Statistical Methods with Application to Finance

R. Pappadà

a.a. 2025-2026

Lab 1

Introduction

R is a free software environment for statistical computing and graphics that runs on most operating systems. To obtain **R**, point your browser to the *Comprehensive R Archive Network* (CRAN), http://cran.r-project.org/ and download and install it. The installation includes help files and some user manuals. **RStudio** (https://www.rstudio.com/) is a open source integrated development environment for **R**. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management. A user types a command, presses enter, and the answer is returned.

After starting **R**, the first thing to do is to set the *working directory*, the computer directory where data sets reside and output will be stored. In **RStudio** this can be done from the command menu `Session` → `Set working directory`, that allows one to select the desired directory. An alternative method is to type in the desired directory in the **R** Console using the command `setwd()`.

Exploratory Data Analysis

R comes with some standard packages that are installed when you install it. However, additional **R** packages may be required for performing more complex analysis such as time series analysis. First, the quantmod package, used for obtaining financial data directly from some open sources, is installed and loaded. Once you have started **R**, you can install an additional package (e.g. the "quantmod" package) by choosing

`Install Packages` from the `Tools` menu at the top of the **RStudio** console, or by the line-command

```
install.packages("quantmod")
```

Then, whenever you want to use the quantmod package after this, you first have to load the package

```
library(quantmod)
```

Other packages that will be used during the lab and need to be installed are

```
library(moments)
library(tseries)
```

To collect your commands, create a new **R** script from the `File` menu; you can also use it to source a file (the same as the `source()` function). From the new script, the code can be executed using the `Ctrl+Enter` key (or the `Run` toolbar button). You can save the script file in your working directory and reuse or modify it in other sessions.

How to Get Data using `quantmod`

The function `getSymbols()` in the quantmod package can be used to load and manage data from multiple sources. The first argument of this function is a character vector specifying the name of the symbol to be downloaded and the second one specifies the environment where the object is created. The help page of this function (`?getSymbols`) provides more information.

By default, objects are created in the workspace. The following code downloads the S&P 500 time series (2000–2016) from *Yahoo Finance* and stores the data in a new object, which we call `sp500`

```
#download the S&P 500 time series from Yahoo Finance
getSymbols("^GSPC", src = "yahoo",
           from = "2000-01-02", to = "2016-12-31")

## [1] "GSPC"

sp500 <- GSPC
#show the first six rows of the data
# head(sp500)
```

The data object is an *extensible time series* (xts) object that allows for conveniently selecting single time series using $. We can obtain the *adjusted daily prices* that have been adjusted for dividends and stock splits, by selecting the appropriate column:

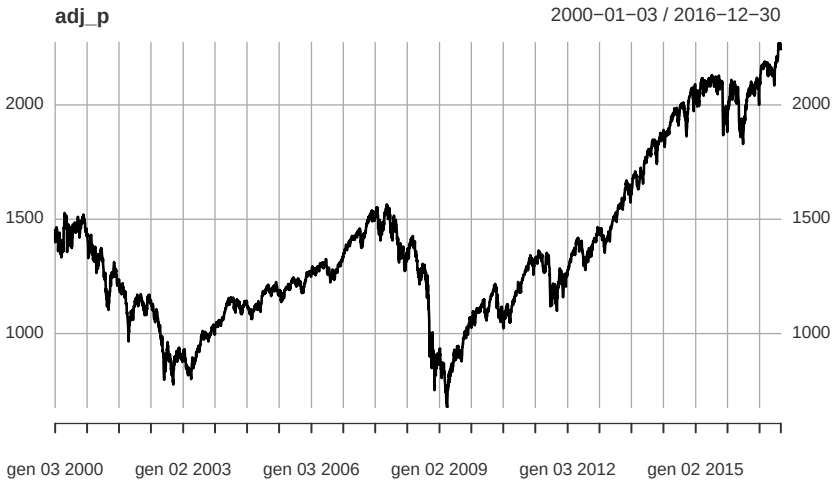
```
#extract the closing prices
adj_p<-sp500$GSPC.Adjusted

head(adj_p)   #show the first rows of the data

##            GSPC.Adjusted
## 2000-01-03       1455.22
## 2000-01-04       1399.42
## 2000-01-05       1402.11
## 2000-01-06       1403.45
## 2000-01-07       1441.47
## 2000-01-10       1457.60
```

One can also select a subset of the rows/colums by using a character row index, as in the following example

```
sp500["2015-03-10", "GSPC.Adjusted"]

##            GSPC.Adjusted
## 2015-03-10       2044.16
```

A *time series plot* is a plot of the series in chronological order. The **R** function `plot()` will produce the time series plot for the S&P 500 index:

```
plot(adj_p)
```



We can also create a vector containing the price series by using the function `as.vector()`

```
val<-as.vector(adj_p$GSPC.Adjusted)
head(val)

## [1] 1455.22 1399.42 1402.11 1403.45 1441.47 1457.60

summary(val)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   676.5  1126.2  1295.0  1376.1  1508.3  2271.7
```

The time series plot appears non-stationary, because the level of the process is clearly non constant over the observed period.
The data downloaded before are stored in the .csv file *sp-500.csv* that can be read in the following way:

```
# Load csv file of SP500 Index (03 Jan 2000 - 30 Dec 2016)
Index <- read.csv("sp500.csv")

#Dimension of the data object
dim(Index)

## [1] 4276    2

#See the first 6 rows
head(Index)

##       date   price
## 1 20000103 1455.22
## 2 20000104 1399.42
## 3 20000105 1402.11
## 4 20000106 1403.45
## 5 20000107 1441.47
## 6 20000110 1457.60

#See the last rows
#tail(sp500)
```

Note that, in this case, the column 'date' is formed by integer values, not characters.

Log returns

Daily log returns are simply the change series of log prices:

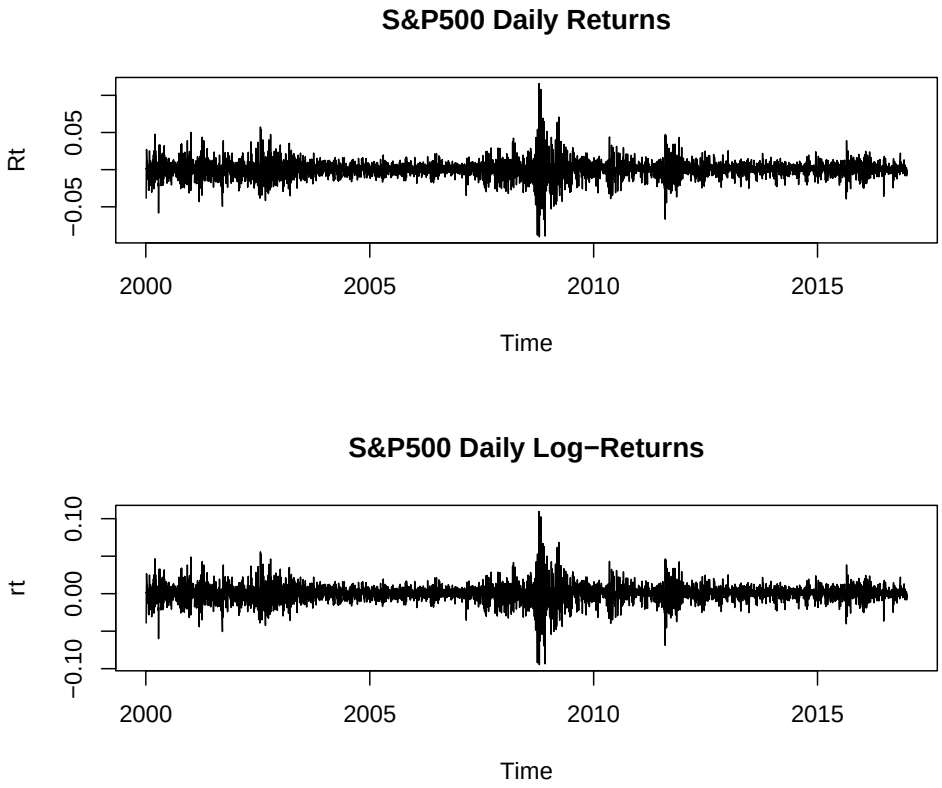
$$r_t = \log(P_t) - \log(P_{t-1}), \quad t = 2, \ldots, N$$

where P_t is the stock price at time t. In the code below `rt` is the series of log returns for the S&P 500 data.

```
#extract index values
It<-Index[,2]

#Compute log returns
rt <- diff(log(It))

# n. of obs.
N <- length(rt)

# Simple returns
Rt <- exp(rt) - 1
```

The time plots of daily simple and log returns are shown below.

```
par(mfrow=c(2,1))
time<-as.Date(as.character(Index[-1,1]), "%Y%m%d")
plot(time, Rt, type="l", xlab="Time", main="S&P500 Daily Returns")
plot(time, rt, type="l", xlab="Time", main="S&P500 Daily Log-Returns")
```



From the plots, the behavior of log returns r_t is similar to that of the simple returns R_t. The Pearson's correlation coefficient between the simple and log returns is obtained as

```
#correlation coefficient
cor(Rt, rt)

## [1] 0.9998029
```

As expected, daily simple returns of S&P 500 are small and thus $\log(1 + R_t) \approx R_t$ in this case.

Remark (Differencing)**.** *In finance, price series are commonly believed to be non-stationary. Many empirical studies on series of this type have indicated that an approximation to stationarity can be obtained by taking the first differences of successive log-transformed observations, say,*

$$\nabla \log(x_t) = \log(x_t) - \log(x_{t-1})$$

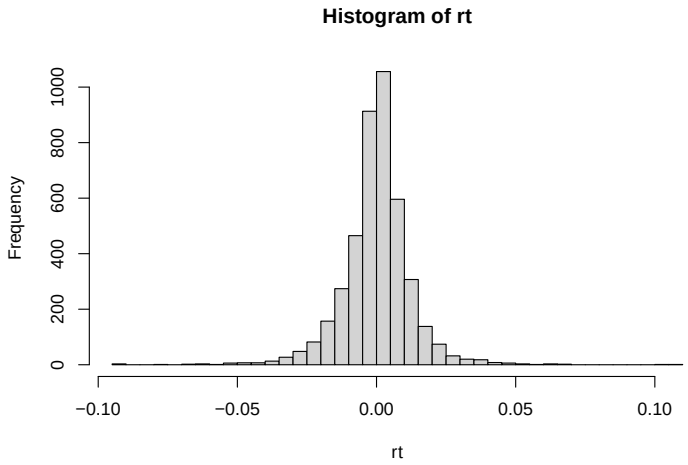
which can be interpreted as being the approximate relative change in the original variable, in the same way that a log return approximates a return that is the relative change in price.

The time plot of the daily log returns on the S&P 500 index exhibits a phenomenon called *mean-reversion*, that is, the plot shows random oscillation around some fixed level. Thus, log returns seem suitable for modelling as stationary. On the other hand, the returns exhibit periods of low and high volatility even though they maintain a mean near 0. This phenomenon, known as volatility clustering is modeled by the so-called GARCH processes.

Histograms and Kernel Density Estimation

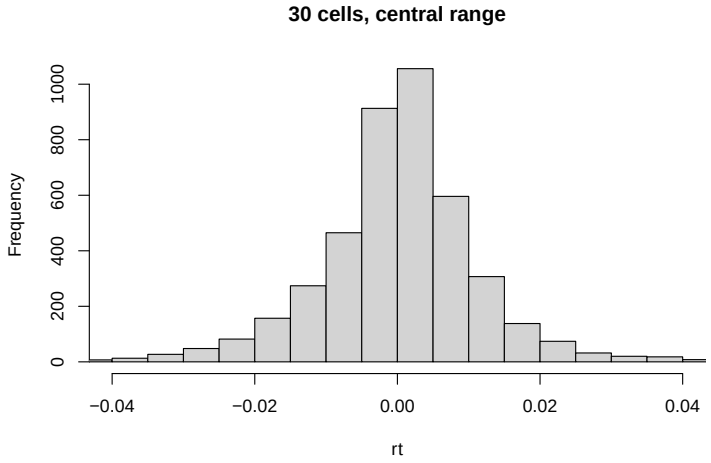
To gain a better visualization of the distribution of S&P 500 returns, we can exam either the histogram or empirical density function of the data. To obtain the histogram run the following code:

```
hist(rt, breaks=30)
```



This is obtained by dividing the data range into 30 cells or bins. The plot confirms that the returns appear to be symmetric with respect to its mean zero. When the sample size is large, the outliers are difficult, or even impossible, to see in the histogram, except that they have caused the x-axis to expand. We can zoom in on the high-probability region by running the following lines:

```
hist(rt, breaks=30, main="30 cells, central range", xlim=c(-.04,.04))
```

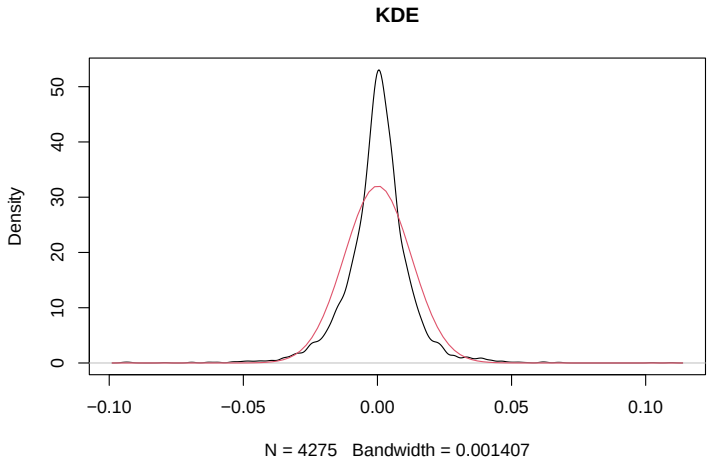


The *empirical density function* can be regarded as a refined version of the histogram. The so-called *kernel-density estimator (KDE)* based on $r_1, r_2, \ldots, r_N$ is

$$\hat{f}(r) = \frac{1}{Nb} \sum_{i=1}^{N} K\left(\frac{r - r_i}{b}\right)$$

where K is the *kernel* – a non-negative function – and $b > 0$ is a smoothing parameter called the *bandwidth* that determines the resolution of the estimator. The standard normal (mean 0 and variance 1) can be used as kernel function:

```
KDE<-density(rt, kernel="gaussian") #Kernel Density Estimation
plot(KDE, main="KDE")
# add normal density curve
curve(dnorm(x, mean(rt), sd(rt)), add=TRUE, col=2)
```



The red line shows the density function of a normal distribution that has mean and standard deviation equal to the sample mean and standard deviation of the returns. We see that the kernel estimate and the normal density are somewhat dissimilar. The empirical density function has a higher peak and longer tails than the normal density. This phenomenon is common for daily stock returns.

Summary Statistics

In finance, the first fourth moments of a random variable are used to describe the behavior of asset returns. The mean, standard deviation, skewness coefficient, and kurtosis are moments-based location, scale, and shape parameters. Moments of a random variable can be estimated by their sample counterparts. Given the series of returns $\{r_1, \ldots, r_N\}$, the sample mean is

$$\bar{r} = \frac{1}{N} \sum_{t=1}^{N} r_t,$$

and the sample standard deviation is

$$s = \sqrt{\frac{1}{N-1} \sum_{t=1}^{N} (r_t - \bar{r})^2},$$

For asset returns, the standard deviation is a measure of uncertainty sometimes known as *volatility*.

The first two moments of a random variable uniquely determine a normal distribution. For other distributions, higher order moments are also of interest. The sample skewness is

$$\widehat{sk} = \frac{1}{(N-1)s^3}\sum_{t=1}^{N}(r_t - \bar{r})^3,$$

and the sample kurtosis is

$$\widehat{kur} = \frac{1}{(N-1)s^4}\sum_{t=1}^{N}(r_t - \bar{r})^4$$

The skewness and kurtosis of the normal distribution are constants that are equal to 0 and 3, respectively. These summary statistics for daily S&P 500 returns from 2000 to 2016 can be easily computed in **R**:

```
 # Commands for individual moments
 mean(rt)

## [1] 0.0001007711

 var(rt)

## [1] 0.0001550384

 sd(rt)

## [1] 0.01245144

 skewness(rt)

## [1] -0.1937244

 kurtosis(rt)

## [1] 11.16

 # Compute an overview of basic statistics
 summary(rt)

##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -0.0946951 -0.0053047  0.0004881  0.0001008  0.0058475  0.1095720
```

The daily mean is very small while daily volatility is around 1.2%. The lowest daily return of -9.5% corresponds to the crisis of 2008. The returns have a small negative skewness and, more importantly, quite high kurtosis.

An interesting question is whether the distribution of returns has zero mean. Consider the hypotheses

$$H_0 : \mu = \mu_0 = 0; \qquad H_1 : \mu \neq 0$$

where μ denotes the population mean. The test statistic is then

$$t = \frac{\bar{r} - \mu_0}{s/\sqrt{N}}$$

which follows a Student-t distribution with $N - 1$ degrees of freedom under H_0, i.e., for $\mu = \mu_0 = 0$. The rejection region is

$$\left| \frac{\bar{r}}{s/\sqrt{N}} \right| > t_{1-\alpha/2,N-1}$$

where $t_{1-\alpha/2,N-1}$ is the $(1 - \alpha/2)$-quantile of a Student-t distribution with $N - 1$ degrees of freedom, and can be approximated with the quantile of order $(1 - \alpha/2)$ of the standard normal distribution, for large N (for instance, if $\alpha = 0.05$, then $t_{1-\alpha/2,N-1} \approx z_{1-\alpha/2} = 1.96$).

The function `t.test` is used to perform a one sample t-test at level $\alpha = 0.05$:

```
# Testing mean return = 0
t.test(rt)

##
##  One Sample t-test
##
## data:  rt
## t = 0.52916, df = 4274, p-value = 0.5967
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.0002725845  0.0004741267
## sample estimates:
##    mean of x
## 0.0001007711
```

If we look at the p-value for this test, we cannot reject H_0, that is, the data support the hypothesis that the population mean is zero.

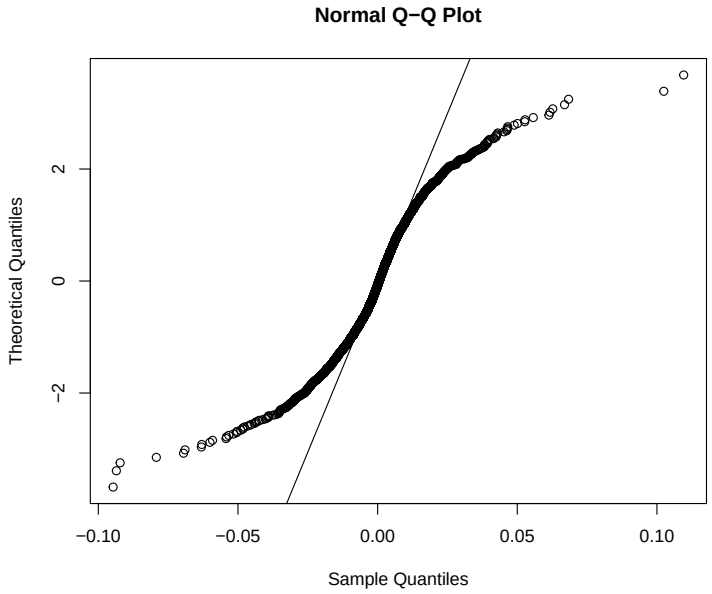
Quantile-Quantile plots

Recall that we found excess kurtosis to be 8, which is quite strong evidence against normality. One can use different tools for assessing the normality of asset returns. *QQ*

plots are used to assess whether the observations come from a particular distribution, or whether two datasets have the same distribution. A normal QQ plot can indicate departures from normality, by means of the comparison between the normal quantiles and the empirical quantiles. Two common examples are skewed data and data with heavy tails (large kurtosis).

The function `qqnorm()` creates a normal Quantile-Quantile plot. More generally, the `qqplot()` function creates a Q-Q plot for any theoretical distribution. The option `datax=TRUE` is used, rather than the default, in order to display sample quantiles on the x-axis, while the y-axis shows the theoretical quantiles of the standard normal; the `qqline()` command allows to add the reference line through the first and third quartiles:

```
qqnorm(rt, datax=TRUE)
qqline(rt, datax=TRUE)
```



We can clearly see the lack of linearity both on the downside and on the upside, showing evidence against the hypothesized normal distribution. In particular, the pattern indicates heavier tails than a normal distribution.

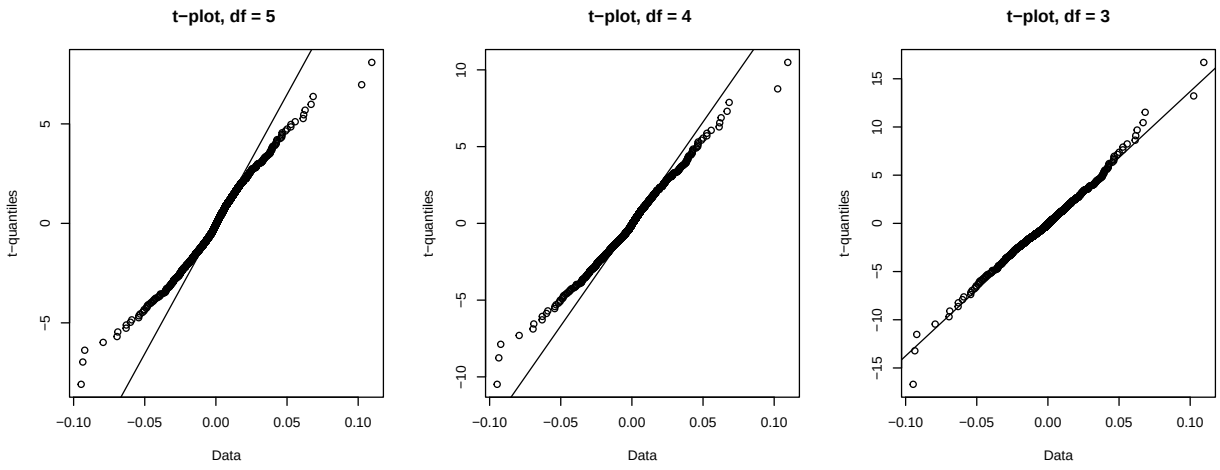
Remark. *One important feature of financial returns is they have 'heavy tails' (or 'fat tails'). This means that there are more extreme outcomes in both tails of the distribution, than would arise if data were normally distributed. The inadequacy of the normal distribution for modelling asset returns has been repeatedly observed in various market data. Many empirical results can be summarized by saying that the distribution of returns tends to be non-Gaussian, sharp peaked and heavy tailed, these properties being more pronounced for intraday values.*

Statistical properties common to a wide variety of markets and instruments, emerging from many empirical studies of financial time series, can be referred to as ***stylized empirical facts****.*

How can we discover how heavy the tails are? Some idea of tail fatness can be obtained by comparing the data with a heavy-tailed distribution. For example, the Student-t has fat tails, where the degrees of freedom indicate how fat the tails actually are.

The following code produces QQ plots where the t-distribution with 3, 4, and 5 degrees of freedom (df) is chosen as the reference distribution:

```
 # t probability plots of daily log returns of S&P 500
s_SPreturn <- sort(rt)
grid <- (1:N)/(N+1)

par(mfrow=c(1,3))
qqplot(s_SPreturn, qt(grid,df=5), main="t-plot, df = 5 ",
        xlab="Data",ylab="t-quantiles")
 lmfit <- lm(qt(c(.25,.75),df=5) ~ quantile(s_SPreturn,c(.25,.75)) )
abline(lmfit)

qqplot(s_SPreturn, qt(grid,df=4), main="t-plot, df = 4 ",
       xlab="Data",ylab="t-quantiles")
lmfit <- lm(qt(c(.25,.75),df=4) ~ quantile(s_SPreturn,c(.25,.75)) )
 abline(lmfit)

qqplot(s_SPreturn, qt(grid,df=3), main="t-plot, df = 3",
        xlab="Data",ylab="t-quantiles")
lmfit <- lm(qt(c(.25,.75),df=3) ~ quantile(s_SPreturn,c(.25,.75)) )
abline(lmfit)
```



The t-plot with 3 degrees of freedom is rather straight, even though there are some points in the left and right tail that deviate from the reference line. Nonetheless, historical data have more extreme outliers than a t-disribution, which does not properly account for a negative return as extreme as on Black Monday in 1987.

Normality Tests

One way of testing for normality is to check if skewness and excess kurtosis are significantly different from zero. A well-known test in this category is the **Jarque-Bera (JB) test**.

To test the null hypothesis of normality, the JB test uses the statistic

$$JB = \frac{(\widehat{sk})^2}{6/N} + \frac{(\widehat{kur} - 3)^2}{24/N}$$

which is distributed as a chi-squared random variable with 2 degrees of freedom using a large-sample approximation, under the null hypothesis. In **R**, the test statistic and its p-value are returned by the `jarque.bera.test()` function. One rejects H_0 of normality if the p-value of the JB statistic is less than the significance level α.

```
jarque.bera.test(rt)
```

```
##
##  Jarque Bera Test
##
## data:  rt
## X-squared = 11887, df = 2, p-value < 2.2e-16
```

An alternative is to use the **Shapiro-Wilk test** of normality, which uses something similar to a normal plot. Specifically, the Shapiro-Wilk test is based on the *association* between the ordered sample and the quantiles of the standard normal distribution. The Shapiro-Wilk test can be implemented using the `shapiro.test()` function.

```
shapiro.test(rt)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  rt
## W = 0.91478, p-value < 2.2e-16
```

For the S&P 500 returns, the Shapiro-Wilk test rejects the null hypothesis of normality with a p-value less than 2.2×10^{-16}.

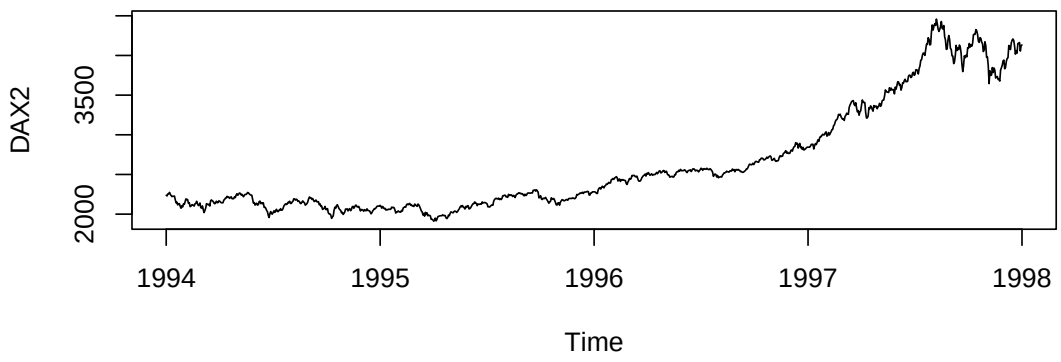
Stylized Facts of Financial Time Series

In the following we use daily closing prices of DAX Stock Inde (1991-1998) available in the the **R**'s EuStockMarkets database. We select observations from 1994 to 1997.

DAX returns

The first task is select the index in the period 1994-1998, and compute the log returns by logarithmic differencing of the original series:

```
#Daily Closing Prices from the EuStockMarkets data set
data(EuStockMarkets)

#select DAX index values in the first column
daxI<-EuStockMarkets[ ,1]
length(daxI)

## [1] 1860

class(daxI)

## [1] "ts"

DAX2<-window(daxI, start=1994, end=1998)
str(DAX2)

##  Time-Series [1:1041] from 1994 to 1998: 2237 2230 2255 2255 2275 ...

plot(DAX2)
```



Log returns are obtained as follows

```
lrt = diff(log(DAX2))
str(lrt)

##  Time-Series [1:1040] from 1994 to 1998: -0.00326 0.01145 0 0.00853 -0.01095 ...
```

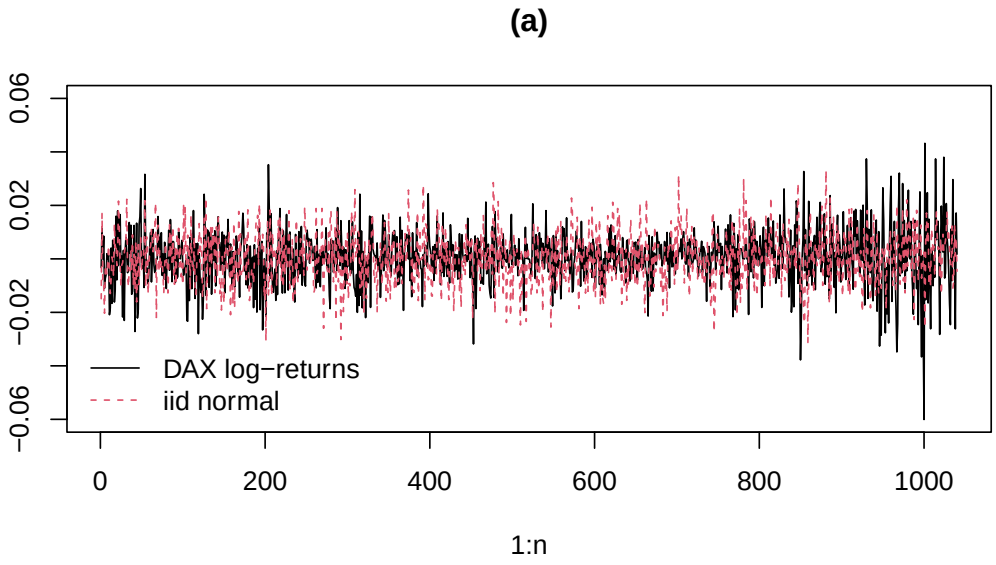
```
head(lrt)

## Time Series:
## Start = c(1994, 2)
## End = c(1994, 7)
## Frequency = 260
## [1] -0.003264282  0.011447400  0.000000000  0.008534438 -0.010949458
## [6] -0.007244437
```

The stylized facts of financial time series are a collection of empirical observations that seem to apply to the majority of daily series of risk-factor changes, such as log-returns on equities and indexes. Some properties are given below.

1. Return series are not *iid* although they show little serial correlation;
2. Series of absolute or squared returns show profound serial correlation;
3. Volatility appears to vary over time;
4. Return series are heavy-tailed.

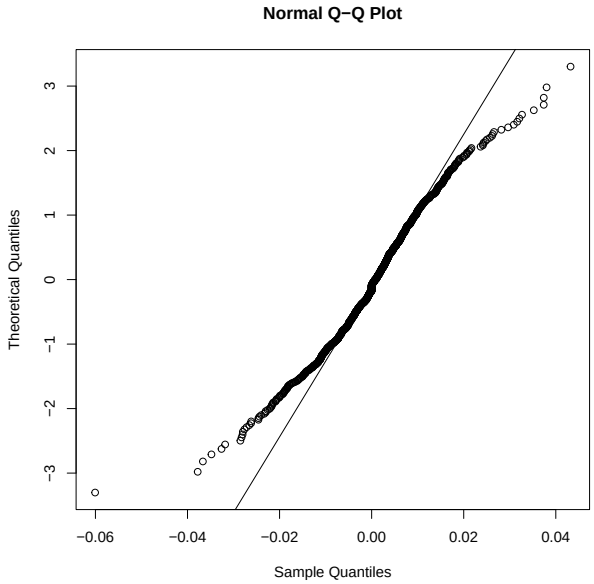
Evidence for the first stylized fact is presented in Figure (a) that shows the daily log-returns for the DAX index from 1995 to 1998 compared to simulated *iid* data from a normal distribution with mean and variance given by the sample mean and sample variance of the returns, respectively.

```
n<-length(lrt)  # n of values in the ts
set.seed(12)    # for reproducible results
Sim <- rnorm(n, mean=mean(lrt), sd=sqrt(var(lrt))) #simulated iid Normal data
plot(1:n, lrt, main="(a)",  type="l", ylab="", ylim=c(-0.06, 0.06))
points(1:n, Sim, type="l", col=2, lty=2)
legend("bottomleft", lty=c(1,2), col=c(1,2), legend=c("DAX log-returns",
            "iid normal"), bty="n")
```



The DAX return data and the normal iid series behave in a different way, and do not show the same range of extreme values, which is not surprising given the inadequacy of the Gaussian model for financial data. Further evidence for this phenomenon is found in the normal QQ-plot obtained via the following code:

```
Y<-as.vector(lrt)
qqnorm(Y, datax=TRUE)
qqline(Y, datax=TRUE)
```



The normal QQ-plot confirms that the normal distribution is a poor model for DAX log-returns. In general, daily financial return data appear to have a much higher kurtosis than is consistent with the normal distribution: their distribution is more narrow in the centre but has longer and heavier tails than the normal distribution.

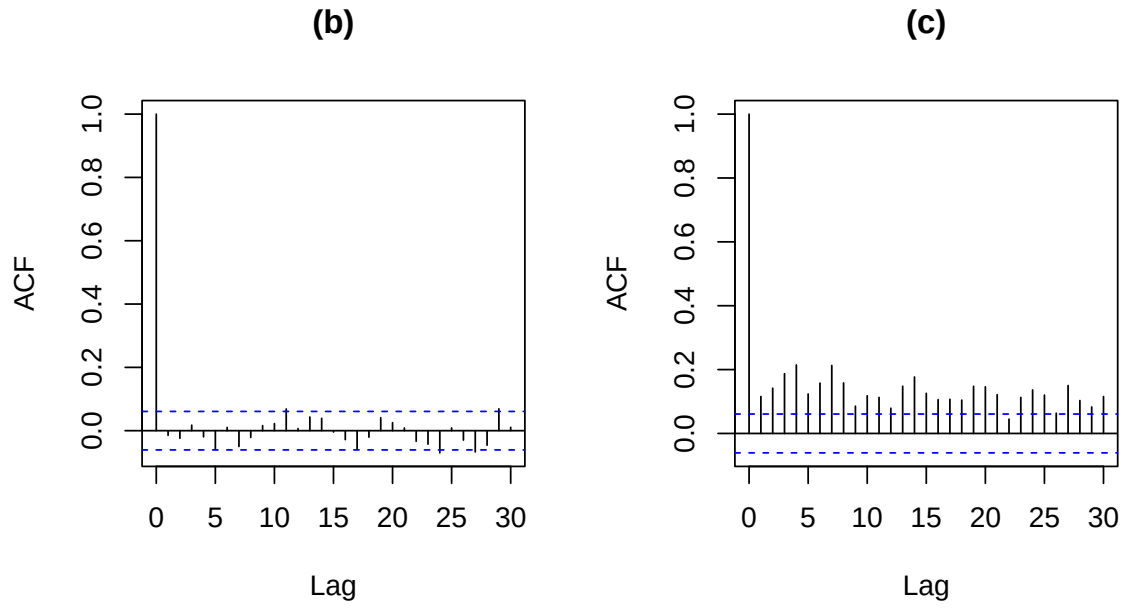
Moreover, the DAX returns exhibit a phenomenon known as **volatility clustering**, which is not present in the simulated series. *Volatility clustering* can be described as the tendency for extreme returns to be followed by other extreme returns, although not necessarily with the same sign. In other words, volatility clusters are present when the magnitudes of the volatilities (the standard deviations of returns) tend to cluster together, so that we observe many days of high volatility, followed by many days of low volatility.

Note that the fact that volatility is very low in some periods does not imply that risk in financial markets was low in those years, since volatility can be low while the tails are fat. For this reason, volatility is a misleading measure of risk.

Volatility clusters and the ACF

Evidence for the first two stylized facts is given in the ACF plot of the raw data (Figure (b)) and the absolute values (Figure (c)) of the data.

```
acf(Y, main="(b)")
acf(abs(Y), main="(c)")
```



From the left plot, we observe that there is very little evidence of serial correlation in the log-returns data, where most autocorrelations lie within the 95% confidence interval.

The lack of serial correlation in the DAX return series suggests that our best guess for tomorrow's return based on our observations up to today is zero, since conditional expected returns are close to zero.

Contrast this with the **ACF of absolute returns** in the right panel, where most of the estimated correlations lie outside the dashed lines. Despite the fact that correlograms of the raw data may show little evidence of serial correlation, it is common to observe evidence of strong serial dependence in the correlograms of the absolute (or squared) returns data.

We can test for the joint significance of autocorrelation coefficients over several lags by using the Ljung-Box test, applied to the raw and absolute values, respectively.

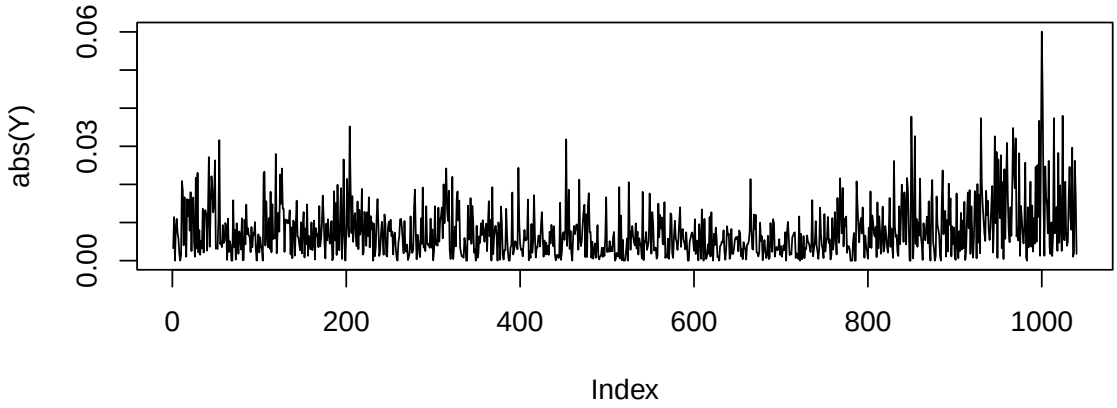
```
#perform Ljung-Box test on DAX log returns
Box.test(Y, lag=20, type="Ljung")

##
##  Box-Ljung test
##
## data:  Y
## X-squared = 25.162, df = 20, p-value = 0.1953
```

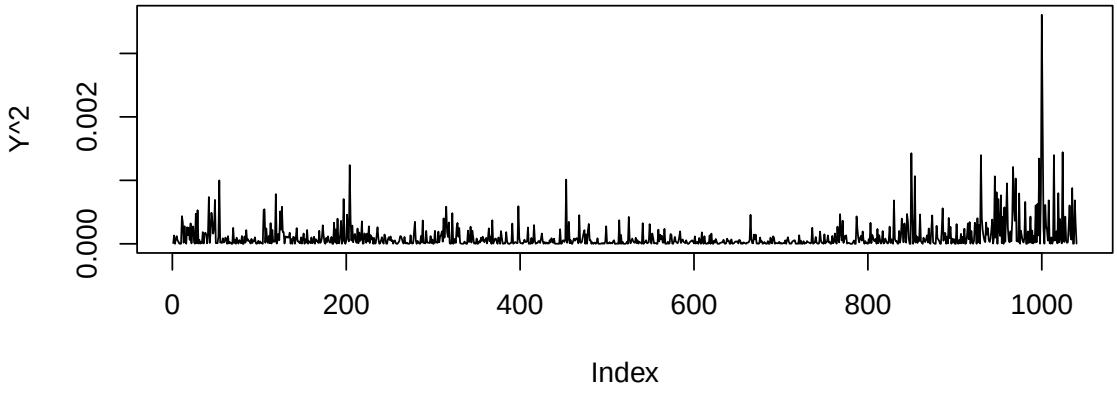
```
#perform Ljung-Box test on absolute log returns
Box.test(abs(Y), lag=20, type="Ljung")

##
##  Box-Ljung test
##
## data:  abs(Y)
## X-squared = 432.35, df = 20, p-value < 2.2e-16
```

Results confirm that the null hypothesis of no serial correlations cannot be rejected when considering DAX returns, while there is strong evidence of serial dependence in the absolute return series. If a series of random variables are *iid* with finite variance (they form a purely random process), then the series of absolute or squared variables must also be iid. Thus, the data support a white noise model but **not** an iid white noise. If log-returns are neither iid nor normal, then this contradicts the popular *normal random walk hypothesis* for log prices.

Remark. *The reason for focusing on absolute or squared returns is that they are proxies for volatilities, which allow us to capture the magnitude of market movements. Volatility is often formally modelled as the standard deviation of financial returns in a given time period, conditional on what happened before, i.e., given historical information. To account for volatility clustering (returns of large magnitude followed on subsequent days by further returns of large magnitude) models for changing volatility (GARCH) may be appropriate.*

```
plot(abs(Y), type="l")    #absolute values for DAX returns
plot(Y^2, type="l")    #squared values for DAX returns
```





EXERCISE.

Consider the FTSE Index (4th column of `EuStockMarkets`) and obtain

a. the series of log-returns and time series plot;

b. the histogram of the data and summary statistics;

c. the normal Q-Q plot and normality test;

d. ACF and Ljung-Box test statistic for examining the null hypothesis of independence at $m = 10$ and $m = 15$ lags.