Series 5

1. In this exercise, we look at the time series `sunspotarea`, which is available in the package `fpp`. It contains yearly data from 1875-2011, where for each year the average over the daily sunspot areas (in units of millionths of a hemisphere) for the full sun is given. Sunspots are magnetic regions that appear as dark spots on the surface of the sun.

a) Plot the time series. Why does it make sense to log-transform the time series?

b) From now on, we work with the log-transformed series and we use only the first 100 observations (1875 - 1974). Choose a suitable AR-model for this time series.
R-Hint: `window()`

c) For your chosen model, predict the next 100 observations of the log-transformed time series and plot them together with the log-transformed time series. Also add a line for the estimated global mean to the plot. What do you observe?

R hint: `predict(....,n.ahead=...)`

d) Perform an out-of-sample evaluation, i.e. compare your prediction with the last 37 observed values of the time series. Plot the full log-transformed time series (1875 - 2011) and add your prediction (1975 - 2011) as well as prediction intervals to the plot and comment on the plot. Also compute the mean squared forecasting error of your prediction:

$$\frac{1}{k}\sum_{i=1}^{k}(x_{n+i} - \hat{X}_{n+i;1:n})^2$$

2. During their yearly spring melt, glaciers deposit layers of sand and mud. These annual sediments, known as *varves*, can be reconstructed in New England for the whole time between the beginning (about 12'600 years ago) till the end (6'000 years ago) of glacial retreat. From these varves, approximations of paleoclimatic parameters can in turn be computed, such as temperature (a warmer year yields more sediment).

In the dataset `varve.dat`, you will find 350 annual sediment diameters (contained in lines 201 through 550) starting at 11'660 years ago. After loading these data, first construct a time series object from them:

```
> t.url <- "http://stat.ethz.ch/Teaching/Datasets/WBL/varve.dat"
> d.varve <- ts(scan(t.url)[201:550])
```

Comment: The procedure `scan()` is a more general data loading function than `read.table()`. We use it here to avoid putting the data into a data frame. Do not worry about the exact choice of procedure for reading data here: simply believe us when we say that `scan()` does what we need, or read the help file.

a) Our data analyst suggested to log transform the data before continuing with the analysis. Does that make sense? Argue based on the time series plots.
She also argued that it is required to take first differences to end up with a stationary time series. Do you agree? Why?

b) Fit an $ARIMA(p, d, q)$ model on the transformed (but not differenced) data with reasonable values for p, d and q. By reasonable we mean possibly low values for p and q and a sensible d such that the residuals look the way we want them to.

c) Use the $ARIMA(1,1,1)$ model on the log transformed values to compute predictions for the **next 30 values**. Plot the predicted value and the prediction intervals. What do you notice? (What happens to the prediction the further you go into the future? What happens to the confidence bands?)

R hint: `predict(....,n.ahead=...)`, `plot()`, `lines()`

d) Compute the first-order differences `y = diff(log(d.varve))`, and then fit an $ARMA(1,1)$ process to them. For this new series y, now, make and plot predictions based on the $ARMA(1,1)$ model. What do you notice when comparing these to part **c)**? (Does the prediction and do the confidence bands behave in the same way?)

e) Which transformations are necessary if we want to obtain predictions for the original time series? Describe how to proceed (without actually performing the calculations you describe) when using the data from parts **c)** and **d)**, respectively.

f) Perform the decomposition approach for prediction by fitting a loess smoother for the trend

```
> fit.loess <- loess(...)
> ts.rem <- ... - fitted(fit.loess)
```

and finding a suitable model for the residuals. Is there some seasonal effect in the data that we have to model? Is the remainder stationary?

You can compare and get ideas from the decomposition approach described in the script on page 146.

Predict 30 values in the future and plot them. How would you compute correct prediction intervals here?

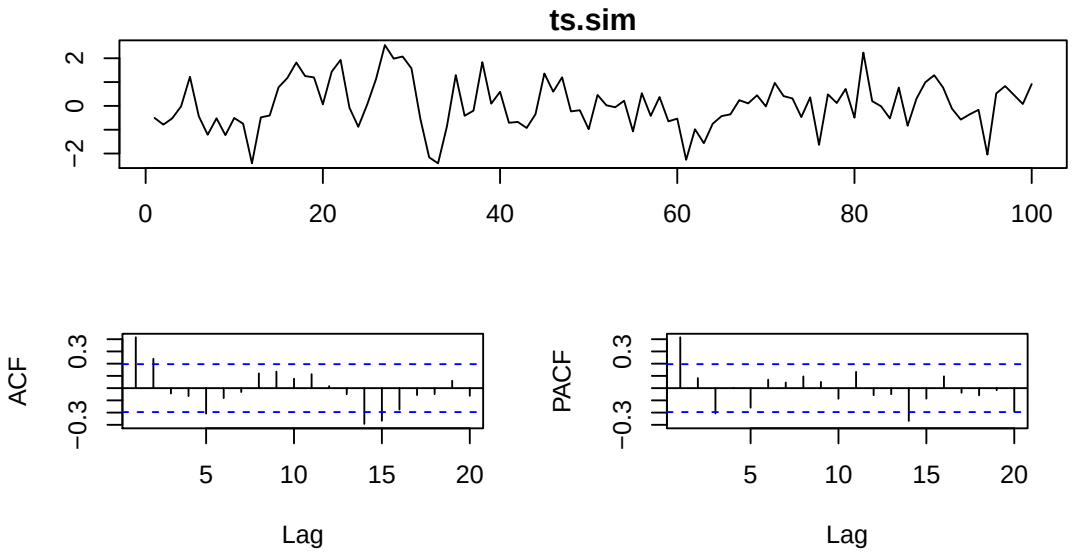
Note if you try to fit stl instead of loess it will let you know if it can't find any periodicity. This might be an extra indication to confirm your conclusions about seasonal effects.

g) Now use exponential smoothing. Again make predictions, plot the predicted timeseries, including confidence bounds, and check the residuals.
R-hint: `?HoltWinters`, set the option `gamma=FALSE` if you want to fit a non-seasonal model.

h) (Optional) Perform an out-of-sample evaluation of the different prediction methods. Leave out the last 20 values of the original time series and compute the mean squared forecasting error as done in `1.d)`.

3. In this exercise, we want to investigate the impact of overfitting on prediction. To this end, we simulate a time series from an AR(1) process as follows:

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



a) Suppose we didn't know the data generating time series process and decide to fit three different AR-models. Which models would you choose? Fit the suggested models. If you increase the order p, how does the estimated error variance $\widehat{\sigma}_E^2$ change and what do you expect to happen with the (1-step) prediction intervals?

b) Fit the suggested models and perform a 1-step prediction for each model.Compare the three different prediction intervals. What do you observe?

c) Let us now have a closer look at the effect of overfitting. Is it really better to fit a higher order AR(p)-process to a time series generated by an AR(1)-process, in terms of prediction? To this end, we perform a small simulation study:
Simulate 100 realizations of length 101 of an AR(1)-process. For each realization, fit an AR(1)-, an AR(5)- and an AR(10)-model and perform a 1-step prediction $\hat{X}_{101;\ 1:100}$. Furthermore, for each realization and fitted model, compute the mean squared forecasting error (compare your prediction $\hat{X}_{101;\ 1:100}$ with the $101th$ observation x_{101} of your simulated time series as in Exercise 1, d) and check, if your 1-step prediction interval $\hat{X}_{101;\ 1:100} \pm 1.96 \cdot \hat{\sigma}_E$ contains the true value x_{101}.
R-Hint: You can use the following code skeleton for your simulation:

```
> # 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 = ...)
   fit5 <- arima(window(sim, start = 1, end = 100), order = ...)
   fit10 <- arima(window(sim, start = 1, end = 100), order = ...)

   # 1-step prediction:
   pred1 <- predict(...)
   pred5 <- predict(...)
   pred10 <- predict(...)
```

```
    # 1-step prediction interval:
    PI1 <- ...
    PI5 <- ...
    PI10 <- ...

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

Preliminary discussion: Monday, May 08.