Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Notes for the WBL-Course

Statistical Analysis of Financial Data

Held in January 2017 at ETH Zurich

Dr. Marcel Dettling

Institute for Data Analysis and Process Design

Zurich University of Applied Sciences

CH-8401 Winterthur

1 INTRODUCTION 1

1.1 EXAMPLES 1
1.1.1 SWISS MARKET INDEX 1
1.1.2 CHF/USD EXCHANGE RATE 2
1.1.3 THE GOOGLE STOCK 3

1.2 WHAT IS A TIME SERIES? 4
1.2.1 THE DEFINITION 4
1.2.2 STATIONARITY 4

1.3 SIMPLE RETURNS AND LOG RETURNS 5

1.4 GOALS IN SAFD 6

2 BASIC MODELS 8

2.1 THE RANDOM WALK 8
2.1.1 SIMULATION EXAMPLE 8
2.1.2 IMPLICATIONS TO PRACTICE 9

2.2 DESCRIPTIVE ANALYSIS OF LOG RETURNS 9

3 DISTRIBUTIONS FOR FINANCIAL DATA 13

3.1 SKEWNESS AND KURTOSIS 13
3.1.1 SKEWNESS 13
3.1.2 KURTOSIS 14

3.2 TESTING NORMALITY 15
3.2.1 JARQUE-BERA TEST 16
3.2.2 ALTERNATIVE TESTS 16

3.3 HEAVY TAILED DISTRIBUTIONS 16
3.3.1 T-DISTRIBUTIONS 17
3.3.2 MIXTURE DISTRIBUTIONS 18

3.4 RANDOM WALK WITH HEAVY TAILS 19

4 VOLATILITY MODELS 22

4.1 ESTIMATING CONDITIONAL MEAN AND VARIANCE 22

4.2 ARCH MODELS 23
4.2.1 DEFINITION AND PROPERTIES OF ARCH(1) 23
4.2.2 SIMULATION EXAMPLE 24
4.2.3 ARCH(P) 27
4.2.4 FITTING ARCH MODELS TO DATA 28

4.3 GARCH MODELS 30
4.3.1 FITTING GARCH MODELS TO DATA 31
4.3.2 GARCH MODEL EXTENSIONS 34

5 RISK MANAGEMENT 36

5.1 VALUE AT RISK 36
5.1.1 EMPIRICAL VAR 37
5.1.2 VAR WITH THE RANDOM WALK MODEL 37
5.1.3 VAR WITH GARCH MODELS 38

5.2 EXPECTED SHORTFALL 40
5.2.1 EMPIRICAL COMPUTATION 40
5.2.2 RANDOM WALK COMPUTATION 41
5.2.3 GARCH COMPUTATION 43

1 Introduction

This course is about the statistical analysis of financial time series. These can, among other sources, stem from individual stocks' prices or stock indices, from foreign exchange rates or interest rates. All these series are subject to random variation. While this offers opportunities for profit, it also bears a serious risk of losing capital.

The aim of this document is to present some basics for dealing with financial time series. We first introduce a statistical notion of financial time series and point out some of their characteristic properties that require special attention. Later, we provide several statistical models for financial data, with a focus on how to fit them and what their implications to everyday practice are. Finally, we lay our attention to measuring the risk of serious loss with an investment.

1.1 Examples

We start out by presenting some financial data. There are various sources from which they can be obtained. While some built-in R datasets will be used throughout this course, others were acquired from non-commercial websites.

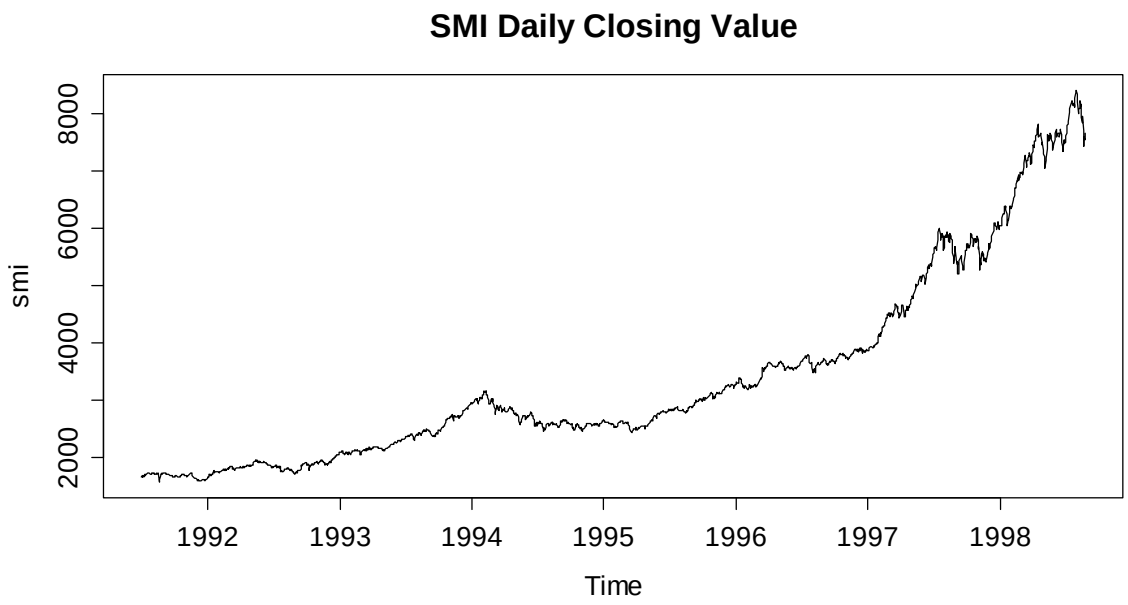
1.1.1 Swiss Market Index

First, we present the SMI series: this is the blue chip index of the Swiss stock market. It summarizes the value of the shares of the 20 most important companies, and contains around 85% of the total capitalization. Daily closing data for 1860 consecutive days from 1991-1998 are available in R:

```
> data(EuStockMarkets)
> EuStockMarkets
Time Series:
Start = c(1991, 130)
End = c(1998, 169)
Frequency = 260
             DAX    SMI    CAC   FTSE
1991.496 1628.75 1678.1 1772.8 2443.6
1991.500 1613.63 1688.5 1750.5 2460.2
1991.504 1606.51 1678.6 1718.0 2448.2
1991.508 1621.04 1684.1 1708.1 2470.4
1991.512 1618.16 1686.6 1723.1 2484.7
1991.515 1610.61 1671.6 1714.3 2466.8
```

As we can see, `EuStockMarkets` is a multiple time series object, which also contains data from the German DAX, the French CAC and UK's FTSE. We will focus on the SMI and thus extract and plot the series:

```
esm <- EuStockMarkets
tmp <- EuStockMarkets[,2]
smi <- ts(tmp, start=start(esm), freq=frequency(esm))
plot(smi, main="SMI Daily Closing Value")
```



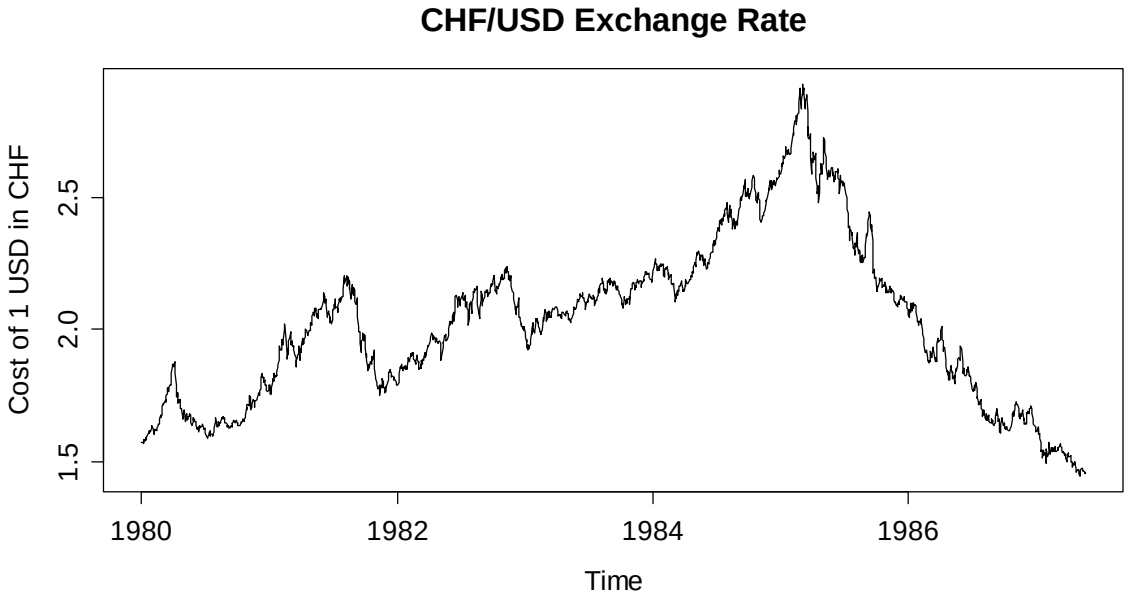
We observe that the series has a trend, i.e. the mean is obviously non-constant over time. This is typical for financial time series. As we will see, such trends in financial time series are nearly impossible to predict, and difficult to characterize mathematically. We will not much embark in this, but try to understand other important aspects of financial time series.

1.1.2 CHF/USD Exchange Rate

The R package `Ecdat` holds examples of many financial and economic time series. Among these, we find the exchange rate between US Dollars and the Swiss Franc for all working days from January 2, 1980 through to May 21, 1987. We choose to display the (for us more familiar) cost of 1 US Dollar in Swiss Francs. For doing so, we need to juggle around the data for a bit:

```
> library(Ecdat)
> data(Garch)
> dat  <- as.Date(as.character(Garch$date), format="%y%m%d")
> chf.usd <- ts(1/Garch$sf)
> plot(dat, chf.usd, type="l", ...)
```

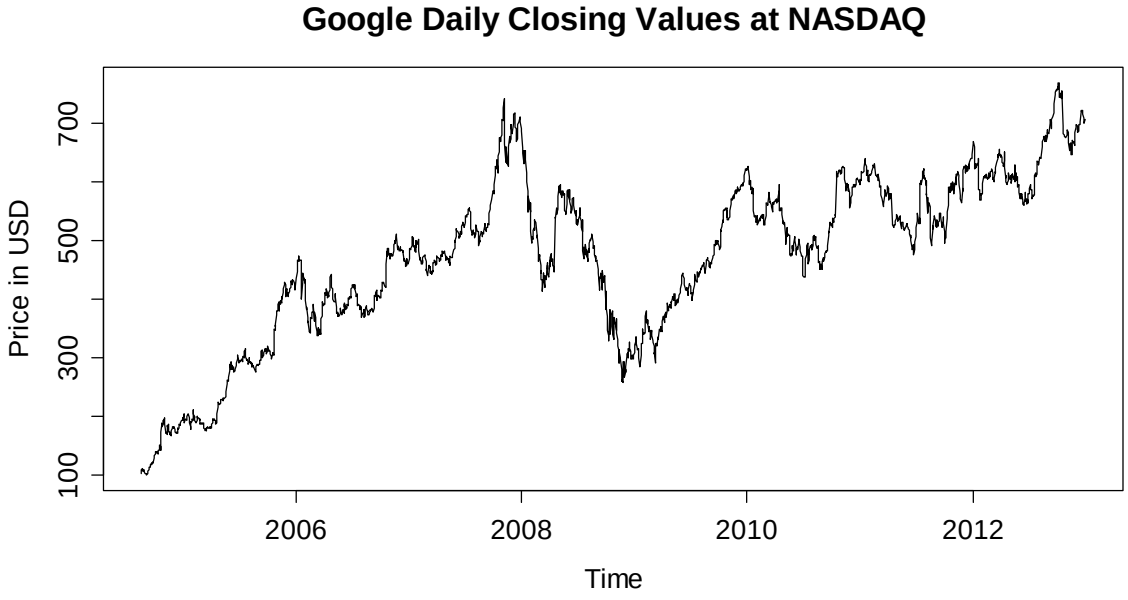
As can be seen on the next page, the mean of this series is obviously non-constant over time; hence it is to be considered as a non-stationary time series. Again, the evolution of the trend will be near-impossible to predict. The best we can do in that situation is to understand the conditional variability of the series, i.e. the day-to-day changes in the CHF/USD rates.



1.1.3 The Google Stock

Finally, we present the daily closing values from each trading day for the Google stock. The data range from the initial public offering (IPO) on August 19, 2004 to December 31, 2012 and include 2107 records. The data are publicly available from the Nasdaq at http://www.nasdaq.com/symbol/goog/historical. After some preprocessing of the Excel sheet by the author, a flat table with the closing prices was generated, which can be plotted in R

```
> plot(google, ylab="Price in USD", ...)
```



As easily visible, an investment in the Google stock at IPO paid off very well.

1.2 What is a Time Series?

1.2.1 The Definition

Throughout this document a time series will be a set of observations that has been collected over a fixed sampling interval. Following a statistical approach, we consider such a series as a realization of a sequence of random variables. A sequence of random variables, defined at such fixed sampling intervals, is sometimes referred to as a *discrete-time stochastic process*, though the shorter names *time series model* or *time series process* are more popular and will mostly be used in this document. It is very important to make the distinction between a time series, i.e. observed values, and a process, i.e. a probabilistic construct.

Definition: A *time series process* is a set of random variables $\{X_t, t \in T\}$, where T is the set of times at which the process was, will or can be observed. We assume that each random variable X_t is distributed according some univariate distribution function F_t. Please note that for this document, we exclusively consider time series processes with equidistant time intervals, as well as real-valued random variables X_t. This allows us to enumerate the set of times, so that we can write $T = \{1, 2, 3, \ldots\}$.

An observed time series, on the other hand, is seen as a realization of the random vector $X = (X_1, X_2, \ldots, X_n)$, and is denoted with small letters $x = (x_1, x_2, \ldots, x_n)$. It is important to note that in a multivariate sense, a time series is only *one single* realization of the n-dimensional random variable X, with its multivariate, n-dimensional distribution function F. As we all know, we cannot do statistics with a single observation. As a way out of this situation, we need to impose some conditions on the joint distribution function F.

1.2.2 Stationarity

The aforementioned condition on the joint distribution F is the concept of *stationarity*. In colloquial language this means that the probabilistic character of the series must not change over time, i.e. that any section of the time series is “typical” for every other section with the same length. More mathematically, we require that for any s, t and k, the observations $x_t, \ldots, x_{t+k}$ could have just as easily occurred at times $s, \ldots, s+k$.

Imposing even more mathematical rigor, we introduce the concept of strict stationarity. A time series is said to be *strictly stationary* if and only if the (k+1)-dimensional joint distribution of $X_t, \ldots, X_{t+k}$ coincides with the joint distribution of $X_s, \ldots, X_{s+k}$ for any combination of indices t, s and k. For the special case of $k = 0$ and $t = s$, this means that the univariate distributions F_t of all X_t are equal. For strictly stationary time series, we can thus leave off the index t on the distribution. As the next step, we will define the moments:

Expectation	μ_t	=	$E[X_t]$,	for stationary series: $\mu_t = \mu$.
Variance	σ_t^2	=	$Var(X_t)$,	for stationary series: $\sigma_t^2 = \sigma^2$.
Covariance	$\gamma(t_1,t_2)$	=	$Cov(X_{t_1}, X_{t_2})$,	for stationary series: $Cov(X_t, X_{t+h}) = \gamma(h)$.

In other words, strictly stationary series have *constant expectation*, *constant variance* , and the covariance, i.e. the *dependency structure, depends only on the lag* h, which is the time difference between the two observations. However, the covariance terms are generally different from 0, and thus, the X_t are usually dependent.

In practice, except for simulation studies, we usually have no explicit knowledge of the latent time series process. Since strict stationarity is defined as a property of the process' joint distributions (all of them), it is impossible to verify from a single realization, i.e. an observed time series. We can, however, always check whether a time series process shows *constant mean and variance*, and whether the *dependency only depends on the lag* h. This much less rigorous property is known as *weak stationarity*.

In order to do well-founded statistical analyses with time series, weak stationarity is a necessary condition. It's obvious that if a series' observations do not have common properties such as constant mean/variance and a stable dependency structure, it will be impossible to statistically learn from it. Unfortunately, asset prices are often non-stationary. The way out is to study the log-returns, an approximation to the relative changes in the series.

1.3 Simple Returns and Log Returns

As we had seen and argued above, financial time series are often non-stationary. Hence, it is difficult to perform statistical analysis on the *prices* (or *index values*, *exchange rates*, *interest rates*), which we will denote by P_t. For a number of reasons (explained below), it is more attractive to study the relative changes in the prices. The natural definition is:

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}}.$$

R_t is called the *simple return* of the asset with price series P_t. However, in statistical analysis of financial data, we usually consider *log returns* r_t, which are defined as:

$$r_t = \log\left(\frac{P_t}{P_{t-1}}\right) = \log(P_t) - \log(P_{t-1}) = \log(1 + R_t)$$

Simple and log returns are not one and the same, although for small relative changes in the price series P_t, they do not differ much. For example, if $R_t = 0.00\%$, then $r_t = 0.00\%$, if $R_t = 1.00\%$, then $r_t = 0.995\%$ and if $R_t = 5.00\%$, then $R_t = 4.88\%$.

But what is the rationale for working with log-returns?

- **Empirical Experience:** price series typically are non-stationary and right-skewed, i.e. have positive spikes larger than negative spikes. In that situation, empirical experience from time series analysis tells us that we should log-transform the series and take first-order differences at lag 1. What we obtain is r_t.

- **Limited Liability:** for many financial assets, the maximum that one can lose is the amount that was put into them. I.e., if one holds a stock that is 49$ worth, the most one can lose is 49$. This is also called *limited liability*. In that case, the simple return would be $R_{\min} = -100\%$. On the other hand, the maximum possible simple return is $R_{\max} = +\infty$. This asymmetry does not exist with log returns: a complete loss yields $r_{\min} = -\infty$, and the maximum is $r_{\max} = +\infty$.

- **Compounding:** to get the simple return of an asset over two periods, a rather complicated and non-intuitive computation is required:

$$R_{2,t} = \frac{P_t - P_{t-2}}{P_{t-2}} = (1 + R_{1,t})(1 + R_{1,t-1}) - 1$$

  Please note that the first subscript is for the horizon of the return computation, and the second is for time. With log returns, calculating multi-period returns is much simpler:

$$r_{2,t} = r_{1,t} + r_{1,t-1},$$

  i.e. log returns are additive, while simple returns are not.

Another very important reason to use log returns is the compatibility with the Random Walk model. This will be discussed in section 2, but first, we study goals and purpose of the statistical analysis of financial data.

1.4 Goals in SAFD

The ultimate dream in the analysis of financial data is an accurate prediction of future prices or returns. This is difficult to realize. Empirical evidence and several economic theories, suggest that:

$$E[r_{t+1} \mid r_t, r_{t-1}, \ldots] \approx E[r_{t+1}] = \mu \approx 0.$$

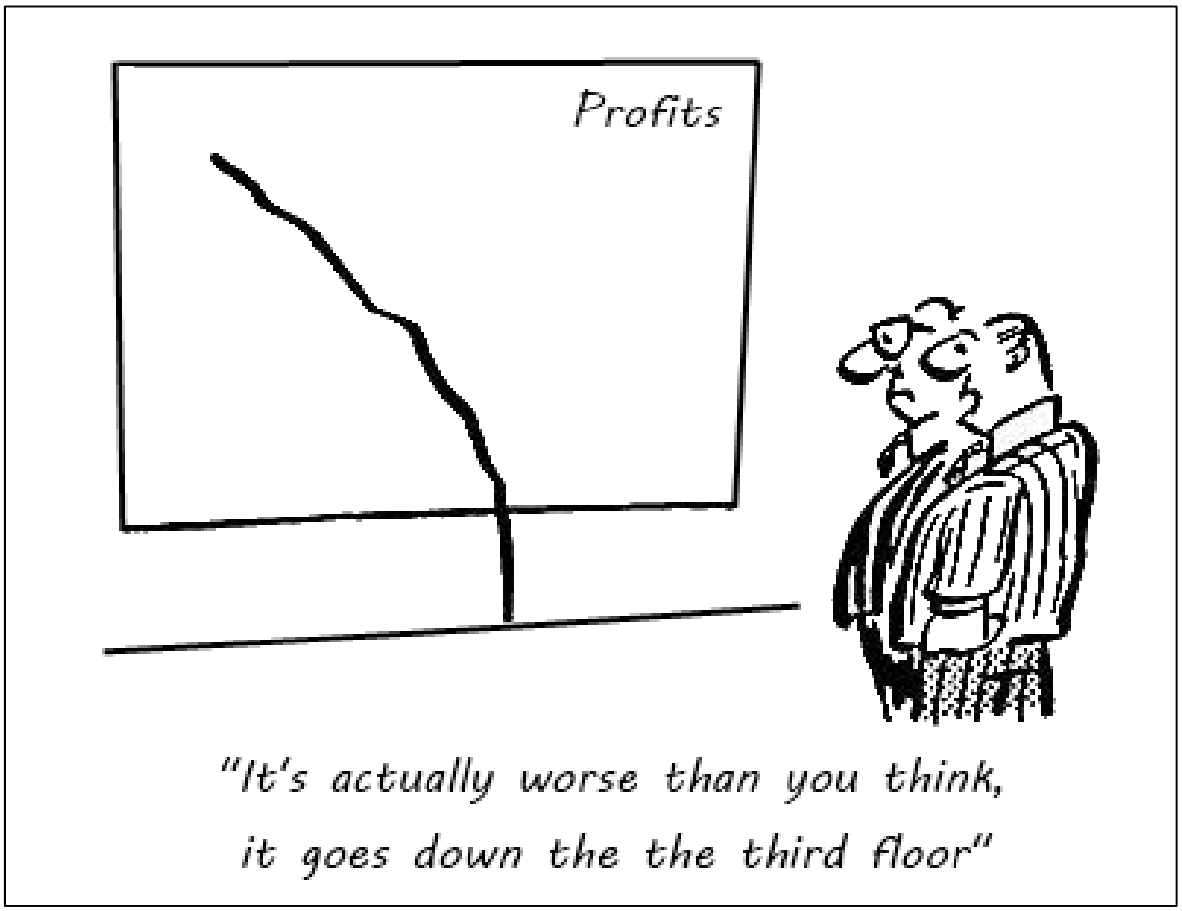
The previous returns do not contain much (if any) information about tomorrow's expected market movements. Thus, we expect a time- and history-independent return of μ. This can be a very small positive value, e.g. due to general economic growth and/or inflation, but it does not tell us whether it is a good time to invest in that financial instrument or not.

What else remains to do for statisticians in the analysis of financial data? Well, future log returns r_{t+k} are random variables. Their expectation does not depend (much) on previous returns, but we still need to understand their distribution. That includes location, scale and shape. The purpose of doing so is specifying which profits and losses will appear, i.e. to attribute scenarios with probabilities. This will make for much of the content of chapters 2 and 3 on Random Walk models.

Finally, it will turn out that while there is hardly any difference between the expectation of the unconditional distribution r_{t+k} and its conditional counterpart $r_{t+k} \mid r_{t-1}, r_{t-2}, \ldots$, other aspects of these two distributions will not be identical. Namely, this is the scale parameter. Financial data exhibit periods of high and low volatility, i.e. the conditional log return distribution can be less or more dispersed, meaning that previous log returns predict how wide the actual distribution needs be. Thus, we discuss different volatility models in section 4.

In summary, the goal in the statistical analysis of financial data is to understand the (time-dependent conditional) distribution of log returns. Practitioners mostly focus on a key indicator characterizing that distribution, i.e. the associated current risk of substantial losses. Two such indicators, in particular *Value-at-Risk* and *Expected Shortfall*, will be discussed in chapter 5.

The scope of this document solely encompasses univariate financial time series, i.e. considers individual assets. When analyzing portfolios with several instruments, several novel aspects will come into play.



2 Basic Models

In this section, we will present some basic models for the analysis of financial data. Historically, the Random Walk plays a special role, as in 1905, it was introduced as the first stochastic model for representing stock prices.

2.1 The Random Walk

The Random Walk model is based on the notion that single-period log returns $r_{1,t}$ are *independent* and *normally distributed*. Applying the compounding property from 1.3 repeatedly, multi-period returns can be expressed as follows:

$$r_{k,t} = r_{1,t} + ... + r_{1,t-k+1}.$$

In verbatim: the k-period return at time t is the sum of all single-period returns back to time $t-k+1$. If one (conveniently) further assumes that the log returns follow a Gaussian distribution, we have $r_{1,t} \sim N(\mu, \sigma^2)$. Because sums of independent normal random variables are themselves normal, it is obvious that

$$r_{k,t} \sim N(k\mu, k\sigma^2)$$

It is now relatively simple to rearrange terms and formulate the random walk model for the price process. It is:

$$\frac{P_t}{P_{t-k}} = \exp(r_{1,t} + ... + r_{1,t-k+1}).$$

By going back to an arbitrary starting point $t = 0$, we have:

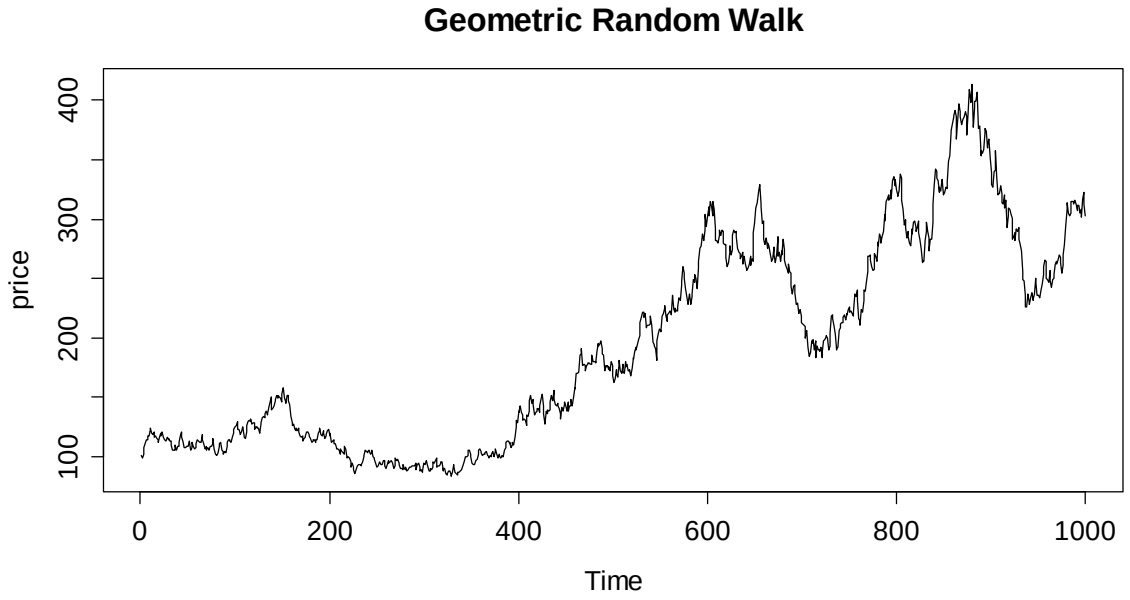
$$P_t = P_0 \cdot \exp(r_{1,t} + ... + r_{1,1})$$

Such a process whose logarithm is a random walk is called a *Geometric Random Walk*. Under the assumption that $r_{1,1}, ..., r_{1,t}$ are iid normal, then the price process P_t follows a lognormal distribution at all times t.

2.1.1 Simulation Example

For illustrating the Random Walk model, we here consider a simulation example. We assume the asset's initial price to be $P_0 = 100$, and $r_{1,t} \sim N(\mu = 0, \sigma^2 = 0.03^2)$. The random variables are drawn in R, and the price process is derived with the following code:

```
> set.seed(23)
> lret  <- rnorm(1000, 0, 0.03)
> price <- 100*exp(cumsum(lret))
> plot(price, type="l", xlab="Time", main="...")
```



As an exercise, using many realizations, one can show that the k-period return is indeed Gaussian with (here) expectation zero and variance $1000 \cdot 0.03^2$, or that the price at time $t = 1000$ follows a lognormal distribution.

2.1.2 Implications to Practice

To conclude this section, we state if the two key assumptions for the Random Walk model, namely independence and normality of the single-period log returns $r_{1,t}$ are met, most of the issues for the statistical analysis of financial data were nearly solved. Unfortunately, as the next section shows, the real world financial data are usually generated by more complex models than the Random Walk.

2.2 Descriptive Analysis of Log Returns

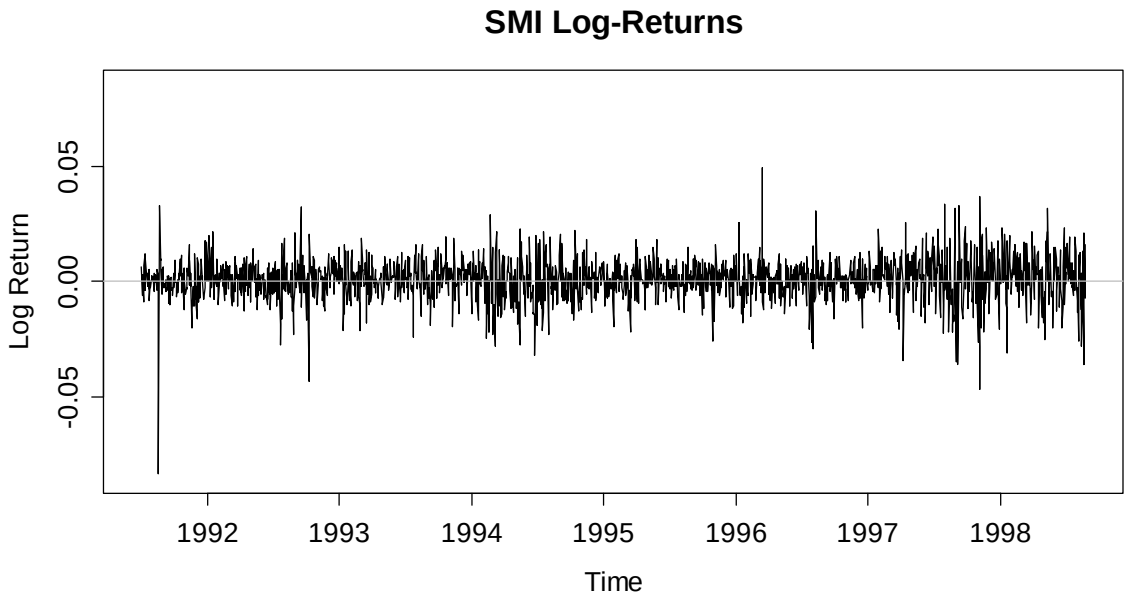
We will now study the empirical properties of single-period log returns, with a special focus on their distribution and potential independence. The standard visualization for log returns is a *time series plot*. The next page shows these for our three example series. The easiest way to generate the log returns in R is to apply the `diff()` function to the logged series. If the prices are appropriately defined as time series objects, R also sets the starting times correctly.

```
> lr.smi    <- diff(log(smi))
> lr.fex    <- diff(log(chf.usd))
> lr.google <- diff(log(google))
```

Please note that the log return series have one record less than the original price series. In particular, r_1 is not available, since we cannot compare to P_0. Often, it proves beneficial to enhance the time series plot with a horizontal line indicating zero. Moreover, because log returns are by construction symmetrical, it makes

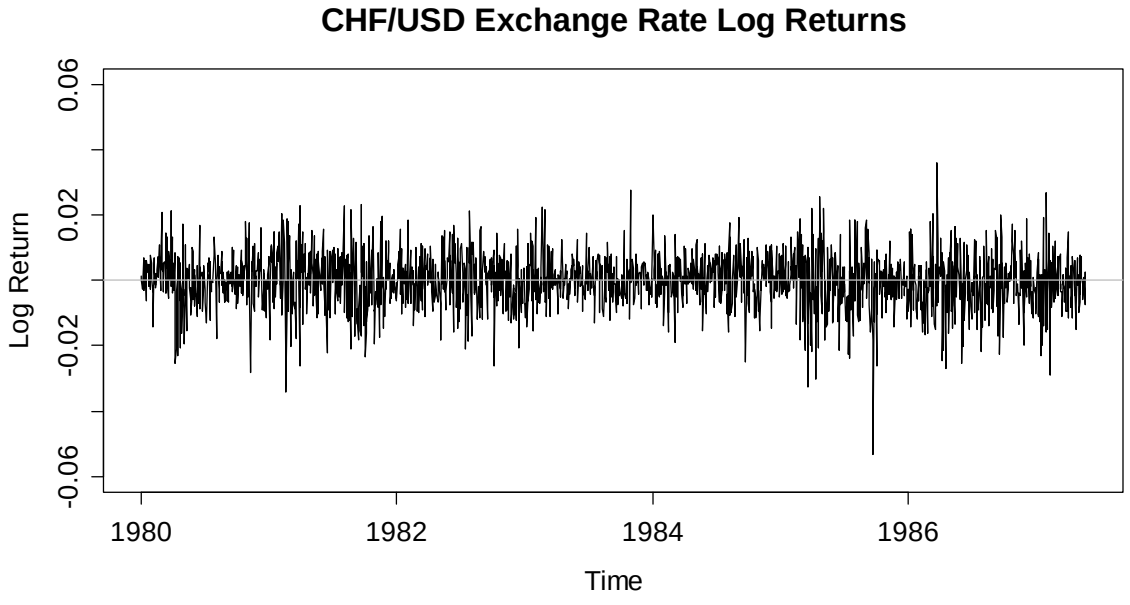
sense to force the y-range to be symmetrical around zero. The code for the SMI log returns is:

```
> plot(lr.smi, ylim=c(-0.085, 0.085), ...)
> abline(h=0, col="grey")
```

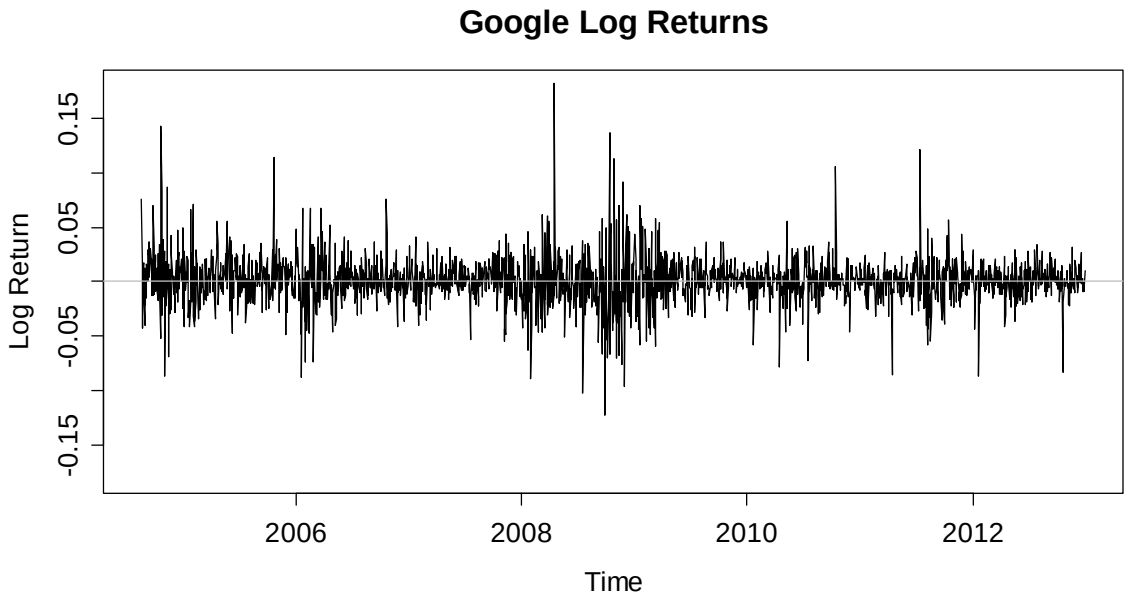


The biggest loss occurred on August 19, 1991, which is the date of the Soviet August Coup, where a group of communist hardliners tried to take control of the government from the reform-friendly Mikhail Gorbachev. Next, we display the log returns of the CHF/USD exchange rate.

```
> plot(lr.fex, ylim=c(-0.06, 0.06), ...)
> abline(h=0, col="grey")
```



```
> plot(lr.google, ylim=c(-0.18, 0.18), ...)
> abline(h=0, col="grey")
```



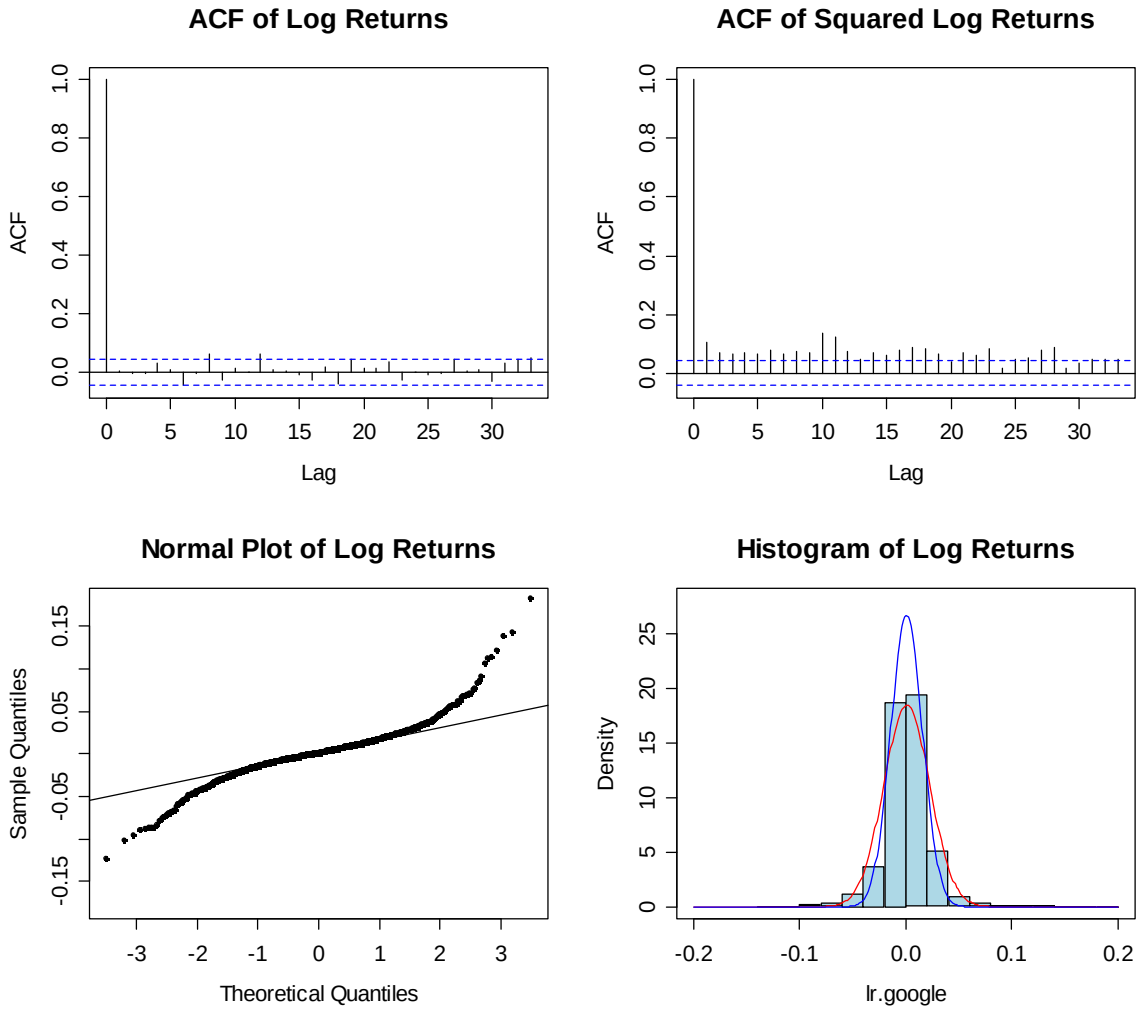
In absolute value, the Google returns show more extreme behavior than the index or the exchange rate. That does not come as a surprise due to the nature of Google, a single (though big) company operating in the rather volatile ICT business. Despite some differences, the three plots show some common features that are very typical for financial data. While perhaps not extremely obvious to a non-expert, a skilled eye clearly detects:

- Nearly uncorrelated log-returns with a mean close to zero.
- Clusters of volatility, i.e. periods where log returns are either big or small
- Some extreme spikes, i.e. outliers that correspond to very big/small returns

We try to better visualize these points by some dedicated plots. First, the autocorrelation function (ACF) of the log returns addresses the issue of uncorrelatedness. Second, the dependency in the conditional variance of the process can be captured by showing the ACF of the squared log returns. In particular, whenever volatility clusters do exist, the squared log returns will show autocorrelation. Finally, histograms and normal quantile-quantile plots serve for verifying the (Gaussian) distributional assumption.

Due to space constraints, we restrict the visualization to the Google shares:

```
> acf(lr.google)
> acf(lr.google^2)
> qqnorm(lr.google)
> qqline(lr.google)
> hist(lr.google, freq=FALSE)
```



We observe that there is hardly any autocorrelation in the log returns. But since the squared instances show clearly significant ACF estimates, the log returns are not independent, which is principally due to volatility clustering. Furthermore, the normal plot clearly shows that the assumption of a Gaussian distribution is off the mark. The log returns are prominently long-tailed – a property which needs to be taken into account for proper modeling of financial time series. Hence, the Gaussian Random Walk cannot be considered as a good model for the type of financial data that we consider.

Another important issue is stationarity: for the prices, it is clearly rejected, but what about the log-returns? In this regard, the long-tailed distribution does not bother; the series' mean seems constant but what about the variance? We will see later that the most powerful notion is to regard log returns as stationary, and employ GARCH type models that allow for dependence in the conditional variance of the series.

3 Distributions for Financial Data

Under the Random Walk model from section 2, by assuming independent Gaussian single-period returns, the distribution of both multi-period returns and the prices could be derived. However, log returns are typically heavy tailed and thus, these results are in question. In this chapter, we will discuss some leptokurtic distributions that are better suited for financial data. Finally, we will also study the Random Walk with heavy-tailed innovations.

3.1 Skewness and Kurtosis

In basic statistics and probability theory, we almost exclusively deal with the first and second central moment of a random variable, namely expectation and variance. The definitions are as follows:

k^{th} moment of X : $m_k = E[X^k]$, e.g. expectation $\mu = E[X]$

k^{th} central moment of X : $\mu_k = E[(X-\mu)^k]$, e.g. variance $Var(X) = E[(X-\mu)^2]$

In the statistical analysis of financial data, or better, in risk management, one is often also interested in the third and fourth central moments, which are the basis for skewness and kurtosis.

3.1.1 Skewness

The third central moment tells us how symmetrical a distribution gathers around its mean. Rather than working with the third central moment directly, it is, by convention, standardized. The definition of *skewness* is as follows:

$$Skew = \frac{E[(X-\mu)^3]}{\sigma^3}.$$

Any random variable with a symmetric distribution will have $Skew = 0$. Values greater than zero indicate positive skewness, i.e. distributions that have a heavy tail on the right hand side. Conversely, $Skew < 0$ indicates a left-skewed distribution.

Let us consider a situation where two investments' return distributions have identical mean and variance, but different skewness parameters. Which one is to prefer? Typically, risk managers are wary of negative skew: in that situation, small gains are the norm, but big losses can occur, carrying the risk of going bankrupt. The *sample skewness* is usually estimated as follows:

$$\hat{Skew} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{\hat{\sigma}}\right)^3$$

In R, several extension packages (e.g. `e1071`, `timeDate`, `TSA`) hold functions for estimating the skewness. We here rely on the one from `timeDate` and use (the default) `method="moment"` for correspondence to the formula given above:

```
> skewness(lr.google)
[1] 0.4340404
attr(,"method")
[1] "moment"
> skewness(lr.smi)
[1] -0.6316853
attr(,"method")
[1] "moment"
> skewness(lr.fex)
[1] -0.3421189
attr(,"method")
[1] "moment"
```

The results confirm what is visible in the plots from section 2.2: the Google log returns are right skewed, the ones of SMI and the CHF/USD exchange rate are left-skewed – though all of them only moderately so. However, it is important to keep in mind that the skewness estimator is (and needs to be) very sensitive to outliers. That is fine as long as the outliers are not "bad" (i.e. wrong) data.

3.1.2 Kurtosis

The *kurtosis* is the standardized fourth central moment. Similar to the variance it measures how spread out a distribution is, but it puts more weight on the tails. The exact definition is:

$$Kurt = \frac{E[(X-\mu)^4]}{\sigma^4}$$

It is important to note that the kurtosis is not very meaningful for skewed distributions, because it will measure both asymmetry and tail weight. Hence, it is an indicator that is aimed at symmetric distributions. Its minimal value is 1, and is achieved for any random variable that only takes two distinct values with probability $1/2$. The normal distribution has $Kurt = 3$; that value is independent of the location and scale parameters μ and σ^2. Due to the popularity of the Gaussian, it is common to compute the *excess kurtosis*, which is simply:

$$Kurt_{Ex} = Kurt - 3.$$

Distributions with heavier tails than the Gaussian, and thus $Kurt_{Ex} > 0$ are called *leptokurtic*. An important example falling into this class is all t-distributions. Their kurtosis depends on the shape parameter, the degrees of freedom ν, i.e.:

$$Kurt(\nu) = 3 + \frac{6}{\nu - 4}$$

This also shows that the maximum value that the kurtosis can take is $+\infty$. In financial analysis, an asset with leptokurtic log returns needs to be taken seriously. It means that big losses (as well as big gains) can occur, and one should be prepared for it. Estimation of the kurtosis happens by:

$$\hat{K}urt = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{(x_i - \bar{x})}{\sigma}\right)^4$$

Implementations for kurtosis estimation can again be found in several extension packages (e.g. `e1071`, `timeDate`, `TSA`). We are using the one from `timeDate`, which by default computes the excess kurtosis:

```
> kurtosis(lr.google)
[1] 7.518994
attr(,"method")
[1] "excess"
> kurtosis(lr.smi)
[1] 5.72665
attr(,"method")
[1] "excess"
> kurtosis(lr.fex)
[1] 1.683527
attr(,"method")
[1] "excess"
```

Again, the estimate is (and needs to be) very sensitive to outliers. That is not problematic as long as they are correct, but false values can have a big impact. We observe that Google has the most heavy tailed log returns, while the changes in the CHF/USD rate show a relatively mild behavior with not much more heavy tails than the Gaussian.

3.2 Testing Normality

The question whether log returns are Gaussian or not is central to the practice of financial data analysis. If yes, and if the log returns are independent, then the Random Walk model from section 2 applies. In that case, understanding the risk that an investment holds is straightforward, and we even know the distributions of the price process. This underlines the importance of verifying if normality holds on financial data.

So far, and usually, normality was/is tested visually, by inspecting time series plots (or much more powerfully) using the normal plot. Also, the above introduced measures skewness and kurtosis can help. In some cases, it may be desirable though to formally test the hypothesis that the data stem from a Gaussian distribution. There is a battery of tests available for this task. We cannot present all of these here, but focus on the *Jarque-Bera test*, that is based on the skewness and kurtosis estimates.

3.2.1 Jarque-Bera Test

The *Jarque-Bera test* of normality compares the sample skewness and kurtosis to 0 and 3, their values under normality. The test statistic is:

$$JB = \frac{n}{24}\left(4 \cdot \hat{S}kew^2 + \hat{K}urt_{Ex}^2\right) \sim \chi_2^2$$

In R, library `tseries` holds an implementation of this test in function `jarque.bera.test()`. We apply it to our 3 example series:

```
> jarque.bera.test(lr.google)
data:  lr.google
X-squared = 5040.39, df = 2, p-value < 2.2e-16

> jarque.bera.test(lr.smi)
data:  lr.smi
X-squared = 2672.383, df = 2, p-value < 2.2e-16

> jarque.bera.test(lr.fex)
data:  lr.fex
X-squared = 258.1409, df = 2, p-value < 2.2e-16
```

Not surprisingly, the null hypothesis of a Gaussian distribution is rejected in all cases. Thus, the Random Walk model with normal increments does not apply here, and risk management decisions based on that approach will be flawed.

3.2.2 Alternative Tests

As mentioned above, there is a number of alternative tests for evaluating normality. We will not discuss any further tests here, but refer to the *Kolmogorov-Smirnov test*, respectively its adaptation, the *Lilliefors test*, and the *Shapiro-Wilk test*. Instructions on how to apply these are found in many textbooks, R implementations are also readily available.

3.3 Heavy Tailed Distributions

We have acquired lots of empirical evidence that the normal distribution is not appropriate for financial returns, because their tails are just too heavy. A closer look shows that the Gaussian probability density function decays with $\exp(-x^2)$ as $x \to \infty$. That is very quickly, and the question is if there are other distributions with different behavior. Not surprisingly, the answer is yes.

We will here consider the *t-distribution*. It is well familiar to the experienced statistician because of its very important role in statistical testing and with confidence intervals. On the other hand it is a popular model for financial data analysis due to its heavy tails, which decay more slowly, with a polynomial rate.

3.3.1 t-Distributions

To construct t-distributed random variables, we need a $Z \sim N(0,1)$ and a $W \sim \chi^2_\nu$. Then, if we take the standardized quotient of the two, the result follows a *t-distribution with ν degrees of freedom*.

$$T = \sqrt{\nu} \cdot \frac{Z}{W} \sim t_\nu .$$

The degrees of freedom ν are a *shape parameter*. The lower they are, the heavier tails result. While in classical statistics ν is a positive integer, it can take any positive value in financial data analysis. The density function of the t_ν-distribution is defined as:

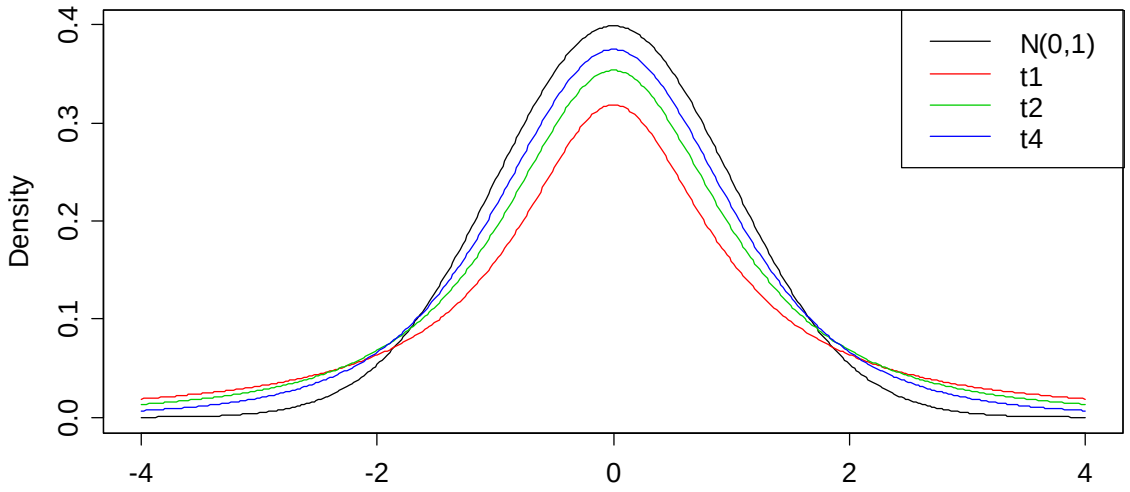
$$f_{t_\nu}(x) = \frac{\Gamma((\nu+1)/2)}{\sqrt{\pi\nu}\cdot\Gamma(\nu/2)} \cdot \frac{1}{(1+(x^2/\nu))^{(\nu+1)/2}}$$

The first term is just a normalizing constant, though quite a complicated one. The symbol $\Gamma(\cdot)$ stands for the *Gamma function* which is defined as:

$$\Gamma(t) = \int_0^{+\infty} x^{t-1} e^{-x} dx \text{ for } t > 0 .$$

From a naïve point of view, i.e. just visually, the difference to the Gaussian bell curve does not seem that big. However, that is deceptive: the variance only exists if $\nu > 2$. In that case, it equals $\nu/(\nu-2)$. The mean only exists if $\nu > 1$, and then takes the value 0. The higher moments require more degrees of freedom for existence, i.e. for the skewness we need $\nu > 3$ and for the kurtosis $\nu > 4$.

The Gaussian and t-Distributions with df=1,2,4



The plot shows that the higher ν, the closer to the Gaussian the t_ν-distribution is. In fact, we have convergence $t_\nu \to N(0,1)$ for $\nu \to \infty$, which is also apparent from the probability density function.

While it seems as if the t_ν-distributions could be very useful for financial analysis because we can adapt to the tail behavior of the data, it is, in its pure form, not a very flexible model. The reason is the absence of a location and/or scale parameter. It is thus attractive and popular to enhance the definition:

If $T \sim t_\nu$, then $S = \mu + \lambda T$ is said to have a $t_\nu(\mu, \lambda^2)$-distribution.

Apparently, μ is the location parameter with $E[S] = \mu$, and λ is the scale parameter with $Var(S) = \lambda^2 \cdot \nu / (\nu - 2)$. The conditions for existence of the moments remain as explained above. No matter whether using the t-distribution in pure or enhanced form, the tail decay is polynomial: more precisely, the density function goes to zero proportional to $x^{-(\nu+1)}$ for $x \to \infty$. Which is, for low ν, a much slower rate than for the Gaussian.

3.3.2 Mixture Distributions

Another class of heavy-tailed models is the set of *mixture distributions*. We here consider a simple example made up of 90% $N(0,1)$ and 10% $N(0,25)$. The density function of such a construct can be written as:

$$f_{mix}(x) = 0.9 \cdot f_{N(0,1)}(x) + 0.1 \cdot f_{N(0,25)}(x).$$

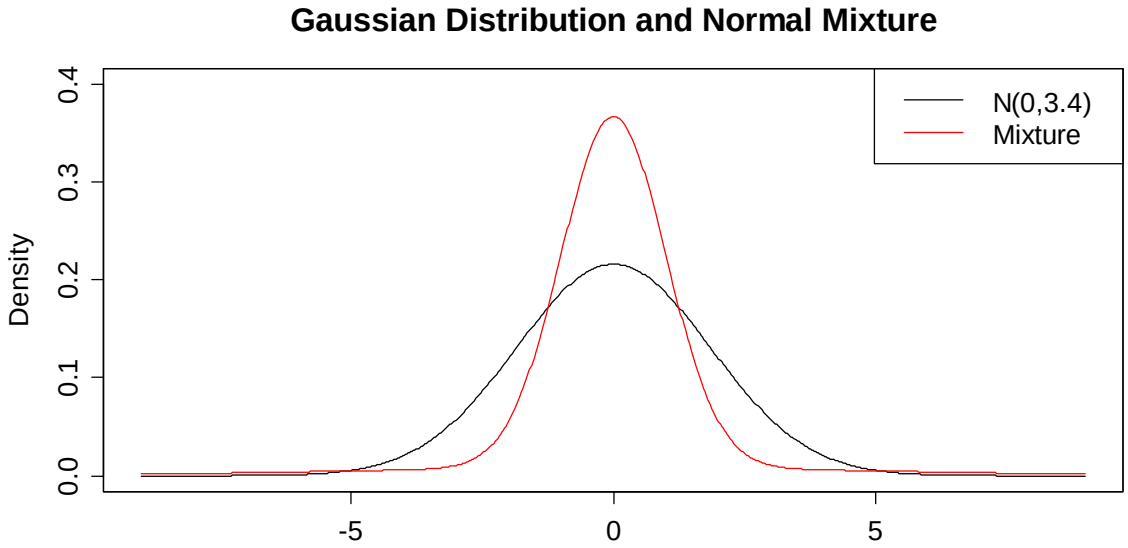
If we need to generate a random variable according to that distribution, we can do that by a two-step process. First, we draw from a $[0,1]$ Uniform distribution. Whenever the result is ≤ 0.9, we use the standard normal, else we draw our final result from the $N(0,25)$. Note that this model could be appropriate for a stock that for most of the time shows little variability, but occasionally, e.g. after some earnings announcement or other events, makes much bigger movements.

What are the consequences for the moments of such a mixture distribution? Due to symmetry, the mean is still zero. For the variance of such a random variable M, we have a linear combination:

$$Var(M) = 0.9 \cdot 1 + 0.1 \cdot 25 = 3.4$$

However, and that is very important to comprehend, the mixture distribution is fundamentally different from a $N(0,3.4)$. It has much more mass in the tails, as can easily be seen from the plot of the density function on the next page.

```
> xx <- seq(-9,9,length=701)
> yy <- dnorm(xx, 0, sqrt(3.4))
> mm <- 0.9*dnorm(xx,0,1) + 0.1*dnorm(xx,0,5)
> plot(xx, yy, type="l", ylim=c(0,0.4))
> lines(xx, mm, col="red")
> title("Gaussian...")
> box()
```



Strikingly, the mixture also has more mass in the center. But since both distributions have equal variance, there must be a much higher chance for outliers, i.e. more mass in the extreme tails, too. Indeed, we can compute the ratio of extreme events by comparing the probability for observations that are further than three standard deviations away from the mean:

```
> sdev  <- sqrt(3.4)
> gauss <- 2*pnorm(-3*sdev,0,sdev)
> mixt  <- 2*0.9*pnorm(-3*sdev,0,1)+2*0.1*pnorm(-3*sdev,0,5)
> mixt/gauss
[1] 9.948061
```

We learn that the mixture distribution produces 10 times more extreme events. This also translates to the kurtosis of M, which takes the value 16.45. Thus, there is a lot of mass in the tails. However, the downside of this relatively simple mixture model is that the large values do not come in sequence, but independently. That is not fully appropriate for real-world financial time series, where extreme returns seem to cluster. We can expect a more realistic behavior from the GARCH models that will be discussed later.

3.4 Random Walk with Heavy Tails

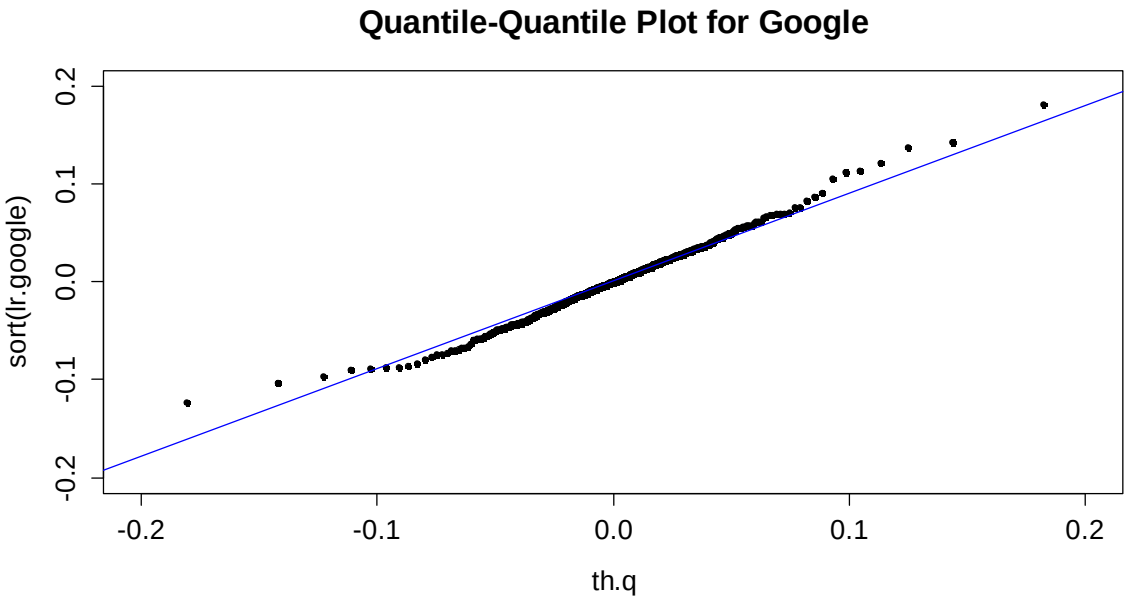
For obtaining a model that reflects the stylized facts of financial data more genuinely, we might be tempted to use a Random Walk with heavy-tailed increments. The natural distributional candidate is the $t_\nu(\mu,\lambda^2)$.

In R, `library(MASS)` has function `fitdistr()`: it offers maximum likelihood estimation for a number of distributions. Among them is the $t_\nu(\mu,\lambda^2)$, thus we can use it for determining the appropriate location, scale and shape parameters of the log returns in our examples. In particular, this works as follows:

```
> fitdistr(lr.google, "t")
        m              s             df
  0.0009455952   0.0133499234   2.9431358498
 (0.0003562337) (0.0003839158) (0.2159852731)
> fitdistr(lr.smi, "t")
        m              s             df
  0.0010642582   0.0069097689   4.5325087792
 (0.0001872645) (0.0001935879) (0.5016358966)
> fitdistr(lr.fex, "t")
        m              s             df
  0.0001331319   0.0071590641   7.1829446212
 (0.0001853914) (0.0001981779) (1.2208518745)
```

The output shows μ,λ and ν. The numbers in parentheses below are standard errors that were obtained from the maximum likelihood approach. Mathematical statistics tells us the estimates are asymptotically normal, and thus we can construct approximate 95% confidence intervals for the parameters by taking the estimate plus/minus twice the standard error. For evaluating how well the model fits to the data, we can do a quantile-quantile plot vs. the estimated distribution. This is more laborious than the normal plot for which the R code already exists.

```
> ## Determine theoretical quantiles
> th.q <- qt(seq(0,1,length=2106+2),2.9431358498)[2:2107]
> th.q <- 0.0009455952+0.0133499234*th.q
> ## Do the QQ-Plot
> plot(th.q, sort(lr.google), ylim=c(-0.2,0.2))
> ## Adding the line
> th.q.75 <- 0.0009455952+0.0133499234*qt(.75,2.9431358498)
> vdiff <- quantile(lr.google,3/4)-quantile(lr.google,1/4)
> slope <- vdiff/(2*th.q.75)
> inter <- quantile(lr.google,3/4)-slope*th.q.75
> abline(inter, slope, col="blue")
```



We observe that the fit is really good, especially in the upper tail. On the left hand side, the smallest true negative returns are not quite as extreme as the model suggests. But still, we judge the model as adequate.

For Gaussian Random Walks, assuming independence of the increments, using the continuous compounding property and the fact that the sum of independent Gaussian random variables still have a normal distribution, it was straightforward to perform multi-period risk management. That is no longer the case if we work with heavy-tailed distributions. Unfortunately, the sum of (even independent) $t_\nu(\mu,\lambda^2)$ random variables belongs to a different distributional family. Thus, there is no alternative than to run Monte Carlo simulations.

Example:

The 5%-quantile of the log return distribution turns out to be:

```
> 0.0009455952+0.0133499234*qt(0.05,2.9431358498)
[1] -0.03072064
```

Thus until tomorrow, with a probability of 95%, we will not lose more than 3.07% of our money if we invest in the Google stock today. What is the respective value for a horizon of the next 20 trading days? There is no closed form solution for that, and we can only resort to computations in R:

```
> set.seed(23)
> res <- c()
> for (i in 1:100000)
+ {
+ lretr  <- rt(20,2.9431358498)
+ lret   <- 0.0009455952+0.0133499234*lretr
+ res[i] <- sum(lret)
+ }
```

What we did is 100’000 runs where for each of the next 20 trading days, a log return was drawn independently from the respective t-distribution. The continuous compounding property can then be used to determine the net return over the next 20 trading days. Finally, we just compute the empirical 5%-quantile of these 100’000 results:

```
> quantile(res, 0.05)
        5%
-0.1454524
```

It turns out that with a probability of 95%, we will not lose more than 14.54% of our money if we invest in the Google stock today. With respect to the distribution that was used, this result is trustworthy. However, it also includes the assumption of independence, which is clearly violated. There are apparent periods of high and low volatility, and we could certainly produce a more exact result if we managed to incorporate these into our model. The volatility models will do so.

4 Volatility Models

Throughout this document, we have collected quite a bit of empirical evidence that financial log returns feature volatility clustering: there are periods where the prices change more substantially, and others where the market is more quiet and the movements are relatively little. Hence, the magnitude of financial log-returns is usually serially correlated. The Random Walk model cannot accommodate for time-varying volatility, neither in its Gaussian formulation nor with heavy-tailed increments. Hence, there is a need for novel approaches. Here, we will present the GARCH class of models.

4.1 Estimating Conditional Mean and Variance

The usual notion of a time series in statistical analysis is:

$$X_t = \mu_t + E_t .$$

Hereby, μ_t is the conditional mean of the series, i.e. $\mu_t = E[X_t \mid X_{t-1}, X_{t-2}, ...]$ and E_t is a disturbance term. In traditional time series analysis, e.g. $AR(p)$-modeling, the disturbance term is usually assumed to be a White Noise innovation, and the conditional mean is expressed as a function of past observations:

$X_t = \alpha_0 + \alpha_1 X_{t-1} + ... + \alpha_p X_{t-p} + E_t$, where E_t is a White Noise innovation.

Under these assumptions, the conditional mean of X_t is non-constant and time-dependent, but the conditional variance is a fixed quantity and equal to the marginal variance. In other words, there is some short-term memory in the mean, but not in the variance.

$$Var(X_t \mid X_{t-1}, X_{t-2}, ...) = Var(E_t) = \sigma_E^2 = const$$

As a conjecture, $AR(p)$-models cannot deal with volatility clustering. Thus, we are seeking an enhanced formulation that allows for non-constant conditional variance. One possibility is to decompose the disturbance term into $E_t = \sigma_t W_t$:

$X_t = \mu_t + \sigma_t W_t$, with $\mu_t = E[X_t \mid X_{t-1}, X_{t-2}, ...]$ and $\sigma_t^2 = Var(X_t \mid X_{t-1}, X_{t-2}, ...)$

Here, μ_t is still the conditional mean, σ_t is the conditional standard deviation and W_t is a White Noise innovation. If we assume that σ_t is a function of previous instances, we obtain a process that also has short-term memory in the variance and hence can be used for volatility modeling. Because σ_t is a (conditional) standard deviation, we need to make sure that the function of previous instances is non-negative. That is cumbersome to achieve with linear combinations, because coefficients restrictions are always awkward. Instead, it is more popular to work with non-linear variance function models. Examples include the ARCH/GARCH class discussed below.

4.2 ARCH Models

ARCH is an acronym meaning *autoregressive conditional heteroskedastic* and stands for models where the conditional variance of a time series is modeled with an autoregressive approach. We assume that we are dealing with a series

$$X_t = \mu + \sigma_t W_t .$$

X_t has time-dependent conditional variance, but the conditional mean $\mu_t = E[X_t \mid X_{t-1}, X_{t-2}, ...]$ equals a constant μ. This reflects the situation of financial log returns well, as we have seen that they usually have little (or better: no significant) direct correlation. Extensions to models that address both conditional mean and variance will be discussed in 4.3.2.

4.2.1 Definition and Properties of ARCH(1)

For simplicity, the formulation of ARCH models is for the disturbance term E_t only. Of course, we can always center a (log return) series $r_t = X_t$ to get rid of a non-zero mean and obtain the disturbance term:

$E_t = X_t - \mu$ for which we assume $E_t = \sigma_t W_t$.

Definition: A series E_t is said to follow a *first-order autoregressive conditional heteroskedastic process*, or short, is $ARCH(1)$, if:

$$E_t = \sigma_t W_t \text{ with } \sigma_t^2 = \alpha_0 + \alpha_1 E_{t-1}^2 .$$

W_t is a White Noise innovation process, with mean zero and unit variance. The two parameters α_0, α_1 are the model coefficients. By construction, the conditional variance of an $ARCH(1)$ process behaves just like an $AR(1)$ model. Later, we will exploit the ACF of squared log returns for deciding the order of ARCH models.

The practical interpretation of an $ARCH(1)$ is straightforward: if a return r_{t-1} was unusually large (in absolute value), then σ_t is larger than usual and thus we also expect a larger-than-usual return r_t. And conversely, unusually small absolute return r_{t-1} implies low σ_t and hence also smaller-than-usual r_t. That way, it is obvious that high/low volatility propagates to $r_{t+1}, r_{t+2}, \ldots$. However, persistence of the volatility is not forever if the stationarity condition of $\alpha_1 < 1$ is met. In that case, the process can always revert back from high to low and from low to high volatility. The unconditional (marginal) variance of E_t turns out to be:

$$Var(E_t) = \frac{\alpha_0}{1-\alpha_1} .$$

Also, the conditional mean and all autocorrelations of $ARCH(1)$ models are zero:

$$E[E_t \mid E_{t-1}, E_{t-2}, ...] = 0 \text{ and } \rho_h = Cor(E_t, E_{t+h}) = 0 .$$

Thus, an $ARCH(1)$ is completely useless for prediction whether a stocks' value will in- or decrease on the following day. It just allows for a more precise understanding of the width (i.e. the dispersion parameter) of the return distribution, and thus for locally adaptive risk management, i.e. better control of the exposition to big losses. If such predictions are accurate, institutional investors are able to make profit from the size of the expected market movements.

4.2.2 Simulation Example

For a deeper comprehension of what $ARCH(1)$ processes are like, we run a simulation study. Therefore, we draw 1100 instances of White Noise with standard normal distribution, i.e. W_t for $t = 1,...,1100$. We set $E_0 = 0$ and then determine:

$$\sigma_t = \sqrt{0.0001 + 0.9 \cdot E_{t-1}^2}$$

$$E_t = \sigma_t W_t ,$$

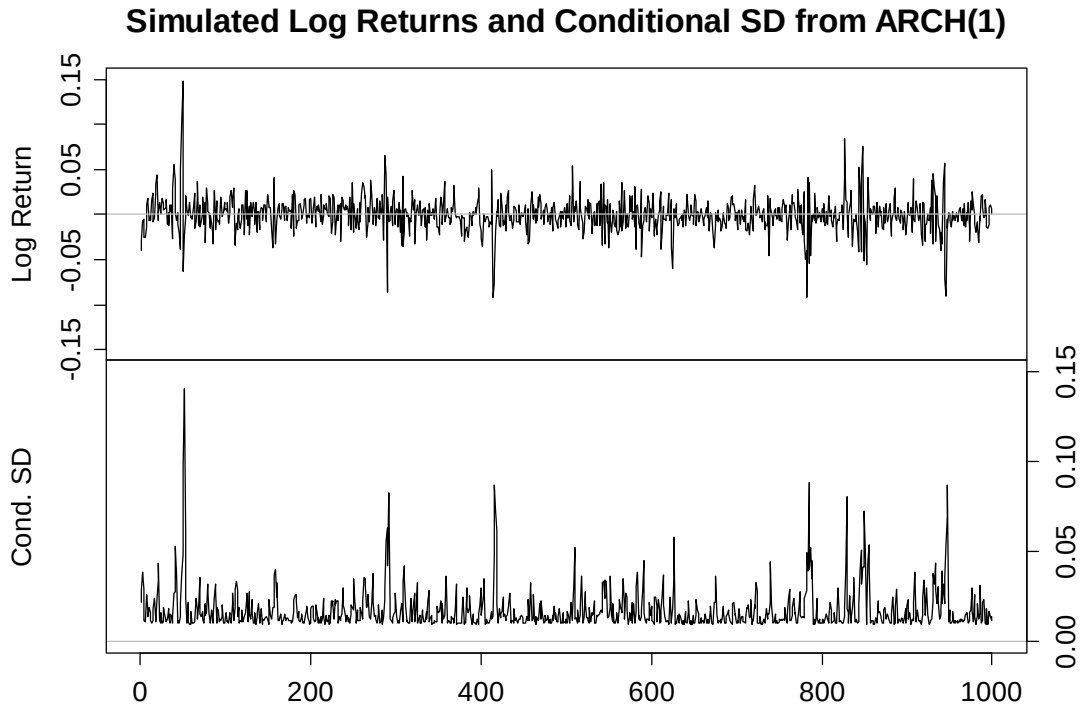
again for all $t = 1,...,1100$. We treat the first 100 observations as a burn-in period into this simulation due to the arbitrary (but realistic) choice of the initial value $E_0 = 0$. Thus, we discard them, so that we are left with a series that has 1000 instances. The goal was to generate a result that coincides with the properties of true log returns. The simulation code is:

```
> set.seed(4659346)
> wn    <- rnorm(1101, 0, 1)
> et    <- st <- c()
> et[1] <- 0
> for (i in 2:1101)
+ {
+   st[i] <- sqrt(0.0001+0.9*et[i-1]^2)
+   et[i] <- st[i]*wn[i]
+ }
```

The results for the $ARCH(1)$ simulated log returns E_t and the conditional standard deviation σ_t are displayed in two time series plots, see next page. We observe that the simulated log returns reflect some of the typical features of real world financial log returns: there are some outliers, and there is volatility, though it does not seem to persist very long after a burst, i.e. the market returns to its normal state in short time. But still, the $ARCH(1)$ process seems like a pretty good generator for log returns. It is also apparent that the conditional standard deviation of the process shows some huge fluctuations. From the setup, it is clear that the minimal value of σ_t is 0.01, but the most extreme value is close to 0.15. On average, as an estimate of the marginal standard deviation, we have:

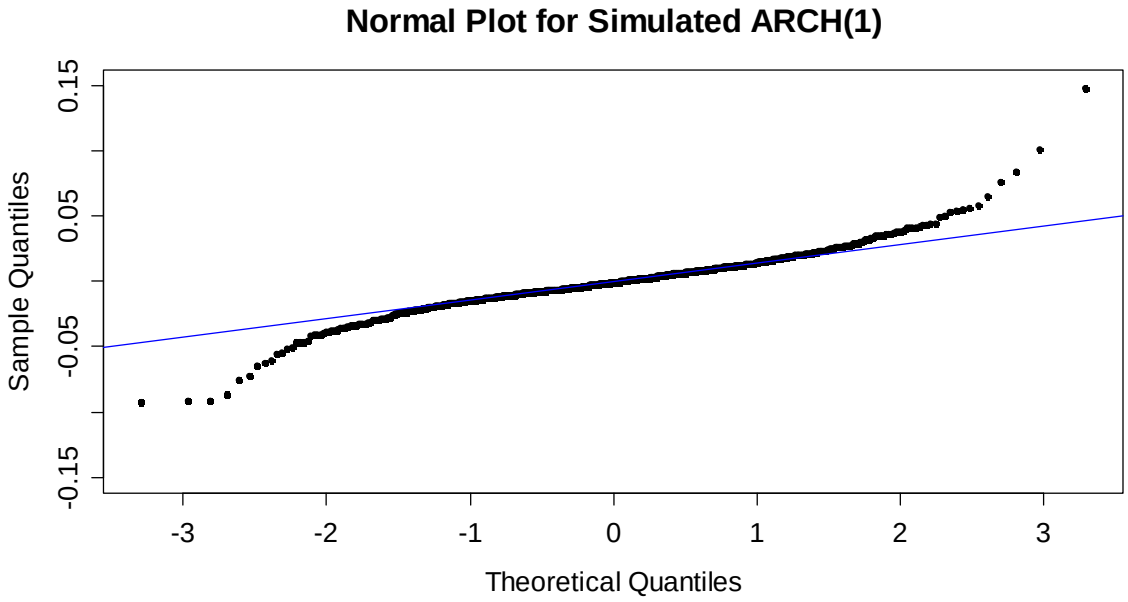
```
> mean(st[102:1101])
[1] 0.01742821
```

This is a value that well matches with true data, e.g. 0.0216 for Google.



We will now put some more emphasis on the marginal distribution of the $ARCH(1)$ process E_t. From the time series plot, we can guess that it is heavy-tailed, but a normal plot is more instructive:

```
> qqnorm(et[102:1101], pch=20, ylim=c(-.15,.15), ...)
> qqline(et[102:1101], col="blue")
```

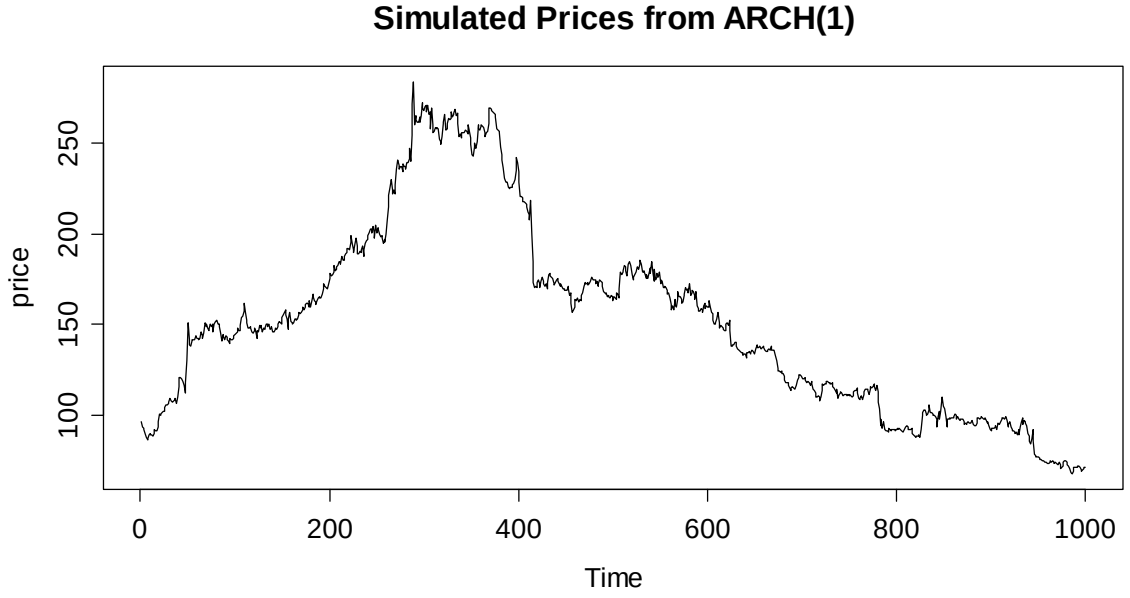


This confirms that despite the use of Gaussian White Noise for the innovation term, the resulting $ARCH(1)$ is heavy-tailed. After some deeper mathematical study, this does not come as a surprise: the marginal distribution of E_t is a mixture

of Gaussian distributions. It is not just two components with fixed coefficients that play a role here, but the mixture is continuous. Nevertheless, what happens is the very same as in the illustrative example from section 3.3.2: we obtain heavy tails.

We can of course use the $ARCH(1)$ simulated log returns to reconstruct the associated price process. We assume an initial value of $P_0 = 100$ and display the prices over the following 1000 time units:

```
> price <- ts(100*exp(cumsum(et[102:1101])))
> plot(price, main="...")
```



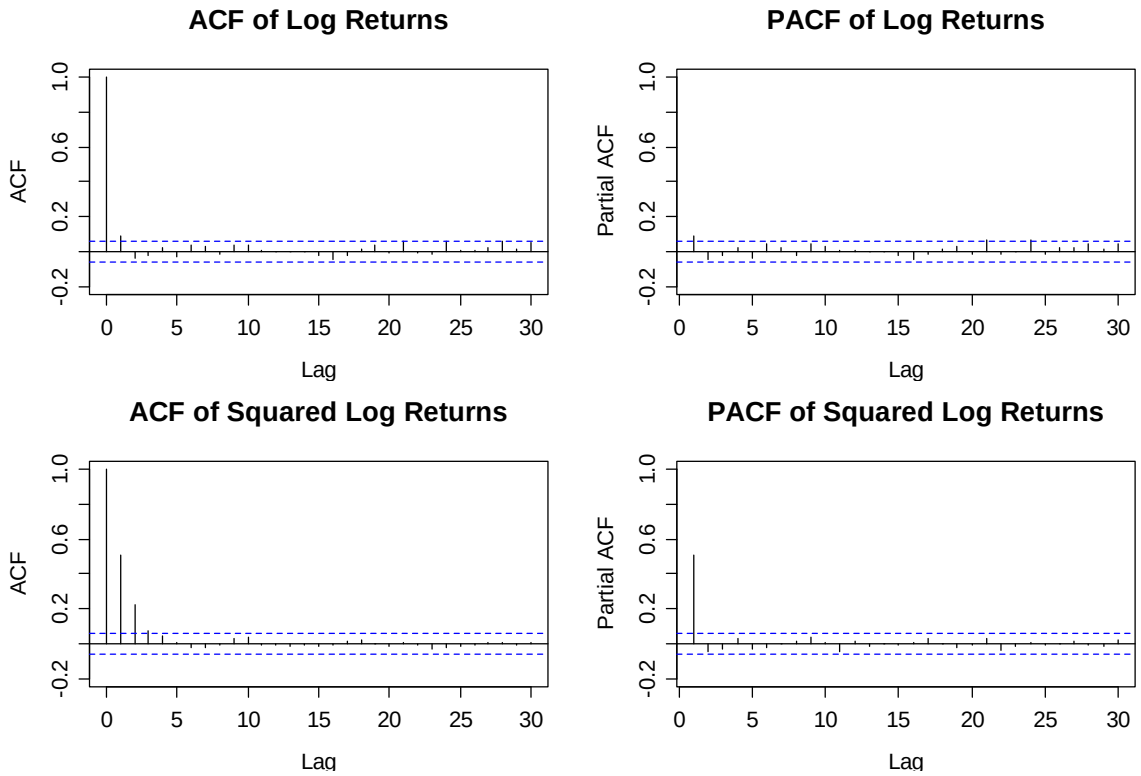
Certainly, we could not dismiss the price trajectory as being totally unrealistic for true data. It is eye-catching though that even after some very big losses or profits, the price adjustments seem to be fairly little. As we will see later, such behavior can better be captured by ARCH models of higher order, or with the Generalized ARCH approach from section 4.3.

However, as a next step, we will verify that the theoretically claimed dependency structure is indeed present on the simulated data and matches what we observed on true log returns. Therefore, we show ACF/PACF for both the straight and squared E_t:

```
> par(mfcol=c(2,2), mar=c(4,4,3.8,2)+0.1)
> acf(et[102:1101], ylim=c(-0.2,1), main="ACF...")
> pacf(et[102:1101], ylim=c(-0.2,1), main="PACF...")
> acf(et[102:1101]^2, ylim=c(-0.2,1), main="ACF...")
> pacf(et[102:1101]^2, ylim=c(-0.2,1), main="PACF...")
```

The result can be seen on the next page and shows that there is hardly any correlation in the straight E_t. Theory says it is exactly nil, the sample estimates of course are not, but mostly fall within the confidence bounds. The situation is

different for E_t^2: we have an exponential decay in the ACF and a clear cut-off at lag 1 in the PACF, which is a typical $AR(1)$ structure.



4.2.3 ARCH(p)

We could now ask the question "what do we do if squared financial log returns show an ACF that resembles the one from an $AR(p)$ with $p>1$?". In fact, we don't know yet, but the answer is very obvious. When fitting an $ARCH(p)$, the conditional variance follows an autoregressive structure with order p. The definition is:

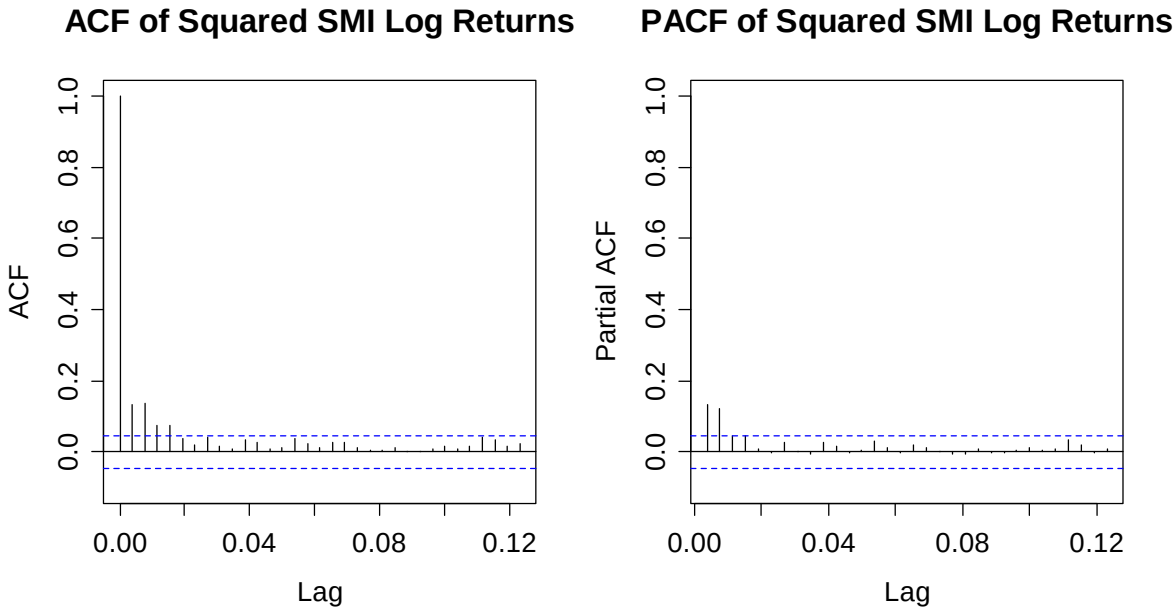
$$E_t = \sigma_t W_t \text{ with } \sigma_t^2 = \alpha_0 + \alpha_1 E_{t-1}^2 + ... + \alpha_p E_{t-p}^2$$

Again, W_t is a (typically Gaussian) White Noise innovation with zero mean and unit variance. The process E_t has mean zero and no correlation, but shows volatility and a heavy-tailed marginal distribution, very much like the $ARCH(1)$ process. The difference is that $ARCH(p)$ can reflect more complex dependency in the conditional variance and by including more terms from the past, also achieves somewhat higher volatility persistence. However, accurate modeling of real-world data usually requires quite large p and thus, many parameters need to be estimated. Often, the $GARCH(p,q)$ models discussed below allow for a more parsimonious representation. However, we will first shed some light on how to fit $ARCH(p)$ models to financial data.

4.2.4 Fitting ARCH Models to Data

The first step with fitting ARCH models is to verify that the ACF and PACF of the squared log returns meet the structure that we would expect from an autoregressive process, i.e. exponential decay in the ACF, and a cut-off in the PACF. If that is plausible, we can determine the order from the cut-off lag p in the partial autocorrelation function.

```
> acf(lr.smi^2, ylim=c(-0.1,1), main="ACF...")
> pacf(lr.smi^2, ylim=c(-0.1,1),main="PACF...")
```



For the squared SMI log returns, up to some random variation, the ACF shows a behavior that is compatible with the exponential decay that theory suggests. The PACF has significant values at lags 1 and 2 but none thereafter, thus a cut-off seems plausible. As a conclusion, fitting an $ARCH(2)$ to these data is a valid choice.

A fitting routine is available with function `garch()` in `library(tseries)`. It works based on the assumption that we have a pure ARCH process $E_t = \sigma_t W_t$ with Gaussian White Noise innovations W_t and mean zero, $\mu = 0$. If the latter is not the case, i.e. if the data are non-centered, we need to estimate and subtract μ from the log returns first. As an alternative to this hand-weaved iterative approach, we can use function `garchFit()` from the `fGarch` package. It allows for simultaneous estimation of the mean and the model parameters. And in-depth discussion of `garchFit()` can be found in section 4.3.1.

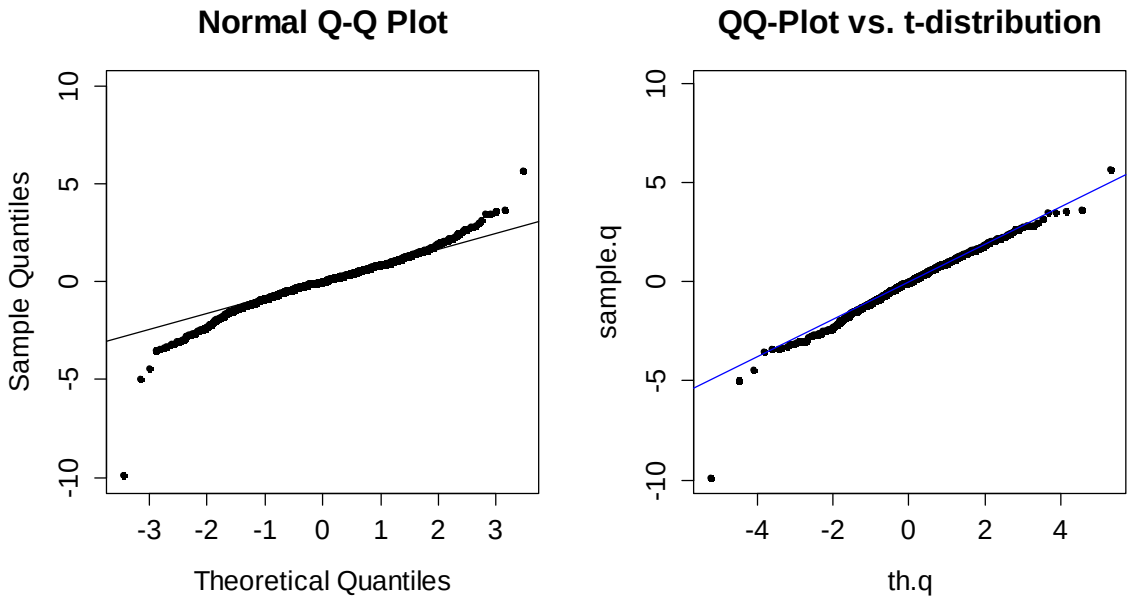
The `garch()` function estimates the coefficients $\alpha_0,...,\alpha_p$ by maximizing the likelihood. This requires numerical optimization, it is thus important to verify that the algorithm converged and not just dismiss potential warning messages. It is also very important to note that the order p stands at second position in the `order=` argument. Hence the command, and the (slighty shortened) output are:

```
> fit <- garch(lr.smi-mean(lr.smi), order=c(0,2), trac=FALSE)
> summary(fit)

Model:
GARCH(0,2)

Residuals:
      Min        1Q    Median        3Q       Max
-9.867029 -0.518727  0.006919  0.584446  5.697691

Coefficient(s):
a0 6.326e-05   1.862e-06   33.976  < 2e-16 ***
a1 1.441e-01   2.392e-02    6.021 1.73e-09 ***
a2 1.203e-01   2.200e-02    5.471 4.48e-08 ***
---
Diagnostic Tests:

Jarque Bera Test
data:  Residuals
X-squared = 3536.448, df = 2, p-value < 2.2e-16

Box-Ljung test
data:  Squared.Residuals
X-squared = 0.183, df = 1, p-value = 0.6688
```

The summary provides the point estimates for α_0, α_1 and α_2, along with asymptotic standard errors. All coefficients are significantly different from zero, but from that alone it is not clear that the model provides a good fit. The diagnostic tests in the second half of the summary aim for answering that latter question. Both these tests are based on examining the residuals. Remember that the ARCH residuals are estimates of the White Noise innovation W_t. Thus, they should neither show dependence, nor volatility nor (here, because `garch()` works under the normal assumption) heavy tails.

Volatility in the residuals would be present if the model was not powerful enough to capture all of what is present in the SMI log returns. The Box-Ljung test evaluates whether the autocorrelation at lag 1 of the squared residuals is significantly different from zero. With a p-value of 0.67, this is clearly not the case. However, the normality of the residuals is strongly rejected. The Jarque-Bera statistic takes a very large value and hence, the model that we fitted here is not fully appropriate. We try to illustrate this by showing QQ-plots versus the Gaussian distribution, and versus a $t_\nu(\mu, \lambda^2)$-distribution with 4.81 degrees of freedom that was fitted to the $ARCH(2)$ residuals by the maximum likelihood based R function `fitdistr()` as discussed in section 3.3.1.

```
> qqnorm(resid(fit))
> qqline(resid(fit))
> qqt(resid(fit))
```



We observe that the residuals are heavy tailed. When doing a QQ-plot versus a $t_\nu(\mu,\lambda^2)$-distribution with, as it turns out, 4.81 degrees of freedom, things look better than when using the Gaussian. Nevertheless, there is an outlier that is far beyond the envelope of even the t-distribution and some left-skewness is also present. The question is if ARCH models can be adjusted to accommodate for that. The answer is yes, as will be shown in the following chapters.

4.3 GARCH Models

What is the difference between an ARCH and a GARCH model? We have seen that with ARCH models, the conditional variance has an $AR(p)$ dependence. With GARCH models, that dependence is generalized to have $ARMA(p,q)$ structure. The model equation for a pure $GARCH(p,q)$ process E_t is thus:

$$E_t = \sigma_t W_t \text{, where } \sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i E_{t-i}^2 + \sum_{i=1}^{q} \beta_i \sigma_{t-i}^2 .$$

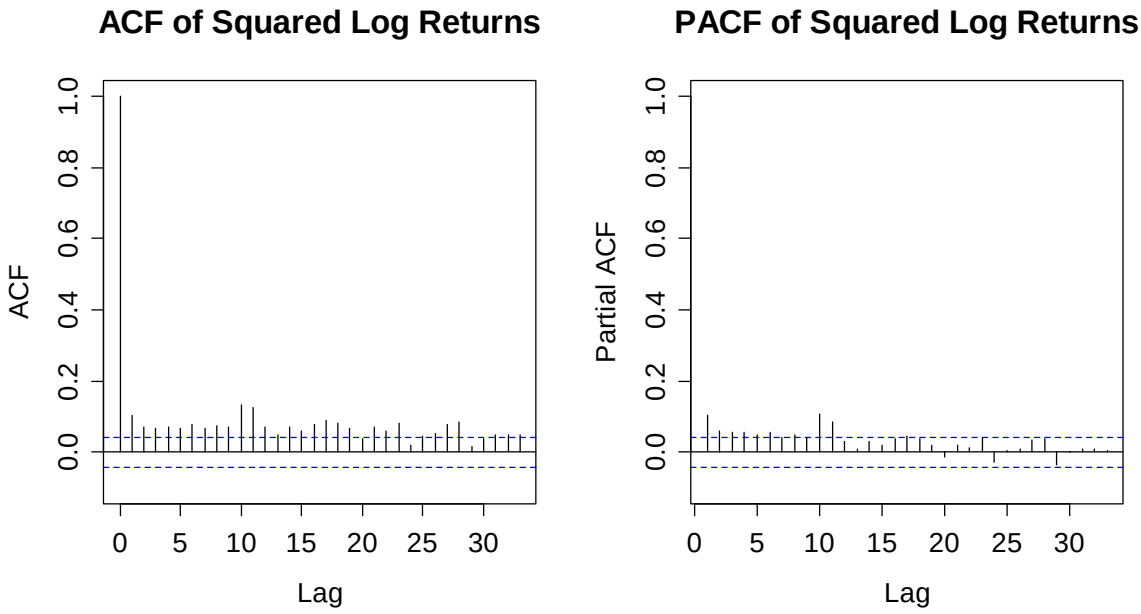
Again, we can also deal with shifted $GARCH(p,q)$ where $X_t = \mu + E_t$. Because past values of σ_t are fed back into the present value, the conditional standard deviation can exhibit more persistent periods of high or low volatility than an ARCH process shows, by spending much fewer parameters, that is! Again, W_t is a White Noise process with zero mean and unit variance. It is standard to assume normal distribution for W_t, but extensions to heavy-tailed and/or skewed situations exist.

The properties of the $GARCH(p,q)$ process E_t include much of which we already know, or correspond to a straightforward transfer from the theory of ARCH models: E_t is uncorrelated, but E_t^2 has ACF and PACF like an $ARMA(p,q)$. Moreover, the marginal distribution of E_t is a mixture of normal distributions and thus heavy-tailed, even if W_t is Gaussian.

4.3.1 Fitting GARCH Models to Data

Our goal with this section is to show another example of GARCH fitting, and to point out an alternative fitting routine that also allows specifying models where the innovation term W_t is from a heavy-tailed, non-Gaussian distribution. Our example will be the Google log returns. We have already seen that they show volatility. The next step is to analyze ACF/PACF of the squared log returns:

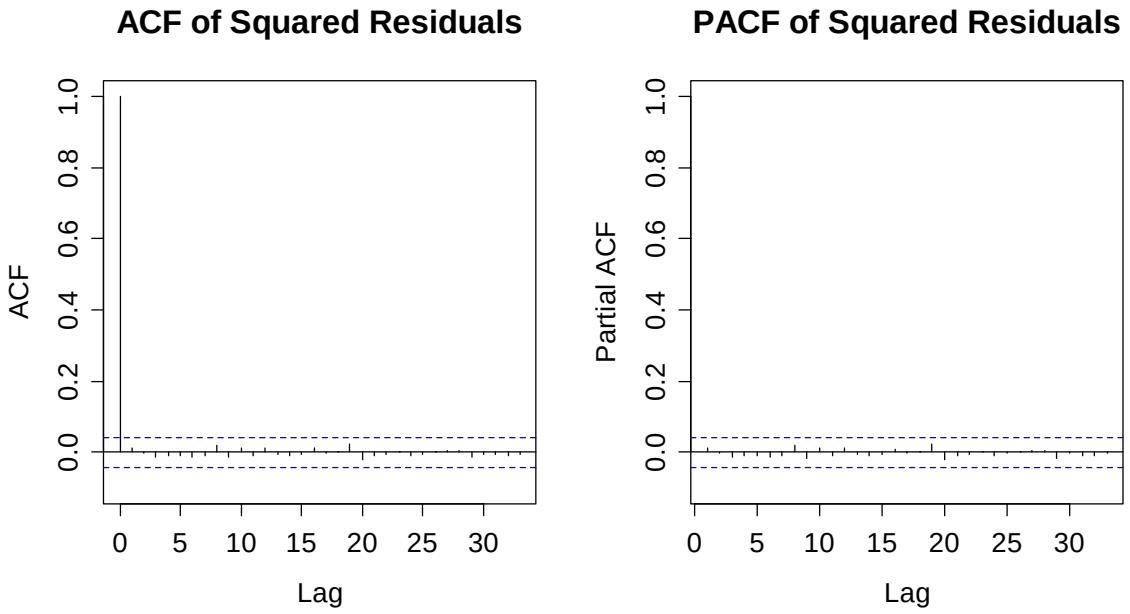
```
> par(mfrow=c(1,2))
> acf(lr.google^2, ylim=c(-0.1,1), main="...")
> pacf(lr.google^2, ylim=c(-0.1,1), main="...")
```



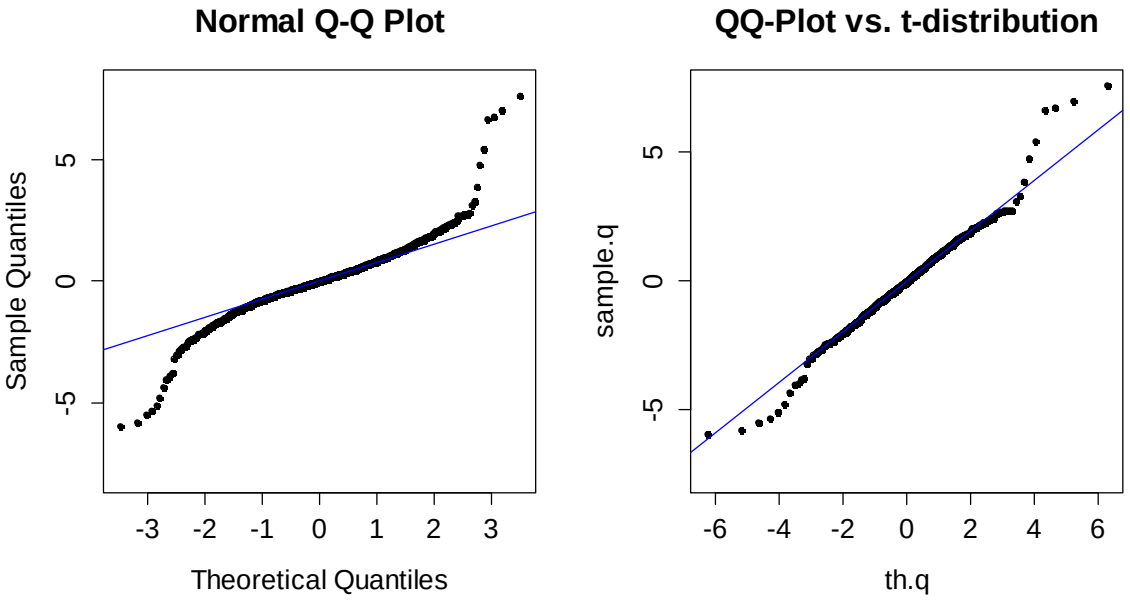
There is a clear dependence present. However, deciding for the correct model order is by no means easy here, because a relatively large number of coefficients significantly differ from zero. What $ARMA(p,q)$ could have generated that ACF and PACF? As it is hard to give an educated guess from the plots, common sense says that we should start with a simple model. A reasonable standard choice is a $GARCH(1,1)$, for which we repeat the fitting process from section 4.2.4.

```
> summary(garch(lr.google-mean(lr.google), order = c(1,1))
Coefficient(s):
    Estimate  Std. Error  t value Pr(>|t|)
a0 7.529e-06   8.914e-07    8.446   <2e-16 ***
a1 4.111e-02   4.085e-03   10.065   <2e-16 ***
b1 9.417e-01   5.293e-03  177.924   <2e-16 ***
---
Diagnostic Tests:
Jarque Bera Test of Residuals
X-squared = 5167.979, df = 2, p-value < 2.2e-16

Box-Ljung test of Squared Residuals
X-squared = 0.2962, df = 1, p-value = 0.586
```

All coefficients are highly significant, suggesting that they are urgently required. The Box-Ljung test has a p-value of 0.586, indicating that there is no "underfit" of the $GARCH(1,1)$. On the other hand, the null hypothesis of Gaussian innovations is strongly rejected with the Jarque-Bera test. We display ACF and PACF of the estimated innovation terms, as well as a Normal Plot of the residuals, plus a QQ-plot where we evaluate against $t_\nu(\mu,\lambda^2)$-distribution with 3.88 degrees of freedom.



None of the estimated autocorrelation coefficients exceeds the confidence bounds. This further emphasizes that the $GARCH(1,1)$ is capable of covering all the volatility there is in the data. However, we have problems with the distribution:



Apparently, the innovations W_t follow a heavy-tailed distribution. With the `garch()` function from `library(tseries)` it is not possible to accommodate

for this. However, there is an alternative. R also offers `library(fGarch)` that has function `garchFit()`. It also allows evaluating models in which the innovation terms follow heavy-tailed distributions. Let's try to do so: the command and the (slightly edited) output are as follows.

```
> gfit.ht <- garchFit(~garch(1,1), lr.google, cond.dis="std")
> summary(fit)

Coefficient(s):
        mu       omega      alpha1       beta1       shape
1.1503e-03  2.5905e-06  3.7093e-02  9.5797e-01  3.9148e+00

Std. Errors: based on Hessian
Error Analysis:
        Estimate  Std. Error  t value Pr(>|t|)
mu     1.150e-03   3.288e-04    3.499 0.000467 ***
omega  2.590e-06   1.186e-06    2.183 0.029014 *
alpha1 3.709e-02   7.139e-03    5.196 2.04e-07 ***
beta1  9.580e-01   7.442e-03  128.718  < 2e-16 ***
shape  3.915e+00   3.330e-01   11.756  < 2e-16 ***
---
Log Likelihood:
5462.3    normalized:  2.593685

Standardised Residuals Tests:
                             Statistic p-Value
 Jarque-Bera Test   R    Chi^2  6062.21   0
 Shapiro-Wilk Test  R    W      0.9215834 0
 Ljung-Box Test     R    Q(10)  13.07772  0.2193591
 Ljung-Box Test     R    Q(15)  17.63791  0.2821800
 Ljung-Box Test     R    Q(20)  24.48442  0.2218716
 Ljung-Box Test     R^2  Q(10)  3.726359  0.9588483
 Ljung-Box Test     R^2  Q(15)  5.358023  0.9886365
 Ljung-Box Test     R^2  Q(20)  7.292896  0.9956080
 LM Arch Test       R    TR^2   4.498502  0.9726726

Information Criterion Statistics:
      AIC       BIC       SIC      HQIC
-5.182621 -5.169201 -5.182632 -5.177706
```

The output is pretty overwhelming! First of all, we note that `garchFit()` allows for simultaneous estimation of μ, of which we make use here. The result is given as `mu` in the output, with a result that is similar (but not equal) to the arithmetic mean in the Google log returns. As it turns out, also the other coefficient estimates are (except for α_0, which is termed `omega` in the output) not very much different from the first fit with the `garch()` procedure (that assumes Gaussian innovations). The shape parameter of the $t_\nu(\mu,\lambda^2)$-distribution is estimated as 3.92, and thus in the region of what we had found previously when we fitted a $t_\nu(\mu,\lambda^2)$-distribution to the residuals from a $GARCH(1,1)$ that was fitted under the assumption of Gaussian innovations.

Furthermore, a battery of tests is carried out, both on the residuals and squared residuals. The Box-Ljung tests show that there is neither autocorrelation nor volatility in the residuals. Thus, the $GARCH(1,1)$ model seems reasonable. Note that the Jarque-Bera still tests whether the innovations are Gaussian. They are not, a matter which we already incorporated into the model by specifying t-distributed innovations. There is no test (in the summary output) which evaluates the adequacy of the chosen distributional model. What we can do is comparing the AIC value under usage of the heavy-tailed innovations versus the use of Gaussian innovations. They are -4.98 and-5.18 for the Gaussian and the heavy-tailed. We take this as further evidence that the $t_\nu(\mu,\lambda^2)$-innovations yield a better fit.

4.3.2 GARCH Model Extensions

There is a number of extensions of the GARCH model that can be fitted with procedure `garchFit()`. We will here give an overview, but keep the section short. First and foremost, a combination of $ARMA(p,q)$ and $GARCH(p,q)$ can be fitted. This is appropriate for time series (i.e. log returns) where we observe volatility, but also significant autocorrelation. The approach would be to let:

$$(r_t =)X_t = \mu_t + \sigma_t W_t$$

Here, μ_t is the conditional expectation, which is not constant anymore. That part of the time dependence in the data is accounted for by an $ARMA(p,q)$:

$$\mu_t = \mu + \sum_{i=1}^{p} \phi_i r_{t-i} + \sum_{i=1}^{q} \theta_i E_{t-i} \text{ , where } E_t = \sigma_t W_t .$$

Since the data are also assumed to have volatility, σ_t is the conditional standard deviation of the data. It is not constant, and its fluctuations are addressed by a $GARCH(p,q)$ model, defined as:

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i E_{t-i}^2 + \sum_{i=1}^{q} \beta_i \sigma_{t-i}^2$$

The typical assumption for financial log returns is (direct) uncorrelatedness and constant conditional mean. That is appropriate for all our examples. Just for illustration, we fit a combined $ARMA(1,1)/GARCH(1,1)$ to the Google log returns.

```
> garchFit(~arma(1,1)+garch(1,1), lr.google, cond.dist="std")
> summary(...)
         Estimate  Std. Error  t value Pr(>|t|)
mu      1.018e-03   3.663e-04    2.778  0.00547 **
ar1     1.106e-01   1.741e-01    0.636  0.52510
ma1    -8.080e-02   1.745e-01   -0.463  0.64323
omega   2.691e-06   1.222e-06    2.202  0.02767 *
alpha1  3.752e-02   7.306e-03    5.136  2.8e-07 ***
beta1   9.573e-01   7.685e-03  124.556  < 2e-16 ***
shape   3.933e+00   3.350e-01   11.741  < 2e-16 ***
```

The previous page shows the most important section from the lengthy summary output. It turns out that neither the AR-parameter ϕ_1 nor the MA-parameter θ_1 are significantly different from zero. That does not come as a surprise, because the tests on the residuals on the previous page had clearly indicated correlation-free residuals.

Finally, `garchFit()` also allows to fit $APARCH$ models. They are based on the notion that large negative returns sometimes seem to increase volatility more than large positive returns of the same magnitude do. The issue is solved with asymmetric power $ARCH$ models, hence the name. We will not discuss these models here more deeply.

5 Risk Management

One of the principal goals in the statistical analysis of financial data is to understand the risk that is associated with an investment. We had argued above that asset prices are non-stationary, so that we must base our considerations on their log returns. As it turned out, they hardly show dependence, thus it is generally considered as impossible to predict whether future returns will be positive or negative. However, there are other important aspects of the return distribution that need to be understood:

- the form (and family) of the distribution
- its location and scale, and also its quantiles
- the timely evolution of the distribution, dependence

For addressing the first two points, we introduced the Random Walk model which operates under assuming independent increments that are either Gaussian or heavy-tailed. In this setup, we can specify the return distribution either analytically (Gaussian case) or at least using Monte Carlo simulations (heavy-tailed case). Doing so opens the door to calculate arbitrary loss or profit quantiles.

Unfortunately, true financial data mostly show volatility. Thus, they are not independent and the Random Walk model can at best be seen as a rough approximation to the truth. We addressed the issue by introducing the $GARCH(p,q)$-technique which allows for modeling the conditional variance. The output can be used to derive the return distribution at a specific time t, but it will change over time. Logically, also loss/profit-quantiles will evolve over time. Here, we take the opportunity to give some formal definitions of the risk management terms that are most widely used in financial analysis. Then, we will also discuss how they need to be implemented in the context of our examples.

5.1 Value at Risk

The most widely used risk measure in financial analysis is the *value-at-risk* (VaR). Without giving an explicit definition, we lived up to it already in previous chapters when we searched for the 5%-quantile of the log return distribution. We define:

$$P[r_{k,t} \leq VaR_{k;(1-\alpha)}] = \alpha$$

It could for example turn out that $VaR_{1;0.95} = -0.038$, which means that with 95% probability the 1-day log return will not be below -3.8%. Or in other words, with a chance of only 5%, we will face a loss that exceeds the $VaR_{1;0.95}$, i.e. -3.8%. As the definition shows, the VaR concept always requires a confidence level $(1-\alpha)$, and a time horizon k. It is typical to set $(1-\alpha) = 0.95$ or 0.99, as well as $k = 1$, but longer horizons are also common. The following examples illustrate the concept.

5.1.1 Empirical VaR

The VaR can be computed in a model-free way from data alone: we simply take the respective empirical quantile of the data. In case of Google, the 1-day empirical 95%-VaR is:

```
> quantile(lr.google, 0.05)
          5%
-0.03134189
```

Thus, on our observed data, we did lose more than -3.13% on maximal 5% of the trading days since the IPO in 2004. While this model-free VaR computation can work quite well with enough data, the disadvantage is that we cannot account for volatility, and it is impossible to determine the VaR of a portfolio of assets. These two cases, which are realistic and important for practice, can only be dealt with when working with at least the Random Walk, or better, with GARCH models.

5.1.2 VaR with the Random Walk Model

The simplest model we discussed was the Gaussian Random Walk. We calculate the $VaR_{1;0.95}$ for the Google stock:

```
> qnorm(0.05, mean(lr.google), sd(lr.google))
[1] -0.03468317
```

We assume the log returns to be independent and Gaussian. Thus, we compute the 5%-quantile of the normal distribution with μ equal to the average return, and σ equal to the sample standard deviation. The 1-day 95%-VaR turns out to be -3.47%. On average, that loss will only be exceeded on every twentieth trading day. For longer horizons k, we derived that the return has a $N(k\mu, k\sigma^2)$ distribution. Thus, the 20-day 95%-VaR is computed as:

```
> qnorm(0.05, 20*mean(lr.google), sqrt(20)*sd(lr.google))
[1] -0.1407081
```

Because the Google log returns are clearly long-tailed, we introduced a heavy-tailed random walk model later in section 3.4. The 1-day 95%-VaR according to that model turned out to be:

```
> 0.0133499234*(qt(0.05,2.9431358498)+0.0009455952)
[1] -0.03165361
```

Thus, the heavy-tailed model is less conservative than the Gaussian Random Walk. That may seem surprising, but since the t-distribution has more mass in the tails, it has less in the center. As mentioned above, the VaR for longer horizons can only be computed using Monte Carlo simulations, because the sum of independent $t_\nu(\mu, \lambda^2)$ random variables is no longer in the same family. We do not repeat the code here, the result was -16.41%.

5.1.3 VaR with GARCH Models

With GARCH models, the VaR concept works almost identically. The exception is that time dependency in the scale parameter of the return distribution transfers to the concept, thus the value at risk obtains an index t, i.e. $VaR_{t;k;0.95}$. The subscripts now indicate time of computation, horizon and confidence level.

1-Day VaR for Gaussian Innovations

Suppose we have n observations of daily log returns $r_1,...,r_n$ and want to compute the 95%-VaR for the next day. For Google, we have $n = 2106$. The current conditional standard deviation is σ_n and can be taken from the n^{th} fitted value of the GARCH model. However, we require the one for the next day, which is $\hat{\sigma}_{n+1|n}$. This can be obtained by a (time series) forecast, which is easy to produce from the `garchFit()` output. We just apply the `predict()` command to the fitted output:

```
> predict(gfit.nd,1)
  meanForecast  meanError standardDeviation
1            0 0.01474702        0.01474702
```

Thus, the predicted log return standard deviation for the next trading day is 0.0147. That is comparably low to the marginal standard deviation of:

```
> sd(lr.google)
[1] 0.02164966
```

However, a closer look at the data (not shown here) proves that at the end of 2012, we are in a period of low volatility. The strong point of using GARCH models is that it can adapt to such behavior. The actual $VaR_{t=2106;1;0.95}$ for the next trading day is computed as the 5%-quantile of the respective Gaussian distribution:

```
> qnorm(0.05, mean=coef(gfit.nd)[1], sd=0.01474702)
[1] -0.02314113
```

We predict not to lose more than 2.314% tomorrow with 95% probability. That is much more optimistic than what we had computed above under independence. Please note that we here use the estimate from the `garchFit()` procedure as a mean for the log return distribution.

k-Day VaR for Gaussian Innovations

For longer VaR-horizon, we produce long-range forecasts of the conditional standard deviation. These are obtained by adjusting argument `n.ahead=` and eventually converge to the unconditional standard deviation of the log returns. The 20-day-return distribution is again a Gaussian, obtained by addition of 20 Gaussian random variables with individual standard deviations but identical mean:

```
> csd <- predict(fit, n.ahead=20)$standardDeviation
> qnorm(0.05, mean=20*coef(gfit.nd)[1], sd=sqrt(sum(csd^2)))
[1] -0.09394989
```

The $VaR_{t=2106;k=20;0.95}$ for a 20-day-horizon according to the Gaussian-$GARCH(1,1)$ is only -9.39%. This is quite a bit lower than the the -14.07% we had computed from the Gaussian Random Walk, owing to the fact that we are in a low volatility period.

1-Day VaR for Heavy-Tailed Innovations

As we had seen above, there is little credibility for a $GARCH(1,1)$ with Gaussian innovations. Instead, they seem long-tailed, and we should use that better model for risk management. The procedure is not too different, we again create a forecast of $\hat{\sigma}_{n+1|n}$.

```
> predict(gfit.ht, n.ahead=1)
  meanForecast  meanError standardDeviation
1  0.001150291 0.01376373        0.01376373
```

We have to convert that result to a forecast for the conditional scale parameter of the $t_\nu(\mu,\lambda^2)$-distribution via:

$$\hat{\lambda}_{n+1|n} = \sqrt{(\hat{\nu}-2)/\hat{\nu}} \cdot \hat{\sigma}_{n+1|n} = 0.00963$$

The degrees of freedom can be taken from the $GARCH(1,1)$ output and are 3.915. Thus, tomorrows return follows a $t_{3.915}(0.0012, 0.00963^2)$-distribution. The mean was again taken from the `garchFit()` procedure. The 1-day 95%-VaR is:

```
> 0.0012+0.00963*qt(0.05,3.915)
[1] -0.01945842
```

The numerical result is -1.95%, which is less conservative than the $GARCH(1,1)$ results of -2.31% under Gaussian distribution. We made a very similar observation when comparing the Random Walk results. Again, the reason is that the leptokurtic distribution has heavier tails and thus more (very) extreme events. In contrast, this also means that there is more mass in the center of the distribution, which leads to a less negative 5%-quantile of the distribution.

k-Day VaR for Heavy-Tailed Innovations

For computing a multi-period VaR from the $GARCH(1,1)$ model with heavy-tailed innovations, we again need to run Monte Carlo simulations. The basis is to determine the (conditional) return distribution, very much like above, for each of the next 20 trading days. However, since the sum of these distributions is no longer a $t_\nu(\mu,\lambda^2)$, there is no way around drawing random variates of the respective distributions and determine the resulting 20-day-return.

```
> res  <- c()
> lamb <- sqrt((cf[5]-2)/cf[5])*predict(gfit.ht,20)[,3]
> for (i in 1:100000){
+   lretr   <- rt(20,cf[5])
+   lret    <- cf[1]+(lamb*lretr)
+   res[i]  <- sum(lret)
+ }
```

The process is repeated many times, and so the Monte Carlo distribution of the 20-day-return is obtained. Its 5%-quantile is the VaR that we are looking for. The result lies in the range of -8.1% and is again less conservative than the result under the assumption of Gaussian innovations.

5.2 Expected Shortfall

The VaR is an intuitive way of measuring the risk associated with an investment. However, there are other risk measures, too. To clarify the concept, theoreticians have described a number of properties that a risk measure might or might not have. These include monotonicity, sub-additivity, homogeneity and translational invariance. As it turns out, the VaR is not sub-additive and hence one can show that it discourages diversification when portfolios of several assets are constructed. To overcome this drawback, better risk measures have been suggested. We here only discuss one alternative, and that is the *expected shortfall*, abbreviated as ES. It is defined as:

$$ES_{t;k;(1-a)} = E[r_{k,t} \mid r_{k,t} < VaR_{t;k;(1-\alpha)}]$$

In words, it is the expected loss given that the return violates the VaR. Of course, this typically depends on the time t, the horizon k and the level $(1-\alpha)$. Again, the most typical case is the 1-day 95%-ES.

5.2.1 Empirical Computation

Computing the ES in a model-free environment is straightforward. For a 1-day horizon $k=1$, we take all returns that are below the VaR and average these:

```
> mean(lr.google[lr.google<quantile(lr.google,0.05)])
[1] -0.04978501
```

As it turns out, 106 of the 2106 log returns are below the empirical 95%-VaR. The average of these is -4.98% - that is our empirical estimate of the 1-day 95%-ES. While this is by no means required, one can also use a pre-existing R function from `library(PerformanceAnalytics)`:

```
> ES(lr.google, p=0.95, method="historical")
          [,1]
ES -0.04978501
```

Empirical computation of the 20-day ES is possible, but requires attention. When we use empirical 20-day returns, we have to make sure that they are not overlapping. This reduces the number of observations, so that only 100 are left:

```
> lr20.tmp <- diff(log(google), lag=20)
> lr20     <- lr20.tmp[seq(1,2087,by=21)]
> mean(lr20[lr20<=quantile(lr20,0.05)])
[1] -0.1866832
```

5.2.2 Random Walk Computation

The basis for model-based estimation of the ES is two formulae that show how it is computed in case of $N(\mu,\sigma^2)$ and $t_\nu(\mu,\lambda^2)$-distribution. In case of the normal distribution, the general formula for the $(1-\alpha)\%$-ES is:

$$ES = \mu - \sigma \cdot \frac{f(\Phi^{-1}(\alpha))}{\alpha},$$

where $\alpha = 0.05$, $f(\cdot)$ is the density and $\Phi^{-1}(\cdot)$ is the quantile function of the standard normal distribution. For example, when computing the 95%-ES

$$\frac{f(\Phi^{-1}(0.05))}{0.05} = 2.063.$$

Thus, the ES is always a bit more than two standard deviations below the mean. Please also note that the 95%-VaR is at -1.645 standard deviations below the mean. Thus, when working with the Gaussian distribution, it does not make a difference which of the two risk measures is employed. However, that is not the case for all distributional models. Of course, the mean μ and the standard deviation σ need to be determined accordingly with respect to the the horizon k. By assuming the Random Walk model, there is no volatility, and hence the ES will be time-invariant. For the 1-day 95%-ES in case of Google, we obtain:

```
> mean(lr.google)-sd(lr.google)*dnorm(qnorm(0.05))/0.05
[1] -0.04372968
```

Again, there is a dedicated R function that does the job. It is again `ES()`, only the `method="gaussian"` needs to be set.

```
> ES(lr.google, p=0.95, method="gaussian")
          [,1]
ES -0.04371908
```

There is a slight, practically irrelevant difference in the numerical result which is due to some rounding errors. When assuming the Gaussian Random Walk model, computing the ES for a multi-period horizon is straightforward, since we can make use of the compounding property:

```
> 20*mean(lr.google)-sqrt(20)*sd(lr.google)*2.063
[1] -0.1811931
```

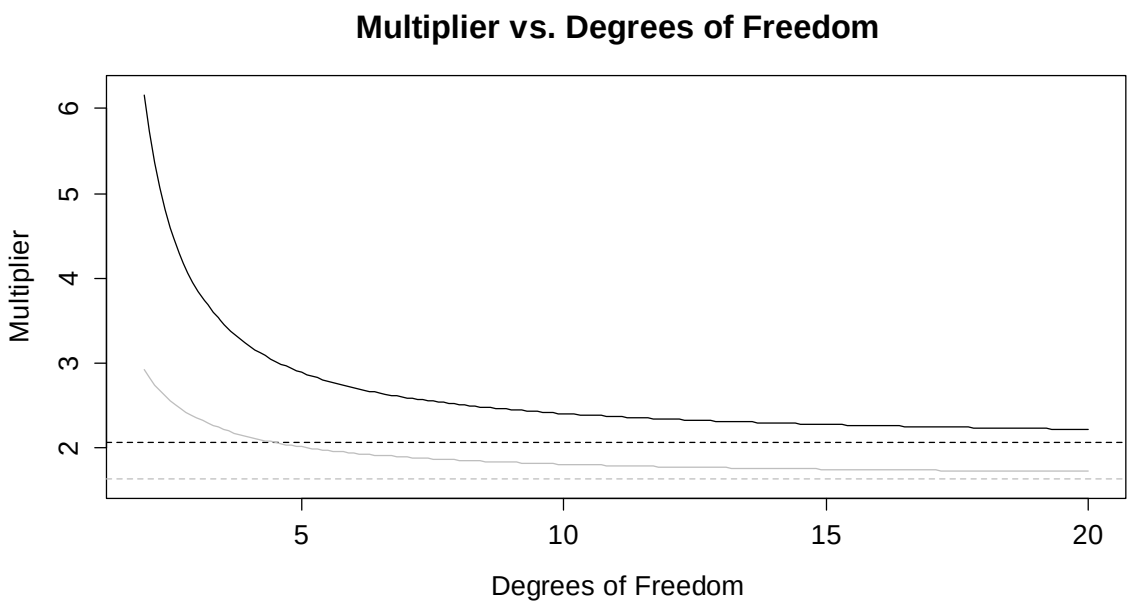
The result turns out to be very similar to what we had computed empirically. The downside of the 20-day empirical computation is that it is based on relatively few observations, but it is methodologically sound. Here, we have a sound estimate of more than 2000 observations for estimating mean and standard deviation. However, the computations are based on assuming a Gaussian Random Walk and there is plenty of evidence in the data, that this model is not 100% accurate.

For the $t_\nu(\mu,\lambda^2)$-distribution, a similar formula for computing the ES exists:

$$ES = \mu - \lambda \cdot \frac{f_\nu(F_\nu^{-1}(\alpha))}{\alpha} \cdot \frac{\nu + (F_\nu^{-1}(\alpha))^2}{\nu - 1}$$

Not surprisingly, the multiplier for the scale parameter depends on the degrees of freedom. The heavier the tails of the distribution are, the further away from the mean the ES is. The diagram below shows the relation between the multiplier and the degrees of freedom by assuming $\alpha = 0.05$.

```
> mm <- function(nü, aa=0.05){
+     dt(qt(aa,nü),nü)/aa * (nü+qt(aa,nü)^2)/(nü-1)
+ }
> plot(seq(2,20,by=0.1),mm(seq(2,20,by=0.1)), ...)
> abline(h=dnorm(qnorm(0.05))/0.05, col="grey")
```



The function converges to 2.063, the multiplier value for the Gaussian. The grey line corresponds to the multiplier for the 95%-VaR for the respective distribution. We observe that the ratio between ES and VaR is not constant, but depends on the shape parameter ν. Let us now implement the ES computation for the case of Google. We had estimated 2.943 degrees of freedom for the log returns. Hence:

```
> out <- fitdistr(lr.google, "t")$estimate
> out[1]-out[2]*mm(out[3])
-0.05159754
```

Under this model, the ES is estimated as -5.16%. This is now more conservative than for the Gaussian Random Walk, owing to the fact that the heavy tail has more effect in computing the ES than in computing the VaR. We conclude this section by noting that for multi-period ES under heavy tails, Monte Carlo simulations are again required for determining the multi-period log return distribution.

5.2.3 GARCH Computation

When working with a GARCH model that assumes Gaussian innovations, the time series forecast for the conditional standard deviation is obtained in exactly the same manner as we had demonstrated for the VaR in section 5.1.3. The result can be plugged into the formula from 5.2.2 and directly yields the GARCH-ES:

```
> gfit.nd <- garchFit(~garch(1,1), lr.google)
> pred.sd <- predict(gfit.nd, n.ahead=1)[3]
> coef(gfit.nd)[1]-pred.sd*2.063
-0.02926222
```

The empirical result was -4.98%, the Gaussian Random Walk yielded -4.37% and here, we only have -2.93%. The reason is again the fact that at the end of 2012, we are in a very low volatility period. If computing multi-period ES is the goal, this works analogously to what we had shown for the VaR in section 5.1.3. A multi-step prediction of the conditional standard deviation is produced and these are converted into a scale parameter for the multi-period log return.

```
> pred.sd <- predict(gfit.nd, n.ahead=20)[,3]
> 20*coef(gfit.nd)[1]-sqrt(sum(pred.sd^2))*2.063
-0.1235052
```

The result is -12.35%. The final task which remains to do is to implement ES computation using a GARCH model with heavy tails. For a 1-day horizon, this is straightforward: we generate a 1-step time series forecast of the conditional standard deviation, convert it to the conditional scale parameter, determine the multiplier according to the degrees of freedom, et voilà:

```
> gfit.ht <- garchFit(~garch(1,1), lr.google, cond.dis="std")
> cf      <- coef(gfit.ht)
> lpred   <- sqrt((cf[5]-2)/cf[5])*predict(gfit.ht, 1)[3]
> cf[1]-lpred*mm(cf[5])
-0.03004255
```

The result is -3.00%. As usual, when we want multi-period risk measures from the heavy-tailed approach, we have to employ Monte Carlo. This works exactly the same as demonstrated in the VaR chapter, except that we do not determine the 5%-quantile of the Monte Carlo distribution, but instead compute its empirical ES. The result turns out to be -11.13%.