LAB MANUAL:07

MAT 3208: TIME SERIES

DATE:

AR model estimation and forecasting

For a given time series `x` we can fit the autoregressive (AR) model using the `arima()` command and setting `order` equal to `c(1, 0, 0)`.

Note for reference that an AR model is an **ARIMA(1, 0, 0)** model.

Problem 01

Instructions

- Use `arima()` to fit the AR model to the series `x`. Closely examine the output from this command.
- What are the slope (`ar1`), mean (`intercept`), and innovation variance (`sigma^2`) estimates from your previous command? Type them into your R workspace.
- Now, fit the AR model to `AirPassengers`, saving the results as `AR`. Use `print()` to display the fitted model `AR`.
- Finally, use the commands provided to plot the `AirPassengers`, calculate the fitted values, and add them to the figure.

```
# Fit the AR model to x
arima(___, order = ___)
# Copy and paste the slope (ar1) estimate

# Copy and paste the slope mean (intercept) estimate

# Copy and paste the innovation variance (sigma^2) estima
te

# Fit the AR model to AirPassengers
AR <-
print(AR)

# Run the following commands to plot the series and fitte
d values
ts.plot(AirPassengers)
AR_fitted <- AirPassengers - residuals(AR)
points(AR_fitted, type = "l", col = 2, lty = 2)
```

Simple forecasts from an estimated AR model

The **`predict()`** function can be used to make forecasts from an estimated AR model. In the object generated by your `predict()` command, the `$pred` value is the forecast, and the `$se` value is the standard error for the forecast.

To make predictions for several periods beyond the last observations, you can use the `n.ahead` argument in your `predict()` command. This argument establishes the forecast horizon (`h`), or the number of periods being forecast. The forecasts are made recursively from 1 to h-steps ahead from the end of the observed time series.

Problem 02

- Use `arima()` to fit an AR model to the `Nile` time series. Save this as `AR_fit`.
- Use `predict()` to make a forecast for flow of the Nile in 1971.
- Use `predict_AR` along with `$pred[1]` to obtain the 1-step forecast.
- Use another call to `predict()` to make forecasts from 1 step ahead to 10 steps ahead (1971 to 1980). To do so, set the `n.ahead` command equal to `10`.
- Run the pre-written code to plot your `Nile` data plus the forecasts and a 95% prediction interval.

```
# Fit an AR model to Nile
AR_fit <- arima(___, order  = ___)
print(AR_fit)

# Use predict() to make a 1-step forecast
predict_AR <- predict(___)

# Obtain the 1-step forecast using $pred[1]


# Use predict to make 1-step through 10-step forecasts
predict(___, n.ahead = ___)

# Run to plot the Nile series plus the forecast and 95% p
rediction intervals
ts.plot(Nile, xlim = c(1871, 1980))
AR_forecast <- predict(AR_fit, n.ahead = 10)$pred
AR_forecast_se <- predict(AR_fit, n.ahead = 10)$se
points(AR_forecast, type = "l", col = 2)
points(AR_forecast - 2*AR_forecast_se, type = "l", col =
2, lty = 2)
points(AR_forecast + 2*AR_forecast_se, type = "l", col =
2, lty = 2)
```

MA model estimation and forecasting

For a given time series `x` we can fit the simple moving average (MA) model using `arima(..., order = c(0, 0, 1))`. Note for reference that an MA model is an **ARIMA(0, 0, 1)** model.

Problem 03

- Use `arima()` to fit the MA model to the series `x`.
- What are the slope (`ma1`), mean (`intercept`), and innovation variance (`sigma^2`) estimates produced by your `arima()` output? Paste these into your workspace.
- Use a similar call to `arima()` to fit the MA model to the `Nile` data. Save the results as `MA` and use `print()` to display the output.
- Finally, use the pre-written commands to plot the `Nile` data and your fitted MA values.

```
# Fit the MA model to x
arima(___, order = ___)

# Paste the slope (ma1) estimate below


# Paste the slope mean (intercept) estimate below


# Paste the innovation variance (sigma^2) estimate below


# Fit the MA model to Nile
MA <- arima(___, order = ___)
print(MA)

# Plot Nile and MA_fit
ts.plot(Nile)
MA_fit <- Nile - resid(MA)
points(MA_fit, type = "l", col = 2, lty = 2)
```

Simple forecasts from an estimated AR model

you can use the `predict()` function to make simple forecasts from your estimated MA model. Recall that the `$pred` value is the forecast, while the `$se` value is a standard error for that forecast, each of which is based on the fitted MA model.

Once again, to make predictions for several periods beyond the last observation you can use the `n.ahead = h` argument in your call to `predict()`. The forecasts are made recursively from 1 to h-steps ahead from the end of the observed time series. However, note that except for the 1-step forecast, all forecasts from the MA model are equal to the estimated mean (`intercept`).

Problem 04

- Use `predict()` to make a forecast for River Nile flow level in 1971. Store the forecast in `predict_MA`.
- Use `predict_MA` along with `$pred[1]` to obtain the 1-step forecast.
- Use another call to `predict()` to make a forecast from 1971 through 1980. To do so, set the `n.ahead` argument equal to `10`.
- Run the pre-written code to plot the `Nile` time series plus the forecast and 95% prediction intervals.

```
# Make a 1-step forecast based on MA
predict_MA <-

# Obtain the 1-step forecast using $pred[1]


# Make a 1-step through 10-step forecast based on MA


# Plot the Nile series plus the forecast and 95% predicti
on intervals
ts.plot(Nile, xlim = c(1871, 1980))
MA_forecasts <- predict(MA, n.ahead = 10)$pred
MA_forecast_se <- predict(MA, n.ahead = 10)$se
points(MA_forecasts, type = "l", col = 2)
points(MA_forecasts - 2*MA_forecast_se, type = "l", col =
 2, lty = 2)
points(MA_forecasts + 2*MA_forecast_se, type = "l", col =
 2, lty = 2)
```

Problem 05

Exploration data

To elaborate the ARMA models we will use an inbuilt data set of R called AirPassengers. The dataset consists of monthly totals of international airline passengers, 1949 to 1960.

Step 01

Load the data. Then the first pre-requisite is, no prizes for guessing - the data should be a time series data and to check that you can use function ‘`is.ts()`’.

```
#Loading the Data Set
data("AirPassengers")


#This tells you that the data series is in a time series format
is.ts(AirPassengers)
```

Step 02

Now that we know that the data is time series we should do some data exploration. Functions `print()` and `summary()` are used to get the overview of the data. The `start()` and `end()` functions return the time index of the first and last observations, respectively. The `time()` function calculates a vector of time indices, with one element for each time index on which the series was observed. Finally, the `frequency()` function returns the number of observations per unit time.

```
#This will give us the structure of our data
print(AirPassengers)
```

```
#This will give us summary of our data
summary(AirPassengers)
```

```
#Starting index, end index
start(AirPassengers)
```

```
end(AirPassengers)
```

```
time(AirPassengers)
```

```
#This will print the cycle across years.
frequency(AirPassengers)
```

Step 03

It is essential to analyze the trends prior to building any kind of time series model. The details we are interested in pertains to any kind of trend, seasonality or random behaviour in the series. what better way to do so than visualize the Time Series.

```
#This will plot the time series

ts.plot(AirPassengers, xlab="Year", ylab="Number of Passengers", main="Mont
hly totals of international airline passengers, 1949-1960")


# This will fit in a line

abline(reg=lm(AirPassengers~time(AirPassengers)))
```

Autocorrelation

It is a very powerful tool for Time series analysis. Process with auto correlation are more predictable as compared to none.

Total Correlation Chart (also known as Auto – correlation Function / ACF) - auto correlation defined as a function of the time lag. Autocorrelations or lagged correlations are used to assess whether a time series is dependent on its past. For a time series x of length n we consider the n-1 pairs of observations one time unit apart. The first such pair is (x[2],x[1]), and the next is (x[3],x[2]). Each such pair is of the form (x[t],x[t-1]) where t is the observation index, which we vary from 2 to n in this case. The lag-1 autocorrelation of x can be estimated as the sample correlation of these (x[t], x[t-1]) pairs.

If the above doesnt make sense, luckily, the `acf()` command provides a shortcut. Applying `acf(…, lag.max = 1, plot = FALSE)` to a series x automatically calculates the lag-1 autocorrelation.

ACF help us determine what type of series we have, whether it is a White noise, Random walk, Auto regressive or Moving average.

```
acf(AirPassengers)
```

Fit the AR model to AirPassengers

For a given time series x we can fit the autoregressive (AR) model using the `arima()` command and setting order equal to `c(1, 0, 0)`. Note for reference that an AR model is an `ARIMA(1, 0, 0)` model.

```
#Fitting the AR Model to the time series
AR <- arima(AirPassengers, order = c(1,0,0))
print(AR)
#plotting the series along with the fitted values
ts.plot(AirPassengers)
AR_fit <- AirPassengers - residuals(AR)
points(AR_fit, type = "l", col = 2, lty = 2)
```

Forcasting using AR model

The `predict()` function can be used to make forecasts from an estimated AR model. In the object generated by your `predict()` command, the `$pred` value is the forceast, and the $se value is the standard error for the forceast. To make predictions for several periods beyond the last observations, you can use the `n.ahead` argument in your `predict()` command. This argument establishes the forecast horizon (h), or the number of periods being forecast. The forecasts are made recursively from 1 to h-steps ahead from the end of the observed time series.

```
#Using predict() to make a 1-step forecast
predict_AR <- predict(AR)


#Obtaining the 1-step forecast using $pred[1]
predict_AR$pred[1]
```

```
#ALternatively Using predict to make 1-step through 10-step forecasts
predict(AR, n.ahead = 10)
```

```
#plotting the AirPassenger series plus the forecast and 95% prediction inte
rvals
ts.plot(AirPassengers, xlim = c(1949, 1961))
AR_forecast <- predict(AR, n.ahead = 10)$pred
AR_forecast_se <- predict(AR, n.ahead = 10)$se
points(AR_forecast, type = "l", col = 2)
points(AR_forecast - 2*AR_forecast_se, type = "l", col = 2, lty = 2)
points(AR_forecast + 2*AR_forecast_se, type = "l", col = 2, lty = 2)
```

Fit the MA model to AirPassengers

We can fit the simple moving average (MA) model using `arima(…, order = c(0, 0, 1))`. Note for reference that an MA model is an `ARIMA(0, 0, 1)` model.

```
#Fitting the MA model to AirPassengers
MA <- arima(AirPassengers, order = c(0,0,1))
print(MA)
```

```
#plotting the series along with the MA fitted values
ts.plot(AirPassengers)
MA_fit <- AirPassengers - resid(MA)
points(MA_fit, type = "l", col = 2, lty = 2)
```

Forcasting using MA model

```
#Making a 1-step forecast based on MA
predict_MA <- predict(MA)

#Obtaining the 1-step forecast using $pred[1]
predict_MA$pred[1]
```

```
#Alternately Making a 1-step through 10-step forecast based on MA
predict(MA,n.ahead=10)
```

```
#Plotting the AIrPAssenger series plus the forecast and 95% prediction inte
rvals
ts.plot(AirPassengers, xlim = c(1949, 1961))
MA_forecasts <- predict(MA, n.ahead = 10)$pred
MA_forecast_se <- predict(MA, n.ahead = 10)$se
points(MA_forecasts, type = "l", col = 2)
points(MA_forecasts - 2*MA_forecast_se, type = "l", col = 2, lty = 2)
points(MA_forecasts + 2*MA_forecast_se, type = "l", col = 2, lty = 2)
```

Choosing AR or MA: Exploiting ACF plots

Once we have got the models ready we must answer the important question: Should we choose AR or MA process? Goodness of fit such as an Information criterion is a method to help us make the decision. Specifically, **Akaike information criterion (AIC)** and **Bayesian information criterion (BIC)** are used for Time series Models. Information Criteria is a more advanced concept but for either measure a lower value indicates a relatively better fitting model.

While the math underlying the AIC and BIC is beyond the scope of this vignettw, for your purposes the main idea is these indicators penalize models with more estimated parameters, to avoid overfitting, and smaller values are preferred. All factors being equal, a model that produces a lower AIC or BIC than another model is considered a better fit.

```
# Find correlation between AR_fit and MA_fit

cor(AR_fit, MA_fit)
```

```
# Find AIC of AR

AIC(AR)
```

```
# Find AIC of MA

AIC(MA)
```

```
# Find BIC of AR

BIC(AR)
```

```
# Find BIC of MA

BIC(MA)
```

Given the lower value of AIC and BIC in AR model, we should go with that for the time series analysis of AirPassenger data.