Main topics of the lab:

- Fitting a global trend to a time series
- Simple smoothing methods
- Time series decomposition

Warm-up exercises:

Ex. 1 Load the time series data from the first lab.

Ex. 2 Import the Consumer Price Index time series available in Moodle.

1 Fitting a global trend to a time series

A global trend is a concrete function of time around which the time series values randomly fluctuate. Existence of a global trend is usually not a reasonable assumption but sometimes the steps of fitting such a trend are used as a part of determining a more reasonable model.

Let $z_1, z_2, \ldots, z_n$ be observations of a time series. We'll start by learning a way of fitting a trend function of the form

$$\text{trend}(t) = c_0 + c_1 \cdot f_1(t) + c_2 \cdot f_2(t) + \ldots + c_m \cdot f_m(t),$$

where $f_i,\ i = 1, \ldots, m$ are some known functions or correspond to some other time series.

There are several commands in R which can be used to fit such a trend (for example, the commands **lm**() and **lsfit** ()) but if this fitting is only a part of a more complex time series modeling, using a corresponding command is more useful. We are going to use the command arima() which is going to be our main tool for time series modeling.

The procedure is as follows. First, we'll we form vectors F1,F2 ,..., Fm of length n of the values of the regressor functions $f_1, \ldots, f_m$. If $m > 1$ we combine those vectors into a matrix by the command regressors<−**cbind**(F1,...,Fm) and then use the command

```
trend=arima(z,xreg=regressors)
```

to fit the trend function to the series.

Often we want to remove the trend from a series. This can be achieved by the command `residuals(trend)`, the values of the trend series can be computed by z−**residuals**(trend) and future values of the trend can be computed by

```
predict(trend,n.ahead=how_many,newxreg=cbind(F1_new,...,Fm_new))
```

where `Fi_new` corresponds to the values of $f_i(t)$ at the time moments for which we want to predict the trend.

If we want to fit a trend where the fitting parameters appear nonlinearly, we can use the command nls(). Please read the help for this command (which can be seen by typing ?nls in the command window).

Ex. 3 Fit linear, cubic and fifth order polynomial trends to the dwelling price index. (Hint: find the number of observations n by the command **length**(), define vectors F1 <−1:n, F2 <−F1^2, ...)

Ex. 4 It is possible to put several series on the same picture by using the plot command for drawing one of them and later adding other series data by commands of the form **lines**(series , **col**="name_of_color"). Form a graph which shows the original series together with the trends found in the previous exercise.

Ex. 5 Use the consumer price index series for fitting a global trend of the form $f(t) = c_1 \cdot e^{c_2 t}$. Plot the original series and the trend curve. When using nls() one should specify the formula for the trend curve like z~c1∗**exp**(c2∗**t**) (where **t** is a vector of time moments) and suitable starting values for the parameters (for this problem c_1 should be close to the starting value of the series and c_2 should be quite close to 0) by specifying an additional parameter of the form **start**=**list**(c1=100,c2=0.01).

Ex 6 Predicting future values of a series by using a global trend. Let us predict next five values for the dwelling price index by using the fifth order polynomial trend. For this, we need to define a matrix of five next values fore each regressor function and use it in the **predict** command (option newxreg=...). So, in our case, we should define `F1_new<-(n+1):(n+5)`, `F2_new <-F1^2` and so on.

2 Smoothing data with moving averages

The transform of an input series (z_t) of the form

$$y_t = \sum_{i=-q}^{r} w_i z_{t-i},$$

where $w_i \geq 0$ and $\sum_{i=-q}^{r} w_i = 1$, is called finding **moving averages**. Moving averages are often used for removing noise from an input series and for determining the shape of the trend curve.

In R, moving average transform can be applied by filter () command. The vector of coefficients should be given in backward order of time (the coefficient of the newest observation first) witn option filter =vec_of_**weights**. One way to specify the vector of weights is by **c**() command like filter =**c**(1/3,1/3,1/3). If we want to use only the current and earlier observations in the smoothing, se should use also an option sides=1, otherwise the command assumes that we want to use symmetric smoothing (the weights are use symmetrically around the current observation and in the case of even number of weights more weights are applied to future values). For example, the command y <− filter (z, filter =**c**(1/3,2/3)) applies the transform $y_t = \frac{1}{3}z_{t+1} + \frac{2}{3}z_t$ and the command y <− filter (z, filter =**c**(1/3,2/3),sides=1) corresponds to the transform $y_t = \frac{1}{3}z_t + \frac{2}{3}z_{t-1}$.

Ex. 7 Use the series of accommodated tourists. Find the simple (equal weights) 4-month moving average

$$y_t = \frac{1}{4}\sum_{i=0}^{3} z_{t-i}$$

of the series of accommodated foreign visitors and the symmetric 12-months moving average

$$y_t = \frac{1}{24}z_{t+6} + \frac{1}{12}\sum_{i=-5}^{5} z_{t-i} + \frac{1}{24}z_{t-6}.$$

Show the original series and both smoothed series on the same graph. .

Ex. 8 Exponential smoothing

$$y_i = \alpha z_i + (1-\alpha)y_{i-1},$$

can also be applied with the filter () command. The form of the command is as follows:

```
y <- filter(alpha*z, filter=1-alpha, method="recursive", init =z[1])
```

Compute the exponential moving averages of the Apple Inc stock prices for $\alpha = 0.1, 0.5, 0.9$ and show the results on the same graph with the original series.

3 Decomposition of a time series

Often it is reasonable to assume that the observed series as different component like trend, periodic (seasonal) part and noise (irregular component). If we make assumptions about how the different parts are put together and what are the properties of those parts, we can decompose an original series to such components. There are two ways the components appear: additive decomposition $z_t = T_t + S_t + I_t$ and multiplicative decomposition $z_t = T_t \cdot S_t \cdot I_t$. Classical decomposition methods assume that the seasonal part is completely periodic, some newer methods allow the periodic part also to change in time. Two commands which can be used for decomposing a series are decompose() and stl(). Please read help for the commands.

Ex. 9 Experiment with decomposing the series of accommodated foreign tourists. Which decomposition seems to be the most realistic one?