Spring Semester 2023–2024

MATH3026/4022: Time Series Analysis/Time Series and Forecasting
Computer Practical 1

During this practical, we will use R by loading RStudio and running commands. When writing R commands, it is good practice to write comments so that you understand your code.

Before beginning the practical, it is recommended that you create a folder named `MATH3026` or `MATH4022` within your `Documents` folder on your University workspace. Then, within your `MATH3026` or `MATH4022` folder, create a new folder called `Practical1` in which you can save all of the material covered during this practical session.

You should set your working directory to be the `Practical1` folder that you've created. In RStudio, this can be done by selecting the `Session` menu and then `Set Working Directory` and `Choose Directory`, navigating to the correct folder. You can then easily save your R code and plots in this directory.

1. We'll begin by learning how to simulate some time series data. We can do this using the R command `arima.sim`. For help on using this command, you can type `?arima.sim` into the command prompt (and then press enter). Note that the command `arima.sim` assumes that the white noise terms are normally distributed.

   Suppose we wish to simulate 1000 observations from the AR(1) model:

   $$X_t = 0.9X_{t-1} + Z_t$$

   where the variance of the white noise process, $\{Z_t : t \in \mathbb{Z}\}$, is $\sigma_z^2 = 0.25$ (so $\sigma_z = 0.5$).

   (a) Type in and run the following R commands to simulate 1000 observations from this process (please note the comments that I have provided to give you an idea of how you might comment your own R code). To open a new script from which to run code, select `File > New File > R Script` from the menu. The command `set.seed(n)` sets an initial 'seed value' because simulating data involves using a random number generator. Using the same seed ensures that you would obtain the same results when re-running your code.

```
#Set the seed (7436 can be replaced with a number of your choice)
set.seed(7436)

#Assign a value for the parameter phi1
phi1<-0.9

#Assign a value for the parameter sigma_z
```

```
sigma_z<-0.5

#Simulate 1000 observations from the AR(1) model with
#phi1=0.9 and sigma_z=0.5
y<-arima.sim(n=1000,model=list(ar=c(phi1)),sd=sigma_z)
```

Here

- `n` denotes the number of observations we'd like to simulate.
- `model` lists the model parameters (here for the `ar` part of the model as we're simulating from an AR(1) process). We can use `ma` to specify MA parameters if desired.
- `sd` denotes the desired value of σ_z.

(b) Use the following command to produce a time plot of the simulated data

```
#Produce a time plot of the simulated AR(1) data
ts.plot(y)
```

(c) Use the following command to produce a plot of the sample autocorrelation function against the lag.

```
#Produce a plot of the sample ACF for the simulated AR(1) data
acf(y)
```

(d) The function `ARMAacf` is used to calculate the theoretical ACF for an ARMA process. Run the following command to calculate the theoretical ACF for the AR(1) process covered in this question (here, we do this for lags up to 100).

```
#Function to calculate the theoretical ACF for the AR(1) process
Q1.AR1.acf<-ARMAacf(ar=c(phi1),ma=c(0),lag.max=100)
```

Here

- `ar` denotes the parameters for the AR part of the model. We have an AR(1) model so we have one AR(1) parameter equal to `phi1` (with `phi1`=0.9).
- `ma` lists the model parameters for the MA part of the model. Since we have an AR model, we set the MA parameter(s) to be zero.
- `lag.max` denotes the maximum number of lags at which we'd like to calculate the ACF. Here we set the max. lag to be 100.

(e) Using the command below, produce another plot of the sample ACF, this time with the theoretical ACF shown on the plot as a red line. Is the sample ACF close to the theoretical ACF? Repeat this step for simulated AR(1) processes with $n = 10000$ and $n = 100000$ observations. What do you notice?

```
#Produce a plot of the sample ACF for the simulated AR(1) data
acf(y)
```

```
#Add a line (in red) that shows the theoretical ACF against lags 0 to 100
y_lag<-c(0:100)
lines(y_lag,Q1.AR1.acf,col="red")
```

2. Now that you have explored and run the R code in Question 1, carry out the following tasks, using the commands presented in Question 1. as a guide where necessary. Please remember to save and comment your R code.

   (a) Simulate $n = 1500$ observations from an $MA(1)$ process with parameters $\theta_1 = 0.6$ and $\sigma_z^2 = 0.49$.

   (b) For the data simulated in (a) produce:

      (i) A time plot.

      (ii) A plot of the sample ACF against the lag, with a line showing the theoretical ACF included on the plot for comparative purposes.

3. Instead of using `arima.sim` to simulate values from a time series, we can simulate values using code that we have written to do this directly.

   (a) Read the code below to ensure that you understand how it works. Then, run the code to simulate 1000 values from an $AR(1)$ process with $\phi_1 = -0.6$ and $\sigma_z = 0.5$.

```
#Set the seed (82301 may be replaced by a number of your choice).
set.seed(82301)

#State the required number of of simulated values (n=1000).
n<-1000

#Define an empty vector of length n which will eventually contain
#the simulated time series values.
x<-numeric(n)

#Define the values of phi1 and sigma_z
phi1<--0.6
sigma_z<-0.5


#Define an nx1 vector of independent and identically distributed white
#noise terms: z_1,...,z_n. Here, we assume that the white noise terms
#are normally distributed.
z<-rnorm(n,mean=0,sd=sigma_z)
```

```
#Set the first value of the series (first element of x) to be random
#white noise (z_1).
x[1]<-z[1]

#Values 2:n of the vector x will depend on previous values of the
#series, according to the AR(1) model. We use the 'for loop' below to
#populate values of x.
for(t in 2:n){
  x[t]<-phi1*x[t-1]+z[t]
}

#At this point, we should tell R that the data we've simulated are time
#series data. We do this by running the command 'ts' below. This is known
#as creating a 'time series object'. 'start' defines the time point at
#which the series starts and 'end' that where the series ends.
x<-ts(x,start=1,end=n)
```

(b) Produce a time plot for the data simulated in (a).

(c) Using the sample and theoretical ACFs, verify that the data you simulated in (a) are from an $AR(1)$ process with $\phi_1 = -0.6$.

4. (a) Writing you own code (and without using `arima.sim`), simulate 2000 values from an $MA(1)$ process with $\theta_1 = 0.8$ and $\sigma_z^2 = 0.64$. You should assume that the white noise terms are normally distributed. Please remember to comment your R code and to save your work.

   (b) For the data simulated in (a) produce:

      (i) A time plot.

      (ii) A plot of the sample ACF against the lag, with a line showing the theoretical ACF included on the plot for comparative purposes. Do the data appear to be from an $MA(1)$ process?

5. The dataset `eng_rain` can be found on the course moodle page in CSV format and contains annual rainfall measurements (in millimetres) for England for the years 1862–2021, sourced from the UK Met Office databank:

   https://www.metoffice.gov.uk/research/climate/maps-and-data/uk-and-regional-series.

   We will work with these data in this question.

   (a) Download the data from moodle and save this in your `Practical1` folder. Ensuring that the working directory in RStudio is set to your `Practical1` folder (see instructions at the beginning of the practical sheet for guidance on how to do this), run the following command to read the dataset into RStudio:

```
eng_rain<-read.csv("eng_rain.csv",header=TRUE)
```

(b) We need to declare the annual rainfall data as time series data, starting at 1862 and ending at 2021. Run the following command to do this, which creates the time series object `annual_rain`:

```
annual_rain<-ts(eng_rain$annual_rainfall_mm,start=1862,end=2021)
```

(c) Produce a time plot of the annual rainfall data. Does the time series appear to be stationary? Justify your answer.

(d) What type of model would you suggest for the annual rainfall data? Explain your reasoning.