## STAT 443: Time Series and Forecasting

# Lab 2: Trend, Seasonality and Sample Autocorrelation

## Objectives of the lab

1. Examine how smoothing can be used to estimate trend and seasonal effects in time series.
2. Investigate sample autocorrelation plots for several time series with the aim to appreciate how features of an observed time series translate into the properties of its sample acf.

## Instructions

This lab must be completed in `R Markdown`. You can use the template provided in lab 1. Please create a **pdf file** (via 'knit to pdf') and use it as your lab submission.

## Useful R commands for time series analysis

- `acf()`
- `rollmean()`
- `decompose()`
- `stl()`

1. The dataset "dataTempPG.csv" contains monthly and annual means of homogenized daily minimum temperatures at Prince George, BC, over the period from 1919 to 2008.

   (a) (**Features of the data**)

   - Read the data into R using `read.csv()`.
   - The column labelled "Summer" provides the mean summer temperatures. Extract those data, and coerce them into a time series object in R, suitably specifying arguments to function `ts()`.
   - Plot the time series, remembering to label the plot and axes using arguments `main`, `xlab`, `ylab`).
   - Comment on main features of this series (e.g., seasonality, trend).

   (b) (**Sample Autocorrelation**) Use command `acf()` to create the autocorrelation function for the mean summer temperature data. Comment on the behaviour of the sample autocorrelation function.

   (c) (**Smoothing**)

   - Using command `window()`, extract the portion of the time series between years 1968 and 2008. Plot this recent record of the data.
   - The function `rollmean()` allows you to compute rolling mean or moving average of a given order $k$ (just as was done in the in-class activity #1). Setting the moving average parameter to $k = 5$, add the smoothed series to the plot of the temperature data (use function `lines()`). Add the legend to your plot using function `legend()`.

     **Suggestions:** You can define a new variable, say `ma5 <- rollmean(dat,k=5)` and then use `lines(ma5)`. Use the argument `col="red"` to distinguish the smoothed series from the original one. Alternatively, you can combine the two steps into one command `lines(rollmean(dat, k=5),col=2)`.

2. The dataset "LakeLevels.csv" contains the daily depths (in meters) of a lake from 2007 to 2011, inclusive. Read the data into R and coerce it into a time series object using command `ts()`.

   (a) Use command `acf()` to create the sample autocorrelation function for this time series and comment on its behaviour.

(b) The function `decompose()` can be used to decompose a time series into trend, seasonal component and an error term using moving average processes. It can handle both an additive and multiplicative models. Plot the trend, seasonal component and error for the lake level data assuming the additive seasonal decomposition model.

**Suggestions:** You can achieve this in one step combining the two functions: `plot(decompose(dat, type="additive"))`. Alternatively, you can first save the decomposition component into one variable:
`dat.decom <- decompose(dat, type="additive")` and then create a plot: `plot(dat.decom)`.

The individual decomposition components can be extracted as follows:

```
trend <- dat.decom$trend # this is the same as rollmean(level, k=365)
seas <- dat.decom$seasonal # simple seasonal effect estimation using averaging
error <- dat.decom$random # the remaining term
```

(c) An alternative method to perform the trend and seasonal effect decomposition is via a method called *loess* smoothing. The method is implemented in function `stl()`. Set the argument `s.window` to "periodic". Plot the trend, seasonal component and error using the loess method.

**Further notes:** To extract the individual decomposition components, a slightly different approach has to be used. Below is a sample code for your reference only.

```
lev.stl <- stl(dat, s.window="periodic")
trend <- dat.stl$time.series[,"trend"]
seas <- dat.stl$time.series[,"seasonal"]
error <- dat.stl$time.series[,"remainder"]
```