TMA4285 Time Series Models, Project 2

Kristin Benedicte Bakka and Gunhild Elisabeth Berget

November 27, 2016

Problem 1

In this problem models are fitted to two observed time series. The time series used are **tree rings** and **houseprices**. First the time series **tree rings** is considered. This is data of the thickness of tree rings in California.

A general ARIMA(p,d,q) model is given by

$$(1 - \phi_1 B - \cdots - \phi_p B^p)(1 - B)^d Z_t = \theta_0 + (1 - \theta_1 B - \cdots - \theta_q B^q) a_t$$

where a_t is white noise. When fitting a model to time series, we want to find the parameters in this general model.

Fitting of the time series Tree Rings

The first step of fitting a model to a time series is to plot the series. Figure 1 shows a plot of the series tree rings.

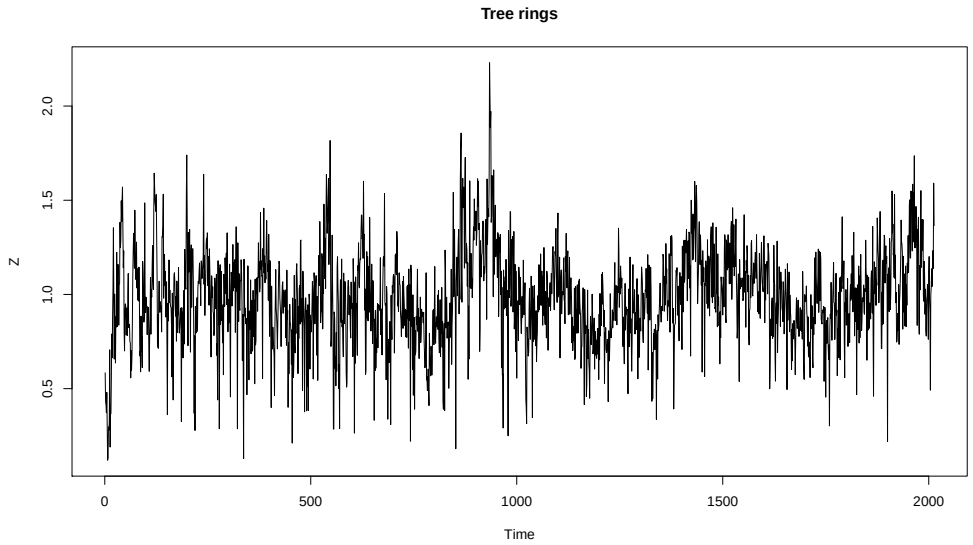

Figure 1: Plot of the time series tree rings

Looking at the plot, the series seems to fluctuate around a stationary mean at about 1. Hence, no differencing is needed and it can be assumed that $d = 0$ in the ARIMA-model. The variance also seems to be stationary so no variance-stabilizing transformation is needed.

The sample autocorrelation function and the sample partial autocorrelation function of the series was calculated and plotted using the R-functions **acf** and **pacf**. The results are shown in Figure 2.

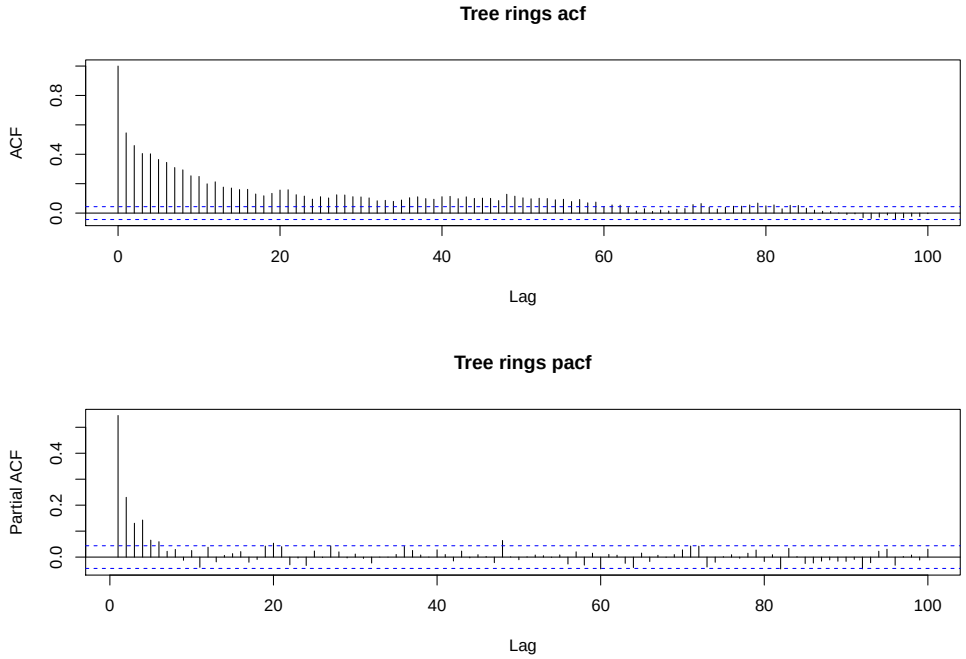

Figure 2: Sample Acf and sample pacf of the time series tree rings

From Figure 2 it can be seen that neither the ACF nor the PACF has a clear cut off, suggesting that the process is a mixed ARMA(p,q). To determine the parameters p and q the extended sample autocorrelation function (eascf) can be studied. The table below shows the asymptotic eascf for tree rings obtained by the R-function **eacf**. The table shows wether the values for the eascf is significant or not (X represents significant and 0 is nonsignificant). The values for p and q can then be found by finding the vertix of a triangle of zeros in this table.

```
> eacf(Z_tr)
```

```
AR/MA
  0 1 2 3 4 5 6 7 8 9 10 11 12 13
0 x x x x x x x x x x x  x  x  x
1 x o x o o o o o o x x  x  o  o
2 x x x o o o o o o o o  o  o  o
3 x x x x o o o o o o o  o  o  o
4 x x x x o o o o o o o  o  o  o
5 x x x o x o o o o o o  o  o  o
6 x x x x x o o o o o o  o  o  o
7 x x x o x o o o o o o  o  o  o
```

Looking at the table the suggested values are $p = 2$ and $q = 3$ leading to an ARMA(2,3)-model.

The R-function **auto.arima()** was also used to fit the model.

```
> auto.arima(Z_tr)
```

```
Series: Z_tr
ARIMA(3,0,2) with non-zero mean
```

```
Coefficients:
         ar1     ar2      ar3     ma1      ma2  intercept
      0.1954  0.8461  -0.1425  0.1572  -0.6400     0.9797
s.e.  0.0750  0.0556   0.0353  0.0712   0.0492     0.0228

sigma^2 estimated as 0.0406:  log likelihood=371.26
AIC=-728.51   AICc=-728.46   BIC=-689.26
```

This function suggest an ARMA(3,2) model. Hence, these two models (ARMA(3,2) and ARMA(2,3)) are fitted to the time series and residuals are calculated.

```
> detach("package:TSA")
> library(stats)
> n_tr <- length(Z_tr)
> fitTR1 <- arima(Z_tr, c(3,0,2), transform.pars=FALSE, xreg = 1:n_tr)
> tsdiag(fitTR1,20)
```

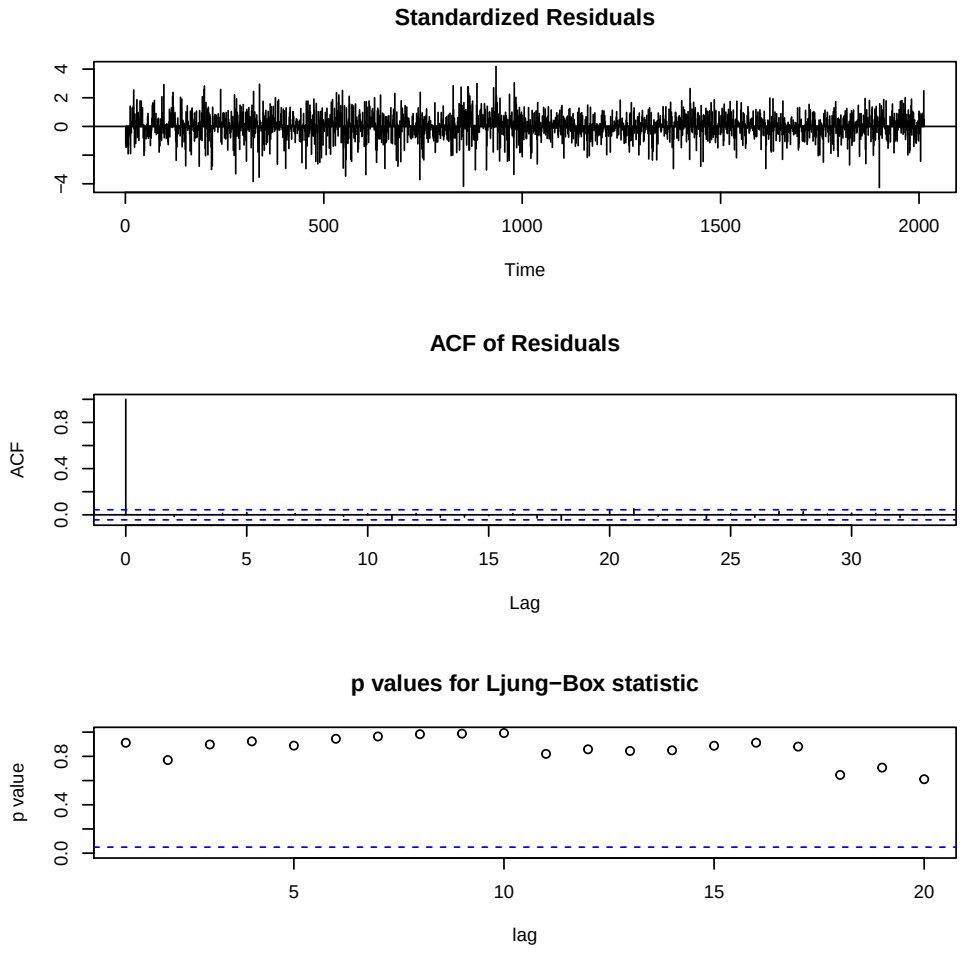

Figure 3: Diagnostics after fitting the time series to an ARMA(3,2)-model.

```
> fitTR2 <- arima(Z_tr, c(2,0,3), transform.pars=FALSE, xreg = 1:n_tr)
> tsdiag(fitTR2,20)
```

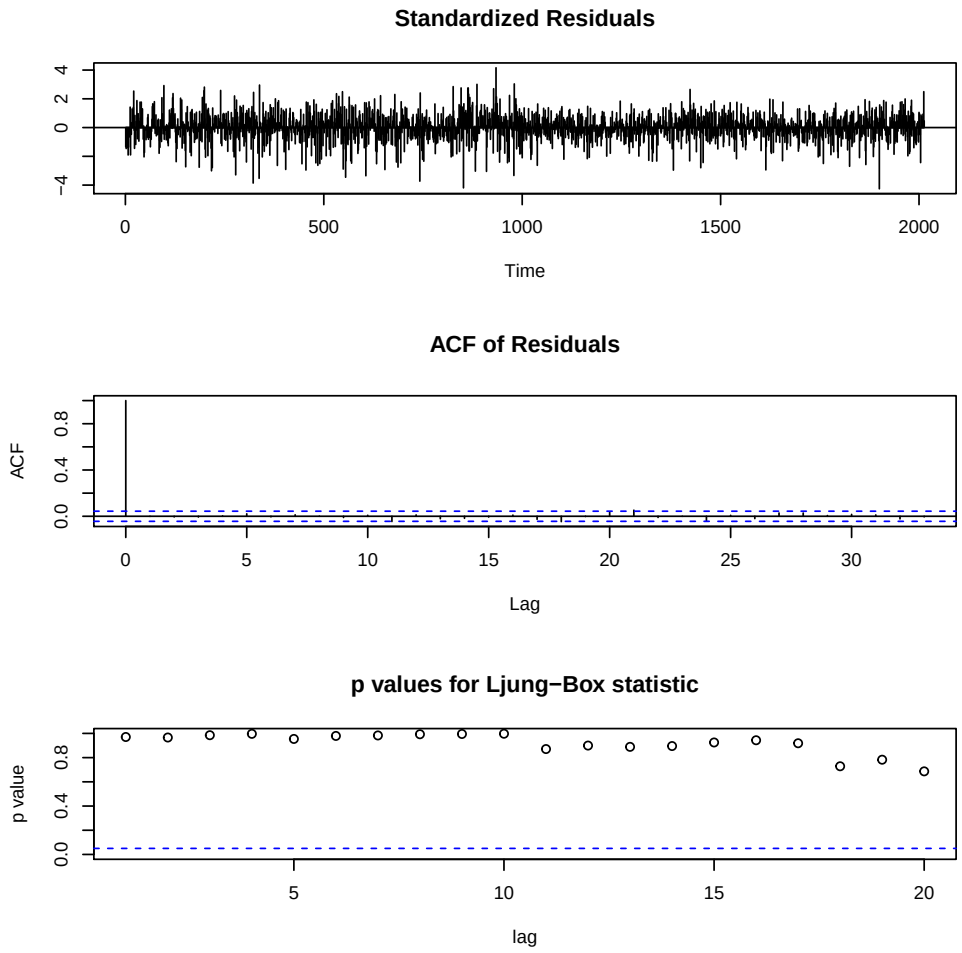

Figure 4: Diagnostics after fitting the time series to an ARMA(2,3)-model.

The residuals in both Figure 3 and Figure 4 looks like white noise and the ACF also shows that the residuals seems independent indicating that the chosen models both makes a good fit. This also indicates that the assumptions of stationarity are correct. From the plot of the p-values it seems like the ARMA(2,3)-model gives slightly higer values and hence this model is chosen for the forecasting.

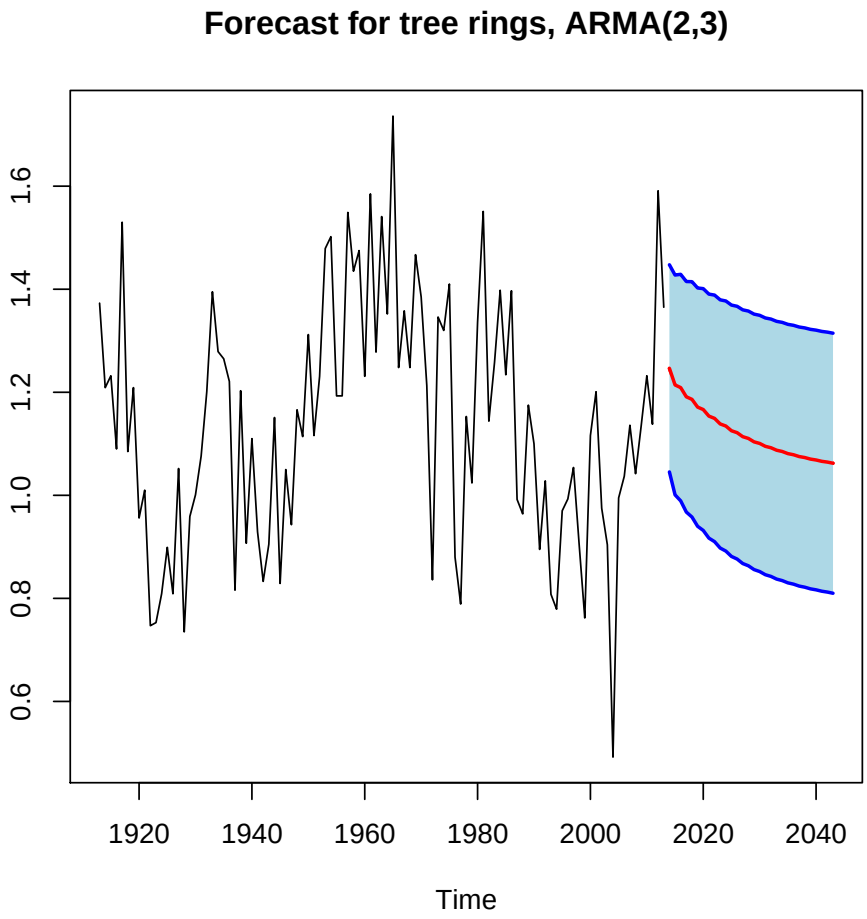

Figure 5: 30 forecasted values using the ARMA(2,3)-model.

Fitting of the time series Houseprices

In this section we will analyze the Quartely houseprices (per square meter relative to the level in year 2000) in Oslo from the first quarter of 1992 until the third quarter of 2015. The seasonality of the Norwegian house prices is well known, so a season of length 4 observations is expected. In addition the Norwegian market has increased steadily in the period, and the household incomes have increased along with them. Norwegian households tend to prioritize owning their own home, so a growing price is expected. When something negative happens in the economy a short fall in houseprices is also to be expected. Whether extreme events affect the market prices in the form of an auto regressive or a moving average term is unclear. If they affect the prices as a moving average term the shocks affect the subsequent prices directly but soon are of no consequence. If extreme events affect the prices as an auto regressive term the effects would last infinetly long into the future. How much something is sold for is not determined by its true inherent value, but by how much people (who can afford it) are willing to pay for it. A commonly recognised psycological statement is that someones long term happiness is only marginally affected by extreme events like being crippled in a car accident or winning the lottery. In that line one could argue that the house prices wolud act like a moving average process. On the other hand it could be reasonable to suggest that peoples long term money spending is affected by prior losses or gains, such that extreme events would affect the market long term. Now to the data.

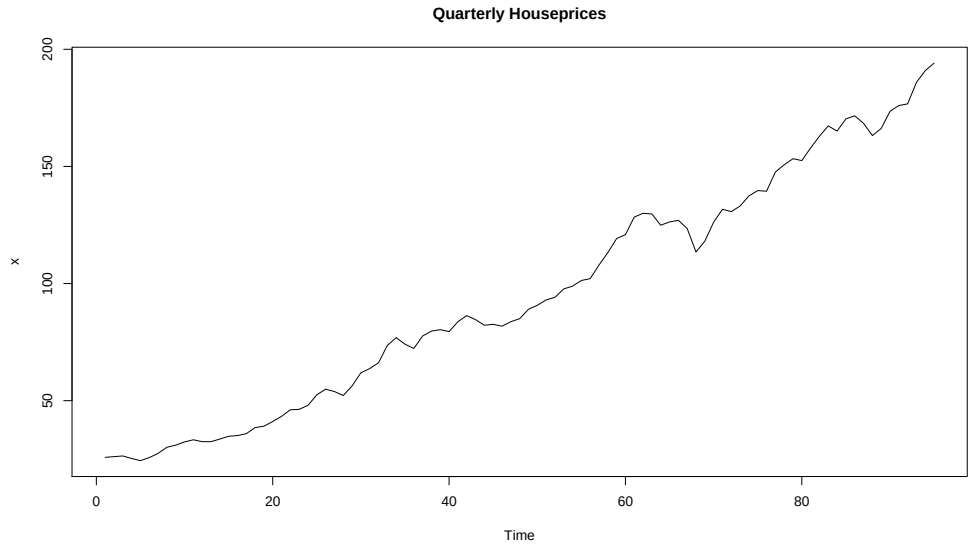

Figure 6: Raw data for Quartely houseprices (per square meter relative to the level in year 2000) in Oslo from the first quarter of 1992 until the third quarter of 2015.

Figure 6 displays an increasing term that looks approximately constant. As the increase is so strong it is difficult to see anything else from the data, although it looks like the variance might be changing. From only seeing the time series itself the conclution is to apply differencing. First we choose to plot the sample acf and partial acf.

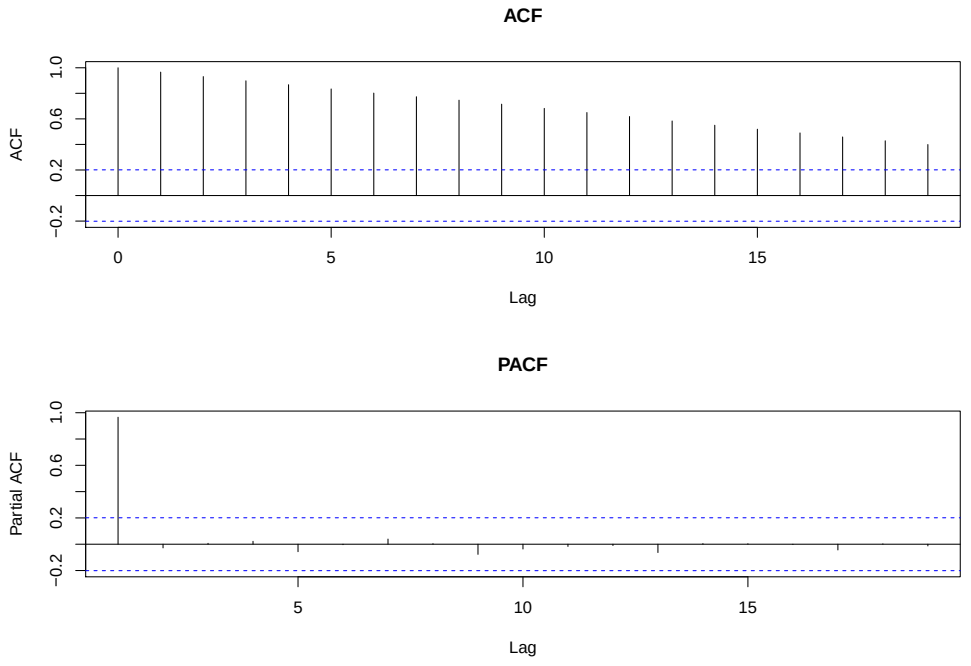

Figure 7: Auto correlation function and partial autocorrelation function for the raw data.

Figure 7 displays a slowly deeclining auto corelation function along with a partial acf that cuts off after lag

1. This alone would also have been enough to suggest differencing. Now the differenced series is plotted along with its acf and pacf.

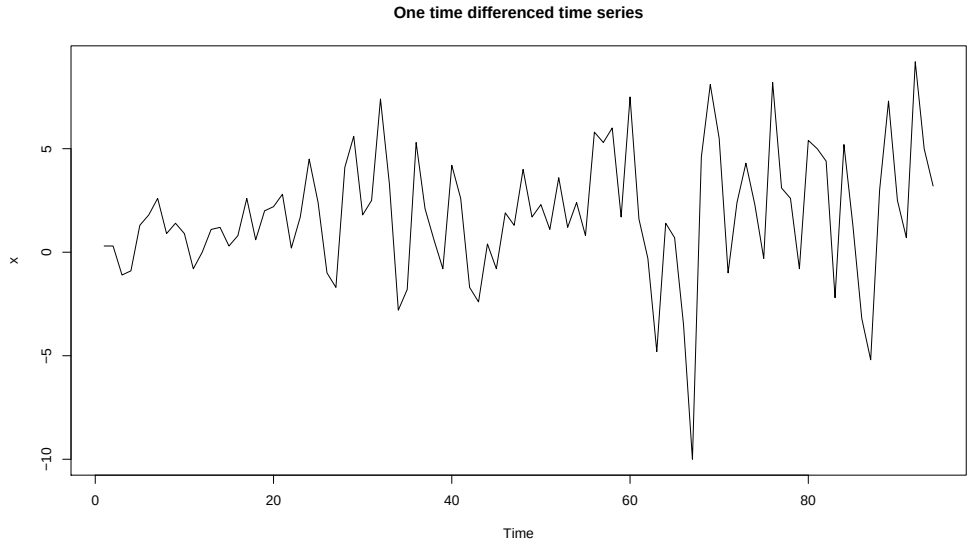

Figure 8: Plot of the one time differenced raw data.

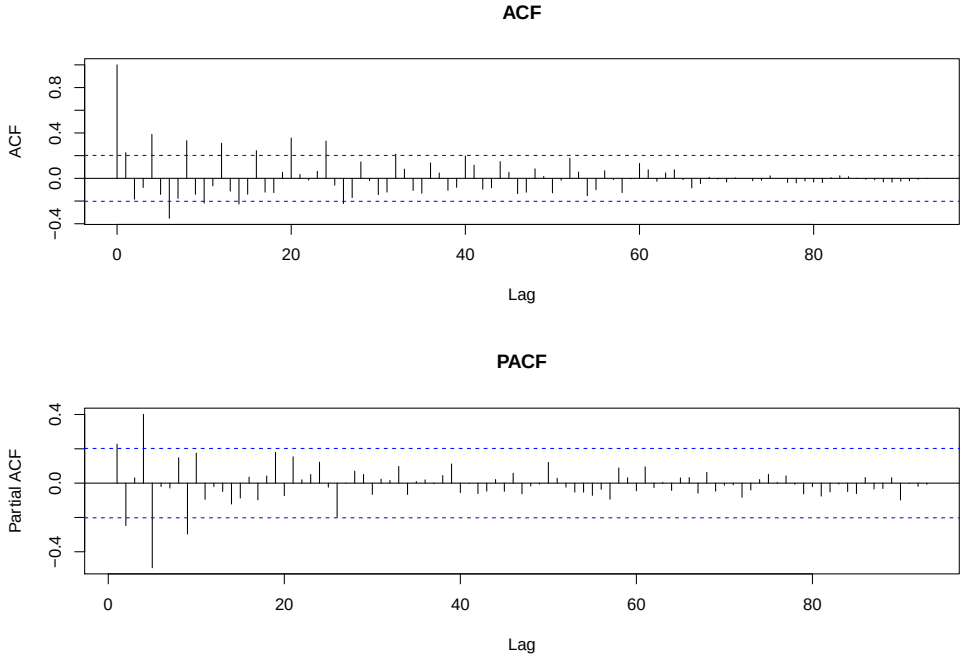

Figure 9: Auto correlation function and partial autocorrelation function for the one time differenced raw data.

The auto correlation function in figure 9 looks like there is a four seasons period, which is what was expected in the discussion of the quarterly house prices. It could also be a situation with two complex AR roots, but the sine waves don't appear to be quite as damped as expected. One thing it is important to note is the

apparent growth of variance in the plot of the differenced time series. It would appear like there is a growth in the order of fluctuations in the market. This seems reasonable taken into accout that the household debt is growing rapidly as people take up large lones to compete in the housing market. The residuals of the fit could be plotted to ascertain the cange in variance, although it is not needed as this is apparant from the plot of the differenced time series. To check what variance transformation to make the Box-Cox transformation test is applied. As the differencing produces some negative values Box-Cox is applied prior to the differencing. The function **BoxCox.lambda**() from the **forecast** library yields $\lambda = 0.3168$, and applying the function **BoxCox**(x, λ) and then differencing gives the plots in figure 14.

Figure 10: Box-Cox transformed and one time differenced data where $\lambda = 0.3167$

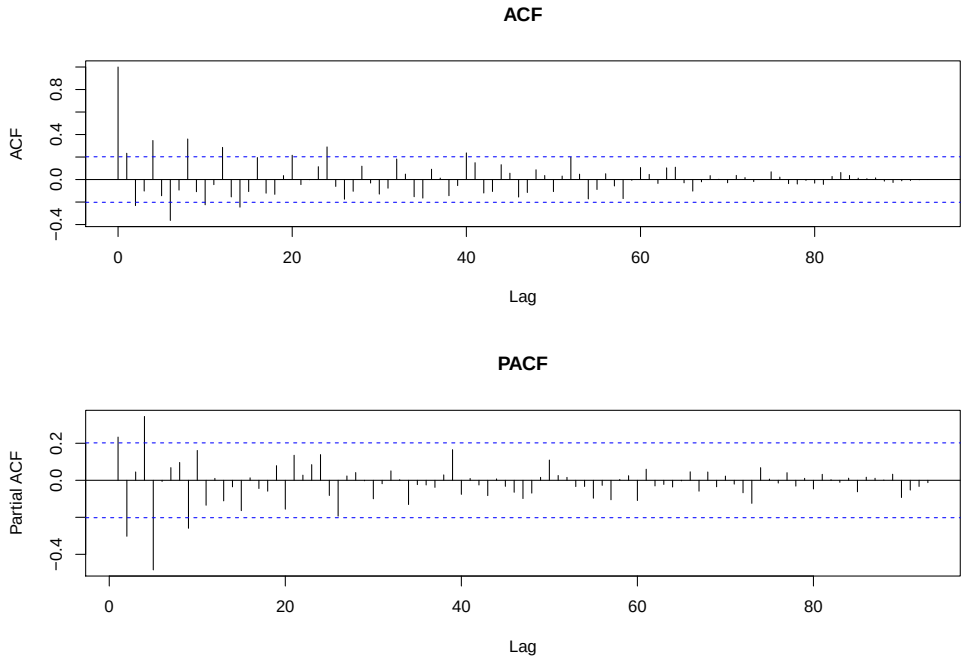

Figure 11: ACF and PACF of Box-Cox transformed and one time differenced data where $\lambda = 0.3167$

The time series in figure 14 looks more constant in variance. The acf and pacf are different from those displayed in figure 8, but still yield the same conclusion that there is an aspect of seasonality. It could also be interpreted for instance as a model with both AR and MA parameters where the roots are quite close in size. With the discussion before tackling the data in mind, the seasonality is investigated further. First the seasonality is investigated with a **seasonplot** from the **forecast** library.

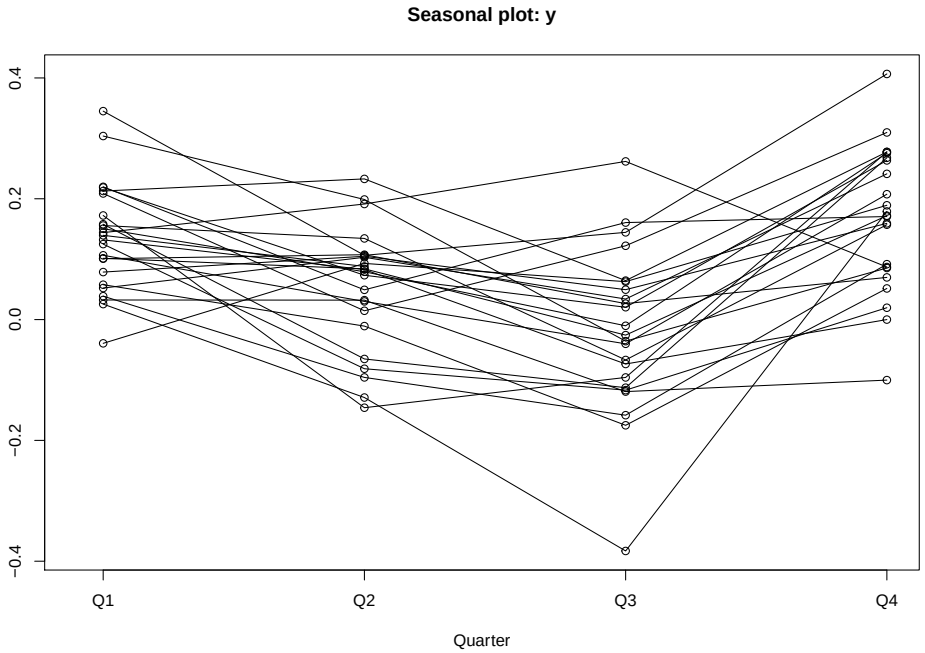

Figure 12: Seasonplot

From the plot in figure 12 it looks like the two middle quarters of the year have a lower housing price than the first and last quarters. It also might look like the variance is a bit different between the seasons, but this could be a result of an outlier in the third quarter. Now the differenced series is investigated.

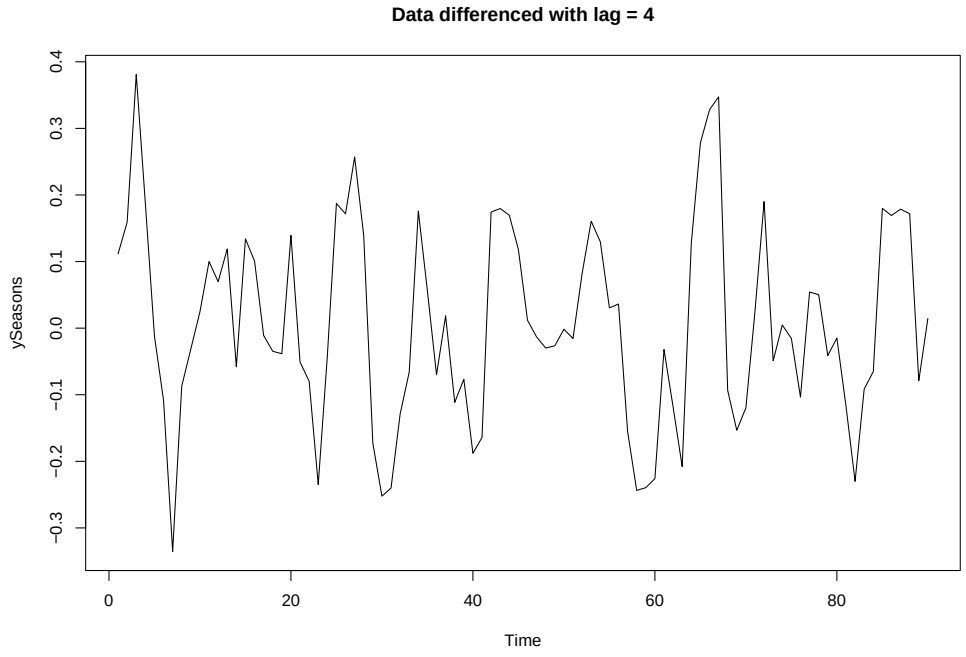

Figure 13: Data differenced with respect to 4 seasons per period.

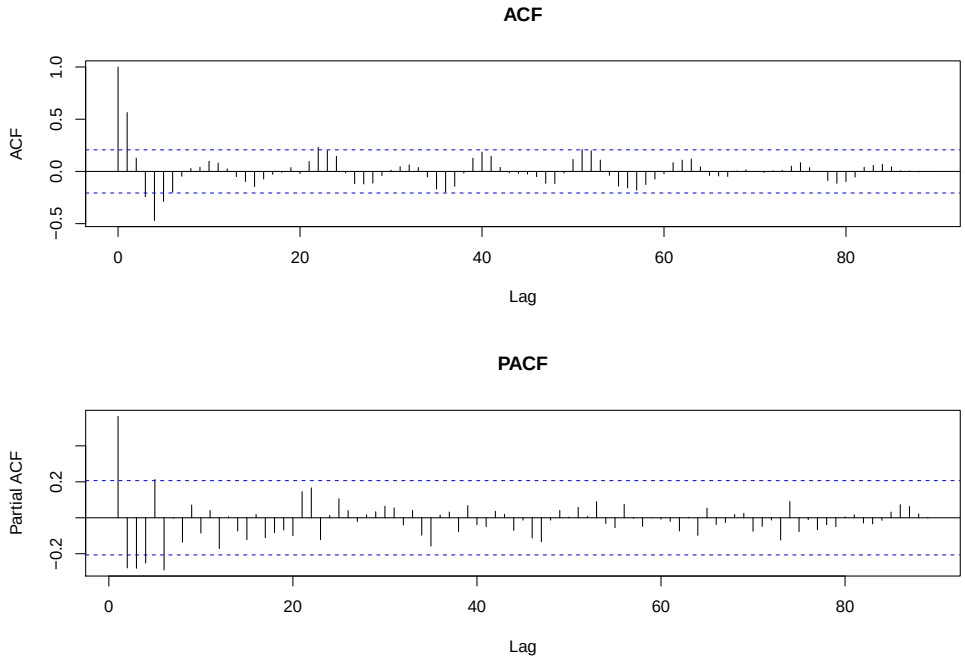

Figure 14: ACF and PACF of data differenced with respect to 4 seasons per period.

The plot of the autocorrelation now clearly shows a damped sine wave with period of approximately 11 lags, and possibly an exponential component. Complex AR roots are known to cause the autocorrelation to behave like damped sine waves. Complex roots come in pairs such that an AR(p) model where p is 2 or higher is considered. The pacf plot has significant spikes at lags 1,2,3,4,5 and 6, although the spikes after lag 1 are not so tall. Although spikes at the first 6 lags and then a cutoff would suggest an AR(6) model this is a model we would be reluctant to choose due to the risk of overfitting the data. The amplitude of the sine wave does vary at approximately a period that is between one and three times the period of the easily visible one. An MA component should result in a gradual decay of the pacf. In theory this might be the case and it might be masked by similar roots in the AR polynomial, or by an AR polynomial of a high order. Applying the R function **auto.arima** on the transformed and differenced time series gives a suggestion to use a AR(2,1,2) model with drift to explain the variance transformed data. Using **Arima** from the **forecast** pakcage on transformed data for multiple models table 1 is generated. A seasonal AR term could act quite like a seasonal difference when the root is close to zero, so it is included as well.

Table 1: Key data of different models fitted

AIC	Log likelihood	Model	σ^2	Drift
-123.29	64.64	ARMA(2,1,0)x(0,1,0)[4]	0.01415	
144.16	76.08	ARMA(2,1,0)x(1,0,0)[4]	0.01168	
-145.44	77.72	ARMA(2,1,0)x(1,0,0)[4]	0.01147	0.0803
-125.21	78.6	ARMA(2,1,2)	0.01133	0.0834
-149.94	81.97	ARMA(4,1,0)x(1,0,0)[4]	0.01067	0.0820
-156.57	88.36	ARMA(6,1,0)x(1,0,0)[4]	0.009466	0.0842

Akaikes AIC is a criterion for evaluating the gain in log likelihood up against increased model complexity, and is calculated as $AIC = -2log(L) + 2k$, where k is the number of parameters estimated. Even though only a penalty of $k_2 = 3$ is applied in the evaluation for the ARMA(2,1,0)x(0,1,0)[4] model versus the five parameters of the ARMA(2,1,0)x(1,0,0)[4] model with drift, the AIC of the most complex model is a lot

higher. This holds when even more parameters are entered, as the later models show. Even though the AIC would suggest to include more parameters this could still result in overfitting. The interpretation of the models with $p = 5$ and $p = 6$ is that the price in the market has a direct dependency on the prices over a year earlier. Whether this is reasonable or not is debateable. The plot of the residuals of the first series in the table is displayed in figure 15.

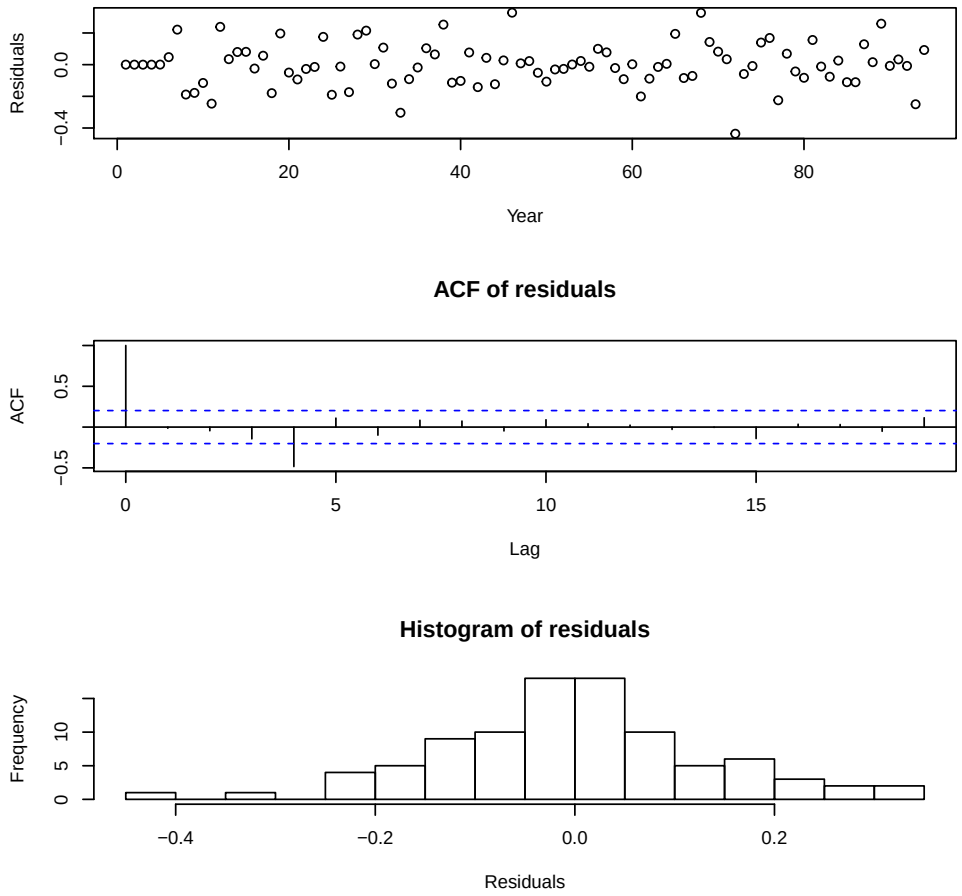

Figure 15: Residuals of the simplest model and their acf. Thecf last graph is a histogram of the residuals.

The residuals look a bit clustered, but might represent independence as the model requires. The shape of the histogram indicates normality. However the acf has a spike at lag 4 which indicates that there is more information to find in the data, but this observation does not rule out that allowing more parameters would not result in over fitting. Complicating to the second model in table 1 the spike in the acf gets a little shorter, and adding the drift term the spike disappears into the insignificance region. Running **auto.arima** on the residuals of the least complex model does however return a white noise process with zero mean. The model that **auto.arima** selected to represent the whole time series have residuals as displayed in figure 16 which gives a nice histogram and residual plot, but where the acf plot is not ideal.

```
Series: res
ARIMA(0,0,0) with zero mean

sigma^2 estimated as 6.225:  log likelihood=-219.33
AIC=440.65   AICc=440.69   BIC=443.19
```

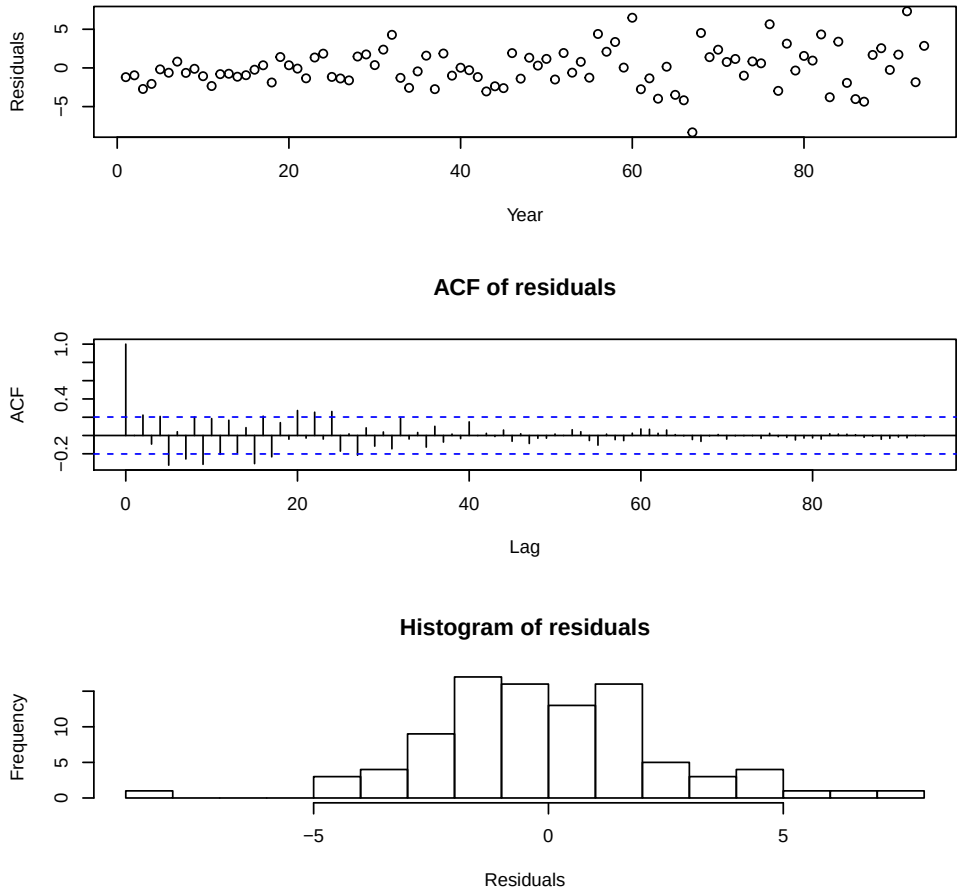

Figure 16: Residuals of the automatic model and their acf. Thecf last graph is a histogram of the residuals.

The different models give quite different interpretations. The model that **auto.arima** chose represent a model dominated by a constant growth term where shocks of both long term and short term consequence might occur. On the other hand the simplest model in the table indicates a process that coincidentally has been increasing the last 23 years, but might turn around quite promptly. For the market to turn around promptly without notice is scarcely expected. With the simple model the market would be expected to go in different directions, sometimes decline and sometimes increas, where the other model would guarantee a steady increase. Neither of these options seem completely sound in the interpretation. The third model in table 1 could represent a sort of middle ground between these viewes on the economy; the process has drift, but is otherwise governed by the long term effects on random shocs.

One could discuss to what degree the parameters of a time series model describe the inner workings of the system. For instance seasonality might only hold in the Norwegian house pricing market as long as everyone who buy houses believes it is. In reality figure 6 and 8 look a bit like there might be different parameters governing different parts times in the series. The underlying assumption in time series modeling is that the behaviour of the system is governed by the same processes at every time point. In practise time series are

used to make forecasts. Assuming that the behaviour of the next point is governed by different processes than the time points we have observed, would leave us with nothing to base the forecasts on. If we only intend to use the data to forecast for instance 20 time steps into the future any of the models discussed in the previous paragraphs might suffice. Below is a figure of those plots for different models.

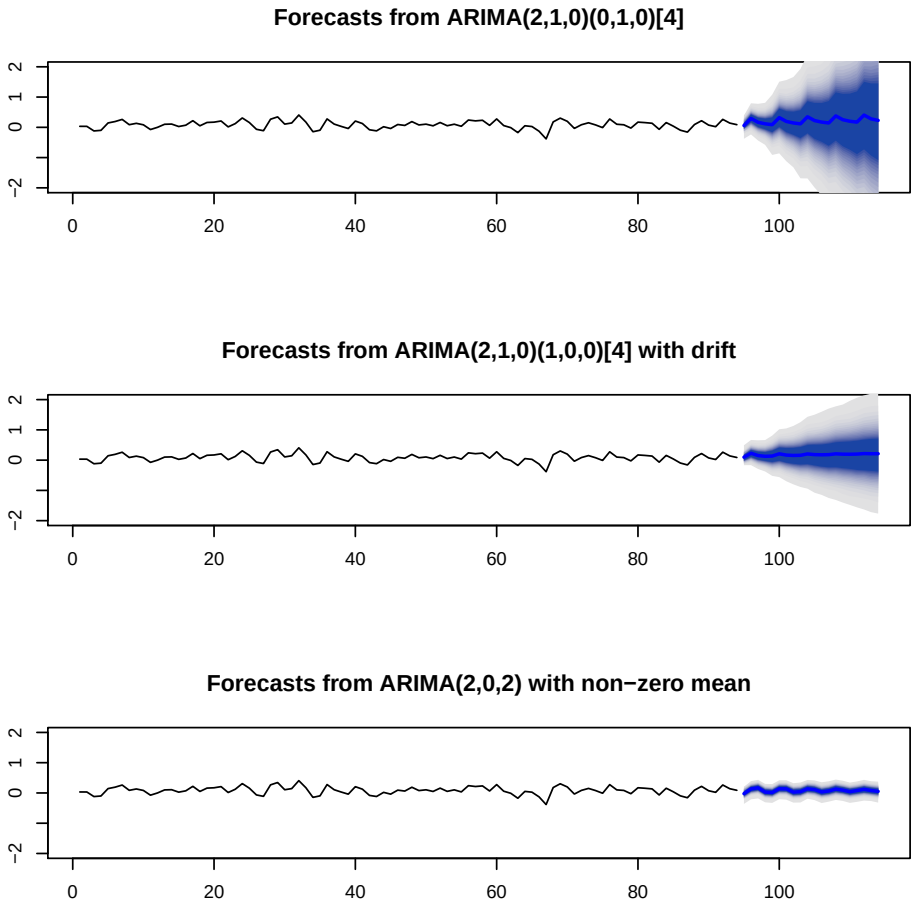

Figure 17: Fan plots of forecast regions 20 steps into the future.

The forecast plots in figure 17 displays what was discussed in the previous sections. All things considered the the third model in table 1, namely the ARMA(2,1,0)x(1,0,0)[4] is chosen for the forecast. The plots relevant for the residuals is displayed in figure 18 and the actual parameters are displayed in table 2. The forecasts for this model is in the middle graph in figure 17. The forecasts are made with the code lines similar to $\mathbf{a1F1 = forecast.Arima(object = a1, h = 20, fan = TRUE, bootstrap = TRUE})$ where the function is from the **forecast** library. This produce a

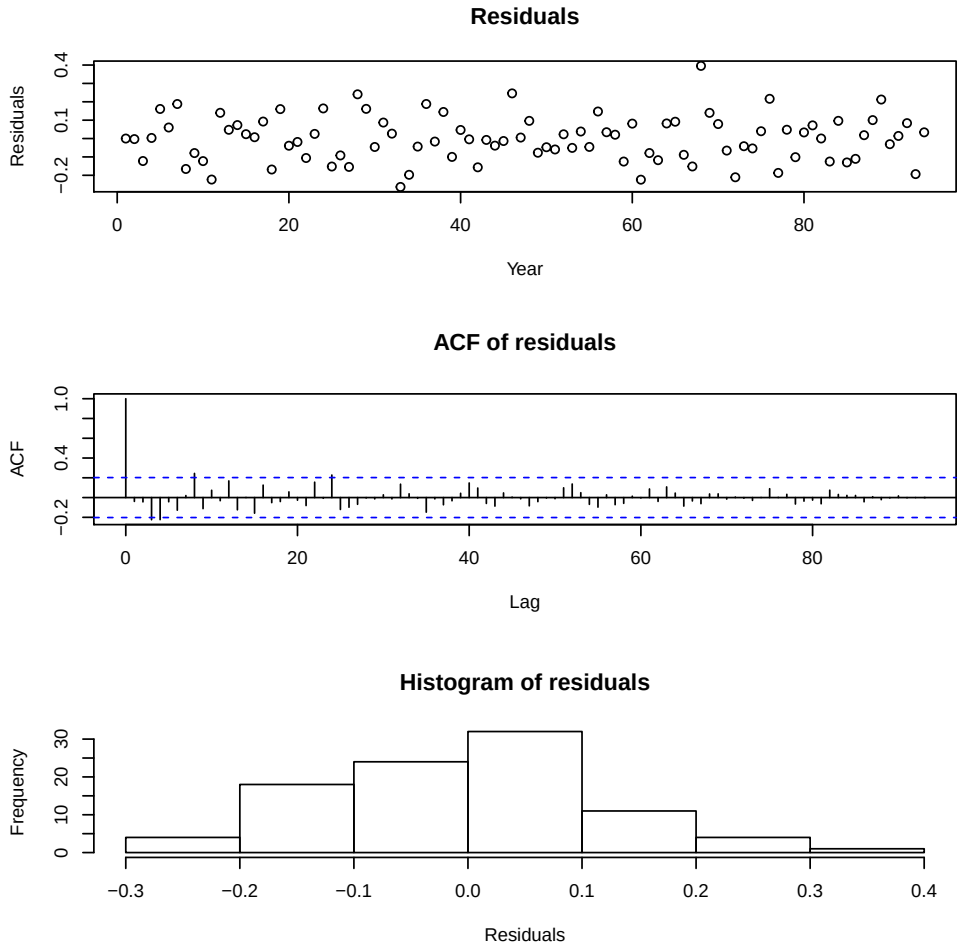

Figure 18: Residuals of the model ARMA(2,1,0)x(1,0,0)[4] with drift and their acf. The last graph is a histogram of the residuals.

Table 2: Parameters of the fitted model ARMA(2,1,0)x(1,0,0)[4] with drift.

	AR1	AR2	SAR1	Drift
Value	0.6185	-0.2545	0.5763	0.0803
Standard error	0.1083	0.0995	0.0915	0.0379

Problem 2

In this section we want to fit a given series of n observations to a stationary AR(2) process without deterministic trend parameter

$$Z_t = \phi_1 Z_{t-1} + \phi_2 Z_{t-2} + a_t,$$

where $a_t \sim N(0, \sigma_a^2)$.

In order to do later computations we need the autocovariance for lag 0, γ_0, and the autocorrelation for lag 1, ρ_1. To obtain the autocovariance we multiply both sides with Z_{t-k} and take the expectation

$$\gamma_k = E[Z_t Z_{t-k}] = \phi_1 E[Z_{t-1} Z_{t-k}] + \phi_2 E[Z_{t-2} Z_{t-k}] + E[a_t Z_{t-k}],$$

which becomes

$$\gamma_k = \begin{cases} \phi_1 \gamma_{k-1} + \phi_2 \gamma_{k-2} + \sigma_a^2 \ , & k = 0 \\ \phi_1 \gamma_{k-1} + \phi_2 \gamma_{k-2} \ , & k \neq 0 \ . \end{cases}$$

To calculate γ_0 we need

$$\begin{aligned} \gamma_0 &= \phi_1 \gamma_1 + \phi_2 \gamma_2 + \sigma_a^2 \ , \\ \gamma_1 &= \phi_1 \gamma_0 + \phi_2 \gamma_1 \ , \\ \gamma_2 &= \phi_1 \gamma_1 + \phi_2 \gamma_0 \ , \end{aligned} \tag{1}$$

which by using

$$\gamma_1 = \frac{\phi_1}{1 - \phi_2} \gamma_0 \ ,$$

becomes

$$\gamma_0 = \phi_1 (\frac{\phi_1}{1 - \phi_2} \gamma_0) + \phi_2 (\phi_1 (\frac{\phi_1}{1 - \phi_2} \gamma_0) + \phi_2 \gamma_0) + \sigma_a^2 \ ,$$

such that

$$\gamma_0 = \frac{\sigma_a^2}{(1 - \frac{\phi_1^2}{1 - \phi_2} - \frac{\phi_1^2 \phi_2}{1 - \phi_2} - \phi_2^2)} \ . \tag{2}$$

From (1) we get that $\rho_1 = \frac{\phi_1}{1 - \phi_2}$.

We want to fit the model by exact maximum likelihood estimation, and write

$$L(\phi_1, \phi_2, \sigma_a^2) = f(Z_1) f(Z_2 | Z_1) \prod_{t=3}^{n} f(Z_t | Z_{t-1}, Z_{t-2}) \ ,$$

or the log likelihood

$$l(\phi_1, \phi_2, \sigma_a^2) = log(f(Z_1)) + log(f(Z_2 | Z_1)) \sum_{t=3}^{n} log(f(Z_t | Z_{t-1}, Z_{t-2})) \ .$$

Rewriting the process in the moving average representation $Z_t = \sum_{j=0}^{\infty} \psi_j a_{t-j}$ we see that Z_t is normally distributed. The variance of the first observation $E[Z_1^2] = \gamma_0$, the zero lag autocorrelation, such that $Z_1 \sim N(0, \gamma_0)$. The next observation can be written as a regression on the previous observation

$$Z_2 | Z_1 = \phi_{11} Z_1 + e_n$$

where ϕ_{11} is the autocorrelation for lag 1 which is $\rho_1 = \frac{\phi_1}{1 - \phi_2}$ for the AR(2) process and $e_n \sim N(0, v_1)$ with $v_1 = (1 - \phi_2^2)\gamma_0$, such that $Z_2 \sim N(\phi_{11} Z_1, v_1)$. For the observations of time $t \geq 3$ all the neccessary previous values are known

$$Z_t | Z_{t-1}, Z_{t-2} = \phi_1 Z_{t-1} + \phi_2 Z_{t-2} + a_t,$$

and $Z_t \sim N(\phi_1 Z_{t-1} + \phi_2 Z_{t-2}, \sigma_a^2)$. Using that the probability density function of a variable $x \sim N(\mu, \sigma^2)$ is

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

the probability densities of our observations are

$$\begin{aligned} P(Z_1) &= (2\pi\gamma_0)^{-\frac{1}{2}} e^{-(2\gamma_0)^{-1}(Z_1)^2} \\ P(Z_2|Z_1) &= (2\pi v_1)^{-\frac{1}{2}} e^{-(2v_1^2)^{-1}(Z_2-\phi_{11}Z_1)^2} \\ P(Z_t|Z_{t-1}, Z_{t-2}) &= (2\pi\sigma_a^2)^{-\frac{1}{2}} e^{-(2\sigma_a^2)^{-1}(Z_t-\phi_1 Z_{t-1}-\phi_2 Z_{t-2})^2} \end{aligned}$$

and the total log likelihood is thus

$$\begin{aligned} l(\phi_1, ph1_2, \sigma_a^2) = &-\frac{n}{2}log(2\pi) - \frac{1}{2}log(\gamma_0) - \frac{1}{2}log(v_1) - \frac{n-2}{2}log(\sigma_a^2) - \frac{Z_1^2}{2\gamma_0} - \frac{(Z_2-\phi_{11}Z_1)^2}{2v_1} \\ &- (2\sigma_a^2)^{-1}\sum_{t=3}^{n}(Z_t - \phi_1 Z_{t-1} - \phi_2 Z_{t-2})^2 \ , \end{aligned}$$

and is calculated in the following R code, which can be maximised using the function **optim** in R.

```
lnL <- function(param,z){
  phi1 <- unlist(param[1])
  phi2 <- unlist(param[2])
  sigma_a2 <- unlist(param[3])
  n <- length(z)
  t <- 3:n
  gamma0= sigma_a2/(1-phi1^2/(1-phi2)- phi1^2*phi2/(1-phi2)-phi2^2)
  phi11 = phi1/(1-phi2)
  v1 = gamma0*(1-phi11^2)
  S <-  sum((z[t]-phi1*z[t-1]-phi2*z[t-2])^2)
  tmp <- -n*log(2 *pi)/2 -log(gamma0)/2 - log(v1)/2 -(n-2)*log(sigma_a2)/2 -
    z[1]^2/(2*gamma0) - (z[2]-phi11*z[1])^2/(2*v1) -(2*sigma_a2)^(-1) *S
  return(-tmp)
}
```

Note that $-tmp$ is returned from the function. This is because minimizing $-l$ is equivalent to maximizing l. In order to maximize this R function we need to hand **optim** a good choise of initial values for ϕ_1, ϕ_2 and σ_a^2. First we compute some parameters of the sample, namely

$$\begin{aligned} \hat{\gamma}_0 &= \frac{\sum_{t=1}^{n} Z_t}{n} \\ \hat{\rho} &= \frac{\sum_{t=1}^{n-k} Z_t Z_{t+k}}{n\gamma_0} \ , \end{aligned}$$

such that we can compute σ_a^2 from equation (2) and ϕ_1 and ϕ_2 from the Durbin Levinson algotithm we made in the last project. The following code executes this when handed time series data and number of observations. The code could be adjusted to fit a time series with a deterministic mean.

```
exactL <-function(n,myModel){
#1 compute sample gamma_0
myMean=0
gamma0 = (1/n)*sum((myModel-myMean)^2) #there are lots of ways to estimate it

#2 compute sample acf
rhos = c(sum((myModel[1:(n-1)]-myMean)*(myModel[2:(n)]-myMean))/(n*gamma0),
  sum((myModel[1:(n-2)]-myMean)*(myModel[3:(n)]-myMean))/(n*gamma0) )
```

```
#2 obtain \phi_1^hat and \phi_2^hat and sima_a^2
phis = acf2pacf(rhos,gamma0)
sigmaa2 = gamma0*(1-sum(phis$weights*rhos))


#fit <- optim(c(phis£weights, sigmaa2),lnL,z=myModel,hessian=TRUE)
fit <- optim(c(0.8897, -0.4858, 1),lnL,z=myModel,hessian=TRUE)

return(c(phis$weights, sigmaa2, parametersExact=fit$par))
}
```

Now the R functions above are verified by simulating 100 processes each with 200 data points from a time series with R function **arima.sim**. The result is displayed in figure 19, and shows that the initial estimates and the exact likelihood estimates are very similar. In addition 200 data points is a lot compared to the two data points that are different for exact likelihood estimation and conditional likelihood estimation. The figure also shows that the estimates does not come quite as close to the true values as would be wished. This is important to note when considering the estimation of the parameters in a realistic case where they are unknown. With more observations these estimates would however be much closer to the true value.

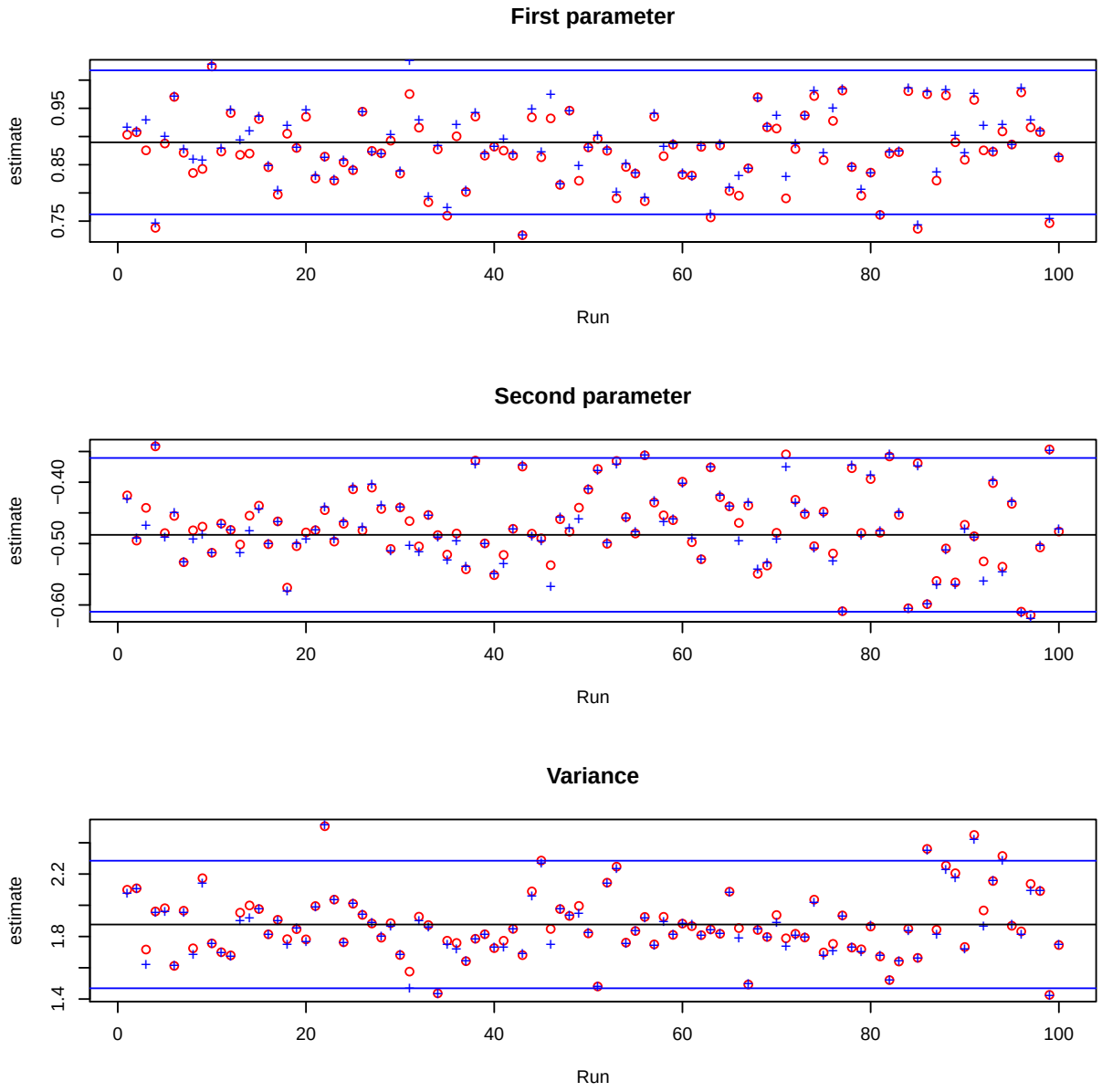

Figure 19: Plots of values for ϕ_1 and ϕ_2 for 100 different observations of a time series with $(\phi_1, \phi_2) = (0.8897, -0.4858)$. Red points are initial data and blue points are found by maximizing the exact likelihood. The true values are indicated by a black line, while the blue lines are 2 standard deviations from the true value.

Kalman Filter

The state space representation is described by the state equation

$$\boldsymbol{Y_{t+1}} = \boldsymbol{A}\boldsymbol{Y}_t + \boldsymbol{G}\boldsymbol{a}_{t+1}$$

and the output equation

$$\boldsymbol{Z}_t = \boldsymbol{H}\boldsymbol{Y}_t + \boldsymbol{b}_t$$

The AR(2)-model can be written in this form with Z_t and Z_{t+1} as state variables. It is also assumed that Z_t is observed with time dependent observation error $\sigma^2_{b,t}$. By choosing $\boldsymbol{A} = \begin{pmatrix} 0 & 1 \\ \phi_2 & \phi_1 \end{pmatrix}$ and $\boldsymbol{G} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$ we obtain

$$\begin{pmatrix} Y_{t+1} \\ Y_{t+2} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \phi_2 & \phi_1 \end{pmatrix} \begin{pmatrix} Y_t \\ Y_{t+1} \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} a_{t+1} \\ a_{t+2} \end{pmatrix}$$

and

$$\begin{pmatrix} Z_t \\ Z_{t+1} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} Y_t \\ Y_{t+1} \end{pmatrix} + \begin{pmatrix} b_t \\ b_{t+1} \end{pmatrix}$$

gives $\boldsymbol{H} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.

Now, let $\hat{Y}_{t|s} = E(Y_t|Z_1, Z_2, \ldots, Z_s)$ and $V_{t|s} = Var(Y_t|Z_1, Z_2, \ldots, Z_s)$. This is the forecasted values and follows the recursions

$$\begin{aligned} \hat{Y}_{t+1|s} &= A\hat{Y}_{t|s} \\ V_{t+1|s} &= AV_{t|s}A^T + G\Sigma G^T \end{aligned}$$

where $\Sigma = \begin{pmatrix} \sigma_a^2 & 0 \\ 0 & \sigma_a^2 \end{pmatrix}$. Now, we want to find the initial variance $V_{1|0}$. Assuming stationarity one must have

$$V_{1|0} = AV_{1|0}A^T + G\Sigma G^T$$

Using the fact that $\text{vec}(ABC) = C^T \otimes A \text{ vec}(B)$.

$$\text{vec}(V_{1|0}) = (I - (A \otimes A))^{-1}\text{vec}(G\Sigma G^T)$$

This gives the following expression for $V_{1|0}$

$$\text{vec}(V_{1|0}) = \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -\phi_2 & -\phi_1 \\ 0 & -\phi_2 & 1 & -\phi_1 \\ -\phi_2^2 & -\phi_1\phi_2 & -\phi_1\phi_2 & 1-\phi_1^2 \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ 0 \\ 0 \\ \sigma_a^2 \end{pmatrix}$$

By applying Bayes theorem the following filtering recursions can be obtained

$$\begin{aligned} \hat{Y}_{t+1|t+1} &= \hat{Y}_{t+1|t} + K_{t+1}(Z_{t+1} - H\hat{Y}_{t+1|t}) \\ V_{t+1|t+1} &= (I - K_{t+1}H)V_{t+1|t} \end{aligned}$$

where $K_{t+1} = V_{t+1|t}H^t(HV_{t+1|t}H^T + \Omega)^{-1}$ and $\Omega = \begin{pmatrix} \sigma^2_{b,t} & 0 \\ 0 & \sigma^2_{b,t+1} \end{pmatrix}$. A missing data point at time t means that no new information is added at this time, such that the probability distribution of Y_{t+1} is only conditioned on observations up to time t, which is equivalent to Ω approaching infinity. This leads to a Kalman gain $K_{t+1} = 0$ such that the filtering equations become

$$\begin{aligned} \hat{Y}_{t+1|t+1} &= \hat{Y}_{t+1|t} \\ V_{t+1|t+1} &= V_{t+1|t} \, . \end{aligned}$$

The likelihood can be computed via the forward Kalman recursions. The likelihood is given by

$$L(\theta) = f(Z_1, Z_2, \ldots, Z_n) = \prod_{t=1}^{n} f(Z_t|Z_{t-1}, Z_{t-2}, \ldots, Z_{Z_1})$$

and the distribution of $f(Z_t|Z_{t-1}, Z_{t-2}, \ldots, Z_{Z_1})$ is normally distributed with mean and covariance given by

$$E(Z_t|Z_{t-1}, Z_{t-2}, \ldots, Z_{Z_1}) = E(HY_t + b_t|Z_{t-1}, Z_{t-2}, \ldots, Z_{Z_1}) = H\hat{Y}_{t|t-1}$$
$$Var(Z_t|Z_{t-1}, Z_{t-2}, \ldots, Z_{Z_1}) = Var(HY_t + b_t|Z_{t-1}, Z_{t-2}, \ldots, Z_{Z_1}) = H\hat{V}_{t|t-1}H^T + \Omega_t$$

By backward recursion the mean and covariance given all the data can be found. These smoothed values are given by

$$\begin{aligned}\hat{Y}_{t|n} &= \hat{Y}_{t|t} + J_t(\hat{Y}_{t+1|n} - \hat{Y}_{t+1|t}) \,, \\ V_{t|n} &= V_{t|t} + J_t(V_{t+1|n} - V_{t+1|t})J_t^T \,, \\ J_t &= V_{t|t}A^T V_{t+1|t}^{-1},\end{aligned}$$

The function below uses kalman recursion to obtain forecasted, filtered and smoothed means and covariance matrices. The input to the function is the model parameters, a vector of observed non-missing or missing observations and a vector containing known values for $\sigma_{b,t}$.

```
kalman = function(par, Z, sb2, lnL = TRUE) {
    phi1 = par[1]
    phi2 = par[2]
    sa2 = par[3]

    n = length(Z)
    Z = rbind(Z[-n], Z[-1])

    # Initializing matrices for storing forecasted and filtered values
    Yforecast <- Yfilter <- rbind(numeric(n), numeric(n))
    Vforecast <- Vfilter <- array(0, dim = c(2, 2, n))

    # Finding the initial variance V10
    mmat = c(1, 0, 0, -1)
    mmat = rbind(mmat, c(0, 1, -phi2, -phi1))
    mmat = rbind(mmat, c(0, -phi2, 1, -phi1))
    mmat = rbind(mmat, c(-phi2^2, -phi1 * phi2, -phi1 * phi2, (1 - phi1^2)))
    vecV10 = solve(mmat) %*% matrix(c(0, 0, 0, sa2))
    V10 = matrix(vecV10, nrow = 2)

    Yforecast[, 1] = c(0, 0)
    Vforecast[, , 1] = V10

    # G*Sigma*G
    gs = diag(c(0, sa2))

    sa2 = diag(sa2, 2)
    A = matrix(c(0, phi2, 1, phi1), nrow = 2)

    # Setting initial value for likelihood
    if (is.na(Z[1, 1]) | is.na(Z[2, 1])) {
        sum.lnL = 0
    } else {
        sum.lnL = dmvnorm(Z[, 1], mean = Yforecast[, 1], sigma = Vforecast[,
            , 1] + diag(c(sb2[1], sb2[2])), log = TRUE)
    }
```

```
    for (t in 1:(n - 1)) {
        Omega = diag(c(sb2[t], sb2[t + 1]))
        # Filtering
        if (is.na(Z[1, t]) | is.na(Z[2, t])) {
            Yfilter[, t] = Yforecast[, t]
            Vfilter[, , t] = Vforecast[, , t]
        } else {
            sum.lnL = sum.lnL + dmvnorm(Z[, t], mean = Yforecast[, t], sigma = Vforecast[,
                , t] + Omega, log = TRUE)
            K = Vforecast[, , t] %*% solve(Vforecast[, , t] + Omega)
            Yfilter[, t] = Yforecast[, t] + K %*% (Z[, t] - Yforecast[, t])
            Vfilter[, , t] = (diag(2) - K %*% diag(2)) %*% Vforecast[, , t]
        }
        # Forecasting
        Yforecast[, t + 1] = A %*% Yfilter[, t]
        Vforecast[, , t + 1] = A %*% Vfilter[, , t] %*% t(A) + gs
    }

    # Returning the log likelihood
    if (lnL) {
        sum.lnL
    } else {
        # Initializing variables used for smoothing
        Ysmooth = rbind(numeric(n), numeric(n))
        Vsmooth = array(0, dim = c(2, 2, n))
        Ysmooth[, (n - 1)] = Yfilter[, (n - 1)]
        Vsmooth[, , (n - 1)] = Vfilter[, , (n - 1)]
        for (t in (n - 2):1) {
            # Smoothing
            J = Vfilter[, , t] %*% t(A) %*% solve(Vforecast[, , t + 1])
            Ysmooth[, t] = Yfilter[, t] + J %*% (Ysmooth[, t + 1] - Yforecast[,
                t + 1])
            Vsmooth[, , t] = Vfilter[t] + J %*% (Vsmooth[, , t + 1] - Vforecast[,
                , t + 1]) %*% t(J)
        }
        # Returning list of values
        list(Yforecast = Yforecast, Vforecast = Vforecast, Yfilter = Yfilter,
            Vfilter = Vfilter, Ysmooth = Ysmooth, Vsmooth = Vsmooth)
    }
}
```

Example with non-missing values

Now, 100 values from an AR(2)-model was simulated using **arima.sim** in R and time dependent error was added.

```
## Simulate some data from the above state space model
n = 100
Y = arima.sim(model = list(order = c(2, 0, 0), ar = c(0.4, 0.2)), n = n)

# Adding the measurement error vector
sb2 = seq(2, 3, length.out = n)
Z = numeric(n)
```

```r
for (i in 1:n) {
    Z[i] = Y[i] + rnorm(1, sd = sqrt(sb2[i]))
}
```

Then the kalman function was used to fit parameters to the observations and the results is compared to a fitting done by using the function **arima** in R. The code and the results are shown below.

```r
# Fit the model by maximum likelihood
fit = optim(c(0.4, 0.2, 1), kalman, lower = c(-0.999, 0.001, 0.001), upper = c(0.999,
    Inf, Inf), Z = Z, sb2 = sb2, control = list(fnscale = -1), method = "L-BFGS-B",
    hessian = TRUE)

# Finding parameters using arimaFit
arimaFit = arima(Z, order = c(2, 0, 0))
fit$par
```

```
## [1] 0.24932934 0.01370475 2.70982101
```

```r
arimaFit
```

```
##
## Call:
## arima(x = Z, order = c(2, 0, 0))
##
## Coefficients:
##          ar1     ar2  intercept
##       0.1552  0.0045    -0.2935
## s.e.  0.1009  0.1010     0.2367
##
## sigma^2 estimated as 3.966:  log likelihood = -210.79,  aic = 429.59
```

The estimates of the parameters ϕ_1 and ϕ_2 in the AR(2)-model using the two different methods does not give the exact same result, but the estimates are quite similar. This is as expected. The estimate using the function **kalman** lies within the standard error of the estimates from **arima**. This indicates that the function **kalman** is implemented correct.

After the fitting was done, the function was used to calculate filtered and smoothed means, and also the corresponding covariance matrices. The results are shown in Figure 20. The black dots is the observed states, the red line is the true line, the blue line is the filtered values and the green line is the smoothed values.

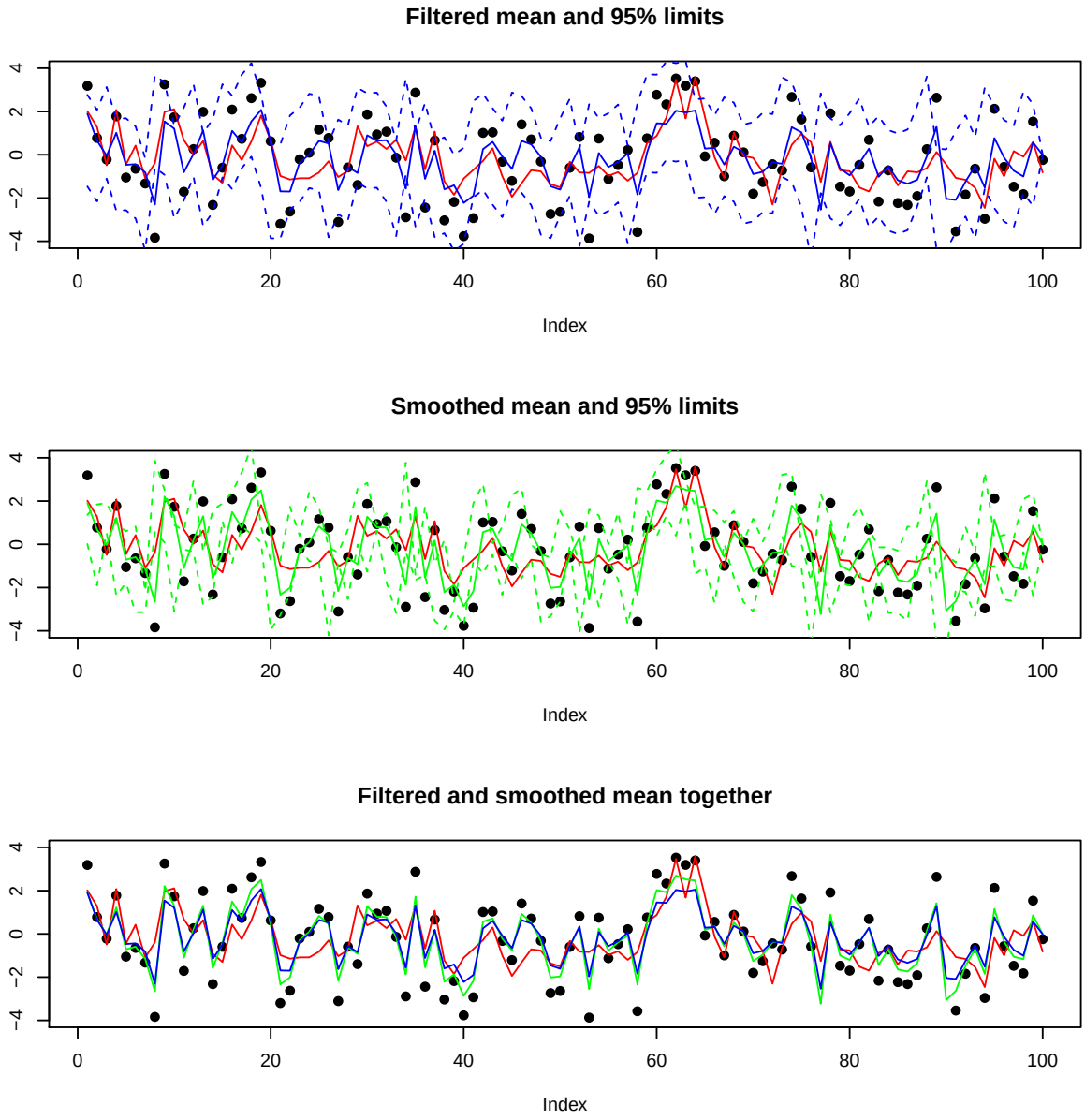

Figure 20: Results for non-missing data.

The filtered and the smoothed values seems to be quite similar. The smoothed values seems to tend more towards the observed data, than the filtered values. This is as expected since the filtered values is conditioned only on the the previous observations, while the smoothed values are conditioned on all the observations.

Example with missing values

This example uses the same approach as the previous example only this time the data contains missing values. The code for constructing the data is shown below.

```r
## Simulate some data from the above state space model
n = 100
Y = arima.sim(model = list(order = c(2, 0, 0), ar = c(0.4, 0.2)), n = n)
Y[c(1, 8, 9, 10, 11, 20, 21, 22, 23, 24, 25, 26, 27)] = NA

# Adding the measurement error vector
sb2 = seq(2, 3, length.out = n)
Z = numeric(n)
for (i in 1:n) {
    Z[i] = Y[i] + rnorm(1, sd = sqrt(sb2[i]))
}
```

Again, the kalman function was used to fit parameters to the observations and the results was compared to a fitting done by using the function **arima** in R. The code and the results are shown below.

```r
# Fit the model by maximum likelihood
fit = optim(c(0.4, 0.2, 1), kalman, lower = c(-0.999, 0.001, 0.001), upper = c(0.999,
    Inf, Inf), Z = Z, sb2 = sb2, control = list(fnscale = -1), method = "L-BFGS-B",
    hessian = TRUE)

# Finding parameters using arimaFit
arimaFit = arima(Z, order = c(2, 0, 0))
fit$par
```

```
## [1] 0.1740518 0.1793911 2.7194228
```

```r
arimaFit
```

```
##
## Call:
## arima(x = Z, order = c(2, 0, 0))
##
## Coefficients:
##          ar1     ar2  intercept
##       0.1211  0.1240    -0.2155
## s.e.  0.1134  0.1135     0.2892
##
## sigma^2 estimated as 4.283:  log likelihood = -186.8,  aic = 381.61
```

The estimates of the parameters ϕ_1 and ϕ_2 in the AR(2)-model using the two different methods does not give the exact same result, but the estimates are quite similar.

Then filtered and smoothed means and covariance matrices was found. Figure 21 shows results of this. The black dots is the observed states, the red line is the true line, the blue line is the filtered values and the green line is the smoothed values.

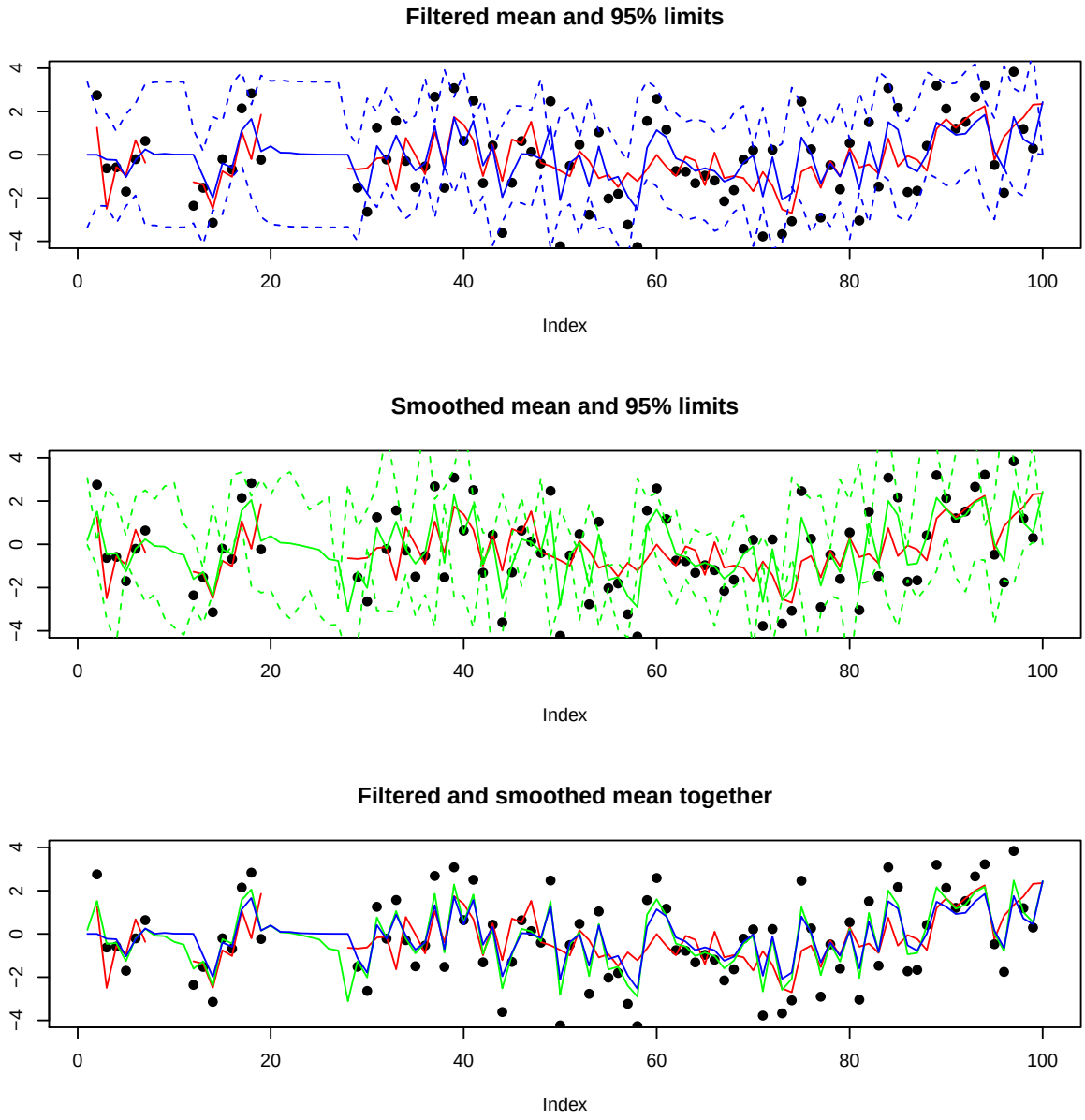

Figure 21: Results for missing data.

In this example there is a large gap of missing values between time index 20 and 27. In this gap the filtered and the smoothed values behaves a bit differently. The filtered mean does not change in this gap. This is because these values are conditioned on previous observations and since no new information is added in this gap the mean stays the same. The smoothed mean is conditioned on all the observations (on both sides of the gap) and hence this is not a straight line.