RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

Self-Study

Volatility analysis

of the DAX index 2010-2016

Eldar Keller

05.07.2017

Abstract

This work is intended as a summary of the knowledge acquired, when conducting the first volatility analysis on a data sample, consisting of financial returns. The goal was, to get a feeling for the current "state-of-the-art" in volatility modelling and lay the groundwork for more extensive studies, not necessarily bounded to this specific topic. As such, many basic points are covered, which are not only exclusively important to analyze volatility, but also in other contexts of data analysis.
The data used, are the interday returns of the german stock index DAX, spanning from the beginning of 2010, to the end of 2016. We cover the specific properties that are typical for financial returns, and apply several models that are commonly used for volatility modelling nowadays, like GARCH, Beta-t-EGARCH and SV-type models. While doing this, we also look into their structures, properties and the motivation behind them, to build a deeper understanding for them.
Following this, we assess their performance in an in- and out-of-sample analysis, using additional DAX return data of January 2017. We conclude that the two-component Beta-Skew-t-EGARCH model seems the most promising of all contenders, as it covers all the financial return properties of our sample.
The estimations and calculations are done using R, with the code added in the appendix.

Table of Contents

1 Introduction

Stock markets all over the world play a central part in the respective countries' economy. They are important means of acquiring capital, from the companies' side and essential to accumulate returns from capital, from the investor's side. Hence, the situation on the stock markets, which can be measured by stock indices, is a good indicator for the overall state of the economy.
The most important german stock index is the DAX, which encompasses the 30 largest german companies, with respect to market capitalization and trading volume. As the index is formed by non-deterministic supply and demand, its value can be described by a stochastic time series process. Naturally, some phases in the DAX index exhibit greater fluctuations than others. Particularly the global politics and economical situations have a great influence on the market sentiment. Events, unexpected by the market participants, like the outcome of the "Brexit" referendum on the 23rd June 2016, or the electoral outcome in the United States on the 8th November 2016, can lead to short term crashes. While these political shocks were mostly short-termed, other episodes exhibited longer periods of insecurity on the stock markets - a prime example being the financial crisis of 2008, which lead to an increase in the (implied) volatility index VDAX-NEW that lasted several months and formed the VDAX all-time high, after the insolvency of the Lehman Brothers investment bank on the 15th September 2008.

Figure 1: VDAX-NEW: DAX (implied) volatility index, from 01/2008 - 01/2010;
Describes the (annualized) implied volatility of the DAX, i.e. an index value of 50% means an expected deviation from the current value of 50% in a year.

In the analysis of this paper, we focus on the DAX performance from the beginning of the year 2010, to the end of 2016, though. The goal will be to quantify the size of fluctuations, by modelling the standard deviation of the stochastic process that describes the DAX closing price returns.
Besides the widely known standard GARCH-model (see [Bo1]), we will incorporate sev-

eral other submodels, such as the EGARCH, TGARCH and GJR-GARCH models (see [Ne1], [Za1] and [GJR1]), which allow for an asymmetric parameter update feedback. This proves especially useful for financial time series of returns, as they often are negatively correlated with the volatility, i.e. negative returns tend to increase it, while positive ones tend to decrease it. Said property is also called the "leverage effect" and was firstly described in [Bl1].
Besides these models from the GARCH family, we will involve the relatively new GAS and Beta-t-EGARCH models. They have been suggested more or less simultaneously in [CKL1] and [HC1], employing similar approaches to the modelling of time-varying parameters. While both use the score as a driving factor in the volatility updating function, their motivation was quite different and resulted in different models.
The GAS approach can be seen as more of a framework to time series modelling that encompasses many other models, such as GARCH and Beta-t-EGARCH. Its specification is very general and allows for several specific parameter adjustments.
Harvey's Beta-t-EGARCH on the other hand, was motivated directly from the fact that the distribution of returns usually have heavy tails (see for example [AFT1]), which was not accounted for in the gaussian GARCH models. To alleviate this problem, Bollerslev suggested the use of the student's t, instead of the gaussian distribution, see [Bo2]. But as the updating function for the volatility estimate remains unchanged, it is still prone to outliers. Hence, in order to adapt to the heavy tails property of returns, Harvey adjusted the EGARCH parameter update and chose the score as the driving factor, which allows for a new updating function that is less prone to extreme deviations due to outliers.
Lastly, SV-models are used, which have firstly been suggested in [Ta1]. In contrast to the others, the SV model assumes an AR(1) process for the logarithmed variance, which in particular is non-deterministic.

The content of the upcoming analysis is structured as follows: We will start off with a few basic definitions and explanations of the used terminology. Next comes the description of the data set, accounting for its various properties, some of which are typical for financial returns. After laying the groundwork, we will apply the volatility models on the DAX data from 2010 to 2016, after specifying them. We follow up with an evaluation of the results, quantifying the goodness of the in-sample performance and out-of-sample forecasts with tests and several criteria. The out-of-sample performance will be tested with the DAX values of January 2017. Lastly comes the conclusion, summing up the results and suggesting several points for possible further work. An appendix is also added, containing the R code used.

1.1 Basics & Definitions

We begin with the basic terminology:

DAX

As an index, the DAX covers the 30 largest german companies, with respect to market capitalization and trading volume. It should me noted that it is a total return index (in contrast to a price index), which means that e.g. dividend payments are included in the DAX, assuming that all dividend earnings are reinvested. This leads to a steady increase in the DAX, with a growth faster than the actual stock prices.
Nearly all other famous indices are price indices, so this is a distinct feature of the DAX.

Realized Volatility

The realized (historical) volatility is given by the root of the squared (demeaned) returns, which are either based on inter- or intraday data. It can be used as a measure of the latent standard deviation of returns.

Implied Volatility

In contrast to realized volatility that is calculated with already observed returns, this term describes the expected market price deviation over a certain time (in the future). It is an important variable in financial option pricing.

VDAX-NEW

This stock index describes the implied volatility of the DAX in annualized form. It is calculated by the approximation of the underlying volatilities of financial options, which are based on the DAX.

Leverage Effect

Stock returns usually are negatively correlated with (future) volatility in a linear way, i.e. negative returns lead to a higher volatility, while positive ones tend to decrease it. This effect was firstly observed by [Bl1]. The most common explanation is the following:
It can be noted that dropping stock prices lead to an increased debt to equity ratio, as the value of the equity decreases. This leads to a higher financial leverage and equity risk, which in turn increases volatility.
However, it is likely that this is not the only factor leading to this negative correlation, as described in more detail in [CW1].

In order to check for the existence of the leverage effect property in a time series, one may use Pearson's sample correlation coefficient $\hat{\rho}$, to look into the (linear) correlation between the squared sample values y_t^2 and their previous return y_{t-1}. The use of the squared returns is suggested as a measure of (realized) volatility, considering that $Var(y_t) = \mathbb{E}[y_t^2] - \mathbb{E}[y_t]^2$ and $\mathbb{E}[y_t]^2$ is usually very small for returns.

Pearson's sample correlation coefficient is given by

$$\hat{\rho} = \frac{\sum_{i=2}^{n} (y_i^2 - \bar{y}_2) \sum_{i=1}^{n-1} (y_i - \bar{y}_1)}{\sum_{i=2}^{n} (y_i^2 - \bar{y}_2)^2 \sum_{i=1}^{n-1} (y_i - \bar{y}_1)^2}, \tag{1.1}$$

where n denotes the sample size of the time series data, $\bar{y}_2$ is the sample second moment of the data without the first value and $\bar{y}_1$ is the sample mean without the last value.

One should note that the accuracy of this coefficient gravely relies on a sufficient sample size. According to [BD1], chapter 7, the sample correlation coefficient is asymptotically normal distributed under certain assumptions. Its mean is zero and variance $\frac{1}{\sqrt{N}}$ under the null hypothesis of no correlation and if the underlying sample is iid, with finite second moments. Here, we have $N = n-1$. When looking at the sample autocorrelation $\hat{\rho}_k$, which is introduced below, one takes $N = n - k$ for the sample size that is used to estimate the correlation coefficient.

Using the asymptotically normal distribution, the critical values for a two-sided test can be deducted as $\frac{q_{0.975}}{\sqrt{N-k}}$ and $-\frac{q_{0.975}}{\sqrt{N-k}} = \frac{q_{0.025}}{\sqrt{N-k}}$, with $q_{0.975} \approx 1.96$ describing the respective quantile of the normal distribution.

Hence, if the absolute size of the sample correlation coefficient is larger than $\frac{q_{0.975}}{\sqrt{N}}$, we can reject the null hypothesis of no correlation (i.e. $\rho = 0$) at a 5% significance level.

After checking, whether it is significantly different from zero, we can see, whether there is a positive or negative correlation. The latter would suggest that the sample is affected by the leverage effect, as then low returns are associated with high squared returns on the next day, indicating higher volatility.

(Logarithmic) Returns

Returns could be described, by simply calculating the ratio between the price difference over a time period and the current price, i.e.

$$R_t = \frac{p_{t+1} - p_t}{p_t}. \tag{1.2}$$

These returns don't add up over time, though, as e.g. a 5% return for two consecutive days leads to a total return that is larger than 10%. Logarithmic returns in contrast, which are given by

$$R_t^{\log} = \log(R_t + 1) = \log(p_{t+1}) - \log(p_t), \tag{1.3}$$

are additive in this sense and will be thus be used in the following parts. Unless differently specified, "returns" will refer to logarithmic returns.

Augmented Dickey Fuller test

The Augmented Dickey Fuller (ADF) test is a generalization of the Dickey Fuller test introduced in [DF1]. It tests for the existence of an unit root in an autoregressive process of order p, while the Dickey fuller test only covers the $p = 1$ case. The

presence of an unit root implies that wide-sense stationarity is not possible and that temporary shocks have a permanent influence on the mean, which particularly means that the process is not mean-reverting. The test procedure is as follows:
First, one has to fit an AR(p) model to the data, given by

$$y_t = \alpha + \sum_{i=1}^{p} \phi_i y_{t-i} + \epsilon_t. \tag{1.4}$$

It is known that this process has a unit root for $\sum_{i=1}^{p} \phi_i = 1$, while values bigger than one result in an explosive process. Usually, this standard formulation is rewritten, using the difference operator Δ, with $\Delta y_t = y_t - y_{t-1}$ and $\gamma = (\sum_{i=1}^{p} \phi_i) - 1$, so that the process has a unit root for $\gamma = 0$. The reformulated version of (1.4) is then

$$\Delta y_t = \alpha + \gamma y_{t-1} + \sum_{i=1}^{p-1} \delta_i \Delta y_{t-i} + \epsilon_t, \tag{1.5}$$

for some coefficients δ_i. The motivation behind this is to acquire an estimate for γ by using the OLS method. In practice, this can be implemented by embedding y_t and its lagged values up to order p into an euclidean space of dimension $p+1$ and then performing linear regression, to acquire the necessary coefficient estimates (see Miscellaneous subsection of the code in the Appendix for implementation). The resulting estimate $\hat{\gamma}$ and its standard error is then plugged into the test statistic

$$DF_p = \frac{\hat{\gamma}}{\sigma_\gamma}, \tag{1.6}$$

where σ_γ denotes the OLS estimate's standard error.
To further generalize this test to models with deterministic trend, we add the term βt to (1.5), resulting in

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \sum_{i=1}^{p-1} \delta_i \Delta y_{t-i} + \epsilon_t. \tag{1.7}$$

Testing the null hypothesis $\mathcal{H}_0 : \gamma = 0$ against the alternative $\gamma < 0$, we can now reject the null hypothesis, if the DF_p statistic in (1.6) is smaller than the Dickey-Fuller distribution's quantile for the chosen significance level. As the test statistic doesn't follow any commonly used distribution, the resulting quantiles could be approximated, using the quantile table of the DF distribution, if one is interested in the more exact p-value. A table with the most commonly used significance levels can for example be found in the appendix of [Fu1] (Table 10.A.2).
The outcome of the test also depends on the lag value used. To fit the appropriate AR order p, one can test for remaining autocorrelation in the residuals, by employing a portmanteau test (see Basics below), or select the order according to the AIC or BIC (see Basics below). Using the first way, the procedure is to start off with

an AR(0) model and progressively increase the lag order, until the portmanteau test suggests that there is no autocorrelation left. If one consults the AIC and BIC values, one simply fits a few possible orders and chooses the one with the highest AIC and/or BIC value.

ARCH effect testing

In order to test for ARCH effects, i.e. check, whether the variance is time-varying according to an autoregressive process, the following tests may be used:

Score Test

In [En1], Engle used a (locally most powerful) score test, to check whether a sample exhibits heteroscedasticity in an autoregressive fashion. Let $s(\cdot)$ denote the score function and $\mathcal{I}(\cdot)$ the fisher information matrix. Then a score test's null hypothesis is $\mathcal{H}_0 : \theta = \theta_0$ and the test statistic is defined by

$$F(\hat{\theta}_0) = s(\hat{\theta}_0)'\mathcal{I}(\hat{\theta}_0)s(\hat{\theta}_0), \tag{1.8}$$

where $\hat{\theta}_0$ is the maximum likelihood estimate for the remaining parameters that are not assumed to be θ_0 under the null hypothesis. The statistic F is asymptotically χ^2-distributed with $p = \dim \theta_0$ degrees of freedom, if the null hypothesis holds. In our case, testing for ARCH effects, we have

$$\begin{aligned} &\mathcal{H}_0 : \alpha_1 = \alpha_2 = \cdots = \alpha_p = 0, \text{ for} \\ &\sigma_t^2 = \alpha_0 + \alpha_1 y_{t-1}^2 + \cdots + \alpha_p y_{t-p}^2, \end{aligned} \tag{1.9}$$

with σ_t^2 being the conditional sample variance. The squared returns are used as a measure of variance, as the true process is obviously unknown. We reject the null hypothesis at a 5% significance level, if F is bigger than the quantile $\chi^2_{p;1-\alpha}$.
Two things should be remarked:
Firstly, we are testing a joint hypothesis - in particular not the Hypothesis $\mathcal{H}_0 : \alpha_1 = 0, \ldots, \alpha_p = 0$. This means that higher lag orders don't necessarily imply a higher probability of rejecting the null hypothesis (1.9), as the coefficients may be insignificant when regarded jointly, while some are significant individually.
Secondly, the score and fisher information are used, hence the test statistic (1.8) depends on the assumed sample distribution. These can not always be easily calculated or approximated. Engle used a normal distribution and noted in [En1] that in this case, the statistic is equivalent to the rescaled coefficient of determination R^2 that one gets by OLS optimization:

$$F(\alpha) = (n-p) \cdot R^2(\alpha) = (n-p) \cdot \frac{\sum_{i=1}^{n-p} \left(\hat{y}_i(\alpha) - \bar{y}\right)^2}{\sum_{j=1}^{n-p} \left(y_j - \bar{y}\right)^2}, \tag{1.10}$$

for $\alpha = (\alpha_1, \ldots, \alpha_p)$ being the regression coefficients, resulting in the fitted values $\hat{y}_i(\alpha)$ and $\bar{y}$ denoting the mean of the sample $\{y_i\}_{i=1}^n$. It is then equivalent to the "Breusch-Godfrey" test, introduced in [Br1] and [Go1].
However, this normality assumption may not always justified. Especially financial

returns exhibit several distribution properties, which are not in line with a normal distribution.

Portmanteau Test

This sort of test checks, whether the (estimated) autocorrelations of a sample are significant, or not. A commonly used variant of portmanteau tests was introduced in [LB1], as the Ljung-Box Q test. It incorporates the (biased) sample autocorrelation that is given by

$$\hat{\rho}_k = \frac{\sum_{i=1}^{n-k} (y_{i+k} - \bar{y})(y_i - \bar{y})}{\sum_{i=1}^{n} (y_i - \bar{y})^2}, \tag{1.11}$$

where $\bar{y}$ again denotes the mean of the sample $\{y_i\}_{i=1}^n$ and k is the lag for the estimated autocorrelation.

With this, the null hypothesis of the test can be defined: $\mathcal{H}_0 : \rho_1 = \cdots = \rho_p = 0$. In particular, we again check a joint hypothesis, just like in the aforementioned score test.

The corresponding Ljung-Box Q statistic is built as a rescaled squared sum of the sample autocorrelation:

$$Q = n(n+2) \sum_{k=1}^{p} \frac{\hat{\rho}_k^2}{n-k}, \tag{1.12}$$

which is χ^2-distributed with p degrees of freedom. Again, the null hypothesis can be rejected at a 5% significance level, if Q is bigger than the quantile $\chi^2_{p;1-\alpha}$.

If one of the autocorrelations is non-zero, we can follow that there exists a (linear) dependence between an observation and its lagged value. Hence, by applying this test on the demeaned, squared observations, we can check for the existence of ARCH effects.

The optimal choice of p for both tests may be tricky. In non-obvious cases, a lag length that is too short may miss some dependencies with higher lagged variables. However, if it is chosen too large, the overall significance of the lag values may decrease, even if some lag values would be significant otherwise. These effects also depend on the sample size.

A sensible solution would be to test for several lags and according to the expected attributes of the data sample, maybe including seasonality. For the Ljung-Box test on a sample of size n, Tsay suggests the use of $p = \ln(n)$ in [Ts1].

Kurtosis

To describe the shape of probability distributions, particularly their tails, kurtosis can be used. In this work we will use the standard kurtosis as specified by Pearson that is given by the fourth central moment divided by the squared variance:

$$\kappa(X) = \frac{\mathbb{E}\left[(X-\mu)^4\right]}{\left(\mathbb{E}\left[(X-\mu)^2\right]\right)^2}, \text{ estimated by } \hat{\kappa}(X) = \frac{\frac{1}{n}\sum_{i=1}^{n}\left(X_i - \bar{X}\right)^4}{\left(\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2\right)^2}, \tag{1.13}$$

where $\bar{X}$ describes the sample mean. Its value is often compared to the kurtosis of the normal distribution, which takes the value 3. If $\kappa > 3$, a distribution has heavier tails than the normal distribution and for a value smaller 3 they are lighter.

Skewness

As a measure of asymmetry, the skewness if often used to classify probability distributions. We will use the skewness by Pearson, defined as the quotient of the third central moment and a power of the variance:

$$\gamma = \frac{\mathbb{E}\left[(X-\mu)^3\right]}{\left(\mathbb{E}\left[(X-\mu)^2\right]\right)^{3/2}}, \text{ estimated by } \hat{\gamma}(X) = \frac{\frac{1}{n}\sum_{i=1}^{n}\left(X_i - \bar{X}\right)^3}{\left(\frac{1}{n}\sum_{i=1}^{n}(X_i-\bar{X})^2\right)^{3/2}}, \tag{1.14}$$

where $\bar{X}$ describes the sample mean. A negative-/left-skewed distribution exhibits a longer left tail, while a positive value indicates a longer right tail, the standard value being zero.

(Non-standardized) student's t-distribution

The standard student's t-density function for $\nu > 0$ degrees of freedom, is given by

$$p(x \mid \nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\Gamma\left(\frac{\nu}{2}\right)}\left(\frac{x^2}{\nu}+1\right)^{-\frac{\nu+1}{2}}. \tag{1.15}$$

Its mean is zero for $\nu > 1$ and the variance is $\frac{\nu}{\nu-2}$ for $\nu > 2$. If $\nu \leq 1$, neither are defined and if $1 < \nu \leq 2$, the variance is infinite.
To generalize a t-distributed random variable X, one can now rescale it with the location parameter μ and the scale parameter σ: $\frac{X-\mu}{\sigma}$. We then acquire the density function, specifying the non-standardized t-distribution with location parameter μ, scale parameter σ and ν degrees of freedom:

$$p(x \mid \nu, \mu, \sigma) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\sigma\Gamma\left(\frac{\nu}{2}\right)}\left(\frac{1}{\nu}\left(\frac{x-\mu}{\sigma}\right)^2+1\right)^{-\frac{\nu+1}{2}}. \tag{1.16}$$

The mean is then equal to the location parameter and the variance is given by $\sigma^2\frac{\nu}{\nu-2}$. In particular, σ doesn't denote the standard deviation in this case, but the scale parameter. To acquire a variance of σ^2, we rescale X once more with $\frac{\nu-2}{\nu}$, resulting in the density

$$p(x \mid \nu, \mu, \sigma) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{(\nu-2)\pi}\sigma\Gamma\left(\frac{\nu}{2}\right)}\left(\frac{1}{\nu-2}\left(\frac{x-\mu}{\sigma}\right)^2+1\right)^{-\frac{\nu+1}{2}}. \tag{1.17}$$

Skewed t-distribution

In order to induce skewness in the t-distribution, one can use the method defined in [FS1]. It can be applied to any unimodal, symmetric density f around zero, i.e.

any density f with $f(x) = f(|x|)$ that is decreasing in $|x|$. The method introduces a skew-parameter γ to the density:

$$f(x \mid \gamma) = \frac{2}{\gamma + \gamma^{-1}} f\left(\frac{x}{\gamma^{\mathrm{sgn}(x)}}\right). \tag{1.18}$$

The skew parameter $\gamma \in (0, \infty)$ makes f a symmetric density for $\gamma = 1$, as then $f(x \mid \gamma) = f(x)$, while a value smaller or larger than 1 results in a negatively or positively skewed density.
As described in [FS1], the skewing parameter affects the mean and variance, which means that we cannot specify them directly by the use of rescaling via $\frac{X-\mu}{\sigma}$. In particular, f has to be symmetric to apply the skewing method. Thus, we let $f = f_\nu$ be the standard student's t-density with ν degrees of freedom, apply the method and rescale it afterwards:

$$p(x \mid \gamma, \nu, \mu, \sigma) = \frac{2}{\gamma + \gamma^{-1}} f_\nu\left(\frac{x-\mu}{\sigma\gamma^{sgn(x-\mu)}}\right). \tag{1.19}$$

Here, μ and σ don't denote the mean and standard deviation, but only affect them. Hence, they are named location and scale parameter respectively.

Kernel density estimation

A common, non-parametric way of estimating a (density) function is by the use of kernels. Kernels are symmetric, non-negative functions, whose integral over $\mathbb{R}$ is 1. For our density estimation, we use the gaussian kernel, equal to the standard normal density function

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}. \tag{1.20}$$

The principle behind the kernel estimation for any x is that we evaluate the averaged, shifted kernel functions for every data point. In order to make the kernel's effective range more flexible, it is usually scaled with a smoothing parameter, called bandwidth.
For an easily computable and sensible bandwidth, the rule of thumb formula for kernel density estimation bandwidth, introduced in [Sc1], could be used:

$$h = \left(\frac{3n}{4}\right)^{-\frac{1}{5}} \sigma \approx 1.06 n^{-\frac{1}{5}} \left(\hat{\sigma} \vee \frac{\hat{q}_{0.75} - \hat{q}_{0.25}}{1.349}\right). \tag{1.21}$$

Here, n is the sample size and the real standard deviation σ is approximated by either the sample standard deviation $\hat{\sigma}$, or the sample's interquartile range, divided by 1.349. The motivation behind this is that the interquartile range's expected value is $2q_{0.75}\sigma \approx 1.349\sigma$ for normally distributed samples, with upper quartile $q_{0.75}$. However, as we will see that our data sample is not normally distributed, this possibility is omitted and only the standard deviation is used for the estimate

instead.
Altogether, for a (gaussian) kernel K as in (1.20), with K_h being the smoothed kernel for a bandwidth h and a sample $(x_1, \ldots, x_n)$, we get the following estimator:

$$\hat{f}_h(x) = \frac{1}{n} \sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right). \tag{1.22}$$

Long-memory property of returns

As described in [ABCD1], the effect of shocks on the volatility of financial returns is often very persistent. If we consider a GARCH model or any of its subtypes, the autocorrelation decays at an exponential rate. Some studies suggest that this is too fast, like for example [DGE1] (which also introduces the APARCH model). Considering a standard GARCH model (see section 3 for its specification), the effect of an observation at time t on the volatility forecast at time $t + h$ can be quantified by

$$\frac{\partial \sigma_{t+h}^2}{\partial y_t^2} = \alpha(\alpha + \beta)^{h-1}, \tag{1.23}$$

which can be derived directly from the updating equation for σ_t^2. As $\alpha + \beta < 1$ is assumed for stationarity, this implies an exponential decrease in h. For a slower rate, an alternative submodel that shall specifically capture this long-memory behavior was introduced in [EL1], by splitting the GARCH updating equation into a short-term and a long-term component.
In general, a model is said to have the long-memory property, if its autocorrelation decreases at a polynomial rate for increasing lag. More details and examples can be found in [Ts1].

Significance testing with t-statistic

In order to check, whether one of our models is overfitting the sample, i.e. if there are any unnecessary parameters, we can apply a widely used test. For any parameter estimate $\hat{\beta}$, it tests the null hypothesis $\mathcal{H}_0 : \hat{\beta} = \beta_0$ against the alternative $\hat{\beta} \neq \beta_0$. As we want to see, if our parameters are significant, we apply the test with $\beta_0 = 0$. Then the t-statistic is given by

$$t_{\hat{\beta}} = \frac{\hat{\beta} - \beta_0}{\sigma_{\hat{\beta}}} = \frac{\hat{\beta}}{\sigma_{\hat{\beta}}}, \tag{1.24}$$

where $\sigma_{\hat{\beta}}$ denotes the standard error of our estimate $\hat{\beta}$, i.e. the standard deviation of its sampling distribution.
As we only employ maximum likelihood estimators in this work, the asymptotic distribution of $\hat{\beta}$ is $\mathcal{N}(0, \sigma_{\hat{\beta}}^2)$ under the null hypothesis and some regularity conditions. Then, the asymptotic distribution of the t-statistic is standard normal.

Considering that the standard normal distribution is symmetric and our sample size is sufficiently large, we can now approximate the p-value by

$$p = 2 \cdot (1 - \Phi(|t_{\hat{\beta}}|)), \tag{1.25}$$

with $\Phi(\cdot)$ denoting the standard normal cumulative distribution function. To use this test, we firstly have to compute the standard error $\sigma_{\hat{\beta}}$. This can be done for every estimated parameter at once, by approximating the covariance matrix Σ with the inverse observed information matrix $\mathcal{I}^{-1}$. The latter is given by the negative hessian of the likelihood $-\hat{\mathbf{H}}$, or the hessian of the negative likelihood, evaluated at the maximum likelihood estimate. Altogether, this means

$$\Sigma \approx \mathcal{I}^{-1} = (-\hat{\mathbf{H}})^{-1}, \tag{1.26}$$

which yields the standard errors for every parameter estimate, by calculating

$$\sqrt{\operatorname{diag}\left((-\hat{\mathbf{H}})^{-1}\right)}. \tag{1.27}$$

Some problems might be encountered, when applying this test. For example, the approximation by the use of the asymptotic gaussian distribution can be inaccurate for small sample sizes and when the regularity conditions are not met. Additionally, the inversion of the hessian may prove to be difficult and require high numerical precision, specifically for highly (in)significant parameters.
Altogether, the test can provide some insight into the impact that a parameter has on a model's estimates. However, one should always consult some information criteria like the AIC or BIC, before omitting a parameter.

Model selection with AIC or BIC

The AIC was introduced as "An Information Criterion" in [Ak1], but is mostly referred to as "Akaike's Information Criterion" nowadays. Even though its formula is very simple, its motivation is founded in the information theoretic quantity "Kullback-Leibler divergence", which quantifies the difference between the true and an approximative distribution. As the true distribution is unknown, the Kullback-Leibler divergence's expected value is considered.
Thus, the AIC shall determine the model that maximizes the expected log likelihood and in turn minimizes the expected Kullback-Leibler divergence. It is given by

$$AIC = -\ell_{\max} + 2k, \tag{1.28}$$

which is the negative, maximized log likelihood $\ell_{\max}$, plus two times the number of the estimated parameters. The model that minimizes the AIC is to be chosen. Using it, we can find out, which one of our (model implied) conditional distributions approximates the truth best. Thus, the criterion doesn't just look at the actual volatility estimates, but also the shape of the conditional distribution. However, it

can only be used as a comparison measure, not to establish the actual performance. If the possible model set is full of poor choices, it might choose the best of them, but this model's actual performance still is bad.
There also are some misconceptions about the AIC: Some claim that it is only valid, when working with the true or nested model distributions. This is not correct, as the kullback-leibler divergence measures the difference between the true and any estimated distribution, which has also been addressed in a comment by the authors of [BA1], which can be found at [BA2].
If the sample size n were small, or the number of parameters k relatively high, we would resort to the corrected AIC (AICc), which adds another penalty term that is decreasing in n:

$$AICc = AIC + \frac{2k(k+1)}{n-k-1}. \tag{1.29}$$

For our sample and models, this added term would be smaller than 0.1 in all cases and is thus omitted.
In contrast to the AIC and its own name, the BIC ("Bayesian Information Criterion") is not based on information theory. It was introduced in [Sch1] and derived through bayesian inference. Most commonly, the BIC is written as

$$BIC = -\ell_{\max} + \log(n)k, \tag{1.30}$$

with the same notation as for the AIC and n denoting the sample size. We can see that in contrast to the AIC model, the penalty term with the number of estimated parameters k is dependent on the sample size and bigger in pretty much all applications with $\log(n) > 2$.

The differing penalty term leads to different outcomes, each having its own legitimation. If they give contrasting results, one has to choose according to one's aim of modelling: If the model has to be good at forecasting, more parameters might be sensible, than if one just tries to explain a sample's properties. Thus, choosing according to the AIC is suggested when forecasting, while the BIC is more sensible when looking for a parsimonious model to explain the data at hand. More details about each criterion's advantages and disadvantages can be found in [BA1].

(Forecasting) Loss functions

In this work, the following loss functions will be used, to asses forecasting quality:

$$MSE = \frac{1}{n}\sum_{t=1}^{n}\left(\hat{\sigma}_t^2 - \sigma_t^2\right)^2 \tag{1.31}$$

$$MAE = \frac{1}{n}\sum_{t=1}^{n}\left|\hat{\sigma}_t^2 - \sigma_t^2\right| \tag{1.32}$$

$$MSD = \frac{1}{n}\sum_{t=1}^{n}\left(\hat{\sigma}_t^2 - \sigma_t^2\right) \tag{1.33}$$

$$QLIKE = \frac{1}{n}\sum_{t=1}^{n}\left(\log(\sigma_t^2) + \frac{\hat{\sigma}_t^2}{\sigma_t^2}\right) \tag{1.34}$$

where $\hat{\sigma}_t^2$ denotes the (variance) estimator and σ_t^2 is its true value. In practice, σ_t^2 is unknown and will be represented by a so-called volatility proxy. The proxy applied here are the squared, demeaned returns $(y_t - \bar{y}_t)^2$.

Our first loss function, the Mean Squared Error (MSE) is probably the most widely used one. However, one downside of it is that it heavily weighs outliers, due to the squared error term. Hence, a natural alternative is the Mean Absolute Error (MAE) in (1.32). To find out, if the forecasts have a tendency to be too small or too large, we also consider the Mean Signed Deviation (MSD), even though it is obviously flawed as a loss function. Lastly, the QLIKE loss function is included that was introduced in [BEN1] and motivated by the gaussian likelihood function. In [Pa1], the author shows that from the above loss functions, only the MSE and QLIKE are "robust", in the sense that

$$\mathbb{E}\left[L(\sigma_t^2, \hat{\sigma}_t^2)\right] \underset{<}{\overset{\geq}{=}} \mathbb{E}\left[L(\sigma_t^2, \hat{\sigma}_t'^2)\right] \Leftrightarrow \mathbb{E}\left[L(p_t^2, \hat{\sigma}_t^2)\right] \underset{<}{\overset{\geq}{=}} \mathbb{E}\left[L(p_t^2, \hat{\sigma}_t'^2)\right] \tag{1.35}$$

holds, where σ_t^2 is the true (conditional) variance with unbiased proxy p_t^2, such that $\mathbb{E}_t\left[p_t^2\right] = \sigma_t^2$ and $\hat{\sigma}_t^2, \hat{\sigma}_t'^2$ are any two estimators/forecasts.

Thus, if the loss functions give conflicting results, the QLIKE or MSE values should be used.

2 The Data

In this section, the properties and special features of the used data set will be described, in order to get a feeling for it and prepare for the following volatility modelling section. The data is based on the Closing prices of the DAX, i.e. the index value at the end of each trading day, given by the Xetra stock exchange. As mentioned before, the data spans from the beginning of the year 2010 to the end of 2016, containing 1780 index values. While the data from 2010 to 2016 will be used for an in-sample volatility analysis, additional data for January 2017 will be used to assess the out-of-sample performance of our models' forecasts. In the beginning, we will hence focus on the first part of the data set.
We start off, by checking the basic properties and plotting the raw data, to get a feeling for it:

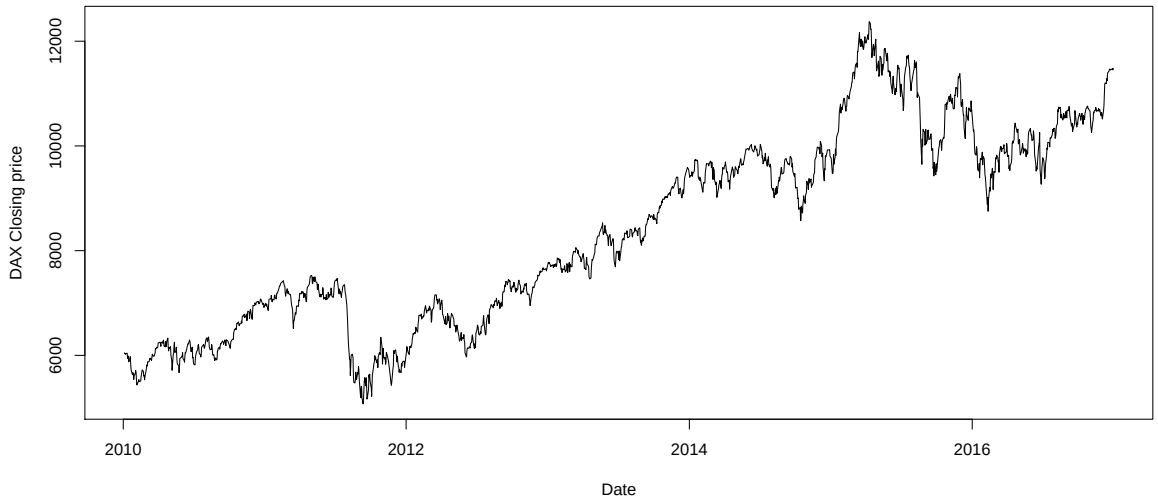

Figure 2: DAX Closing price 2010-2016, spanning over 1780 values.

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   5072    6754    8244    8366    9892   12370
```

One firstly notes that the index value doubled from 2010 to 2016 and shows a steady increase. This, however, is not only attributed to rising stock prices after the crash following the financial crisis in 2007/2008, but also to the fact that the DAX is a total return index and hence increasing faster than actual stock prices. The price index version of the DAX suggests only a 60% increase for the same time period.
Due to this positive trend, a (notably?) positive mean of the returns can be expected.
One of the most basic assumptions on a time series is wide-sense stationarity. A violation of it is usually caused by a unit root, which implies that temporary shocks have a permanent effect on the time series. To test for the existence of a unit root, we use the

Augmented Dickey Fuller test (see basics) and get the following results for the actual DAX values and their returns, after assessing an appropriate lag order p for the test (see Appendix):

Test statistics for DAX, with AR order of 5 and 7:	-2.756	-2.865
Corresponding p-values:	0.258	0.212
Test statistics for DAX log returns, with AR order of 2 and 5:	-24.560	-18.560
Corresponding p-values(too low to display):	0.010	0.010

Thus, the tests suggest that a unit root is present in the course data. However, we can see that this can be handled by considering the logarithmed returns instead, which are more convenient for modelling the volatility and commonly used. Hence, we will work with the (logarithmic) returns of the DAX, to highlight large deviations from the expected return.
Simply by looking at the log returns, we can clearly see that there are some periods, which are more volatile than others:

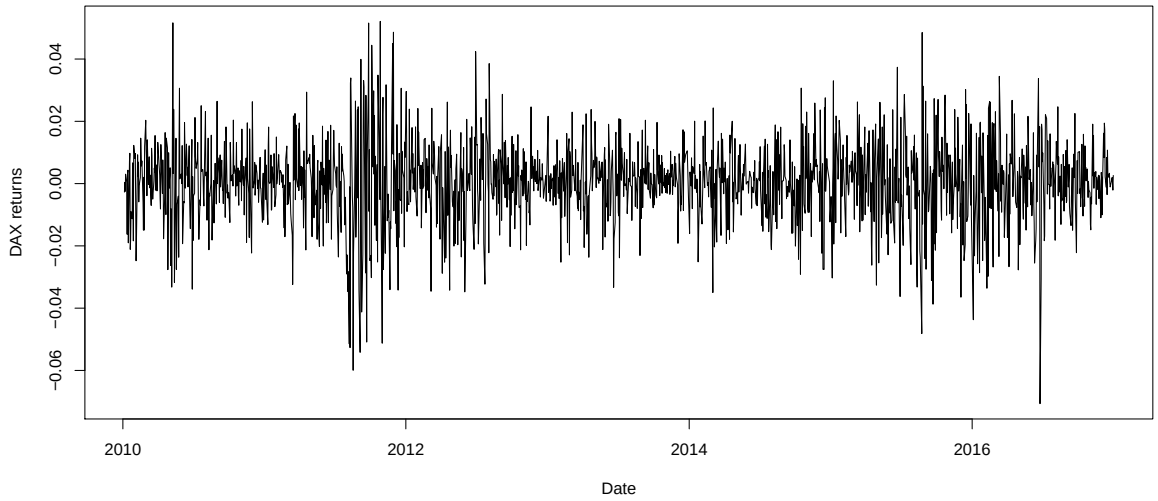

Figure 3: DAX logarithmic returns 2010-2016, covering 1779 values.

This suggests that the data is heteroscedastic, which was to be expected for financial index data. Returns often exhibit "volatility clustering", which means that large financial shocks lead to further large shocks, i.e. high volatility.
To verify that this is the case here, we test for the existence of heteroscedasticity in form of ARCH effects, by applying two tests. First, we will use a score test, with the assumption of normally distributed data, as specified by Engle in [En1]. It is also referred to as the "ARCH LM test", which will be applied for several lag variables here. Then, we will conduct the "Ljung-Box test". More details and the specifications of these tests can be found in the Basics section, while the implementations and some pointers regarding

their computation are given in the appendix.
The ARCH LM test results for the specified lag orders are given as follows:

lag	F_statistic	p_value
3	143.73	0.0000
8	197.04	0.0000
15	212.17	0.0000
25	260.27	0.0000

Table 1: ARCH LM Test on DAX returns.

The p-value is too small to be displayed, hence the test suggests that the null hypothesis, saying no ARCH effects are present, can be rejected at a 5% significance level.
The results derived from the Ljung-Box test are similar:

lag	Q_statistic	p_value
3	196.46	0.0000
8	434.16	0.0000
15	633.29	0.0000
25	992.15	0.0000

Table 2: Ljung-Box Test on DAX returns.

It was quite obvious that the data exhibits volatility clustering/ARCH effects in Figure 3, hence these unambiguous test results are not surprising.

We have seen that the data set shows volatility clustering and is in particular heteroscedastic, but haven't looked into the other distributional properties yet. For this purpose, we firstly check its kurtosis and skewness:

sample kurtosis:
5.2096
sample skewness:
-0.2878

As expected, the DAX returns' kurtosis is higher than the kurtosis of the normal distribution, indicating that the returns have comparatively heavier tails. In particular, this means that the returns exhibit more extreme values, than what would be expected under the assumption of normally distributed data. As the difference in kurtosis is fairly obvious and our sample size is sufficiently large, we can conclude that their distribution is most likely non-gaussian, without testing the returns for normality.
One can also note that the distribution is slightly left-skewed, meaning that the left tail is longer than the right. It can therefore be followed that there are more extreme negative outliers than there are positive ones.
To visualize this, we firstly do a Q-Q-plot, with the estimated sample quantiles against

the quantiles of a normal distribution and a student's t distribution. Then, we compare a histogram of the sample, scaled to the corresponding density values, with the density curves of a fitted normal and t-distribution. The mean and variance of the normal distribution are estimated by maximum likelihood estimation with the sample mean and sample variance. For the (non-standardized) t-distribution, we also use maximum likelihood estimation, approximating the location, scale and degrees of freedom. The resulting estimates are:

	mean/location	standard deviation/scale	degrees of freedom
normal	0.000360	0.013025	
student's t	0.000710	0.009615	4.004476

Table 3: Parameters estimated by maximum likelihood.

We note that the sample mean is slightly larger than zero, which is not surprising, as the DAX nearly doubled over the analysis time period. The relatively small estimate of 4 for the degrees of freedom reflects the heavy tailed distribution of the sample.
To see this, the Q-Q-plots of the normal and student's t distribution, with estimated degrees of freedom, can be checked:

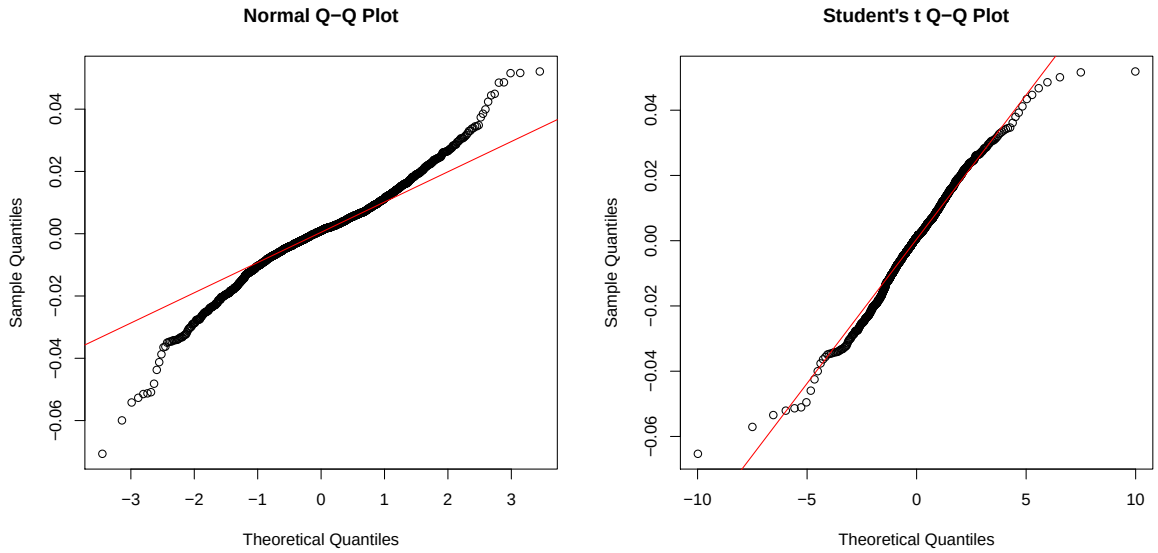

Figure 4: Q-Q-plots of sample quantiles against the standard normal and student's t quantiles

While the normal distribution is completely off and obviously not fitting, the student's t distribution performs a lot better and seems to model the heavy tail property of the returns relatively well.
This is also confirmed, when taking a look at the histogram, scaled to density. We add the density curves of the normal and student's t distribution, as well as a kernel density estimate as a comparison benchmark:

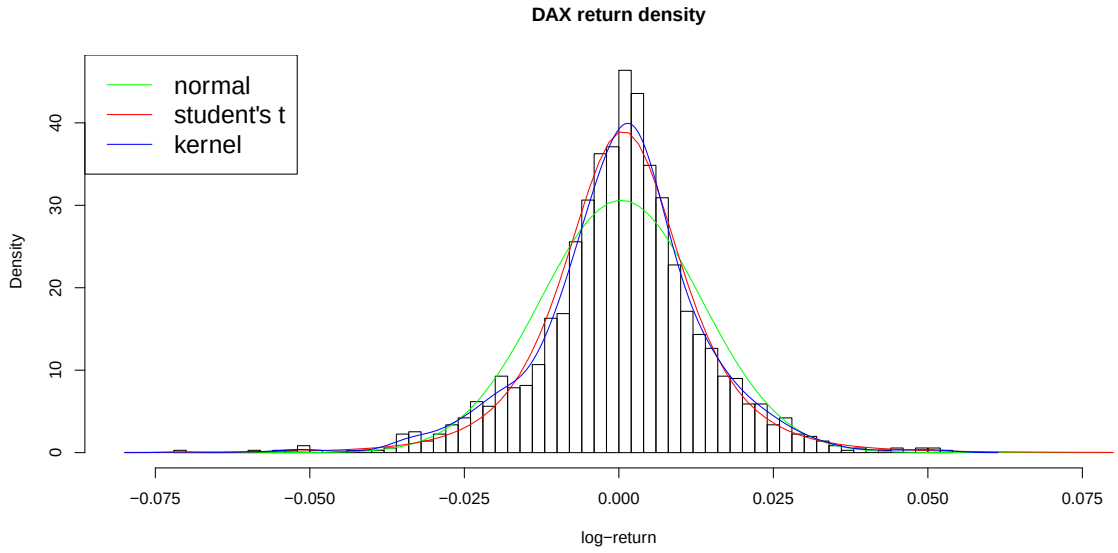

Figure 5: Density Histogram of the DAX returns, with estimates following the normal and student's t distribution, as well as a kernel density estimate (using gaussian kernel function).

The densities were fitted, using the previously estimated mean/location, standard deviation/scale and degrees of freedom. For more details, regarding the here applied non-standardized t-distribution and kernel estimate, check the Basics section.
We can clearly see in the histogram that the normal distribution's shape doesn't fit the sample density very well. While the non-standardized t-distribution looks relatively well-fitting, regarding the tails and distribution's peak, it visibly doesn't account for the negative skew that the sample exhibits. This could have been done better by the skewed t-distribution (see basics section for a skewing method).
It should be kept in mind that these estimates were created under the assumption of non-correlated data, which is not justified, as we have seen. However, they serve our goal to get a better feeling for the data, helping in the upcoming modelling section.

Summarizing the results from the analysis so far, the sample returns are negatively skewed, rather heavy tailed and exhibit volatility clustering in the form of ARCH effects. Under the (wrong) assumption of a non-autocorrelated series, the data's distribution is best approximated by the use of a skewed t-distribution. A normal distribution doesn't represent it sufficiently well, as it exhibits too many extreme returns.
These can be easily identified, by the use of a "leave-one-out" technique on the sample standard deviation, i.e. we compute the effect that every single return has on the overall standard deviation estimate. Their relative influence can be visualized by the following plot:

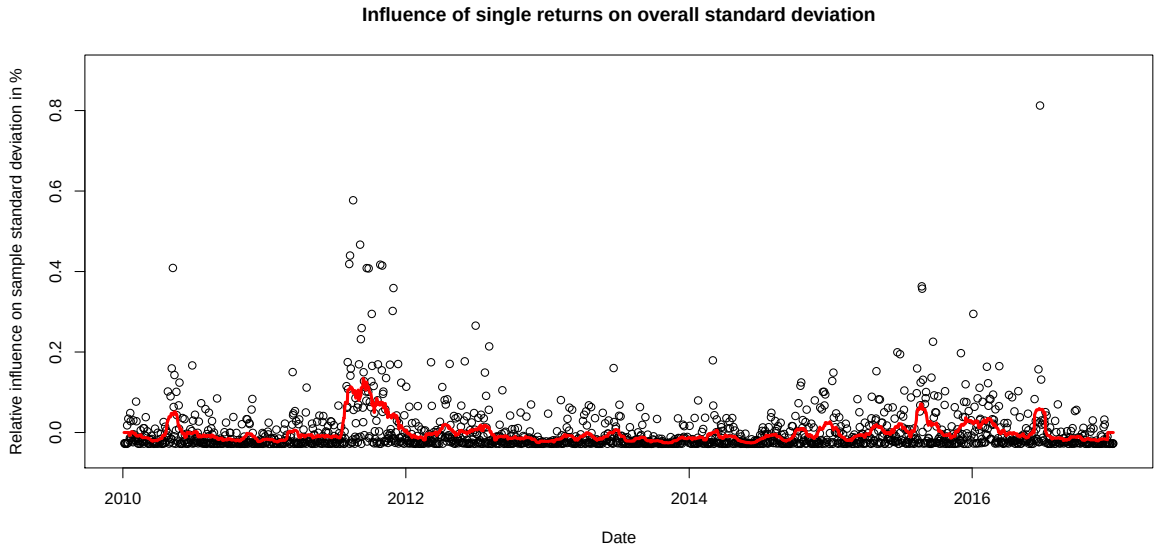

Figure 6: Relative, pointwise influence of the returns on the sample standard deviation in %. Positive values imply that the sample standard deviation would be smaller without this return and negative ones mean that it would be larger.
The red line indicates the pointwise sample mean over the surrounding 50 values for each time point.

A "relative influence" of e.g. 0.5% means that the standard deviation would be 0.5% lower without this return value. It can be seen that the bulk of points have a very small, negative influence on the sample standard deviation, while a few values have a very large influence of up to nearly 1%. This is plenty, if we consider the sample size of 1779 returns. For an overview and to identify these extreme values, we rank the ten largest ones with the respective dates and log returns:

Influence Rank	Index	Date	(log) return in %	Relative influence in %
1	1646	2016-06-23	-7.067	0.812
2	417	2011-08-17	-5.995	0.577
3	429	2011-09-02	-5.419	0.467
4	411	2011-08-09	-5.268	0.440
5	409	2011-08-05	-5.148	0.419
6	467	2011-10-26	5.210	0.417
7	470	2011-10-31	-5.125	0.415
8	88	2010-05-07	5.163	0.409
9	442	2011-09-21	-5.089	0.409
10	445	2011-09-26	5.157	0.408

Table 4: Returns ranked, with respect to influence on sample standard deviation

The returns are chosen as the DAX return between the given date and the following day - e.g. for the highest ranked value, we take the log return between the 23rd and 24th June 2016.
Firstly, one can note that the highest value is quite a lot bigger than the lower ones. Its cause is of political nature, as the outcome of the United Kingdom's EU membership referendum on that day was highly unexpected by the market participants.
This shock was rather short-natured, though, in comparison to the fluctuant time period between august and october 2011. Most listed values fall into this time-frame that was considered the apex point of insecurity, caused by the European sovereign debt crisis. During this time, there was substantial fear of a spillover of the crisis in Europe to the rest of the world, leading to plummeting stock markets worldwide and a downgrading of the US's credit rating for the first time ever.
The remaining positive shock on the 7th May 2010 is also attributed to the "Euro-crisis", being the day when the European Union decided for a united financial assistance program for Greece that was on the verge of bankruptcy.
These notable events should be kept in mind for the upcoming volatility analysis, as they foreshadow higher volatility estimates during those time periods.

Lastly, we want to check, whether the DAX returns exhibit a leverage effect, as defined in the basics section. We have noted that this can be done by the use of the sample correlation coefficient (1.1) on the returns and the subsequent squared returns. The values for the sample correlation coefficient and the boundaries of the 95% confidence interval, in which the coefficient should be under the null hypothesis of no correlation, are:

Sample correlation coefficient:	-0.1229
Boundaries (+/-):	0.0465

As the coefficient is outside the range, we can reject the null hypothesis of no correlation at a 5% significance value and find that there is a negative correlation between the two quantities. This suggests the presence of the leverage effect property in our return sample.
In addition to this, we compute the sample mean and standard deviation of a time series of lagged return values, resulting in a form of "local" mean and standard deviation. The first is also referred to as a "(simple) Moving Average" (should not be confused with the MA model) and used in many financial indicators, as a tool for technical analysis of stock prices.
The following graph shows the relation between these two series:

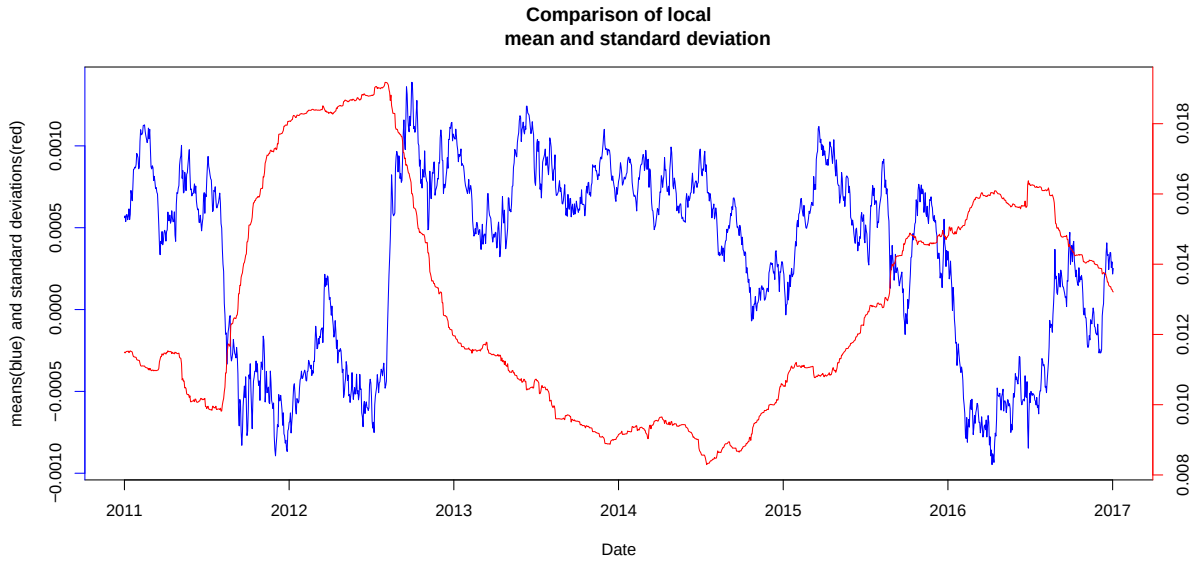

Figure 7: Relationship between simple moving average (blue) and "local standard deviations" (red) for a lag length of 254, corresponding to about one year of data. Hence, there is no data for 2010, as it is the "burn-in" time period. The respective associated y-scales are denoted by color.

Their calculation is done as follows, for both cases:

$$TS_f(t) = \frac{1}{n}\sum_{i=0}^{n-1} f(y_{t-i}). \tag{2.1}$$

If $f(\cdot)$ is the identity function, we get the simple moving average, for $f(y) = \frac{n}{n-1}(y-\bar{y})^2$, with $\bar{y}$ being the respective value of the moving average, results in the local (unbiased) variance series. Taking the square root gives the desired standard deviation estimate (but reintroduces a bias).
In the graph, we can see that time periods with positive returns, on average, also exhibit a relatively low volatility - here indicated by the local standard deviation, with lag length of a year. It becomes relatively obvious that there is a negative correlation between returns and volatility. Therefore, not only the absolute value of a return matters, but also its sign, with negative returns leading to sharper increases in volatility than positive ones, which rather tend to decrease it.
When using more elaborate models for the volatility, than the previously mentioned "local standard deviation", we might be able to document this effect better. For now it is sufficient to acknowledge the existence of a leverage effect in our DAX return sample, though, as this is just a provisional analysis, preparing for the now following section about volatility modelling.

Our goal of this section was to analyze the basic properties of the DAX return data sample. Most of the properties we found are typical for financial returns - we have seen

that the data exhibits heavy tails, a (negative) skew, the leverage effect and volatility clustering in the form of ARCH effects. The data's distribution, unconditional on time, can best be described by a skewed t-distribution, while the normal distribution performs rather bad.
Using this knowledge, we will now follow up with the volatility modelling sections.

3 The Models

After handling the necessary preliminaries, we can now proceed with the main part of the volatility analysis. We will start off with the specification of every model type that will be used, including some pointers regarding their motivation and how the actual estimation of the models will be approached.
As it has been mentioned in the Introduction, this will encompass the popular GARCH model and several subtypes, like GARCH-t, EGARCH, TGARCH and GJR-GARCH. But also less widely known models are used, like the t-GAS and Beta-t-EGARCH models, which take a different approach in volatility updating, involving the predictive score function. Last but not least, SV models are covered, which model volatility in a non-deterministic way.
Following the introduction, we firstly compare the different approaches in an overview and check, which effects can be handled by each respective model. In particular, we will summarize what their driving components are and if they account for the observed properties of our sample, like the heavy tails and leverage effect.
Lastly, we follow up with a short summary of the estimation results, discussing some of the (computational) difficulties that may be or actually were encountered during the modelling process.
The comparison between the models and testing their performance will be addressed in the next section.

3.1 Model Specification, Motivation, Properties and Estimation

For every subsection, we define $\{y_t\}_{t=1}^n$ as the time series sample of size n, whose time-varying standard deviation σ_t we want to model. Its true conditional density is given by $y_t \sim p\left(y_t \mid F^t, Y^{t-1}\right) =: p_t$, where Y^{t-1} denotes the past observations $\{y_1, \ldots, y_{t-1}\}$ and F^t is defined as the set of the time-varying parameters at time t, including σ_t.
Our estimates for the true conditional density p_t will be defined as
$\tilde{p}_t := \tilde{p}\left(y_t \mid \tilde{F}^{t-1}, Y^{t-1}; \Theta\right)$. Here, $\tilde{F}^{t-1} = \{\tilde{\sigma}_0, \tilde{\sigma}_1, \ldots, \tilde{\sigma}_{t-1}\}$ contains the filtered estimates for σ_t up to time t and Θ is the set of estimates for the other possibly existing (fixed) parameters. The model's initial guess for the standard deviation is defined as $\tilde{\sigma}_0$.
It should be remarked that we assume only the sample's standard deviation as time-varying, while the rest of the parameters remain fixed and have to be estimated only once, e.g. through maximum likelihood estimation.
At time t, we have not yet observed y_t, but know the previous observations Y^{t-1}, as well as our estimates $\tilde{F}^{t-1}$. We use this knowledge, to estimate σ_t, by updating our estimate to $\tilde{\sigma}_t$ with an updating function $\phi(\cdot)$, i.e. $\phi(\tilde{F}^{t-1}, Y^{t-1}; \Theta) = \tilde{\sigma}_t$.
Note that not all previous estimates are necessary in this specification, as it would be sufficient to know $\{\tilde{\sigma}_0, Y^{t-1}\}$, to estimate every remaining $\tilde{\sigma}_i$ in $\tilde{F}^{t-1}$ with the updating function. Hence, the estimates $\tilde{\sigma}_0, \ldots, \tilde{\sigma}_t$ can be considered a filtered time series, with respect to any starting value $\tilde{\sigma}_0 > 0$.
To make notation more simple, we will omit the "$\sim$" here, i.e. we write, as if the model

was correctly specified in every case. This is obviously not the case, as the true model is unknown and can only be approximated. Now, we firstly cover each model seperately and then summarize the results in an overview.

3.1.1 "Moving sample standard deviation"

Specification
We already employed a (rather primitive) way of modelling the volatility in the previous section, with the use of the sample standard deviation time series, given by

$$\sigma_t = \sqrt{\frac{1}{n-1}\sum_{i=0}^{n-1}\left(y_{t-i} - \bar{y}_t\right)^2} \quad \text{, with} \quad \bar{y}_t = \frac{1}{n}\sum_{i=0}^{n-1} y_{t-i}. \tag{3.1}$$

It can also be thought of, as the root of a simple moving average on the demeaned returns, which are later used as a volatility proxy, to assess forecasting quality.

Motivation&Properties
The distinctive advantage of this model is its simplicity, as it doesn't require much computation, unlike other models. However, there are several disadvantages:
First of all, the statistical prerequisites are not met in our data sample, as it isn't independently distributed, in particular exhibiting autocorrelation. Hence, the model has little statistical legitimacy as an estimator for the standard deviation and can only serve as an approximation. Secondly, the choice of the lag factor n is crucial for the outcome. If the value is too big, the volatility estimate responds too slowly and falls behind the actual realized volatility. A value that is too small, gives the opposite problem and additionally exhibits a greater bias, as the bias decreases in n. Lastly, this model needs n values before the beginning of the series, i.e. it can have a rather large burn-in phase and performs badly in the very beginning.

Estimation
As the optimal choice of the lag factor n is not obvious, we will check several values, in order to compare them. Which one should be preferred, depends on whether one wants to model short-term or long-term-volatility. The following graph shows the estimates for several lag values, pointing out the model's reducing responsiveness with increasing lag:

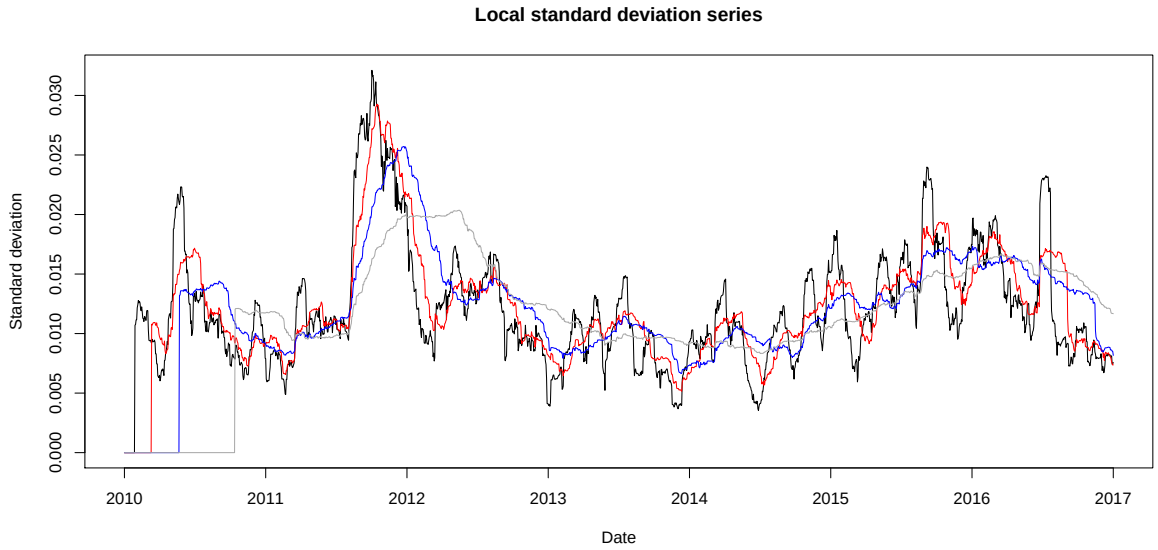

Figure 8: Moving sample standard deviation for the lag values 20 (black), 50 (red), 100 (blue) and 200 (gray). The burn-in phases of the estimators are set to zero.

Even though it is obviously not a very good estimate, we can at least get a feeling for the range of the volatility that is to be expected. Several prominent peaks can be observed in the estimate with 20 lags, three of which fall into periods that we already described in the previous section: The near-bankruptcy and financial assistance program for Greece in May 2010, the stock market crash in the end of 2011 and the "Brexit" referendum on June 2016.
Keeping this in mind, we proceed to the "actual" volatility models, omitting this one in the analysis in the end. We go on with the most famous group of volatility models.

3.1.2 GARCH

Specification
Introduced in [Bo1], the standard GARCH(1,1) model is probably the most popular way of modelling autoregressive heteroscedasticity. For a sample y_t, it is specified by the two time series

$$y_t = \mu + \epsilon_t \sigma_t, \text{ with } \epsilon_t \overset{iid}{\sim} \mathcal{N}(0,1) \tag{3.2}$$

and

$$\sigma_t^2 = \omega + \alpha(y_{t-1} - \mu)^2 + \beta\sigma_{t-1}^2, \tag{3.3}$$

where $\omega, \alpha, \beta > 0$ affect the updating behavior of the model's standard deviation and μ denotes the mean of y_t. While α quantifies the impact of new observations on the standard deviation, β describes the influence of past estimates and ω is a sort of intercept/linear trend term.

To ensure that the process has a finite variance, we only consider $\alpha+\beta < 1$, as $\alpha+\beta = 1$ would result in the Integrated GARCH model, which is persistent. This means that the effect of past shocks doesn't wear off, as a unit root is introduced, which makes the process non- covariance stationary. The resulting properties from persistence and the introduction of the IGARCH model is given in [EB1].

Motivation&Properties
The intuition behind the GARCH model is relatively simple. The model implied distributions for each observation at time t is given by $p(y_t \mid \sigma_t) \sim \mathcal{N}(\mu, \sigma_t^2)$, i.e. the sample follows a normal distribution with mean μ and a time-varying variance. To model this variance, the driving factor $(y_t - \mathbb{E}[y_t])^2$ is used in (3.3). This seems sensible, as its expected value is, by definition, the variance: $Var(y_t) = \mathbb{E}\left[(y_t - \mathbb{E}[y_t])^2\right]$.
If we were to model a sample after discounting the mean, the factor $(y_t - \mathbb{E}[y_t])^2$ could be replaced by y_t^2. On the one hand, this would result in one parameter less to estimate. On the other hand, excluding μ from the model and instead demeaning the data, might result in less exact estimation, affecting the bias of quasi maximum likelihood estimators. For the case of MA models, this has been discussed in [Ba1]. Hence, we include the mean in some of our models.

Considering the model implied distribution, it can be said that the standard GARCH model does not attribute to the heavy tails and negative skew we observed in out return data sample. Hence, the results of the QMLE might be inaccurate.
Another downside is that the driving factor of the updating function (3.3) uses squared sample values. Thus, the effect of extreme outliers is heavily amplified and might lead to overestimation of σ_t^2.
Furthermore, the model doesn't account for the leverage effect in our data, as negative returns are valued the same as positive ones, i.e. only the absolute size of the returns matters.

Estimation
As the true distribution of our data is unknown, we don't have access to the true likelihood function. Thus, we approximate it by our model's likelihood function instead. The fixed coefficients μ, ω, α and β, as well as the starting value for the standard deviation σ_0, are then estimated by quasi-maximum likelihood estimation (QMLE), i.e. we maximize the log likelihood

$$\begin{aligned} \ell(\mu, \omega, \alpha, \beta, \sigma_0) &= \sum_{t=1}^{n} \ell_t(\mu, \omega, \alpha, \beta, \sigma_0) = \\ &= \sum_{t=1}^{n} -\frac{1}{2}\log\left(2\pi\sigma_t^2(\mu, \omega, \alpha, \beta, \sigma_0)\right) - \frac{(y_t-\mu)^2}{2\sigma_t^2(\mu, \omega, \alpha, \beta, \sigma_0)}. \end{aligned} \tag{3.4}$$

The variance function σ_t^2 is the one specified in (3.3) for values $t = 1, \ldots, n$.
Implementation of the estimation process in R is described in the Appendix and manually executed. It slightly varies from the usually used "rugarch" package in R (see [Gh1]),

as it includes the starting value σ_1 in the quasi maximum likelihood estimation, while the implementation of the rugarch package chooses the sample standard deviation as its starting value.
The difference can be seen in the following graph, showing the estimates for the time-varying standard deviation of our model:

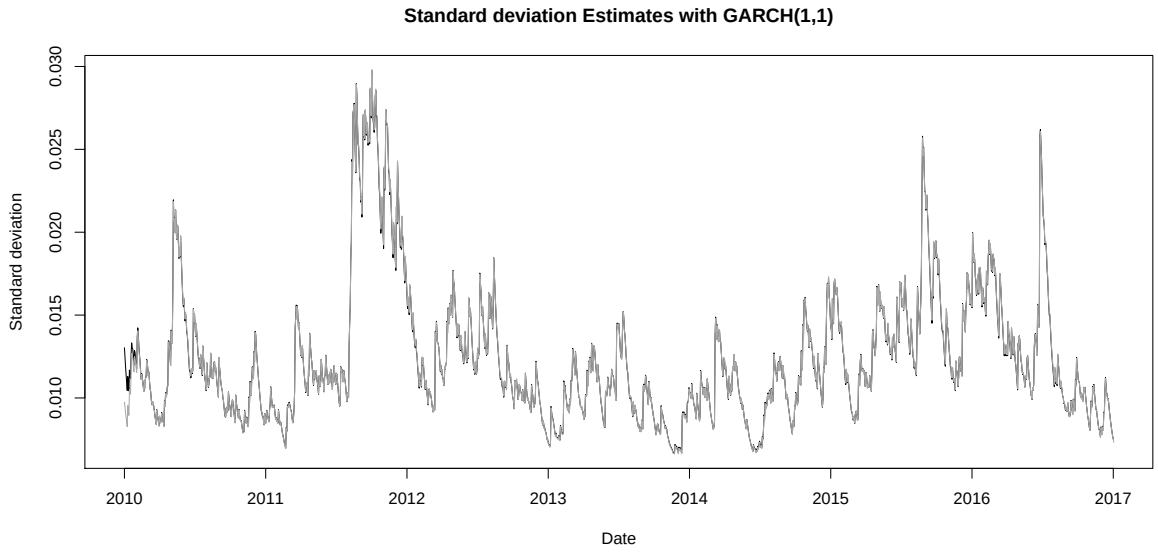

Figure 9: Resulting estimates for the standard deviation with the GARCH(1,1) model. The grey line denotes the own implementation with more appropriate starting value and the black line is from the "rugarch" package implementation.

Besides the first few estimates, the outcome is the same for both implementations. If one includes the starting value σ_1 in the QMLE, the burn-in phase can be avoided. This is favorable for us, as we want to model the standard deviation for the whole time-frame, including the beginning. If we added a few return samples from 2009 in the estimation process and omitted the estimation results before 2010, we could exclude the starting value, though, resulting in one parameter less to estimate.

When computing the maximum of a (pseudo-) likelihood function with many variables, it can be troublesome to find the global maximum, as there may be many local maxima, masking the best choice, or discontinuities resulting in a computational dead end. The computation of the GARCH coefficients was not very troublesome, though, as a relatively robust optimization function was used and appropriate starting values were easy to find. Typical for financial returns is an α value around 0.1 and a β value of about 0.9, with their sum $\alpha + \beta$ being smaller, but very close to 1. The actual R code and more details are given in the appendix.
One might further consider GARCH models of higher order, or some that model the mean sequence with an ARMA process. However, that is not necessary here, as we will see in the performance analysis section. For now, we proceed with the next GARCH

subtype.

3.1.3 GARCH-t

Specification
The specification of the GARCH-t(1,1) model proposed in [Bo2], is basically the same as the one of the standard GARCH(1,1) model. The only difference is in the distribution of the error terms:

$$y_t = \mu + \epsilon_t \sigma_t, \text{ with } \epsilon_t \overset{iid}{\sim} t_\nu \tag{3.5}$$

and

$$\sigma_t^2 = \omega + \alpha(y_{t-1} - \mu)^2 + \beta\sigma_{t-1}^2. \tag{3.6}$$

Here, t_ν denotes the non-standardized t-distribution with ν degrees of freedom and unit variance. The model implied conditional distribution of y_t is hence given by the non-standardized t-distribution with mean μ, variance σ_t^2 and ν degrees of freedom, whose density we specified in (1.17).

Motivation&Properties
The motivation for this adjustment is rather obvious and addressed to in the GARCH subsection: In order to accommodate the model to heavy-tailed data, a distribution with possibly heavy tails is chosen. This gives better QMLE results for our sample.
However, the downside is that this results in one more parameter to estimate. As our data is obviously not normally distributed, the use of the t-distribution still is much more sensible, though.
It should be noted that this variation of the standard GARCH model also doesn't handle the skew in the data, as well as the leverage effect. Additionally, the updating function (3.6) remains unchanged, still not reacting very robust to outliers.

Estimation
Similarly to the standard GARCH model, we use QMLE for the parameters $\mu, \omega, \alpha, \beta, \sigma_0$ and now also ν, for the degrees of freedom of the t-distribution. Hence, we maximize the log likelihood

$$\begin{aligned}\ell(\mu, \omega, \alpha, \beta, \sigma_0, \nu) = \sum_{t=1}^{n} \log\left(\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{(\nu-2)\pi\sigma_t^2(\mu, \omega, \alpha, \beta, \sigma_0)}\Gamma\left(\frac{\nu}{2}\right)}\right) \\ -\frac{\nu+1}{2}\log\left(1 + \frac{(y_t - \mu)^2}{(\nu-2)\sigma_t^2(\mu, \omega, \alpha, \beta, \sigma_0)}\right)\end{aligned} \tag{3.7}$$

that we acquired through the corresponding density function (1.17), with σ_t^2 again defining the model's variance like in (3.6).
As seen in the standard GARCH model, the implementation from the "rugarch" package doesn't include the starting value in its QMLE. Hence, we use the selfmade estimate. To

find out, whether the change in model implied distribution had a substantial influence on the resulting estimates, we take a look at the following graph with both estimates for σ_t:

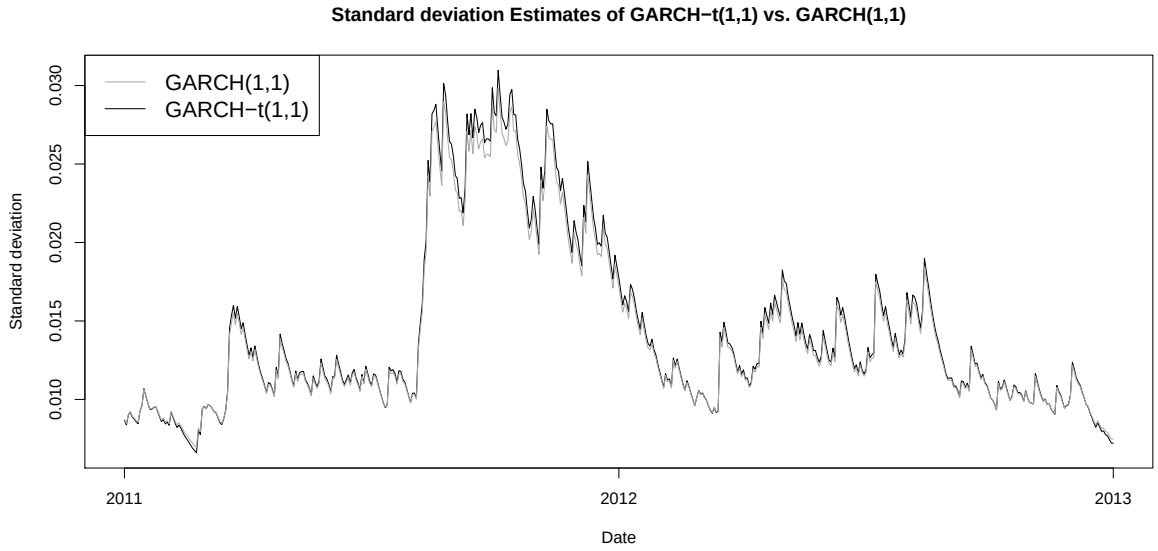

Figure 10: Resulting estimates for the standard deviation with the standard GARCH(1,1) model (grey) and the GARCH-t(1,1) model (black); The time-frame ranges from 2011 to 2013.

We can see that the GARCH-t estimates result in an updating function that is reacting stronger to the change in observations. It gives higher estimates during the high-volatility time in the end of 2011, but lower ones in the low-volatility times with estimates $\sigma_t < 0.01$. This is caused by the larger α and smaller ω values of the GARCH-t model (the exact values can be found in the parameter table in the appendix). The difference between both is not that big, though, with a mean absolute value of the procentual difference of about 1.8% and a maximum absolute difference of 6.2% if one disregards the first 25 estimates, which are very different due to dissimilar initial estimates for σ_0. Whether this change is an improvement or not will be seen in the next section.

Actually computing the estimate is again not very complex, as the process only differs in the choice of the log likelihood function. The new coefficient ν makes it slightly more difficult and adds the possibility for discontinuities through the terms $\frac{1}{\sqrt{\nu-2}}$ and $\frac{1}{\nu-2}$ in the log likelihood. However, these can be avoided by including a lower bound for ν, making it greater than 2 and the use of sensible starting values. More details regarding the computation can be found in the Appendix.

3.1.4 EGARCH

Specification

In Contrast to the GARCH-t model, the exponential GARCH model actually involves

a new updating function. It was introduced in [Ne1] and shall specifically attribute to the properties of asset return data. As we have mentioned in the GARCH-t subsection, a t-distribution is more sensible for our analysis. Hence we assume t-distributed errors again. The specification for the EGARCH(1,1) model is then given by

$$y_t = \mu + \epsilon_t \sigma_t, \text{ with } \epsilon_t \overset{iid}{\sim} t_\nu \tag{3.8}$$

and an updating function

$$\log\left(\sigma_t^2\right) = \omega + \alpha z_{t-1} + \gamma\left(|z_{t-1}| - \mathbb{E}\left[|z_{t-1}|\right]\right) + \beta \log\left(\sigma_{t-1}^2\right). \tag{3.9}$$

While the first equation is the same as in (3.5), the updating equation greatly differs. In his introductory work, Nelson specified the updating function with variables z_t that follow a generalized error distribution with mean zero and unit variance, without including the mean equation (3.8), i.e. he specified the general case. Here, the scaled observations z_t (with zero mean and unit variance) are used, which are given by $z_t = \frac{y_t - \mu}{\sigma_t}$ and follow a t-distribution.
What remains unchanged, compared to the standard GARCH, are the coefficients ω and β, which take the same role as in the standard GARCH model. Thus, ω is a constantly added term and β the GARCH factor, determining at which rate the influence of past estimates decays.
As a novelty, the non-squared observations are used in αz_{t-1} . This results in possibly negative factors in the updating equation and allows for a sign effect. In particular, this means that, for negative α values, negative z-values increase the volatility and positive ones decrease it - resulting in a leverage effect - if α is sufficiently bigger than γ.
The term $\gamma\left(|z_{t-1}| - \mathbb{E}\left[|z_{t-1}|\right]\right)$ is responsible for the size effect of the scaled observation. For large enough values of $\gamma > 0$ (compared to α), this theoretically allows positive returns to increase the volatility estimate for sufficiently large returns, instead of decreasing it in every case. It can also be noted that absolute terms are used, in contrast to the squared terms in the previously covered GARCH models.
Lastly, through the use of a logarithmed updating equation, negative values for σ_t^2 are made possible without invalidating the need for positive variance or standard deviation. However, this introduces an estimate bias, as can be seen by the use of Jensen's inequality for the (convex) exponential function:

$$\mathbb{E}\left[\sigma_t^2\right] = \mathbb{E}\left[\exp\left(\log\left(\sigma_t^2\right)\right)\right] \geq \exp\left(\mathbb{E}\left[\log \sigma_t^2\right]\right). \tag{3.10}$$

Motivation&Properties
The EGARCH model is designed to model asset returns through an adjustment in the updating function. A possibly existing leverage effect is accounted for by the term αz_{t-1}. When it is present, α is estimated as smaller zero and gives returns smaller than the model's mean a greater positive impact on the model's (logarithmed) variance.
Furthermore, the usually heavy tails of return data might lead to an overestimation of the variance, if squared observations are used as a driving factor, like in the aforementioned models. As absolute values are used here, one might think that more conservative

variance updates are to be expected in the case of unusually high returns. However, this is not true, as one has to consider that the updating equation is about the logarithmed variance. Thus, the updating steps can be even larger than the ones of the standard GARCH updating equation. In effect this increases the responsiveness, but might be too sensitive to large absolute returns.
Lastly, it should be mentioned that we specified the EGARCH model with t-distributed errors, as we have seen that it is more fitting and probably results in better QMLE results. The use of another heavy tailed distribution with skew might result in even better estimates - however, as the skewness is not too large, we omit this possibility here, for easier modelling.

Estimation
Estimation for the EGARCH model specified in (3.8) and (3.9) is only slightly more difficult than the one for the GARCH-t model. It incorporates an additional variable and another nontrivial term that has to be analytically calculated first: The expected value of $|z_{t-1}|$. Its result is given for the generalized error distribution in [Ne1] and specifically for the t-distribution in [Ts1], as

$$\mathbb{E}\left[|z_t|\right] = \sqrt{\frac{\nu-2}{\pi}}\frac{2}{(\nu-1)}\frac{\Gamma((\nu+1)/2)}{\Gamma(\nu/2)}, \tag{3.11}$$

which can be simplified to

$$\mathbb{E}\left[|z_t|\right] = \sqrt{\frac{\nu-2}{\pi}}\frac{\Gamma((\nu-1)/2)}{\Gamma(\nu/2)}. \tag{3.12}$$

If we had used a gaussian distribution, we would get $\mathbb{E}[|z_t|] = \sqrt{\frac{2}{\pi}}$ instead.

The log likelihood function can be taken from (3.7), as we still have the same model implied distribution. The only difference is the added dependency on γ, which is added through the function $\sigma_t^2(\cdot)$.
As for the previous subsections, we employ a selfmade computation that involves the starting value σ_1 in the QMLE. Besides the change in the updating function, the computation process is exactly the same as for the GARCH-t model. The only additional thing one has to look out for is the α value, as too large values lead to an explosive behavior in the σ_t^2 estimates, impairing the optimization process.
Lastly, we take a look into the resulting estimates for the volatility and compare it to the GARCH-t estimates in the following figure:

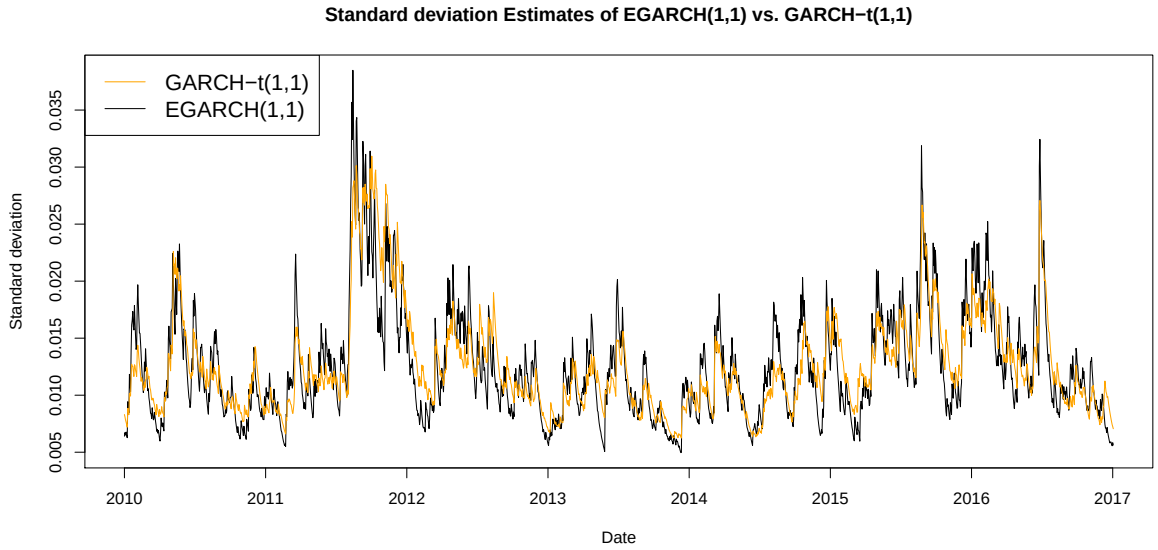

Figure 11: Resulting estimates for the standard deviation with the GARCH-t(1,1) model (orange) and the EGARCH(1,1) model (black)

It is obvious that the EGARCH estimates greatly differ from the GARCH and GARCH-t results. As expected, they are more responsive to returns of all sizes, as the updating equation is specified for the logarithmed variance. This leads to a faster increase during high-volatility periods, but also to quicker decreases during low-volatility phases and hence gives less conservative estimates. We can also see that the shocks, caused by the Brexit referendum and European debt crisis, led to notably larger estimates than in the GARCH-t model. This would suggest the existence of a leverage effect. However, taking into account that the positive shock in May 2010 didn't lead to unrealistically low volatility estimates, it can be seen that positive returns don't decrease the estimates in every case.
If this outcome is more exact than the GARCH-t model's, will be evaluated in the next section.

3.1.5 GJR-GARCH

Specification
This subsection will cover the Glosten-Jagannathan-Runkle (GJR-)GARCH model, introduced in [GJR1].
As with the previous models, we will use a non-standardized t-distribution, in order to get a solid approximation for the true return distribution. The idea behind the model's updating function is quite simple, as the only difference to the standard GARCH one is the introduction of an indicator function, in order to model asymmetric sign effects. Its specification is hence given by

$$y_t = \mu + \epsilon_t \sigma_t, \text{ with } \epsilon_t \overset{iid}{\sim} t_\nu \tag{3.13}$$

and

$$\sigma_t^2 = \omega + \alpha(y_{t-1} - \mu)^2 + \beta\sigma_{t-1}^2 + \gamma(y_{t-1} - \mu)^2 \mathbb{1}_{\{y_{t-1}-\mu<0\}}. \tag{3.14}$$

The mean equation's specification is again analogous to the ones in the previous sections, as well as the role of ω, α and β. Through the use of the indicator function, the factor γ only comes into effect if the respective return is smaller than its mean, i.e. $y_{t-1} < \mu$.

Motivation&Properties
Basically the only difference of aforementioned specification to the GARCH-t(1,1) model is the indicator term, hence the motivation behind this model is quite obvious. It shall make the model more flexible, allowing for an asymmetric influence on the model's variance, depending of the sign of the observation (assuming the mean is negligibly small). Therefore, this model is capable of modelling leverage effects, when γ is bigger than zero, with otherwise similar properties to the GARCH-t model.
As the term only allows for one sign to be more influent and doesn't cause the other to have an opposite effect, it can not be considered to allow for "pure" leverage effects, even in the most extreme case of $\alpha = 0$. In that case, all positive returns slightly reduce the volatility estimate at a constant rate, depending on β, but the decrease is independent from the size of the positive returns and hence doesn't correspond to a standard leverage effect. This might prove useful, to find out, whether the returns exhibit the standard leverage effect or just an asymmetry - in the first case, this model should perform slightly worse than models, which allow positive returns to actively decrease the volatility.

Estimation
As only one term was added to the updating function, the computation process is nearly the same as for the GARCH-t model. The model implied log likelihood function is the same and only γ is added as a new variable in the QML optimization process. The used implementation can be found in the appendix.
The way that the EGARCH and GJR-GARCH models handle leverage effects differs. Hence, it is of interest, if they result in substantially different estimates:

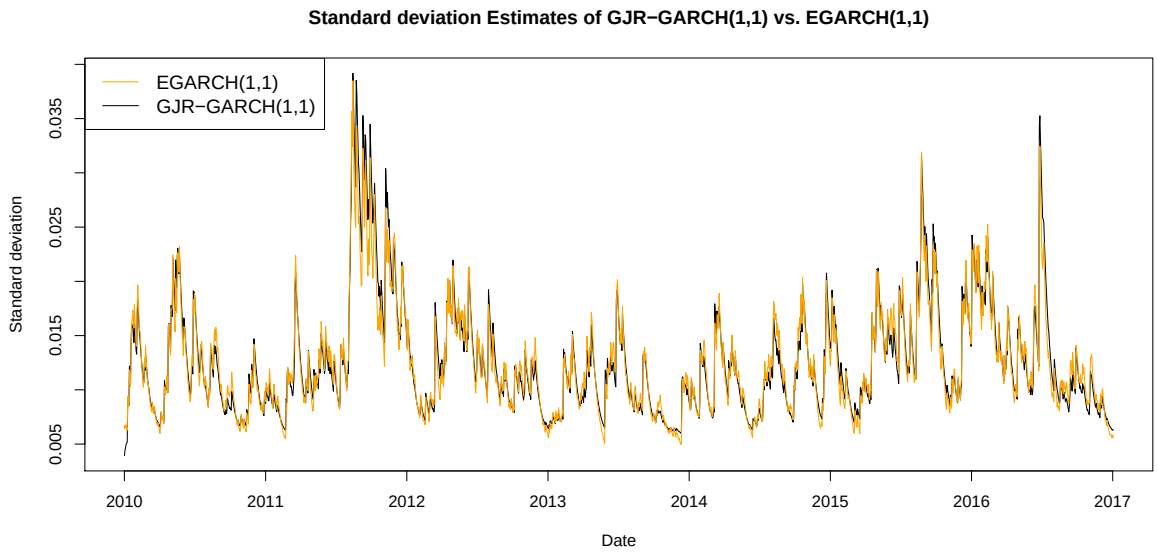

Figure 12: Resulting estimates for the standard deviation with the EGARCH(1,1) model (orange) and the GJR-GARCH(1,1) model (black)

We can see that the two results vary only a little - mostly during the volatility spike in the end of 2011, caused by the European debt crisis. There, the EGARCH model seems to drop more quickly than its competitor, after the first rapid increase. This might have been caused by the previously mentioned difference in leverage effect modelling. A more detailed evaluation will follow later on. For now, we proceed with the next model used.

3.1.6 TGARCH

Specification

We will cover our last "direct" subtype of the standard GARCH model in this subsection: The Threshold GARCH (TGARCH) model, which was introduced in [Za1]. As it follows pretty much the same principle as the GJR-GARCH model, we will keep this section rather short. The distinctive feature of its specification is that it directly models the standard deviation, not the variance.

We use the non-standardized t-distribution, resulting in a mean equation like in (3.5) and combine it with the TGARCH updating equation

$$\sigma_t = \omega + \alpha^+(y_{t-1} - \mu)\mathbb{1}_{\{y_{t-1}-\mu>0\}} + \alpha^-(y_{t-1} - \mu)\mathbb{1}_{\{y_{t-1}-\mu\leq 0\}} + \beta\sigma_{t-1}. \tag{3.15}$$

Here, ω, α and μ play the same roles as before. In order to get sensible updating effects and prevent negative standard deviations, $\alpha^+ > 0$ and $\alpha^- < 0$ have to be imposed, as we use non-squared observations. This way, the equation obviously allows for differently sized updates, depending on the sign of the observation, while ensuring positive outcomes for the model's conditional standard deviation for $\omega > 0$. However, just like in the GJR-GARCH model, even large positive returns can at most reduce the standard deviation at rate β (disregarding ω), as a negative α^+ might violate the positivity of σ_t

after a series of positive returns.

Motivation&Properties
All points of GJR-GARCH apply here, too. It is yet unsure, if the use of the standard deviation instead of the variance actually contributes anything or makes any palpable difference.

Estimation
The computational process is very similar to the one of the previous model's and due to the parameter restrictions, which ensure positivity of our volatility estimate, the optimization function converges without a problem. As we can see in the following graph, there seems to be a slight difference between the GJR- and TGARCH model throughout the whole time-frame:

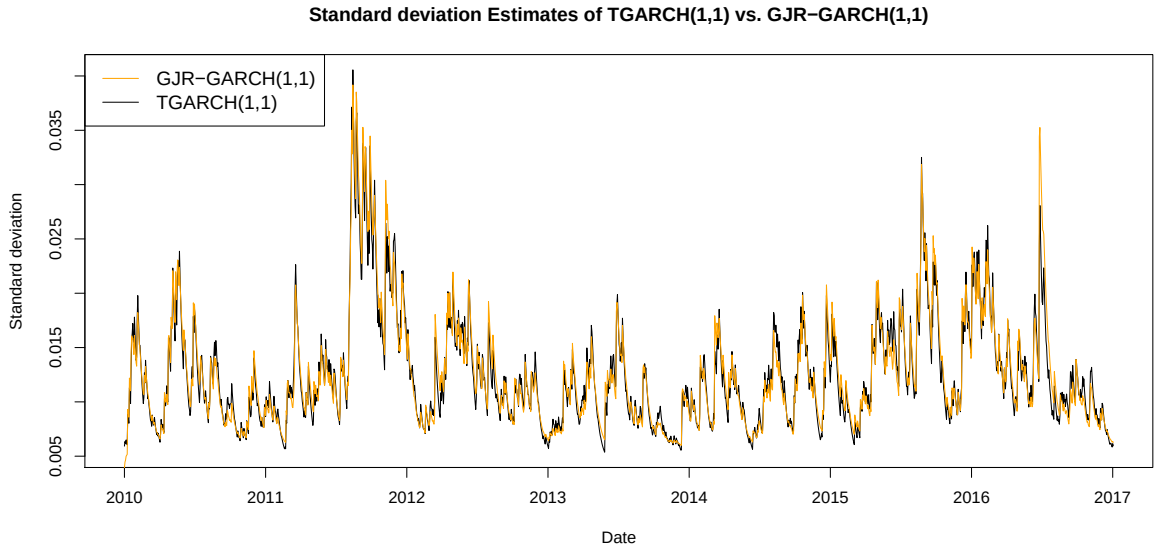

Figure 13: Resulting estimates for the standard deviation with the GJR-GARCH(1,1) model (orange) and the TGARCH(1,1) model (black)

This difference is probably caused by the differing time-varying parameter that is modelled, as the GJR updating function is optimized for the variance equation, while the TGARCH one is directly modelling the variable of interest - the standard deviation. Which one out of the three GARCH models that can handle leverage effects is best suited to model our data, remains yet unclear and has to be examined later on. We will now leave the basic GARCH framework and move on to relatively new ways of modelling heteroscedastic data, involving a new driving factor for the updating function.

3.1.7 GAS

Specification

In contrast to the GARCH models we covered so far, the GAS model, which was widely published in [CKL1], doesn't rely on some power of the demeaned observations as a driving factor. Its specification is kept very general and can result in a multitude of models. What these resulting models have in common, though, is that their driving factor is in all cases the score function, which is derived from the model implied log likelihood. In particular, this means that the driving factor of the updating function depends on the model implied distribution. Thus, the choice of error distribution doesn't just affect the QMLE process, but also the form of updating equation in use. To illustrate this, one may consider the case of normally distributed errors vs. student's t-distributed ones.

We start off, by giving the model's general specification for any model distribution, using the terminology defined in the beginning of this chapter:

Let y_t denote our return data, coming from the true, unknown distribution with density $p(y_t \mid F^t, Y^{t-1})$, conditional on the true time-varying parameter data $F^t = \{f_1, \dots, f_t\}$ and the past observations $Y^{t-1} = \{y_1, \dots, y_{t-1}\}$.

Then, the GAS(p, q) model is given by:

$$\begin{aligned} &y_t \sim \tilde{p}(y_t \mid \tilde{f}_t, \mathcal{F}^t; \Theta), \text{ with} \\ &\mathcal{F}^t = \{Y^{t-1}, \tilde{F}^{t-1}\},\ \tilde{F}^t = \{\tilde{f}_1, \dots, \tilde{f}_t\},\ \tilde{f}_{t+1} = \phi(\mathcal{F}^{t+1}; \Theta) \\ &\text{and } \phi(\mathcal{F}^{t+1}; \Theta) = \omega + \sum_{i=1}^{p} A_i \tilde{s}_{t-i+1} + \sum_{j=1}^{q} B_j \tilde{f}_{t-j+1}. \end{aligned} \tag{3.16}$$

Here $\phi(\cdot; \Theta)$ denotes the updating function of our filtered time-varying parameter estimate $\tilde{f}_t$, with the set of fixed parameters Θ containing the (real) parameters $\omega, \{A_i\}_{i=1}^p$ and $\{B_j\}_{j=1}^q$. The most significant part of GAS models, the scaled score function s_t, is given by:

$$\begin{aligned} &\tilde{s}_t(\mathcal{F}^{t+1}; \Theta) = S_t(t, \tilde{f}_t, \mathcal{F}^t; \Theta) \cdot \tilde{\nabla}(\mathcal{F}^{t+1}; \Theta), \text{ for} \\ &\tilde{\nabla}(\mathcal{F}^{t+1}; \Theta) = \frac{\partial \ln \tilde{p}(y_t \mid \tilde{f}_t, \mathcal{F}^t; \Theta)}{\partial \tilde{f}_t}. \end{aligned} \tag{3.17}$$

The scaling function S_t can take several values, usually involving some power of the fisher information $\mathcal{I}(\cdot)$. This is motivated by the the score function $\tilde{\nabla}(\cdot)$'s variance, which is given by the fisher information.

We have defined the model without assuming correct specification, to point out that the score and therefore the updating function depends on the model implied conditional distributions $\tilde{p}_t$ of y_t. As our point has been made clear, we will omit "$\sim$" again, as if we assumed correct specification of the model in the following.

It can be see that the definition of GAS models is more of a framework, than a single specific model, as many factors have to be decided before application. We will use a

GAS(1,1) model with both mean equations, which we already employed in the other subsections before:

$$y_t = \mu + \sigma_t \epsilon_t, \text{ with } \epsilon_t \overset{iid}{\sim} p_\theta, \tag{3.18}$$

where p_θ is either a standard normal distribution, or a non-standardized t-distribution with zero mean, unit variance and $\nu = \theta$ degrees of freedom.

For the first case, i.e. $\epsilon_t \overset{iid}{\sim} \mathcal{N}(0,1)$, we can calculate the model implied score function, with respect to the time-varying variance σ_t^2, as

$$\begin{aligned}
\nabla(y_t, \sigma_t^2; \mu) &= \frac{\partial \ln p(y_t \mid \sigma_t^2; \mu)}{\partial \sigma_t^2} = \\
&= \frac{\partial}{\partial \sigma_t^2}\left(-\frac{1}{2}\log\left(2\pi\sigma_t^2\right) - \frac{(y_t-\mu)^2}{2\sigma_t^2}\right) = \\
&= \frac{(y_t-\mu)^2}{2\sigma_t^4} - \frac{1}{2\sigma_t^2}.
\end{aligned} \tag{3.19}$$

We want to use the inverse fisher information $\mathcal{I}(\cdot)^{-1}$ as a scaling function $S(\cdot)$, which is the score function's variance. It is given by

$$\begin{aligned}
\mathcal{I}(\sigma_t^2) &= -\mathbb{E}\left[\frac{\partial \nabla(y_t, \sigma_t^2; \mu)}{\partial \sigma_t^2}\right] = \\
&= -\mathbb{E}\left[-\frac{(y_t-\mu)^2}{\sigma_t^6} + \frac{1}{2\sigma_t^4}\right] = \\
&= \frac{\sigma_t^2}{\sigma_t^6} - \frac{1}{2\sigma_t^4} = \frac{1}{2\sigma_t^4}.
\end{aligned} \tag{3.20}$$

Combining the results from (3.19) and (3.20), we get the scaled score function

$$\begin{aligned}
s(y_t, \sigma_t^2; \mu) &= S(\sigma_t^2) \cdot \nabla(y_t, \sigma_t^2; \mu) = \\
&= 2\sigma_t^4 \cdot \left(\frac{(y_t-\mu)^2}{2\sigma_t^4} - \frac{1}{2\sigma_t^2}\right) = (y_t-\mu)^2 + \sigma_t^2,
\end{aligned} \tag{3.21}$$

which we plug into the general updating equation in (3.16), with $p = q = 1$:

$$\begin{aligned}
\sigma_t^2 &= \omega + \alpha s(y_{t-1}, \sigma_{t-1}^2; \mu) + \beta\sigma_{t-1}^2 = \\
&= \omega + \alpha(y_{t-1}-\mu)^2 + (\alpha+\beta)\sigma_t^2.
\end{aligned} \tag{3.22}$$

Hence, the result is equivalent to the standard GARCH model's updating function.
This gives a new perspective on the intuition behind using the squared observations as a driving factor for variance modelling. However, we have seen that the normal distribution doesn't fit our return data sample. Thus, we will calculate the resulting updating function for a GAS(1,1) model with t-distributed error terms, also called t-GAS(1,1).

We start off with the score function for the non-standardized t-distribution with mean μ, variance σ_t^2 and ν degrees of freedom, whose conditional density is given by

$$p(y_t, \sigma_t^2; \mu, \nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{(\nu-2)\pi}\Gamma\left(\frac{\nu}{2}\right)\sigma_t}\left(1 + \frac{(y_t-\mu)^2}{(\nu-2)\sigma_t^2}\right)^{-\frac{\nu+1}{2}}. \tag{3.23}$$

This yields the score function

$$\begin{aligned}\nabla(y_t, \sigma_t^2; \mu, \nu) &= \frac{\partial \ln p(y_t \mid \sigma_t^2; \mu, \nu)}{\partial \sigma_t^2} = \\ &= -\frac{1}{2\sigma_t^2} + \frac{\nu+1}{2} \cdot \frac{(y_t-\mu)^2/((\nu-2)\sigma_t^2)}{1 + (y_t-\mu)^2/((\nu-2)\sigma_t^2)} \\ &= \frac{(\nu+1)(y_t-\mu)^2}{2\sigma_t^2((\nu-2)\sigma_t^2 + (y_t-\mu)^2)} - \frac{1}{2\sigma_t^2}.\end{aligned} \tag{3.24}$$

We want to use two scaling functions: One being the identity and one defined as the inverse fisher information. Using the second formula for the multivariate t-distribution in [LLT1], Appendix B Proposition 4, we can derive the univariate fisher information (with respect to scale), which is given by

$$\mathcal{I}(\sigma_t^2; \nu) = \frac{\nu}{2\sigma_t^4(\nu+3)}. \tag{3.25}$$

Thus, we get the scaled score

$$\begin{aligned}s_t(y_t, \sigma_t^2; \nu, \mu) &= S(\sigma_t^2; \nu) \cdot \nabla(y_t, \sigma_t^2; \mu, \nu) = \\ &= \frac{\nu+3}{\nu}\left(\frac{(\nu+1)(y_t-\mu)^2\sigma_t^2}{(\nu-2)\sigma_t^2 + (y_t-\mu)^2} - \sigma_t^2\right).\end{aligned} \tag{3.26}$$

Altogether, this results in the updating function

$$\begin{aligned}\sigma_t^2 &= \omega + \alpha\nabla(y_{t-1} \mid \sigma_{t-1}^2; \mu, \nu) + \beta\sigma_{t-1}^2 = \\ &= \omega + \alpha\left(\frac{(\nu+1)(y_{t-1}-\mu)^2}{2\sigma_{t-1}^2((\nu-2)\sigma_{t-1}^2 + (y_{t-1}-\mu)^2)} - \frac{1}{2\sigma_{t-1}^2}\right) + \beta\sigma_{t-1}^2,\end{aligned} \tag{3.27}$$

if we use no scaling, i.e. $S(\cdot) = 1$. If the previously calculated inverse fisher matrix is used, we have

$$\begin{aligned}\sigma_t^2 &= \omega + \alpha s(y_{t-1}, \sigma_{t-1}^2; \mu, \nu) + \beta\sigma_{t-1}^2 = \\ &= \omega + \alpha\left(\frac{\nu+3}{\nu}\left(\frac{(\nu+1)(y_{t-1}-\mu)^2\sigma_{t-1}^2}{(\nu-2)\sigma_{t-1}^2 + (y_{t-1}-\mu)^2} - \sigma_{t-1}^2\right)\right) + \beta\sigma_{t-1}^2.\end{aligned} \tag{3.28}$$

For both equations it can be noted that the driving factor's numerator and denominator both include the demeaned squared returns. In particular, the driving factor's overall power is less than the squared observations, which are used in GARCH.

Motivation&Properties
The general idea behind the score driven parameter update is that it should move the variance estimate into the likelihood increasing direction for every single updating step. If the score is positive at time $t-1$, higher σ_t values would lead to a higher likelihood and the other way round. To ensure this, $\alpha > 0$ has to be used. As we are mainly interested in how the GAS models fare with our return data, we focus more on its practical properties than its theoretical ones here.
First of all, the driving factor likely leads to a more conservative update for both t-distributed cases - with and without the use of a scaling function. This might be sensible, as it deals better with the outliers that are more expected under a t-distribution. Whether this is helpful here, remains yet to be seen, though.
Secondly, the leverage effect is not accounted for in the updating function, as observations smaller than the mean are weighted the same as observations larger than the mean. This is not good, as we have seen that an inclusion of factors that allow an asymmetric feedback of observations verily influence the resulting estimates, likely giving better results.
Just like all previous models, the GAS model has a "GARCH" parameter in the form of $\beta\sigma_{t-1}^2$. Thus, it is capable of modelling the volatility clustering property of our data sample.
Lastly, it should be mentioned that we used a non-skewed t-distribution again, even though the return exhibited negative skewness. Including the skew might lead to better QML estimates and a more fitting updating function, but would also greatly complicate the already difficult computation of the inverse fisher information, if it should be used as a scaling function.

Estimation
Compared to the GARCH models, the estimation process for t-GAS models with and without scaling is more difficult. The score function obviously isn't strictly positive, which can easily result in negative variance estimates, if the coefficients aren't properly chosen. This problem could be alleviated by the use of a logarithmed updating equation, like in the EGARCH model. The resulting model would be equivalent to the Beta-t-EGARCH model (without leverage), described in the next subsection. In our case, the optimizing functions converged and gave sensible resulting estimates, though.
There exists an R package called "GAS" (see [CBA1]) that we will use for the case of t-GAS without scaling. As the GAS package implementation fails for the t-GAS model with inverse fisher information matrix scaling in our case, we will employ our own implementation. The computational details can be found in the Appendix.
To see, whether the models give sensible results, we compare them with the already implemented ones, namely the GARCH-t(1,1) model. Other GARCH subtypes, which include the leverage effect in their estimates, are not included, so that we can see if the different driving factors of GARCH-t and t-GAS result in a notable difference. We get the following estimates:

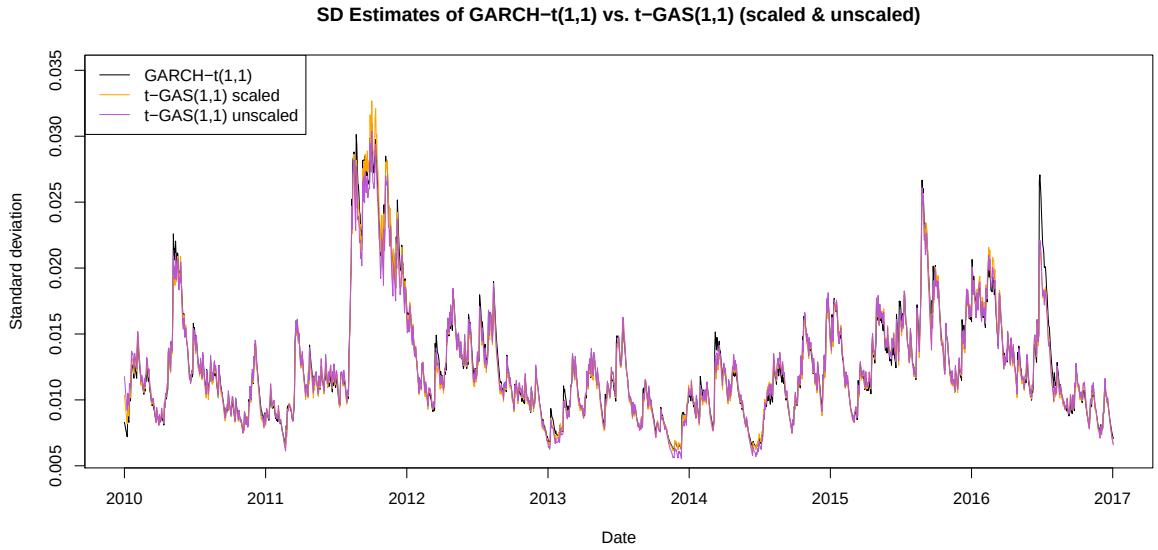

Figure 14: Resulting estimates for the standard deviation with the GARCH-t(1,1) model (black), t-GAS(1,1) model, scaled by the inverse fisher information matrix (orange) and the unscaled t-GAS(1,1) model (purple).

It can be noted that the difference between all models isn't very big most of the time. The only notable one between the GARCH-t and t-GAS type models are the estimates after the Brexit referendum on June 2016, where the t-GAS models suggest a lower volatility than the GARCH-t model. Hence, as expected, the t-GAS model's volatility updates are more conservative, in case there is a single, unusually large absolute return, due to the differing driving factors.
Additionally, the scaling doesn't seem to change the outcome substantially. Only during the time period in the end of 2012, there are slight differences. There, it leads to larger estimates, probably caused by the smaller effect of the scaling function - the bigger the previous estimate is, the larger the term σ_t^4 becomes, increasing the outcome. The scaled version might prevent underestimation this way, while keeping the stability to outliers. In fact, the reaction of the scaled version to outliers is even weaker, than the unscaled version's. Altogether, this means that the scaled t-GAS model reacts less intensely after initial shocks, which is a property of t-GAS models, but accounts for the increased variance in volatility during high-volatility periods, like GARCH models.
In the following part, we will introduce a few more models, which fit into the GAS framework and account for more properties of our return data, considering that the t-GAS models don't cover the leverage and skew properties. Thus, to avoid an abundance of model candidates, we only consider the scaled version in the upcoming analysis.

3.1.8 Beta-t-EGARCH

Specification

During the same time as the GAS model came up, the Beta-t-EGARCH model was

introduced. It was widely published in [Ha1] and specifically designed to handle heavy tailed data. We will use two special subtypes, which have been proposed as an extension to the model: The one-component and the two-component Beta-Skew-t-EGARCH models from [HS1] with leverage coefficients.
The mean equation is the same for both models and given as

$$y'_t = \exp(\lambda_t)(\epsilon^*_t - \mu_\epsilon) = \sigma_t \epsilon_t, \text{ for } \epsilon_t \overset{iid}{\sim} t_{\nu,\gamma}, \tag{3.29}$$

where $t_{\nu,\gamma}$ follows a demeaned, skewed t-distribution with unit scale, $\nu > 2$ degrees of freedom and skew-parameter γ. More details and its density function are given in the Basics section, see (1.19). Note that σ_t is not equal to the modelled standard deviation, as the skewed t-distribution's variance is not equal to 1. To obtain the model implied standard deviation, one has to multiply σ_t with the distribution's standard deviation σ_ϵ, which can be calculated with an estimate for γ and ν. The equations for the mean μ_ϵ and variance σ^2_ϵ of ϵ^*_t, which follows a skewed t-distribution, can be found in [HS1], chapter 3.
Here, we use the centered skew-t-distribution for ϵ_t, in order to avoid making things more difficult than necessary, considering that the model is already sufficiently complex. Thus, we model the volatility with demeaned data y'_t here, by subtracting the sample mean from the observations.
The one-component Beta-Skew-t-EGARCH model is the standard version of the model, handling the short-term influences on the volatility. Its updating function is defined by

$$\begin{aligned} \lambda_t &= \omega + \lambda^\dagger_t, \\ \lambda^\dagger_t &= \phi \lambda^\dagger_{t-1} + \kappa u_{t-1} + \kappa^* \operatorname{sgn}(-y'_{t-1})(u_{t-1} + 1) \end{aligned} \tag{3.30}$$

and

$$\begin{aligned} u_{t-1} = \nabla(y'_{t-1}, \lambda_{t-1}; \nu, \gamma) &= \frac{\partial \log p(y'_{t-1} \mid \lambda_{t-1}; \gamma, \nu)}{\partial \lambda_{t-1}} \\ &= \frac{(\nu+1)\big(y'^2_{t-1} + y'_{t-1}\mu_\epsilon \exp(\lambda_{t-1})\big)}{\nu \exp(2\lambda_{t-1})\gamma^{2\operatorname{sgn}(y'_{t-1}+\mu_\epsilon \exp(\lambda_{t-1}))} + (y'_{t-1} + \mu_\epsilon \exp(\lambda_{t-1}))^2} - 1. \end{aligned} \tag{3.31}$$

As in the EGARCH specification, we have $\log \epsilon_t = \lambda_t$ as the time-varying scale parameter, ω as a constantly added (or subtracted) term and ϕ corresponding to the previously defined persistence influencing parameter β. For a sensible and stable estimation performance, ϕ is assumed to be in $(0, 1)$. The size of the driving factor is affected by κ, while κ^* is the leverage effect coefficient.
In contrast to the GARCH-type models, the driving factor is the score function u_{t-1}, based on the observation's conditional density $p(y'_{t-1} \mid \lambda_{t-1}; \gamma, \nu)$. The score term also coins the name of the Beta-t-EGARCH model: If we omitted the skew by choosing $\gamma = 1$, $\frac{u_t+1}{\nu+1}$ would follow a $Beta(\frac{1}{2}, \frac{\nu}{2})$-distribution.
A little different is the two-component Beta-Skew-t-EGARCH model, which divides the time-varying part of the conditional scale parameter λ_t into the sum of two parameters. The purpose of this is, to let one component model the short-run volatility effects, while

the other one captures the long-run ones. The the mean equation for y'_t remains the same as in (3.30). However, for the updating function of λ_t, we get

$$\begin{aligned} \lambda_t &= \omega + \lambda^\dagger_{1,t} + \lambda^\dagger_{2,t}, \text{ with} \\ \lambda^\dagger_{1,t} &= \phi_1 \lambda^\dagger_{1,t-1} + \kappa_1 u_{t-1} \text{ and} \\ \lambda^\dagger_{2,t} &= \phi_2 \lambda^\dagger_{2,t-1} + \kappa_2 u_{t-1} + \kappa^* \operatorname{sgn}(-y'_{t-1})(u_{t-1}+1). \end{aligned} \tag{3.32}$$

Hence, the only difference is the introduction of the time-varying long-term component $\lambda^\dagger_{1,t}$. As $\lambda^\dagger_{1,t}$ is the long-term component, it obviously should be more persistent than the short-term part. Thus, also ensuring covariance stationarity, $0 < \phi_2 < \phi_1 < 1$ is assumed.

Motivation&Properties
It has already been noted that the standard Beta-t-EGARCH model without leverage component is equivalent to a t-GAS model with logarithmed conditional standard deviation. Thus, all properties of the standard t-GAS(1,1) model apply here, too. In particular, the one- and two-component Beta-Skew-t-EGARCH models react less sensitively to outliers than GARCH models, due to the differing driving factors.
Contrary to the standard t-GAS(1,1) model, these Beta-t-EGARCH subtypes can also handle leverage effects, though. In addition to this, it is the only model we covered that incorporates the skewness of our returns in both, the updating equation of the time-varying scale parameter and the estimation process for the fixed parameters. Whether the skew is large enough to make a substantial difference remains yet to be seen, though.
The introduction of the two-component variant was motivated by the long-memory behavior that financial returns often exhibit (see Basics section). It follows the same concept as the two-component GARCH model from [EL1] - both divide the overall updating function into a long- and short-term component. Effects on volatility, caused by temporary (large) shocks, shall be captured by the short-term component, while the long-memory behavior is modelled by the long-term component. In our case, the short-term component also includes the leverage effect coefficient, as it mostly affects the volatility in the short run, according to [EL1].

Estimation
As for the EGARCH model, one upside of the logarithmed updating equation is the lack of a need for positive values, which makes things easier compared to the previously covered standard t-GAS model. However, the inclusion of the skew in the parameter update leads to differing model implied density functions and hence log likelihood and driving factors.
There already exists a properly working R package for the Beta-t-EGARCH model, hence we refrain from estimating the necessary coefficients through a selfmade implementation and instead use the "betategarch" package, described in [Su1]. Thus, the only thing we do is choosing sensible starting values, so that the optimizing function converges (see Appendix).
To get a first impression, we compare the resulting estimates of our EGARCH(1,1) model

with the one- and two-component Beta-skew-t-EGARCH models:

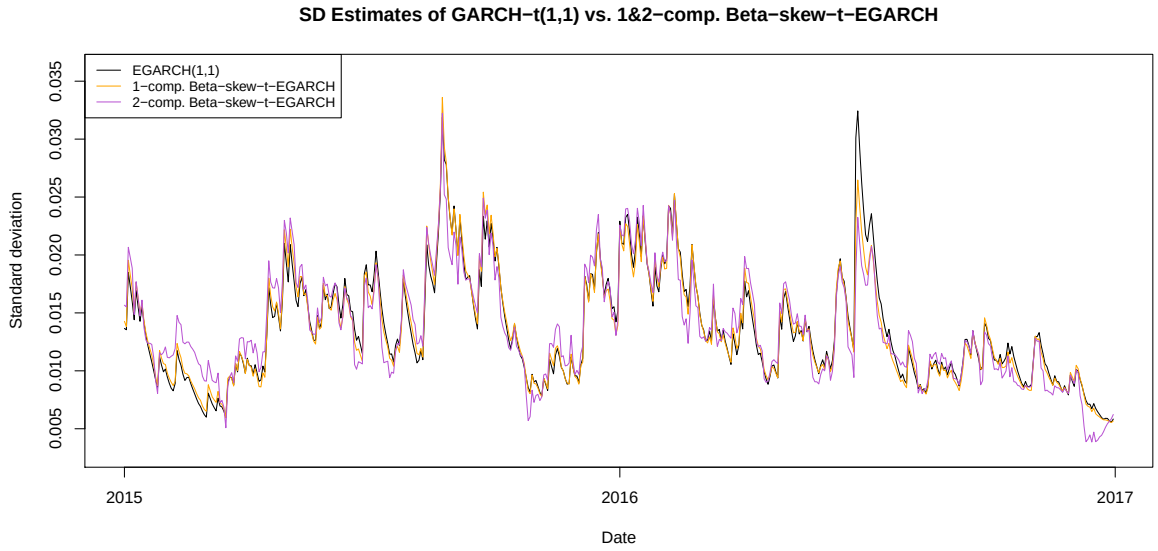

Figure 15: Resulting estimates for the standard deviation in 2015 and 2016 with the EGARCH(1,1) model (black), one-component Beta-skew-t-EGARCH(1,1) model (orange) and the two-component Beta-skew-t-EGARCH(1,1) model (purple).

We can firstly not that the EGARCH model is highly similar to the one-component type model, giving mostly the same results - with exception of the outlier in June 2016, caused by the Brexit referendum. This is in line with our expectations, as the only differences are the different driving factor, which reacts less sensitively to outliers and the inclusion of the skew parameter.
The two-component model on the other hand gives notably different results - sometimes reacting stronger and sometimes weaker to the observations than both other models. Which one is best, and if the additional parameters of the two-component model lead to substantially better results, is yet unclear and will be covered later on, after we introduced the last model that is included in this analysis.

3.1.9 Stochastic Volatility

Specification
As mentioned in the introduction, the standard Stochastic Volatility (SV) model was introduced in [Ta1] and takes a very different approach from all the previous models. While the other models specify the time-varying standard deviation deterministically for given observations and starting value, the standard SV-model's (logarithmed) standard deviation follows an AR(1) process, which is random. The purpose of exponential standard deviation values is to ensure positivity, as the AR(1) process might yield negative values. Again, we demean our returns here, by subtracting the sample mean, defining the centered sample as y_t'.

Thus, the standard SV model is given by

$$\begin{aligned} y_t' &= \exp(\lambda_t)\epsilon_{1,t}, && \text{with } \epsilon_{1,t} \overset{iid}{\sim} \mathcal{N}(0,1) \text{ and} \\ \lambda_t &= \mu + \phi(\lambda_{t-1} - \mu) + \epsilon_{2,t}, && \text{with } \epsilon_{2,t} \overset{iid}{\sim} \mathcal{N}(0, \sigma_\lambda^2). \end{aligned} \tag{3.33}$$

Note that μ doesn't denote the sample's mean, but the unconditional mean of the (logarithmed) standard deviation and σ_λ is the unconditional standard deviation of λ_t's errors.
We acquire the conditional densities $y_t' \sim \mathcal{N}(0, \exp(\lambda_t))$ and $\lambda_t \sim \mathcal{N}(\mu + \phi(\lambda_{t-1} - \mu), \sigma_\lambda^2)$, depending on the parameters $\mu, \phi, \sigma_\lambda^2$ and past values for λ_t. The starting value λ_0 is distributed according to the unconditional mean and standard deviation of the process λ_t, i.e. $\lambda_0 \sim \mathcal{N}(\mu, \sigma_\lambda^2/(1-\phi^2))$.

As we have seen that the normal distribution doesn't account for the heavy tails of our sample, we will also include an SV model with errors $\epsilon_{1,t} \sim t_\nu$ that follow a non-standardized t-distribution with mean zero, variance $(\nu \exp(\lambda_t))/(\nu - 2)$ and $\nu > 2$ degrees of freedom. The resulting model is also referred to as the SV-t model. Its use has been suggested in [HRS1], but it remains to be seen, whether we acquire substantially different and better results.

Motivation&Properties
The general notion of the model is quite clear and very direct: In order to model heteroscedasticity with linearly correlated variance, the variance is assumed to follow an autoregressive process. As this process might yield negative outcomes, its exponential value is used to ensure positivity, similarly to the later introduced EGARCH approach. We can see that the model specifications are rather basic and the AR process doesn't contain specific parameters considering leverage effect, skew or long-memory behavior. They can thus be compared to the GARCH and GARCH-t model respectively, property-wise. However, as [Yu1] has pointed out, a leverage effect would be introduced, if negatively correlated error terms $\epsilon_{1,t}$ and $\epsilon_{2,t}$ were considered.

Estimation
Probably the main reason why SV models didn't catch on as well as the GARCH model, is that estimation for SV models is not as straightforward as for the other models. The conditional density of the demeaned returns y_t' are assumed to stem from a gaussian distribution with gaussian distributed variance, i.e. y_t' follows a compound probability distribution. Thus, the density contains an integral over the gaussian distributed (logarithmed) variance, which makes evaluation difficult.
To resolve this issue, one could resort to QMLE. However, according to [JPR1], Markov Chain Monte Carlo methods with a bayesian approach yield better results.
Hence, we use the R package "stochvol" from [Ka1] and [Ka2] for the sampling of posterior draws. The choice of priors and other computational details are discussed there, as well as in [KFS1]. Obviously, this way of estimation is more computationally intensive than the comparatively simple maximum likelihood estimation, as we have to sample

many draws for every single time point, resulting in plenty evaluations.
To assess the outcome, we firstly compare the resulting posterior quantiles from the SV and SV-t model, checking whether there is a palpable difference between the two:

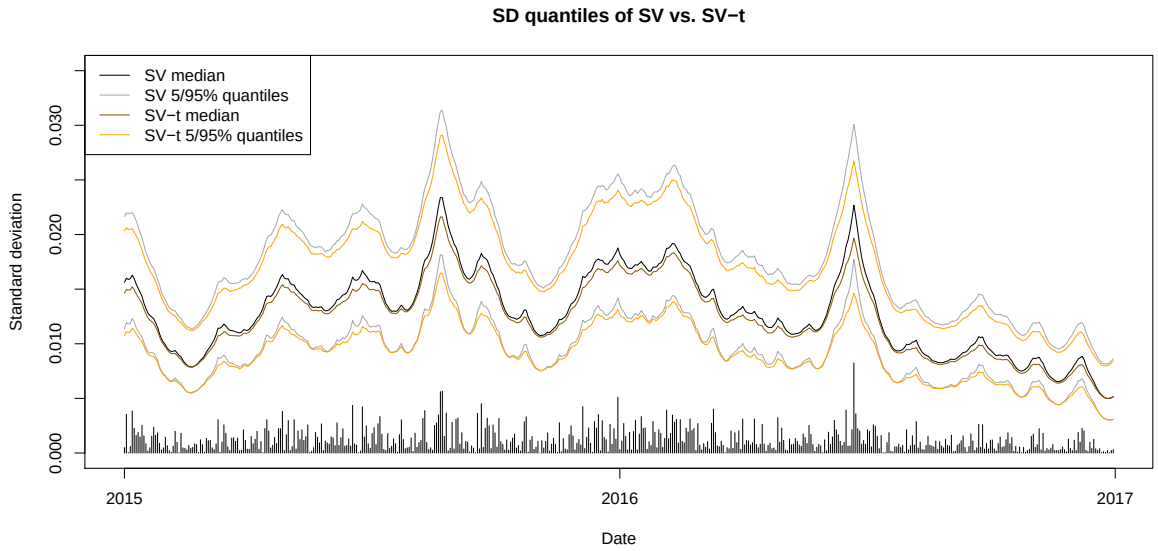

Figure 16: Resulting quantiles for the posterior standard deviation draws in 2015 and 2016 with the SV-model (black & gray) and SV-t model (brown & orange). The black bars indicate the absolute size of the returns (not scaled to y-axis).

We can see that the SV-t model reacts less strongly to outliers, but also all other shocks of arbitrary size, as its quantiles give smaller values most of the time. This indicates that SV-t underestimates the standard deviation at times, or the other way round.
Moreover, considering that these models take a different approach than all of the previous ones, we will also compare the outcome with the other models' estimates, to see whether there are major differences. For this, we graph the SV-t results against the estimates of GARCH-t(1,1), t-GAS(1,1) and the two-component Beta-t-EGARCH model:

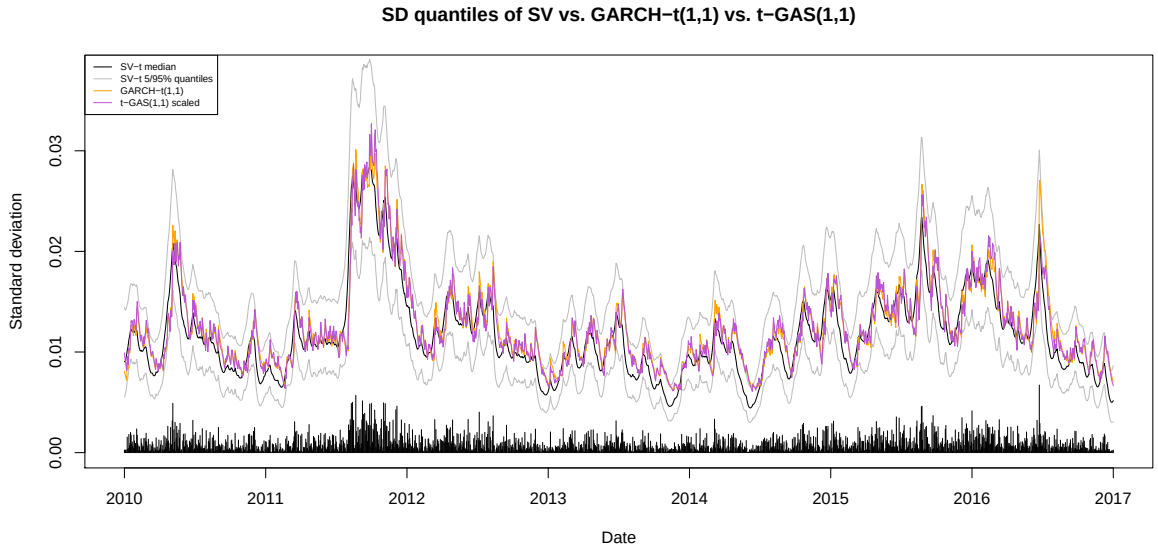

Figure 17: Resulting quantiles for the SV draws (black & gray) with the estimates from the GARCH-t(1,1) model (orange) and scales t-GAS(1,1) model (purple).The black bars indicate the absolute size of the returns (not scaled to y-axis).

Firstly, it can be noted that the GARCH and GAS models mostly are inside the range of the 5% and 95% quantiles. Secondly, it seems that the SV-t quantiles adapt quicker to changes in volatility. This can be seen nicely, when looking at the median of the SV draws, which seems to drop quicker and rise faster than the other estimates. Additionally, it also yields smaller results for most time periods - whether these are sensible or not, remains yet unclear.

Now that we introduced every model and looked at them superficially, we will summarize them in a short overview, to recapitulate their properties. Afterwards, we will proceed with the model assessment, in order to decide, which performs best for our sample.

3.2 Overview

In the first table, we summarize the respective model implied distributions and whether they are heavy tailed:

Model	Distribution	Heavy Tails
GARCH (1,1)	$\mathcal{N}(\mu, \sigma_t^2)$	×
GARCH-t(1,1)	non-standard $t_\nu(\mu, \sigma_t^2)$	✓
EGARCH-t(1,1)	non-standard $t_\nu(\mu, \exp(\sigma_t)^2)$	✓
GJR-GARCH-t(1,1)	non-standard $t_\nu(\mu, \sigma_t^2)$	✓
TGARCH-t(1,1)	non-standard $t_\nu(\mu, \sigma_t^2)$	✓
t-GAS(1,1)	non-standard $t_\nu(\mu, \sigma_t^2)$	✓
1-comp. Beta-Skew-t-EGARCH	skewed $t_{\nu,\gamma}(0, (\exp(\lambda_t)\sigma_\epsilon)^2)$	✓
2-comp. Beta-Skew-t-EGARCH	skewed $t_{\nu,\gamma}(0, (\exp(\lambda_t)\sigma_\epsilon)^2)$	✓
SV	$\mathcal{N}(0, \exp(\lambda_t)^2)$ and $\lambda_t \sim \mathcal{N}(\mu + \phi(\lambda_{t-1} - \mu), \sigma_\lambda^2)$	×
SV-t	$t_\nu(0, (\nu \exp(\lambda_t))/(\nu - 2))$ and $\lambda_t \sim (\mu + \phi(\lambda_{t-1} - \mu), \sigma_\lambda^2)$	✓

Table 5: Model implied distributions - the variables in the brackets denote the respective means and variances.

Next, we cover the driving factors of each of the time-varying parameters, without the leverage part (if existent). It is denoted additionally, whether they handle heavy tailed data well, by being less responsive to single outliers.

Model	tvp	Driving Factor of tvp	H.T.
GARCH (1,1)	σ_t^2	$(y_{t-1} - \mu)^2$	×
GARCH-t(1,1)	σ_t^2	$(y_{t-1} - \mu)^2$	×
EGARCH-t(1,1)	σ_t^2	$\lvert\frac{(y_{t-1}-\mu)}{\sigma_{t-1}}\rvert - \mathbb{E}\lvert\frac{(y_{t-1}-\mu)}{\sigma_{t-1}}\rvert$	×
GJR-GARCH-t(1,1)	σ_t^2	$(y_{t-1} - \mu)^2$	×
TGARCH-t(1,1)	σ_t	$y_{t-1} - \mu$	×
t-GAS(1,1)	σ_t^2	$\frac{\nu+3}{\nu}\left(\frac{(\nu+1)(y_{t-1}-\mu)^2\sigma_{t-1}^2}{(\nu-2)\sigma_{t-1}^2+(y_{t-1}-\mu)^2} - \sigma_{t-1}^2\right)$	✓
1-comp. Beta-Skew-t-EGARCH	λ_t	$\frac{(\nu+1)\left(y_{t-1}'^2+y_{t-1}'\mu_\epsilon \exp(\lambda_{t-1})\right)}{\nu \exp(2\lambda_{t-1})\gamma^{2\,\mathrm{sgn}(y_{t-1}'+\mu_\epsilon \exp(\lambda_{t-1}))}+(y_{t-1}'+\mu_\epsilon \exp(\lambda_{t-1}))^2} - 1$	✓
2-comp. Beta-Skew-t-EGARCH	λ_t	$\frac{(\nu+1)\left(y_{t-1}'^2+y_{t-1}'\mu_\epsilon \exp(\lambda_{t-1})\right)}{\nu \exp(2\lambda_{t-1})\gamma^{2\,\mathrm{sgn}(y_{t-1}'+\mu_\epsilon \exp(\lambda_{t-1}))}+(y_{t-1}'+\mu_\epsilon \exp(\lambda_{t-1}))^2} - 1$	✓
SV	λ_t	AR(1) process	–
SV-t	λ_t	AR(1) process	–

Table 6: Overview of the driving factors for the respective time-varying parameters (tvp) and whether they are made for heavy-tailed data, i.e. not overresponsive to outliers.

The third and last table sums up some other properties, including the existence of a leverage component, skew or long memory component.

Model	Leverage Component	Skew	Long Memory
GARCH (1,1)	×	×	×
GARCH-t(1,1)	×	×	×
EGARCH-t(1,1)	$\frac{(y_{t-1}-\mu)}{\sigma_{t-1}}$	×	×
GJR-GARCH-t(1,1)	$(y_{t-1}-\mu)^2\mathbb{1}_{\{y_{t-1}-\mu<0\}}$	×	×
TGARCH-t(1,1)	$\alpha^+(y_{t-1}-\mu)\mathbb{1}_{\{y_{t-1}-\mu>0\}}+$ $\alpha^-(y_{t-1}-\mu)\mathbb{1}_{\{y_{t-1}-\mu\leq 0\}}$	×	×
t-GAS(1,1)	×	×	×
1-comp. Beta-Skew-t-EGARCH	$\text{sgn}(-y'_{t-1})(u_{t-1}+1)$ $(u_{t-1}=\text{driving factor})$	✓	×
2-comp. Beta-Skew-t-EGARCH	$\text{sgn}(-y'_{t-1})(u_{t-1}+1)$ $(u_{t-1}=\text{driving factor})$	✓	✓
SV	×	×	×
SV-t	×	×	×

Table 7: Review of the leverage and long memory components and whether it includes a skewed distribution. In the case of the Beta-Skew-t-EGARCH model, the skew is also accounted for in the driving component of the time-varying parameter.

Additionally, a table containing the estimated parameters can be found in the Appendix. The parameter draws for the stochastic volatility model are also added in form of graphs, showing their respective trace and overall distribution density.
Following this brief summary, we will now proceed with a performance analysis of the models, for a more in-depth examination.

4 Model Performance Analysis

This section will be divided into two parts: Firstly, we will investigate, how well each models' in-sample fit is and which one gives the best results. Afterwards, we will do several forecasts up to 1 month (22 days) in advance and compare the outcomes with the actual DAX return data of the January 2017, to evaluate their out-of-sample performance.

4.1 In-sample

The intention behind this work is to see, how the different factors of each model come into effect in the volatility analysis of financial returns. Hence, some models, which are relatively simple, are very likely to be outperformed by other models that take into account the specific properties of financial returns. Additionally, we included some factors, which are most likely unnecessary and could be omitted in a "real" analysis. An example for such a parameter is the estimate for an appropriate starting value for the standard deviation σ_0: In practice, this additional coefficient could be omitted by replacing it with the sample standard deviation and/or using a small sample before the beginning of the time period to calculate an appropriate value and then discard this burn-in phase.
To see, which parameters have little effect and might be left out without much loss of fit, we calculate the standard errors and significance levels, by the use of their respective t-statistics. Again, more details about this method are given in the basics section and the full test data can be found in the parameter table in the Appendix. Here, we will simply draw the conclusions from the tests and mention notable parameter estimates and their implications about the model and data itself:

1. Throughout all models that include a mean coefficient μ and starting value σ_0 in their estimates, these parameters are very close to zero and have a rather large p-value. Particularly σ_0 yields a relatively large p-value in most models, suggesting that the null hypothesis $\mathcal{H}_0 : \sigma_0 = 0$ can be accepted at 5% significance level in most cases. This seems reasonable, considering that σ_0 mostly impacts the first few estimates and slowly diminishes in importance over time. Obviously, this doesn't mean that the true σ_0 is zero, rather that it has a very small influence on the overall model performance and that it might be more sensible, to initiate the estimation process with 0, instead of adding an additional parameter. If the data sample was smaller, though, it would likely have a higher significance.
   In our case, the use of a sensible starting value makes sense, as we want to model the volatility over the whole time period and don't include a burn-in phase. Whether μ should be included or not, can be regarded as a matter of taste: If demeaned data is used, or alternatively the parameter μ is included, is not relevant to the volatility modelling part.

2. As expected, the ω estimates are rather small for most models, where it is a non-varying and thus not flexible additive. There, ti can be thought of as something

like an intercept term. The only models with significantly large absolute value of ω, are the ones that model an exponential time-varying parameter, namely the EGARCH and Beta-t-EGARCH models. They also are the only models that can handle a negative ω, without risking a negative outcome for the variance or standard deviation. While the values of about -0.36 and -4.6 for the EGARCH and Beta-t-EGARCH models seem unusually high by themselves, it has to be considered that the exponential function is applied to the whole updating term, thus changing the role of ω: Unlike in the non-exponential models, it can't be seen as an added "intercept" term here, but rather an overall scaling factor. This can be easily derived, by considering $\sigma_t = \exp(\lambda_t) = \exp(\omega) \cdot \exp(\lambda_t - \omega)$.

3. The models with a t-distributed model implied density all estimate the degrees of freedom with values around 7, stemming from the leptokurtic sample.

4. In the specifications, where the asymmetric sign effect is modelled by separate terms for positive and negative returns, i.e. the GJR-GARCH and TGARCH models, we acquire results, which suggest a strong leverage effect: The parameter that determines the influence of positive returns, is estimated as nearly zero and highly insignificant in both cases. This means that all positive returns of realistic magnitude reduce the estimated volatility by about 5-10%, considering the small ω values and β estimates around 0.9. The notion that positive returns of all sizes reduce the volatility is somewhat in line with the leverage effect, but it is unrealistic that the size of these returns doesn't play a role.
   To make up for the lacking impact of positive returns, the impact of negative returns is doubled, compared to the standard GARCH models: The scalar coefficient that determines the influence of negative returns is 0.2 for the leverage models, while it is 0.1 for the standard GARCH ones.
   The same holds for the EGARCH model: as α is negative, negative values tend to increase and positive values tend to decrease the volatility estimates. However, as there is an additional size component, unusually high positive returns might still lead to an increasing volatility estimate in this case.

5. The last notable parameter estimate is in the short-term component of the 2-component Beta-Skew-t-EGARCH model. The coefficient κ_2 is rather small in absolute size, with a relatively large standard error and p-value, indicating little significance. This might be caused by the fact that the volatility updating step size is influenced by three parameters, resulting in a reduced individual significance and size of the three.
   The long-term component parameter estimates seem legitimate, considering that $\phi_1 > \phi_2$ indicates a higher persistence of the influence of positive or negative shocks.

After looking into the parameter estimates and their individual significance, we check, whether the order of the non-stochastic volatility models is sufficient, as we chose a lag

order of one for all models. We do this, by conducting Score and Portmanteau tests on the (squared) standardized Residuals, to see, if there is significant autoregressive heteroscedasticity left (see Basics). Moreover, the autocorrelation functions, i.e. the sample autocorrelations, as a function of lag, of the respective squared standardized residuals are depicted.
We start off with the Ljung-Box test results:

	lag = 1	lag = 4	lag = 7	lag = 10
GARCH(1,1)	0.6622	0.9277	0.9913	0.9973
GARCH-t(1,1)	0.5587	0.9347	0.9895	0.9978
EGARCH-t(1,1)	0.5963	0.9490	0.9891	0.9956
GJRGARCH-t(1,1)	0.1590	0.5678	0.7885	0.8703
TGARCH-t(1,1)	0.5122	0.9382	0.9819	0.9879
t-GAS(1,1)	0.8857	0.8837	0.9811	0.9942
1-comp. Beta-Skew-t-EGARCH(1,1)	0.5167	0.9245	0.9908	0.9966
2-comp.Beta-Skew-t-EGARCH(1,1)	0.3804	0.9027	0.9820	0.9941

Table 8: p-values of Ljung-Box test on the squared standardized residuals.

The p-values from the Ljung-box test suggest that the null hypotheses can be rejected at a 5% significance level in all cases. Hence, the sample autocorrelations up to the specified lag order are (jointly) negligible and it can be followed that the ARCH effects are modelled appropriately for all models, by using a lag order of 1.
The results of the ARCH LM test are highly similar:

	lag = 1	lag = 4	lag = 7	lag = 10
GARCH(1,1)	0.6625	0.9279	0.9912	0.9979
GARCH-t(1,1)	0.5591	0.9340	0.9901	0.9988
EGARCH-t(1,1)	0.5967	0.9468	0.9900	0.9975
GJRGARCH-t(1,1)	0.1594	0.5407	0.7990	0.8704
TGARCH-t(1,1)	0.5127	0.9353	0.9845	0.9931
t-GAS(1,1)	0.8858	0.8875	0.9809	0.9952
1-comp. Beta-Skew-t-EGARCH(1,1)	0.5171	0.9237	0.9911	0.9975
2-comp.Beta-Skew-t-EGARCH(1,1)	0.3808	0.9009	0.9841	0.9965

Table 9: p-values of ARCH LM test on the standardized residuals.

Lastly, we check the autocorrelation functions of the squared standardized residuals:

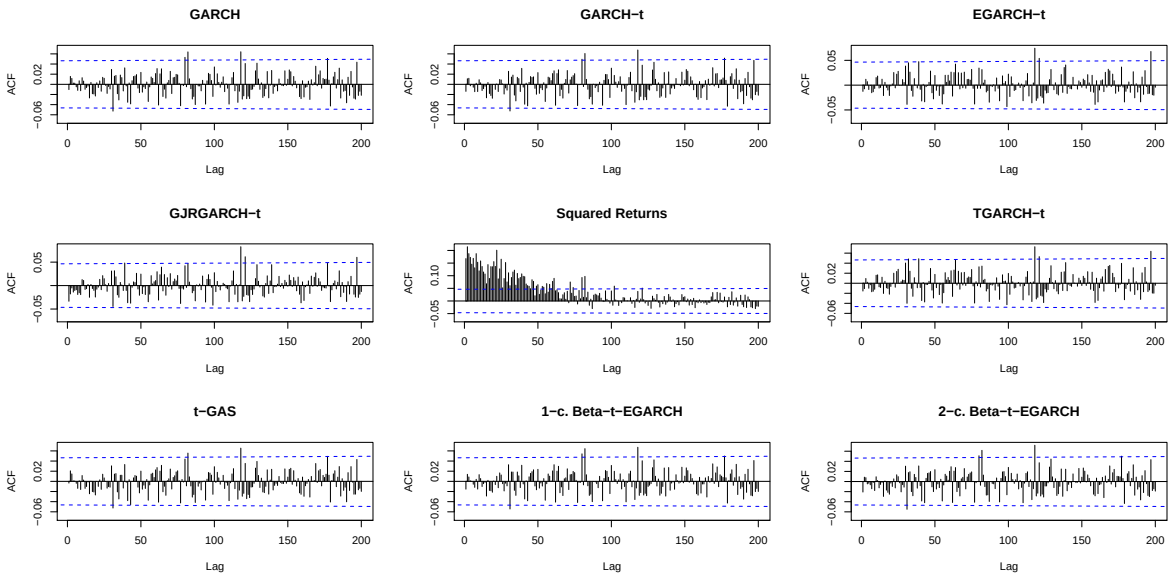

Figure 18: Autocorrelation functions for each model's squared standardized residuals. The blue lines denote the confidence intervals and the autocorrelation function of the squared returns is added in the middle as a comparison.

The dashed lines are the borders of the 95% confidence interval for the sample autocorrelations (see "Leverage" Basics). Considering that only about 4 to 5 of the 200 respective sample autocorrelations are outside of the 95% interval, we can follow that the autocorrelation functions of the standardized residuals show no notable sign of autocorrelation.

In Summary, all tests and the autocorrelation functions suggest that there are no significant ARCH effects left after standardizing. Hence, increasing the lag order is not necessary. The same conclusion could be reached by comparing the AIC and BIC values of some lag order combinations with the (1,1) alternative, as our sample is not too big. This might be a non-feasible or undesirable way of determining the optimal lag order, though, if the sample size or optimal lag order is very large.
Next, we apply the autocorrelation test on the non-squared residuals, in order to check for the existence of autocorrelation in the mean series. If this would be the case, we would have to model it with an AR process. We get the following Ljung-Box test results:

	lag = 1	lag = 4	lag = 7	lag = 10
GARCH(1,1)	0.1871	0.2583	0.1244	0.2970
GARCH-t(1,1)	0.2083	0.2737	0.1365	0.3192
EGARCH-t(1,1)	0.3761	0.3392	0.1486	0.3350
GJRGARCH-t(1,1)	0.3375	0.3104	0.1210	0.2883
TGARCH-t(1,1)	0.3127	0.3158	0.1319	0.3126
t-GAS(1,1)	0.2118	0.3071	0.1379	0.3179
1-comp. Beta-Skew-t-EGARCH(1,1)	0.1877	0.2638	0.1277	0.3072
2-comp.Beta-Skew-t-EGARCH(1,1)	0.2100	0.2817	0.1409	0.3333

Table 10: p-values of Ljung-Box test on the standardized residuals.

Again, the null hypothesis of Autocorrelation in the Standardized Residuals can be rejected at a 5% significance level for all models. However, the p-values are collectively not as large as for the previous tests, specifically around the values for lag order 7. Hence, we exemplarily compare the AIC and BIC values (see basics) for the standard GARCH and EGARCH-t models with included ARMA orders of (1,0), (0,1) and (1,1). In these estimations, we omit the starting value σ_0 and the mean μ for convenience:

	AIC	BIC
ARMA(0,0)-GARCH(1,1)	-10712.092	-10695.640
ARMA(1,0)-GARCH(1,1)	-10710.701	-10688.766
ARMA(0,1)-GARCH(1,1)	-10710.730	-10688.795
ARMA(1,1)-GARCH(1,1)	-10709.490	-10682.071
ARMA(0,0)-EGARCH-t(1,1)	-10840.848	-10813.429
ARMA(1,0)-EGARCH-t(1,1)	-10840.841	-10807.939
ARMA(0,1)-EGARCH-t(1,1)	-10840.903	-10808.000
ARMA(1,1)-EGARCH-t(1,1)	-10840.399	-10802.013

Table 11: AIC and BIC comparison of GARCH-t and EGARCH-t for different ARMA orders.

For the standard GARCH model, both criteria suggest leaving out the ARMA terms, while the AIC results are not as clear in the EGARCH case. However, the difference in AIC is minuscule and the BIC strongly suggests the ARMA(0,0) case, i.e. modelling the mean series by a constant term.
Hence, additionally considering the test results from table 10, we proceed without specifically modelling the mean process.

Next, we want to look into the long-memory property of our sample and point out the difference between the models with short-memory behavior, versus the one model with long-memory. For this purpose, we firstly compare the respective autocorrelation functions of simulated EGARCH and 1 & 2-component Beta-Skew-t-EGARCH processes. The model coefficients used for the simulation are the estimates we acquired from fitting these models to our return sample. This way, we can also indirectly find out, if our return sample exhibits long-memory behavior, as then substantially larger sample

autocorrelations for higher lag orders are to be expected in the 2-component model over the short-memory models.
In [BD1] it is shown that the sample autocorrelation is a consistent estimator for the true autocorrelation under certain assumptions. Thus, we simulate relatively many sample values, to possibly get a clearer shape of the autocorrelation functions.
The following graph now compares said models' autocorrelation functions with each other, as well as with the one of the squared returns:

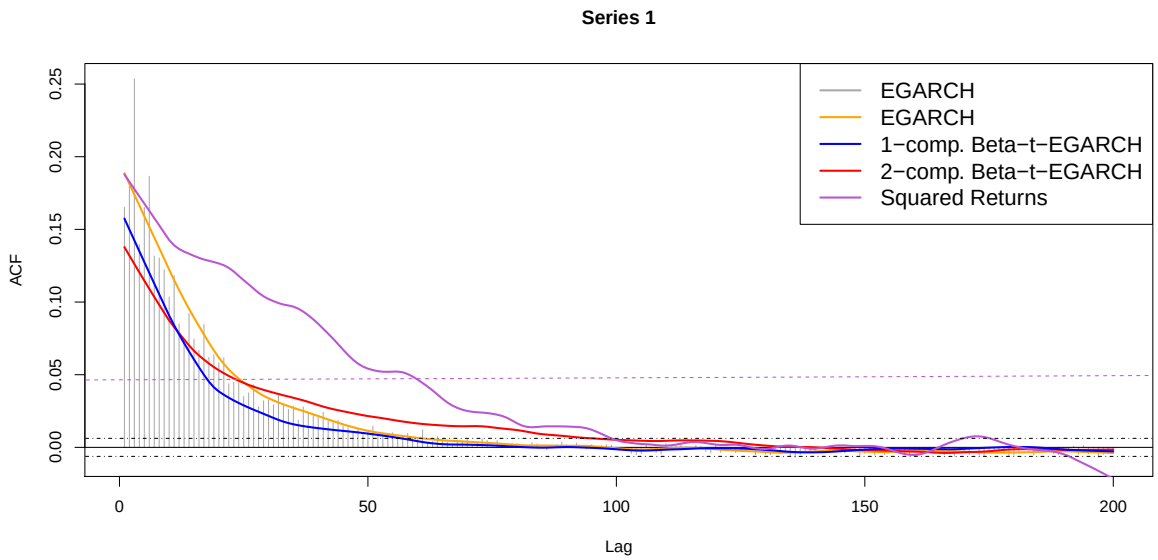

Figure 19: The graph contains the Autocorrelation function (ACF) values of the simulated EGARCH process (gray), as well as the smoothed ACF values of each simulated process and the squared returns as a comparison (purple). The confidence intervals for significance are included for the squared returns with 1779 values (purple dashed line) and the simulated processes with 100.000 sample values (black dashed lines). The smoothing is achieved by locally weighted polynomial regression, with a local span of 10% of the points (see for example [Cl1]).

We can extract the following from the depiction above:
Firstly, we can see that the two-component variant's sample autocorrelations stay outside of the confidence interval, i.e. significant, up to a lag of 100, while the others only go as far as 50. Additionally, its values are smaller than the ones of the other models in the beginning. Altogether, the two-component process shows a slower, more constant decline in the shape of its autocorrelation function, which is in line with long-memory behavior and suggests a "more polynomial decrease" instead of an exponential one in the sample autocorrelations.
As all the processes were simulated with the estimates from our sample, this suggests that the maximum likelihood estimate tends towards the model with long-memory behavior, as the long-term components aren't insignificant.
Another point that can be noted is that the line from the squared returns is way larger

than all the other ones, with very slow decline. This suggests long-memory behavior, too, with significant sample autocorrelations up to an order of about 50 to 100.
However, taking the confidence intervals as the deciding factor, when checking whether the sample autocorrelations of any order are significantly different from zero, might not be validated. The intervals should be taken with a grain of salt, considering that we didn't check, whether their assumptions are met. But even if this were not the case, the first point still remains valid, suggesting that using a model with the long-memory property for our sample is sensible.
An easier way to decide, whether incorporating long-memory components yields a better fit, is to consult information criteria, such as the AIC and BIC (see basics). We get the following results:

	Log-Likelihood	AIC	BIC
1-component	5432.1217	-10852.2433	-10819.3405
2-component	5447.5549	-10879.1098	-10835.2393

Table 12: AIC and BIC comparison of 1-component versus 2-component Beta-Skew-t-EGARCH.

As we can see, both criteria prefer the 2-component model over the 1-component variant with non-negligible difference between the two.
Altogether, this suggests the use of long-memory models for our sample and implies that adding a long-memory component to the other models might have been sensible, too, for example using the two-component GARCH model of [EL1], instead of its standard counterparts. We should keep this advantage of the two-component model in mind for the further model analysis.

So far, we have looked into the model estimates, noting the possible flaws and potentially unnecessary parameters and looked into their properties one-by-one. To compare all our possible models' performances at once, one might "rank" them according to the previously used AIC and BIC criteria. However, we first have to face the following difficulties:
We used different response values in the modelling section, by working with meaned data at first and switching to demeaned data in the end with the Beta-Skew-t-EGARCH models. Even though the difference is not that big, when comparing single return values, these might stack and potentially lead to faulty decisions. In general, inference based on AIC and BIC is only valid, if the return sample, that the models are based on, is the same in all cases.
To fix this problem, we remove the mean coefficient in all models and work exclusively with demeaned data for this comparison.
Another thing we should keep in mind is that we employed as much as five different model implied distributions: A normal distribution for GARCH, student's t-distribution for GARCH-t, EGARCH-t, GJR-GARCH-t, TGARCH-t and t-GAS, a skewed t-distribution for the Beta-t-EGARCH models and two differently compounded normal distributions

for the stochastic volatility models. Our main goal is to decide, which model yields the best volatility estimates. However, by using the AIC and BIC, we also incorporate the fit of the model implied distribution, which might mask the quality of the estimates. We have seen before that the data's distribution could be nicely approximated by a t-distribution with skew - hence, models with this implied distribution are likely going to be highly preferred by the AIC and BIC, compared to models with gaussian model distribution, even though the differences in volatility estimates might be minuscule.
This can best be illustrated by the comparison of the GARCH and GARCH-t models (here, with μ and σ_0 estimates included). There is only little difference in the volatility estimates, but relatively large difference in log likelihood:

Mean absolute difference in %:	1.861	
Gaussian Log-Likelihood vs. t:	5360.690	5384.657

Obviously, the increase is only made possible, by the differing choice of distribution, as just the degrees of freedom parameter ν was added, which solely affects the MLE. It is not clear, though, how big the influence of the slightly differing volatility estimates is. We can see that the model implied distribution plays a great role in the AIC and BIC calculations, which should be remembered, when comparing the values.

To make our life easier and not re-compute every model's estimates manually without mean coefficient, we use the respective existing implementations for all but the GAS and TGARCH models, as the implementations from the respective packages are suboptimal. We also omit the estimation of the starting value and replace it by the sample standard deviation, as our goal here is to simply compare the models to one another. As usual, the implementation and more details can be found in the Appendix.
The results are given in the following table:

	k	Log Likelihood	AIC	BIC
GARCH(1,1)	3	5359.046	-10712.092	-10695.640
GARCH-t(1,1)	4	5381.519	-10755.038	-10733.103
t-GAS(1,1)	4	5381.720	-10755.440	-10733.505
EGARCH-t(1,1)	5	5425.424	-10840.848	-10813.429
GJRGARCH-t(1,1)	5	5417.135	-10824.270	-10796.851
TGARCH-t(1,1)	5	5427.292	-10844.583	-10817.164
1-comp. Beta-t-EGARCH	5	5425.154	-10840.308	-10812.889
1-comp. Beta-Skew-t-EGARCH	6	5432.122	-10852.243	-10819.340
2-comp.Beta-t-EGARCH	7	5441.211	-10868.42	-10830.04
2-comp. Beta-Skew-t-EGARCH	8	5447.555	-10879.110	-10835.239

Table 13: AIC and BIC comparison of all models, where k denotes the number of parameters. In these calculations, the volatility starting value and mean were omitted.

Note that the SV-models can't be included in the comparison, as they weren't estimated by maximum likelihood estimation, which is necessary for an AIC/BIC comparison. The log likelihood values are also not represented, even though we could have used,

for example, the respective medians as fitted volatility estimates. However, as their likelihoods aren't available in closed form and would have to be approximated, they are left out.
The results from the table can be split according to the added model properties, which we covered in the overview of the previous section: First comes the standard GARCH model, which is the most basic one. Its gaussian model distribution is not very accurate, which is why the t-distribution is used for the other models. Hence, another parameter ν for the distribution's degrees of freedom is added.
Now one can choose between a score driven volatility update, as in the GAS model, or the standard GARCH driving factor y_t^2. Which one to choose is not obvious, as the resulting AIC and BIC are very similar. However, as we have noted before, score updating is reacting less sensitive to single outliers. Thus, one might prefer the GAS model in times of large, short-term volatility spikes, while a GARCH-t model could be preferred for its comparative simplicity.
We have seen that financial returns are affected by the leverage effect. Hence, we also used models that update the volatility asymmetrically, depending on the signs of the observed returns. These are the 5-parameter-models, which differ mostly in their choice of driving component and modelled time-varying parameter. While all these models are preferred over the aforementioned ones without leverage component, the TGARCH model yields the smallest AIC and BIC and is hence to be the preferred one here, unless we consider a case with lots of outliers again. Then, the score driven one-component Beta-t-EGARCH should be chosen, which reacts slightly weaker to outliers.
Next, we added a skew-parameter, which we have done only for the Beta-t-EGARCH model. As a more extensive comparison, this could have been done for all of the above models. However, this one application can be considered sufficient for this study. Again, the additional parameter is sensible, according to the AIC, BIC and the conclusions we drew about the distribution of our return sample in the previous part.
The same holds for the inclusion of a long-memory component, which seems to be more influent, than the addition of the skew parameter, considering that the two-component Beta-t-EGARCH model outperforms the one-component-Beta-Skew-t-EGARCH model, with respect to AIC and BIC.
On the very top, rated by AIC and BIC, is the two-component Beta-Skew-t-EGARCH model. Again, this is not a big surprise, as it specifically covers all the properties of the data sample we discussed.
Remember that we used three different likelihood functions here, therefore the magnitude of improvement, when switching from a gaussian to a t-distribution and then to a skewed-t distribution, has to be carefully evaluated, if one is most interested in the volatility estimates and not the "overall model fit". The difference between the one-comp. Beta-t-EGARCH estimates with and without skew is notably small, even smaller than the one between GARCH and GARCH-t:

Mean absolute difference in %:	0.611	
Log-Likelihood without/with Skew:	5425.154	5432.122

As we can not fully rely on the criteria here, the inclusion of the skew parameter should rather be seen as a choice based on knowledge of the data sample. Whether the approximately half percent average difference in estimates is to be seen as palpable and legitimates the inclusion of a parameter, depends on one's goal. Trying to explain the behavior of the return data, its inclusion certainly makes sense.

By the use of AIC and BIC, we have confirmed that the inclusion of leverage and long memory components in our models is sensible, but there was only little difference between using score driven or comparable GARCH models. To look into the differences, we check the impact of a single return on each model's update. To do this, we consider the relative impact of each new observation, conditional on the previous values, calculated by the difference in volatility estimates. Thus, we quantify the impact of any return y_t, with $\sigma_t - \sigma_{t-1}$, where σ_0 denotes the starting value. The result can be visualized as follows:

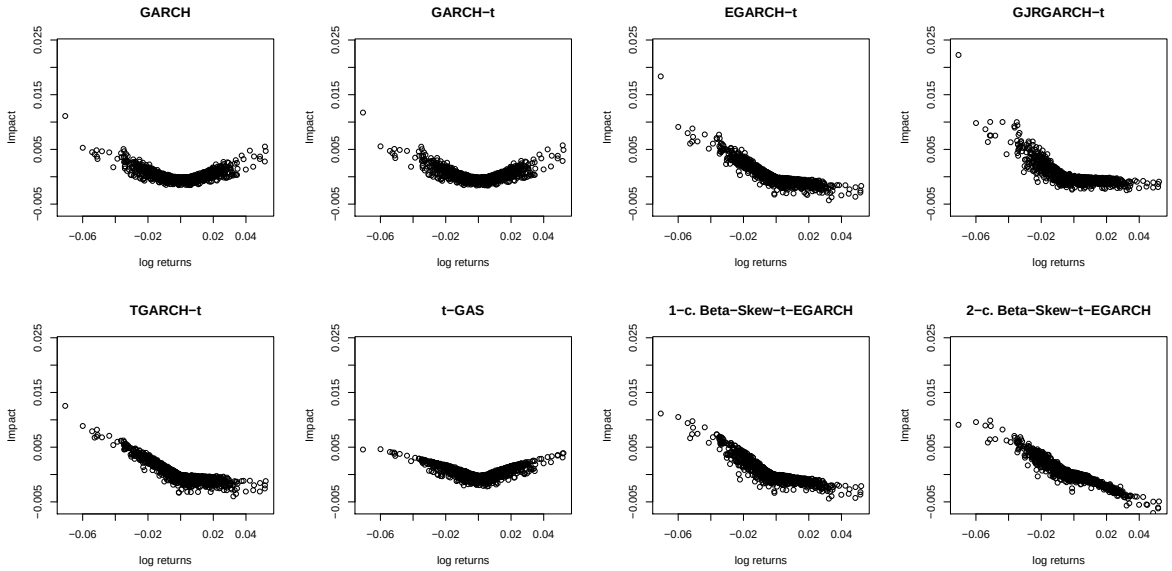

Figure 20: The relative impact of each single (log) return is depicted for all previously handled models.

Here, we can clearly see the difference that the leverage component makes. Previously, when looking into the coefficient estimates in the beginning of this section, we noted that the impact of positive returns is probably very close to zero in the leverage models. Hence, their volatility estimates always decrease by the persistence factor in times of positive returns, nearly unaffected by their magnitude. This effect is even stronger in the long-memory model, where large positive returns additionally decrease the estimates, resulting in a nearly linearly decreasing impact shape, which strongly suggests the presence of a leverage effect. However, one should keep in mind that large positive returns most often are observed as countermovements, following (a series of) large negative returns. Hence, whether these leveraged impacts are realistic and don't lead to a short-term underestimation, remains unclear.

Besides this, we can note that the score driven models react less strongly to large returns, confirming our previous considerations and observations from the graphs. The four models in the middle give a direct comparison of the similar GARCH-t vs. the t-GAS model and the EGARCH-t vs. the one-component Beta-Skew-t-EGARCH model. In contrast to the upper two models, the lower ones show very conservative updates after the larger absolute returns - specifically after the lowest return, caused by the Brexit referendum. This suggests the use of score-driven updating functions as a more robust, but also more complex alternative to GARCH models.
Now that we have summarized the effects of every property our models have and ranked the candidates according to their in-sample fit to the data at hand, we proceed with the evaluation of their forecasting performance in a short out-of-sample analysis.

4.2 Out-of-sample

In this part we briefly discuss the forecasting of volatility in general and use the previously fitted models, to forecast the DAX volatility for january 2017.

When forecasting volatility, there is one big issue that has to be considered: The standard deviation is a latent parameter and can't be directly observed. For in-sample modelling, this is resolved by using some functions of the returns, to approximate the volatility. This procedure can be continued, to acquire out-of-sample forecasts, too. However, unlike in a standard regression environment, we can't explicitly measure the goodness of these forecasts. In particular, using the (demeaned) squared daily returns as an unbiased estimator for the variance, as employed in the GARCH models, might lead to inaccurate forecast evaluations. This stems from the fact that it is a very noisy estimator, as [AB1] have pointed out. They suggest using intraday returns instead, as a more exact measure of volatility. However, without the intraday data at hand, we have to resort to using the interday realized volatility as a volatility measure. Thus, this part of the analysis is kept relatively short and only encompasses the quantile forecasts for each model, spanning over 22 days. Additionally, the model parameters are not reestimated in every step, by the use of the simulated returns.
Each covariance stationary model reverts to its unconditional mean and standard deviation in the absence of shocks. Thus, all volatility forecasts approach the unconditional standard deviation over time - how quickly it is reached, depends on the starting point of the forecast and the persistence of past shocks.
To acquire forecasts for all models, we simulate many paths for all models. This is done, by sampling returns from the corresponding error distributions and rescaling them to fit the currently fitted standard deviation. Using these sampled returns, we calculate the volatility estimates for every model and their respective sample mean and quantiles, using the median as the forecasting value, which is then evaluated by several loss functions.
The models considered are mostly the same as in the AIC/BIC comparison. For convenience, different implementations are applied in the estimation process for the TGARCH and scaled t-GAS model, using the pre-existent packages. More details can be found in

the Appendix, containing the code.
We start off by displaying five resulting quantiles and the sample mean graphically. It should be noted that all, but the SV model's first forecasted value is actually given deterministically from the last return in 2016, which means only 21 returns have to be sampled. The last estimate from 2016 is included, being the starting point for the respective forecasts:

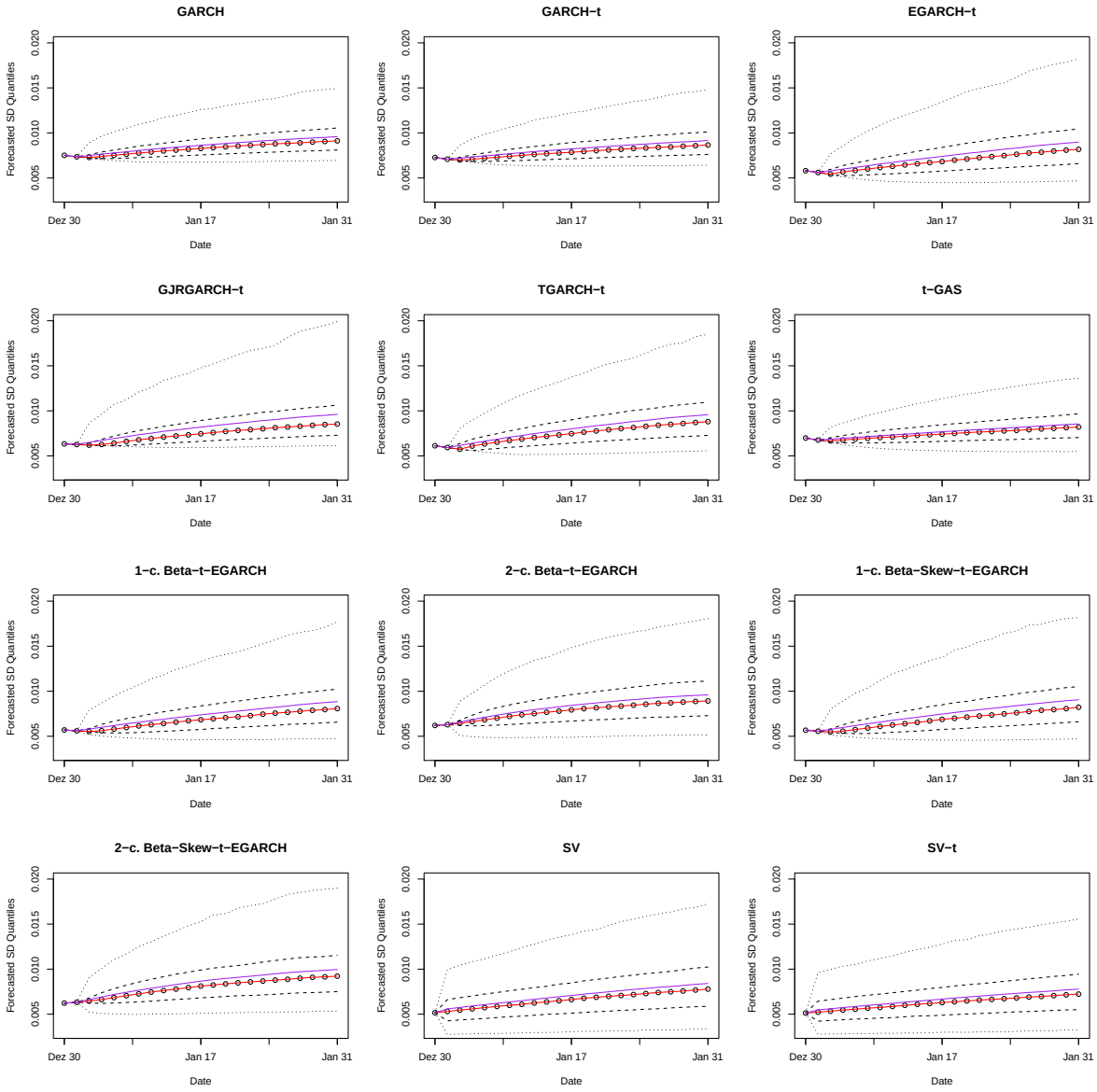

Figure 21: Forecasting quantiles based on 10.000 samples for each model: Dotted lines denote the 2.5%/97.5% quantiles, the dashed lines are the 25%/75% quantiles and the points with the red line show the median. Additionally, the mean is displayed by the purple lines. It should be noted that the t-GAS forecasts are represented by a Beta-t-EGARCH model without leverage.

What all forecasts have in common, is that they increase over the time period. This is because the starting values are relatively small - particularly for the SV models and models with leverage - and the forecasts tend towards the unconditional standard deviation.
We will use the median of the simulated volatility values as the forecast for each model. This necessarily results in lower estimates, than if the mean was used, as the upper forecast quantiles are notably further away from the median than the lower ones. Exemplary considering the Beta-t-EGARCH models, one can see that the expected values of the future standard deviations (conditional on the information up to time t), are not easily computed. They are given by

$$\sigma_\epsilon \cdot \mathbb{E}_t\left[\sigma_{t+n}\right] = \sigma_\epsilon \cdot \exp\left(\omega + \phi_1^n \lambda_t^\dagger\right) \prod_{i=1}^{n} \mathbb{E}_t\left[\exp\left(\phi_1^{n-i}\left(\kappa_1 u_t + \kappa^* \operatorname{sgn}(-\epsilon)(u_t+1)\right)\right)\right] \tag{4.1}$$

for the n steps ahead forecasts of one-component models. The notation used is the same, as in the model introduction before. The 2-component version is

$$\begin{aligned}\sigma_\epsilon \cdot \mathbb{E}_t\left[\sigma_{t+n}\right] = \sigma_\epsilon \cdot \exp\left(\omega + \phi_1^n \lambda_t^\dagger + \phi_2^n \lambda_t^\dagger\right) \cdot \\ \cdot \prod_{i=1}^{n} \mathbb{E}_t\left[\exp\left(\phi_1^{n-i}\kappa_1 u_t + \phi_2^{n-i}\left(\kappa_2 u_t + \kappa^* \operatorname{sgn}(-\epsilon)(u_t+1)\right)\right)\right].\end{aligned} \tag{4.2}$$

In both cases, the expected value on the right can only be approximated with simulation methods for $n > 1$, while 1-step ahead forecasts can be acquired, by simply using the updating function.
To see, which model forecasts fare best, we apply several loss functions (see Basics), to measure the discrepancy between the forecasts and the actual squared, demeaned returns. The coefficient of determination is also included, as a measure of general quality of the forecasts.

	MSE x10^{10}	MAE x10^6	MSD x10^6	QLIKE	R^2
GARCH(1,1)	59.793	60.129	17.624	34.493	0.021
GARCH-t(1,1)	58.329	57.319	11.103	30.310	0.045
EGARCH-t(1,1)	**55.773**	50.166	-3.019	18.609	**0.086**
GJRGARCH-t(1,1)	56.320	54.012	4.886	24.642	0.077
TGARCH-t(1,1)	**55.581**	53.873	5.400	23.759	**0.090**
t-GAS/Beta-t-EGARCH	57.760	54.453	4.561	26.087	0.054
1-c. Beta-t-EGARCH	56.098	50.027	-3.628	18.381	**0.081**
2-c. Beta-t-EGARCH	57.228	57.059	11.167	28.798	0.063
1-c. Beta-Skew-t-EGARCH	**55.762**	50.047	-3.201	18.538	**0.087**
2-c. Beta-Skew-t-EGARCH	57.574	57.983	13.795	30.028	0.057
SV	56.933	**49.158**	-6.457	**17.172**	0.067
SV-t	58.575	**47.632**	-10.931	**14.744**	0.041

Table 14: Evaluation of all forecasts, using the MSE, MAE, MSD and QLIKE loss functions, as well as the coefficient of determination. For better displayability, some loss functions were rescaled and the best values were emphasized.

It can be noted that the MSE and R^2 suggest different models, than the MAE and QLIKE loss functions. This is probably due to some relatively large deviations in the SV and SV-t models, whose effect is amplified by the square term. From the MSD values we can deduct that some forecasts, particularly the GARCH(1,1) ones, gravely overestimate the variance.
In summary, the outcome is highly inconclusive and the R^2 values indicate a bad forecasting performance for all models. Both of these facts can be attributed to the poor accuracy of the proxy, as [AB1] have remarked. How bad it is becomes evident, by simply looking at its values:

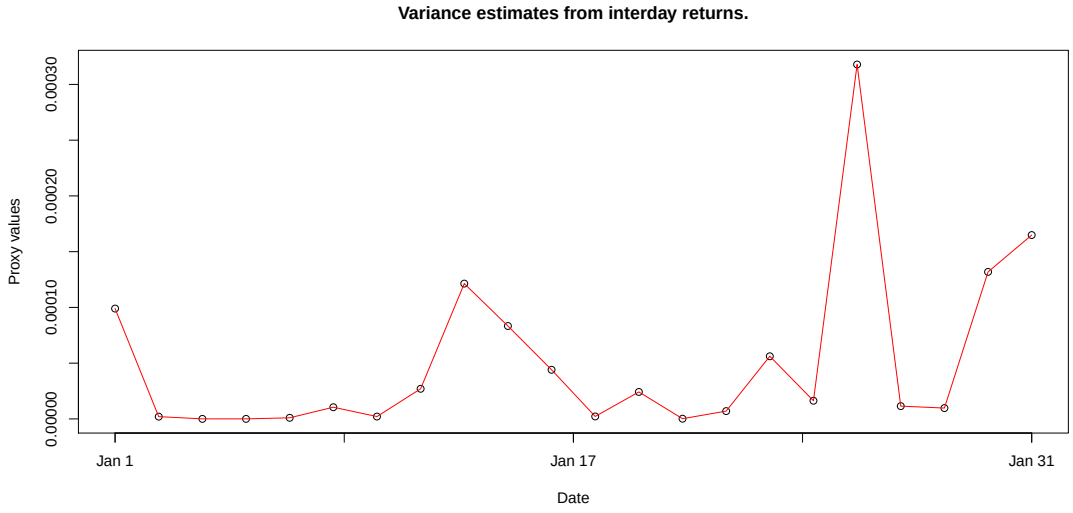

Figure 22: Variance estimates, given by the squared, demeaned returns.

Most values are very close to zero and there is one extraordinarily large value. Obviously, this can't be considered a good replacement for the true variance in the forecasting evaluation. Without the availability of a good proxy, such as one that incorporates intraday data, a forecast evaluation is bound to be inaccurate.
Due to this, the forecasting part is intentionally kept short, as a sensible analysis is not possible with the data at hand.

In summary, only considering the in-sample analysis in the first subsection, we have seen that the 2-component Beta-Skew-t-EGARCH model fares best, according several criteria and due to the fact that it covers all properties the return data has. Sadly, the comparison to SV models proved difficult, without appropriate ways to measure each models' fit to the data and subsequently their forecasting performance.
All further details, such as, whether the mean and starting value σ_0 should also be estimated, using the maximum likelihood method, have to be decided with respect to the model's purpose. It can be said that there is no "perfect" model, just that some are more useful than others, depending on one's goals - be it forecasting, or inference based on the models.
If we had intraday return data for the whole or just a part of the time-frame, we could have further assessed the quality of 1- or n-step ahead forecasts. This way, it were possible to look into the volatility modelling quality better, as for example the information criteria AIC and BIC are obscured by the shape of the model implied density.

5 Conclusion

The purpose of this self study was, to get a feeling for volatility analysis and the potential issues one might encounter during modelling and forecasting in general. This was done by firstly covering the very basics, in order to lay the groundwork and specify the notation and methodology we used. Following this, we analyzed the data set at hand and used the findings, to choose appropriate models, which were introduced in detail in the next section. After covering their properties and the motivation behind their specifications, we applied them on the return sample and evaluated the outcome in an in- and out-of-sample analysis. However, we found the data at hand insufficient for a proper forecasting evaluation. Altogether, we noted that the two-component Beta-Skew-t-EGARCH model seems the most promising, when modelling financial returns, as it models not only the autoregressive conditional heteroscedasticity, but also includes the heavy tails in its volatility updating function, considers the long-memory property and leverage effect and possible skewness.
As this initial work can't be considered a complete analysis, there are several topics that could be expanded and things, which could be included:

- We have already seen that interday closing prices don't capture the whole volatility. The day of the US election on the 8th November 20016 can be considered a prime example: Following the unanticipated victory of Donald Trump, the DAX plummeted by 300 points ($\approx 3\%$) between the closing on 8th november and opening on the 9th. However, the course rallied up again by about 450 points during the 9th and closed with a plus of approximately 150 points, compared to the previous closing prize. Hence, the closing prize completely failed to capture the unusually large variation of the index course.
  A possible solution to this would be to simply incorporate the opening prizes in the models, as well as the daily highs and lows of the index, for a more accurate image of the course variation. Alternatively, if one wanted to model only one value, the intraday range (difference between highest and lowest course) could be used.

- Another indicator of high volatility is a large trading volume. Including it in the models in some way might result in a more accurate outcome.

- As will be obvious, when looking into the appendix, the biggest part of the implementation is formulated suboptimally in a numerical sense. The computational time could be vastly improved, by not only using some quicker functions and less loops, but also by running some of them via C++, using the "Rcpp" package from [Ed1]. In particular, this would be necessary, if the data set was larger, or more computationally intensive (forecasting) procedures were used.

- In some parts, like for example test applications, the approaches were somewhat sloppy, in the way that assumptions were not always fully considered and better alternatives, with potentially higher test power, might have been used. Also, most of the models used, have already been studied in detail and several criteria on the

parameters are known, which ensure properties like stationarity and ergodicity, or asymptotic normality for the QMLE. However, including these would have further increased the size and complexity of this already rather lengthy work.

- The analysis of the SV models is relatively short and could be extended, by including numerical approximations of their likelihood functions and a more in-depth comparison to the deterministic models. Moreover, other SV-type models, which can handle more properties, like for example leverage and long-memory behavior, could be considered.

- Forecast evaluation proved to be impossible, without a more appropriate variance proxy. Besides fixing this, by involving intraday returns and possibly the trading volume, other forecasting forms, such as rolling forecasts could have been used, which re-estimate the parameters with every step, instead of keeping them fixed.

In summary, this can be considered a relatively broad introductory work, which levels the path for a more advanced and detailed analysis on more extensive data.

6 Appendix

6.1 Parameter estimates

GARCH(1,1)				
α	β	ω	μ	σ_0
0.09148727	0.88937884	0.00000342	0.00077041	0.00975188
0.01451864	0.01716830	0.00000106	0.00025412	0.00398653
0.00000000	0.00000000	0.00124756	0.00243181	0.01443689
GARCH-t(1,1)				
α	β	ω	μ	σ_0
0.09922232	0.89009806	0.00000275	0.00093662	0.00830526
0.01865049	0.01983086	0.00000120	0.00024072	0.00638834
0.00000010	0.00000000	0.02175713	0.00009985	**0.19357898**
ν				
6.73571355				
1.18484835				
0.00000001				
EGARCH-t(1,1)				
α	β	ω	μ	σ_0
-0.18041585	0.96007592	-0.35860670	0.00050250	0.00640753
0.02443955	0.01003728	0.08879087	0.00025087	0.00360862
0.00000000	0.00000000	0.00005373	0.04517614	**0.07579569**
ν	γ			
6.95727182	0.14954981			
1.19081090	0.02513032			
0.00000001	0.00000000			
GJR-GARCH-t(1,1)				
α	β	ω	μ	σ_0
0.00000013	0.87538166	0.00000406	0.00063223	0.00393596
0.00155037	0.01937853	0.00000121	0.00023295	0.00297991
0.99993239	0.00000000	0.00083156	0.00664712	**0.18655821**
ν	γ			
6.84166515	0.20669337			
1.14127117	0.03607877			
0.00000000	0.00000001			

TGARCH-t(1,1)				
α^+	α^-	β	ω	μ
0.00000001	-0.18805228	0.88998965	0.00047454	0.00048758
0.00025599	0.02507588	0.01651463	0.00011702	0.00021936
0.99997907	0.00000000	0.00000000	0.00005006	0.02623249
σ_0	ν			
0.00592699	7.29036035			
0.00402993	1.29773932			
0.14136060	0.00000002			
t-GAS(1,1)				
α	β	ω	μ	σ_0
0.08640802	0.98300683	0.00000313	0.00093579	0.01036361
0.01415410	0.00903459	0.00000116	0.00024028	0.00504931
0.00000000	0.00000000	0.00703580	0.00009837	0.04012282
ν				
7.59224891				
1.43613295				
0.00000012				
1-component Beta-Skew-t-EGARCH				
ω	ϕ_1	κ_1	κ^*	ν
-4.60975113	0.95755244	0.04808598	0.06497380	6.43304229
0.05323711	0.00947057	0.00769812	0.00888649	0.89344592
0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
γ				
0.89447715				
0.02695057				
0.00000000				
2-component Beta-Skew-t-EGARCH				
ω	ϕ_1	ϕ_2	κ_1	κ_2
-4.65698602	0.98361038	0.91899486	0.04615514	-0.02488514
0.08763709	0.00815452	0.01099049	0.01627550	0.02059273
0.00000000	0.00000000	0.00000000	0.00457015	**0.22687694**
κ^*	ν	γ		
0.07835134	6.50270628	0.89795596		
0.00750487	0.91548007	0.02629293		
0.00000000	0.00000000	0.00000000		

Table 15: Estimated parameters with standard errors and t-test p-values for all models. The parameter values that can be considered insignificant at a 5% significance level are emphasized.

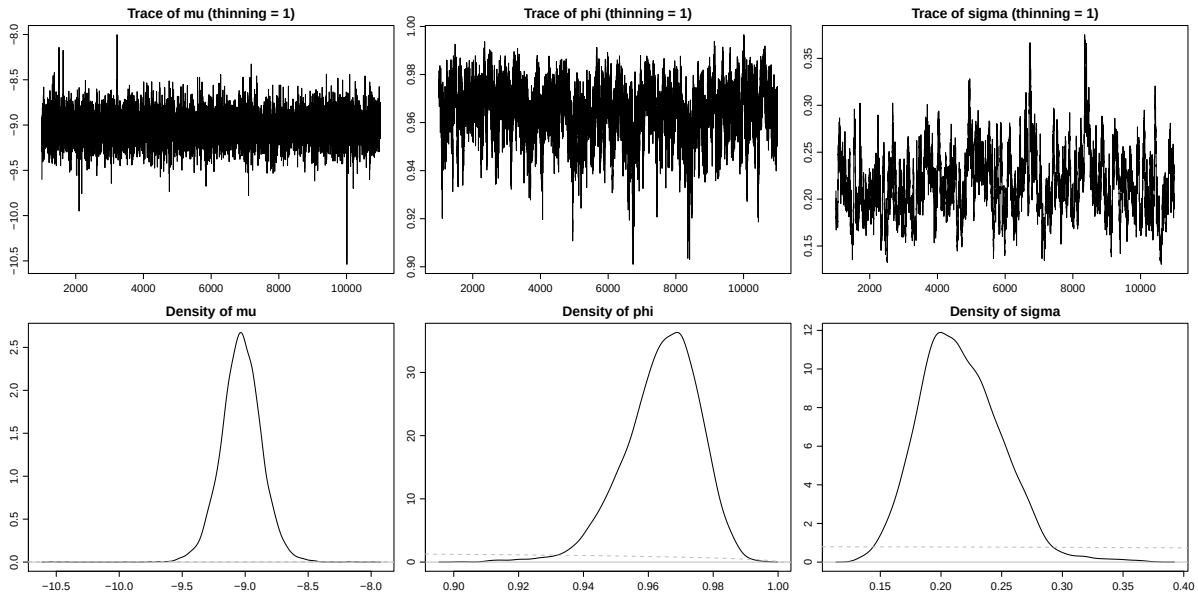

Figure 23: SV parameter plots, showing the trace of the MCMC and the respective parameter densities. The prior distribution is denoted by the dashed gray line.

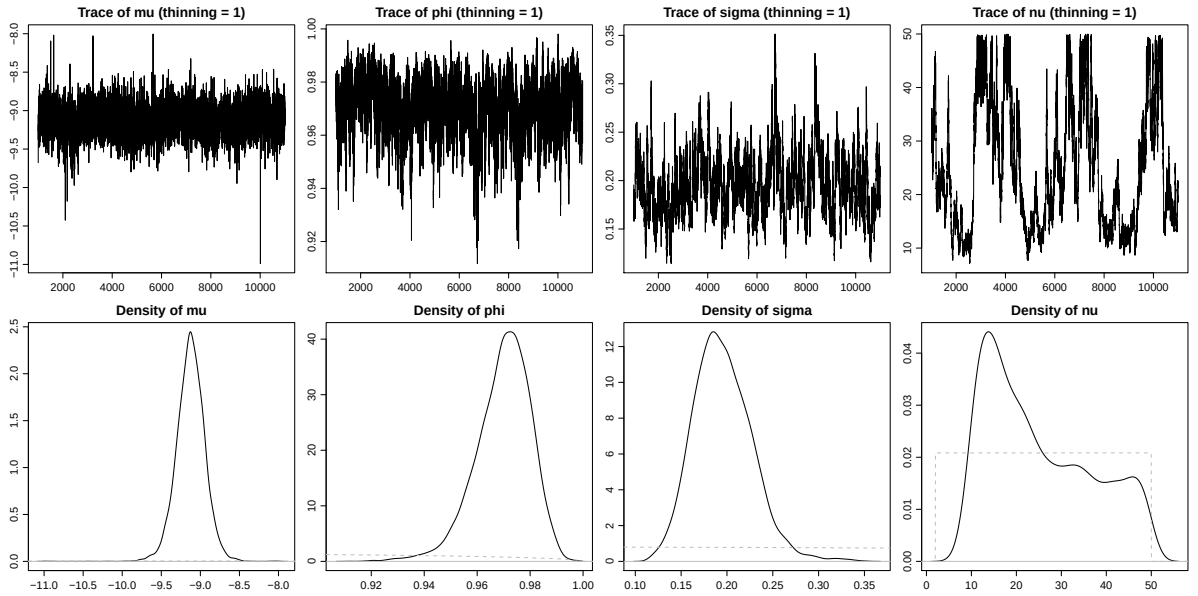

Figure 24: SV-t parameter plots, showing the trace of the MCMC and the respective parameter densities. The prior distribution is denoted by the dashed gray line.

6.2 R-Code

Loading the Data

```
#Deleting old environment
rm(list=ls())

#Loading data from working directory
DAX <- read.csv("DAX2010-2016.csv", header=TRUE, sep=";", dec=",")
DAX[,1] <- rev(DAX[,1]) #reversing data s.t. it starts in 2010
DAX[,2] <- rev(DAX[,2])
DAX[,3] <- rev(DAX[,3])
DAX[,4] <- rev(DAX[,4])
DAX[,5] <- rev(DAX[,5])
DAX[,6] <- rev(DAX[,6])
str(DAX) #1780 open, high, low, close and volume values with dates

#Loading out-of-sample data:
DAX2 <- read.csv("DAX2017-01.csv", header=TRUE, sep=";", dec=",")
DAX2 <- DAX2[-23,] #removing NA value in the end
DAX2[,1] <- rev(DAX2[,1]) #reversing
DAX2[,2] <- rev(DAX2[,2])
DAX2[,3] <- rev(DAX2[,3])
DAX2[,4] <- rev(DAX2[,4])
DAX2[,5] <- rev(DAX2[,5])
DAX2[,6] <- rev(DAX2[,6])
str(DAX2) #22 values, same parameters as above

#Loading packages
library(Rsolnp) #For solnp (nonlinear optimization)
library(stats) #For AR order checking
library(GAS) #For Rsolnp, MASS und GAS
library(rugarch) #For GARCH
library(betategarch) #For Beta-t-EGARCH
library(FinTS) #LM-ARCH, Ljung-Box-Tests and Acf plots
library(forecast) #For Acf function
library(moments) #For moment(), skewness(), kurtosis()
#library(nortest) #For some normality tests
#library(sn) #For skewed normal and t-test
library(stochvol) #For SV models
library(xtable) #For Sweave-tables
library(Hmisc) #Has useful "latex()"-function
library(tseries) #For ADF test

#Defining Closing prices and (logarithmic) returns
SK <- as.numeric(DAX$Schlusskurs) #Closing prices
SK2 <- as.numeric(DAX2$Schlusskurs) #Out-of-sample closing prices
nSK <- length(SK)
```

```
DATUM <- as.character(DAX[,1]) #Dates
DATUM <- strptime(DATUM, format="%d.%m.%Y", tz="UTC")
DATUM2 <- as.character(DAX2[,1])
DATUM2 <- strptime(DATUM2, format="%d.%m.%Y", tz="UTC")

SKR <- diff(log(SK)) #Log returns
d_ret <- SKR-mean(SKR) #Demeaned log returns
```

Preparations for Output

```
##SECTION 1: Introduction




##SECTION 2: The Data

#Checking the AR order of SK and SKR for the ADF test
ar.mle(SK)  #recommends order 7, according to AIC
for (i in 1:10)
  {
   print(c("ARMA lag order: ", i), quote=FALSE);
   print(AutocorTest(arma(SK, order=c(i,0))$residuals))
  }   #suggests 5-6 as lag order
ar.mle(SKR)  #suggests 5 as lag order
for (i in 1:5)
  {
   print(c("ARMA lag order: ", i), quote=FALSE);
   print(AutocorTest(arma(SKR,order=c(i,0))$residuals))
  }   #suggests 2-4 as lag order

#ADF test
DFstat1 <- as.numeric(round(c(adf.test(SK,k=5)$statistic, adf.test(SK,k=7)$statistic),
                            digits=3))
DFstat2 <- as.numeric(round(c(adf.test(SKR,k=2)$statistic, adf.test(SKR,k=5)$statistic),
                           digits=3))
DFp1 <- round(c(adf.test(SK,k=5)$p.value, adf.test(SK,k=7)$p.value), digits=3)
DFp2 <- round(c(adf.test(SKR,k=2)$p.value, adf.test(SKR,k=5)$p.value), digits=3)
ADFtab <- rbind(DFstat1, DFp1,DFstat2,DFp2)
rownames(ADFtab) <- c("Test statistics for DAX, with AR order of 5 and 7: ",
                      "Corresponding p-values: ",
                      "Test statistics for DAX log returns, with AR order of 2 and 5: ",
                      "Corresponding p-values(too low to display): ")


#ARCH LM testing of DAX returns
lag <- c(3, 8, 15, 25)
F_statistic <- rep(0, 4)
p_value1 <- rep(0, 4)
for (i in 1:4){
  F_statistic[i] <- ArchTest(SKR, lag=lag[i])$statistic;
  p_value1[i] <- ArchTest(SKR, lag=lag[i])$p.value;
```

```
  }
ARCHLM <- data.frame(lag=as.integer(lag), F_statistic=F_statistic, p_value=p_value1)


#Ljung-Box testing of squared, demeaned DAX returns
Q_statistic <- rep(0, 4)
p_value2 <- rep(0, 4)
for (i in 1:4){
  Q_statistic[i] <- AutocorTest((SKR-mean(SKR))^2, lag=lag[i])$statistic;
  p_value2[i] <- ArchTest(SKR, lag=lag[i])$p.value;
}
LB <- data.frame(lag=as.integer(lag), Q_statistic=Q_statistic, p_value=p_value2)


#Kurtosis and Skewness
SKRkurt <- kurtosis(SKR)
SKRskew <- skewness(SKR)


#Q-Q-Plots
fdest<-fitdistr(SKR,"t") #gives a few warnings, because of variable df
fdest_mu<-as.numeric(fdest$estimate[1])
fdest_sig<-as.numeric(fdest$estimate[2])
fdest_df<-as.numeric(fdest$estimate[3])

table1 <- data.frame(
      mean=c(mean(SKR), fdest_mu),
      standard_deviation=c(sd(SKR), fdest_sig),
      degrees_of_freedom=c(NA,fdest_df),
      row.names=c("normal", "student's t")
      )
colnames(table1) <- c("mean/location", "standard deviation/scale", "degrees of freedom")

pts <- ppoints(length(SKR)); #equally spread points in (0,1)
quants <- quantile(SKR, probs=pts); #sample quantile estimates


#Approximating density histogram
f.den <- function(t) dnorm(t, mean(SKR), sqrt(var(SKR)))
f.den2 <- function(t) dt((t-fdest_mu)/fdest_sig, df=fdest_df) / fdest_sig


#Effect of single returns on overall sample standard deviation
hatw <- rep(0, length(SKR))
for(i in 1:length(SKR))(hatw[i] <- (sd(SKR)-sd(SKR[-i]))/sd(SKR)*100)

MA4<-function(n, y)
{
    out<-rep(0, length(y));
    if(n%%2 == 0)
    {
        for(i in (n/2+1):(length(y)-n/2+1))(out[i] <- mean(y[(i-n/2):(i+n/2-1)]));
    };
```

```
    if(n%%2 == 1)
    {
        for(i in (n/2+2):(length(y)-n/2+1))(out[i] <- mean(y[(i-n/2+1):(i+n/2-1)]));
    };
    return(out)
} #Moving-Average type function

dhat <- data.frame(infl_rank=as.integer(rank(-hatw)), index=1:length(SKR),
                   date=DATUM[-length(SKR)], log_ret=SKR,rel_infl=hatw)
dhat <- dhat[with(dhat, order(infl_rank)), ]     #sort by rank

dhat2 <- dhat[dhat$infl_rank<11,]
dhat2$log_ret <- dhat2$log_ret*100
colnames(dhat2) <- c("Influence Rank", "Index","Date",
                     "(log) return in %", "Relative influence in %")
dhat2$Date <- as.character(dhat2$Date)


#Checking leverage effect
levcor <- data.frame(round(c(cor(SKR[-1]^2, SKR[-1779], method="pearson"),
                             (sqrt(length(SKR)-1))^(-1)*1.96), digits=4))
rownames(levcor) <- c("Sample correlation coefficient: ", "Boundaries (+/-): ")
colnames(levcor) <- "values"

MA1<-function(n,y)
{
    out<-rep(0,length(y));
    for(i in n:length(y))(out[i]<-sd(y[(i-n):(i-1)]));
    return(out)
} #"local sd function"

MA2<-function(n,y)
{
    out<-rep(0,length(y));
    for(i in (n):length(y))(out[i]<-mean(y[(i-n):(i-1)]));
    return(out)
} #"local mean function"




##SECTION 3: The models

#Model 0: "local standard deviation"


#Model 1: "GARCH(1,1)"
#First, estimation with "rugarch" package:
garch1.spec <- ugarchspec(variance.model = list(garchOrder=c(1,1)),
mean.model = list(armaOrder=c(0,0), include.mean=TRUE))

G1 <- ugarchfit(data=SKR, spec=garch1.spec)
```

```
#Estimation with own implementation (with starting value sigma_0):
LL2<-function(alpha, beta, omega, mu, sig0)
{
    LL_ret <- 1:(nSK-1);
    sigs <- 1:(nSK);
    sigs[1] <- sig0^2 ;
    for(i in 2:(nSK))
      (sigs[i] <- (omega+alpha*(SKR[i-1]-mu)^2+beta*sigs[i-1])) ;
    for(i in 1:(nSK-1))
      (LL_ret[i] <- (-0.5*log(2*pi*sigs[i])-((SKR[i]-mu)^2)/(2*sigs[i]))) ;
    return(list(LL_ret,sigs)) ;
} #returns log likelihood values and sigma^2 for GARCH parameter inputs

solfun <- function(vec)
{
    alpha <- vec[1];
    beta <- vec[2];
    omega <- vec[3];
    mu <- vec[4];
    sig0 <- vec[5];
    LL_ret <- 1:(nSK-1);
    sigs <- 1:(nSK);
    sigs[1] <- sig0^2 ;
    for(i in 2:(nSK))
      (sigs[i] <- (omega+alpha*(SKR[i-1]-mu)^2+beta*sigs[i-1])) ;
    for(i in 1:(nSK-1))
      (LL_ret[i] <- (-0.5*log(2*pi*sigs[i])-((SKR[i]-mu)^2)/(2*sigs[i]))) ;
    return(sum(-LL_ret)) ;
} #negative Log-lik. function for solnp to minimize

sol <- solnp(pars=c(alpha=0.1, beta=0.9, omega=0, mu=mean(SKR), sig0=sd(SKR)),
fun=solfun, LB=c(0,0.7,0,-1,0), UB=c(0.3,1,1,1,1)) #minimizing negative LL

#Log likelihood values and sigma^2 at evaluated for MLE
L2<-LL2(alpha=as.numeric(sol$pars[1]),beta=as.numeric(sol$pars[2]),
omega=as.numeric(sol$pars[3]),mu=as.numeric(sol$pars[4]),sig0=as.numeric(sol$pars[5]))

##Own model data summarized:
GG1<-list(sigma=sqrt(L2[[2]]),llh=L2[[1]],llhs=sum(L2[[1]]),coef=sol$pars)


#Model 2: GARCH-t(1,1)
#First, "rugarch" version:
garch2.spec <- ugarchspec(distribution.model="std",
                variance.model = list(garchOrder=c(1,1)),
                mean.model = list(armaOrder=c(0,0), include.mean=TRUE))

G2 <- ugarchfit(data=SKR, spec=garch2.spec)

#Own version with sigma_0:
LL3 <- function(alpha, beta, omega, mu, nu, sig0)
{
    LL_ret <- 1:(nSK-1);
```

```
    sigs <- 1:(nSK);
    sigs[1] <- sig0^2 ;
    for(i in 2:(nSK))
      (sigs[i] <- (omega+alpha*(SKR[i-1]-mu)^2+beta*sigs[i-1])) ;
    for(i in 1:(nSK-1))
      (LL_ret[i] <- log( gamma((nu+1)/2)/(sqrt((nu-2)*pi*sigs[i])*gamma(nu/2))*
                          (1+(SKR[i]-mu)^2/((nu-2)*sigs[i]))^(-(nu+1)/2) ) ) ;
    return(list(LL_ret, sigs)) ;
}

solfun2 <- function(vec)
{
    alpha <- vec[1];
    beta <- vec[2];
    omega <- vec[3];
    mu <- vec[4];
    sig0 <- vec[5];
    nu <- vec[6];
    LL_ret <- 1:(nSK-1);
    sigs <- 1:(nSK);
    sigs[1] <- sig0^2 ;
    for(i in 2:(nSK))
      (sigs[i] <- (omega+alpha*(SKR[i-1]-mu)^2+beta*sigs[i-1])) ;
    for(i in 1:(nSK-1))
      (LL_ret[i] <- log( gamma((nu+1)/2)/(sqrt((nu-2)*pi*sigs[i])*gamma(nu/2))*
                          (1+(SKR[i]-mu)^2/((nu-2)*sigs[i]))^(-(nu+1)/2) ) ) ;
    return(sum(-LL_ret)) ;
} #Negative log-lik. function

sol2 <- solnp(pars=c(alpha=0.1, beta=0.9, omega=0, mu=mean(SKR), sig0=sd(SKR), nu=5),
fun=solfun2, LB=c(0,0.7,0,-1,0,2.001), UB=c(0.3,1,1,1,1,40)) #Optimization

L3 <- LL3(alpha=as.numeric(sol2$pars[1]), beta=as.numeric(sol2$pars[2]),
omega=as.numeric(sol2$pars[3]), mu=as.numeric(sol2$pars[4]),
nu=as.numeric(sol2$pars[6]), sig0=as.numeric(sol2$pars[5]))

#Summary
GG2<-list(sigma=sqrt(L3[[2]]),llh=L3[[1]],llhs=sum(L3[[1]]),coef=sol2$pars)

#Maximum and mean absolute difference of GARCH and GARCH-t (without starting values)
max(abs((GG2$sigma[-(1:25)]-GG1$sigma[-(1:25)])/GG2$sigma[-(1:25)]*100))
mean(abs((GG2$sigma[-(1:25)]-GG1$sigma[-(1:25)])/GG2$sigma[-(1:25)]*100))


#Model 2E: EGARCH-t(1,1)
#"rugarch" version:
garch2E.spec <- ugarchspec(distribution.model="std",
                           variance.model = list(model="eGARCH",garchOrder=c(1,1)),
                           mean.model = list(armaOrder=c(0,0),include.mean=TRUE))

G2E<-ugarchfit(data=SKR,spec=garch2E.spec)

#Own version with sigma_0:
```

```
solfunEG <- function(vec)
{
  alpha <- vec[1];
  beta <- vec[2];
  omega <- vec[3];
  mu <- vec[4];
  sig0 <- vec[5];
  nu <- vec[6];
  gamma <- vec[7];
  EV <- sqrt((nu-2)/pi)*gamma((nu-1)/2)/gamma(nu/2); #expected value term

  LL_ret <- 1:(nSK-1);
  sigs <- 1:(nSK);
  sigs[1] <- sig0^2 ;
  for(i in 2:(nSK))
    (sigs[i] <- exp(omega+alpha*(SKR[i-1]-mu)/sqrt(sigs[i-1])+
                    gamma*(abs((SKR[i-1]-mu)/sqrt(sigs[i-1]))-EV)+
                    beta*log(sigs[i-1]) )) ;
  for(i in 1:(nSK-1))
    (LL_ret[i] <- log( gamma((nu+1)/2)/(sqrt((nu-2)*pi*sigs[i])*gamma(nu/2)) *
                       (1+(SKR[i]-mu)^2/((nu-2)*sigs[i]))^(-(nu+1)/2) ) ) ;
  return(sum(-LL_ret)) ;
} #Negative LL

#Optimization:
solEG<-solnp(pars=c(alpha=0, beta=0.9, omega=0, mu=mean(SKR),
               sig0=sd(SKR), nu=5, gamma=0), fun=solfunEG,
               LB=c(-0.3,0.7,-1,-1,0,2.001,-0.3),UB=c(0.3,1,1,1,1,40,0.3))

LLEG <- function(vec)
{
  alpha <- vec[1];
  beta <- vec[2];
  omega <- vec[3];
  mu <- vec[4];
  sig0 <- vec[5];
  nu <- vec[6];
  gamma <- vec[7];

  EV <- sqrt((nu-2)/pi)*gamma((nu-1)/2)/gamma(nu/2)
  LL_ret <- 1:(nSK-1);
  sigs <- 1:(nSK);
  sigs[1] <- sig0^2 ;
  for(i in 2:(nSK))
    (sigs[i] <- exp(omega+alpha*(SKR[i-1]-mu)/sqrt(sigs[i-1])+
                gamma*(abs((SKR[i-1]-mu)/sqrt(sigs[i-1]))-EV)+
                beta*log(sigs[i-1]) )) ;
  for(i in 1:(nSK-1))
    (LL_ret[i] <- log( gamma((nu+1)/2)/(sqrt((nu-2)*pi*sigs[i])*gamma(nu/2)) *
                       (1+(SKR[i]-mu)^2/((nu-2)*sigs[i]))^(-(nu+1)/2) ) ) ;
  return(list(LL_ret, sigs)) ;
}
```

```
#Summary:
GGEG<-list(sigma=sqrt(LLEG(solEG$pars)[[2]]),llh=LLEG(solEG$pars)[[1]],
           llhs=sum(LLEG(solEG$pars)[[1]]),coef=solEG$pars) #summary


#Model 2GJR: "GJRGARCH-t(1,1)"
#"rugarch" version:
garch2GJR.spec <- ugarchspec(distribution.model="std",
                      variance.model = list(model="gjrGARCH",garchOrder=c(1,1)),
                      mean.model = list(armaOrder=c(0,0), include.mean=TRUE))

G2GJR<-ugarchfit(data=SKR,spec=garch2GJR.spec)

#Own version with sigma_0:

solfunGJR <- function(vec)
{
  alpha <- vec[1];
  beta <- vec[2];
  omega <- vec[3];
  mu <- vec[4];
  sig0 <- vec[5];
  nu <- vec[6];
  gamma <- vec[7];
  LL_ret <- 1:(nSK-1);
  sigs <- 1:(nSK);
  sigs[1] <- sig0^2 ;
  for(i in 2:(nSK))
    (sigs[i] <- omega+alpha*(SKR[i-1]-mu)^2+gamma*(SKR[i-1]-mu)^2*
                        as.numeric((SKR[i-1]-mu)<0)+beta*sigs[i-1] ) ;
  for(i in 1:(nSK-1))
    (LL_ret[i] <- log( gamma((nu+1)/2)/(sqrt((nu-2)*pi*sigs[i])*
                    gamma(nu/2)) * (1+(SKR[i]-mu)^2/
                    ((nu-2)*sigs[i]))^(-(nu+1)/2) ) ) ;
  return(sum(-LL_ret)) ;
} #negative LL

#MLE:
solGJR<-solnp(pars=c(alpha=0.05, beta=0.9, omega=0, mu=mean(SKR),
                     sig0=sd(SKR), nu=5, gamma=0.05), fun=solfunGJR,
              LB=c(0,0.7,0,-1,0,2.001,0), UB=c(0.3,1,1,1,1,40,0.3))

LLGJR <- function(vec)
{
  alpha <- vec[1];
  beta <- vec[2];
  omega <- vec[3];
  mu <- vec[4];
  sig0 <- vec[5];
  nu <- vec[6];
  gamma <- vec[7];
  LL_ret <- 1:(nSK-1);
  sigs <- 1:(nSK);
```

```
  sigs[1] <- sig0^2 ;
  for(i in 2:(nSK))
    (sigs[i] <- omega+alpha*(SKR[i-1]-mu)^2+gamma*(SKR[i-1]-mu)^2*
       as.numeric((SKR[i-1]-mu)<0)+beta*sigs[i-1] ) ;
  for(i in 1:(nSK-1))
    (LL_ret[i] <- log( gamma((nu+1)/2)/(sqrt((nu-2)*pi*sigs[i])*
                       gamma(nu/2)) *(1+(SKR[i]-mu)^2/((nu-2)*
                       sigs[i]))^(-(nu+1)/2) ) ) ;
  return(list(LL_ret, sigs)) ;
}

GGGJR<-list(sigma=sqrt(LLGJR(solGJR$pars)[[2]]), llh=LLGJR(solGJR$pars)[[1]],
           llhs=sum(LLGJR(solGJR$pars)[[1]]), coef=solGJR$pars)


#Model 2T: "TGARCH-t(1,1)"
#"rugarch" version:
garch2T.spec <- ugarchspec(distribution.model="std",
                    variance.model = list(model="apARCH",
                    submodel="TGARCH", garchOrder=c(1,1)),
                    mean.model = list(armaOrder=c(0,0), include.mean=TRUE))

G2T <- ugarchfit(data=SKR, spec=garch2T.spec)

#own version:
solfunT <- function(vec)
{
  alpha_pl <- vec[1];
  alpha_mi <- vec[2];
  beta <- vec[3];
  omega <- vec[4];
  mu <- vec[5];
  sig0 <- vec[6];
  nu <- vec[7];
  LL_ret <- 1:(nSK-1);
  sig <- 1:(nSK);
  sig[1] <- sig0 ;
  for(i in 2:(nSK))
    (sig[i] <- omega+alpha_pl*(SKR[i-1]-mu)*as.numeric(SKR[i-1]-mu>0)+
                alpha_mi*(SKR[i-1]-mu)*as.numeric(SKR[i-1]-mu<=0)+
                beta*sig[i-1] ) ;
  for(i in 1:(nSK-1))
    (LL_ret[i] <- log( gamma((nu+1)/2)/(sqrt((nu-2)*pi*sig[i]^2)*
                    gamma(nu/2)) * (1+(SKR[i]-mu)^2/((nu-2)*
                    sig[i]^2))^(-(nu+1)/2) ) ) ;
  return(sum(-LL_ret)) ;
} #negative LL

#MLE:
solT<-solnp(pars=c(alpha_pl=0.1, alpha_mi=-0.1, beta=0.9, omega=0,
                   mu=mean(SKR), sig0=sd(SKR), nu=5), fun=solfunT,
            LB=c(0,-0.3,0.7,0,-1,0,2.001), UB=c(0.3,0,1,1,1,1,40))
```

```
LLT <- function(vec)
{
  alpha_pl <- vec[1];
  alpha_mi <- vec[2];
  beta <- vec[3];
  omega <- vec[4];
  mu <- vec[5];
  sig0 <- vec[6];
  nu <- vec[7];
  LL_ret <- 1:(nSK-1);
  sig <- 1:(nSK);
  sig[1] <- sig0 ;
  for(i in 2:(nSK))
    (sig[i] <- omega+alpha_pl*(SKR[i-1]-mu)*as.numeric(SKR[i-1]-mu>0)
          +alpha_mi*(SKR[i-1]-mu)*as.numeric(SKR[i-1]-mu<=0)+beta*sig[i-1] ) ;
  for(i in 1:(nSK-1))
    (LL_ret[i] <- log( gamma((nu+1)/2)/(sqrt((nu-2)*pi*sig[i]^2)*
                  gamma(nu/2)) * (1+(SKR[i]-mu)^2/
                  ((nu-2)*sig[i]^2))^(-(nu+1)/2) ) ) ;
  return(list(LL_ret, sig)) ;
}

#summary:
GGT <- list(sigma=LLT(solT$pars)[[2]], llh=LLT(solT$pars)[[1]],
            llhs=sum(LLT(solT$pars)[[1]]), coef=solT$pars)


#Model 3: "t-GAS(1,1)"
#own implementation:
LL4 <- function(alpha, beta, omega, mu, sig0, nu)
{
    LL_ret <- 1:(nSK-1);
    sigs <- 1:(nSK);
    sigs[1] <- sig0^2 ;
    for(i in 2:(nSK))
      (sigs[i] <- (omega+alpha*(nu+3)/nu*( ((nu+1)*(SKR[i-1]-mu)^2*
             sigs[i-1])/((nu-2)*sigs[i-1]+(SKR[i-1]-mu)^2) -
             sigs[i-1] )+beta*sigs[i-1] )) ;
    for(i in 1:(nSK-1))
      (LL_ret[i] <- log( gamma((nu+1)/2)/(sqrt((nu-2)*pi*sigs[i])*
             gamma(nu/2)) * (1+(SKR[i]-mu)^2/
             ((nu-2)*sigs[i]))^(-(nu+1)/2) ) ) ;
    return(list(LL_ret, sigs)) ;
} #returns LL and variance

solfun3 <- function(vec)
{
    alpha <- vec[1];
    beta <- vec[2];
    omega <- vec[3];
    mu <- vec[4];
    sig0 <- vec[5];
    nu <- vec[6];
```

```
    LL_ret <- 1:(nSK-1);
    sigs <- 1:(nSK);
    sigs[1] <- sig0^2 ;
    for(i in 2:(nSK))
      (sigs[i] <- (omega+alpha*(nu+3)/nu*( ((nu+1)*(SKR[i-1]-mu)^2*
           sigs[i-1])/((nu-2)*sigs[i-1]+(SKR[i-1]-mu)^2) -
           sigs[i-1] )+beta*sigs[i-1] )) ;
    for(i in 1:(nSK-1))
      (LL_ret[i] <- log( gamma((nu+1)/2)/(sqrt((nu-2)*pi*sigs[i])*
           gamma(nu/2)) * (1+(SKR[i]-mu)^2/
           ((nu-2)*sigs[i]))^(-(nu+1)/2) ) ) ;
    return(sum(-LL_ret)) ;
} #Negative LL

sol3 <- solnp(pars=c(alpha=0.1, beta=0.9, omega=0, mu=mean(SKR),
            sig0=sd(SKR), nu=5), fun=solfun3,
            LB=c(0,0.7,0,-1,0,2.001), UB=c(0.3,1,1,1,1,20)) #MLE

L5 <- LL4(alpha=as.numeric(sol3$pars[1]), beta=as.numeric(sol3$pars[2]),
            omega=as.numeric(sol3$pars[3]), mu=as.numeric(sol3$pars[4]),
            nu=as.numeric(sol3$pars[6]), sig0=as.numeric(sol3$pars[5]))

#Summary:
GG4 <- list(sigma=sqrt(L5[[2]]), llh=L5[[1]], llhs=sum(L5[[1]]),
            coef=sol3$pars)

#Versions from "GAS" package, with and without scaling:
gas1.spec <- UniGASSpec(Dist="std", ScalingType="Inv",
              GASPar=list(location=F,scale=T,shape=F))
gas2.spec <- UniGASSpec(Dist="std", ScalingType="Identity",
              GASPar=list(location=F,scale=T,shape=F))

GAS1 <- UniGASFit(gas1.spec, SKR) #Fitting not working, but gives no errors
#Probably problems with inverting the hessian in the optimization process
GAS2 <- UniGASFit(gas2.spec, SKR) #Works fine


#Model 4: "1/2 component Beta-Skew-t-EGARCH"
#Fitting done with the "betategarch" package
#1-component:
G5 <- tegarch(SKR-mean(SKR), initial.values=c(0.02,0.95,0.05,0.01,5,1))
#Initial values needed to be specified - otherwise the optimizing function
#wouldn't converge

#2-component:
G52 <- tegarch(d_ret, components = 2,
             initial.values = c(0.02,0.95,0.9,0.001,0.01,0.005,5,1))


#Model 5: "SV and SV-t"
#Using the stochvol package:
set.seed(1)
SVMG <- svsample(d_ret, draws=10000, burnin=1000) #Gaussian SV
```

```
set.seed(1)
SVMT <- svsample(d_ret, priornu=c(2.00001,50), draws=10000, burnin=1000) #SV-t

#Standard deviation quantiles of the sampled SV processes:
medG<-apply( exp(SVMG$latent/2), 2, function(x)
  ( t(as.numeric(quantile(x,c(0.05,0.5,0.95)))) ) )
medT<-apply( exp(SVMT$latent/2), 2, function(x)
  ( t(as.numeric(quantile(x,c(0.05,0.5,0.95)))) ) )


#Overview:




##SECTION 4: Model Performance Analysis

#Testing existence of ARCH in standardized residuals after modelling:
#Ljung-Box test:
SKRssres <- list() #Calculating squared, standardized residuals
SKRssres$M1 <- ((SKR-GG1$coef[4])/GG1$sigma[-1780])^2
SKRssres$M2 <- ((SKR-GG2$coef[4])/GG2$sigma[-1780])^2
SKRssres$M3 <- ((SKR-GGEG$coef[4])/GGEG$sigma[-1780])^2
SKRssres$M4 <- ((SKR-GGGJR$coef[4])/GGGJR$sigma[-1780])^2
SKRssres$M5 <- ((SKR-GGT$coef[4])/GGT$sigma[-1780])^2
SKRssres$M6 <- ((SKR-GG4$coef[4])/GG4$sigma[-1780])^2
SKRssres$M7 <- ((SKR-mean(SKR))/GG1$sigma[-1780])^2
SKRssres$M8 <- ((SKR-mean(SKR))/GG2$sigma[-1780])^2

Port <- matrix(0,8,4) #Matrix with Portmanteau test results
for(i in 1:8)
  (Port[i,] <- c(AutocorTest(SKRssres[[i]],lag=1)$p.value,
                 AutocorTest(SKRssres[[i]],lag=4)$p.value,
                 AutocorTest(SKRssres[[i]],lag=7)$p.value,
                 AutocorTest(SKRssres[[i]],lag=10)$p.value))
rownames(Port) <- c("GARCH(1,1)","GARCH-t(1,1)","EGARCH-t(1,1)","GJRGARCH-t(1,1)",
                    "TGARCH-t(1,1)","t-GAS(1,1)","1-comp. Beta-Skew-t-EGARCH(1,1)",
                    "2-comp.Beta-Skew-t-EGARCH(1,1)")
colnames(Port) <- c("lag = 1", "lag = 4", "lag = 7", "lag = 10")

#ARCH-LM test:
SKRsres <- list() #Calculating standardized residuals
SKRsres$M1 <- (SKR-GG1$coef[4])/GG1$sigma[-1780]
SKRsres$M2 <- (SKR-GG2$coef[4])/GG2$sigma[-1780]
SKRsres$M3 <- (SKR-GGEG$coef[4])/GGEG$sigma[-1780]
SKRsres$M4 <- (SKR-GGGJR$coef[4])/GGGJR$sigma[-1780]
SKRsres$M5 <- (SKR-GGT$coef[4])/GGT$sigma[-1780]
SKRsres$M6 <- (SKR-GG4$coef[4])/GG4$sigma[-1780]
SKRsres$M7 <- (SKR-mean(SKR))/GG1$sigma[-1780]
SKRsres$M8 <- (SKR-mean(SKR))/GG2$sigma[-1780]

Scor <- matrix(0,8,4) #Score test matrix
for(i in 1:8)
```

```
  (Scor[i,] <- c(ArchTest(SKRsres[[i]],lags=1)$p.value,
                 ArchTest(SKRsres[[i]],lags=4)$p.value,
                 ArchTest(SKRsres[[i]],lags=7)$p.value,
                 ArchTest(SKRsres[[i]],lags=10)$p.value))
rownames(Scor) <- c("GARCH(1,1)","GARCH-t(1,1)","EGARCH-t(1,1)","GJRGARCH-t(1,1)",
                    "TGARCH-t(1,1)","t-GAS(1,1)","1-comp. Beta-Skew-t-EGARCH(1,1)",
                    "2-comp.Beta-Skew-t-EGARCH(1,1)")
colnames(Scor) <- c("lag = 1", "lag = 4", "lag = 7", "lag = 10")

#Autocorrelation function plots of squared standardized residuals
#Confidence intervals:
conf1 <- (sqrt(length(SKR)-(-10:210)))^(-1)*1.96
conf2 <- -conf1


#Testing, whether constant mean model is appropriate
#Box-Ljung test on (non-squared) standardized residuals
Port2<-matrix(0,8,4)
for(i in 1:8)
  (Port2[i,] <- c(AutocorTest(SKRsres[[i]],lag=1)$p.value,
                  AutocorTest(SKRsres[[i]],lag=4)$p.value,
                  AutocorTest(SKRsres[[i]],lag=7)$p.value,
                  AutocorTest(SKRsres[[i]],lag=10)$p.value))
rownames(Port2) <- c("GARCH(1,1)","GARCH-t(1,1)","EGARCH-t(1,1)","GJRGARCH-t(1,1)",
                     "TGARCH-t(1,1)","t-GAS(1,1)","1-comp. Beta-Skew-t-EGARCH(1,1)",
                     "2-comp.Beta-Skew-t-EGARCH(1,1)")
colnames(Port2) <- c("lag = 1", "lag = 4", "lag = 7", "lag = 10")

#AIC and BIC comparison of several ARMA orders for GARCH and EGARCH
#Model fitting with "rugarch" package
GAR1spec <- ugarchspec(variance.model = list(garchOrder=c(1,1)),
                           mean.model = list(armaOrder=c(0,0),include.mean=F))
GAR2spec <- ugarchspec(variance.model = list(garchOrder=c(1,1)),
                       mean.model = list(armaOrder=c(1,0),include.mean=F))
GAR3spec <- ugarchspec(variance.model = list(garchOrder=c(1,1)),
                       mean.model = list(armaOrder=c(0,1),include.mean=F))
GAR4spec <- ugarchspec(variance.model = list(garchOrder=c(1,1)),
                       mean.model = list(armaOrder=c(1,1),include.mean=F))

EGAR1spec <- ugarchspec(distribution.model="std",
                        variance.model = list(model="eGARCH",garchOrder=c(1,1)),
                        mean.model = list(armaOrder=c(0,0),include.mean=F))
EGAR2spec <- ugarchspec(distribution.model="std",
                        variance.model = list(model="eGARCH",garchOrder=c(1,1)),
                        mean.model = list(armaOrder=c(1,0),include.mean=F))
EGAR3spec <- ugarchspec(distribution.model="std",
                        variance.model = list(model="eGARCH",garchOrder=c(1,1)),
                        mean.model = list(armaOrder=c(0,1),include.mean=F))
EGAR4spec <- ugarchspec(distribution.model="std",
                        variance.model = list(model="eGARCH",garchOrder=c(1,1)),
                        mean.model = list(armaOrder=c(1,1),include.mean=F))

GAR1 <- ugarchfit(data=d_ret, spec=GAR1spec)
```

```
GAR2 <- ugarchfit(data=d_ret, spec=GAR2spec)
GAR3 <- ugarchfit(data=d_ret, spec=GAR3spec)
GAR4 <- ugarchfit(data=d_ret, spec=GAR4spec)
EGAR1 <- ugarchfit(data=d_ret, spec=EGAR1spec)
EGAR2 <- ugarchfit(data=d_ret, spec=EGAR2spec)
EGAR3 <- ugarchfit(data=d_ret, spec=EGAR3spec)
EGAR4 <- ugarchfit(data=d_ret, spec=EGAR4spec)

#AIC and BIC functions
aicfun <- function(k,llh)(-2*llh+2*k)
bicfun <- function(k,llh)(-2*llh+log(length(SKR))*k)

#AIC and BIC calculation for all ARMA types
ARMAaic <- c(aicfun(3,GAR1@fit$LLH),aicfun(4,GAR2@fit$LLH),aicfun(4,GAR3@fit$LLH),
           aicfun(5,GAR4@fit$LLH),aicfun(5,EGAR1@fit$LLH),aicfun(6,EGAR2@fit$LLH),
           aicfun(6,EGAR3@fit$LLH),aicfun(7,EGAR4@fit$LLH))
ARMAbic <- c(bicfun(3,GAR1@fit$LLH),bicfun(4,GAR2@fit$LLH),bicfun(4,GAR3@fit$LLH),
           bicfun(5,GAR4@fit$LLH),bicfun(5,EGAR1@fit$LLH),bicfun(6,EGAR2@fit$LLH),
           bicfun(6,EGAR3@fit$LLH),bicfun(7,EGAR4@fit$LLH))

ARMAcomp <- data.frame(ARMAaic, ARMAbic)
rownames(ARMAcomp) <- c("ARMA(0,0)-GARCH(1,1)","ARMA(1,0)-GARCH(1,1)",
                        "ARMA(0,1)-GARCH(1,1)","ARMA(1,1)-GARCH(1,1)",
                        "ARMA(0,0)-EGARCH-t(1,1)","ARMA(1,0)-EGARCH-t(1,1)",
                        "ARMA(0,1)-EGARCH-t(1,1)","ARMA(1,1)-EGARCH-t(1,1)")
colnames(ARMAcomp)<-c("AIC", "BIC")


#Long-Memory behavior analysis of the models
#Simulation of EGARCH, 1&2 comp. Beta-Skew-t-EGARCH with "rugarch" and "betategarch"
#EGARCH:
garchsim.spec <- ugarchspec(variance.model = list(garchOrder=c(1,1),model="eGARCH"),
                            mean.model = list(armaOrder=c(0,0),include.mean=TRUE),
                            fixed.pars=G2E@fit$coef, distribution.model="std")

sim <- ugarchpath(garchsim.spec,n.sim=100000,rseed=1)
simret <- sim@path$seriesSim

#2-comp.:
set.seed(1)
sim2 <- tegarchSim(100000, omega=G52$par[1], phi1=G52$par[2],
                   phi2=G52$par[3], kappa1=G52$par[4], kappa2=G52$par[5],
                   kappastar=G52$par[6], df=G52$par[7], skew=G52$par[8])

#1-comp:
set.seed(1)
sim3<-tegarchSim(100000, omega=G5$par[1], phi1=G5$par[2], kappa1=G5$par[3],
                 kappastar=G5$par[4], df=G5$par[5], skew=G5$par[6])

#ACF plot of squared returns for our sample and several simulated ones
#Confidence intervals
conf3 <- (sqrt(length(simret)-(-10:210)))^(-1)*1.96
conf4 <- -conf3
```

```
#AIC and BIC comparison of Beta-Skew-t-EGARCH:
comp <- data.frame(LL=c(G5$objective,G52$objective),
                   AIC=c(AIC(G5),AIC(G52)), BIC=c(BIC(G5),BIC(G52)))


#Difference between GARCH and GARCH-t
proc.diff <- (GG1$sigma-GG2$sigma)/GG1$sigma*100
tab <- round(rbind(c(mean(abs(proc.diff)),NA),c(GG1$llhs,GG2$llhs)),digits=3)
row.names(tab) <- c("Mean absolute difference in %:",
                    "Gaussian Log-Likelihood vs. t:")


#AIC/BIC model evaluation
#Model (re-)calculation without mean or sigma_0 starting value
#using the packages "rugarch" and "betategarch" for most models
M1.spec <- ugarchspec(variance.model = list(garchOrder=c(1,1)),
                            mean.model = list(armaOrder=c(0,0),include.mean=F))
M1 <- ugarchfit(data=d_ret, spec=M1.spec)

M2.spec <- ugarchspec(distribution.model="std",
                      variance.model = list(garchOrder=c(1,1)),
                      mean.model = list(armaOrder=c(0,0),include.mean=F))
M2 <- ugarchfit(data=d_ret, spec=M2.spec)

M2E.spec <- ugarchspec(distribution.model="std",
                       variance.model = list(model="eGARCH",garchOrder=c(1,1)),
                       mean.model = list(armaOrder=c(0,0),include.mean=F))
M2E <- ugarchfit(data=d_ret, spec=M2E.spec)

M2GJR.spec <- ugarchspec(distribution.model="std",
                         variance.model = list(model="gjrGARCH",garchOrder=c(1,1)),
                         mean.model = list(armaOrder=c(0,0),include.mean=F))
M2GJR <- ugarchfit(data=d_ret, spec=M2GJR.spec)

#"rugarch" implementation contains one unnecessary parameter for TGARCH
# M2T.spec <- ugarchspec(distribution.model="std",
#                   variance.model = list(model="apARCH",
#                   submodel="TGARCH",garchOrder=c(1,1)),
#                   mean.model = list(armaOrder=c(0,0),include.mean=F))
# M2T<-ugarchfit(data=d_ret,spec=M2T.spec)
# M2T

#Own TGARCH without mean and sigma_0:
solfM2T <- function(vec)
{
  alpha_pl <- vec[1];
  alpha_mi <- vec[2];
  beta <- vec[3];
  omega <- vec[4];
  nu <- vec[5];
  LL_ret <- 1:(nSK-1);
  sig <- 1:(nSK);
  sig[1] <- sd(d_ret) ;
```

```
  for(i in 2:(nSK))
    (sig[i] <- omega+alpha_pl*(d_ret[i-1])*as.numeric(d_ret[i-1]>0)
         +alpha_mi*(d_ret[i-1])*as.numeric(d_ret[i-1]<=0)+beta*sig[i-1] );
  for(i in 1:(nSK-1))
    (LL_ret[i] <- log( gamma((nu+1)/2)/(sqrt((nu-2)*pi*sig[i]^2)*gamma(nu/2))*
                         (1+(d_ret[i])^2/((nu-2)*sig[i]^2))^(-(nu+1)/2) ) ) ;
  return(sum(-LL_ret)) ; #negative LL
}

#MLE:
solM2T <- solnp(pars=c(alpha_pl=0.1, alpha_mi=-0.1, beta=0.9, omega=0, nu=5),
            fun=solfM2T, LB=c(0,-0.3,0.7,0,2.001), UB=c(0.3,0,1,1,40))

LLM2T <- function(vec)
{
  alpha_pl <- vec[1];
  alpha_mi <- vec[2];
  beta <- vec[3];
  omega <- vec[4];
  nu <- vec[5];
  LL_ret <- 1:(nSK-1);
  sig <- 1:(nSK);
  sig[1] <- sd(d_ret);
  for(i in 2:(nSK))
    (sig[i] <- omega+alpha_pl*(d_ret[i-1])*as.numeric(d_ret[i-1]>0)
        +alpha_mi*(d_ret[i-1])*as.numeric(d_ret[i-1]<=0)+beta*sig[i-1] ) ;
  for(i in 1:(nSK-1))
    (LL_ret[i] <- log( gamma((nu+1)/2)/(sqrt((nu-2)*pi*sig[i]^2)*gamma(nu/2))*
                         (1+(d_ret[i])^2/((nu-2)*sig[i]^2))^(-(nu+1)/2) ) ) ;
  return(list(LL_ret, sig)) ;
}

#Summary:
M2T<-list(sigma=LLM2T(solM2T$pars)[[2]], llh=LLM2T(solM2T$pars)[[1]],
          llhs=sum(LLM2T(solM2T$pars)[[1]]), coef=solM2T$pars)

#Own GAS without mean and sigma_0:
solgas <- function(vec)
{
  alpha <- vec[1];
  beta <- vec[2];
  omega <- vec[3];
  nu <- vec[4];
  LL_ret <- 1:(nSK-1);
  sigs <- 1:(nSK);
  sigs[1] <- sd(d_ret)^2;
  for(i in 2:(nSK))
    (sigs[i] <- (omega+alpha*(nu+3)/nu*( ((nu+1)*(d_ret[i-1])^2*sigs[i-1])/
          ((nu-2)*sigs[i-1]+(d_ret[i-1])^2) - sigs[i-1] )+beta*sigs[i-1] )) ;
  for(i in 1:(nSK-1))
    (LL_ret[i] <- log( gamma((nu+1)/2)/(sqrt((nu-2)*pi*sigs[i])*gamma(nu/2))*
                  (1+(d_ret[i])^2/((nu-2)*sigs[i]))^(-(nu+1)/2) ) ) ;
  return(sum(-LL_ret)) ; #negative LL
```

```
}

#MLE:
optgas<-solnp(pars=c(alpha=0.1, beta=0.9, omega=0,nu=5),
            fun=solgas, LB=c(0,0.7,0,2.001), UB=c(0.3,1,1,20))

gasfun <- function(vec)
{
  alpha <- vec[1];
  beta <- vec[2];
  omega <- vec[3];
  nu <- vec[4];
  LL_ret <- 1:(nSK-1);
  sigs <- 1:(nSK);
  sigs[1] <- sd(d_ret)^2;
  for(i in 2:(nSK))
    (sigs[i] <- (omega+alpha*(nu+3)/nu*( ((nu+1)*(d_ret[i-1])^2*sigs[i-1])/
          ((nu-2)*sigs[i-1]+(d_ret[i-1])^2) - sigs[i-1] )+beta*sigs[i-1] )) ;
  for(i in 1:(nSK-1))
    (LL_ret[i] <- log( gamma((nu+1)/2)/(sqrt((nu-2)*pi*sigs[i])*gamma(nu/2))*
                  (1+(d_ret[i])^2/((nu-2)*sigs[i]))^(-(nu+1)/2) ) ) ;
  return(list(LL_ret, sigs)) ;
}

#Calculating LLH and sigma^2 at the MLE
optgas2 <- gasfun(optgas$pars)

#Summary:
M3 <- list(sigma=sqrt(optgas2[[2]]), llh=optgas2[[1]],
           llhs=sum(optgas2[[1]]), coef=optgas$pars)

#Proceeding with Beta-t-EGARCH models
M4 <- tegarch(d_ret, skew=FALSE) #score-driven with leverage

M42 <- tegarch(d_ret, components = 2, skew=FALSE,
        initial.values = c(0.02,0.95,0.9,0.01,0.01,0.005,5))

M5 <- G5

M52 <- G52

#Vector with number of parameters of each model
n.pars <- c(length(M1@fit$coef), length(M2@fit$coef), length(M3$coef),
          length(M2E@fit$coef), length(M2GJR@fit$coef), length(M2T$coef),
          length(M4$par), length(M5$par), length(M42$par), length(M52$par))

#Vector with all log-likelihoods
summLLH <- c(M1@fit$LLH, M2@fit$LLH, M3$llhs, M2E@fit$LLH,
             M2GJR@fit$LLH, M2T$llhs, M4$objective, M5$objective,
             M42$objective, M52$objective)

#All AIC and BIC values
summAIC <- rep(0,10)
```

```
for(i in 1:10)
  (summAIC[i] <- aicfun(n.pars[i], summLLH[i]))
summBIC <- rep(0,10)
for(i in 1:10)
  (summBIC[i] <- bicfun(n.pars[i], summLLH[i]))

#Summary
summ <- data.frame(n.pars, summLLH, summAIC, summBIC)
colnames(summ) <- c("k", "Log Likelihood", "AIC", "BIC")
rownames(summ) <- c("GARCH(1,1)", "GARCH-t(1,1)", "t-GAS(1,1)",
                    "EGARCH-t(1,1)", "GJRGARCH-t(1,1)", "TGARCH-t(1,1)",
                    "1-comp. Beta-t-EGARCH", "1-comp. Beta-Skew-t-EGARCH",
                    "2-comp.Beta-t-EGARCH", "2-comp. Beta-Skew-t-EGARCH")


#Checking influence of skew parameter
tab2<-round(rbind(c(mean(abs(fitted(M4)-fitted(M5))/fitted(M4))*100,NA),
                  c(M4$objective,M5$objective)), digits=3)
rownames(tab2)<-c("Mean absolute difference in %:",
                  "Log-Likelihood without/with Skew:")


#Relative impact plots


#Forecasting quantiles
set.seed(1)
F1 <- as.data.frame(ugarchboot(G1,n.ahead=22,method="Partial",n.bootpred=10000))
Q1 <- matrix(0,23,5)
Q1[1,] <- rep(G1@fit$sigma[1779],5)
for(i in 1:22)
  (Q1[i+1,] <- as.numeric(quantile(F1[[i]],c(0.025,0.25,0.5,0.75,0.975))))
P1 <- rep(0,23)
P1[1] <- G1@fit$sigma[1779]
for(i in 1:22) (P1[i+1] <- mean(F1[[i]]))

set.seed(1)
F2 <- as.data.frame(ugarchboot(G2,n.ahead=22,method="Partial",n.bootpred=10000))
Q2 <- matrix(0,23,5)
Q2[1,] <- rep(G2@fit$sigma[1779],5)
for(i in 1:22)
  (Q2[i+1,] <- as.numeric(quantile(F2[[i]],c(0.025,0.25,0.5,0.75,0.975))))
P2 <- rep(0,23)
P2[1] <- G2@fit$sigma[1779]
for(i in 1:22) (P2[i+1] <- mean(F2[[i]]))

set.seed(1)
F3 <- as.data.frame(ugarchboot(G2E,n.ahead=22,method="Partial",n.bootpred=10000))
Q3 <- matrix(0,23,5)
Q3[1,] <- rep(G2E@fit$sigma[1779],5)
for(i in 1:22)
  (Q3[i+1,] <- as.numeric(quantile(F3[[i]],c(0.025,0.25,0.5,0.75,0.975))))
P3 <- rep(0,23)
```

```
P3[1] <- G2E@fit$sigma[1779]
for(i in 1:22) (P3[i+1] <- mean(F3[[i]]))

set.seed(1)
F4 <- as.data.frame(ugarchboot(G2GJR,n.ahead=22,method="Partial",n.bootpred=10000))
Q4 <- matrix(0,23,5)
Q4[1,] <- rep(G2GJR@fit$sigma[1779],5)
for(i in 1:22)
  (Q4[i+1,] <- as.numeric(quantile(F4[[i]],c(0.025,0.25,0.5,0.75,0.975))))
P4 <- rep(0,23)
P4[1] <- G2GJR@fit$sigma[1779]
for(i in 1:22) (P4[i+1] <- mean(F4[[i]]))

set.seed(1)
F5 <- as.data.frame(ugarchboot(G2T,n.ahead=22,method="Partial",n.bootpred=10000))
Q5 <- matrix(0,23,5)
Q5[1,] <- rep(G2T@fit$sigma[1779],5)
for(i in 1:22)
  (Q5[i+1,] <- as.numeric(quantile(F5[[i]],c(0.025,0.25,0.5,0.75,0.975))))
P5 <- rep(0,23)
P5[1] <- G2T@fit$sigma[1779]
for(i in 1:22) (P5[i+1] <- mean(F5[[i]]))

#t-GAS implemented with exponential time-varying parameter as Beta-t-EGARCH
GAS3 <- tegarch(d_ret, skew=FALSE, asym=FALSE)

#As it would be inefficient to sample 10.000 processes of length 22,
#we sample 22 vectors of length 10.000 and model the volatility simultaneously.
#This means, instead of doing this:

# F6<-matrix(0,10000,23)
# for(i in 1:10000){set.seed(i);
#   F6[i,] <- as.numeric(
#   coredata(tegarchSim(23,lambda.initial=c(as.numeric(
#      coredata(fitted(GAS3,verbose=TRUE))[1779,"lambda"]),
#      as.numeric(coredata(fitted(GAS3,verbose=TRUE))[1779,"lambdadagg"])),
#      omega=GAS3$par["omega"],phi1=GAS3$par["phi1"],
#      kappa1=GAS3$par["kappa1"],kappastar=0,
#      df=GAS3$par["df"],skew=1,verbose=TRUE))[,"stdev"]
#   )
# } #Note: doesn't account for first value that can be calculated deterministically

#We apply the following functions for one and two-component Beta-t-EGARCH models:
#1-comp:
predfun1 <- function(temod, n.ahe=23, n.sim=1000, verbose=FALSE){
  moddf <- temod$par["df"];
  omega <- temod$par["omega"];

  kappa <- temod$par["kappa1"];    #short-term
  phi <- temod$par["phi1"];
  if(!is.na(temod$par["kappa2"])) (stop("1-comp. model required."))

  if(is.na(temod$par["skew"])) (skewpar <- 1)
```

```
else(skewpar <- temod$par["skew"]);

if(is.na(temod$par["kappastar"])) (kappastar <- 0)
else(kappastar <- temod$par["kappastar"]);

sigeps <- sqrt(STvar(df=moddf, skew=skewpar));
mueps <- STmean(df=moddf, skew=skewpar);

initvars <- coredata(fitted(temod, verbose=TRUE));

simseri <- matrix(0, n.ahe, n.sim);
simseri[1,] <- rep(as.numeric(initvars[length(temod$y),"y"]), n.sim);

lamb <- matrix(0, n.ahe+1, n.sim);
lamb[1,] <- rep(as.numeric(initvars[length(temod$y),"lambda"]), n.sim);

lambdagg <- matrix(0, n.ahe+1, n.sim);
lambdagg[1,] <- rep(as.numeric(initvars[length(temod$y),"lambdadagg"]), n.sim);

scorseri <- matrix(0, n.ahe, n.sim);
scorseri[1,] <- (moddf+1)*(simseri[1,]^2+simseri[1,]*mueps*exp(lamb[1,]))/
  (
    moddf*exp(2*lamb[1,])*
      skewpar^(2*sign(simseri[1,]+mueps*exp(lamb[1,])))+
      (simseri[1,]+mueps*exp(lamb[1,]))^2
  )-1;

lambdagg[2,] <- phi*lambdagg[1,]+kappa*scorseri[1,]+
                kappastar*sign(-simseri[1,])*(scorseri[1,]+1);
lamb[2,] <- omega+lambdagg[2,];

if(n.ahe>1)(
  for(i in 2:n.ahe){
    simseri[i,] <- (rST(n.sim,df=moddf,skew=skewpar)-mueps)*exp(lamb[i,]);
    scorseri[i,] <- (moddf+1)*(simseri[i,]^2+simseri[i,]*mueps*exp(lamb[i,]))/
     (
       moddf*exp(2*lamb[i,])*
       skewpar^(2*sign(simseri[i,]+mueps*exp(lamb[i,])))+
       (simseri[i,]+mueps*exp(lamb[i,]))^2
     )-1;
    lambdagg[i+1,] <- phi*lambdagg[i,]+kappa*scorseri[i,]+
                      kappastar*sign(-simseri[i,])*(scorseri[i,]+1);
    lamb[i+1,] <- omega+lambdagg[i+1,];
  }
)

stdev <- matrix(0, n.ahe+1, n.sim);
stdev <- exp(lamb[1:(n.ahe+1),])*sigeps;
stdev <- as.data.frame(t(stdev));
if(verbose==TRUE)
  (return(list(data.frame(sd.init=initvars[length(temod$y),"stdev"]),
               data.frame(sd=t(stdev)),data.frame(y=simseri),
               data.frame(u=scorseri),data.frame(lambda=lamb),
```

```
                data.frame(lambdadagg=lambdagg))))
  else
    (return(stdev))
}

#2-comp:
predfun2 <- function(temod, n.ahe=23, n.sim=1000, verbose=FALSE){
  moddf <- temod$par["df"];
  omega <- temod$par["omega"];

  if(is.na(temod$par["skew"])) (skewpar <- 1)
  else(skewpar <- temod$par["skew"]);

  if(is.na(temod$par["kappastar"])) (kappastar <- 0)
  else(kappastar <- temod$par["kappastar"]);

  if(is.na(temod$par["kappa2"])) (stop("2-comp. model required"));

  kappa2 <- temod$par["kappa2"]; phi2 <- temod$par["phi2"]; #short-term
  kappa1 <- temod$par["kappa1"]; phi1 <- temod$par["phi1"]; #long-term

  sigeps <- sqrt(STvar(df=moddf, skew=skewpar));
  mueps <- STmean(df=moddf, skew=skewpar);

  initvars <- coredata(fitted(temod, verbose=TRUE));

  simseri <- matrix(0, n.ahe, n.sim);
  simseri[1,] <- rep(as.numeric(initvars[length(temod$y),"y"]), n.sim);

  lamb <- matrix(0, n.ahe+1, n.sim);
  lamb[1,] <- rep(as.numeric(initvars[length(temod$y),"lambda"]), n.sim);

  lambdagg1 <- matrix(0, n.ahe+1, n.sim);
  lambdagg1[1,] <- rep(as.numeric(initvars[length(temod$y),"lambda1dagg"]),
                       n.sim);   #long-term

  lambdagg2 <- matrix(0, n.ahe+1, n.sim);
  lambdagg2[1,] <- rep(as.numeric(initvars[length(temod$y),"lambda2dagg"]),
                     n.sim);   #short-term

  scorseri <- matrix(0, n.ahe, n.sim);
  scorseri[1,] <- (moddf+1)*(simseri[1,]^2+simseri[1,]*mueps*exp(lamb[1,]))/
    (
      moddf*exp(2*lamb[1,])*
        skewpar^(2*sign(simseri[1,]+mueps*exp(lamb[1,])))+
        (simseri[1,]+mueps*exp(lamb[1,]))^2
    )-1;

  lambdagg2[2,] <- phi2*lambdagg2[1,]+kappa2*scorseri[1,]+
                   kappastar*sign(-simseri[1,])*(scorseri[1,]+1);
  lambdagg1[2,] <- phi1*lambdagg1[1,]+kappa1*scorseri[1,];
  lamb[2,] <- omega+lambdagg1[2,]+lambdagg2[2,];
```

```
  for(i in 2:n.ahe){
    simseri[i,] <- (rST(n.sim,df=moddf,skew=skewpar)-mueps)*exp(lamb[i,]);
    scorseri[i,] <- (moddf+1)*(simseri[i,]^2+simseri[i,]*mueps*exp(lamb[i,]))/
      (
        moddf*exp(2*lamb[i,])*
          skewpar^(2*sign(simseri[i,]+mueps*exp(lamb[i,])))+
          (simseri[i,]+mueps*exp(lamb[i,]))^2
      )-1;
    lambdagg2[i+1,] <- phi2*lambdagg2[i,]+kappa2*scorseri[i,]+
                       kappastar*sign(-simseri[i,])*(scorseri[i,]+1);
    lambdagg1[i+1,] <- phi1*lambdagg1[i,]+kappa1*scorseri[i,];
    lamb[i+1,] <- omega+lambdagg1[i+1,]+lambdagg2[i+1,];
  }

  stdev <- matrix(0, n.ahe+1, n.sim);
  stdev <- exp(lamb[1:(n.ahe+1),])*sigeps;
  stdev <- as.data.frame(t(stdev));

  if(verbose==TRUE)
    (return(list(data.frame(sd.init=initvars[length(temod$y),"stdev"]),
                 data.frame(sd=t(stdev)),data.frame(y=simseri),
                 data.frame(u=scorseri),data.frame(lambda=lamb),
                 data.frame(lambdadagger1=lambdagg1),
                 data.frame(lambdadagger2=lambdagg2))))
  else
    (return(stdev))
}

#Proceeding with forecasting for Beta-t-EGARCH models:
set.seed(1)
F6 <- predfun1(GAS3, n.ahe=22, n.sim=10000)
Q6 <- matrix(0,23,5)
for(i in 1:23)
  (Q6[i,] <- as.numeric(quantile(F6[[i]],c(0.025,0.25,0.5,0.75,0.975))))
P6 <- rep(0,23)
for(i in 1:23)(P6[i] <- mean(F6[[i]]))

set.seed(1)
F7 <- predfun1(M4, n.ahe=22, n.sim=10000)
Q7 <- matrix(0,23,5)
for(i in 1:23)
  (Q7[i,] <- as.numeric(quantile(F7[[i]],c(0.025,0.25,0.5,0.75,0.975))))
P7 <- rep(0,23)
for(i in 1:23)(P7[i] <- mean(F7[[i]]))

set.seed(1)
F8 <- predfun2(M42, n.ahe=22, n.sim=10000)
Q8 <- matrix(0,23,5)
for(i in 1:23)
  (Q8[i,] <- as.numeric(quantile(F8[[i]],c(0.025,0.25,0.5,0.75,0.975))))
P8 <- rep(0,23)
for(i in 1:23)(P8[i] <- mean(F8[[i]]))
```

```
set.seed(1)
F9 <- predfun1(M5, n.ahe=22, n.sim=10000)
Q9 <- matrix(0,23,5)
for(i in 1:23)
  (Q9[i,]<-as.numeric(quantile(F9[[i]],c(0.025,0.25,0.5,0.75,0.975))))
P9 <- rep(0,23)
for(i in 1:23)(P9[i] <- mean(F9[[i]]))

set.seed(1)
F10 <- predfun2(M52,n.ahe=22, n.sim=10000)
Q10 <- matrix(0,23,5)
for(i in 1:23)
  (Q10[i,]<-as.numeric(quantile(F10[[i]],c(0.025,0.25,0.5,0.75,0.975))))
P10 <- rep(0,23)
for(i in 1:23)(P10[i] <- mean(F10[[i]]))

set.seed(1)
F11 <- predict(SVMG, steps=22)
Q11 <- rbind(as.numeric(rep(medG[2,length(medG[2,])]),22),
             sqrt(exp(summary(F11)$quantiles)))
P11 <- rep(0,23)
P11[1] <- Q11[1,1]
for(i in 1:22)
  (P11[i+1] <- mean(sqrt(exp(F11[,i]))))

set.seed(1)
F12 <- predict(SVMT, steps=22)
Q12 <- rbind(as.numeric(rep(medT[2,length(medG[2,])]),22),
             sqrt(exp(summary(F12)$quantiles)))
P12 <- rep(0,23)
P12[1] <- Q12[1,1]
for(i in 1:22)
  (P12[i+1] <- mean(sqrt(exp(F12[,i]))))


#Forecasting Evaluation
d_ret2 <- rep(0,22) #out-of-sample demeaned log returns
d_ret2[1] <- log(DAX2$Schlusskurs[1])-log(DAX$Schlusskurs[1780])
d_ret2[2:22] <- diff(log(DAX2$Schlusskurs[1:22]),lag=1)

proxy<-(d_ret2-mean(d_ret2))^2 #variance proxy values

#Define variance forecasts by squared sd medians
FC1 <- Q1[-1,3]^2
FC2 <- Q2[-1,3]^2
FC3 <- Q3[-1,3]^2
FC4 <- Q4[-1,3]^2
FC5 <- Q5[-1,3]^2
FC6 <- Q6[-1,3]^2
FC7 <- Q7[-1,3]^2
FC8 <- Q8[-1,3]^2
FC9 <- Q9[-1,3]^2
FC10 <- Q10[-1,3]^2
```

```
FC11 <- Q11[-1,3]^2
FC12 <- Q12[-1,3]^2

#Function to calculate R^2
Rsqu <- function(fit, data=proxy){
  SS <- sum((data-mean(data))^2);
  RSS <- sum((data-fit)^2);
  out <- 1-RSS/SS;
  return(out)
}

#Function to calculate MSE, MAE, MSD and QLIKE
FCeval <- function(FC, true=proxy){
  MSE <- sum((FC-true)^2)/length(FC);
  MAE <- sum(abs(FC-true))/length(FC);
  MSD <- sum(FC-true)/length(FC);
  QLIKE <- sum(log(true)+FC/true)/length(FC);
  RSQ <- Rsqu(FC);
  return(c(MSE, MAE, MSD, QLIKE, RSQ))
}

FCsumm <- rbind(FCeval(FC1), FCeval(FC2), FCeval(FC3), FCeval(FC4),
                FCeval(FC5), FCeval(FC6), FCeval(FC7), FCeval(FC8),
                FCeval(FC9), FCeval(FC10), FCeval(FC11), FCeval(FC12))
FCsumm[,2:3] <- FCsumm[,2:3]*10^6
FCsumm[,1] <- FCsumm[,1]*10^10
rownames(FCsumm) <- c(
                    "GARCH(1,1)","GARCH-t(1,1)","EGARCH-t(1,1)",
                    "GJRGARCH-t(1,1)","TGARCH-t(1,1)",
                    "t-GAS/Beta-t-EGARCH","1-c. Beta-t-EGARCH",
                    "2-c. Beta-t-EGARCH", "1-c. Beta-Skew-t-EGARCH",
                    "2-c. Beta-Skew-t-EGARCH","SV","SV-t"
                    )
colnames(FCsumm) <- c("MSE x10^10", "MAE x10^6", "MSD x10^6", "QLIKE", "R^2")



##SECTION 5: Conclusion



#Appendix - Parameter estimates

#Estimates for non-stochastic volatility models with standard errors and
#p-values from t-tests:

#solnp hessian is off, probably because it somehow involves the used boundaries
#Thus, hessian has to be seperately approximated for own models, which can
#be tricky.
#Also: These hessians are from the negative LL, i.e. we can use it directly
#for the calculation, instead of considering the negative Hessian
```

```
all_hess <- list()
all_hess$a <- hessian(solfun, sol$pars,
        method.args=list(zero.tol=.Machine$double.eps/10^10))
#High precision necessary due to small significance values

all_hess$b <- hessian(solfun2, sol2$pars,
        method.args=list(zero.tol=.Machine$double.eps/10^10))

#NANs were produced with hessian function from numDeriv package,
#Hence, we use the "pracma" package for the EGARCH model.
library(pracma)
all_hess$c <- hessian(solfunEG, solEG$pars)
detach("package:pracma")

all_hess$d <- hessian(solfunGJR, solGJR$pars,
        method.args=list(zero.tol=.Machine$double.eps/10^10))
all_hess$e <- hessian(solfunT, solT$pars,
        method.args=list(zero.tol=.Machine$double.eps/10^10))
all_hess$f <- hessian(solfun3, sol3$pars,
        method.args=list(zero.tol=.Machine$double.eps/10^10))

#Function that returns the parameters, their t-test p-values and standard errors:
pars_to_signif <- function(pars, hess) #Input: Parameter estimates and hessian
{
  serr <- sqrt(abs(diag(solve(hess)))); #Calculate std. errors
  pval <- rep(0,length(pars));
  for(i in 1:length(pars))
    (pval[i] <- 2*(1-pnorm(abs(pars[i]/serr[i]))));
  signif <- list(rbind(pars,serr,pval));
  return(signif)
}

signif1 <- pars_to_signif(sol$pars, all_hess$a)    #GARCH(1,1)
signif2 <- pars_to_signif(sol2$pars, all_hess$b)   #GARCH-t(1,1)
signif3 <- pars_to_signif(solEG$pars, all_hess$c)  #EGARCH-t(1,1)
signif4 <- pars_to_signif(solGJR$pars, all_hess$d) #GJR-GARCH-t(1,1)
signif5 <- pars_to_signif(solT$pars, all_hess$e)   #T-GARCH-t(1,1)
signif6 <- pars_to_signif(sol3$pars, all_hess$f)   #t-GAS(1,1)

#signif1 #no non-significant parameters at 0.05 sign. level
#signif2 #sigma_0 insignificant at 0.05 sign. level
#signif3 #sigma_0 insignificant
#signif4 #alpha highly insignificant, i.e. only negative returns can increase
        #volatility, positive ones decrease it; sigma_0 insignificant
#signif5 #sigma_0 and alpha_plus insignificant - again only negative returns with impact;
#signif6 #all significant

signif7 <- pars_to_signif(G5$par, -G5$hessian)
signif8 <- pars_to_signif(G52$par, -G52$hessian)


#signif7 #all significant
#signif8 #kappa_2 in short-memory component insignificant
```

```
         #(most power coming from kappastar, aka leverage coefficient)

#MCMC parameter traces of SV models and the respective density plots
```

Output Code

```
##SECTION 1: Introduction




##SECTION 2: The Models

#DAX from 2010 to 2016
plot(DATUM, SK, type="l", xlab="Date", ylab="DAX Closing price")


#Range and mean of DAX 2010 to 2016
summary(SK)


#ADF test
latex(ADFtab, file="", table.env=FALSE, first.hline.double=FALSE, colheads=FALSE)


#DAX returns from 2010 to 2016
plot(DATUM[-1],SKR, type="l", xlab="Date", ylab="DAX returns")


#ARCH LM testing of DAX returns
print(xtable(ARCHLM,
      caption="ARCH LM Test on DAX returns",
      label="table:ARCHLM",
      align="lccc",
      digits=4
),
include.rownames=FALSE,
caption.placement="bottom"
)


#Ljung-Box testing of squared, demeaned DAX returns
print(xtable(LB,
             caption="Ljung-Box Test on DAX returns",
             label="table:LB",
             align="lccc",
             digits=4
),
include.rownames=FALSE,
caption.placement="bottom"
)
```

```
#Kurtosis and Skewness
latex(c(
          "sample kurtosis:", round(SKRkurt,4),
          "sample skewness:", round(SKRskew,4)
          ),
    file="", table.env=FALSE, first.hline.double=FALSE,
    colheads=FALSE
        )


#Q-Q-Plots
par(mfrow=c(1,2))
qqnorm(SKR); qqline(SKR,col="red")
plot(qt(pts, df=fdest_df), quants, xlab="Theoretical Quantiles",
     ylab="Sample Quantiles", main="Student's t Q-Q Plot")
qqline(quants, distribution=function(t)qt(t, df=fdest_df), col="red")
par(mfrow=c(1,1))


#Approximating density histogram
hist(SKR, freq=F, breaks=50, main="DAX return density",
     xlab="log-return", xlim=c(-0.08,0.08),
     xaxp=c(-0.075,0.075,6))
curve(f.den, xlim=c(-0.08,0.08), add=TRUE, col="green")
curve(f.den2, xlim=c(-0.08,0.08), col="red", add=TRUE)
lines(density(SKR, bw=(3/4*length(SKR))^(-1/5)*sd(SKR)), col="blue")
legend("topleft", col=c("green", "red", "blue"), lty=1,
       legend=c("normal", "student's t", "kernel"),
       cex=1.5)


#Effect of single returns on overall sample standard deviation
plot(DATUM[-1], hatw, xlab="Date",
     ylab="Relative influence on sample standard deviation in %",
     main="Influence of single returns on overall standard deviation",
     ylim=c(-0.05,0.9))
lines(DATUM[-length(SKR)], MA4(20,hatw), col="red", lwd=3)

print(xtable(dhat2,
             caption="Returns ranked, with respect to influence
               on sample standard deviation",
             label="table:asdf",
             align="lccccc",
             digits=3
),
include.rownames=FALSE,
caption.placement="bottom"
)


#Checking leverage effect
latex(levcor, file="", table.env=FALSE, first.hline.double=FALSE, colheads=FALSE)
```

```
plot(MA2(254,SKR)[-(1:253)], type="l", col="blue", axes=FALSE, xlab="Date",
     ylab="means(blue) and standard deviations(red)", main="Comparison of local
       mean and standard deviation")
axis(side=1, col="black", at=c(-10^4,2000))
axis(side=1, col="black", at=seq(1,1526,254), labels=c("2011","2012","2013",
                                                       "2014","2015","2016","2017"))
axis(side=2, col="blue", at=c(-10^4,2000))
axis(side=2, col="blue")
axis(side=3, col="black", at=c(-10^4,2000))
axis(side=4, col="red", at=c(-10^4,2000))
par(new=TRUE)
plot(MA1(254,SKR)[-(1:253)], col="red", type="l", ann=FALSE, axes=FALSE)
axis(side=4, col="red")



##SECTION 3: The Models

#Model 0: "local standard deviation"
plot(MA1(20,SKR), type="l", axes=FALSE, xlab="Date", ylab="Standard deviation",
     main="Local standard deviation series")
box(); axis(side=2);
axis(side=1, at=seq(1,1780,254),
     labels=c("2010","2011","2012","2013","2014","2015","2016","2017"))
lines(MA1(50,SKR), col="red")
lines(MA1(100,SKR), col="blue")
lines(MA1(200,SKR), col="darkgray")


#Model 1: "GARCH(1,1)"
plot(G1@fit$sigma,type="l",ylab="Standard deviation",xlab="Date",
     axes=FALSE,main="Standard deviation Estimates with GARCH(1,1)")
box();axis(side=2)
axis(side=1,at=seq(1,1780,254),
     labels=c("2010","2011","2012","2013","2014","2015","2016","2017"))
lines(GG1$sigma,col="darkgray")


#Model 2: "GARCH-t(1,1)"
plot(GG2$sigma[254:762],type="l",ylab="Standard deviation",xlab="Date",
     axes=FALSE,main="Standard deviation Estimates of GARCH-t(1,1) vs. GARCH(1,1)")
box();axis(side=2)
axis(side=1,at=seq(1,509,254),
     labels=c("2011","2012","2013"))
lines(GG1$sigma[254:762],col="darkgray")
legend("topleft",col=c("darkgray","black"),lty=1,
       legend=c("GARCH(1,1)","GARCH-t(1,1)"),
       cex=1.35)


#Model 2E: "EGARCH-t(1,1)"
plot(GGEG$sigma,type="l",ylab="Standard deviation",xlab="Date",
```

```
     axes=FALSE,main="Standard deviation Estimates of EGARCH(1,1) vs. GARCH-t(1,1)")
box();axis(side=2)
axis(side=1,at=seq(1,1780,254),
     labels=c("2010","2011","2012","2013","2014","2015","2016","2017"))
lines(GG2$sigma,col="orange1")
legend("topleft",col=c("orange1","black"),lty=1,
       legend=c("GARCH-t(1,1)","EGARCH(1,1)"),
       cex=1.35)


#Model 2GJR: "GJRGARCH-t(1,1)"
plot(GGGJR$sigma, type="l", ylab="Standard deviation", xlab="Date",
     axes=FALSE, main="Standard deviation Estimates of GJR-GARCH(1,1) vs. EGARCH(1,1)")
box(); axis(side=2)
axis(side=1, at=seq(1,1780,254),
     labels=c("2010","2011","2012","2013","2014","2015","2016","2017"))
lines(GGEG$sigma, col="orange1")
legend("topleft", col=c("orange1","black"), lty=1,
       legend=c("EGARCH(1,1)","GJR-GARCH(1,1)"),
       cex=1.2)


#Model 2T: "TGARCH-t(1,1)"
plot(GGT$sigma, type="l", ylab="Standard deviation", xlab="Date", axes=FALSE,
     main="Standard deviation Estimates of TGARCH(1,1) vs. GJR-GARCH(1,1)")
box(); axis(side=2)
axis(side=1, at=seq(1,1780,254),
     labels=c("2010","2011","2012","2013","2014","2015","2016","2017"))
lines(GGGJR$sigma, col="orange1")
legend("topleft", col=c("orange1","black"), lty=1,
       legend=c("GJR-GARCH(1,1)","TGARCH(1,1)"),
       cex=1.2)


#Model 3: "t-GAS(1,1)"
<<fig=TRUE, width=12, heigth=6>>=
  plot(GG2$sigma,type="l",ylab="Standard deviation",xlab="Date",
         axes=FALSE,main="SD Estimates of GARCH-t(1,1) vs. t-GAS(1,1) (scaled & unscaled)",
       ylim=c(0.006,0.035))
  box();axis(side=2)
  axis(side=1,at=seq(1,1780,254),
       labels=c("2010","2011","2012","2013","2014","2015","2016","2017"))
  lines(GG4$sigma,col="orange1")
  lines(sqrt(getFilteredParameters(GAS2)[,2]*(getFilteredParameters(GAS2)[1,3]
             /(getFilteredParameters(GAS2)[1,3]-2))), col="mediumorchid")
  legend("topleft",col=c("black","orange1","mediumorchid"),lty=1,
         legend=c("GARCH-t(1,1)","t-GAS(1,1) scaled","t-GAS(1,1) unscaled"),
         cex=1)
  #Note: GAS package uses "non-corrected" t-distribution with variance unequal the
  #squared scale. Thus, the factor (nu/(n-2)) needs to be added, before taking root of
  #the variance.
@
```

```
#Model 4: "1/2 component Beta-Skew-t-EGARCH"
plot(GGEG$sigma[1272:1779], type="l", ylab="Standard deviation", xlab="Date",
     axes=FALSE, main="SD Estimates of GARCH-t(1,1) vs. 1&2-comp. Beta-skew-t-EGARCH",
     ylim=c(0.003,0.036))
box(); axis(side=2)
axis(side=1, at=seq(1,509,254),
     labels=c("2015","2016","2017"))
lines(as.numeric(fitted(G5))[1272:1779], col="orange1")
lines(as.numeric(fitted(G52))[1272:1779], col="mediumorchid")
legend("topleft", col=c("black","orange1","mediumorchid"), lty=1,
       legend=c("EGARCH(1,1)","1-comp. Beta-skew-t-EGARCH",
                "2-comp. Beta-skew-t-EGARCH"),
       cex=0.8)


#Model 5: "SV and SV-t"
plot(medG[2,1272:1779], type="l", ylab="Standard deviation", xlab="Date",
     ylim=c(0,0.035), axes=FALSE, main="SD quantiles of SV vs. SV-t")
for(i in c(1,3))
  (lines(medG[i,1272:1779], col="darkgray"))
box(); axis(side=2)
axis(side=1, at=seq(1,509,254),
     labels=c("2015","2016","2017"))
lines(medT[2,1272:1779], col="orange4")
for(i in c(1,3))
  (lines(medT[i,1272:1779], col="orange1"))
legend("topleft", col=c("black","darkgray","orange4","orange1"), lty=1,
       legend=c("SV median","SV 5/95% quantiles",
                "SV-t median", "SV-t 5/95% quantiles"),
       cex=1)
par(new=TRUE)
plot(abs(SKR)[1272:1779], type="h", ylim=c(0,0.3), axes=FALSE, ann=FALSE)

plot(medG[2,], type="l", ylab="Standard deviation", xlab="Date", ylim=c(0,0.038),
     axes=FALSE, main="SD quantiles of SV vs. GARCH-t(1,1) vs. t-GAS(1,1)")
for(i in c(1,3))
  (lines(medG[i,], col="gray"))
box(); axis(side=2)
axis(side=1, at=seq(1,1780,254),
     labels=c("2010","2011","2012","2013","2014","2015","2016","2017"))
lines(GG2$sigma[-1], col="orange1") #SV and SV-t don't involve starting values
lines(GG4$sigma[-1], col="mediumorchid")
legend("topleft", col=c("black","gray","orange1","mediumorchid"), lty=1,
       legend=c("SV-t median", "SV-t 5/95% quantiles", "GARCH-t(1,1)",
                "t-GAS(1,1) scaled"),
       cex=0.6)
par(new=TRUE)
plot(abs(SKR), type="h", ylim=c(0,0.4), axes=FALSE, ann=FALSE)


#Overview:
```

```
##SECTION 4: Model Performance Analysis

#Testing existence of ARCH in standardized residuals after modelling:
#Ljung-Box test:
print(xtable(Port,
             caption="$p$-values of Ljung-Box test on the squared
             standardized residuals.",
             label="table:Port",
             align="lcccc",
             digits=4
),
include.rownames=TRUE,
caption.placement="bottom"
)

#ARCH-LM test:
print(xtable(Scor,
             caption="$p$-values of ARCH LM test on the squared
             standardized residuals.",
             label="table:Scor",
             align="lcccc",
             digits=4
),
include.rownames=TRUE,
caption.placement="bottom"
)

#Autocorrelation function plots of squared standardized residuals
par(mfrow=c(3,3))
Acf(SKRssres[[1]], lag.max=200, main="GARCH", ci=FALSE)
lines(-10:210, conf1, lty=2, col="blue")
lines(-10:210, conf2, lty=2, col="blue")
Acf(SKRssres[[2]], lag.max=200, main="GARCH-t", ci=FALSE)
lines(-10:210,conf1, lty=2, col="blue")
lines(-10:210,conf2, lty=2, col="blue")
Acf(SKRssres[[3]], lag.max=200, main="EGARCH-t", ci=FALSE)
lines(-10:210,conf1, lty=2, col="blue")
lines(-10:210,conf2, lty=2, col="blue")
Acf(SKRssres[[4]], lag.max=200, main="GJRGARCH-t", ci=FALSE)
lines(-10:210,conf1, lty=2, col="blue")
lines(-10:210,conf2, lty=2, col="blue")
Acf(SKR^2, lag.max=200, main="Squared Returns", ci=FALSE)
lines(-10:210,conf1, lty=2, col="blue")
lines(-10:210,conf2, lty=2, col="blue")
Acf(SKRssres[[5]], lag.max=200, main="TGARCH-t", ci=FALSE)
lines(-10:210,conf1, lty=2, col="blue")
lines(-10:210,conf2, lty=2, col="blue")
Acf(SKRssres[[6]], lag.max=200, main="t-GAS", ci=FALSE)
lines(-10:210,conf1, lty=2, col="blue")
lines(-10:210,conf2, lty=2, col="blue")
Acf(SKRssres[[7]], lag.max=200, main="1-c. Beta-t-EGARCH", ci=FALSE)
lines(-10:210,conf1, lty=2, col="blue")
```

```
lines(-10:210,conf2, lty=2, col="blue")
Acf(SKRssres[[8]], lag.max=200, main="2-c. Beta-t-EGARCH", ci=FALSE)
lines(-10:210,conf1, lty=2, col="blue")
lines(-10:210,conf2, lty=2, col="blue")
par(mfrow=c(1,1))


#Testing, whether constant mean model is appropriate
#Ljung-box on standardized residuals:
print(xtable(Port2,
             caption="$p$-values of Ljung-Box test on the standardized residuals.",
             label="table:Port2",
             align="lcccc",
             digits=4
),
include.rownames=TRUE,
caption.placement="bottom"
)

#AIC and BIC comparison for GARCH and EGARCH with several ARMA orders
print(xtable(ARMAcomp,
             caption="AIC and BIC comparison of GARCH and EGARCH-t
                for different ARMA orders.",
             label="table:ARMAcomp",
             align="lcc",
             digits=3
),
include.rownames=TRUE,
caption.placement="bottom"
)


#Long-Memory behavior analysis of the models
#ACF plot of squared returns for our sample and several simulated ones
Acf(simret^2, lag.max=200, col="darkgray", ci=FALSE)
panel.smooth(1:200, Acf(simret^2,plot=FALSE,lag.max=200)$acf[-1],
             col.smooth="orange1", pch=" ", span=0.1, lwd=2)
panel.smooth(1:200, Acf(sim2^2,plot=FALSE,lag.max=200)$acf[-1],
             col.smooth="red", pch=" ", span=0.1, lwd=2)
panel.smooth(1:200, Acf(sim3^2,plot=FALSE,lag.max=200)$acf[-1],
             col.smooth="blue", pch=" ", span=0.1, lwd=2)
panel.smooth(1:200, Acf(SKR^2,plot=FALSE,lag.max=200)$acf[-1],
             col.smooth="mediumorchid", pch=" ", span=0.1, lwd=2)
lines(-10:210, conf1, lty=2, col="mediumorchid")
lines(-10:210, conf2, lty=2, col="mediumorchid")
lines(-10:210, conf3, lty=4, col="black")
lines(-10:210, conf4, lty=4, col="black")
legend("topright", lty=1, lwd=2,
       col=c("darkgray","orange1","blue","red","mediumorchid"),
       legend=c("EGARCH", "EGARCH", "1-comp. Beta-t-EGARCH",
                "2-comp. Beta-t-EGARCH","Squared Returns"),
       cex=1.35)
```

```
#AIC and BIC comparison of Beta-Skew-t-EGARCH:
print(xtable(comp,
             caption="AIC and BIC comparison of 1-component versus 2-component
             Beta-Skew-t-EGARCH.",
             label="table:comp",
             align="lccc",
             digits=4
),
include.rownames=TRUE,
caption.placement="bottom"
)


#Difference between GARCH and GARCH-t
latex(tab, file="", table.env=FALSE, first.hline.double=FALSE,
      colheads=FALSE, col.just = c("c","c"), vbar=FALSE)


#AIC/BIC model evaluation
print(xtable(summ,
             caption="AIC and BIC comparison of all models, where $k$
                      denotes the number of parameters.
                      In these calculations, the volatility starting
                      value and mean were omitted.",
             label="table:summ",
             align="lcccc",
             digits=3
),
include.rownames=TRUE,
caption.placement="bottom"
)


#Checking influence of skew parameter
latex(tab2, file="", table.env=FALSE, first.hline.double=FALSE,
      colheads=FALSE, col.just = c("c","c"), vbar=FALSE)


#Relative impact plots
par(mfrow=c(2,4))
plot(SKR, GG1$sigma[-1]-GG1$sigma[-1780], ylim=c(-0.006,0.024), main="GARCH",
     xlab="log returns", ylab="Impact")
plot(SKR, GG2$sigma[-1]-GG2$sigma[-1780], ylim=c(-0.006,0.024), main="GARCH-t",
     xlab="log returns", ylab="Impact")
plot(SKR, GGEG$sigma[-1]-GGEG$sigma[-1780], ylim=c(-0.006,0.024), main="EGARCH-t",
     xlab="log returns", ylab="Impact")
plot(SKR, GGGJR$sigma[-1]-GGGJR$sigma[-1780], ylim=c(-0.006,0.024), main="GJRGARCH-t",
     xlab="log returns", ylab="Impact")
plot(SKR, GGT$sigma[-1]-GGT$sigma[-1780], ylim=c(-0.006,0.024), main="TGARCH-t",
     xlab="log returns", ylab="Impact")
plot(SKR, GG4$sigma[-1]-GG4$sigma[-1780], ylim=c(-0.006,0.024), main="t-GAS",
     xlab="log returns", ylab="Impact")
plot(SKR[-1779], as.numeric(fitted(G5))[-1]-as.numeric(fitted(G5))[-1779],
```

```
     ylim=c(-0.006,0.024), main="1-c. Beta-Skew-t-EGARCH",
     xlab="log returns", ylab="Impact")
plot(SKR[-1779], as.numeric(fitted(G52))[-1]-as.numeric(fitted(G52))[-1779],
     ylim=c(-0.006,0.024), main="2-c. Beta-Skew-t-EGARCH",
     xlab="log returns", ylab="Impact")
par(mfrow=c(1,1))


#Forecasting quantiles
par(mfrow=c(2,3))
plot(Q1[,3], ylim=c(0.003,0.02), axes=FALSE, xlab="Date",
     ylab="Forecasted SD Quantiles", main="GARCH")
box(); axis(side=2)
axis(side=1, at=c(1,12,23), labels=c("Dez 30","Jan 17","Jan 31"))
axis(side=1, at=c(1,6.5,12,17.5,23), labels=FALSE)
lines(Q1[,1], lty=3)
lines(Q1[,2], lty=2)
lines(Q1[,3], col="red")
lines(Q1[,4], lty=2)
lines(Q1[,5], lty=3)
lines(P1, col="purple")

plot(Q2[,3], ylim=c(0.003,0.02), axes=FALSE, xlab="Date",
     ylab="Forecasted SD Quantiles", main="GARCH-t")
box(); axis(side=2)
axis(side=1, at=c(1,12,23), labels=c("Dez 30","Jan 17","Jan 31"))
axis(side=1, at=c(1,6.5,12,17.5,23), labels=FALSE)
lines(Q2[,1], lty=3)
lines(Q2[,2], lty=2)
lines(Q2[,3], col="red")
lines(Q2[,4], lty=2)
lines(Q2[,5], lty=3)
lines(P2, col="purple")

plot(Q3[,3], ylim=c(0.003,0.02), axes=FALSE, xlab="Date",
     ylab="Forecasted SD Quantiles", main="EGARCH-t")
box(); axis(side=2)
axis(side=1, at=c(1,12,23), labels=c("Dez 30","Jan 17","Jan 31"))
axis(side=1, at=c(1,6.5,12,17.5,23), labels=FALSE)
lines(Q3[,1], lty=3)
lines(Q3[,2], lty=2)
lines(Q3[,3], col="red")
lines(Q3[,4], lty=2)
lines(Q3[,5], lty=3)
lines(P3, col="purple")

plot(Q4[,3], ylim=c(0.003,0.02), axes=FALSE, xlab="Date",
     ylab="Forecasted SD Quantiles", main="GJRGARCH-t")
box(); axis(side=2)
axis(side=1, at=c(1,12,23), labels=c("Dez 30","Jan 17","Jan 31"))
axis(side=1, at=c(1,6.5,12,17.5,23), labels=FALSE)
lines(Q4[,1], lty=3)
lines(Q4[,2], lty=2)
```

```
lines(Q4[,3], col="red")
lines(Q4[,4], lty=2)
lines(Q4[,5], lty=3)
lines(P4, col="purple")

plot(Q5[,3], ylim=c(0.003,0.02), axes=FALSE, xlab="Date",
     ylab="Forecasted SD Quantiles", main="TGARCH-t")
box(); axis(side=2)
axis(side=1, at=c(1,12,23), labels=c("Dez 30","Jan 17","Jan 31"))
axis(side=1, at=c(1,6.5,12,17.5,23), labels=FALSE)
lines(Q5[,1], lty=3)
lines(Q5[,2], lty=2)
lines(Q5[,3], col="red")
lines(Q5[,4], lty=2)
lines(Q5[,5], lty=3)
lines(P5, col="purple")

plot(Q6[,3], ylim=c(0.003,0.02), axes=FALSE, xlab="Date",
     ylab="Forecasted SD Quantiles", main="t-GAS")
box(); axis(side=2)
axis(side=1, at=c(1,12,23), labels=c("Dez 30","Jan 17","Jan 31"))
axis(side=1, at=c(1,6.5,12,17.5,23), labels=FALSE)
lines(Q6[,1], lty=3)
lines(Q6[,2], lty=2)
lines(Q6[,3], col="red")
lines(Q6[,4], lty=2)
lines(Q6[,5], lty=3)
lines(P6, col="purple")

par(mfrow=c(2,3))
plot(Q7[,3], ylim=c(0.003,0.02), axes=FALSE, xlab="Date",
     ylab="Forecasted SD Quantiles", main="1-c. Beta-t-EGARCH")
box(); axis(side=2)
axis(side=1, at=c(1,12,23), labels=c("Dez 30","Jan 17","Jan 31"))
axis(side=1, at=c(1,6.5,12,17.5,23), labels=FALSE)
lines(Q7[,1], lty=3)
lines(Q7[,2], lty=2)
lines(Q7[,3], col="red")
lines(Q7[,4], lty=2)
lines(Q7[,5], lty=3)
lines(P7, col="purple")

plot(Q8[,3], ylim=c(0.003,0.02), axes=FALSE, xlab="Date",
     ylab="Forecasted SD Quantiles", main="2-c. Beta-t-EGARCH")
box(); axis(side=2)
axis(side=1, at=c(1,12,23), labels=c("Dez 30","Jan 17","Jan 31"))
axis(side=1, at=c(1,6.5,12,17.5,23), labels=FALSE)
lines(Q8[,1], lty=3)
lines(Q8[,2], lty=2)
lines(Q8[,3], col="red")
lines(Q8[,4], lty=2)
lines(Q8[,5], lty=3)
lines(P8, col="purple")
```

```
plot(Q9[,3], ylim=c(0.003,0.02), axes=FALSE, xlab="Date",
     ylab="Forecasted SD Quantiles", main="1-c. Beta-Skew-t-EGARCH")
box(); axis(side=2)
axis(side=1, at=c(1,12,23), labels=c("Dez 30","Jan 17","Jan 31"))
axis(side=1, at=c(1,6.5,12,17.5,23), labels=FALSE)
lines(Q9[,1], lty=3)
lines(Q9[,2], lty=2)
lines(Q9[,3], col="red")
lines(Q9[,4], lty=2)
lines(Q9[,5], lty=3)
lines(P9, col="purple")

plot(Q10[,3], ylim=c(0.003,0.02), axes=FALSE, xlab="Date",
     ylab="Forecasted SD Quantiles", main="2-c. Beta-Skew-t-EGARCH")
box(); axis(side=2)
axis(side=1, at=c(1,12,23), labels=c("Dez 30","Jan 17","Jan 31"))
axis(side=1, at=c(1,6.5,12,17.5,23), labels=FALSE)
lines(Q10[,1], lty=3)
lines(Q10[,2], lty=2)
lines(Q10[,3], col="red")
lines(Q10[,4], lty=2)
lines(Q10[,5], lty=3)
lines(P10, col="purple")

plot(Q11[,3], ylim=c(0.003,0.02), axes=FALSE, xlab="Date",
     ylab="Forecasted SD Quantiles", main="SV")
box(); axis(side=2)
axis(side=1, at=c(1,12,23), labels=c("Dez 30","Jan 17","Jan 31"))
axis(side=1, at=c(1,6.5,12,17.5,23), labels=FALSE)
lines(Q11[,1], lty=3)
lines(Q11[,2], lty=2)
lines(Q11[,3], col="red")
lines(Q11[,4], lty=2)
lines(Q11[,5], lty=3)
lines(P11, col="purple")

plot(Q12[,3], ylim=c(0.003,0.02), axes=FALSE, xlab="Date",
     ylab="Forecasted SD Quantiles", main="SV-t")
box(); axis(side=2)
axis(side=1, at=c(1,12,23), labels=c("Dez 30","Jan 17","Jan 31"))
axis(side=1, at=c(1,6.5,12,17.5,23), labels=FALSE)
lines(Q12[,1], lty=3)
lines(Q12[,2], lty=2)
lines(Q12[,3], col="red")
lines(Q12[,4], lty=2)
lines(Q12[,5], lty=3)
lines(P12, col="purple")
par(mfrow=c(1,1))


#Forecasting Evaluation
#Loss function and R^2 table
print(xtable(FCsumm,
```

```
            caption="Evaluation of all forecasts, using the MSE, MAE,
             MSD and QLIKE loss functions, as well as the coefficient
             of determination. For better displayability, some loss
             functions were rescaled and the best values where emphasized.",
            label="table:FCsumm",
            align="lccccc",
            digits=3
),
include.rownames=TRUE,
caption.placement="bottom"
)

#Variance proxy plot
plot(proxy, axes=FALSE, xlab="Date",
     ylab="Proxy values", main="Variance estimates from interday returns.")
lines(proxy, col="red")
box(); axis(side=2)
axis(side=1, at=c(1,11.5,22), labels=c("Jan 1","Jan 17","Jan 31"))
axis(side=1, at=c(1,6.25,11.5,16.75,22), labels=FALSE)



##SECTION 5: Conclusion



#Appendix - Parameter estimates
#Estimates for non-stochastic volatility models with standard errors and
#p-values from t-tests
#Table was manually created
signif1
signif2
signif3
signif4
signif5
signif6
signif7
signif8

#MCMC parameter traces of SV models and the respective density plots
par(mfrow=c(2,3))
paratraceplot(SVMG)
paradensplot(SVMG,showobs=FALSE,showxlab=FALSE) #dashed line is prior
par(mfrow=c(1,1))

par(mfrow=c(2,4))
paratraceplot(SVMT)
paradensplot(SVMT,showobs=FALSE,showxlab=FALSE)
par(mfrow=c(1,1))
```

Miscellaneous

```
###This part just contains some random code snippets that might be useful
###or clear up the implementation of some functions

##Formulas of Kurtosis, Skew and Leverage
#Kurtosis:
SKRcm <- all.moments(SKR, order.max=4, central=TRUE)[-1] #central moments
SKRcm
kurtosis(SKR) #Note: some packages contain "excess" kurtosis function, which are
              #given by kurtosis-3 (subtracting gaussian kurtosis)
mean((SKR-mean(SKR))^4)/mean((SKR-mean(SKR))^2)^2   #Pearson's measure of kurtosis
SKRcm[4]/SKRcm[2]^2

#Skewness:
skewness(SKR)
mean((SKR-mean(SKR))^3)/mean((SKR-mean(SKR))^2)^(3/2)   #Pearson's skewness
SKRcm[3]/SKRcm[2]^(3/2)


##About the "rugarch" package

#Starting estimate sigma_0 is approximately the sample standard deviation
#of the full data sample
G1@fit$sigma[1]; sd(SKR)
#This can be problematic, if the beginning of the data sample shows much less
#variance than the middle/end, see this (exaggerated) dummy example:
set.seed(1)
A <- rnorm(1000)
for(i in 1:1000) (A[i] <- A[i]*i)
AG <- ugarchfit(data=A, spec=garch1.spec)
AG@fit$sigma[1]-1  #Misestimation of starting standard deviation by about 615
plot(AG@fit$sigma, type = "l")

#Also, the last value that would be available deterministically is not directly
#computed:
GG1$sigma[1780]; G1@fit$sigma[1780] #own vs. rugarch
#but it is implemented as the 1-step-ahead deterministic "forecast":
ugarchboot(G1, n.ahead=1, method="Partial", n.bootpred=5)@fsigma
#(same holds for the "betategarch", but not the "GAS" package that includes all 1780.)


##Maximum likelihood estimation functions
#The implementation with solnp runs smoothly and doesn't require overly restrictive
#parameter bounds to converge to a solution, like for example the mle() function
#from the "stats4" package. Unless the parameter boundaries and initial values
#are perfectly adjusted there, negative or infinitely large sigma values are
#produced -> NANs
Misc1 <- function(alpha, beta, omega, mu, sig0)
{
    LL_ret <- 1:(nSK-1);
    sigs <- 1:(nSK);
    sigs[1] <- sig0^2 ;
```

```
    for(i in 2:(nSK))
      (sigs[i] <- (omega+alpha*(SKR[i-1]-mu)^2+beta*sigs[i-1])) ;
    for(i in 1:(nSK-1))
      (LL_ret[i] <- (-0.5*log(2*pi*sigs[i])-((SKR[i]-mu)^2)/(2*sigs[i]))) ;
    return(sum(-LL_ret)) ;
} #negative LL

mle(Misc1, start=list(alpha=0.1, beta=0.9, omega=0, mu=mean(SKR), sig0=sd(SKR)),
control=list(trace=TRUE), lower=c(0,0.7,0,-0.05,0), upper=c(0.3,0.95,0.10,0.10,0.10),
method="L-BFGS-B") #Produces infinite values, even with more restrictive parameter
#bounds than solnp, which runs propertly:
solnp(pars=c(alpha=0.1, beta=0.9, omega=0, mu=mean(SKR), sig0=sd(SKR)),
fun=solfun, LB=c(0,0,0,-1,0), UB=c(1,1,1,1,1))


#Implementation of ARCH LM test by OLS:
ArchTest(SKR) #From "FinTS" package

#Implemented via projection of lagged values and linear regression(OLS):
Misc2 <- function(inp, lag){
  lag1 <- lag+1 #test with p=lag
  #lag+1 stands for 1 response and 12 coefficients (of lag values)

  emb <- embed(inp^2, lag1); #embeds squared returns into euclidean space of
                             #dimension lag+1
  emb.lm <- lm(emb[,1] ~ emb[,-1]); #OLS estimation with response ~ 12 coefficients
                                    #with intercept (in Wilkinson-Rogers notation)

  #Now calculate R^2, by
  #summary(emb.lm)$r.squared  #drawing R^2 values from linear model summary, or...
  #Rsqu(fit=fitted(emb.lm),data=SKR) #use Rsqu function from before, or..
  #direct calculation:
  rsqu <- sum((fitted(emb.lm)-mean(emb[,1]))^2)/sum((emb[,1]-mean(emb[,1]))^2);
  emb.n <- length(inp)-lag1; #"n-p"
  lmtest.stat <- rsqu*emb.n; #(n-p)*R^2 = F test statistic
  pval <- 1-pchisq(lmtest.stat, df=lag1-1); #Corresponding p-value for Chi-squared
                                            #distribution with p=12 degrees of freedom
  return(c(lmtest.stat, pval))
}
Misc2(inp=SKR, lag=12)


##Same procedure as above for the ADF test:
adf.test(SK)

Misc3 <- function(inp, lag){
  lag1 <- lag+1;
  inpd <- diff(inp);
  emb <- embed(inpd, lag1); #Absolute Difference series embedded in euclidean space
  lint <- lag1:length(inpd);       #Linear trend factor
  gammyt <- inp[lag1:length(SKR)]; #gamma factor
  emb.lm <- lm(emb[,1] ~ lint + gammyt + emb[,-1]); #OLS linear regression
  stat <- summary(emb.lm)$coefficients[3,1]/summary(emb.lm)$coefficients[3,2];
```

```
  return(stat)
}
Misc3(inp=SK, lag=12) #p-values for test statistic values have to be taken from
                      #quantile table
```

References

[AB1] T. G. Andersen, T. Bollerslev (1998): "Answering the skeptics: Yes, standard volatility models do provide accurate forecasts", International Economic Review Vol.39(4), p. 885-905

[ABCD1] T. G. Andersen, T. Bollerslev, P. F. Christoffersen, F. X. Diebold (2006): "Volatility and Correlation Forecasting", Handbook of Economic Forecasting, Amsterdam: North-Holland, p. 778-878

[AFT1] R. Adler, R. Feldman, M. Taqqu (1998); "A Practical Guide to Heavy Tails: Statistical Techniques and Applications", Birkhauser Boston Inc. Cambridge, MA, USA

[Ak1] H. Akaike (1973): "Information theory and an extension of the maximum likelihood principle", 2nd International Symposium on Information Theory(1973), Budapest, p. 267-281

[Ba1] Y. Bao (2015): "Should we Demean Data", Annals of Economics and Finance 16(1), p. 163-171

[BA1] K. P. Burnham, D. R. Anderson (2002): "Model selection and Multimodal Inference: A Practical Information-Theoretic Approach", 2nd edition, Springer-Verlag

[BA2] K. P. Burnham, D. R. Anderson: "AIC Myths and Misunderstandings", `https://sites.warnercnr.colostate.edu/anderson/wp-content/uploads/sites/26/2016/11/AIC-Myths-and-Misunderstandings.pdf`

[BD1] P. J. Brockwell, R. A. Davis (1991): "Time Series: Theory and Methods", 2nd edition, Springer-Verlag New York

[BEN1] T. Bollerslev, R. F. Engle, D. Nelson (1994): "ARCH models", Handbook of Econometrics Vol.4, Elsevier Science B.V., p. 2961-3038

[Bl1] F. Black (1976): "Studies in stock price volatility changes", Proceedings of the 1976 Meetings of the American Statistical Association, Business and Economical Statistics Section (1976), p. 177-181

[Bo1] T. Bollerslev (1986): "Generalized Autoregressive Conditional Heteroskedasticity", Journal of Econometrics, Vol. 31(3), p. 307-327

[Bo2] T. Bollerslev(1987): "A Conditionally Heteroskedastic Time Series Model for Speculative Prices and Rates of Return", The Review of Economics and Statistics, Vol. 69(3), p. 542-547

[Br1] T. S. Breusch (1978): "Testing for Autocorrelation in Dynamic Linear Models", Australian Economic Papers 17, p. 334-355

[CBA1] L. Catania, K. Boudt, D. Ardia (2016): "GAS: Generalized Autoregressive Score Models", R package

[CKL1] D. Creal, S.J. Koopman, A. Lucas (2008): "A General Framework for Observation Driven Time-Varying Parameter Models", Tinbergen Institute Discussion Paper 08-108/4

[Cl1] W. S. Cleveland (1979): "Robust locally weighted regression and smoothing scatterplots", Journal of the American Statistical Association Vol. 74, p. 829-836

[CW1] P. Carr, L. Wu (2011): "Leverage Effect, Volatility Feedback and Self-Exciting Market Disruptions", SSRN Electronic Journal (September 2011), DOI: 10.2139/ssrn.1306495

[DF1] D. A. Dickey, W. A. Fuller (1979): "Autoregressive Time Series with a Unit Root", Journal of the American Statistical Association Vol. 74 (No. 366), p. 427-431

[DGE1] Z. Ding, C. W. J. Granger, R. F. Engle (1993): "A long memory property of stock market returns and a new model", Journal of Empirical Finance Vol. 1, p. 83-106

[EB1] R.F. Engle, T. Bollerslev (1986): "Modelling the persistence of conditional variances", Econometric Reviews 5(1), p. 1-50

[Ed1] D. Edelbüttel (2013): "Seamless R and C++ Integration with Rcpp", Springer New York

[EL1] R. F. Engle, G. G. Lee (1999): "A long-run and short-run component model of stock return volatility", from "Cointegration, Causality and Forecasting", Oxford University Press, Chapter 20, p. 475-497

[En1] R. F. Engle (1982): "Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation", Econometrica 50(4), p. 987-1007

[FS1] C. Fernández, M. Steel (1998): "On Bayesian modelling of fat tails and skewness", Journal of the American Statistical Association Vol. 93, p. 359-371

[Fu1] W. A. Fuller (1995): "Introduction to Statistical Time Series", 2nd. ed., Wiley Series in Probability and Statistics, John Wiley and Sons

[Gh1] A. Ghalanos (2015): "rugarch: Univariate GARCH model", R package version 1.3-6

[GJR1] L. Glosten, R. Jagannathan, D. Runkle (1993): "Relationship between the expected value and volatility of the nominal excess returns on stocks", Journal of Finance, Vol. 48, p. 1779-1802

[Go1] L. G. Godfrey (1978): "Testing Against General Autoregressive and Moving Average Error Models when the Regressors Include Lagged Dependent Variables", Econometrica 46, p. 1293-1301

[Ha1] A. C. Harvey (2013): "Dynamic Models for Volatility and Heavy Tails", Cambridge University Press, New York

[HC1] A. C. Harvey, T. Chakravarty (2008): "Beta-t-(E)GARCH", Discussion Paper University of Cambridge CWPE 08340

[HRS1] A. C. Harvey, E. Ruiz, N. Shephard (1994): "Multivariate Stochastic Variance Models", The Review of Economic Studies Vol. 61(2), p. 247-264

[HS1] A. C. Harvey, Genaro Sucarrat (2014): "EGARCH models with fat tails, skewness and leverage", Elsevier Science, Computational Statistics & Data Analysis Vol. 76, p. 320-338

[JPR1] E. Jacquier, N. G. Polson, P. E. Rossi (1994): "Bayesian analysis of stochastic volatility models", Journal of Business & Economic Statistics Vol. 12, p. 371-389.

[Ka1] G. Kastner (2016): "Dealing with Stochastic Volatility in Time Series Using the R Package stochvol", Journal of Statistical Software Vol. 69(5), p. 1-30

[Ka2] G. Kastner (2015): "Heavy-Tailed Innovations in the R Package stochvol", R Package Vignette

[KFS1] G. Kastner, S. Frühwirth-Schnatter (2014): "Ancillarity-sufficiency interweaving strategy (ASIS) for boosting MCMC estimation of stochastic volatility models", Computational Statistics & Data Analysis Vol. 76, p. 408-423

[LB1] G. M. Ljung, G. E. P. Box (1978): "On a measure of lack of fit in time series models", Biometrika (1978), 65, 2, p. 297-303

[LLT1] K. L. Lange, R. J. A. Little, J. M. G. Taylor (1989): "Robust Statistical Modeling Using the t Distribution", Journal of the American Statistical Association Vol.84, No. 408, p.881-896

[Ne1] D. B. Nelson (1991): "Conditional Heteroskedasticity in Asset Returns: A New Approach", Econometrica, Vol. 59(2), p. 347-370

[Pa1] A. J. Patton (2011): "Volatility forecast comparison using imperfect volatility proxies", Journal of Econometrics Vol. 160 Issue 1„ p. 246–256

[Sc1] D. W. Scott (1992): "Multivariate Density Estimation: Theory, Practice and Visualization", Wiley

[Sch1] G. E. Schwarz (1978): "Estimating the dimension of a model", Annals of Statistics Vol. 6 (2), p. 461-464

[Su1] G. Sucarrat (2013): "betategarch: Simulation, Estimation and Forecasting of First-Order Beta-Skew-t-EGARCH models", The R Journal Vol.5/2, p. 137-147

[Su2] G. Sucarrat (2016): "betategarch: Simulation, Estimation and Forecasting of Beta-Skew-t-EGARCH models", R package version 3.3

[Ta1] S. J. Taylor (1982): "Financial Returns Modelled by the Product of Two Stochastic Processes: A Study of Daily Sugar Prices 1691-79", Anderson, O. D., ed., Time Series Analysis: Theory and Practice, 1. Amsterdam : North-Holland, pp. 203-226

[Ts1] R. S. Tsay (2005): "Analysis of Financial Time Series", John Wiley & Sons

[Yu1] J. Yu (2005): "On leverage in a stochastic volatility model", Journal of Econometrics Vol. 127, p. 165-178

[Za1] J-M. Zakoian (1994): "Threshold Heteroscedastic Models", Journal of Economic Dynamics and Control, Vol. 18(5), p. 931-955

R packages used (R Version 3.3.3)

Rsolnp : Alexios Ghalanos and Stefan Theussl (2015). Rsolnp: General Non-linear Optimization Using Augmented Lagrange Multiplier Method. R package version 1.16.

GAS: Catania, L., Boudt, K., Ardia, D. (2016). GAS: Generalized Autoregressive Score Models. R package version 0.1.8. `https://CRAN.R-project.org/package=GAS`

Ardia, D., Boudt, K., Catania, L. (2016a). Generalized autoregressive score models in R: The GAS package. Working paper. `https://ssrn.com/abstract=2825380`

Ardia, D., Boudt, K., Catania, L. (2016b). Value-at-Risk prediction in R with the GAS package. Working paper. `https://ssrn.com/abstract=2871444`

rugarch: Alexios Ghalanos (2015). rugarch: Univariate GARCH models. R package version 1.3-6.

betategarch: Genaro Sucarrat (2016). betategarch: Simulation, Estimation and Forecasting of Beta-Skew-t-EGARCH Models. R package version 3.3.
`https://CRAN.R-project.org/package=betategarch`

FinTS: Spencer Graves (2014). FinTS: Companion to Tsay (2005) Analysis of Financial Time Series. R package version 0.4-5.
`https://CRAN.R-project.org/package=FinTS`

forecast: Hyndman RJ (2017). forecast: Forecasting functions for time series and linear models . R package version 8.0, http://github.com/robjhyndman/forecast.

Hyndman RJ and Khandakar Y (2008). "Automatic time series forecasting: the forecast package for R." Journal of Statistical Software, *26*(3), pp. 1-22. http://www.jstatsoft.org/article/view/v027i03.

moment: Lukasz Komsta and Frederick Novomestky (2015). moments: Moments, cumulants, skewness, kurtosis and related tests. R package version 0.14. https://CRAN.R-project.org/package=moments

stochvol: Gregor Kastner (2016). Dealing with Stochastic Volatility in Time Series Using the R Package stochvol. Journal of Statistical Software, 69(5), 1-30. doi:10.18637/jss.v069.i05

Gregor Kastner (2016). stochvol: Efficient Bayesian inference for stochastic volatility (SV) models. R package version 1.3.2. https://CRAN.R-project.org/package=stochvol

xtable: David B. Dahl (2016). xtable: Export Tables to LaTeX or HTML. R package version 1.8-2. https://CRAN.R-project.org/package=xtable

Hmisc: Frank E Harrell Jr, with contributions from Charles Dupont and many others. (2016). Hmisc: Harrell Miscellaneous. R package version 4.0-2. https://CRAN.R-project.org/package=Hmisc

tseries: Adrian Trapletti and Kurt Hornik (2017). tseries: Time Series Analysis and Computational Finance. R package version 0.10-38.

pracma: Hans Werner Borchers (2017). pracma: Practical Numerical Math Functions. R package version 2.0.4. https://CRAN.R-project.org/package=pracma