Main topics of the lab:

- Importing a time series data into R
- Graphing time series data
- Time series decomposition

1 Importing data into R

During this course we consider data that is measured over regular intervals (usually each day, each week, each month, each quarter or each year). Usually this data can be obtained in the form of a simple text file, which may contain one or more time series and additional information. It is very important to import the data correctly since otherwise all future work with the data is just waste of time.

We are going to use R software in the labs. R is a free software environment for statistical computing and graphics and can be downloaded from `http://ftp.eenet.ee/pub/cran/`. The software contains a basic user interface program for working with R (called RGui), so it is possible to work with R without installing any additional programs. Still, the basic user interface included in R lacks many convenient feature (like syntax highlighting and code completion). To make working with R more convenient, it is possible to use additional programs that try to make user's life easier. One such program is RStudio, which is also installed in computer labs (and which you can also install to your own computer from `https://www.rstudio.com/`.

Before importing data from a text file into R, it is a good idea to check how the data appears in the file. For this one should **open the file with a plain text editor** (or programmer's text editor) which shows the content of a file exactly as it is and does not add any formatting information (like fonts, size of characters and so on) when saving a file. Examples of such editors are Notepad++ (available from `https://notepad-plus-plus.org/`) and EditPad Lite (`https://www.editpadlite.com/`).

1.1 Getting time series data, determining the format of the data

There are many sources of time series data in the internet. For example, a lot of useful information can be obtained from web pages of statistical agencies of various countries. Stock price data can be obtained from web pages of stock exchanges. There are many sources for geological data of various countries, astrophysical data and so on. The first task is to locate and download the data that is relevant to finding answers to questions we are interested in.

Ex. 1 Locate and download 3 datasets:

1. The number of accommodated foreign visitors of Estonia each months starting from January 1996 (hint: start from `www.stat.ee`)
2. Historical prices of Appple Inc. for one year (from today) from `www.nasdaq.com`

3. A dataset available at the course web page in Moodle, open it in excel and save it as a tab delimited text file.

Ex. 2 Inspect visually (by Notepad++ or EditPad Lite) each of the text files you obtained. Determine

1. the number of the line at which the time series data begins
2. the symbol that separates the data fields
3. the decimal point separator
4. the number of different time series given in the data
5. the number of lines that contain time series data (needed if there are extra lines at the end of the file)
6. how a data for a series is presented - in a column, in a row, in tabular format, what is the sequence of observations from the oldest to the newest observation

There are many commands that can be used to import data from a text file into R. We'll discuss only one, but quite universal possibility to import tabular data. An alternative for experienced users of R is to use data input commands in the tidyverse collection of libraries or in the **data.table** package. Our procedure for importing the time series data is as follows:

1. Use the command **data**=**read.table**(**file_**name,...) to import the data. Here dots indicate the possibility to specify additional parameters. For us the most important options are:
   `skip=x` - ignore first `x` rows. Default value is 0
   `sep="c"` - data fields are separated by the symbol `c`. Default separators are space and tabulation symbol
   `header=TRUE` - first row (after the skipped rows) contains names of variables. Default value is `FALSE`
   `dec=","` - decimal separator. Default value is `"."`
   `na.strings="s"` - what is used for denoting missing values. Default is `NA`.
   `as.is=TRUE` - read the textual values as they appear in the file. Default is to convert them to factors by encoding different values with integers.

2. Forming a vector of values for the time series we are interested in. If the series is in a single column, we can use the commands series=**data$**name (where `name` is the name of the corresponding column) or series=**data**[,j], where `j` is the number of the column. If data is organised in a tabular form so that consecutive observation are in rows (like a row corresponds to a given year and columns correspond to different months), then we can we can form a vector of observations by the command series=**c**(**t**(**data**[,a:b])). Here `a` is the number of the column containing the first observation for each row and `b` is the number of column containing the last observation for a given row. Explanation of the command: **data**[,a:b] separates from the original table the matrix containing observed values of the time series, **t**() transposes the matrix and **c**() transforms the matrix to a vector by ordering the data by columns. If the data is in a tabular form, but consecutive observations are in columns, we can use the command series=**c**(**as.matrix**(**data**[,a:b])).

3. Ordering the data so that the newest observations are at the end. If the observations are ordered in the reverse order with respect to time (newest first), we can reorder the series by the command series =series[**length**(series):1]

4. Getting rid of missing data from the end of the time series. Often the data is exported by full periods (usually years) and the last period contains some missing data since corresponding to time moments that are in the future. It is easier to work with the data if such missing observations are removed. This can be done by the command series =series[1:(**length**(series)−x)], where `x` is the number of missing observations at the end of the series.

5. Forming a time series object from the data. Some commands work better if we have let R know that a vector corresponds to time series. For this, we should specify the number of observations in a period (frequency) and the period number and observation number for the first period. We can form a time series by the command **time_**series=**ts**(series,...), where dots indicate the possibility to specify additional information. The most important options are
frequency=n - the number of observations in a period. The default value is 1, for quarterly data the frequency is 4, for monthly data the frequency is 12.
start=x - Time of the first observation. If frequency is 1, then `x` should be the number of the first observation; if there are many observations in a period, then `x` should be a pear of the period number and observation number in the form **c**(period_number,observation_number).

6. Visual inspection of the imported data. We can see the graph of the series by the command **plot**(**time_**series). If we want to see only a part of the series, we can use the command **window**() in the form

$$\textbf{plot}(\textbf{window}(\textbf{time\_}series, \textbf{start}=first\_obs, \textbf{end}=last\_obs),$$

where first_obs and last_obs specify the first and last observations we want to see on the graph. Additionally, it is a very good idea to look at the beginning and the end of the series by commands head(**time_**series) and tail(**time_**series) and compare this to the data in the original file.

Ex. 3 Import the data obtained in Exercise 1 into R and inspect the results graphically. Save the data for future use by the command

$$\textbf{save}(\textbf{time\_}series1, \textbf{time\_}series2, \textbf{time\_}series3, \textbf{file}="prax1data.RData"))$$

In the future we can load the data by the command `load("prax1data.RData")`.

2 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 pages for the commands.

Ex. 4 Experiment with decomposing the series of accommodated foreign tourists. Which decomposition seems to be the most realistic one?