NTNU

Kunnskap for en bedre verden

Department of Mathematical Sciences

TMA4285 - Time Series

Time series modelling of inflation

Authors:
Christian O. Moen, Jakob B. Heide, Celine Olsson, Gard W. Gravdal

19/02/2024

Table of Contents

1 Abstract

This project focuses on modeling the inflation rate in Norway, measured as the percentage change in the consumer price index. Using monthly data from February 2000 to December 2014, we employ a regression model with autocorrelated errors, incorporating covariates such as household consumption, interest rate, and bankruptcies. The data is obtained from Statistisk sentralbyrå (SSB). The project explores SARIMA and GARCH models, providing a theoretical background and detailing the estimation and diagnostic procedures. Regression with autocorrelated errors is applied to analyze time dependence, yielding coefficient estimates for covariates. Diagnostic tests confirm the adequacy of the chosen models. The SARIMA$(0,0,0) \times (2,0,0)_{12}$ model is selected based on the Akaike information criterion (AIC) for forecasting. Despite challenges in predicting inflation and volatility for 2015, the project highlights the importance of covariate selection in regression modeling.

2 Introduction

In this project, we consider the rate of inflation in Norway. The inflation rate is measured as the percentage change in the consumer price index. To model the inflation, we employ a regression model with autocorrelated errors, where we choose three covariates - household consumption of goods, interest rate, and bankruptcies. The data comprises monthly measurements and ranges from February, 2000 to December, 2014.

All the data is taken from *Statistisk sentralbyrå* 2023, with the exception of the interest rate data, which is taken from the *Norwegian Central Bank* 2023. Looking at the data set, we see that all the values for inflation lie in the interval [-1.3,2.4]. In addition, all the numbers in the list have zero-values after the tenth decimal place. This makes the numbers somewhat discrete in nature, however we consider the inflation to be continuous. In Figure 1 below we can see the plot of the inflation rate.

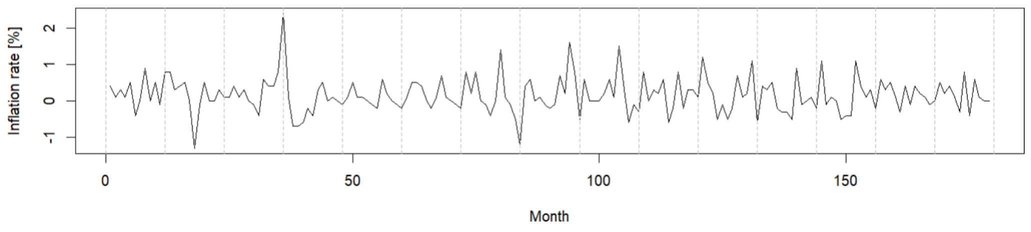

Figure 1: Inflation rate in Norway. Data from Feb 2000 to Dec 2014.

2.1 Regression with autocorrelated errors

To analyse the time dependence of the inflation rate, we will utilize a method often referred to as regression with autocorrelated errors. The idea of this method is to choose some covariates which are related to the inflation rate, and then fit a linear model to the covariates with inflation as the response. More specifically, we fit the linear regression

$$y_i = X_i\beta + x_i$$

where the error term x_i has some time dependence, i.e. $x_t = y_t - X\beta$ is the time series we will analyse. Intuitively, we hope to remove the effect of the covariates on inflation, so that x_t only contains the hidden time structure of inflation.

3 Results

In the linear regression, we found the coefficient estimates to be: $\beta_{consum} = -0.061$, $\beta_{bank} = 0.0005$, $\beta_{interest} = 0.020$, which gives the residuals x_t, shown in Figure 2.

3.1 Model candidates

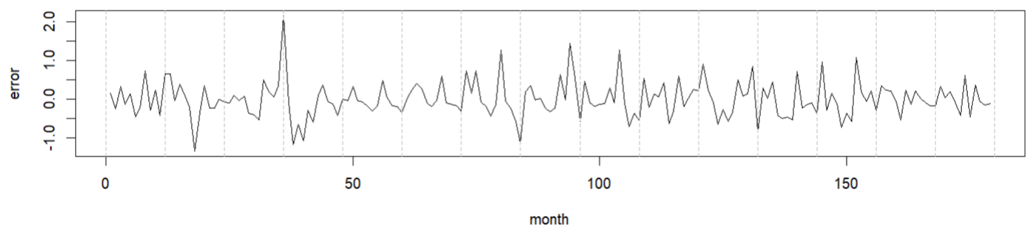

Figure 2: The autocorrelated errors of the linear regression. Staring from February 2000.

The time series that arises from the residuals seems to have zero mean and variance independent of time, which indicates stationarity. To further investigate the time dependence of the series, we plot the sample autocorrelation function (ACF) and the partial autocorrelation function (PACF). How these are calculated can be seen in Appendix D.

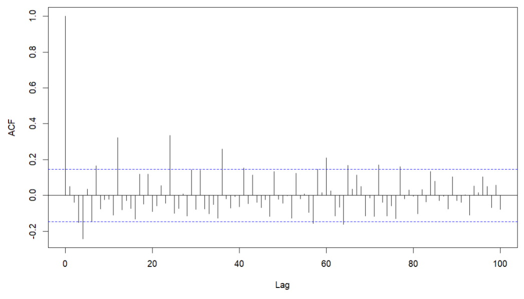

Figure 3: Sample ACF

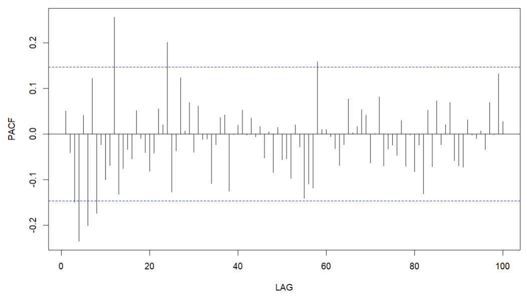

Figure 4: Sample PACF

There is a clear seasonal trend in the ACF plot - we have three significant spikes at lag 12,24 and 36. In addition, there is a significant spike in the PACF at lag 12 and 24. The spikes in the PACF, together with the spikes in the ACF that seem to be trailing off at the seasonal lags, may be indicative of a seasonal, auto-regressive model with either $P = 1$ og $P = 2$. Hence, the first two models we will consider are SARIMA$(0,0,0) \times (1,0,0)_{12}$ and SARIMA$(0,0,0) \times (2,0,0)_{12}$ (see Appendix B for details, including state space forms of the models). Moreover, we will examine a differenced model, SARIMA$(0,0,0) \times (1,1,0)_{12}$, where we attempt to remove the effect of the seasonality. See Appendix C for plots concerning this model. Finally we will consider a GARCH(1,0) (or ARCH(1)) model. We will see that the diagnostics for this simple model suggests a good fit.

3.2 Parameter estimation

We opt for BFGS as our optimization procedure. The first and second order derivatives in Newton's method for optimization are numerically approximated (BFGS estimates the second order, and a finite difference scheme is used for the first order). The first model, SARIMA$(0,0,0) \times (1,0,0)_{12}$, has one unknown parameter, which is estimated to be $\alpha = 0.3167883$. The second model has two parameters, which are estimated to be $\alpha_{12} = 0.2431699$ and $\alpha_{24} = 0.2480338$. The parameter in the differenced model is found to be $\alpha_{diff} = -0.495351$. The three SARIMA-model candidates are thus

$$x_t = 0.3168x_{t-12} + w_t \ , \ x_t = 0.2432x_{t-12} + 0.2480x_{t-24} + w_t \ , \ \nabla_{12}^1 x_t = -0.4954\nabla_{12}^1 x_{t-12} + w_t$$

Numerically estimating the parameters in our GARCH-model follows a similar procedure. We consider a GARCH(1,0)/ARCH(1) model, which is given by

$$r_t = \sigma_t \epsilon_t \tag{1a}$$

$$\sigma_t^2 = \alpha_0 + \alpha_1 r_{t-1}^2 \tag{1b}$$

We know that the inflation rate is the measured percentage change in consumer price index. From Figure 1, we see a tendency for the volatility of the inflation to cluster around certain periods. To capture these short-term changes effectively, we choose to look at the differenced rate $r_t = x_t - x_{t-1}$ where x_t is the inflation rate at time t. This model should capture the volatility or uncertainty associated with month-to-month changes in the inflation rate and is interpreted as how much the inflation rate tends to fluctuate from one month to the next.

From equation (16), we find that the likelihood to be maximized is given by

$$l(\alpha_0, \alpha_1) = \frac{1}{2}\sum_{t=2}^{n} \ln\left(\alpha_0 + \alpha_1 r_{t-1}^2\right) + \frac{1}{2}\sum_{t=2}^{n} \frac{r_t^2}{\alpha_0 + \alpha_1 r_{t-1}^2}$$

Again we opt for BFGS as our optimization procedure. The two unknown parameters in the GARCH-model are estimated to be $\alpha_0 = 0.2699310$ and $\alpha_1 = 0.3828288$.

3.3 Diagnostics

The next step we did was to do some diagnostic tests on our models. For the SARIMA models this included plotting the standardized residuals, looking at the ACF plots and histograms of the standardized residuals. For the GARCH models this included looking at the estimated volatility of our time series and the normality of the return rate.

To calculate the standardized innovations we used: $e_t = (x_t - \hat{x}_t^{t-1})/\sqrt{\hat{P}_t^{t-1}}$. Here $\hat{x}_t^{t-1}$ is the one-step-ahead prediction of x_t derived from the fitted model, and $\hat{P}_t^{t-1}$ is the estimated one-step-ahead error variance. If our model fits well the standardized residuals should behave like a independent and identically distributed (iid) sequence with mean equal to zero and variance equal to one. In other words, we want the standardized residuals to approximate a white noise process.

To investigate the normality we look at the histogram and the Q-Q plot of the standardized residuals. We also want to look at the sample autocorrelations of the residuals. Our main goal when inspecting the ACF is to identify possible deviations from the assumption of independence.

Below, in Figure 5, we see the diagnostics plots for our first model SARIMA$(0,0,0) \times (1,0,0)_{12}$.

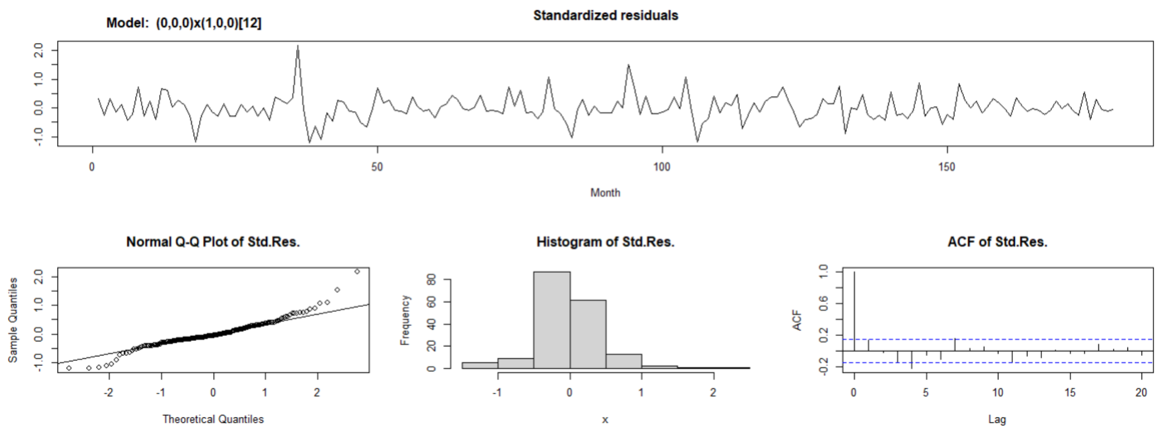

Figure 5: Diagnostic plots of SARIMA$(0,0,0) \times (1,0,0)_{12}$ model.

First of all, we do not see any obvious patterns in the plot of the standardized residuals. It seems that the data points are randomly distributed around zero, indicating zero mean and constant variance. Furthermore we see from the histogram that the standardized residuals look fairly normally distributed. From the Q-Q plot we see that the points mainly follow the straight line, with some deviation at the ends. This means that the assumption of normality is reasonable, with the exception of possible outliers. Looking at the ACF plot of the standardized residuals we see no major deviation from our model assumptions, i.e. there is a lack of correlation. We do see that there is a small spike at lag 4 in the ACF plot. We choose to ignore this since we know that the ACF of a white noise process on average will have 5% chance of getting a lag that is significant. So when looking at 20 lags, it is expected that at least one would be significant, purely by chance. All this put together implies that our model fits well.

The diagnostic plots for our second and third SARIMA model gave almost exactly the same results as the first model, and can be seen in Appendix E. The standardized residuals for all the SARIMA models met the requirements of normality, independence, zero mean and constant variance, i.e. they approximated a white noise model.

Now over to the diagnostics of our GARCH model. To investigate the fit of the model, we check visually if the estimated volatility captures the variability of the return rate r_t. In addition, we check the assumption of normality for the return rate $r_t = x_t - x_{t-1}$. Figure 6 shows the diagnostics of the ARCH(1)-model below.

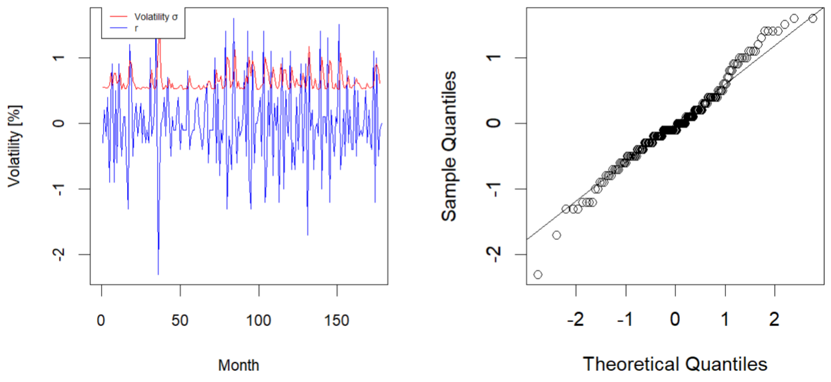

Figure 6: Diagnostics ARCH(1)-model.

From the left side of Figure 6 we observe volatility clustering, meaning that in periods where r_t has larger magnitudes, the volatility has corresponding spikes. In general, it seems like the volatility captures the trends of the return rate r_t, which suggests that the ARCH(1) model is a good fit. On the right side of Figure 6 we see the QQ-plot of the return rate $r_t = x_t - x_{t-1}$. The points follow the straight line to some degree, with deviations at the ends. The staircase-like behavior of the points comes from the discreteness of the data set for the inflation rate discussed in the introduction, however this should not have a great effect on the assumption of normality for the return rate r_t. In general, the QQ-plot indicates that the assumption of normality for r_t is somewhat reasonable, however there are indications that the assumption can be questionable. This is something to keep in mind when considering the ARCH(1)-model.

3.4 Model selection

We have three candidate models, from which we need to choose one to use for forecasting. To select the best model, we turn to the *Akaike information criterion* (AIC), which is defined as

$$AIC = -2\log L_k + 2k$$

where L_k is the maximised likelihood for each model, and k is the number of parameters. We have already calculated and maximised the likelihoods when we estimated the parameters in the models. We denote by AIC_i for $i = 1, 2, 3$, the AIC score for the SARIMA model with $P = 1$, the SARIMA model with $P = 2$, and the differenced model, respectively. We find the AIC scores to be

$$AIC_1 = 35.34462 \quad AIC_2 = 35.04459 \quad AIC_3 = 38.20074$$

We choose the model with the lowest AIC score - model 2, the SARIMA$(0,0,0) \times (2,0,0)_{12}$ is our model of choice for forecasting.

3.5 Forecasting

We now consider the main objective of this project, which is to forecast the inflation rate. To do this, we import data for our covariates (household consumption, interest rate and bankruptcies) for one year, that is, from January 2015 to December 2015. Using the SARIMA model, the Kalman filter forecasts the regression error x_t, where we denote the forecasted errors by $\hat{x}_t$. The forecasted inflation is then given by

$$\hat{y}_t = X\beta + \hat{x}_t$$

where X is the design matrix containing the new data and β are the regression coefficient estimates found in the linear regression. In addition, we can also attempt to forecast the volatility, with the ARCH(1)-model. To forecast the volatility, we use the following estimate:

$$\hat{\sigma}^2_{t+1} = \hat{\alpha}_0 + \hat{\alpha}_1 r_t^2$$

Furthermore, the unknown returns r_{t+1} are drawn from the normal distribution using the approximation

$$r_{t+1} \sim N(0, \sqrt{\hat{\alpha}_0 + \hat{\alpha}_1 r_t})$$

The predicted volatility for the ARCH(1)-model given in equation (1) where r_t is the differenced inflation rate is shown in Figure 8 along with the actual volatility. The predicted inflation rate is shown in Figure 7, along with the actual inflation rate for year 2015. Further model selection is investigated in Appendix F.

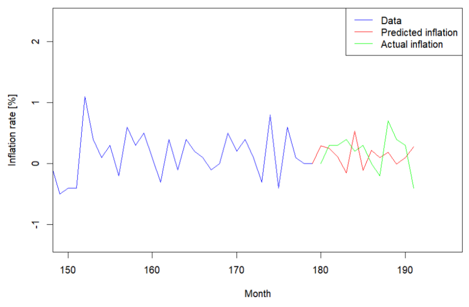

Figure 7: Prediction of inflation rate using the SARIMA$(0,0,0) \times (2,0,0)_{12}$ model.

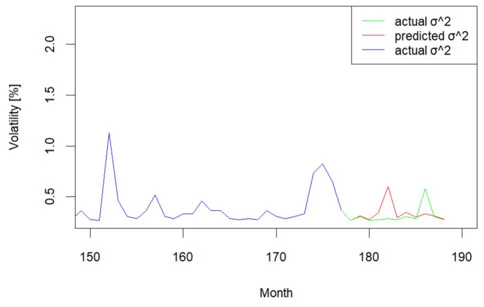

Figure 8: Prediction of the volatility using ARCH(1)-model where r_t is the inflation rate differenced.

4 Discussion

In Figure 7 and 8, we saw that the predictions of inflation rate and volatility were poor, compared to the actual inflation rate in 2015. In this section, we will point to a few possible reasons as to why that might be.

Firstly, the choice of covariates in the linear regression seems important. We chose household consumption, interest rate and bankruptcies, which from a macroeconomic perspective makes sense - these are all good economic indicators. However, in reality, the relationship between inflation and these indicators may be more complex than a simple linear regression - in fact, we saw that the β's found in the linear regression were quite small. This indicates that the covariates were not able to explain the inflation rate to any large degree - which in turn means that the regression error, our time series, is similar to the inflation rate itself. A better choice of covariates might have lead to better results and thus forecasting.

With the GARCH-model, we attempted to estimate and predict the volatility of the return rate $r_t = x_t - x_{t-1}$, where x_t is the inflation rate. We found that the estimated volatility captured the trend of the return rate r_t well. Predicting volatility using GARCH-models is a very debated subject. In general, it is very difficult to predict future volatility due to the chaotic behavior of such time series. It becomes increasingly difficult the further ahead in time we attempt to predict. This reflects the result of forecasting the inflation rate using the ARCH(1)-model, which did not predict the future volatility very accurately.

5 Conclusion

In this project we have employed SARIMA- and GARCH-models for modeling and forecasting inflation in Norway, utilizing the covariates household consumption, interest rate, and bankruptcies. The selected SARIMA$(0,0,0)\times$

$(2,0,0)_{12}$ model, based on AIC scores, demonstrated the most reasonable fit, but faced challenges in forecasting accuracy. The forecasting of the volatility with the ARCH(1)-model faced similar challenges. The challenge in forecasting is presumably due to the inherent complexity in the relationship between economic indicators and inflation, as opposed to deficiencies in the chosen methods or models.

Bibliography

Norwegian Central Bank (2023). Visited November 11, 2023. URL: https://app.norges-bank.no/query/index.html#/no/.

Shumway, Robert H. and David S. Stoffer (2023). *Time Series Analysis and Its Applications: With R Examples.* 4th ed. Visited on November 13, 2023. Springer International Publishing. URL: https://link.springer.com/book/10.1007/978-3-319-52452-8.

Statistisk sentralbyrå (2023). Visited November 11, 2023. URL: https://www.ssb.no/statbank/.

Appendix

A Theory

A.1 SARIMA

A classical decomposition of time series is given by

$$x_t = m_t + s_t + y_t \tag{2}$$

where m_t captures the trend of the data, s_t captures the seasonality and y_t is stationary, random noise. We will mainly consider trends called Moving Average (MA(q)) and Autoregressive (AR(p))

There are many different types of trends, but we will restrict our scope to Autoregressive (AR(p)) and Moving Average (MA(q)). Before we define AR and MA we will define the backshift operator B, difference operator ∇ and two functions for notational purposes. The backshift operator is given by

$$B^q w_t = w_{t-q}, \tag{3}$$

where we require $B^{-1}B = 1$ such that we get an inverse operator B^{-1} called forward-shift. Then, we can write the difference operator $\nabla x_t = x_t - x_{t-1}$ in terms of the backshift operator, that is, $\nabla = (1 - B)$. Furthermore, we denote the differences of order d by

$$\nabla^d = (1 - B)^d. \tag{4}$$

Using the backshift operator we can define a general expression for a MA(q) model,

$$x_t = \theta_q(B)w_t, \tag{5}$$

where $\theta_q(B) = 1 + \theta_1 B + ... + \theta_q B^q$, and an AR model of order p,

$$\phi_p(B)x_t = w_t, \tag{6}$$

where the autoregressive operator $\phi_p(B) = 1 - \phi_1 B - ... - \phi_p B^p$. Combining the left hand side (LHS) of (6) and the right hand side (RHS) of (5) gives a model called autoregressive moving average, ARMA(p,q), which is given by,

$$\phi_p(B)x_t = \theta_q(B)w_t. \tag{7}$$

MA and AR models can be shown to be stationary. If a model is not stationary, the difference could still be stationary. E.g., the simple random walk, $x_t = x_{t-1} + w_t$, differenced, $\nabla x_t = x_t - x_{t_1} = w_t$, is clearly stationary since white noise is stationary. That is, in some cases, one could difference the data to get stationarity. The order, d, of the differencing can also vary in order for the data to obtain stationarity. Now, a process x_t is called ARIMA(p, d, q) if

$$y_t = \nabla^d x_t \tag{8}$$

is ARMA(p, q). That is, and ARIMA model is given by $\phi_p(B)y_t = \theta_q(B)w_t$, or equivalently,

$$\phi_p(B)(1 - B)^d x_t = \theta_q(B)w_t. \tag{9}$$

Data may contain sequentially repeating trends called seasonality. One such seasonality could be the temperature of the morning hours of a day. It would typically start at low temperatures and increase as the sun heats up the area. This process would repeat itself the day after and so on. If we were to consider the first five hours of a morning, we could see that each hour of the morning is similar to the same hour for a different morning. In the dataset we would see that the temporal dependence is strongest at multiples of the seasonal lag $s = 5$. The seasonality can be incorporated into the ARIMA model by setting the backshift operator to multiples of s. That is, a seasonal ARIMA, or SARIMA is given by

$$\Phi_P(B^s)\phi_p(B)\nabla_s^D \nabla^d x_t = \delta + \Theta_Q(B^s)\theta_q(B)w_t, \tag{10}$$

where seasonal difference of order D is given by $\nabla_s^D = (1 - B^s)^D$, and seasonal autoregressive operator $\Phi_P(B^s)$ and seasonal moving average operator $\Theta_Q(B^s)$ are analogous to their ordinary counterparts, but with different parameters and backshift raised to multiples k of s, where k goes to Q for MA and P for AR. For more details we refer the reader to chapter 3.9 of Shumway and Stoffer 2023.

A.2 GARCH

Knowledge about the conditional mean (conditioned on the past) may provide very little information about a time series process. In some cases it can be of greater interest to consider the temporal variability of the process. In finance, this variability is commonly known as *volatility.* E.g., modelling waves in a short time span (shorter than the tide). Clearly, the mean height of the waves are at sea level, but the variability may change greatly depending on wind and other factors. High variability may suggest a storm in that time period. The volatility of a process is modelled by considering the return r_t, or relative slope, given by

$$r_t = \frac{x_t - x_{t-1}}{x_{t-1}}. \tag{11}$$

Note that for small percentage changes in the return, $\nabla \ln(x_t) \approx r_t$ (footnote page 17 of Shumway and Stoffer 2023). The model we will consider is given by two equations,

$$r_t = \sigma_t \epsilon_t, \quad \epsilon \overset{iid}{\sim} \mathcal{N}(0,1) \tag{12}$$

$$\sigma_t^2 = \alpha_0 + \sum_{j=1}^{p} \alpha_j r_{t-j}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2, \tag{13}$$

and is called a *generalized autoregressive conditionally heteroscedastic* model, or GARCH(p, q). It can be shown that $r_t|\mathcal{R}_2 \sim \mathcal{N}(0, \sigma_t^2)$, where $\mathcal{R}_2 = (r_t, ..., r_{t+m})$ and $m = \max(p, q)$. Because of the normality of the conditional return we can find MLE of $\boldsymbol{\alpha} = (\alpha_0, ..., \alpha_p)$ and $\boldsymbol{\beta} = (\beta_1, ..., \beta_q)$. The likelihood to be maximized is given by

$$\begin{aligned} L(\boldsymbol{\alpha}, \boldsymbol{\beta}|\mathcal{R}_1) &= \prod_{t=m+1}^{n} f_{\boldsymbol{\alpha},\boldsymbol{\beta}}(r_t|\mathcal{R}_2) \\ &= \prod_{t=m+1}^{n} \left(2\pi\sigma_t^2\right)^{-1/2} \exp\left(-\frac{1}{2}\frac{r_t^2}{\sigma_t^2}\right) \qquad (14) \\ &\propto \prod_{t=m+1}^{n} \left(\sigma_t^2\right)^{-1/2} \exp\left(-\frac{1}{2}\frac{r_t^2}{\sigma_t^2}\right) \qquad (15) \end{aligned}$$

where $\mathcal{R}_1 = (r_1, ..., r_m)$ and σ_t^2 is given by equation (13). Following the notation from Shumway and Stoffer 2023 we define the criterion function l to be minimized as proportional to $-\ln L$, such that

$$l(\boldsymbol{\alpha}, \boldsymbol{\beta}|\mathcal{R}_1) = \sum_{t=m+1}^{n} \ln\left(\sigma_t^2\right) + \frac{r_t^2}{\sigma_t^2}, \tag{16}$$

where σ_t^2 is given by Equation (13). We will denote the parameter estimates of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ by $\hat{\boldsymbol{\alpha}}$ and $\hat{\boldsymbol{\beta}}$, respectively. With the parameter estimates we can get *one-step-ahead forecasts* of the volatility given by

$$\hat{\sigma}_{t+1}^2 = \hat{\alpha}_0 + \sum_{j=1}^{p} \hat{\alpha}_j r_{t+1-j}^2 + \sum_{j=1}^{q} \hat{\beta}_j \hat{\sigma}_{t+1-j}^2. \tag{17}$$

Furthermore, the volatility forecasts can be used to get return forecasts by setting $\hat{r}_{t+1} = \hat{\sigma}_{t+1}\epsilon$.

A.3 State space form and the Kalman filter

A linear and Gaussian state space model takes the form

$$x_{t+1} = \mathrm{A}x_t + \mathrm{H}v_t \tag{18}$$

$$y_t = \mathrm{B}x_t + \mathrm{D}w_t \tag{19}$$

where (18) is the state equation, and (19) is the observation equation. A, H, B and D are matrices depending on the model, y_t is the observation vector, and x_t is the state vector. v_t and w_t are white noises for the state and observations, respectively.

To numerically estimate the parameters in our SARIMA models, we implement a procedure involving the Kalman filter and a Newton-Raphson optimization method. The goal of the optimization is to maximize the likelihood of the data, or in practice, minimize the negative log-likelihood. The Kalman filter generates estimators for the unobserved state x_t, conditional on the data y_t, where we denote the state vector conditional on $y_1, y_2, \ldots, y_{t-1}$ as x_t^{t-1}. In particular, the Kalman filter calculates the *innovations*

$$\epsilon_t = y_t - \mathrm{B}x_t^{t-1}$$

which are independent, normally distributed random variables with zero mean and covariance matrices

$$\Sigma_t := \mathrm{B}S_t^{t-1}\mathrm{B}^T + \mathrm{D}\mathrm{D}^T$$

where

$$\mathrm{S}_t^{t-1} = \mathrm{A}(\mathrm{I} - \mathrm{M}_{t-1}\mathrm{B})\mathrm{S}_{t-1}^{t-2}\mathrm{A}^T + \mathrm{H}\mathrm{H}^T$$

and M is the projection matrix,

$$\mathrm{M}_{t-1} = \mathrm{S}_{t-1}^{t-2}\mathrm{B}^T\Sigma_{t-1}^{-1}$$

Thus, the Kalman filter computes Σ_t iteratively with some initial condition $\mathrm{S}_1^0 = \mathrm{S}_0$.

The log-likelihood is then defined as

$$L(\theta; y) = \frac{1}{2}\left(\sum_{t=1}^{n} \ln|\Sigma_t(\theta)| + \sum_{t=1}^{n} \epsilon_t(\theta)^T \Sigma_t^{-1}(\theta)\epsilon_t(\theta)\right)$$

where θ contains the unknown parameters in our state space model. We summarize the procedure below:

1. Choose initial values $\theta^{(0)}$.
2. for each iteration of the Newton-Raphson procedure:
    (a) Run the Kalman filter, using $\theta^{(t)}$ to obtain updated values of ϵ_t and Σ_t.
    (b) Obtain $\theta^{(t+1)}$ by Newton-Raphson.

In practice, we include the Kalman filter in the objective function.

B Models

A SARIMA$(0,0,0) \times (1,0,0)_{12}$ model can be written as

$$(1 - \alpha B^{12})y_t = w_t \Leftrightarrow y_t = \alpha y_{t-12} + w_t$$

In state space form, the model is

$$\begin{aligned} x_{t+1} &= \mathrm{A}x_t + \mathrm{H}v_t \\ y_t &= \mathrm{B}x_t \end{aligned}$$

where $x_t, x_{t+1} \in \mathbb{R}^{12}$, $\mathrm{A} \in \mathbb{R}^{12\times 12}$, $\mathrm{H} \in \mathbb{R}^{12}$, and $\mathrm{B} \in \mathbb{R}^{12}$ is a row vector. More specifically,

$$\mathrm{A} = \begin{bmatrix} 0 & 0 & \cdots & 0 & \alpha \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & & \vdots & \\ \vdots & & \ddots & & \\ 0 & & \cdots & 1 & 0 \end{bmatrix}, \; x_t = \begin{bmatrix} y_t \\ y_{t-1} \\ \vdots \\ y_{t-11} \end{bmatrix}, \; \mathrm{H} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \; \mathrm{B} = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}$$

A SARIMA$(0,0,0) \times (2,0,0)_{12}$ model can be written as

$$(1 - \alpha_{12}B^{12} - \alpha_{24})y_t = w_t \Leftrightarrow y_t = \alpha_{12}y_{t-12} + \alpha_{24}y_{t-24} + w_t$$

In state space form, the model is

$$\begin{aligned} x_{t+1} &= \mathrm{A}x_t + \mathrm{H}v_t \\ y_t &= \mathrm{B}x_t \end{aligned}$$

where $x_t, x_{t+1} \in \mathbb{R}^{24}$, $\mathrm{A} \in \mathbb{R}^{24\times 24}$, $\mathrm{H} \in \mathbb{R}^{24}$, and $\mathrm{B} \in \mathbb{R}^{24}$ is a row vector. More specifically,

$$\mathrm{A} = \begin{bmatrix} 0 & \cdots & 0 & \alpha_{12} & 0 & \cdots & \alpha_{24} \\ 1 & 0 & \cdots & & & 0 & 0 \\ 0 & 1 & \cdots & & & & \\ \vdots & & \ddots & & & & \\ 0 & & \cdots & & & 1 & 0 \end{bmatrix}, \; x_t = \begin{bmatrix} y_t \\ y_{t-1} \\ \vdots \\ y_{t-23} \end{bmatrix}, \; \mathrm{H} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \; \mathrm{B} = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}$$

C ACF, PACF for differenced model

The differenced time series is shown:

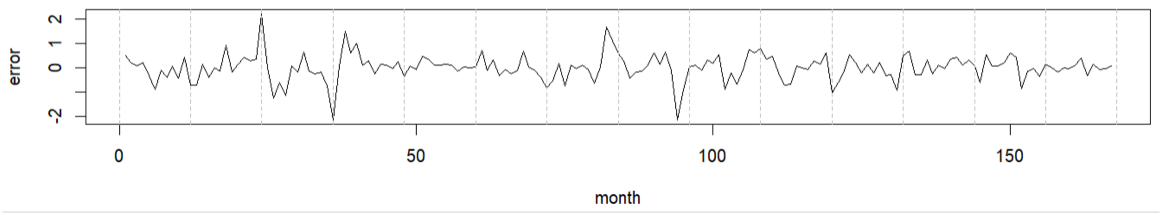

Figure 9: The 12-lag differenced time series.

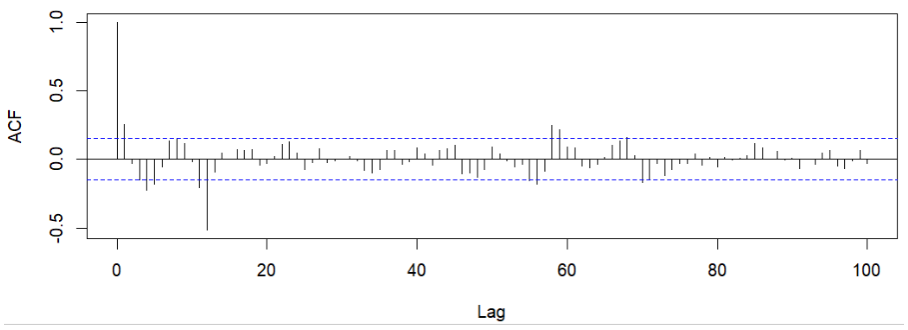

Figure 10: ACF of the 12-lag differenced time series.

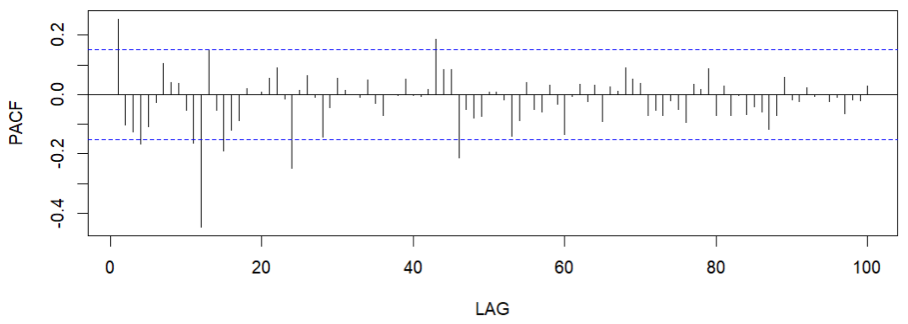

Figure 11: Sample PACF of the 12-lag differenced time series.

D Sample ACF and PACF

The sample autocorrelation (ACF) at lag h is defined as

$$\hat{\rho}(h) = \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)},$$

where $\hat{\gamma}(h)$ is the autocovariance function defined as

$$\hat{\gamma}(h) = \frac{1}{N}\sum_{t=0}^{N-h}(x_{t+h} - \bar{x})(x_t - \bar{x}).$$

$\bar{x}$ refers to the sample mean.

The partial autocorrelation function (PACF), denoted ϕ_{hh} for $h = 1, 2, \cdots$, is defined as

$$\begin{aligned}\phi_{11} &= \text{corr}(x_{t+1}, x_t) = \rho(1)\\ \phi_{hh} &= \text{corr}(x_{t+h} - \hat{x}_{t+h}, x_t - \hat{x}_t), \quad \text{for } h > 1\end{aligned}$$

where $\hat{x}_{t+h} = \sum_{j=1}^{h-1} a_j x_{t+j}$, and the a's are found by minimizing $E(x_{t+h} - \sum_{j=1}^{h-1} a_j x_{t+j})$.

We used the Durbin-Levinson algorithm to iteratively calculate the sample PACF. The definition of this algorithm is found as Property 3.4 in Shumway and Stoffer 2023.

E Diagnostic plots for SARIMA model 2 and 3

Our second model, SARIMA$(0, 0, 0) \times (2, 0, 0)_{12}$, have the following diagnostic plots.

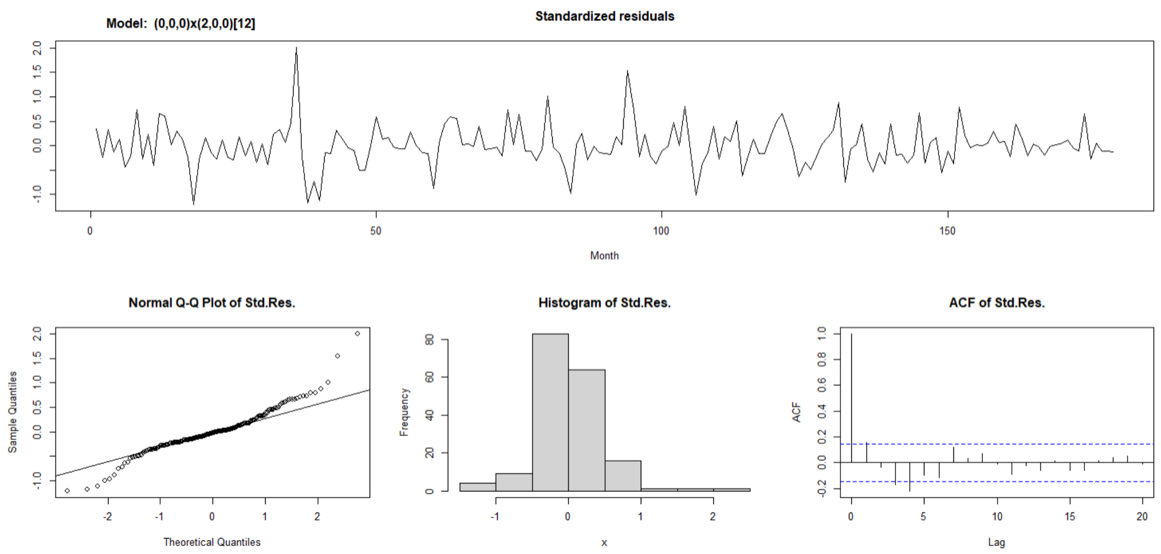

Figure 12: Diagnostic plots of SARIMA$(0, 0, 0) \times (2, 0, 0)_{12}$ model.

In Figure 12 we see something fairly similar to our first model. There is no clear patterns in the plot of the standardized residuals, and both the histogram and Q-Q plot indicate normal distribution (with some deviation at the ends in the Q-Q plot). Again we see some minor correlation at lag 4 which is even smaller than for the previous model, so again we disregard this. We therefore conclude that the ACF plot shows no clear sign of correlation, indicating independence. Again it looks like the model fits well.

Now for the last SARIMA model, SARIMA$(0, 0, 0) \times (1, 1, 0)_{12}$. Figure 13 shows the diagnostic plots for this model.

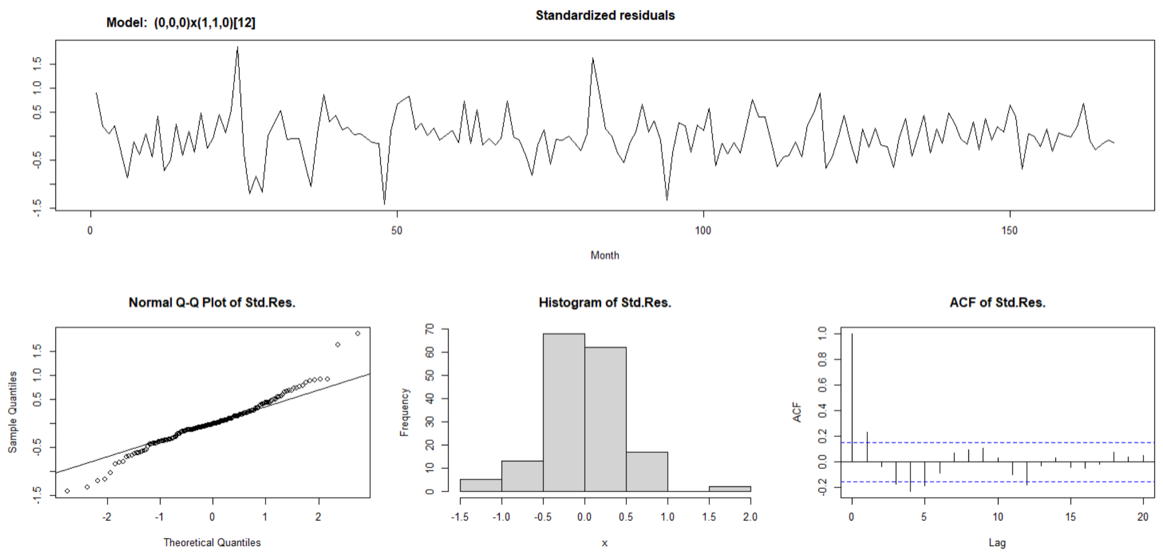

Figure 13: Diagnostic plots of SARIMA$(0,0,0) \times (1,1,0)_{12}$ model.

Again we see no patterns in the plot of the standardized residuals, and both the histogram and Q-Q plot indicates normality. There seems to be no significant correlation in the ACF plot, indicating independence. There is a small correlation at lag 1 and 4, but we ignore these for the same reasons as before. So again our model seems to fit well.

F GARCH models

Following are results from fitting ARCH and GARCH with various parameters using the theory given in Section A.2. By Table 1 we see that the likelihood typically increase with the number of parameters. The few exceptions might be because of inaccuracies in numerical optimization. As GARCH models typically have a flat likelihood, it is prone to give local optimums, which ultimately give results that not necessarily coincide with the theory. For both the regression residual data and the differenced data we see that the AIC suggest the simplest models, namely ARCH(1). The fits on the regression residual data have better likelihood and AIC scores, but Figure 14 may suggest a better fit on the differenced data.

Model	l	$l-m$	AIC	AIC$-m$
ARCH$(1)_d$	-4.473	-4.473	10.946	10.946
ARCH$(2)_d$	-4.476	-5.008	12.953	14.017
ARCH$(10)_d$	-0.636	-0.464	21.273	20.928
GARCH$(1,2)_d$	-4.473	-5.004	14.946	16.009
ARCH$(1)_r$	46.458	46.458	-90.915	-90.915
ARCH$(2)_r$	47.273	46.614	-90.546	-89.228
ARCH$(10)_r$	53.940	49.289	-87.880	-78.579
GARCH$(1,2)_r$	47.376	46.419	-88.752	-86.837

Table 1: Log likelihood and AIC of GARCH(p, q) models with or without the m first values of the data. The data was either differenced or the residual of a linear regression denoted with subscript d or r, respectively.

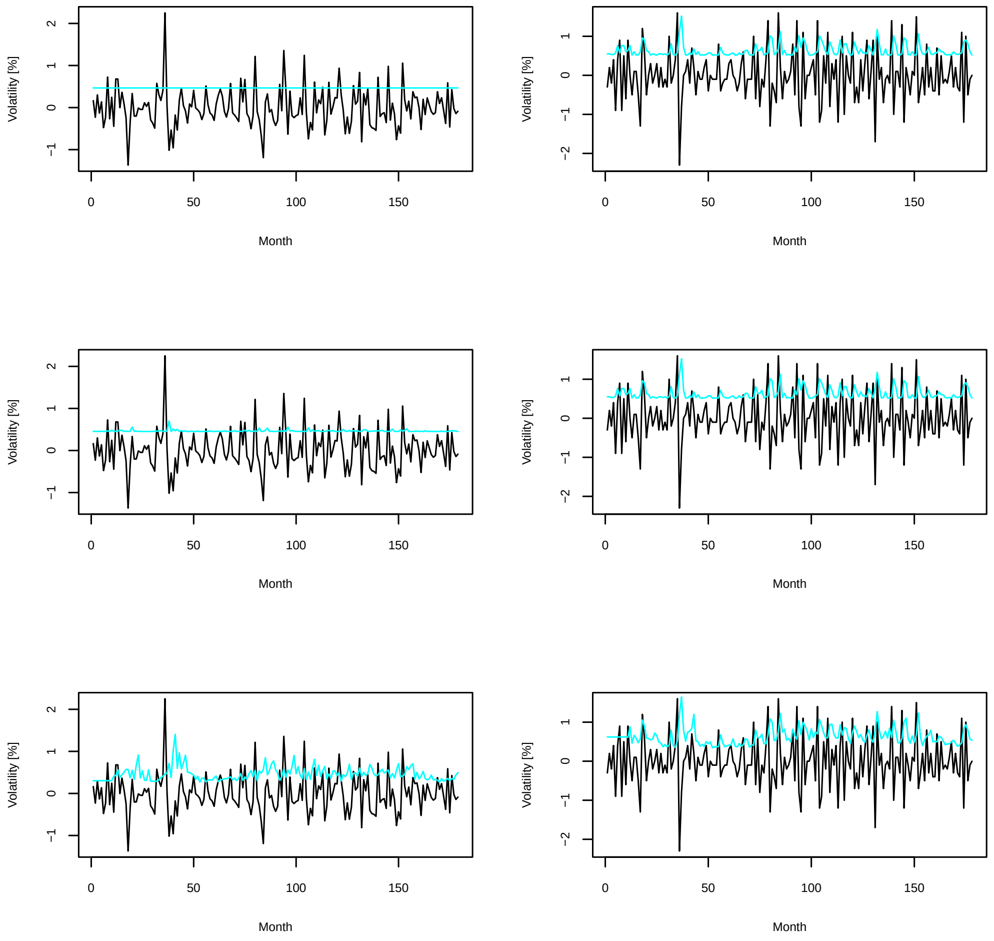

Figure 14: ARCH(p) one-step-ahead forecasts of $\hat{\sigma}_t$ on regression residual (left) and differenced (right) data, where $p = 1, 2, 10$ from top to bottom.

As a prediction test we reduced the dataset by a year and fitted three ARCH models with three different values for p. In Figure 15 we can see the forecasted returns with the real data. The squared residuals where 9.6, 2.6 and 11.2 for ARCH(1), ARCH(2) and ARCH(10), respectively. The code for this section can be found in Appendix H.

G SARIMA implementation

```
setwd("C:\\Users\\celin\\OneDrive\\Dokumenter\\FYSMAT 7.semester\\Tidsrekker\\Project")
df <- read.csv('projectdata.csv', header = T, sep=";", dec=",", stringsAsFactors=FALSE)
head(df)
source("functions.r")

# Run ordinary regression of Y_t on z_t1,z_t2,..., and estimate the betas.
model <- lm(Inflation~InterestRate+Consumption+Bankrupt, data=df)
summary(model)

# Plot inflation rate
par(mfrow=c(1,1))
plot(df$Inflation, type="l", xlab="Month", ylab="Inflation rate [%]", main="Inflation from 2000 to 2014
```

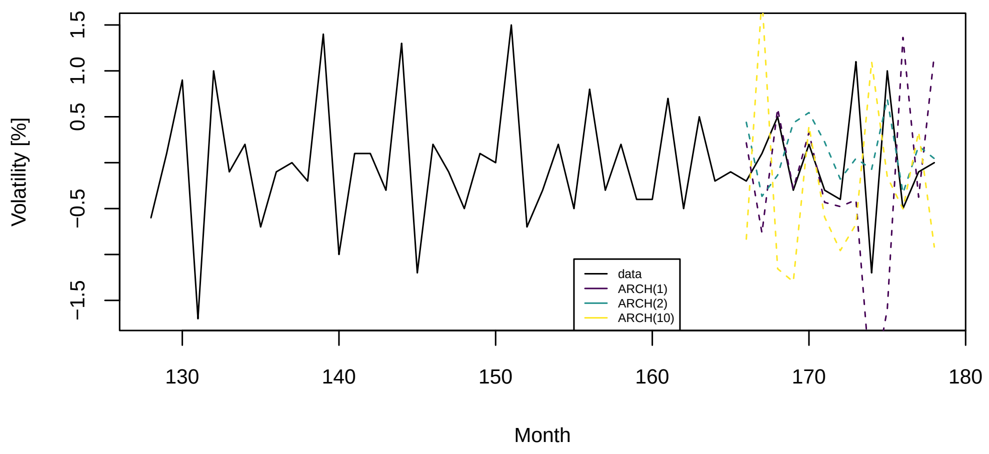

Figure 15: One-step-ahead forecast of the return using differenced data without the last year.

```
# Identify ARMA models using the residuals: x_t = Y_t - sum(beta*z_ti)
x_series <- df$Inflation - model$fitted.values

# Plot residuals
par(mfrow=c(1,1))
year_divider <- c(0:15)*12
plot(c(1:length(x_series)), x_series, type="l", main="Timeseries", xlab="month", ylab="error") + abline

# Need to look at ACF and PACF, to identify ARMA
plot_acf(100,x_series)
PACF(100,x_series)

# SARIMA MODEL 1 (P=1):
# Calculate estimates
initial_params <- c(1) #alpha
{
A <- diag(nrow = 11)
firstrow <- rep(0,11)
A <- rbind(firstrow,A)
lastcol <- rep(0,12)
lastcol[1] <- 1 #alpha
A <- cbind(A,lastcol)

H <- matrix(0,nrow = 12)
H[1] <- 1

B <- matrix(0,ncol = 12)
B[1] <- 1

D <- 0
x <- matrix(0,nrow = 12)
}

alpha_P1 = find_parameters(x,x_series,length(x_series),A,H,B,D,initial_params)
alpha_P1
```

```r
# Insert found parameter (alpha) into matrix A
A[1,12] <- alpha_P1

# Use kalman filter
T = length(x_series)
x_filter_P1 <- kalman_filter(x,x_series,T, A,H,B,D)
plot_kalman(x_filter_P1, x_series, T)

# Analyse standardized residuals
error_P1 = x_filter_P1$innovations / x_filter_P1$sds
diagnostic(error_P1)

# SARIMA MODEL 2 (P=2):
# Calculate estimates
initial_params <- c(1,2) #alpha_12, alpha_24
{
  A <- diag(nrow = 23)
  firstrow <- rep(0,23)
  firstrow[12] <- 1 # alpha_12
  A <- rbind(firstrow,A)
  lastcol <- rep(0,24)
  lastcol[1] <- 2 #alpha_24
  A <- cbind(A,lastcol)

  H <- matrix(0,nrow = 24)
  H[1] <- 1

  B <- matrix(0,ncol = 24)
  B[1] <- 1

  D <- 0
  x <- matrix(0,nrow = 24)
}

alphas <- find_parameters2(x, x_series, T, A, H, B, D, initial_params)
alphas
A[1,12] <- alphas[1]
A[1,24] <- alphas[2]

T = length(x_series)
x_filter_P2 <- kalman_filter(x, x_series, T, A, H, B, D)
plot_kalman(x_filter_P2, x_series, T)

# Analyse standardized residuals
error_P2 = x_filter_P2$innovations / x_filter_P2$sds
diagnostic(error_P2, model = "(0,0,0)x(2,0,0)[12]")


# SARIMA MODEL 3 (differenced):
diff_x <- difference(x_series,lag=12)

# Plotting time series
par(mfrow=c(1,1))
plot(c(1:length(diff_x)), diff_x, type="l", main="Differenced timeseries", xlab="month", ylab="error")

# Look at ACF/PACF to identify SARIMA
plot_acf(100,diff_x)
PACF(100,diff_x)
```

```r
# Calculate estimates
initial_params <- c(1) #alpha_diff
{
  A <- diag(nrow = 11)
  firstrow <- rep(0,11)
  A <- rbind(firstrow,A)
  lastcol <- rep(0,12)
  lastcol[1] <- 1 #alpha_diff
  A <- cbind(A,lastcol)

  H <- matrix(0,nrow = 12)
  H[1] <- 1

  B <- matrix(0,ncol = 12)
  B[1] <- 1

  D <- 0
  x <- matrix(0,nrow = 12)
}
alpha_diff = find_parameters(x,diff_x,length(diff_x),A,H,B,D,initial_params)
alpha_diff
A[1,12] <- alpha_diff

# Calculate new x using kalman filter
T = length(diff_x)
x_filter_diff <- kalman_filter(x, diff_x, T, A, H, B, D)
plot_kalman(x_filter_diff, diff_x,T)

# Analyse standardized residuals
error_diff = x_filter_diff$innovations / x_filter_diff$sds
diagnostic(error_diff, model = "(0,0,0)x(1,1,0)[12]")

# AIC
#Calculating AIC for the models
{
  #SARIMA MODEL 1
  A <- diag(nrow = 11)
  firstrow <- rep(0,11)
  A <- rbind(firstrow,A)
  lastcol <- rep(0,12)
  lastcol[1] <- 0.3167883 #alpha_P1
  A <- cbind(A,lastcol)

  H <- matrix(0,nrow = 12)
  H[1] <- 1

  B <- matrix(0,ncol = 12)
  B[1] <- 1

  D <- 0
  x <- matrix(0,nrow = 12)
}
max_ll_1 <- -ll_sarima1(0.3167883,x_series) #Returns the maximum likelihood
{
  #SARIMA MODEL 3 (differenced)
  A <- diag(nrow = 11)
  firstrow <- rep(0,11)
  A <- rbind(firstrow,A)
  lastcol <- rep(0,12)
  lastcol[1] <- alpha_diff #alpha_P1
  A <- cbind(A,lastcol)

  H <- matrix(0,nrow = 12)
  H[1] <- 1

  B <- matrix(0,ncol = 12)
  B[1] <- 1

  D <- 0
  x <- matrix(0,nrow = 12)
}
max_ll_diff <- -ll_sarima1(alpha_diff,diff_x) #For the differenced model aswell
{
  #SARIMA MODEL 2
  A <- diag(nrow = 23)
  firstrow <- rep(0,23)
  firstrow[12] <- alphas[1] # alpha_12
  A <- rbind(firstrow,A)
  lastcol <- rep(0,24)
  lastcol[1] <- alphas[2] #alpha_24
  A <- cbind(A,lastcol)

  H <- matrix(0,nrow = 24)
  H[1] <- 1

  B <- matrix(0,ncol = 24)
  B[1] <- 1

  D <- 0
  x <- matrix(0,nrow = 24)
}
max_ll_2 <- -ll_sarima2(alphas,x_series) #Returns the maximum likelihood

aic_1 <- -2 * max_ll_1 + 2*1 #One parameter
aic_2 <- -2 * max_ll_2 + 2*2 #Two parameters
aic_3 <- -2 * max_ll_diff + 2*1 #One parameter
aic_1
aic_2
aic_3



#Forecasting the inflation rate one year ahead
#using the SARIMA(0,0,0)x(2,0,0) model
{
  A <- diag(nrow = 23)
  firstrow <- rep(0,23)
  firstrow[12] <- alphas[1]  # alpha_1
  A <- rbind(firstrow,A)
  lastcol <- rep(0,24)
  lastcol[1] <- alphas[2]   # alpha_2
  A <- cbind(A,lastcol)
  H <- matrix(0,nrow = 24)
  H[1] <- 1
  B <- matrix(0,ncol = 24)
  B[1] <- 1

  D <- 0
  x <- matrix(0,nrow = 24)
}
kalman_df <- kalman_filter(x,x_series,length(x_series)+12,A,H,B,D)
forecasted_x <- tail(kalman_df$states,12)
forecasted_x
design_matrix <- matrix(c(df$newInterestRate,df$newConsumption,df$newBankrupt),ncol = 3)
design_matrix <- na.omit(design_matrix)
design_matrix <- cbind(rep(1,12),design_matrix)
forecasted_inflation <- design_matrix %*% model$coefficients + forecasted_x


len <- length(df$Inflation)
par(mfrow = c(1,1))
plot(1:len,df$Inflation,type="l",col="blue",xlim = c(0,195),main = "1 year prediction of the inflation
lines(len:(len+1),c(tail(df$Inflation,n=1),forecasted_inflation[1]),col="red")
lines((len+1):(len+12),forecasted_inflation, col = "red")
lines((len+1):(len+12),na.omit(df$newInflation), col = "green")
legend(x="topright", legend=c("Data", "Predicted inflation", "Actual inflation"), col=c("blue", "red",
```

G.1 ”functions.r” file

```
sample_acf <- function(h,x){
  #Function to calculate the sample autocorrelation function.
  #h: lag to evaluate ACF at
  #x: time series data

  x_mean = mean(x)
  t <- length(x)
  sample_acvf <- 0
  for (i in 1:(t-h)){
    sample_acvf <- sample_acvf + (x[i+h]-x_mean)*(x[i]-x_mean)
  }
  sample_acvf <- sample_acvf * (1/t)

  if (h == 0){
    return (sample_acvf)
  }
  else{
    return (sample_acvf / sample_acf(0,x))
  }
}

plot_acf <- function(h,x, title="Sample autocorrelation"){
  #Function to plot the sample autocorrelation.
  #h: lag to evaluate ACF up to
  #x: time series
  lags <- 0:h
  result <- sapply(lags,sample_acf,x = x)
  result[1] <- result[1] / sample_acf(0,x)
  CI <- c(1.96/sqrt(length(x)), -1.96/sqrt(length(x)))
  plot(lags, result, type="h", main=title, ylab = "ACF", xlab = "Lag", ylim = c(min(CI[2],result),1)) +
    abline(h=0) + abline(h=CI, col=c("blue","blue"), lty=c(2,2))
  return(result)
}

durbin_levinson <- function(h,x){
  #Simplified version of Durbin-Levinson, to calculate coefficients of PACF.
  phi <- matrix(nrow = h+1,ncol = h+1)
  phi[1,1] <- 0
  phi[2,2] <- sample_acf(1,x)

  for (n in 2:(h)){
    numer <- sample_acf(n,x)
    denom <- 1
    for (k in 1:(n-1)){
      if ((n-1) != k){
        phi[n,k+1] <- phi[n-1,k+1] - phi[n,n]*phi[n-1,n-k]
        phi[k+1,n] <- phi[n,k+1]
      }

      numer <- numer - phi[n,k+1]*sample_acf(n-k,x)
      denom <- denom - phi[n,k+1]*sample_acf(k,x)
    }
    phi[n+1,n+1] <- as.numeric(numer/denom)
  }
  return (diag(phi)[-1])
}


PACF <- function(h,x, title="Sample partial autocorrelation"){
  #Computes and plots the partial autocorrelation function with a 95% confidence interval.
  #h: lag to compute PACF up to.
  #x: time series

  coeffs <- durbin_levinson(h,x)
  lower_lim <- -1.96/sqrt(length(x))
  upper_lim <- 1.96/sqrt(length(x))

  plot(1:h,coeffs,xlab = "LAG",ylab = "PACF",type="h", main=title)
  #points(1:h,coeffs, xlab="LAG", pch=16,col="red", type = "p")
  abline(h = lower_lim,col = "blue",lty = "dashed")
  abline(h = upper_lim,col = "blue", lty = "dashed")
  abline(h=0)
}


difference <- function(data, lag = 12){
  # Function for differencing timeseries
  len <- length(data)
  diff_data <- vector("numeric",length=len-lag)

  for (i in lag:len) {
    diff_data[i-lag] <- data[i] - data[i-lag]
  }
  return(diff_data)
}
```

```
kalman_filter <- function(x,y,T,A,H,B,D){
  #Function for computing the Kalman filter, either with filtering, smoothing or forecasting.

  #Initial conditions
  s <- length(y)
  states <- vector("numeric",length = T)
  sds <- vector("numeric",length = T)
  innovations <- vector("numeric",length = T)

  S <- diag(x = dim(A)[1])*var(y) #Initial covariance matrix

  #Filtering
  for (t in 1:(s)){
    states[t] <- B %*% x
    #Innovations
    I <- y[t] - states[t]
    I <- as.numeric(I)
    innovations[t] <- I

    #Projection matrix (Kalman gain)
    C <- (B %*% S) %*% t(B) + D %*% t(D)
    C <- as.numeric(C)
    sds[t] <- sqrt(C)

    M <- (S %*% t(B)) %*% solve(C)

    #Condition on innovation/observation
    x_tt <- x + M %*% I

    #Update x,S
    x <- A %*% x_tt

    S <- (diag(dim(A)[1]) - M %*% B) %*% S
    S <- A %*% S %*% t(A) + H %*% t(H)
  }

  #Prediction
  if (T > s){
    for (t in s:(T)){
      x <- A %*% x
      S <- A %*% S %*% t(A) + H %*% t(H)
      states[t] <- B %*% x

      C <- (B %*% S) %*% t(B) + D %*% t(D)
      C <- as.numeric(C)
      sds[t] <- sqrt(C)
      innovations[t] <- B%*%x
    }
  }

  return(data.frame("states" = states,"sds" = sds,"innovations" = innovations))
}

ll_sarima1 <- function(params,y){
    #Function to calculate the likelihood of the first SARIMA model. (used for AIC)
  #Initial conditions
  #Change according to model
  A[1,dim(A)[2]] <- params[1]
  s <- length(y)
  S <- diag(x = dim(A)[1])*var(y) #Initial covariance matrix

  ll <- 0
  #Filtering
  for (t in 1:(s)){
    #Innovations
    I <- y[t] - B %*% x
    I <- as.numeric(I)

    #Projection matrix (Kalman gain)
    C <- (B %*% S) %*% t(B) + D %*% t(D)
    C <- as.numeric(C)

    M <- (S %*% t(B)) %*% solve(C)

    #Condition on innovation/observation
    x_tt <- x + M %*% I

    #Update x,S
    x <- A %*% x_tt

    S <- (diag(dim(A)[1]) - M %*% B) %*% S
    S <- A %*% S %*% t(A) + H %*% t(H)

    ll <- ll + 0.5*(log(abs(C)) + I^2/C)
  }
  return (ll)
}

ll_sarima2 <- function(params,y){
    #Function to calculate the likelihood of the second SARIMA model. (used for AIC)
  #Initial conditions
  #Change according to model
  A[1,dim(A)[2]] <- params[1]

  s <- length(y)
  S <- diag(x = dim(A)[1])*var(y) #Initial covariance matrix

  ll <- 0
  #Filtering
  for (t in 1:(s)){
    #Innovations
    I <- y[t] - B %*% x
    I <- as.numeric(I)

    #Projection matrix (Kalman gain)
    C <- (B %*% S) %*% t(B) + D %*% t(D)
    C <- as.numeric(C)

    M <- (S %*% t(B)) %*% solve(C)

    #Condition on innovation/observation
    x_tt <- x + M %*% I

    #Update x,S
    x <- A %*% x_tt
    S <- (diag(dim(A)[1]) - M %*% B) %*% S
    S <- A %*% S %*% t(A) + H %*% t(H)

    ll <- ll + 0.5*(log(abs(C)) + I^2/C)
  }
  return (ll)
}

find_parameters <- function(x,y,T,A,H,B,D,initial_params){
  #Function to estimate unknown parameters in SARIMA-model.
  likelihood <- function(params){
    #Initial conditions
    #Change according to model
    A[1,dim(A)[2]] <- params[2]
    A[1,12] <- params[1]

    s <- length(y)
    S <- diag(x = dim(A)[1])*var(y) #Initial covariance matrix

    ll <- 0
    #Filtering
    for (t in 1:(s)){
      #Innovations
      I <- y[t] - B %*% x
      I <- as.numeric(I)

      #Projection matrix (Kalman gain)
      C <- (B %*% S) %*% t(B) + D %*% t(D)
      C <- as.numeric(C)

      M <- (S %*% t(B)) %*% solve(C)

      #Condition on innovation/observation
      x_tt <- x + M %*% I

      #Update x,S
      x <- A %*% x_tt

      S <- (diag(dim(A)[1]) - M %*% B) %*% S
      S <- A %*% S %*% t(A) + H %*% t(H)

      ll <- ll + 0.5*(log(abs(C)) + I^2/C)
    }
    return (ll)
  }

  new_params <- optim(initial_params,likelihood,method = "BFGS",control = list(maxit = 300))$par
  return (new_params)
}



find_parameters2 <- function(x,y,T,A,H,B,D,initial_params){
  #Function to estimate unknown parameters in SARIMA-model.
  likelihood <- function(params){
    #Initial conditions
    #Change according to model
```

```r
    A[1,dim(A)[2]] <- params[2]
    A[1,12] <- params[1]

    s <- length(y)
    S <- diag(x = dim(A)[1])*var(y) #Initial covariance matrix

    ll <- 0
    #Filtering
    for (t in 1:(s)){
      #Innovations
      I <- y[t] - B %*% x
      I <- as.numeric(I)

      #Projection matrix (Kalman gain)
      C <- (B %*% S) %*% t(B) + D %*% t(D)
      C <- as.numeric(C)

      M <- (S %*% t(B)) %*% solve(C)

      #Condition on innovation/observation
      x_tt <- x + M %*% I

      #Update x,S
      x <- A %*% x_tt

      S <- (diag(dim(A)[1]) - M %*% B) %*% S
      S <- A %*% S %*% t(A) + H %*% t(H)

      ll <- ll + 0.5*(log(abs(C)) + I^2/C)
    }
    return (ll)
  }

  new_params <- optim(initial_params,likelihood,method = "BFGS",control = list(maxit = 300,reltol=1e-8)
  return (new_params)
}



diagnostic <- function(x, model = "(0,0,0)x(1,0,0)[12]"){
  # Function for making diagnostic plots for SARIMA model
  layout(matrix(c(1,1,1,2,3,4), 2, 3, byrow = TRUE))
  plot(x, type="l", xlab="Month", ylab="", main="Standardized residuals")
  qqnorm(x, main="Normal Q-Q Plot of Std.Res.")
  qqline(x)
  hist(x, main="Histogram of Std.Res.")
  plot_acf(x, h=20, title="ACF of Std.Res.")
  main_t <- paste("Model: ", model)
  title(main_t, outer=T, line=-3, adj=0.1)
}



plot_kalman <- function(data, x, T){
  # Function for plotting filtering and prediction of time series returned by the kalman filter, as wel
  upper <- data$innovations+1.96*data$sds
  lower <- data$innovations-1.96*data$sds
  par(mfrow=c(1,1))
  plot(c(1:length(x)), x, type="l", ylim=c(-10,10), xlim=c(1,T), xlab="Month", ylab="Error", main="Kalm
  lines(x=1:T, y=data$innovations, col="red")
  lines(x=1:T, y=upper, col="gray", lty=2)
  lines(x=1:T, y=lower, col="gray", lty=2)
  legend(x="topright", legend=c("Data", "Filter", "95% CI"), col=c("black", "red", "gray"), lty=c(1,1,2
}

\end{lstlisting}

\subsubsection{"garch.r" file}
\begin{lstlisting}
# Finds parameters in ARCH(1) by maximum likelihood estimation
find_parameters_garch <- function(x, initial_params){
  likelihood_garch <- function(params){
    alpha0 <- exp(params[1])
    alpha1 <- exp(params[2])

    if (alpha0 <= 0 || alpha1 <= 0) {
      return(Inf)
    }

    ll <- 0
    for (i in 2:length(x)){
      variance_term <- alpha0 + alpha1 * x[i-1]^2
      ll <- ll + 0.5 * (log(variance_term) + x[i]^2 / variance_term)
    }
    return(ll)
  }

  # Use logs of parameters for optimization to ensure positivity
  initial_params_log <- log(initial_params)
  new_params_log <- optim(initial_params_log, likelihood_garch, method = "BFGS", control = list(maxit =

  # Transform parameters back to original scale
  new_params <- exp(new_params_log)
  return(new_params)
}


# ARCH(1)
x_series <- df$Inflation
# rt inflation data differenced
r <- rep(0, length(x_series)-1)
for (i in 2:length(x_series)){
  r[i-1] <- (x_series[i] - x_series[i-1])
}
# Finding estimated alpha's
initial_params_garch <- c(0.5,0.5)
alpha_garch <- find_parameters_garch(r, initial_params_garch)
alpha_garch

# Simulate the series using the estimated parameters
sigma_garch <- rep(0, length(r)-1)
for (i in 2:length(sigma_garch)){
  sigma_garch[i] <- sqrt(alpha_garch[1] + alpha_garch[2]*r[i-1]^2)
}
sigma_garch[1] <- sigma_garch[2]
```

```
# Plot r_t and the volatility magnitude (not squared...) to see the trend more clearly
par(mfrow=c(1,2))
plot(1:length(r),r, type = "l", col = "blue", main = " Volatility of the differenced inflation rate", x
     xlim = c(0,175), cex.main=0.8, cex.axis = 0.8, cex.lab = 0.8)
lines(1:length(sigma_garch), sigma_garch, col = "red")
legend(0.1, 1.8, legend=c("Volatility s", "r"),
       col=c("red", "blue"), lty=1:1, cex=0.5)
# QQ-plot for rt
qqnorm(r, main = "Normal Q-Q plot for r(t)")
qqline(r)


# Forecasting volatility
set.seed(1)
# Dataset to predict from
x.pred <- na.omit(df$newInflation)[1:12]
r.pred <- rep(0, length(x.pred))
r.pred[1] <- r[length(r)]
sigma.pred <- rep(0,length(x.pred))
# Predicting unknown r_t+1 and volatility
for (i in 2:length(r.pred)){
  r.pred[i] <- rnorm(1,sd = sqrt(alpha_garch[1] + alpha_garch[2]*r.pred[i-1]^2))
  sigma.pred[i] <- sqrt(alpha_garch[1] + alpha_garch[2]*r.pred[i-1]^2)
}

# The actual r_t and volatility in the testing set newInlation
x.actual <- na.omit(df$newInflation)
r.actual <- rep(0, length(x.actual))
sigma.actual <- rep(0,length(x.actual))
sigma.actual[1] <- sigma_garch[length(sigma_garch)]
r.actual[1] <- r[length(r)]
sigma.pred[1] <- sigma_garch[length(sigma_garch)]

# Simulating the actual r_t and volatility in the test set
for (i in 2:length(x.actual)){
  r.actual[i] <- (x.actual[i] - x.actual[i-1])
  sigma.actual[i] <- sqrt(alpha_garch[1] + alpha_garch[2]*r.actual[i-1]^2)
}

# Plotting the prediction with the actual volatility
par(mfrow=c(1,1))
plot(1:length(sigma_garch),sigma_garch^2, type = "l", col = "blue", main = "1 year prediction of volati
     xlim = c(0,195), cex.main=0.8, cex.axis = 0.8, cex.lab = 0.8)
# x_values tells 'lines' where to place the test set - at the end
x_values <- seq(length(sigma_garch), length.out = length(r.pred), by = 1)
lines(x_values, sigma.pred^2, col = "red")
lines(x_values, sigma.actual^2, col = "green")
legend(0.1, 2.2, legend=c("actual s^2", "predicted s^2", "actual s^2"),
       col=c("green", "red", "blue"), lty=1:1:1, cex=0.5)
```

H GARCH(p,q) implementation

```
    #!/usr/bin/Rscript

library(viridis)
```

```
figPath = "./figures/"

# Garch(p,q) implementation ----
garchLL = function(params, r, p=1, q=0){
  # LogLike of alpha and beta given the first r_t values t=(1,..., max(p,q))(Eq. 5.46)
  # Defaults to GARCH(1,1) and uses only the first three params (alpha0, alpha1, beta1)
  # params: vector of log(parameters). First p+1 are alphas, rest are betas
  # r: vector of returns (Eq. 5.34)
  # p,q: GARCH(p,q) model parameters
  # -> returns -log likelihood

  n = length(r)
  m = max(p,q)

  alpha = exp(params[1:(p+1)])
  if (q==0){beta = 0}
  else {beta = exp(params[-(1:(p+1))]) }

  # initialize first m values of the variance
  sigma_sq = numeric(n)
  sigma_sq[1:m] = t(alpha) %*% c(1,r[p:1]^2) # should they be zero?

  # Iteratively compute each variance
  for (t in (m+1):n){
    sigma_sq[t] = sum(alpha * c(1,r[(t-1):(t-p)]^2)) + sum(beta*sigma_sq[(t-1):(t-q)])
  }

  return(sum(log(sigma_sq) + r^2/sigma_sq))
}


sigmaForecast = function(mod){
  # One-step-ahead forecasts of the volatility sigma (Eq. 5.52)
  # mod: list of estimated parameters and the data it was fitted on.
  # -> return one-step-ahead forecasts

  # Extract values for readability.
  n = mod$n
  m = mod$m
  p = mod$p
  r = mod$r

  alpha = mod$estim[1:(p+1)]
  if (mod$q==0){beta = 0; q=1} # If ARCH(p) model
  else{beta = mod$estim[-(1:(p+1))]; q = mod$q}

  sf = numeric(n) # sigma squared forecasts
  sf[1:m] = sum(alpha*c(1,r[1:p]^2)) # first m values

  for (t in (m+1):n){
    sf[t] = sum(alpha * c(1,r[(t-1):(t-p)]^2)) + sum(beta*sf[(t-1):(t-q)])
  }
  return(sf)
}


Garch = function(r, p=1, q=0, init=c(0.1,0.1)){
  # Make, estimate and forecast volatility of a GARCH(p,q) given the parameters
```

```r
  # r: the returns
  # p,q: GARCH(p,q)
  # init: initial parameters for alphas and betas

  n = length(r)
  rq = q

  alpha = init[1:(p+1)]
  if (q==0){beta = 0; q=1} # If ARCH(p) model
  else{beta = init[-(1:(p+1))]; q = q}

  estim = exp(optim(par = log(init), fn = garchLL, r = r, p=p, q=q, method = "BFGS")$par)

  mod = list(
    p = p,
    q = rq,
    n = n,
    m = max(p,q),
    init = init,
    estim = estim,
    r=r
  )

  mod$sigmaForecast = sigmaForecast(mod)
  return(mod)

}


# Test functions ----
testModel = function(mod){
  # Tests model compared to sampled data given real values of the alphas and the
  # betas.

  n = mod$n
  m = mod$m
  p = mod$p

  alpha = mod$paramSim[1:(p+1)]
  if (mod$q==0){beta = 0; q=1}
  else{beta = mod$paramSim[-(1:(p+1))]; q = mod$q}

  r = numeric(n)
  r[1:m] = rnorm(m)
  sigma_sq = numeric(n)
  sigma_sq[1:m] = sum(alpha*c(1,r[1:p]^2))

  for (t in (m + 1):n){
    sigma_sq[t] = sum(alpha*c(1,r[(t-1):(t-p)]^2)) + sum(beta*sigma_sq[(t-1):(t-q)])
    r[t] = rnorm(1, sd=sqrt(sigma_sq[t]))
  }

  # estimation
  estim = exp(optim(par = log(mod$init), fn = garchLL, r = r, p=p, q=q, method = "BFGS")$par)
  comparison = (rbind(real = mod$paramSim, estim = estim, init = mod$init))
  return(list(r = r, estim = estim, comparison = comparison))
}
```

```r
# Convenience functions ----
modResults = function(mod, main="", plot=T){
  if (plot){
    plot(mod$r, type="l", main=main, xlab = "Month",
         ylab = "Volatility [%]", cex.main=0.8, cex.axis = 0.8, cex.lab = 0.8)
    lines(sqrt(mod$sigmaForecast), col="cyan")
  }
}

# Tests ----
set.seed(420)
# ARCH(1) Test
arch1 = list(
  p = 1,
  q= 0,
  m= 1,
  n = 1e4,
  init = rep(0.1,2),
  paramSim = c(0.01, 0.2)
)


testResult = testModel(arch1)
testResult$comparison

# GARCH(1,2) test
garch12 = list(
  p = 1,
  q= 2,
  m= 2,
  n = 1e4,
  init = rep(0.1,4),
  paramSim = c(0.01, 0.2, 0.1, 0.5)
)


testResult = testModel(garch12)
garch12$estim = testResult$estim
round(testResult$comparison,3)


# Real data ----
df <- read.csv('projectdata.csv', header = T, sep=";", dec=",", stringsAsFactors=FALSE)

# fit GARCH(1,2) and plot results
fit = lm(Inflation~Unemployed+Consumption+InterestRate, data = df)
r = df$Inflation - fit$fitted.values
d = diff(df$Inflation)
par(mfrow=c(2,1))

## inflation - regression
garch12regr = Garch(r, p=1, q=2, init=rep(0.1, 1+2+1))
modResults(garch12regr, main="garch(1,2) on regression")

## diff inflation
garch12 = Garch(d, p=1, q=2, init=rep(0.1, 1+2+1))
modResults(garch12, "garch(1,2) on diff")
```

```r
par(mfrow=c(1,1))

# fit several ARCH(p)
arch1regr = Garch(r, p=1, q=0, init=rep(0.1, 2))
arch1d = Garch(d, p=1, q=0, init=rep(0.1, 2))
arch2regr = Garch(r, p=2, q=0, init=rep(0.1, 3))
arch2d = Garch(d, p=2, q=0, init=rep(0.1, 3))
arch10regr = Garch(r, p=10, q=0, init=rep(0.1, 11))
arch10d = Garch(d, p=10, q=0, init=rep(0.1, 11))

## plot
par(mfrow=c(3,2))
modResults(arch1regr, main="arch(1) on inflation - regression")
modResults(arch1d, main = "arch(1) on diff(inflation)")
modResults(arch2regr, main="arch(2) on inflation - regression")
modResults(arch2d, "arch(2) on diff(inflation)")
modResults(arch10regr, main="arch(10) on inflation - regression")
modResults(arch10d, "arch(10) on diff(inflation)")
par(mfrow=c(1,1))

if (T){
  pdf(file = paste0(figPath, "garchVolatility.pdf"))
  par(mfrow=c(3,2))
  modResults(arch1regr, main="")
  modResults(arch1d, main = "")
  modResults(arch2regr, main="")
  modResults(arch2d, "")
  modResults(arch10regr, main="")
  modResults(arch10d, "")
  par(mfrow=c(1,1))
  dev.off()
}

## compare parameter estimates
list(
  arch1regr = arch1regr$estim,
  arch1d = arch1d$estim,
  arch2regr = arch2regr$estim,
  arch2d = arch2d$estim,
  arch10regr = arch10regr$estim,
  arch10d = arch10d$estim
)

models = list(
  arch1d = arch1d,
  arch2d = arch2d,
  arch10d = arch10d,
  arch1regr = arch1regr,
  arch2regr = arch2regr,
  arch10regr = arch10regr,
  garch12regr = garch12regr,
  garch12 = garch12
)


# Model selection ----
logLike = function(mod, m=1){
  i = (m+1):(mod$n) # can burn m first values
  ll = sum(-0.5*(log(mod$sigmaForecast[i]) + mod$r[i]^2/mod$sigmaForecast[i]))
  return(ll)
}


aic = function(mod, m=1){
  return(2*(mod$p + mod$q) -2*logLike(mod, m))
}

loglikes = numeric(length(models))
aics = numeric(length(models))
loglikesM = numeric(length(models))
aicsM = numeric(length(models))

for (i in 1:length(models)){
  loglikes[i] = logLike(models[[i]])
  aics[i] = aic(models[[i]])
  loglikesM[i] = logLike(models[[i]], models[[i]]$m)
  aicsM[i] = aic(models[[i]], models[[i]]$m)
  # print(sum(models[[i]]£r^2/models[[i]]£sigmaForecast)) # Interesting!
}

modSel = cbind(ll = loglikes, llM = loglikesM, aic = aics, aicM = aicsM)
rownames(modSel) = names(models)
modSel

plot(arch1d$r^2, type = "l")
lines(arch1d$sigmaForecast, col="cyan")
sum(arch1d$r^2/arch1d$sigmaForecast)
length(arch1d$sigmaForecast)


# return forecast ----
forecast = function(mod, n){
  m = mod$m
  p = mod$p
  r = mod$r
  nTot = mod$n + n

  alpha = mod$estim[1:(p+1)]
  if (mod$q==0){beta = 0; q=1} # If ARCH(p) model
  else{beta = mod$estim[-(1:(p+1))]; q = mod$q}

  rf = r
  length(rf) = nTot
  sf = mod$sigmaForecast
  length(sf) = nTot
  for (t in mod$n:(nTot)){
    sf[t+1] = sum(alpha * c(1,rf[(t+1-1):(t+1-p)]^2)) + sum(beta*sf[(t+1-1):(t+1-q)])
    rf[t+1] = rnorm(1, sd = sqrt(sf[t+1]))
  }
  return(rf)
}

plotForecast = function(models){
  c = viridis_pal(option = "D")(length(models))

  for (i in 1:length(models)){
    nTot = length(models[[i]]$forecast)
    w = models[[i]]$w
    lines((nTot-without):nTot, models[[i]]$forecast[(nTot - without):nTot], col=c[i], lty = 2)
  }
  return(c)
}

set.seed(420)
without = 12
dTot = diff(df$Inflation)
nTot = length(dTot)
d = diff(dTot[1:(nTot-without)])


arch1df = Garch(d, p=1, q=0, init=rep(0.1, 2))

arch2df = Garch(d, p=2, q=0, init=rep(0.1, 3))

arch10df = Garch(d, p=10, q=0, init=rep(0.1, 11))

modelsf = list(
  arch1df = arch1df,
  arch2df = arch2df,
  arch10df = arch10df
)



# for (mod in modelsf){
for (i in 1:length(modelsf)){
  # browser()
  w = 12
  modelsf[[i]]$w = w
  # mod£w = w
  modelsf[[i]]$forecast = forecast(modelsf[[i]], w)
}


back = 50
plot((nTot-back):nTot,dTot[(nTot-back):nTot], type = "l", xlab = "Month", ylab = "Volatility [%]", cex.
c = plotForecast(modelsf)
legend(160, -1, legend=c("data", "ARCH(1)", "ARCH(2)", "ARCH(10)"),
       col=c("#000000", c), lty=1:1:1, cex=0.5)


pdf(file = paste0(figPath, "garchForecasts.pdf"), height = 4)
plot((nTot-back):nTot,dTot[(nTot-back):nTot], type = "l", xlab = "Month", ylab = "Volatility [%]", cex.
c = plotForecast(modelsf)
legend(155, -1.05, legend=c("data", "ARCH(1)", "ARCH(2)", "ARCH(10)"),
       col=c("#000000", c), lty=1:1:1, cex=0.5)
dev.off()

forecastRes = numeric(length(modelsf))
for (i in 1:length(modelsf)){
  w = modelsf[[i]]$w
  forecastRes[i] = sum(dTot[(nTot-w):nTot] - modelsf[[i]]$forecast[(nTot-w):nTot])

}
```

```
forecastRes^2
# 9.590208  2.636010 11.154583
```