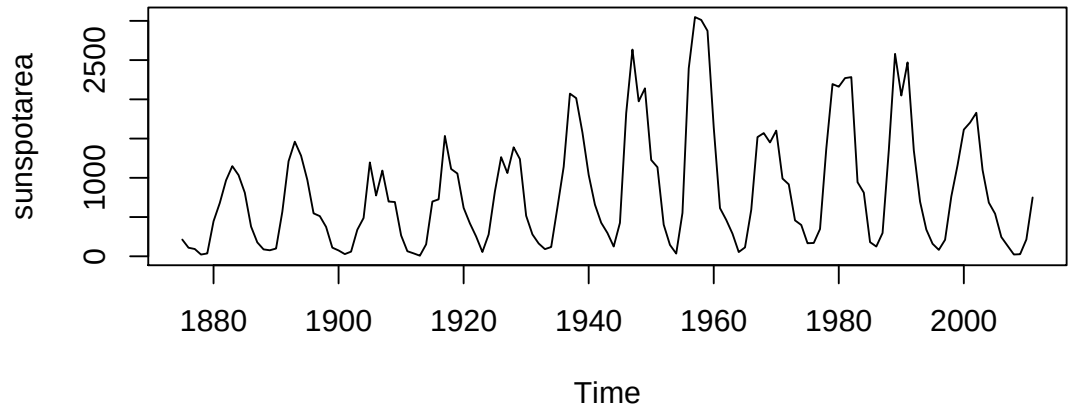
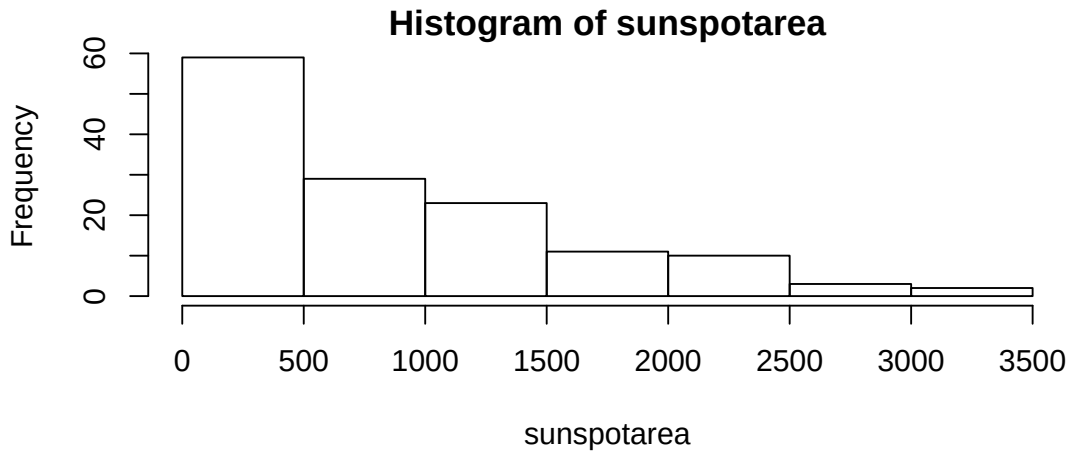
Solution to Series 5

1. a)
```
> library(fpp)
> plot(sunspotarea)
```

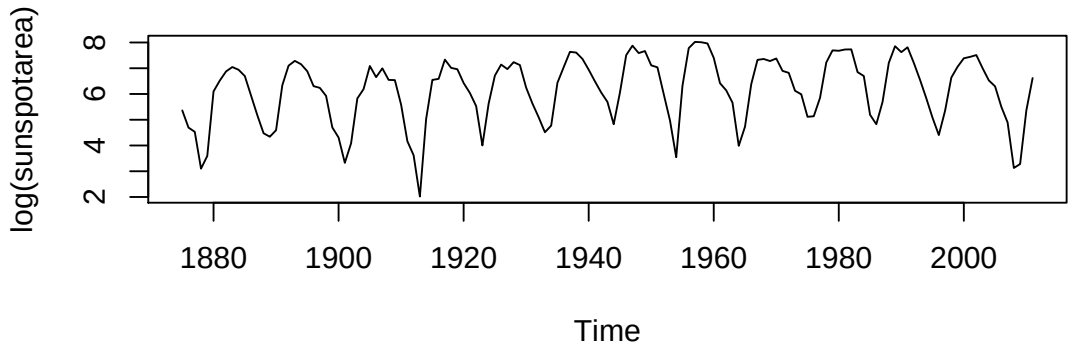



```
> hist(sunspotarea)
```

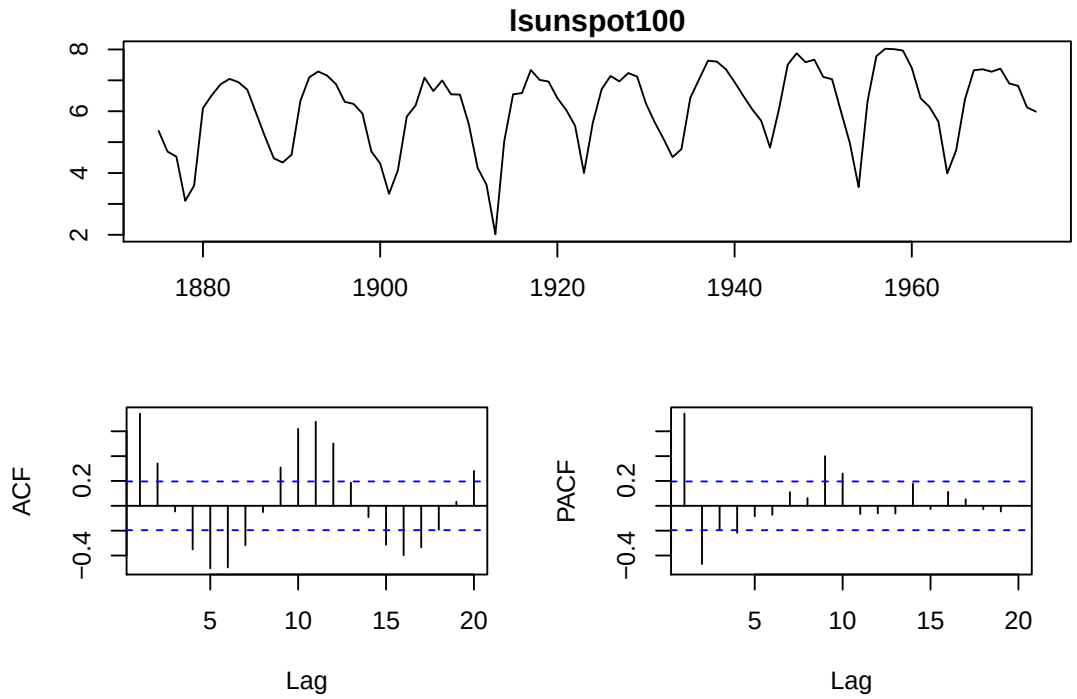


The time series has a right-skewed distribution, meaning that it obtains few very large values and many small values. In order to make the distribution more centered, a log-transformation is advisable.

```
> plot(log(sunspotarea))
```

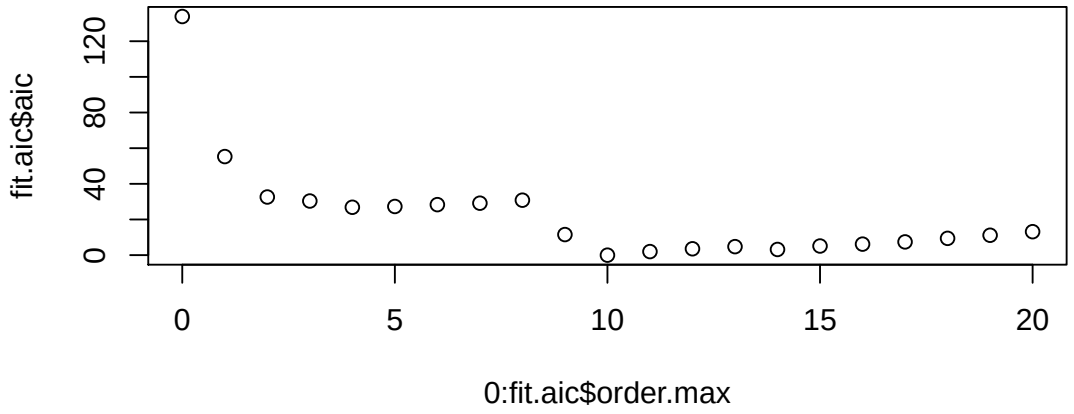


b)
```
> lsunspot100 <- window(log(sunspotarea), start = 1875, end = 1974)
> tsdisplay(lsunspot100, points = FALSE)
```



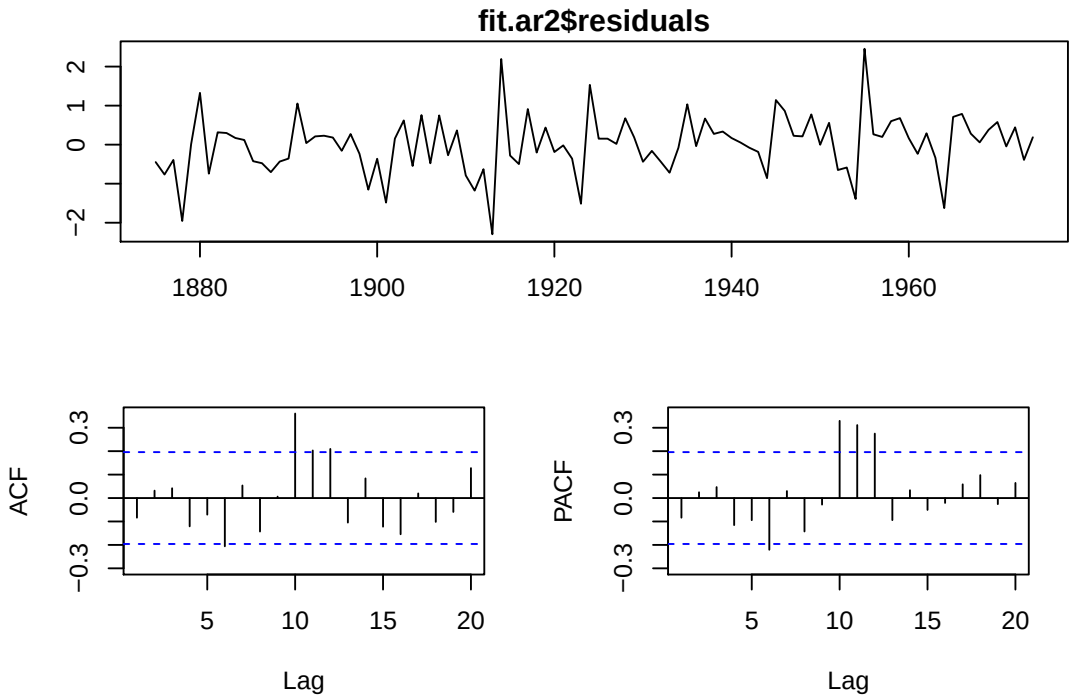
Based on the ACF (damped sinusoid) and PACF (cut-off) we would try the following models: AR(2), AR(9), AR(10). If we also consider the AIC scores for different p, then

```
> fit.aic <- ar.burg(lsunspot100)
> plot(0:fit.aic$order.max, fit.aic$aic)
```



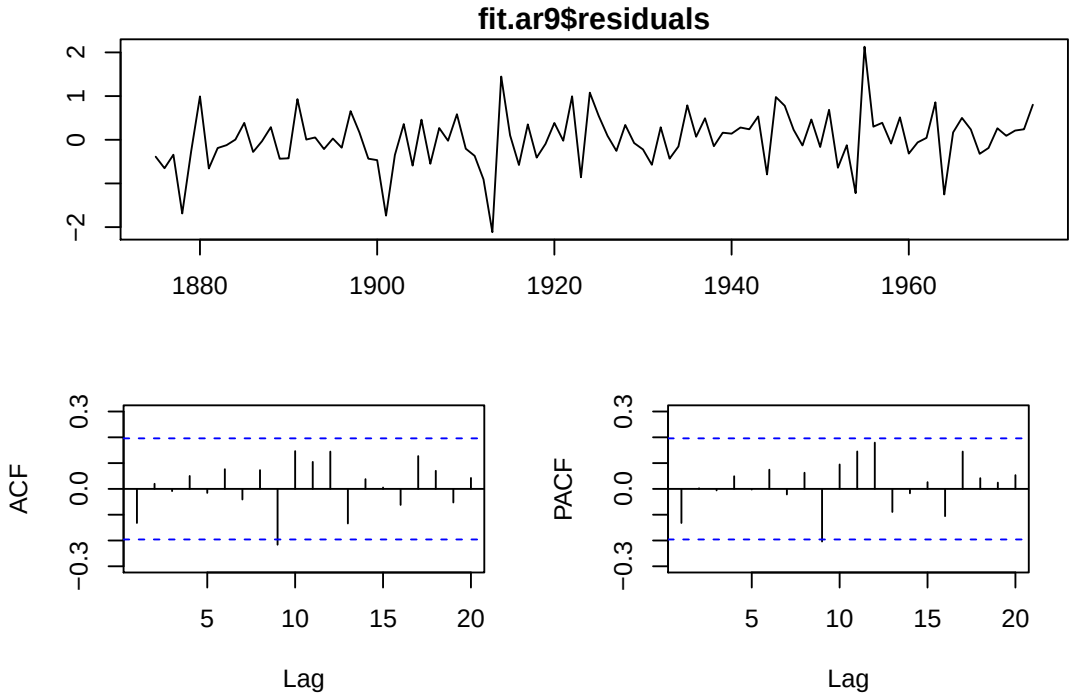
we would choose AR(10), which has the lowest AIC. Let us also have a look at the residuals of the different fits:

```
> fit.ar2 <- arima(lsunspot100, order = c(2,0,0))
> tsdisplay(fit.ar2$residuals, points = FALSE)
```

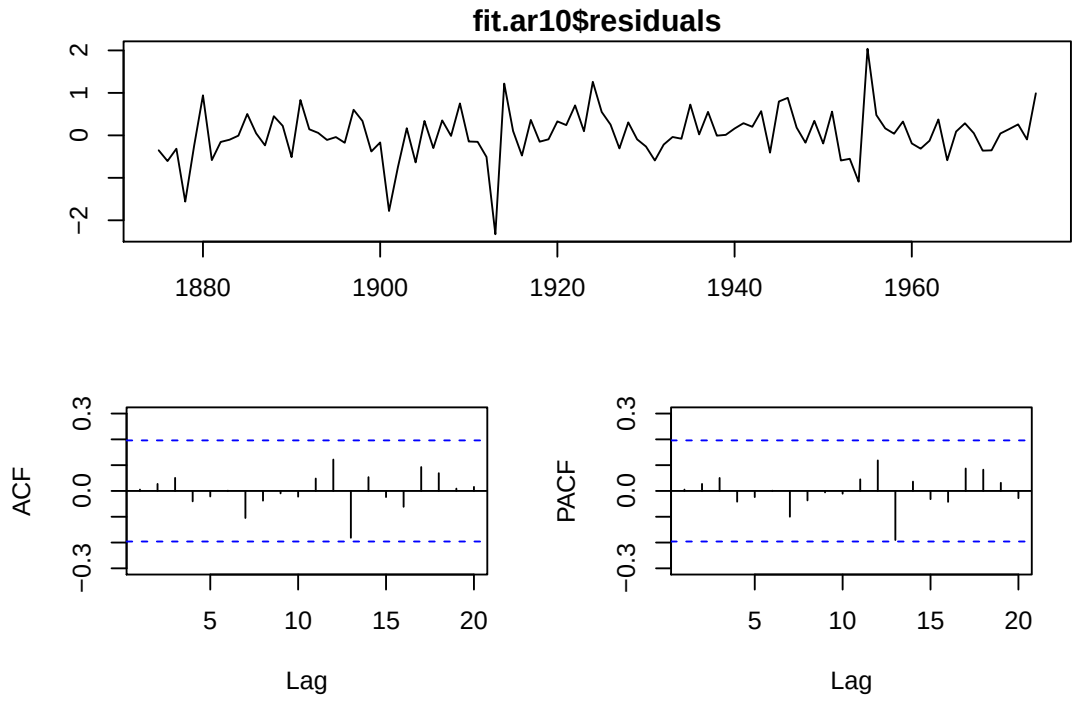


There is still some dependency left in the residuals of an AR(2) process, hence let us have a look at the residuals of the higher order fits:

```
> fit.ar9 <- arima(lsunspot100, order = c(9,0,0))
> tsdisplay(fit.ar9$residuals, points = FALSE)
```

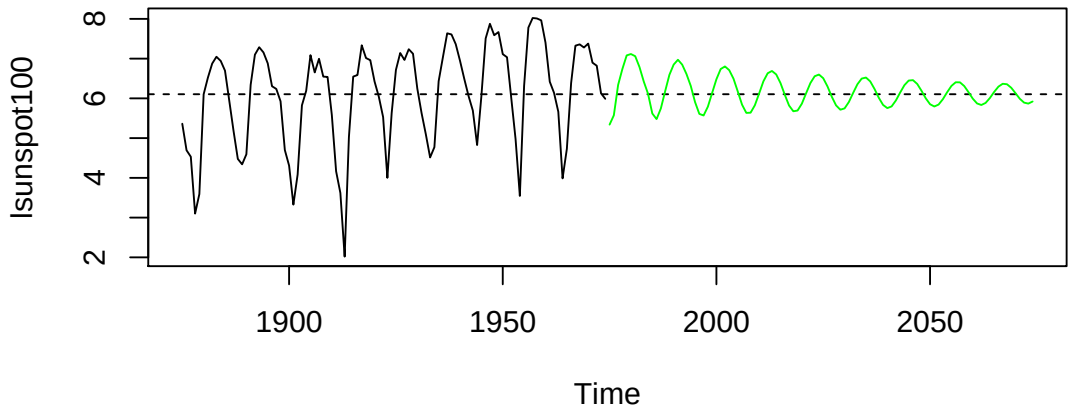



```
> fit.ar10 <- arima(lsunspot100, order = c(10,0,0))
> tsdisplay(fit.ar10$residuals, points = FALSE)
```



The residuals look best for the AR(10), which also has the lowest AIC, hence we choose AR(10).

c)
```
> pred <- predict(fit.ar10, n.ahead = 100)
> plot(lsunspot100, xlim = c(1875, 2074))
> abline(h = fit.ar10$coef["intercept"], lty = 2)
> lines(pred$pred, col = "green")
```



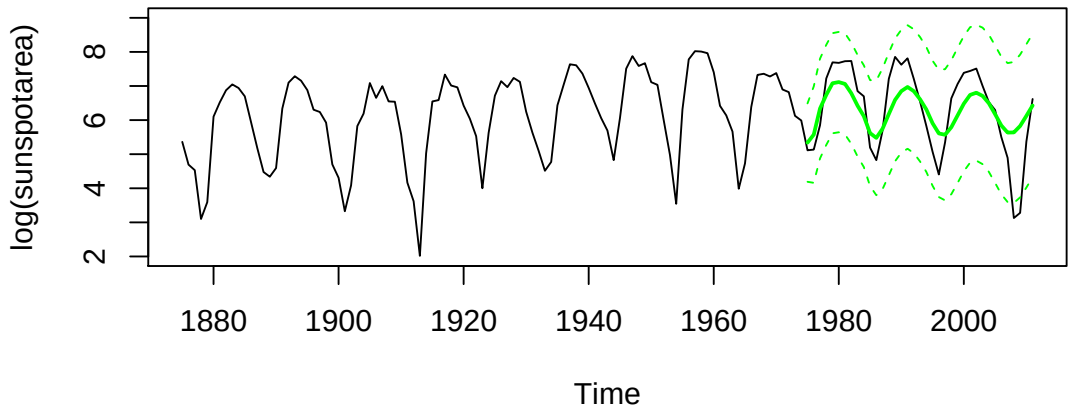
We see that the predictions converge to the global mean. This is due to the definition of the k-step forecast for an AR(p) process, which is defined recursively:

$$\begin{aligned}
\text{1-step forecast: } & \hat{X}_{n+1;1:n} = \alpha_1 x_n + ... + \alpha_p x_{n+1-p} \\
\text{2-step forecast: } & \hat{X}_{n+2;1:n} = \alpha_1 \hat{X}_{n+1;1:n} + \alpha_2 x_n + ... + \alpha_p x_{n+2-p} \\
& = \alpha_1(\alpha_1 x_n + ... + \alpha_p x_{n+1-p}) + \alpha_2 x_n + ... + \alpha_p x_{n+2-p} \\
& \text{etc.}
\end{aligned}$$

We see that in each forecasting step, some AR-coefficients are multiplied, and since they are smaller than 1 in absolute value for a stationary AR(p)-process (Note that we only showed this for an AR(1)-process in class, but this also holds for general AR(p)-processes), the forecasts get smaller and smaller the further they are in the future.

d)
```
> pred37 <- predict(fit.ar10, n.ahead = 37)
> plot(log(sunspotarea), ylim = c(2,9))
> lines(pred37$pred, col = "green", lwd = 2)
```

```
> lines(pred37$pred + 1.96 * pred37$se, col = "green", lty = 2)
> lines(pred37$pred - 1.96 * pred37$se, col = "green", lty = 2)
```



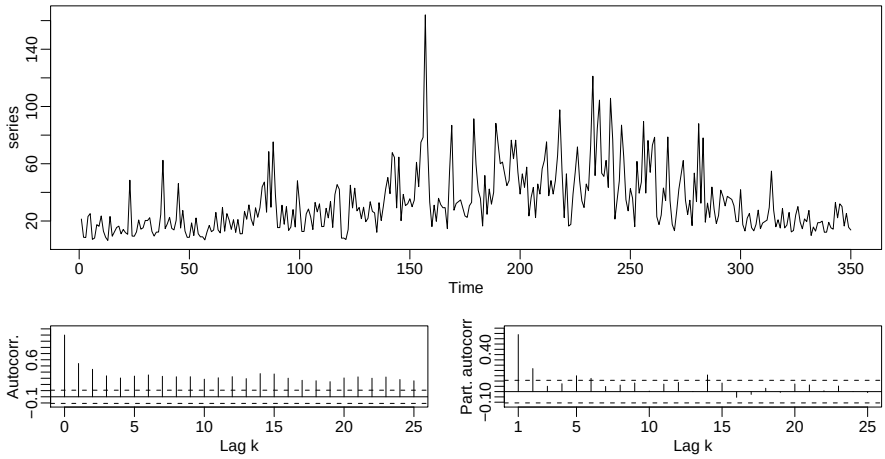
We observe that our prediction catches the (deterministic) seasonal pattern of the time series quite well, but the peaks of the time series are usually underestimated. (One reason is, as we have seen in the previous subtask, that the forecasts converge to the global mean of the time series). Furthermore, the observed time series lies (except for the peak in the end) within the prediction interval.
The mean squared forecasting error is given by:

```
> pred.error <- window(log(sunspotarea), start = 1975, end = 2011) - pred37$pred
> (pred.mse <- mean(pred.error^2) )
```

```
[1] 0.7557876
```

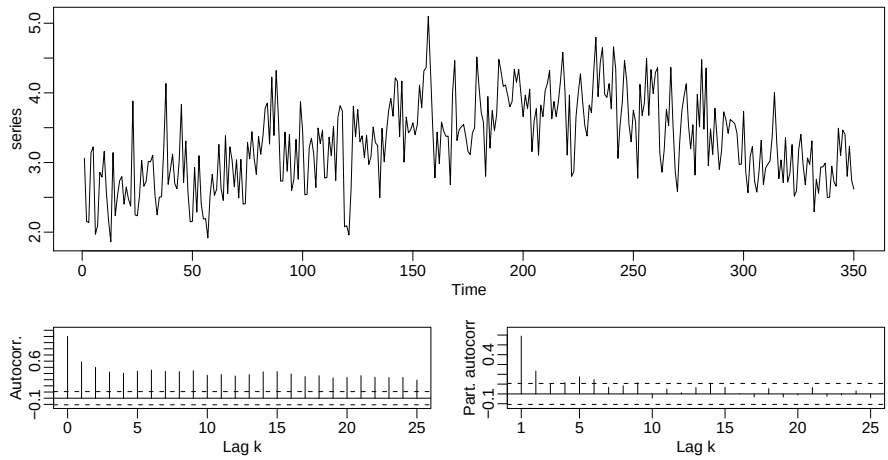
2. **a)** In some areas the variance is much smaller than in others. The "peak" in the middle indicates that a logarithmic transformation must first be applied to the data. If we look at the correlogram, we notice that the ordinary autocorrelations decay far too slowly. Even for large lags, they still lie outside the confidence band:

```
> plot(d.varve)
> acf(d.varve)
> pacf(d.varve)
```



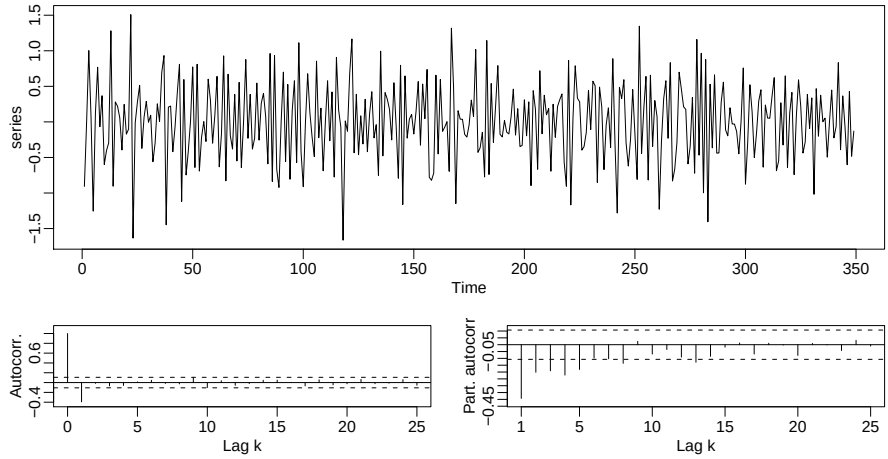
Even the time series of logarithmic data cannot yet be regarded as stationary, since it exhibits clear trends (first increasing, then decreasing), which can however be eliminated by taking first differences:

```
> plot(log(d.varve))
> acf(log(d.varve))
> pacf(log(d.varve))
```



The plot of first differences for the transformed series shows that stationarity can now be assumed:

```
> plot(diff(log(d.varve)))
> acf(diff(log(d.varve)))
> pacf(diff(log(d.varve)))
```



b) The correlogram plotted in part a) indicates an ARIMA(1,1,1) process (or perhaps an ARIMA(0,1,1) process). Fitting these two models, we see that the ARIMA(1,1,1) model is very good at describing the logarithmic data. In both fitted models, the algorithm converges; of the two models, ARIMA(1,1,1) has a smaller AIC.

The estimated coefficients are $\widehat{\beta}_1 = -0.84$ for the fitted ARIMA(0,1,1) model and $\widehat{\alpha}_1 = 0.25$, $\widehat{\beta}_1 = -0.91$ for the fitted ARIMA(1,1,1) model. For both models, the estimated mean is $\widehat{\mu} = -0.00127$, which leads us to assume the data do not need correcting by their mean. Furthermore, the estimated error variances are 0.224 (for the ARIMA(0,1,1) model) and 0.2138 (for the ARIMA(1,1,1) model).

Thus the ARIMA(0,1,1) model looks as follows:

$$Y_t = X_t - X_{t-1}$$

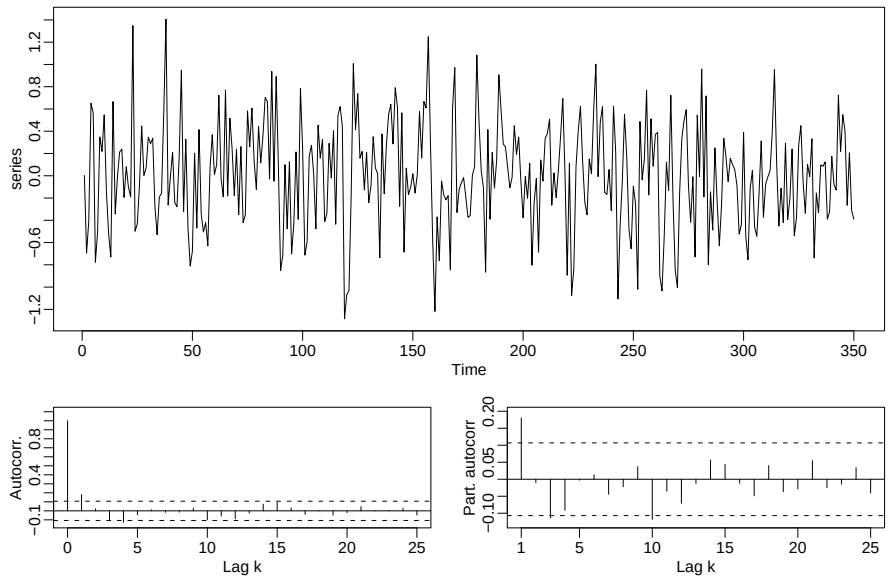
$$Y_t = E_t - 0.84 \cdot E_{t-1}; \quad \sigma^2_{E_t} = 0.224$$

For the ARIMA(1,1,1) model, we similarly have

$$Y_t = X_t - X_{t-1}$$

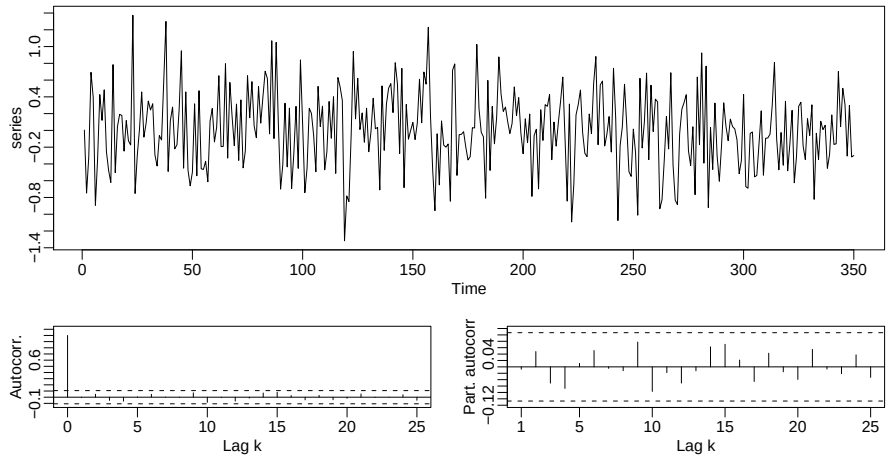
$$Y_t = 0.25 \cdot Y_{t-1} + E_t - 0.91 \cdot E_{t-1}; \quad \sigma^2_{E_t} = 0.214$$

Residuals of the fitted ARIMA(0,1,1) process:



The first ordinary (=first partial) autocorrelation clearly lies outside the confidence band. Thus the residuals cannot be considered independent.

Residuals of the fitted ARIMA(1,1,1) process:



These residuals no longer exhibit any undesired structure.

R commands:

ARIMA(0,1,1) model:

```
> mean(diff(log(d.varve)))                                  # -0.001271813
> r.varve.m1 <- arima(log(d.varve), order=c(0,1,1))
> r.varve.m1$code                                           # Code = 0, i.e. convergence
> r.varve.m1
```

Result:

```
   Call:
   arima(x = log(d.varve), order = c(0, 1, 1))

   Coefficients:
             ma1
         -0.8421
   s.e.   0.0411

   sigma^2 estimated as 0.224:  log likelihood = -234.77,  aic = 473.53
> plot(resid(r.varve.m1))
> acf(resid(r.varve.m1))
> pacf(resid(r.varve.m1))
```

ARIMA(1,1,1) model:

```
> r.varve.m2 <- arima(log(d.varve),order=c(1,1,1))
> r.varve.m2$code                                           # Code = 0, i.e. convergence
```

```
> r.varve.m2
```

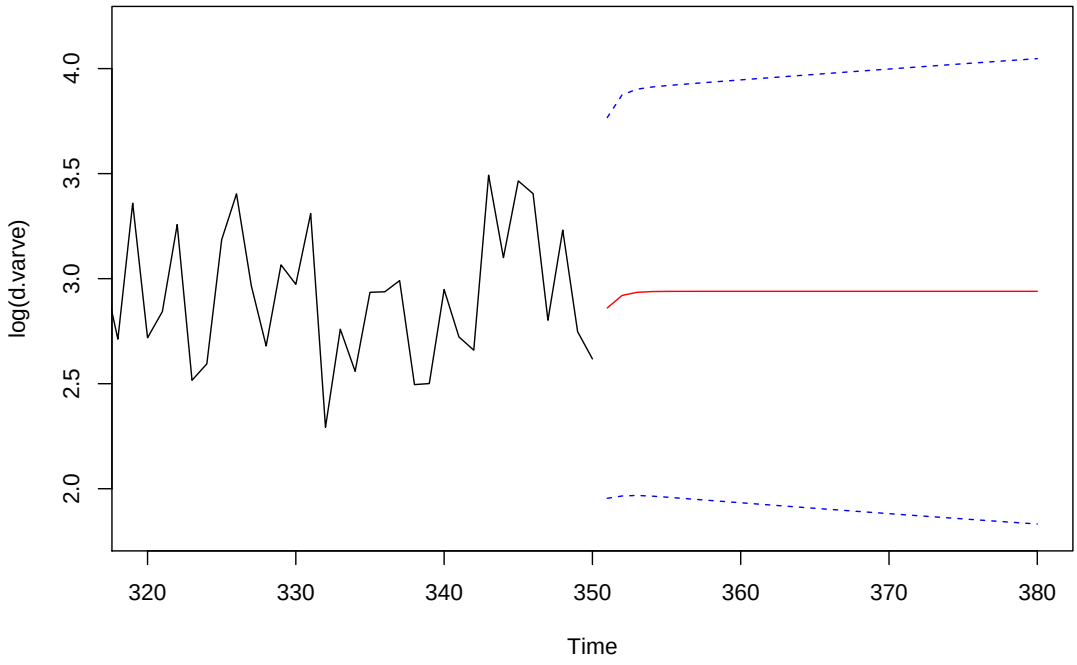
Result:

```
   Call:
   arima(x = log(d.varve), order = c(1, 1, 1))

   Coefficients:
            ar1      ma1
         0.2461  -0.9140
   s.e.  0.0590   0.0234

   sigma^2 estimated as 0.2138:  log likelihood = -226.65,  aic = 459.3
> f.acf(resid(r.varve.m2))  > acf(resid(r.varve.m2))
> pacf(resid(r.varve.m2))
```

c) The predicted value stabilizes after just a few points in time, and the prediction band grows linearly. Both these properties were to be expected from the theory (cf. lecture notes).

R code:

```
> ## Fitting the model
> d.varve <- ts(d.varve)
> r.varve <- arima(log(d.varve), order=c(1,1,1))
> r.pred <- predict(r.varve, n.ahead=30)                    ## Prediction
> plot(log(d.varve), xlim=c(320,380), ylim=c(1.8, 4.2))     ## Plotting
> lines(r.pred$pred, col=2)
> lines(r.pred$pred + 1.96 * r.pred$se, lty=2, col=4)
> lines(r.pred$pred - 1.96 * r.pred$se, lty=2, col=4)
```

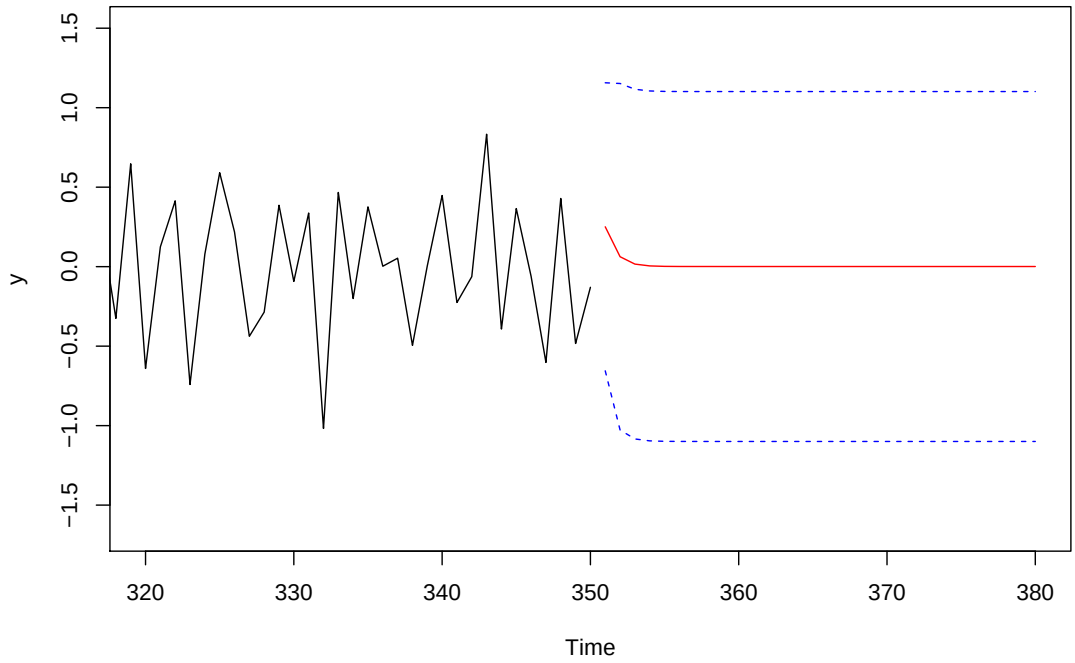


d) Here the predicted change is also quite quick to stabilize at the mean of first-order differences. The prediction variance is constant after just a few time points, and is only marginally smaller than the variance of the process (cf lecture notes).

R code:

```
> y <-  diff(log(d.varve))
> ## Fitting the model
> r.varve.2 <- arima(y, order=c(1,0,1))
> r.pred.2 <- predict(r.varve.2, n.ahead=30)   ## Prediction
> plot(y, xlim=c(320,380))                     ## Plotting
> lines(r.pred.2$pred, col=2)
```

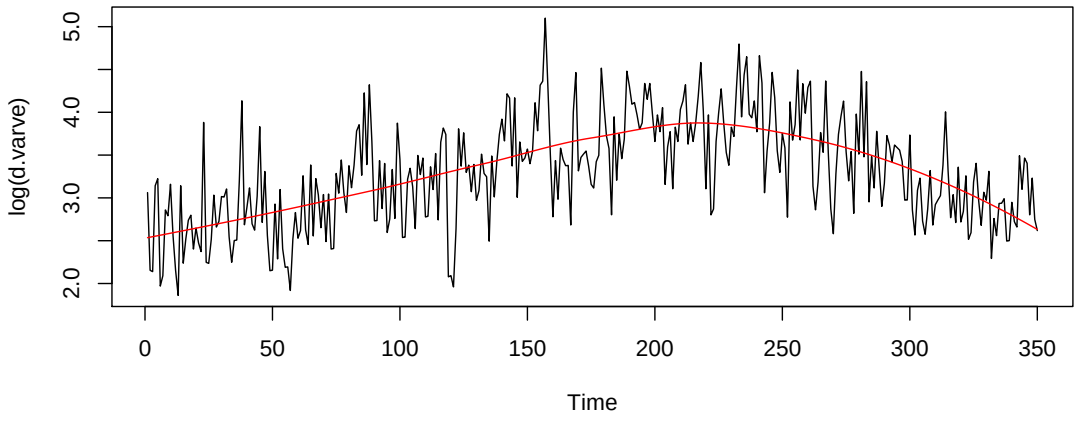
```
> lines(r.pred.2$pred + 1.96 * r.pred.2$se, lty=2, col=4)
> lines(r.pred.2$pred - 1.96 * r.pred.2$se, lty=2, col=4)
```



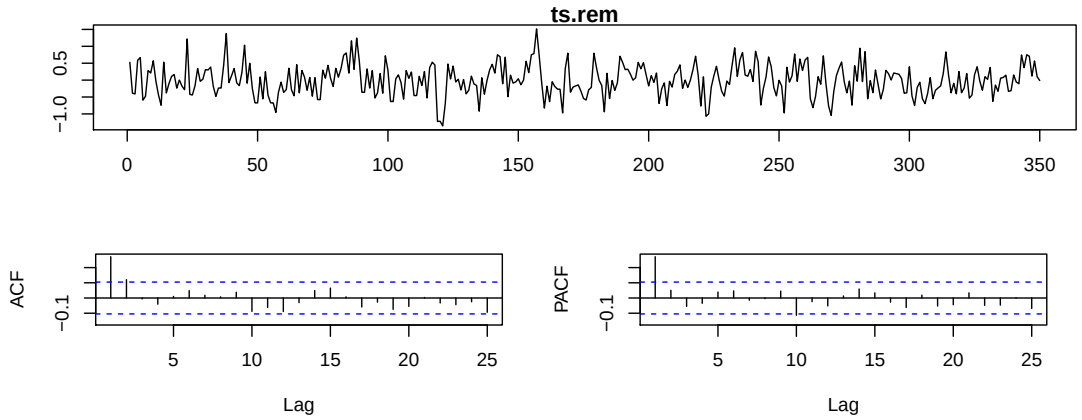
e) For Part **c)** the predicted values need only undergo an exp transformation. This means if we calculate `exp(r.pred$pred)`, we get the predicted values on the original scale.
However, as mentioned on the blackboard in the lecture, if you truly want an unbiased estimate, one would have to add the standard error of the squared prediction as follows `exp(r.pred$pred + 1/2 s.e.(r.pred$pred`^{2}`)`.
In Part **d**), *however, the formation of differences needs reversing (by taking the cumulative sum). If we apply* `predic`
`transformation and then add the last known entry of the timeseries: exp(cumsum(r.pred.2$pred)`

f) `We fit a loess smoother for the trend and look at the residuals`

```
> plot(log(d.varve))
> time.varve <- time(log(d.varve))
> fit.loess <- loess(log(d.varve) ~ time.varve)
> lines(as.numeric(time(log(d.varve))), fitted(fit.loess), col = "red")
```

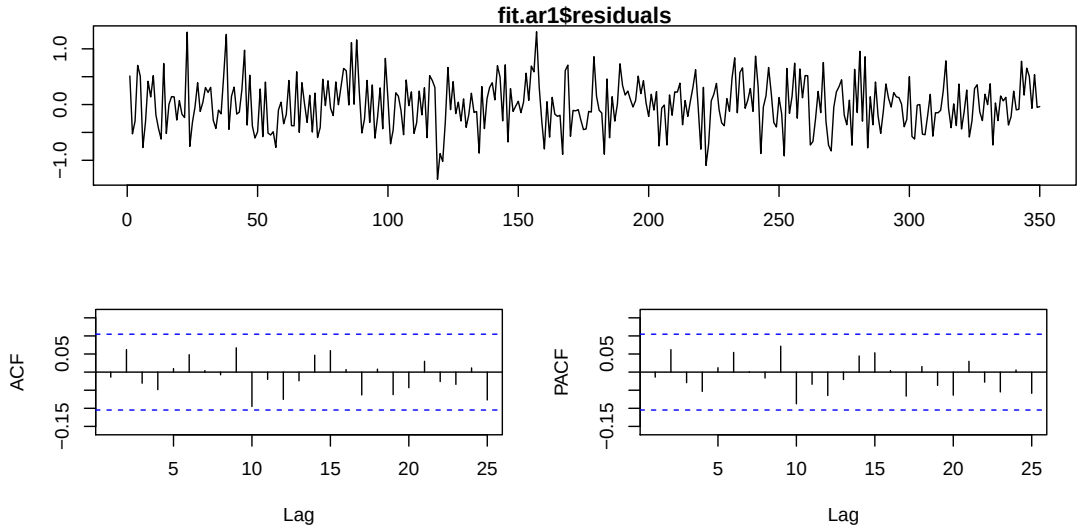



```
> ts.rem <- log(d.varve) - fitted(fit.loess)
> tsdisplay(ts.rem, points=FALSE)
```

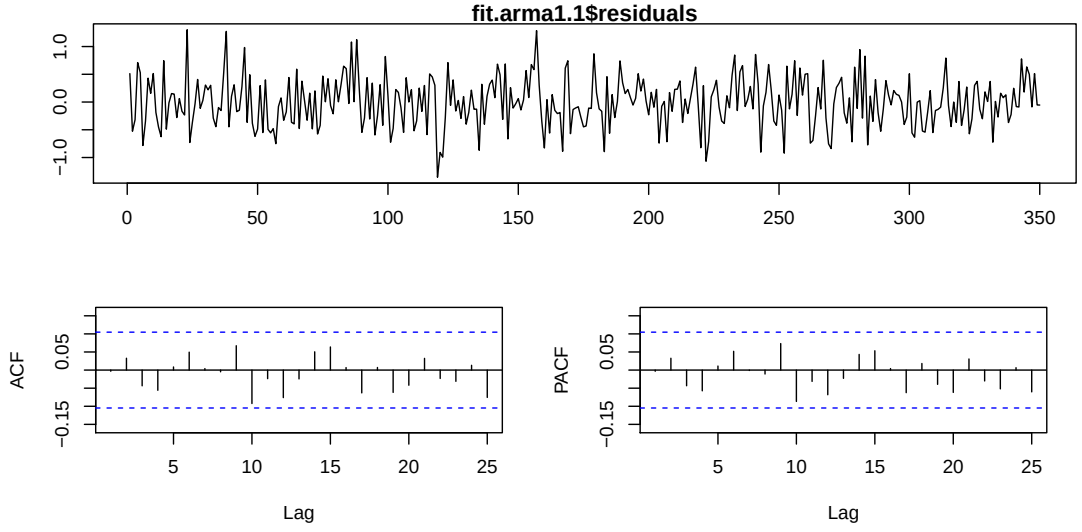


The remainder looks stationary. No seasonal effect seems present.
Based on the ACF and PACF, we choose to fit an AR(1) and an ARMA(1,1) model.

```
> fit.ar1 <- arima(ts.rem, order = c(1,0,0))
> tsdisplay(fit.ar1$residuals, points = FALSE)
> fit.ar1$aic
[1] 437
```

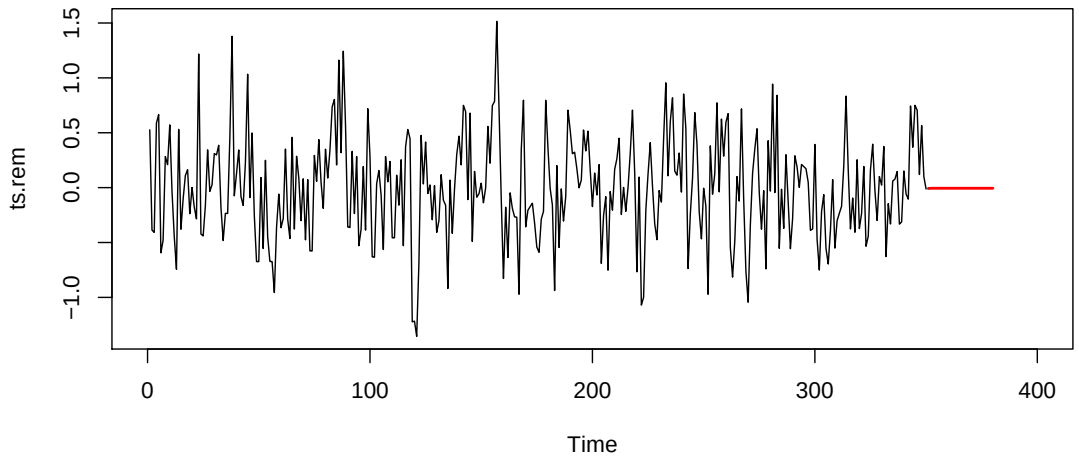



```
> fit.arma1.1 <- arima(ts.rem, order = c(1,0,1))
> tsdisplay(fit.arma1.1$residuals, points = FALSE)
> fit.arma1.1$aic
[1] 438
```



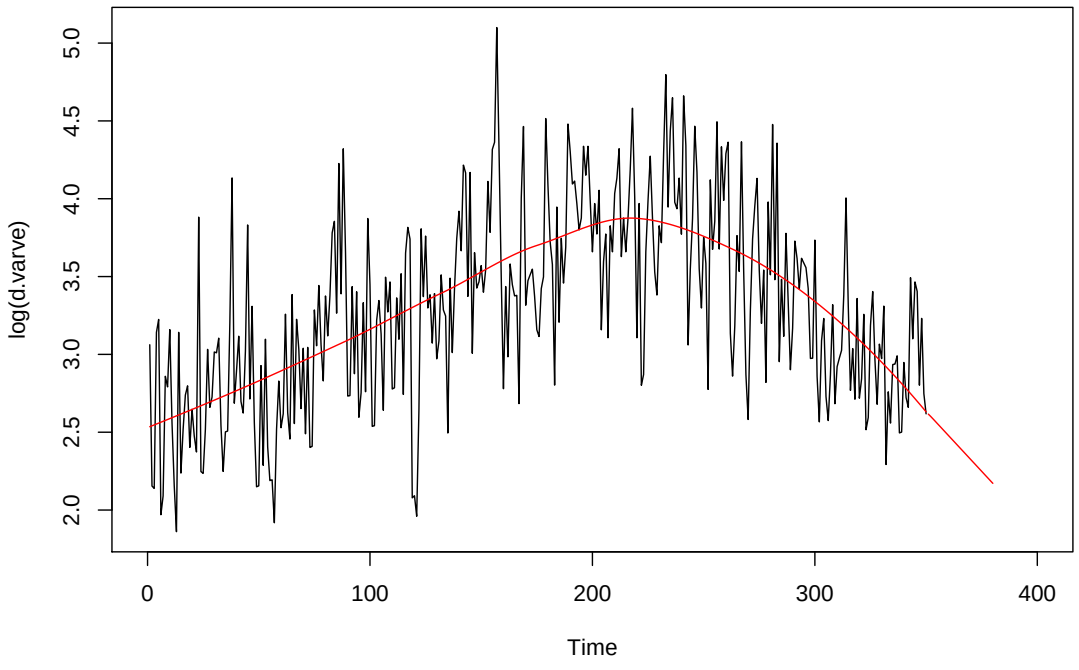
The residuals look good in both cases (no dependence left in ACF and PACF), and AR(1) has a lower AIC, hence we choose the AR(1)-model.

```
> rem.pred <- predict(fit.ar1, n.ahead = 30)
> plot(ts.rem, xlim = c(0, 400))
> lines(rem.pred$pred, col = "red", lwd = 2)
```



Now, as recommended in the script on page 147, we fit a line based on the last n.ahead = 30 trend values. Then, we perform linear extrapolation using this line with intercept/start point the last value of the trend we fitted.

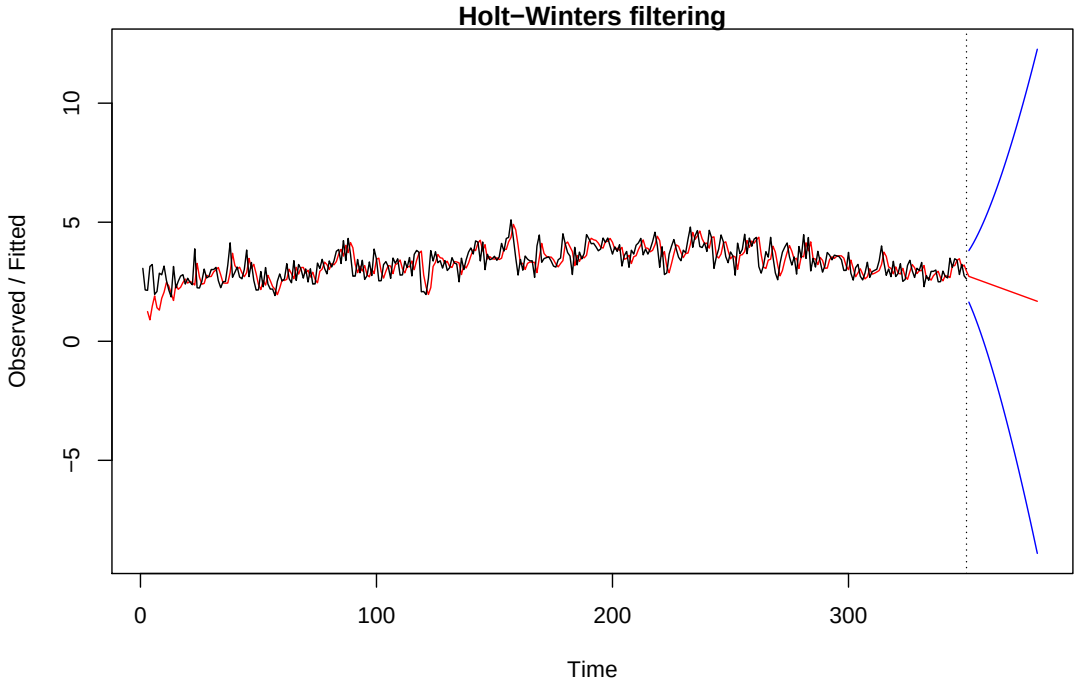
```
> starting.point <- tail(predict(fit.loess),n=1)
> yy <- tail(fitted(fit.loess),n=30)##fit the last 30 values of the trend
> xx <- tail(time(log(d.varve)),n=30)
> fit.regr <- lm(yy~xx)
> trend.forecast <- predict(fit.regr,newdata=data.frame(xx=(max(time(log(d.varve)))+1:30)))
> ##Or alternatively
> ##trend.forecast <- starting.point + (1:30) * coef(fit.regr)[2]
>
> plot(log(d.varve),
      xlim=c(0,400))
> lines(as.numeric(time(log(d.varve))), fitted(fit.loess), col = "red")
> lines((max(time(log(d.varve)))+1:30),
        trend.forecast + rem.pred$pred,
        col="red")
```



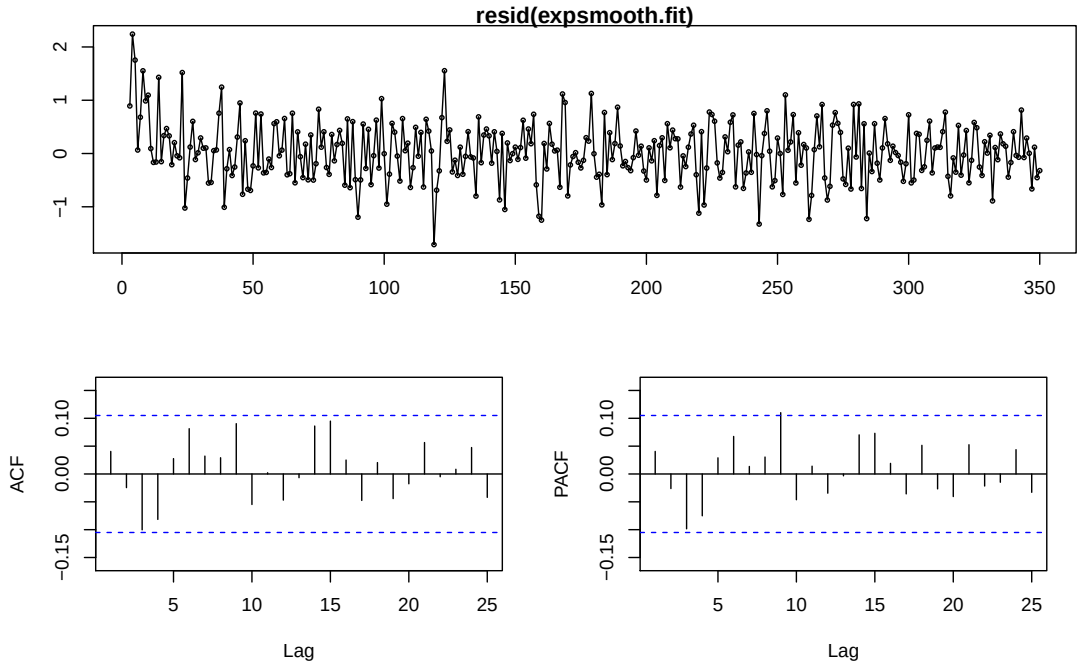
About prediction intervals: (as described on page 170 in the script) most of the uncertainty in the prediction comes from the trend extrapolation, for which it is not possible to give a reasonable uncertainty interval! It maybe be more genuine here to not provide an interval at all. There is no way to compute correct prediction intervals here.

g) We fit exponential smoothing

```
> expsmooth.fit <- HoltWinters(log(d.varve), gamma=FALSE)
> t.pr <- predict(expsmooth.fit, 30, prediction.interval=T)
> plot(expsmooth.fit, t.pr)
```




```
> tsdisplay(resid(expsmooth.fit))
```

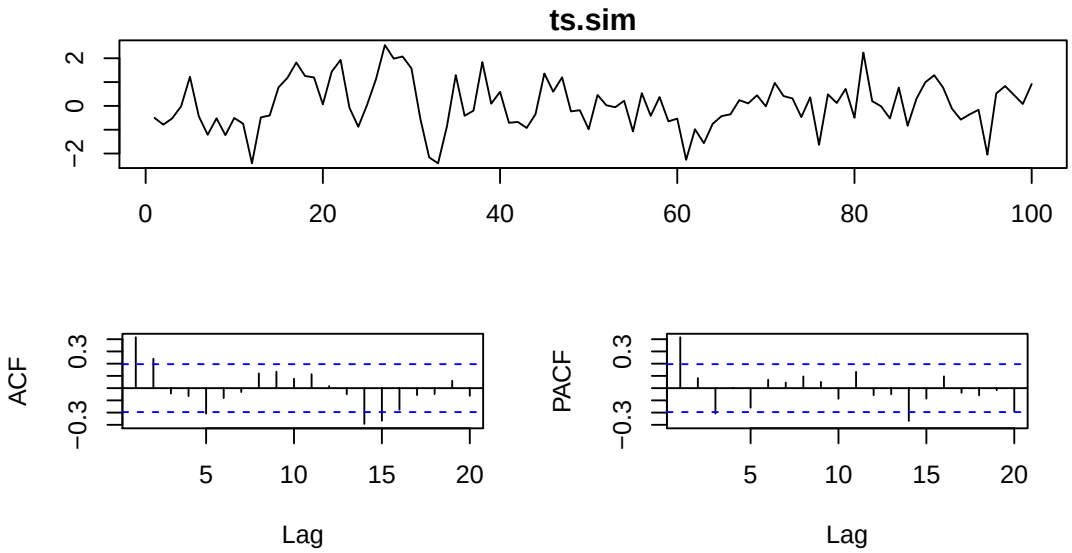


The residuals look ok except for the one peak at lag 9 in the pacf plot.

h) We leave this as an exercise to reader.

3.

```
> set.seed(5)
> ts.sim <- arima.sim(model = list(ar = c(0.3)), n = 100)
```



a) Based on the ACF (damped sinusoid) and PACF (cut-offs at lag 1, 3 and 14) we fit an AR(1), an AR(3) and an AR(14)-process:

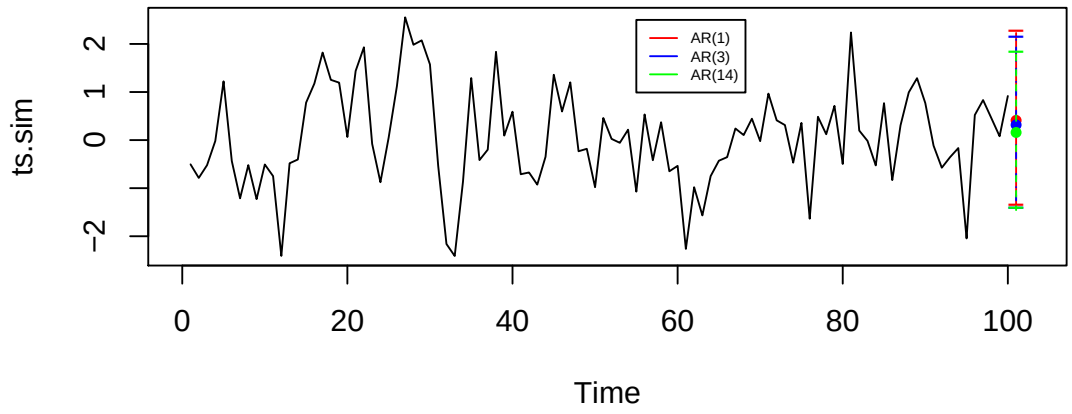
```
> (fit1 <- arima(ts.sim, order = c(1,0,0)) )

Call:
arima(x = ts.sim, order = c(1, 0, 0))

Coefficients:
         ar1  intercept
      0.4160     0.0487
s.e.  0.0906     0.1577

sigma^2 estimated as 0.8605:  log likelihood = -134.47,  aic = 274.95
> (fit3 <- arima(ts.sim, order = c(3,0,0)) )

Call:
arima(x = ts.sim, order = c(3, 0, 0))

Coefficients:
         ar1     ar2      ar3  intercept
      0.3979  0.1601  -0.2032     0.0492
s.e.  0.0974  0.1040   0.0971     0.1399

sigma^2 estimated as 0.8178:  log likelihood = -132,  aic = 274
> (fit14 <- arima(ts.sim, order = c(14,0,0)) )

Call:
arima(x = ts.sim, order = c(14, 0, 0))

Coefficients:
         ar1     ar2      ar3     ar4      ar5     ar6
      0.3946  0.1252  -0.1209  0.0274  -0.1791  0.0968
s.e.  0.0944  0.1027   0.1025  0.1023   0.1039  0.1078
          ar7     ar8      ar9     ar10    ar11    ar12
      -0.0199  0.1323  -0.0038  -0.1351  0.0985  0.0383
s.e.   0.1124  0.1093   0.1117   0.1093  0.1096  0.1129
         ar13     ar14  intercept
      0.0573  -0.3024     0.0603
s.e.  0.1113   0.1010     0.1082

sigma^2 estimated as 0.6794:  log likelihood = -123.72,  aic = 279.44
```

We observe that with increasing order p, the estimated error variance decreases. In terms of (1-step) prediction intervals, which for AR(p)-processes depend on the estimated standard deviation of the error, $\hat{\sigma}_E$, this implies that the higher p is, the narrower the prediction interval gets.

b)
```
> pred1 <- predict(fit1, n.ahead = 1)
> pred3 <- predict(fit3, n.ahead = 1)
> pred14 <- predict(fit14, n.ahead = 1)
> PI1 <- pred1$pred[1] + c(-1, 1)*1.96 * pred1$se[1]
> PI3 <- pred3$pred[1] + c(-1, 1)*1.96 * pred3$se[1]
> PI14 <- pred14$pred[1] + c(-1, 1)*1.96 * pred14$se[1]
> (width1 <- PI1[2] - PI1[1])
[1] 3.636213
> (width3 <- PI3[2] - PI3[1])
[1] 3.544871
> (width14 <- PI14[2] - PI14[1])
[1] 3.231066
```

As we reasoned in the previous subtask, the prediction intervals do indeed get narrower with increasing p. This can also be seen in the following plot:

```
> plot(ts.sim, xlim = c(0, 103))
> points(pred1$pred, col = "red", pch = 20)
> points(pred1$pred + 1.96 * pred1$se, col = "red", pch = "-")
> points(pred1$pred - 1.96 * pred1$se, col = "red", pch = "-")
> lines(x = c(101,101), y = c(pred1$pred + 1.96 * pred1$se,
                              pred1$pred - 1.96 * pred1$se), col = "red")
> points(pred3$pred, col = "blue", pch = 20)
> points(pred3$pred + 1.96 * pred3$se, col = "blue", pch = "-")
> points(pred3$pred - 1.96 * pred3$se, col = "blue", pch = "-")
> lines(x = c(101,101), y = c(pred3$pred + 1.96 * pred3$se,
                              pred3$pred - 1.96 * pred3$se), col = "blue", lty = 4)
> points(pred14$pred, col = "green", pch = 20)
> points(pred14$pred + 1.96 * pred14$se, col = "green", pch = "-")
> points(pred14$pred - 1.96 * pred14$se, col = "green", pch = "-")
> lines(x = c(101,101), y = c(pred14$pred + 1.96 * pred14$se,
                              pred14$pred - 1.96 * pred14$se), col = "green", lty =2)
> legend(55, 2.5, legend = c("AR(1)", "AR(3)", "AR(14)"),
        col = c("red", "blue", "green"), lwd = c(1,1,1), cex = 0.5)
```



c)
```
> # Simulation
> set.seed(1)
> # Initialization
> coverage.1 <- numeric(length = 100)
> coverage.5 <- numeric(length = 100)
> coverage.10 <- numeric(length = 100)
> ms.pred.error.1 <- numeric(100)
```

```
> ms.pred.error.5 <- numeric(100)
> ms.pred.error.10 <- numeric(100)
> for(i in 1:100){
   # simulate AR(1) of length 101
   sim <- arima.sim(model = list(ar = c(0.3)), n = 101)

   # fit AR(1), AR(5) and AR(10)
   fit1 <- arima(window(sim, start = 1, end = 100), order = c(1,0,0))
   fit5 <- arima(window(sim, start = 1, end = 100), order = c(5,0,0))
   fit10 <- arima(window(sim, start = 1, end = 100), order = c(10,0,0))

   # 1-step prediction:
   pred1 <- predict(fit1, n.ahead = 1)
   pred5 <- predict(fit5, n.ahead = 1)
   pred10 <- predict(fit10, n.ahead = 1)

   # 1-step prediction interval:
   PI1 <- pred1$pred[1] + c(-1, 1)*1.96 * pred1$se[1]
   PI5 <- pred5$pred[1] + c(-1, 1)*1.96 * pred5$se[1]
   PI10 <- pred10$pred[1] + c(-1, 1)*1.96 * pred10$se[1]

   # MSE
   ms.pred.error.1[i] <- (sim[101] - pred1$pred[1])^2
   ms.pred.error.5[i] <- (sim[101] - pred5$pred[1])^2
   ms.pred.error.10[i] <- (sim[101] - pred10$pred[1])^2

   # Is true 101th obs contained in prediction interval?
   coverage.1[i] <- ifelse(sim[101] >= PI1[1] & sim[101] <= PI1[2], TRUE, FALSE)
   coverage.5[i] <- ifelse(sim[101] >= PI5[1] & sim[101] <= PI5[2], TRUE, FALSE)
   coverage.10[i] <- ifelse(sim[101] >= PI10[1] & sim[101] <= PI10[2], TRUE, FALSE)
 }
> # How often in 100 times is true value contained in prediction interval?
> sum(coverage.1)
[1] 92
> sum(coverage.5)
[1] 90
> sum(coverage.10)
[1] 86
> # Mean squared error averaged over 100 simulations
> mean(ms.pred.error.1)
[1] 1.161098
> mean(ms.pred.error.5)
[1] 1.341005
> mean(ms.pred.error.10)
[1] 1.437393
```

We see that the higher p is, the worse the coverage gets (for an AR(1), in 92% of the cases the true observation x_{101} is contained in the prediction interval, for an AR(5), the coverage is 90% and for an AR(10) only 86% of the prediction intervals contain the true value). Furthermore, the mean forecasting error increases from 1.16 (AR(1)) to 1.44 (AR(10)).
This shows that overfitting (i.e. fitting a higher order AR(p) to a time series coming from an AR(1) process) doesn't pay off, i.e. the narrower prediction intervals for overfitted AR(p)-processes are misleading: A narrower prediction interval does not necessarily mean better prediction.