of fx and the absolute values of fx
acf(fx_xts)
```



Hide

```
acf(abs(fx_xts))
```



Hide

```
# Apply the Ljung-Box test to the components of fx and their absolute values
apply(fx_xts, 2, Box.test, lag = 10, type = "Ljung")
```

```
$GBP.USD

    Box-Ljung test

data:  newX[, i]
X-squared = 206.44, df = 10, p-value < 2.2e-16


$EUR.USD

    Box-Ljung test

data:  newX[, i]
X-squared = 195.33, df = 10, p-value < 2.2e-16


$USD.JPY

    Box-Ljung test

data:  newX[, i]
X-squared = 141.48, df = 10, p-value < 2.2e-16


$CHF.USD

    Box-Ljung test

data:  newX[, i]
X-squared = 310.65, df = 10, p-value < 2.2e-16
```

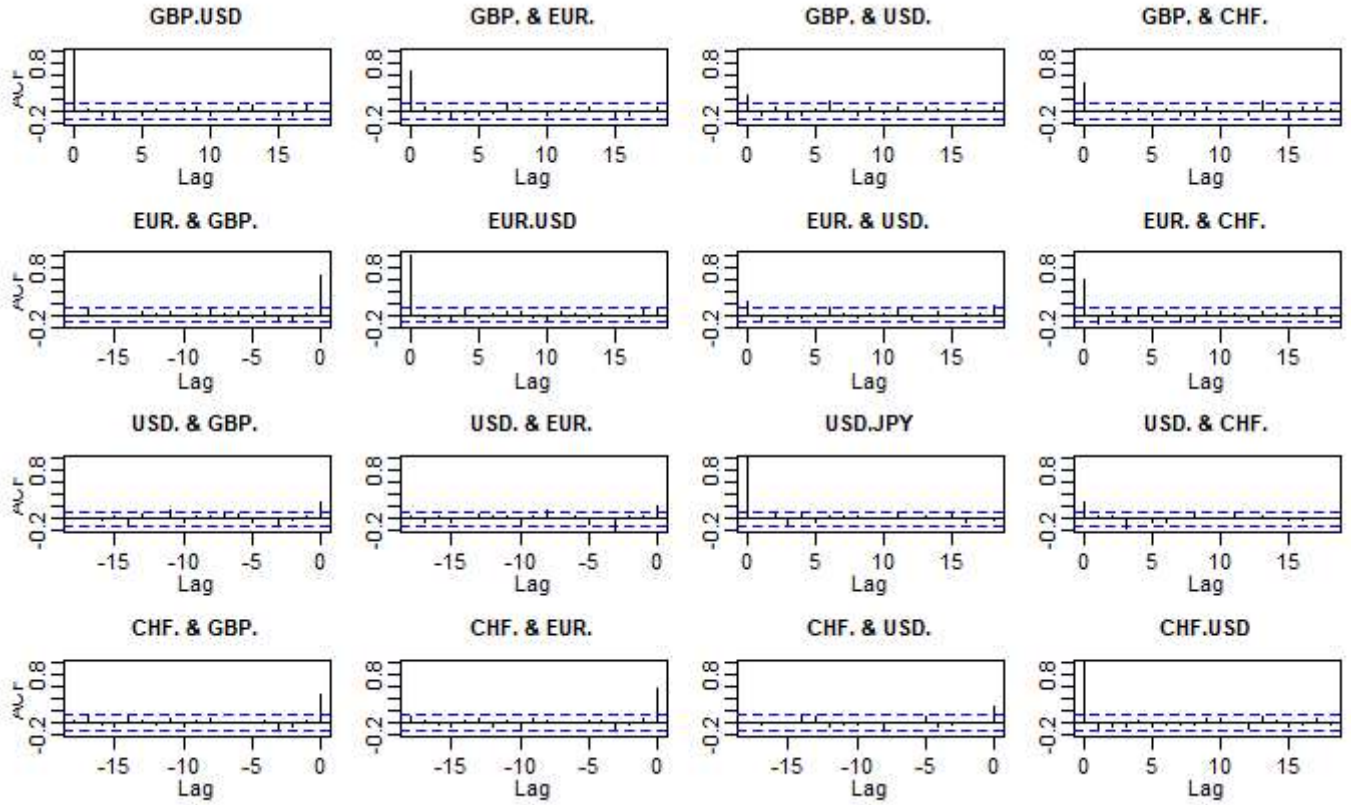
Hide

```
apply(abs(fx_xts), 2, Box.test, lag = 10, type = "Ljung")
```

```
$GBP.USD

    Box-Ljung test

data:  newX[, i]
X-squared = 235.96, df = 10, p-value < 2.2e-16


$EUR.USD

    Box-Ljung test

data:  newX[, i]
X-squared = 306.92, df = 10, p-value < 2.2e-16


$USD.JPY

    Box-Ljung test

data:  newX[, i]
X-squared = 264.36, df = 10, p-value < 2.2e-16


$CHF.USD

    Box-Ljung test

data:  newX[, i]
X-squared = 377.12, df = 10, p-value < 2.2e-16
```

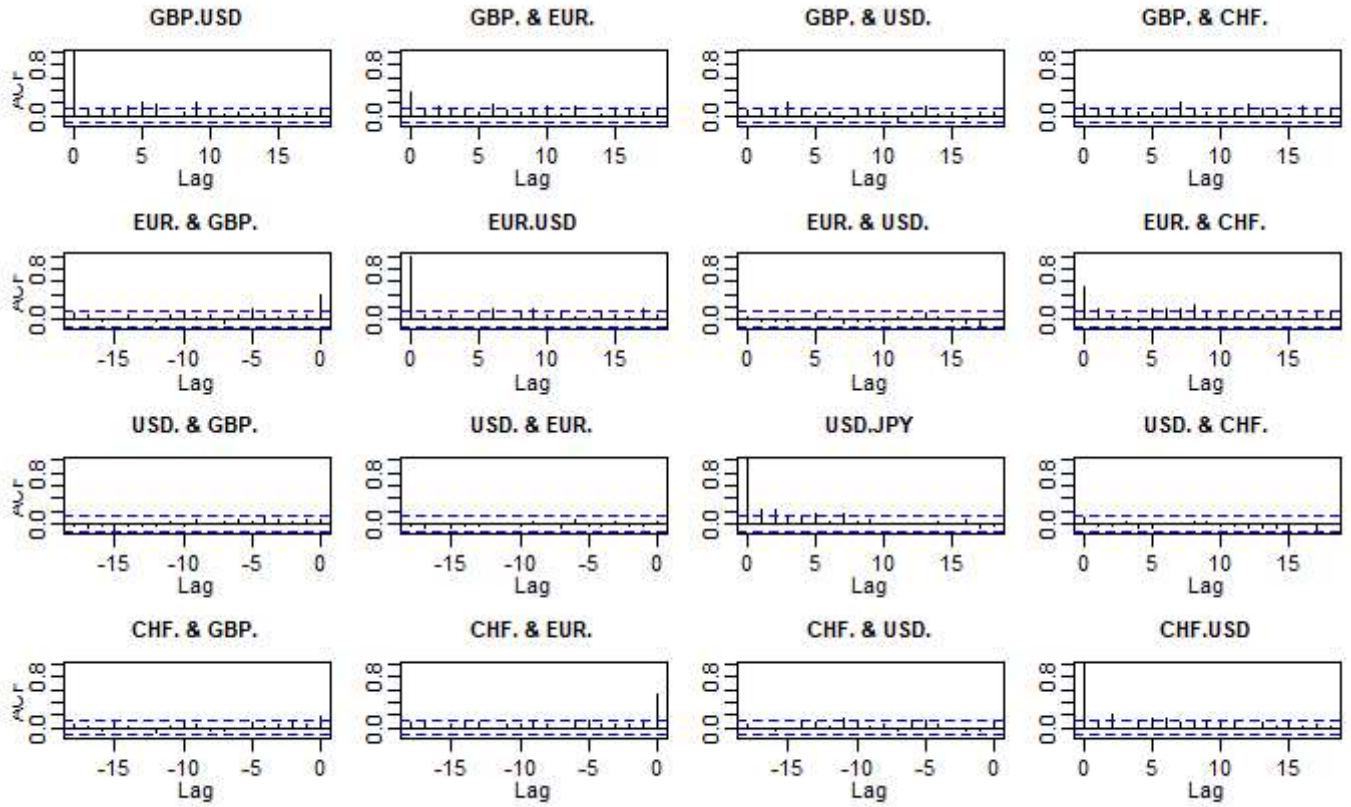
Hide

```
# Make acf plots of fx_w and the absolute values of fx_w
acf(fx_w_xts)
```



Hide

```
acf(abs(fx_w_xts))
```



Hide

```
# Apply the Ljung-Box test to the components of fx_w and their absolute values
apply(fx_w_xts, 2, Box.test, lag = 10, type = "Ljung")
```

```
$GBP.USD

    Box-Ljung test

data:  newX[, i]
X-squared = 8.888, df = 10, p-value = 0.5428


$EUR.USD

    Box-Ljung test

data:  newX[, i]
X-squared = 10.359, df = 10, p-value = 0.4095


$USD.JPY

    Box-Ljung test

data:  newX[, i]
X-squared = 7.7924, df = 10, p-value = 0.6491


$CHF.USD

    Box-Ljung test

data:  newX[, i]
X-squared = 10.977, df = 10, p-value = 0.3593
```

Hide

```
apply(abs(fx_w_xts), 2, Box.test, lag = 10, type = "Ljung")
```

```
$GBP.USD

    Box-Ljung test

data:  newX[, i]
X-squared = 47.086, df = 10, p-value = 9.108e-07


$EUR.USD

    Box-Ljung test

data:  newX[, i]
X-squared = 29.121, df = 10, p-value = 0.001191


$USD.JPY

    Box-Ljung test

data:  newX[, i]
X-squared = 46.153, df = 10, p-value = 1.345e-06


$CHF.USD

    Box-Ljung test

data:  newX[, i]
X-squared = 34.438, df = 10, p-value = 0.0001557
```

Volatility and correlation of interest-rate data

When volatility is present in a time series we can predict the magnitude of future returns.

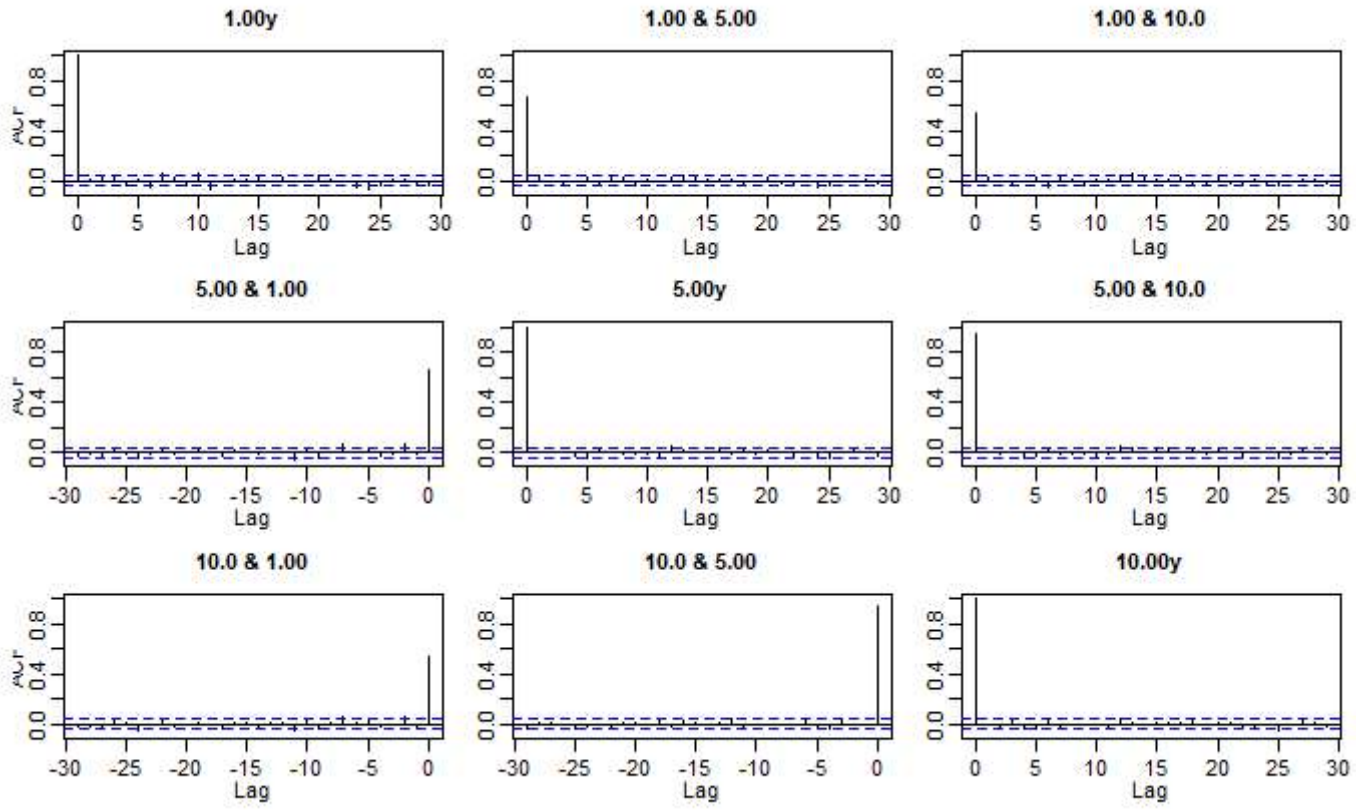
Task: explore whether volatility and serial dependence is also a feature of daily and monthly interest-rate log-returns.

Task: Explore whether volatility and serial dependence is also a feature of daily and monthly interest-rate log-returns.

Results: The monthly log-returns for the 5 and 10 year yields don’t appear to show much serial dependence.

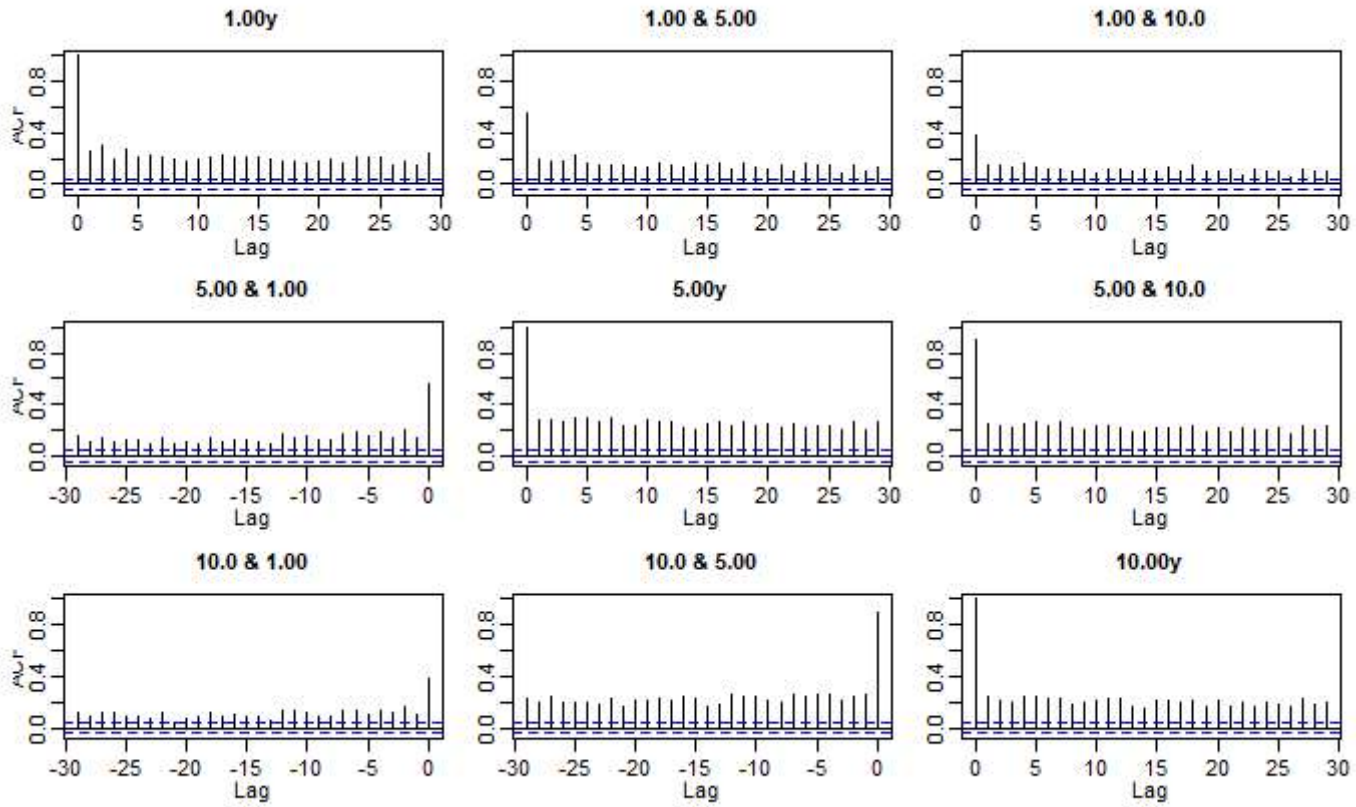
Hide

```
zcb_m <- read_csv("C:/Users/Y/Documents/1.R_scripts/ML/Risk_Modeling/Quantitative Risk Managemen
t in R/zcb_m.csv",
    col_types = cols(date = col_date(format = "%m/%d/%Y")))
zcb_m_xts <- as.xts(zcb_m[,-1], order.by = zcb_m$date, "%d-%m-%Y")
zcb_x <- read_csv("C:/Users/Y/Documents/1.R_scripts/ML/Risk_Modeling/Quantitative Risk Managemen
t in R/zcb_x.csv",
    col_types = cols(date = col_date(format = "%m/%d/%Y")))
zcb_x_xts <- as.xts(zcb_x[,-1], order.by = zcb_x$date, "%d-%m-%Y")
# Make acf plots of zcb_x and the absolute values of zcb_x
acf(zcb_x_xts)
```



Hide

```
acf(abs(zcb_x_xts))
```



Hide

```
# Apply the Ljung-Box test to the components of zcb_x and their absolute values
apply(zcb_x_xts, 2, Box.test, lag = 10, type = "Ljung")
```

```
$`1.00y`

    Box-Ljung test

data:  newX[, i]
X-squared = 34.513, df = 10, p-value = 0.0001512


$`5.00y`

    Box-Ljung test

data:  newX[, i]
X-squared = 11.894, df = 10, p-value = 0.2922


$`10.00y`

    Box-Ljung test

data:  newX[, i]
X-squared = 17.073, df = 10, p-value = 0.07277
```

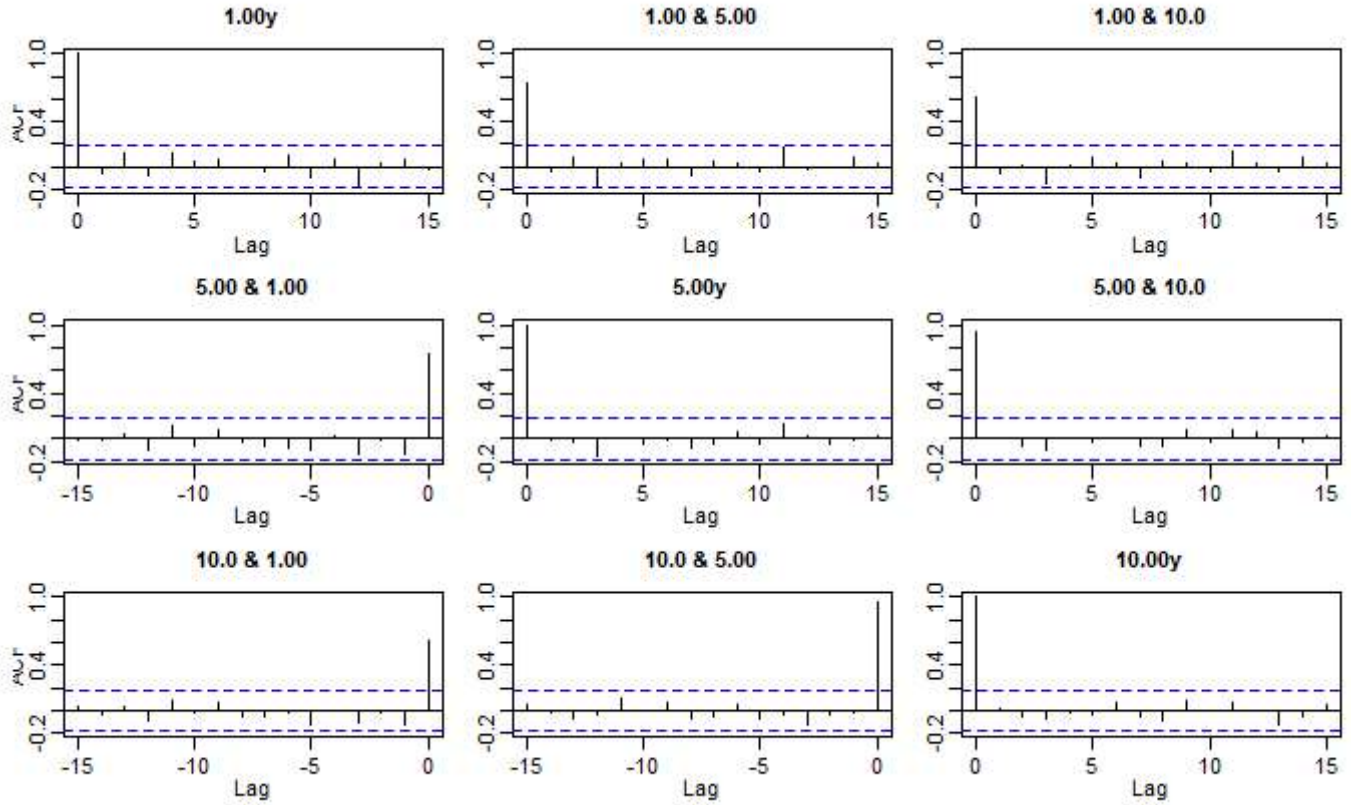
Hide

```
apply(abs(zcb_x_xts), 2, Box.test, lag = 10, type = "Ljung")
```

```
$`1.00y`

    Box-Ljung test

data:  newX[, i]
X-squared = 1247.1, df = 10, p-value < 2.2e-16


$`5.00y`

    Box-Ljung test

data:  newX[, i]
X-squared = 1827.2, df = 10, p-value < 2.2e-16


$`10.00y`

    Box-Ljung test

data:  newX[, i]
X-squared = 1147.6, df = 10, p-value < 2.2e-16
```

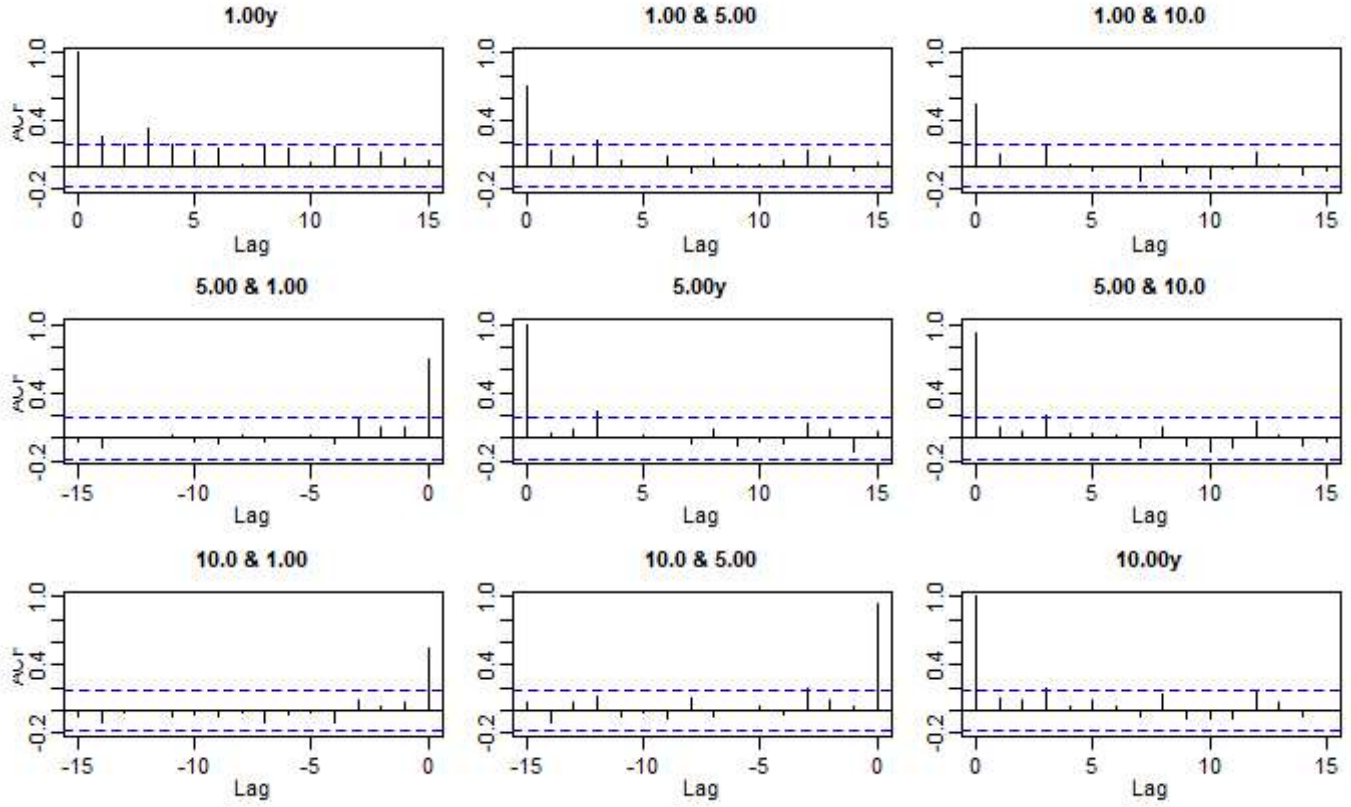
Hide

```
# Make acf plots of zcbx_m and the absolute values of zcbx_m
acf(zcbx_m_xts)
```



Hide

```
acf(abs(zcbx_m_xts))
```



Hide

```
# Apply the Ljung-Box test to the components of zcbx_m and their absolute values
apply(zcbx_m_xts, 2, Box.test, lag = 10, type = "Ljung")
```

```
$`1.00y`

    Box-Ljung test

data:  newX[, i]
X-squared = 8.3951, df = 10, p-value = 0.5903


$`5.00y`

    Box-Ljung test

data:  newX[, i]
X-squared = 5.4826, df = 10, p-value = 0.8567


$`10.00y`

    Box-Ljung test

data:  newX[, i]
X-squared = 4.2486, df = 10, p-value = 0.9354
```

Hide

```
apply(abs(zcbx_m_xts), 2, Box.test, lag = 10, type = "Ljung")
```

```
$`1.00y`

    Box-Ljung test

data:  newX[, i]
X-squared = 41.992, df = 10, p-value = 7.524e-06


$`5.00y`

    Box-Ljung test

data:  newX[, i]
X-squared = 8.9816, df = 10, p-value = 0.5338


$`10.00y`

    Box-Ljung test

data:  newX[, i]
X-squared = 11.917, df = 10, p-value = 0.2906
```

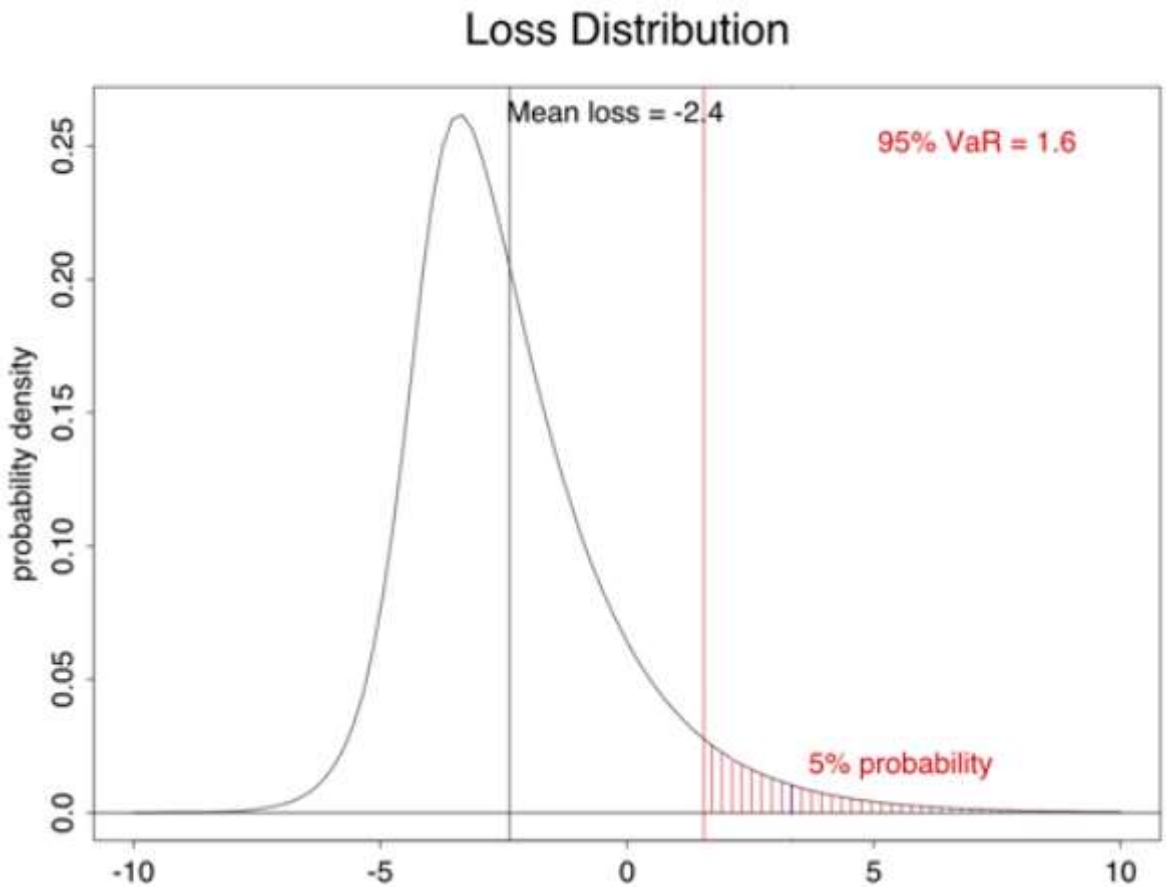
Value-at-risk and expected shortfall

Value at Risk: Statistic that describes the tail of the loss distribution and the potential for large losses.

The banks consider the distribution of losses they could incur over two trading weeks if the trading positions were held fixed. A risk measure is really just a statistic that describes the loss distribution. You could take the mean or std dev of the distribution. What is most common in QRM is to take a statistic that describes the tail of the loss distribution and the potential for large losses. In particular, it is common to take a quantile of the distribution in the tail, i.e. 95, 99, 99.5% percentile.

Alpha var is the alpha quantile of the portfolio loss distribution for a specified time period.

The var is used as a guide to tell how much a bank or other institution should have so that it can remain solvent in the face of large trading losses.



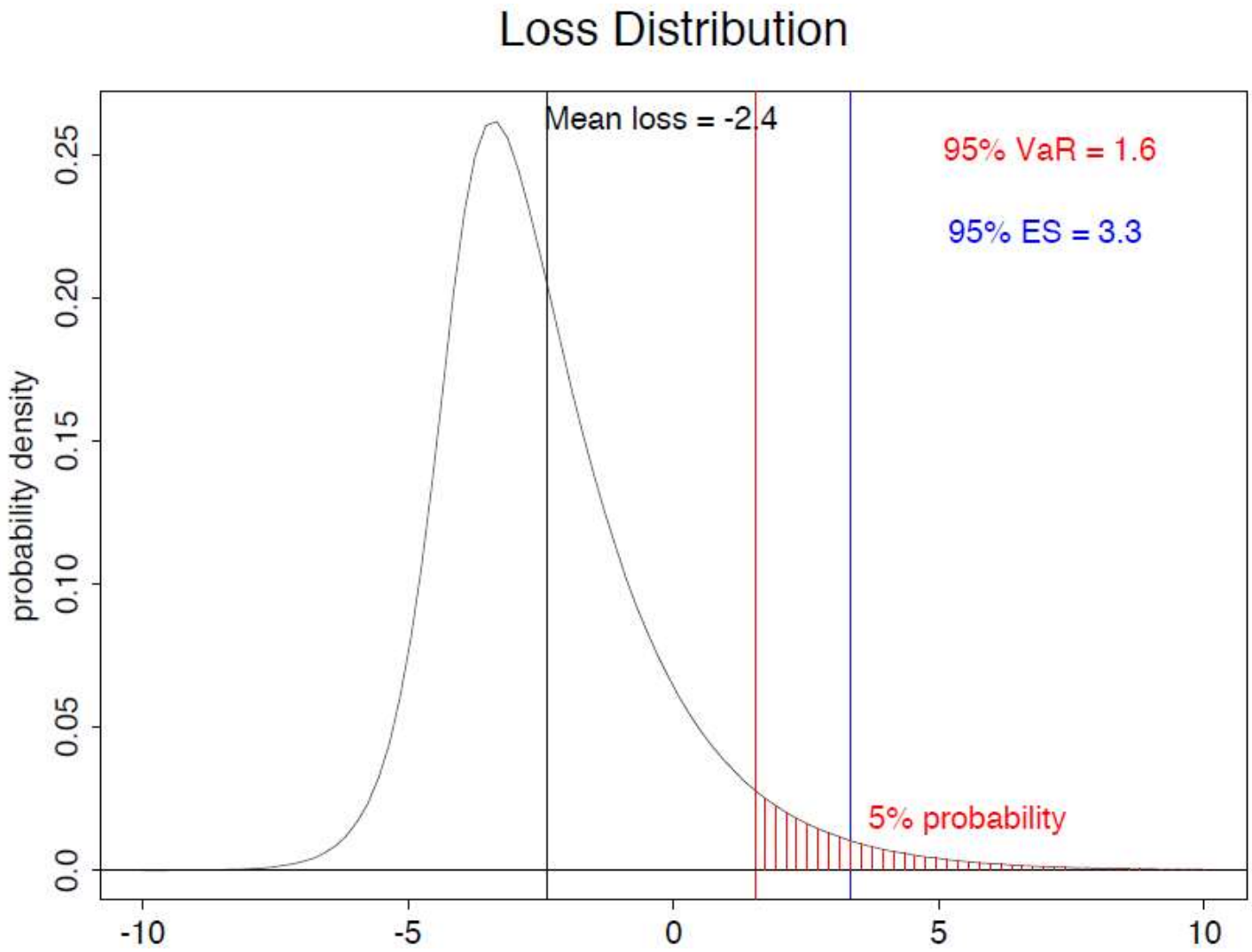
95% Var

Expected shortfall (ES) . Increasingly important in banking regulation .

Tail VaR (TVaR), conditional VaR (CVaR) or expected shortfall (ES)

a-ES is expected loss given that loss exceeds a-VaR

Can be improved: . Parametric tail models, heavy-tailed distributions, extreme value theory



95% Expected Shortfall

Computing VaR and ES for normal distribution

Task: compute and display VaR and ES for a normal distribution N(??,???2)N(??,???2) with mean ???? and standard deviation ????.

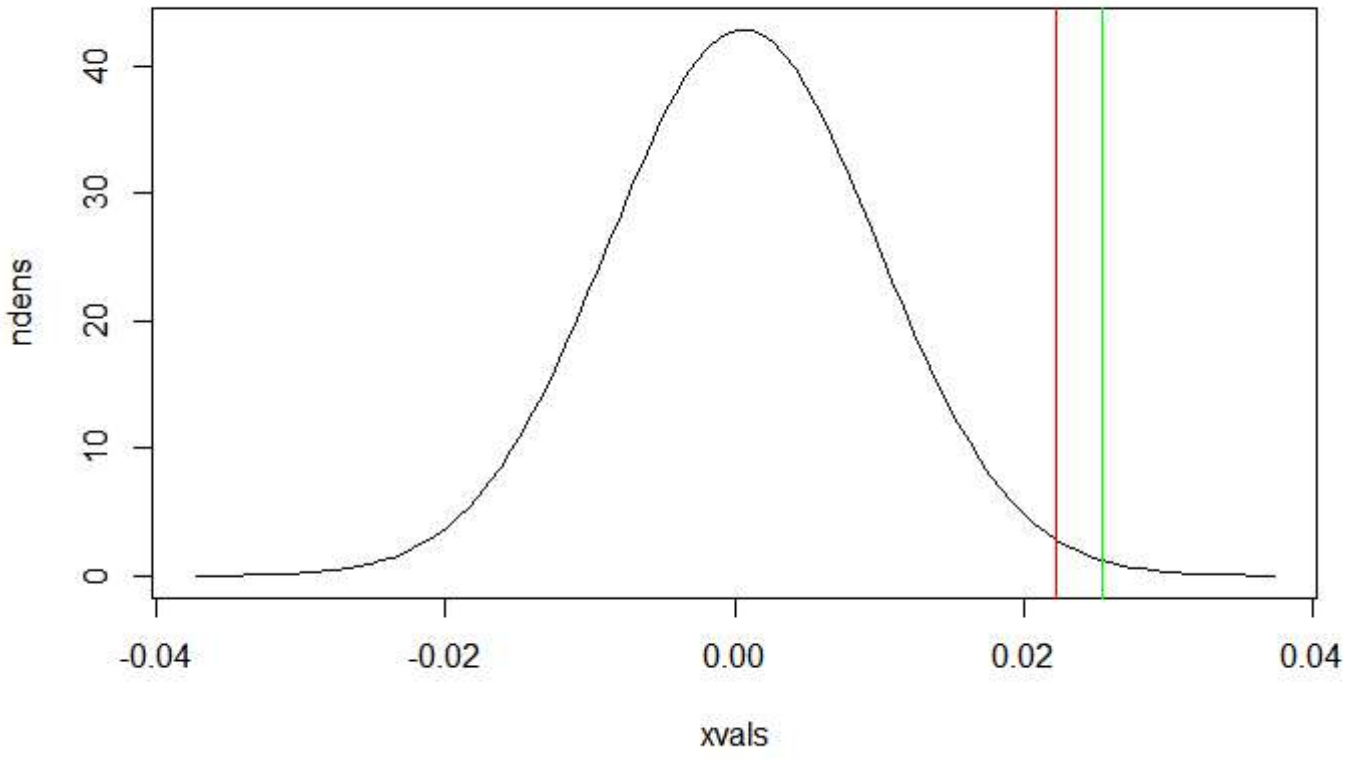
Result: ES99 is only 14.7% bigger than VaR99. For heavy-tailed distributions, the difference can be much greater.

Hide

```
# Make a sequence of 100 x-values going from -4*sigma to 4*sigma
xvals <- seq(from = -4*sigma, to = 4*sigma, length.out = 100)
# Compute the density of a N(mu, sigma^2) distribution at xvals
ndens <- dnorm(xvals, mean = mu, sd = sigma)
# Plot ndens against xvals
plot(xvals, ndens, type = "l")
# Compute the 99% VaR and 99% ES of a N(mu, sigma^2) distribution
VaR99 <- qnorm(.99, mean=mu, sd=sigma)
ES99 <- ESnorm(.99, mu=mu, sd=sigma)
# Draw vertical lines at VaR99 and ES99 in red and green
abline(v = VaR99, col = "red")
```

Hide

```
abline(v = ES99, col = "green")
```



International equity portfolio

Historical Simulation

Historical simulation . Resample historical risk-factor returns and examine their effect on current portfolio . Loss operator shows effect of different risk-factor returns on the portfolio

To estimate the portfolio's loss distribution you are going to use historical simulation. Based on the non-parametric idea of resampling past data. The historical risk factors are resampled and applied to the current portfolio in order to construct the series of losses and profits that would result if those risk factor changes happened again. In order to implement historical simulation, it is necessary to write a function called a loss operator.

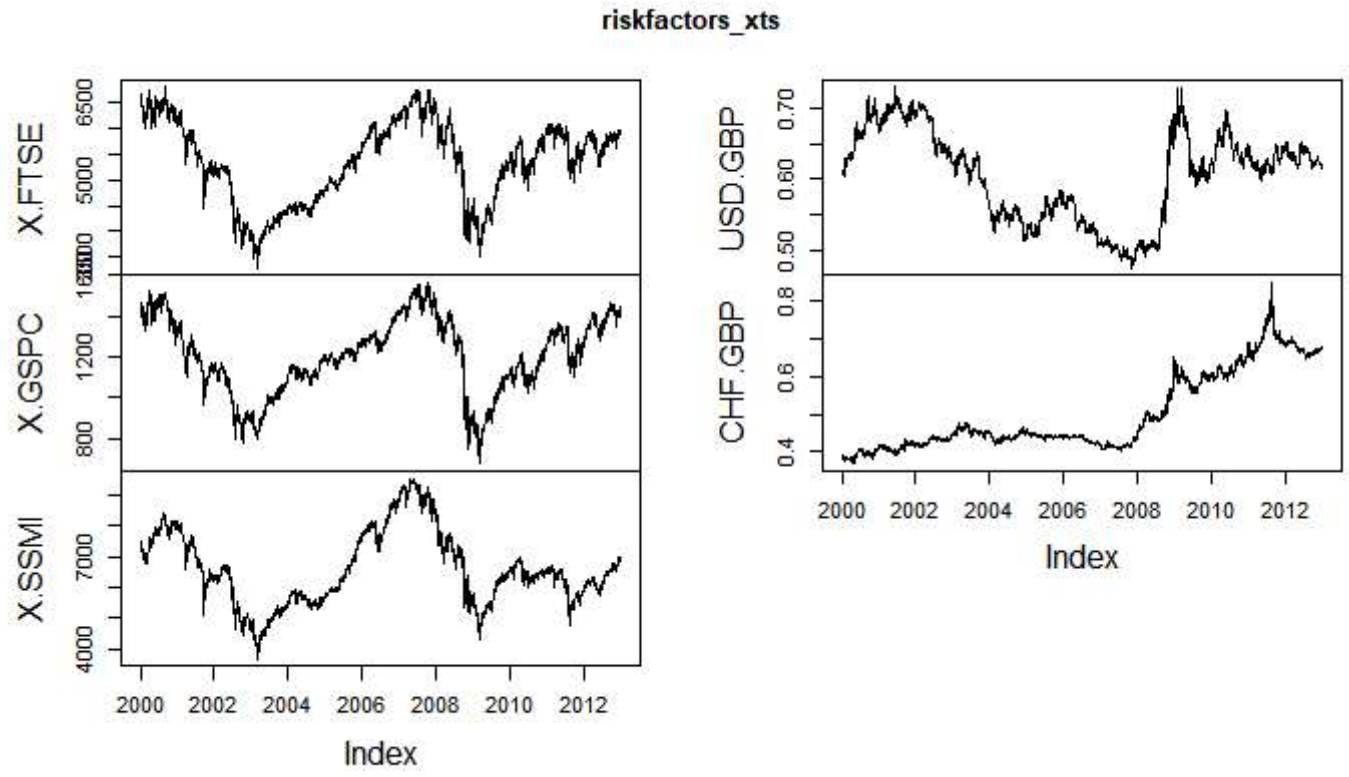
Examining risk factors for international equity portfolio

Task: Examine five risk factors to determine if they are normal and if they show serial and cross dependencies.

Result:All five risk factors are clearly non-normal and show strong serial and cross dependencies.

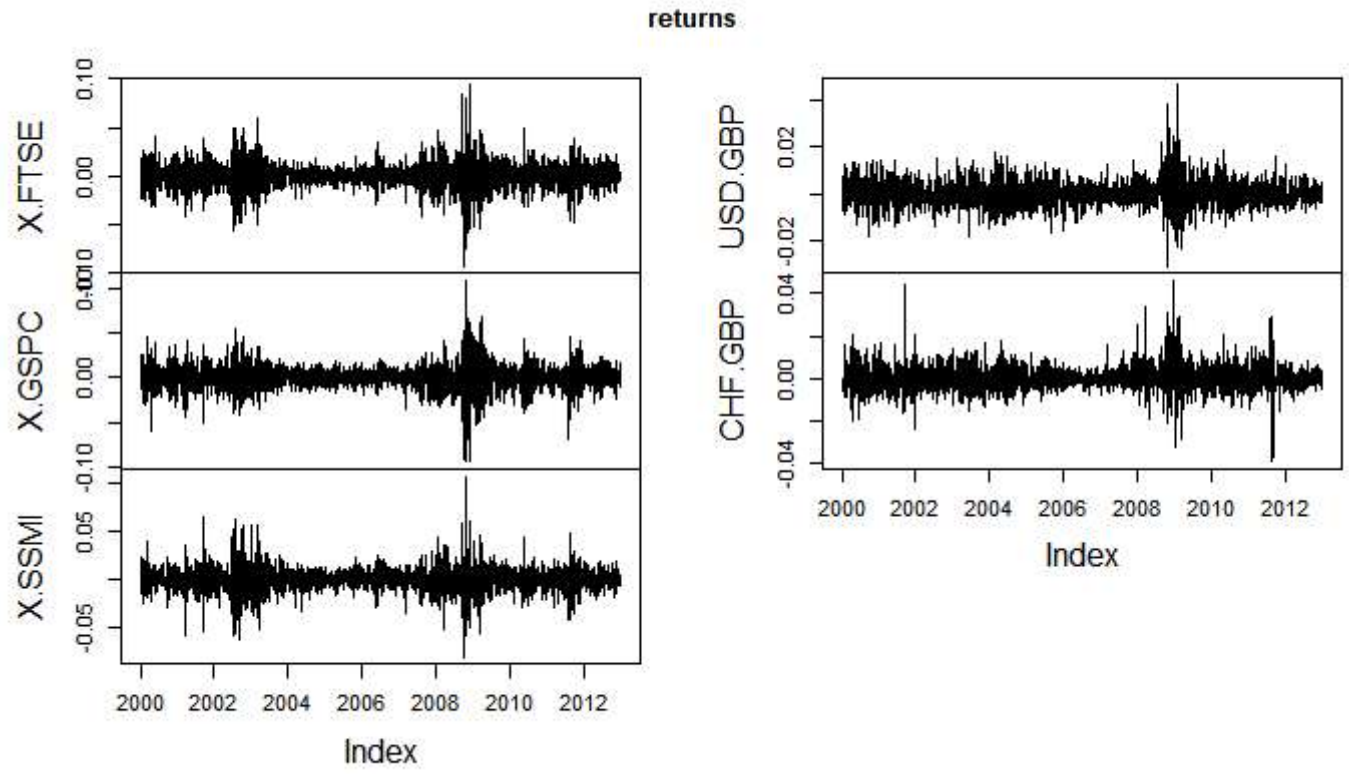
Hide

```
riskfactors <- read_csv("C:/Users/Y/Documents/1.R_scripts/ML/Risk_Modeling/Quantitative Risk Man
agement in R/riskfactors.csv",
    col_types = cols(date = col_date(format = "%m/%d/%Y")))
riskfactors_xts <- as.xts(riskfactors[,-1], order.by = riskfactors$date, "%d-%m-%Y")
# Plot the risk-factor data
plot.zoo(riskfactors_xts)
```



Hide

```
# Calculate the log-returns, assign to returns, and plot
returns <- diff(log(riskfactors_xts))[-1, ]
plot.zoo(returns)
```



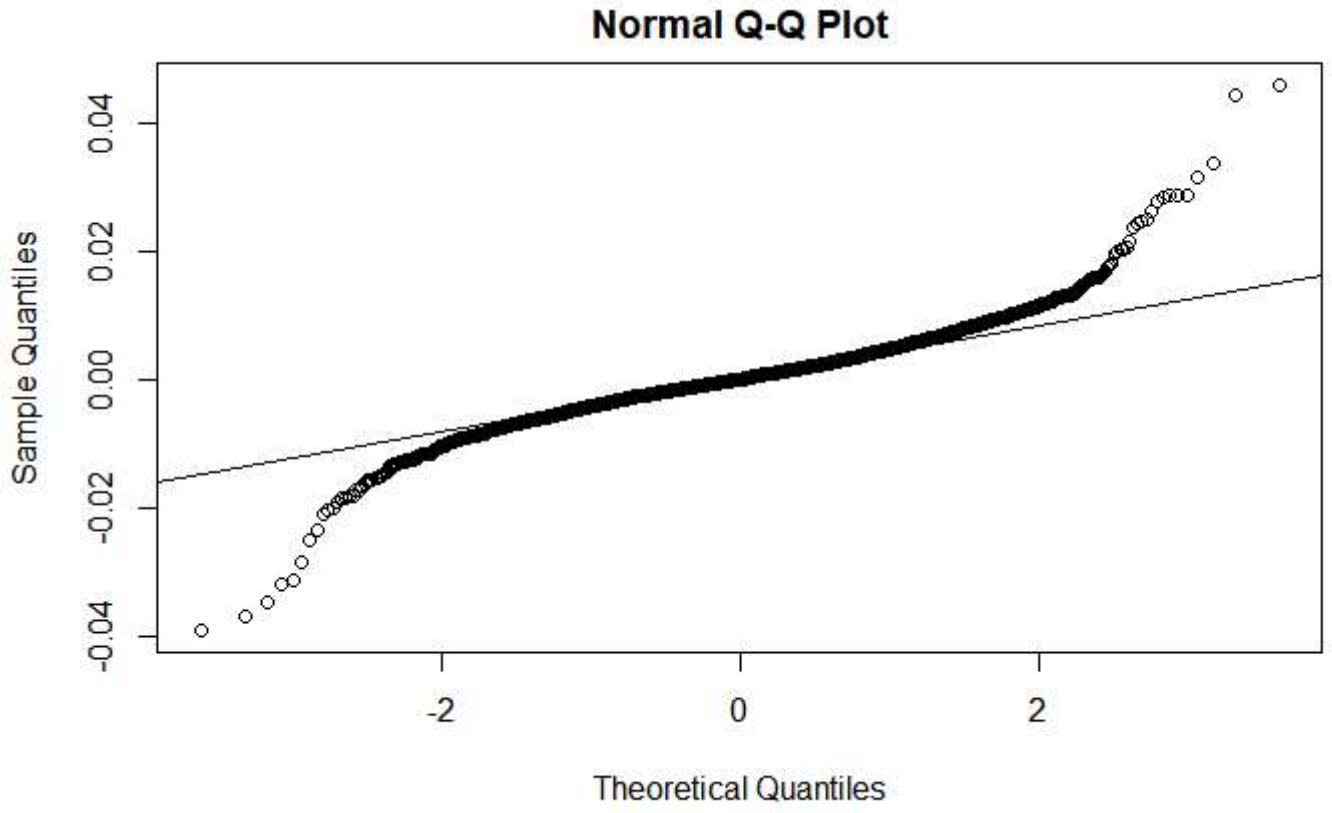
Hide

```
# Use apply() to carry out the Jarque-Bera test for all 5 series
apply(returns, 2, jarque.test)
```

```
$X.FTSE

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 4209.5, p-value < 2.2e-16
alternative hypothesis: greater


$X.GSPC

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 6961.6, p-value < 2.2e-16
alternative hypothesis: greater


$X.SSMI

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 5158.1, p-value < 2.2e-16
alternative hypothesis: greater


$USD.GBP

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 2586.4, p-value < 2.2e-16
alternative hypothesis: greater


$CHF.GBP

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 7759.2, p-value < 2.2e-16
alternative hypothesis: greater
```

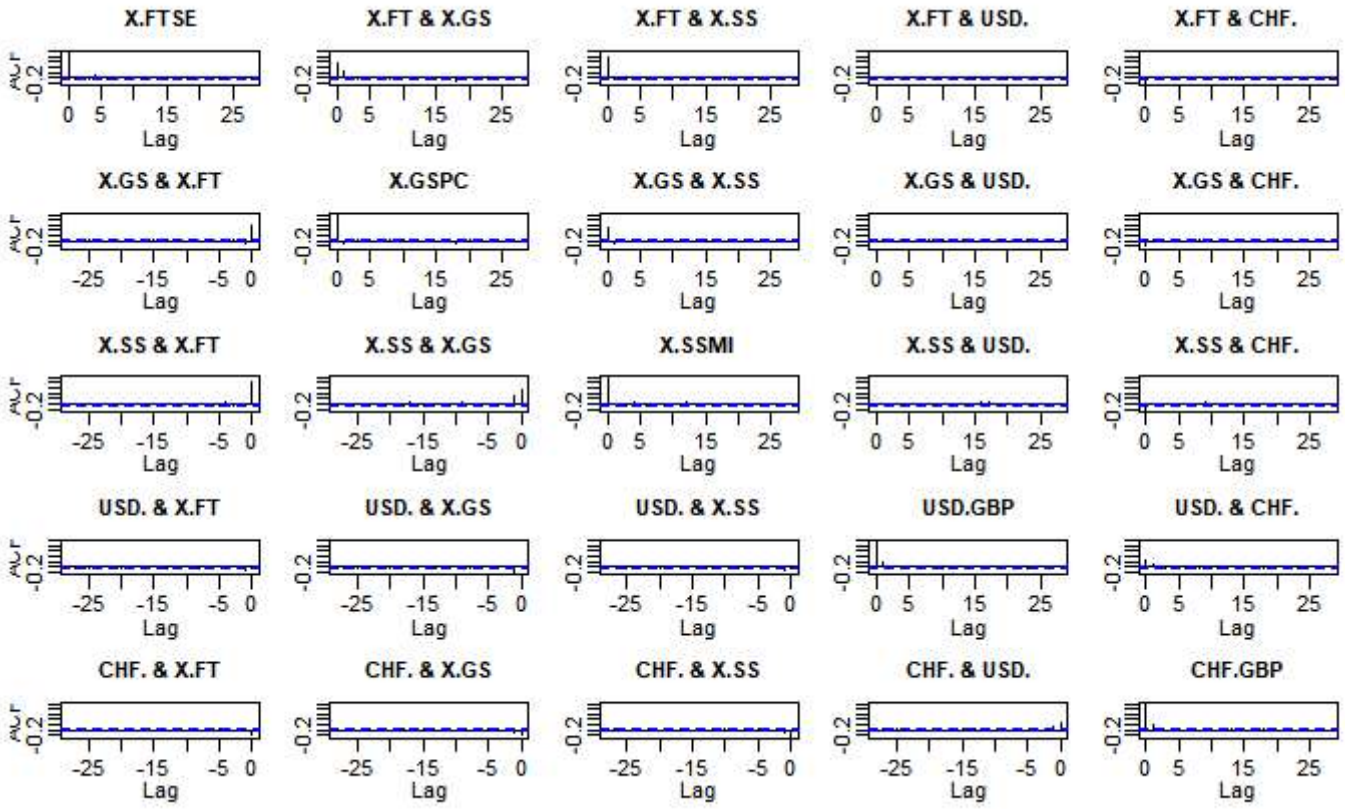
Hide

```
# Make a Q-Q plot against normal for the 5th return series and add a reference line
qqnorm(returns[, 5])
qqline(returns[, 5])
```



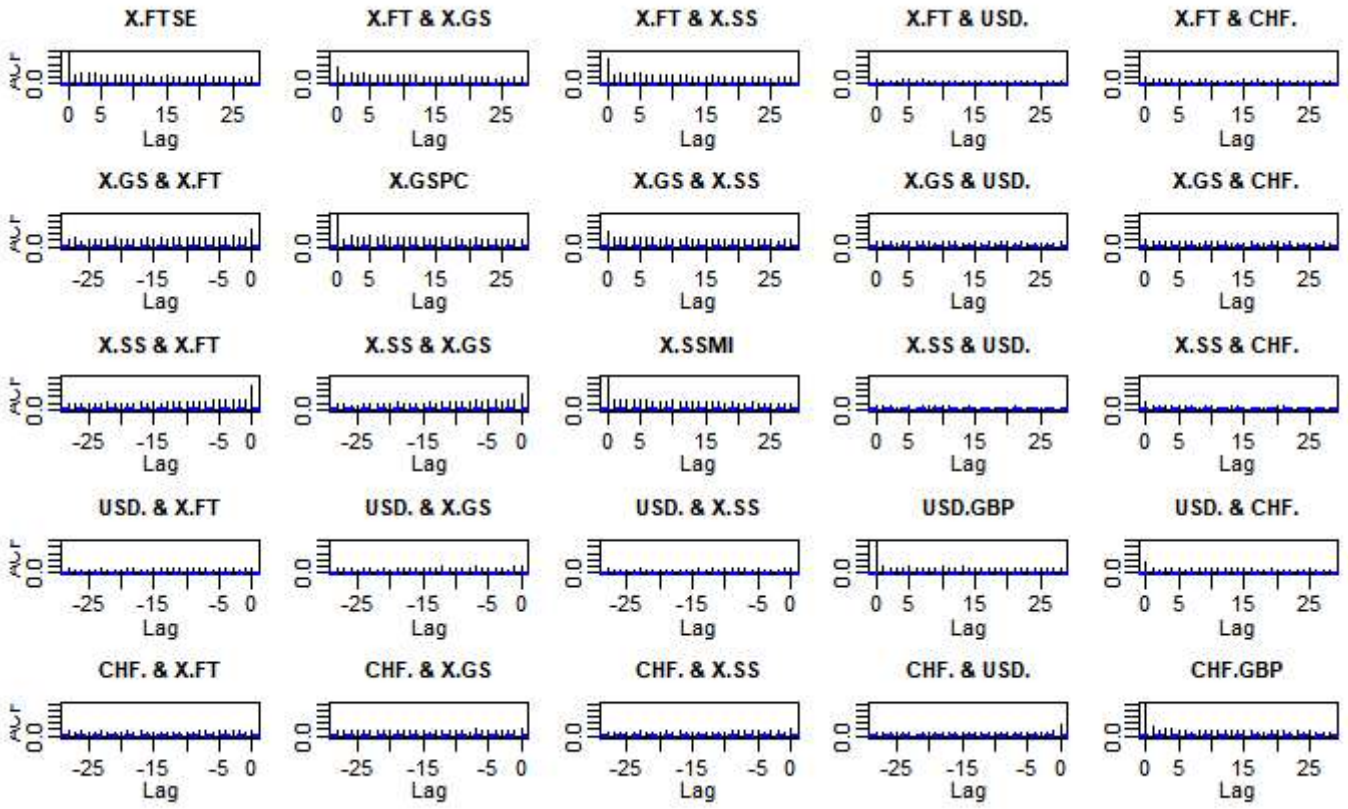
Hide

```
# Make a picture of the sample acfs for returns and their absolute values
acf(returns)
```



Hide

```
acf(abs(returns))
```



Option portfolio and Black Scholes

Implied Volatiity: If there is a market for options on a particular stock, then the prices that are quoted can be used to infer the implied values that would be used to price the options with the Black-Scholes formula. You need to consider implied volatility as a risk factor.

Historical Simulation

Task: Form historically simulated losses and examine them.

Result: The features of the underlying risk-factor returns (heavy tails and serial dependence) are present in the historically simulated losses.

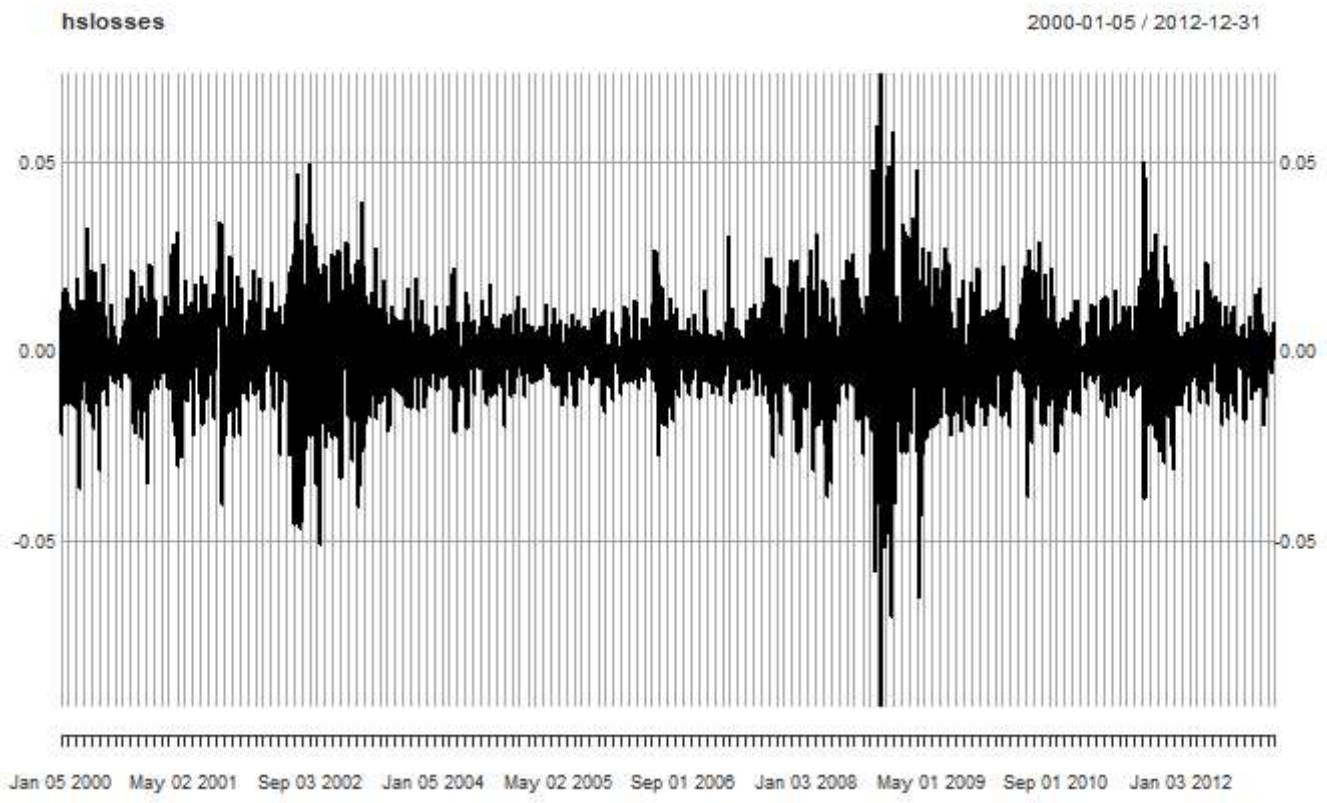
Hide

```
lossop <- function(xseries,wts=c(0.3,0.4,0.3)){
  if (is.xts(xseries))
    x <- coredata(xseries)
  else if (is.matrix(xseries))
    x <- xseries
  else
    x <- matrix(xseries,nrow=1)
  ll <- apply(x,1,function(x,wts){
  1-(wts[1]*exp(x[1]) + wts[2]*exp(x[2]+x[4]) + wts[3]*exp(x[3]+x[5]))},wts=wts)
  if (is.xts(xseries))
    ll <- xts(ll,time(xseries))
  ll
}
# Calculate the loss from a log-return of -0.1 for all risk factors
lossop(rep(-0.1, 5))
```

```
[1] 0.1554372
```

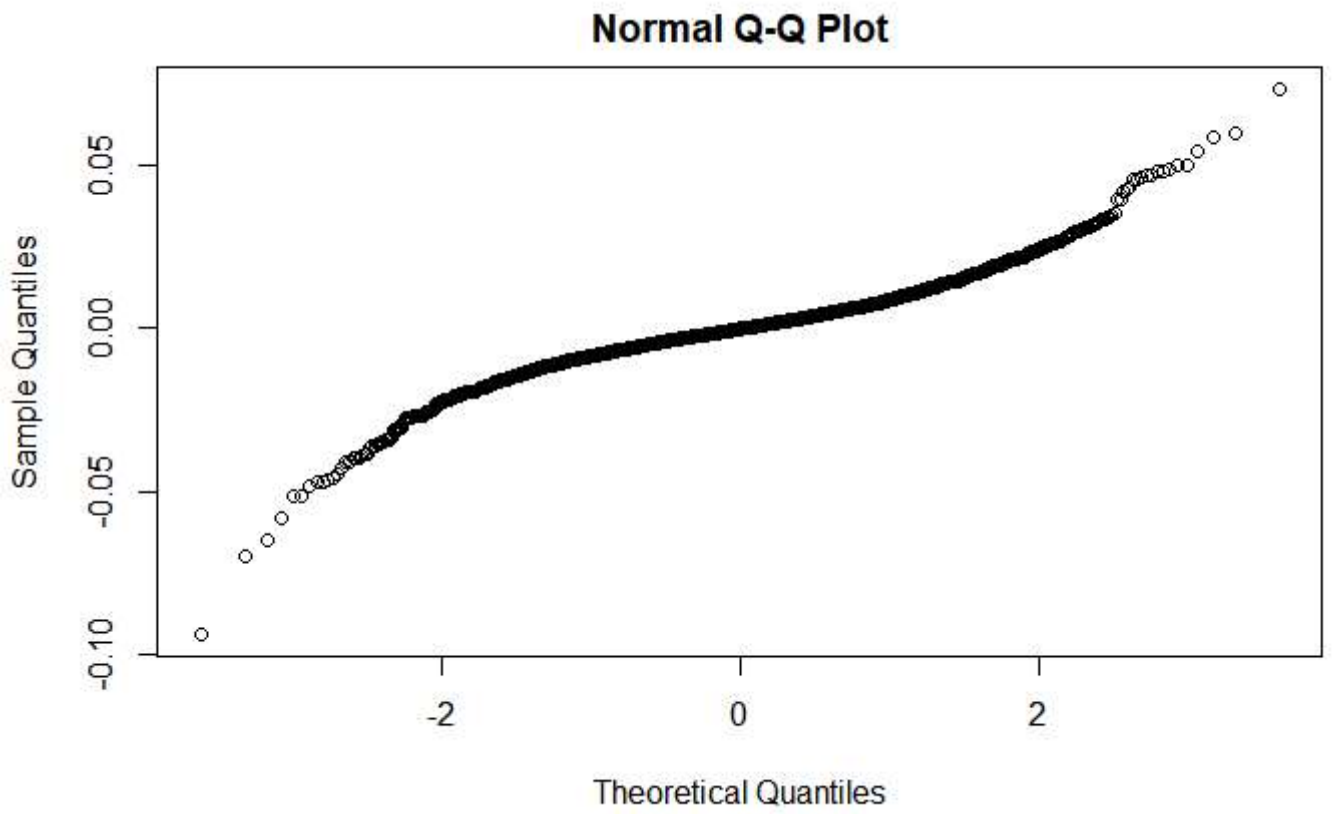
Hide

```
# Apply lossop() to returns and plot hslosses
hslosses <- lossop(returns)
plot(hslosses)
```



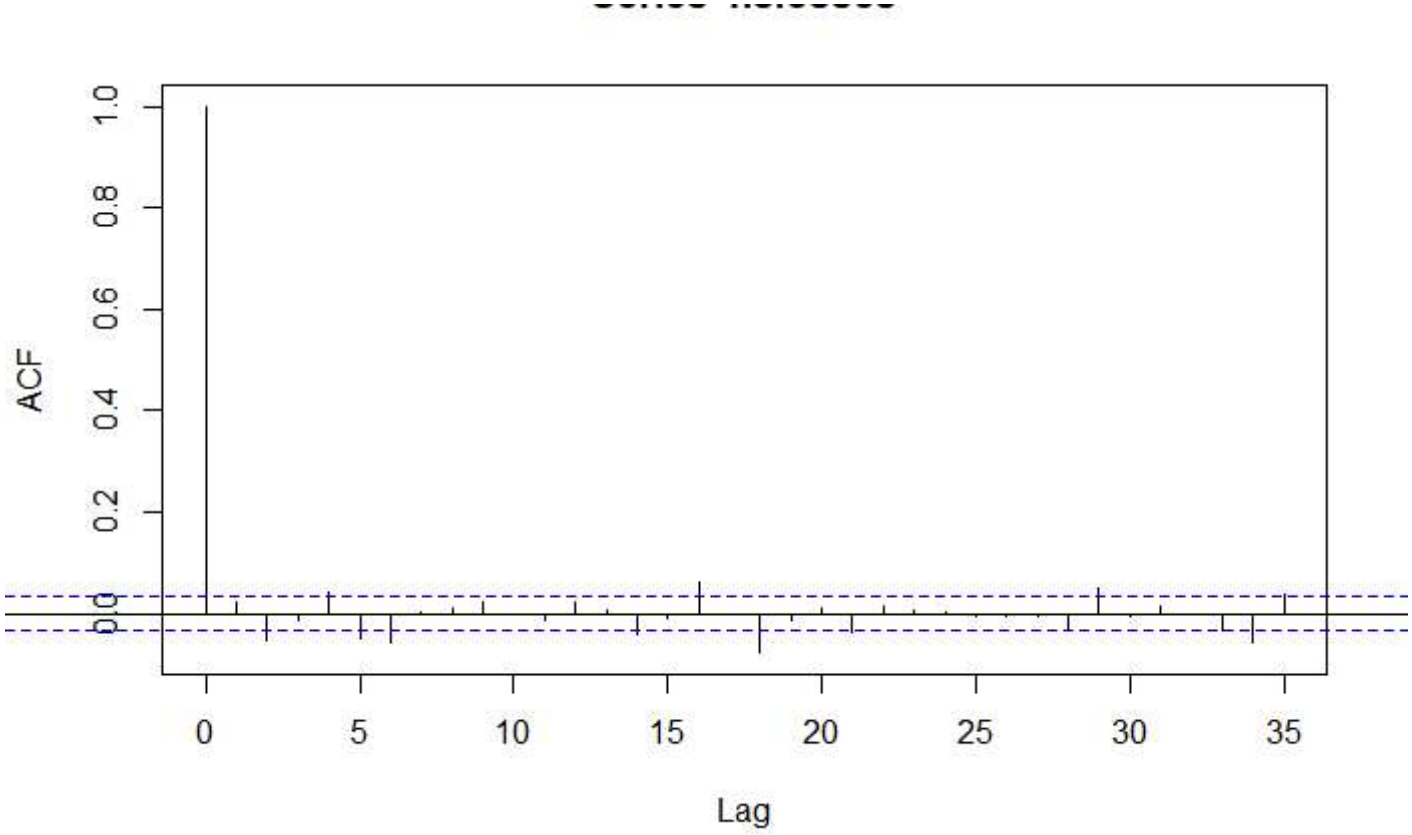
Hide

```
# Form a Q-Q plot of hslosses against normal
qqnorm(hslosses)
```



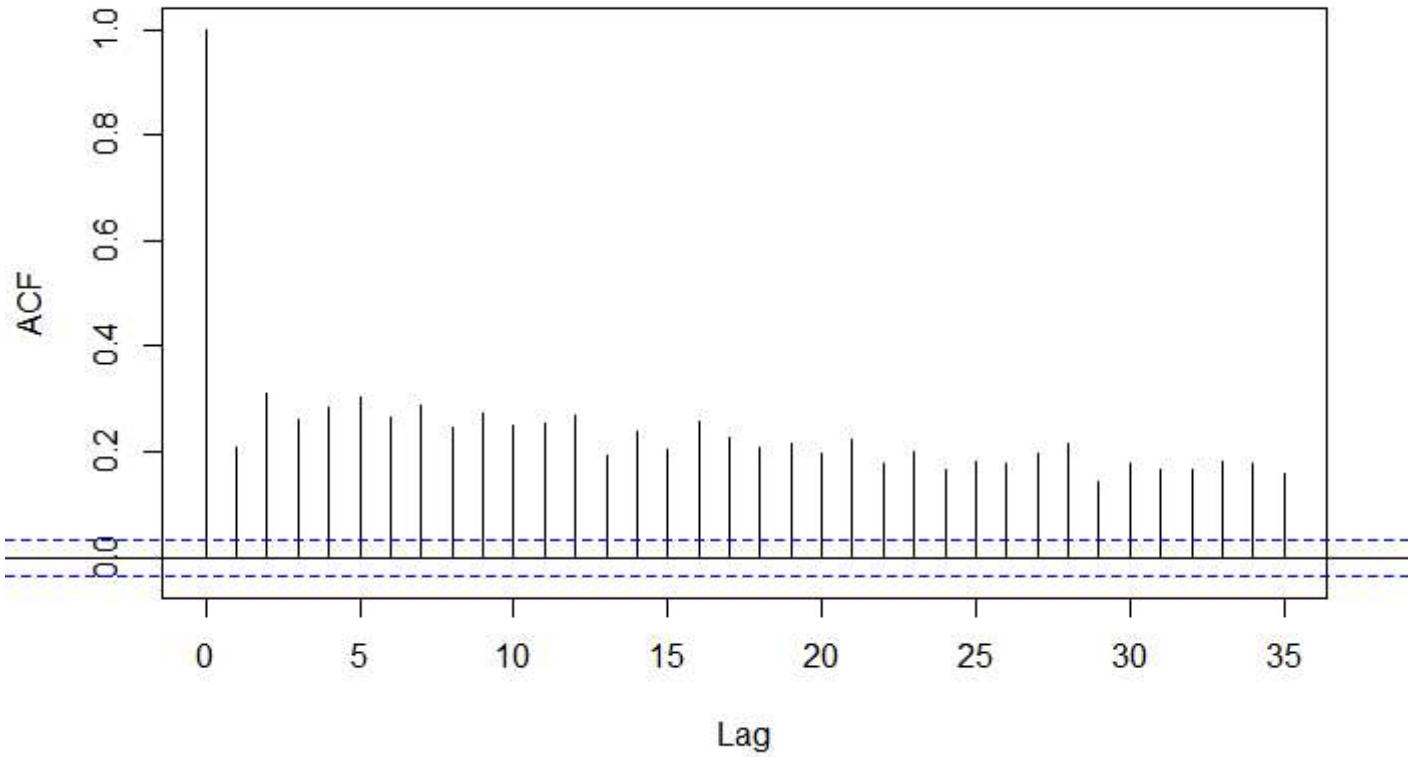
Hide

```
# Plot the sample acf of hslosses and their absolute values
acf(hslosses)
```



Hide

```
acf(abs(hslosses))
```



Estimating VaR and ES

Task: Apply a simple non-parametric method using a sample quantile to estimate VaR and the average of values exceeding the samle quantile to estimate ES. Then, compare these estimates with the values obtained when you assume that the hslosses have a normal distribution.

Result: The estimates derived from a normal assumption are much less conservative than the estimates derived using the non-parametric method.

Hide

```
# Estimate the 99th sample percentile of the distribution of hslosses
quantile(hslosses, .99)
```

```
      99%
0.03076173
```

Hide

```
# Estimate the 99% ES
mean(hslosses[hslosses >= quantile(hslosses, 0.99)])
```

```
[1] 0.04184655
```

Hide

```
# Estimate the mean and standard deviation of hslosses
mu <- mean(hslosses)
sigma <- sd(hslosses)
# Compute the 99% quantile of a normal distribution
qnorm(.99, mean = mu, sd = sigma)
```

```
[1] 0.02602973
```

Hide

```
# Compute the 99% ES of a normal distribution
ESnorm(.99, mu, sigma)
```

```
[1] 0.02983979
```

Option portfolio and Black Scholes

If there is a market for options on a particular stock, then the prices that are quoted can be used to infer the implied values that would be used to price the options with the Black-Scholes formula. You need to consider implied volatility as a risk factor.

Compute Black-Scholes price of an option

Result: The option values change dramatically as the stock price moved from out-of-the-money to in-the-money or vice versa.

Hide

```
# Set the interest rate r to be 0.01, the volatility sigma to be 0.2 and the strike K to be 100
r <- 0.01
sigma <- .2
K <- 100
# Look at the arguments of the Black_Scholes function
args(Black_Scholes)
```

```
function (t, S, r, sigma, K, T, type = c("call", "put"))
NULL
```

Hide

```
# Price a European call option that matures in one year if the current stock price is 80
Black_Scholes(0, S=80, r, sigma, K, T=1, "call")
```

```
[1] 1.302245
```

Hide

```
# Price a European call option that matures in one year if the current stock price is 120
Black_Scholes(0, S=120, r, sigma, K, T=1, "call")
```

```
[1] 22.94188
```

Hide

```
# Price a European put option that matures in one year if the current stock price is 80
Black_Scholes(0, S=80, r, sigma, K, T=1,"put")
```

```
[1] 20.30723
```

Hide

```
# Price a European put option that matures in one year if the current stock price is 120
Black_Scholes(0, S=120, r, sigma, K, T=1,"put")
```

```
[1] 1.94686
```

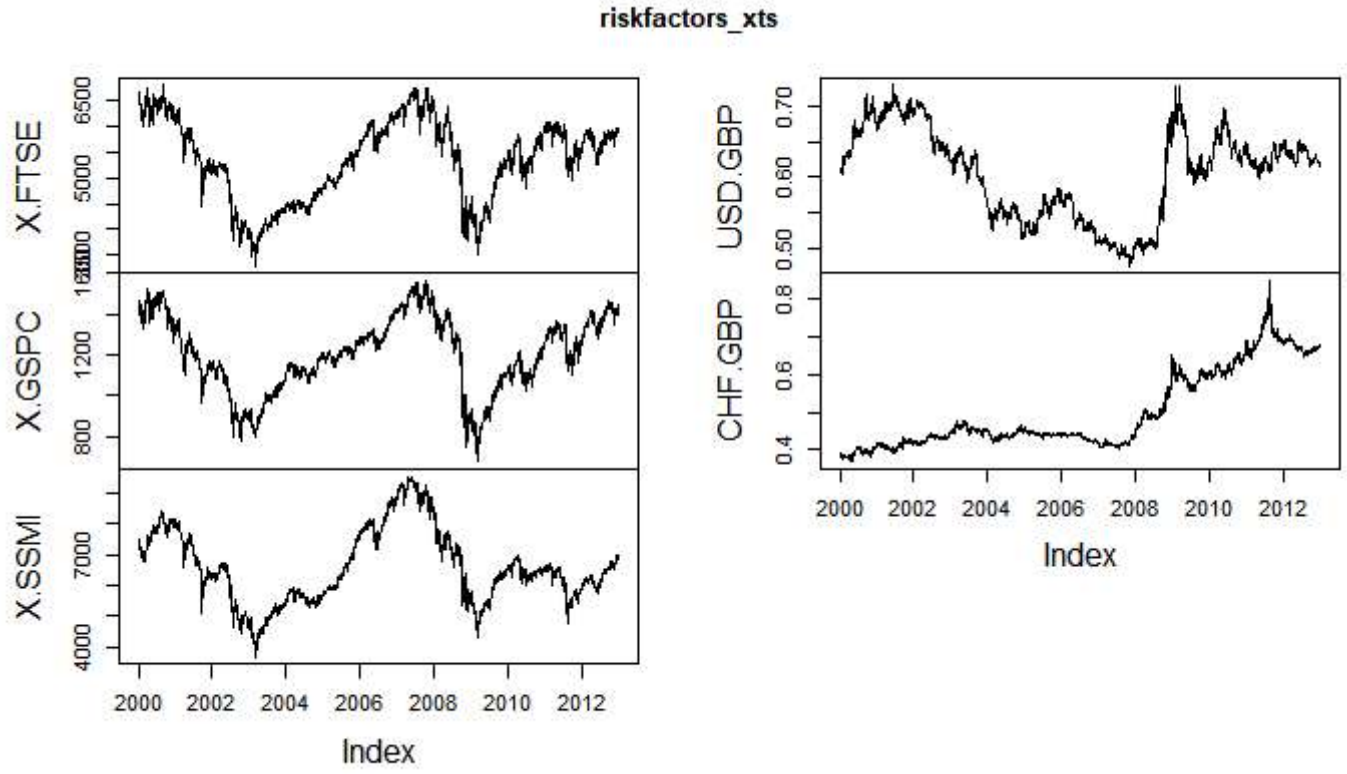
Equity and implied volatility risk factors

Task: Verify whether the log-returns of volatility behave like other return data you have encountered, and to see how they vary with the log-returns of the S&P 500 index.

Result: It is clear that the log-returns of the VIX index show the same stylized facts as other returns that you have analyzed - non-normality, heavy tails, volatility, serial dependence in the absolute values but not the raw values. Moreover, they are negatively correlated with the log-returns of the SP500 index.

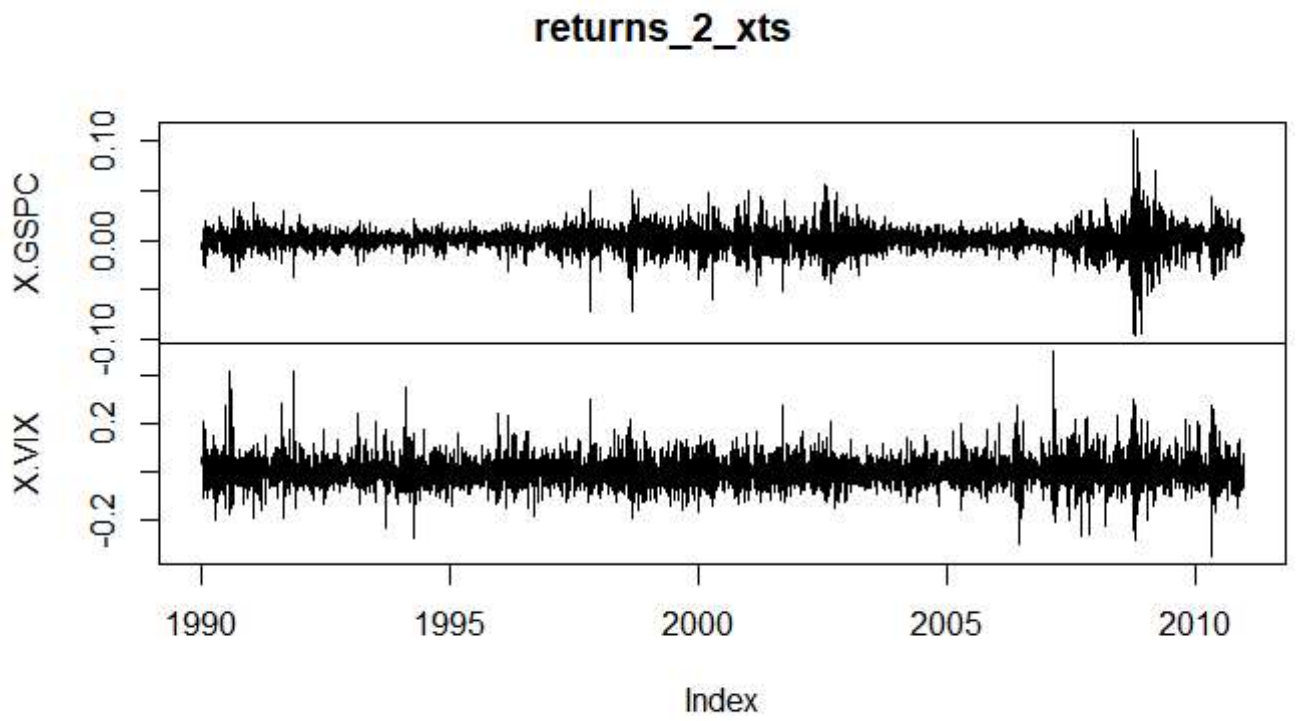
Hide

```
returns_2 <- read_csv("C:/Users/Y/Documents/1.R_scripts/ML/Risk_Modeling/Quantitative Risk Manag
ement in R/returns_2.csv", col_types = cols(date = col_date(format = "%m/%d/%Y")))
returns_2_xts <- as.xts(returns_2[,-1], order.by = returns_2$date, "%d-%m-%Y")
# Plot the risk factors and the log-returns
plot.zoo(riskfactors_xts)
```



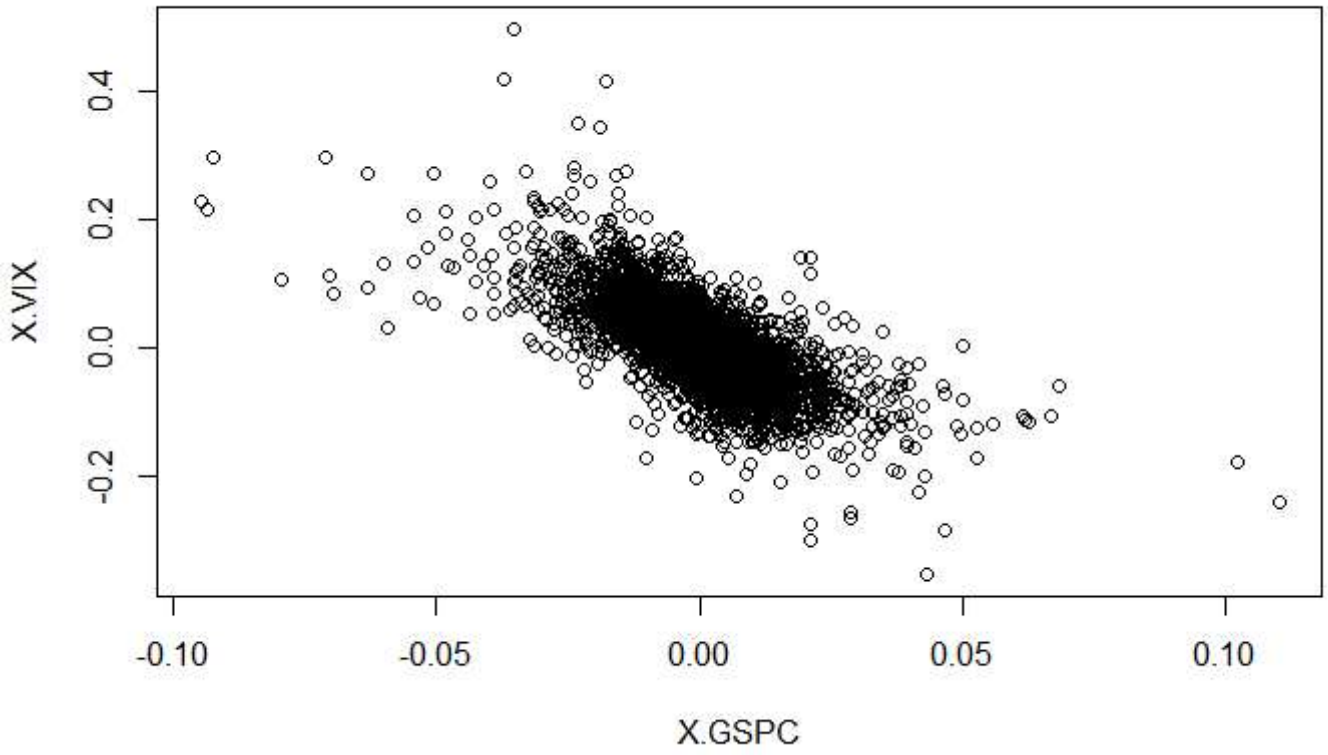
Hide

```
plot.zoo(returns_2_xts)
```



Hide

```
# Make a scatterplot of the two return series
plot(as.matrix(returns_2_xts))
```



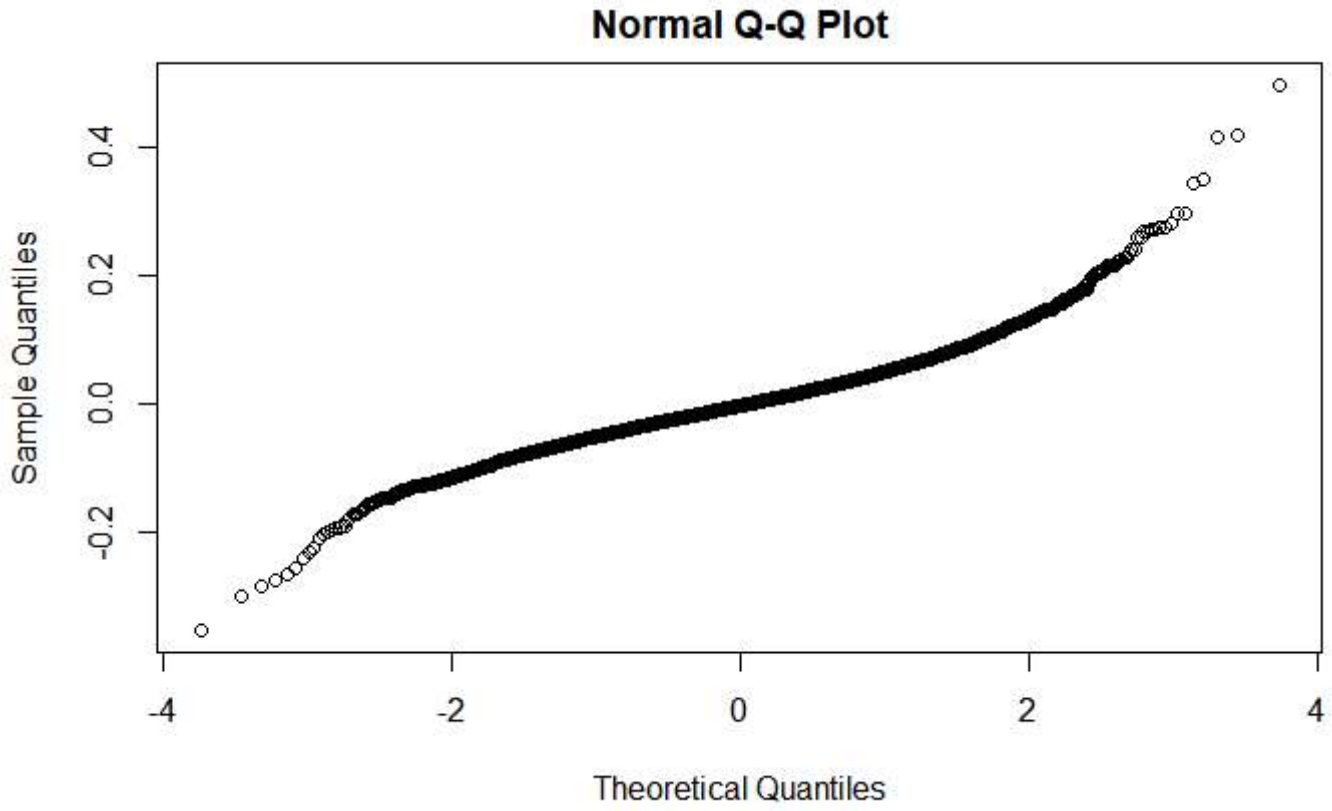
Hide

```
# Apply the Jarque-Bera test to the returns and make a Q-Q plot of the volatility log-returns
apply(returns, 2, jarque.test)
```

```
$X.FTSE

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 4209.5, p-value < 2.2e-16
alternative hypothesis: greater


$X.GSPC

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 6961.6, p-value < 2.2e-16
alternative hypothesis: greater


$X.SSMI

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 5158.1, p-value < 2.2e-16
alternative hypothesis: greater


$USD.GBP

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 2586.4, p-value < 2.2e-16
alternative hypothesis: greater


$CHF.GBP

    Jarque-Bera Normality Test

data:  newX[, i]
JB = 7759.2, p-value < 2.2e-16
alternative hypothesis: greater
```

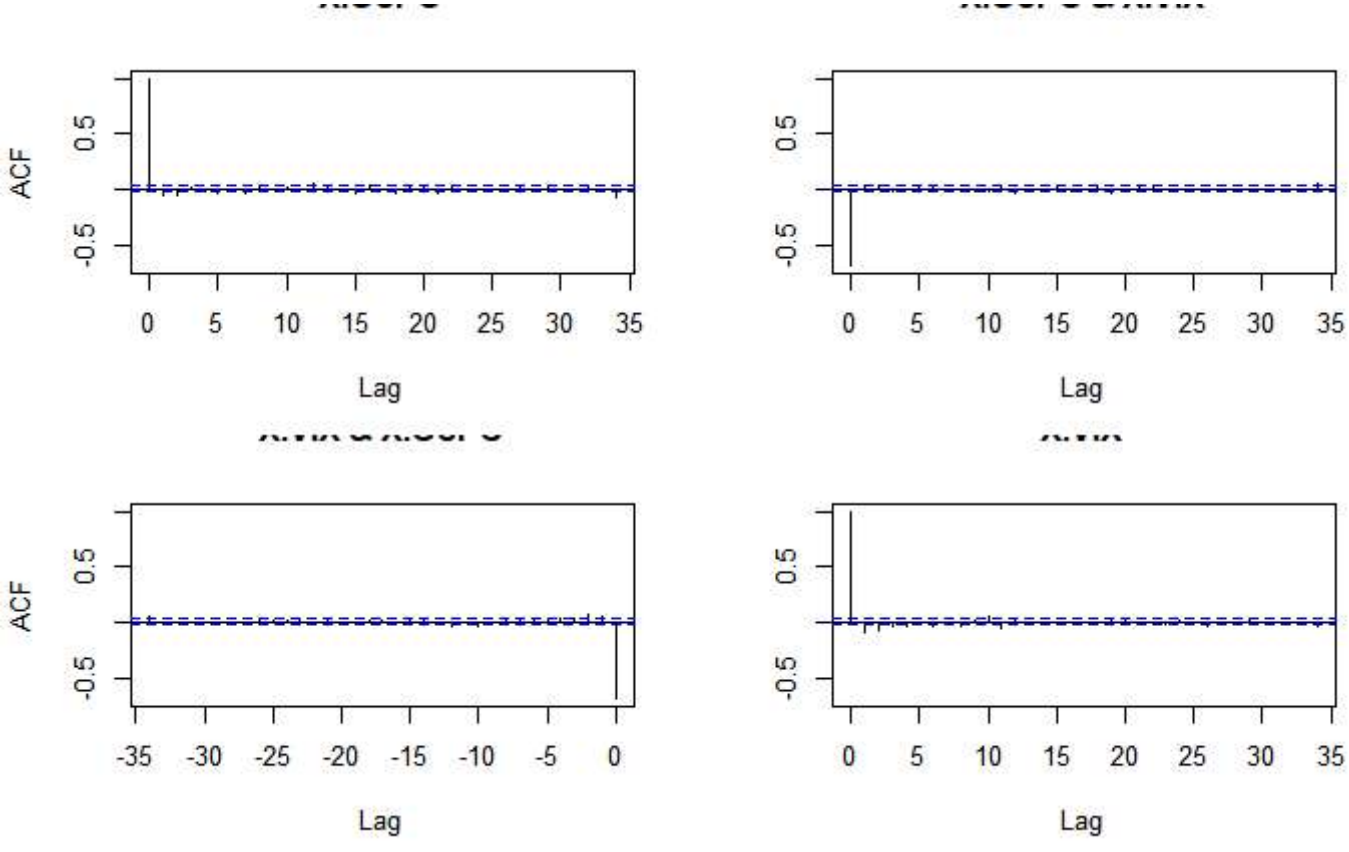
Hide

```
qqnorm(returns_2_xts[,2])
```



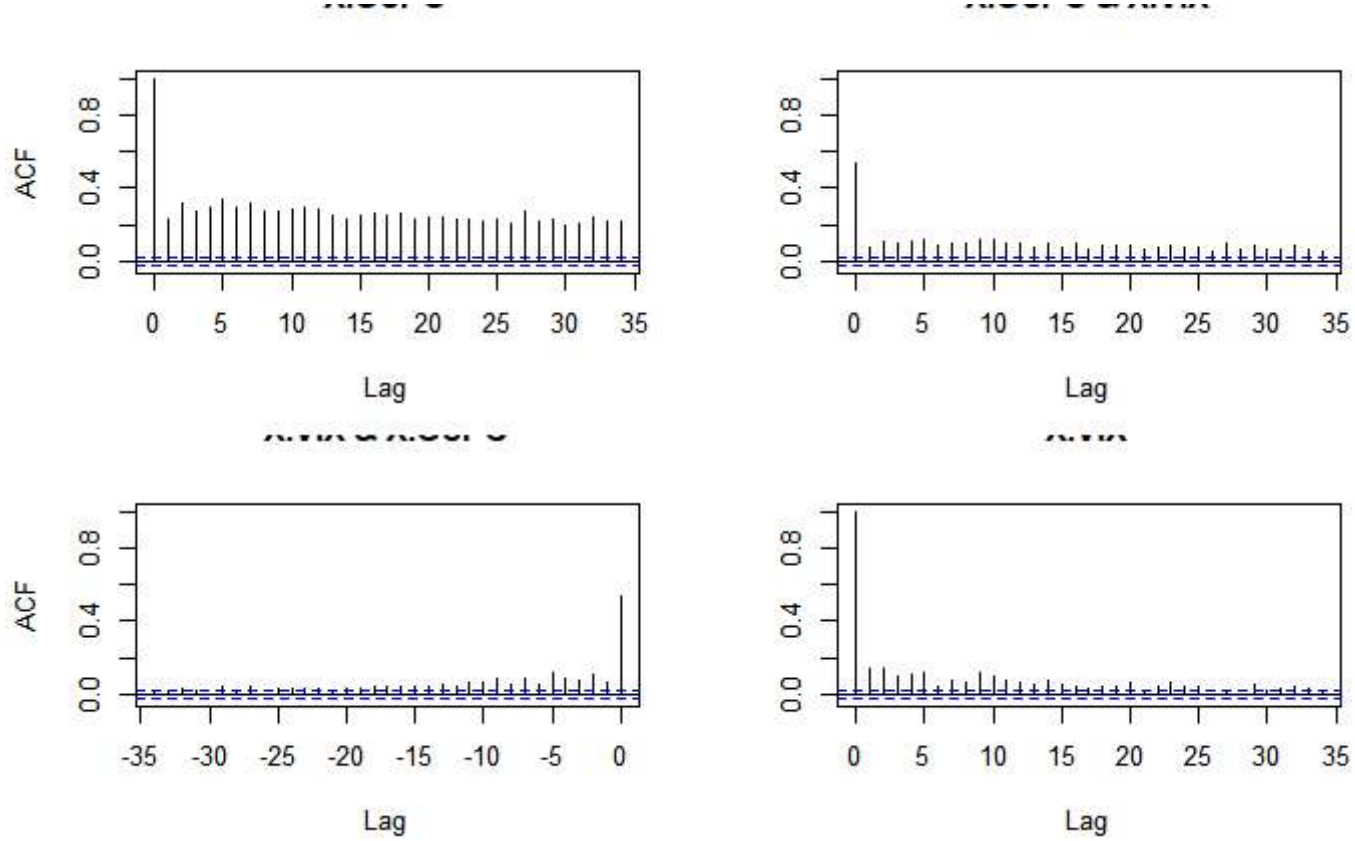
Hide

```
# Create the sample acf of the returns and absolute returns
acf(returns_2_xts)
```



Hide

```
acf(abs(returns_2_xts))
```



Hide

```
# Calculate the correlation between the log-returns
cor(returns_2_xts)
```

```
          X.GSPC      X.VIX
X.GSPC  1.0000000 -0.6978512
X.VIX  -0.6978512  1.0000000
```

Historical simulation for the option example

Esitmate VaR and ES for option simulation

Historical simulation of losses for option portfolio

Task: Form the historically simulated losses for the option portfolio and examine their properties

Result: These historically simulated losses are highly non-normal and very volatile.

Hide

```
lossop <- function(xseries,r=0.01, K=100, T=1, sigma=0.2,S=100){
  if (is.xts(xseries))
    x <- coredata(xseries)
  else if (is.matrix(xseries))
    x <- xseries
  else
    x <- matrix(xseries,nrow=1)
  ll <- apply(x,1,function(x,r,K,T,sigma,S){
    deltat <- 1/250
    V_t0 <- Black_Scholes(0, S, r, sigma, K, T, "call")
    V_t1 = Black_Scholes(deltat, exp(log(S)+x[1]), r, exp(log(sigma)+x[2]), K, T, "call")
    - (V_t1 - V_t0)/V_t0
  },
              r=r,K=K,T=T,sigma=sigma,S=S)
  if (is.xts(xseries))
    ll <- xts(ll,time(xseries))
  ll
}

  # Calculate the first loss
lossop(c(-.1,-.1), S=80, sigma=.2 )
```

```
[1] 0.8030928
```

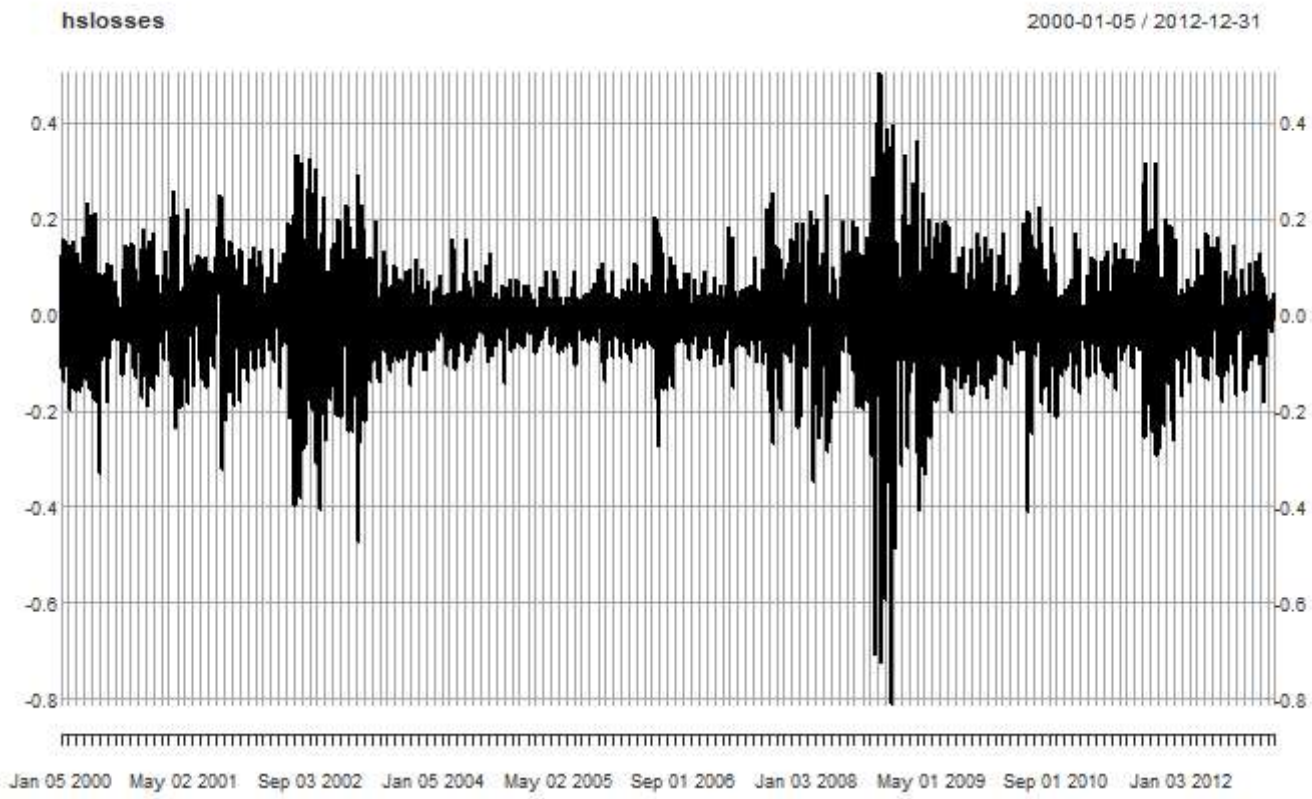
Hide

```
# Calculate the second loss
lossop(c(-.1,.1), S=100, sigma=.2 )
```

```
[1] 0.4380754
```

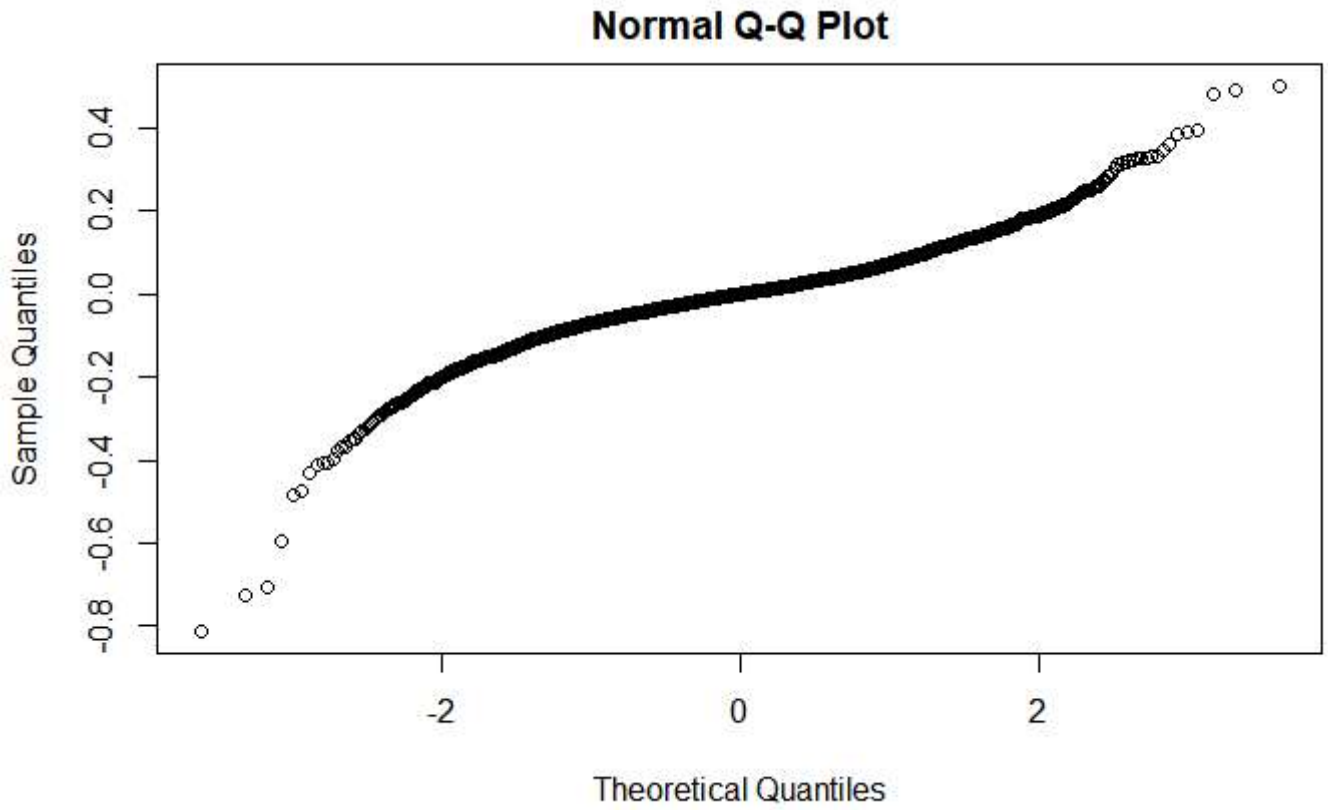
Hide

```
# Create and plot hslosses
hslosses <- lossop(returns, S=100, sigma=.2)
plot(hslosses)
```



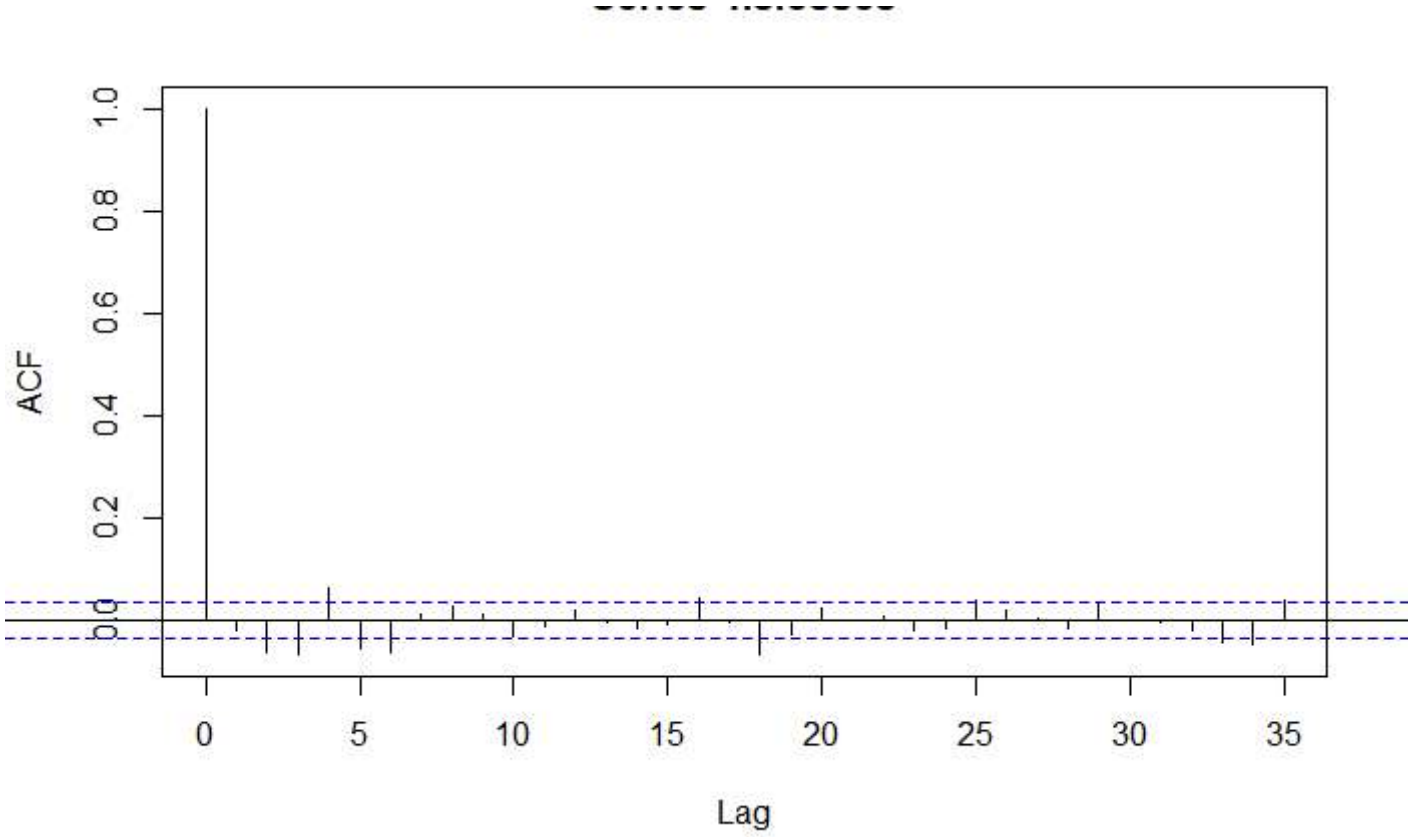
Hide

```
# Form a Q-Q plot of hslosses against normal
qqnorm(hslosses)
```



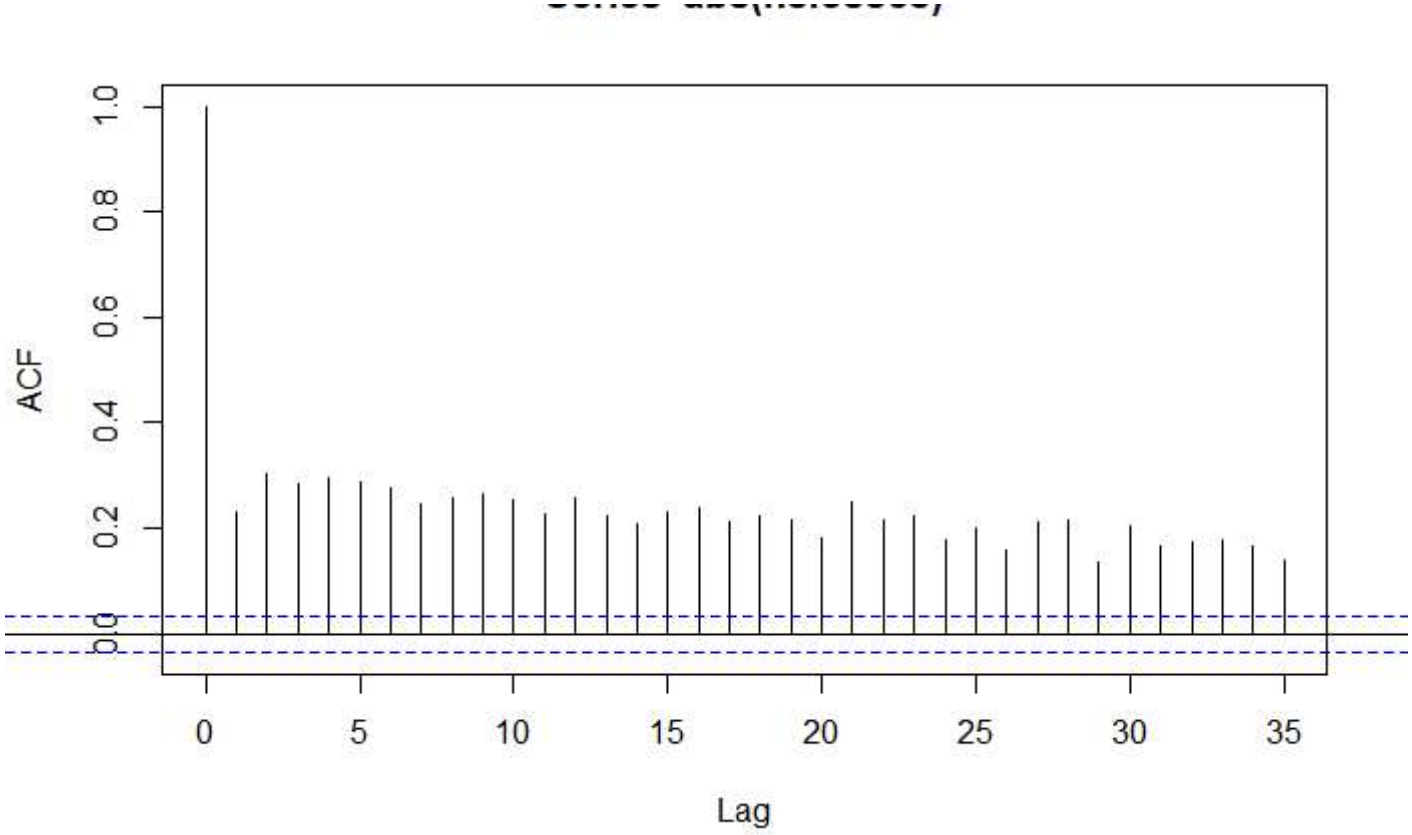
Hide

```
# Plot the sample acf of raw data and absolute values in hslosses
acf(hslosses)
```



Hide

```
acf(abs(hslosses))
```



Estimating VaR and ES for option portfolio

Task: Estimate VaR and ES for the investor in the European call option using the historically simulated losses and gains in hslosses.

Result: The normal distribution greatly underestimates these measures of risk.

Hide

```
# Estimate the 99.5% percentile of the distribution
quantile(hslosses, .995)
```

```
    99.5% 
0.3166161 
```

Hide

```
# Estimate the 99.5% ES
mean(hslosses[hslosses >= quantile(hslosses, 0.995)])
```

```
[1] 0.3708309
```

Hide

```
# Estimate the mean and standard deviation of hslosses
mu <- mean(hslosses)
sigma <- sd(hslosses)
# Compute the 99.5% quantile of a normal distribution
qnorm(.995, mean=mu, sd=sigma)
```

```
[1] 0.2428193
```

Hide

```
# Compute the 99.5% ES of a normal distribution
ESnorm(.995, mu, sigma)
```

```
[1] 0.2726587
```

Computing VaR for weekly losses

Task: Find the weekly log-returns of returns. Use these weekly log-returns to simulate the losses of the two risk factors.

Hide

```
returns_w <- apply.weekly(returns_2_xts, colSums)
hslosses <- lossop(returns_w, S = 120, sigma = 0.25)
# Estimate the 99% ES
mean(hslosses[hslosses >= quantile(hslosses, 0.99)])
```

```
[1] 0.2475119
```

Hide

```
# Estimate the mean and standard deviation of hslosses
mu <- mean(hslosses)
sigma <- sd(hslosses)
# Compute the 99% quantile of a normal distribution
qnorm(.99, mean = mu, sd = sigma)
```

```
[1] 0.1596881
```

Filtered historical simulation: GARCH, EWMA

This can be achieved by filtered historical simulation. The Var & ES estimated are scaled by the predicted volatility for the risk horizon in question. If recent data indicate the volatility is likely to be high, a higher scaling is applied to the Var or ES estimate. If recent data indicate that the volatility is going to be low, a lower scaling is applied to the

estimate. To do this use GARCH (Generalized Autoregressive Conditional Heteroskedasticity) models and EWHa (exponential weighted moving average) volatility filters.