Basic Time Series Analysis using R*

Jeric C. Briones

Department of Mathematics

Ateneo de Manila University

1 Preliminaries

Before working with real dataset, we begin by doing preliminary analysis on simulated time series data.

1.1 Loading Simulated Data

We begin by loading the simulated dataset `ar2.s` and `ma1.1.s` from the library `TSA`. This simulated time series data will be used for the succeeding preliminary analyses. Codes used are in `3TSA.R`.

Aside from the loaded simulated data, we also generate another time series data using random standard normal deviates. Standard normal deviates are generated using `rnorm()`, where the required input is the number of deviates to generate. When unspecified, the default mean and standard deviation are `mean=0` and `sd=1`, respectively. The deviates are converted to time series data using the function `ts()`.

```
# Time-series library
library("TSA")
library("tseries")

# simulated data
data("ar2.s")
data("ma1.1.s")
z = ts(rnorm(120))
```

By default, the parameters of `ts()` are `start=1`, `end=T`, and `frequency=1`. Here, `T` refers to the length of the time series data, while `frequency` refers to the number of observations in a given time step. Commonly used frequencies include `frequency=12` to represent monthly data, and `frequency=4` to represent quarterly data. These frequencies can easily by changed for a given time series object, as shown below.

```
# change frequency; even if frequency was changed, index is unchanged
## set frequency to 10 per unit time
z1 = ts(z, frequency=10)
## set frequency to monthly, from Dec 2012 to Mar 2023
z2 = ts(z, start=c(2012,12), end=c(2023,3),frequency=12)
plot(z)
plot(z1)
plot(z2)
```

*Supplementary notes for MATH 62.2 Time Series and Forecasting, Second Semester, SY 2022-2023

Here, `z1` has the same values as `z`. However, instead of having 120 unit time steps like the latter, the former has $\frac{120}{10} = 12$ unit time steps, where each unit time steps has 10 sub-steps. Similarly, `z2` also has the same values as `z` but with monthly frequency (`frequency=12`). To indicate that the time series starts December 2012 and ends March 2023, the parameters `start=c(2012,12)` and `end=c(2023,3)` are used. It can be noticed that when `z`, `z1`, and `z2` are plotted, the figures are similar except for the x-axis scale.

1.2 Exploratory Analysis

After loading the dataset, we begin performing exploratory tests to the data: test for autocorrelation and unit-root test. Test for autocorrelation is performed using `Box.test()`. To specify the portmanteau test used, we use the parameter `type`. Here, we consider the Ljung-Box Test, specified using `type="Ljung"`. Moreover, by default, this function tests for lag-1 autocorrelation. To test for lag-l autocorrelation, the parameter `lag=l` is used. Recall that should there be no autocorrelation, modeling using linear time series model would not be meaningful[1].

```
# check for autocorrelation
# default is lag = 1
Box.test(ar2.s, type="Ljung")
Box.test(ma1.1.s, type="Ljung")
Box.test(z, type="Ljung", lag=3) # test for lag-3 correlation
```

To test for stationarity, we perform unit-root test using augmented Dickey-Fuller Test[2] (`adf.test()`). Here, the null hypothesis is the presence of unit root, which means the time series data is not stationary. Thus, if the null hypothesis is rejected, the time series data is considered as stationary. Recall that should the data be non-stationary, preprocessing steps (such as differencing) needs to be performed.

```
# test for stationarity (AR)
adf.test(ar2.s)
adf.test(ma1.1.s)
adf.test(z)
```

1.3 Model Building

After checking for autocorrelation and stationarity, we then proceed with time series modelling. We start with lag order selection. This is done by checking the ACF (using `acf()`) and PACF (using `acf()` and adding the parameter `type="partial"`).

```
# compute for ACF
acf(ar2.s)
acf(ma1.1.s)
acf(z)

# compute for PACF
acf(ar2.s, type="partial")
acf(ma1.1.s, type="partial")
acf(z, type="partial")
```

[1] For illustrative purposes, we continue modelling `z` despite having no autocorrelation.

[2] Recall that MA processes are weakly stationary. So, we basically test for stationarity for AR processes. On the other hand, recall that the ADF test assumes an AR model.

Here, notice that based on their respective plots, there is no autocorrelation between the data and its lag-1 value, consistent with earlier results. This is because none of the ACF and PACF values of z are statistically significant. That is, since none of these values exceed the dotted lines, then none of them are statistically different from zero. On the other hand, for ar2.s, ACF values decay while PACF values cut off at lag $h = 2$ (with another significant ACF value at lag $h = 9$). For ma1.1.s, ACF values seem to cut off at lag $h = 1$ (but with other significant ACF value at lags $h = 5, 14$).

Based on these lags, we then estimate the model coefficients using arima(). The lag orders are specified using the parameter order=c(p,0,q), where p refers to the AR lag order, while q the MA lag order[3]. Here, we test AR(2) and AR(9) for ar2.s, and MA(1), MA(5), and MA(14) for ma1.1.s. We also test AR(1) for ar2.s for illustrative purposes.

```
# fit ARMA models
# order = AR lag, integration, MA lag
arima(ar2.s, order=c(1,0,0)) # test AR(1)
arima(ar2.s, order=c(2,0,0)) # test AR(2)
arima(ar2.s, order=c(9,0,0)) # test AR(9)

arima(ma1.1.s, order=c(0,0,1)) # test MA(1)
arima(ma1.1.s, order=c(0,0,5)) # test MA(5)
arima(ma1.1.s, order=c(0,0,14)) # test MA(14)
```

Recall that we can use AIC to determine the best model, with the *best* model having the lowest AIC. Based on the results, the time series data ar2.s is an AR(2) process (AIC of 331.95 vs 451.16 for AR(1) and 339.58 for AR(9)), while ma1.1.s is an MA(1) process (AIC of 363.66 vs 372.58 for MA(5) and 339.58 MA(14)).

2 Fitting the Linear Time Series Model

We continue by loading the datasets SP and usd.hkd from the library tseries. The former dataset is the quarterly S&P composite index, from Quarter 1 of 1936 to Quarter 4 of 1977, while the latter is the daily USD/HKD exchange rate from January 1, 2005 to March 7, 2006. The FOREX rates can be seen in usd.hkd$hkrate.

2.1 Data: S&P Composite Index

As earlier discussed, after loading the time series data, we begin with the preliminary tests, specifically the test for stationarity using adf.test().

```
# Quarterly S&P Composite Index, 1936Q1 - 1977Q4.
data("SP")

# test for stationarity (AR)
adf.test(SP)
```

Here, we note that the null hypothesis of having a unit root (that is, the data is non-stationary) was not rejected. Hence, preprocessing steps are needed to transform the data into a stationary process. Possible preprocessing steps include log transformation, differencing, and log differencing. In this example, we focus on log differencing, since the data values are quite large. Moreover, we note that log transformation is generally used to handle large-valued data, while differencing is mostly used to remove non-stationarity.

[3] The middle parameter, 0, refers to the order of integration d, where $d = 0$ for ARMA processes.

```
# perform pre-processing
SP.ln = log(SP) # take log transformation
SP.ln.d = diff(SP.ln) # take first difference of log
plot(SP.ln.d)

# perform stationarity test for differenced data
adf.test(SP.ln.d)

# check autocorrelation
Box.test(SP.ln.d, type="Ljung")
```

Performing the unit-root test on the transformed data, we note that the resulting time series data is now stationary. Given it is stationary, we then proceed with the test for autocorrelation. Using the Ljung-Box Test, we note that the data has no autocorrelation since the test failed to reject the null hypothesis. As such, further modelling for this time series data will not be pursued.

2.2 Data: USD/HKD FOREX Rate

We begin by converting the current data into a time series object using the function `ts()`. Then, like earlier, we start with the preliminary tests `adf.test()` and `Box.test()`.

```
# Daily USD/HKD exchange rate from January 1, 2005 to March 7, 2006
data("usd.hkd")
forex = ts(usd.hkd$hkrate)

# perform initial tests
adf.test(forex)
Box.test(forex, type="Ljung")
```

Here, both tests rejected their respective null hypotheses. Since the data is stationary and serially correlated, we then proceed with (lag) order determination.

```
# check ACF, PACF
acf(forex)
acf(forex, type="partial")
```

Based on the ACF and PACF plots, notice that the ACF plot seems to cut off at lag $h = 1$, while the PACF plot seems to cut off at lag $h = 2$. Thus, we test for AR(2), MA(1), and ARMA$(2, 1)$.

```
# check models, lower AIC is better
arima(forex, order=c(0,0,1)) # test MA(1)
arima(forex, order=c(2,0,0)) # test AR(2)
arima(forex, order=c(2,0,1)) # test ARMA(2,1)
```

Checking the AIC, the *best* model is ARMA$(2, 1)$ with an AIC of -1890.64 (vs -1888.41 for MA(1) and -1889.49 for AR(2)). Thus, `usd.hkd$hkrate` follows an ARMA$(2, 1)$ process.