MTH 325, Lab #10, Spring 2023 Name ____

Instructions: Follow along with the tutorial portion of the lab. Replicate the code examples in R on your own, along with the demonstration. Then use those examples as a model to answer the questions/perform the tasks that follow. Copy and paste the results of your code to answer questions where directed. Submit your response file and the code used (both for the tutorial and part two). Your code file and your lab response file should each include your name inside. Be sure to follow the write-up directions in the Lab Directions file.

Time series models come in a number of flavors. Some are based on moving averages, some on seasonal patterns, some on autoregressive models, linear and non-linear trends, and can even incorporate other non-time variables into the model, and combinations of these components. We aren't going to be able to look at a lot of variations—this could take a whole semester—but we'll look at some common basic types.

For the first example, we're going to look at a built-in dataset called AirPassengers. First load the data and look at it to confirm it is a time series.

```
data("AirPassengers")
AirPassengers
plot(AirPassengers)
```

Then decompose it to look at the various components.

```
decomp1<-decompose(AirPassengers)
plot(decomp1)
```

Examine the data using such tools as histograms, qqplots, and an acf graph.

```
hist(AirPassengers)
qqnorm(AirPassengers, pch = 1, frame = FALSE)
qqline(AirPassengers, col = "steelblue", lwd = 2)

acf(AirPassengers)
```

Since we are not only looking at a stationary series, the histogram and qqplot are not normal. We can make a histogram and qqplot of the noise part of the decomposed time series, where we would expect normality.

```
hist(decomp1$random)
qqnorm(decomp1$random, pch = 1, frame = FALSE)
qqline(decomp1$random, col = "steelblue", lwd = 2)
```

We're going to build an AR1 model for the AirPassengers data. This is an autoregressive model which models future values based on past values.

```
AR <- arima(AirPassengers, order = c(1, 0, 0))
print(AR)
```

The print() function prints out the model coefficients. It also prints out some diagnostic information such as standard errors for each coefficient, as well as things like the AIC for the model as a whole. This is a very simple model, but we can compare it against the original data, by plotting the resulting model on the same graph as the original time series.

```
ts.plot(AirPassengers)
AR_fitted <- AirPassengers - residuals(AR)
points(AR_fitted, type = "l", col = 2, lty = 2)
```

The model does at an okay job, but does seem to lag the original data a bit.

Furthermore, we can use the model to predict out beyond the original data and forecast future values, along with the margin of error for our predictions. To make the predictions a bit easier to see, you can redo the initial plot, otherwise these will plot on the same graph on the far right end.

```
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)
```

We could also look at a moving average model to see how good a job that does.

```
MA <- arima(AirPassengers, order = c(0, 0, 1))
print(MA)
```

Then we want to plot the model against the original time series to see how good a job it does.

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

For this particular data, the moving average model only does less well than the autoregressive model. Let's look at how we make predictions from one of these models.

```
predict_MA <- predict(MA)
predict_MA$pred[1]
predict(MA, n.ahead = 10)
```

The second line will print out just the next predicted value, the last one will allow you to predict several steps out, in this case, 10.

We can plot the next sequence of values on a graph. Here, we've adjusted the right edge of the graph to extend a bit past the original data so we can see the predictions a bit better.

```r
ts.plot(AirPassengers, xlim = c(1949, 1962))
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)
```

The graph plots the predictions, but also the margins of error. We can use plots like these to compare models, but we can also print out various diagnostic measures to perform the comparison.

```r
cor(AR_fitted, MA_fit)
AIC(AR)
AIC(MA)
BIC(AR)
BIC(MA)
```

While there is a fairly high correlation between the models, it does appear that the AR model does a better job than the MA model. Both the AIC and BIC are lower for the AR model.

We can combine both types of models in an ARMA model. Let's see if this improves our model any.

```r
ARMA <- arima(AirPassengers, order = c(1, 0, 1))
print(ARMA)
```

As we did with the two previous models, we're going to plot the predictions against the original time series, both for the original times series period, and also for 10 steps into the future.

```r
ts.plot(AirPassengers)
ARMA_fitted <- AirPassengers - residuals(ARMA)
points(ARMA_fitted, type = "l", col = 2, lty = 2)

ts.plot(AirPassengers, xlim = c(1949, 1962))
ARMA_forecasts <- predict(ARMA, n.ahead = 10)$pred
ARMA_forecast_se <- predict(ARMA, n.ahead = 10)$se
points(ARMA_forecasts, type = "l", col = 2)
points(ARMA_forecasts - 2*ARMA_forecast_se, type = "l", col = 2, lty = 2)
points(ARMA_forecasts + 2*ARMA_forecast_se, type = "l", col = 2, lty = 2)
```

Let's compare the ARMA model to the AR model.

```r
cor(AR_fitted, ARMA_fitted)
AIC(ARMA)
BIC(ARMA)
```

The correlation between this model and the AR model are very high, so they are very similar, but the AIC and BIC are also a bit lower for the ARMA model, so the combination of models did improve the model.

Rerun the code for an ARIMA model with order=c(1,1,1), and you can also try order=c(2,0,1). What is the best model? How do the components of the model change as you adjust these model parameters?

Tasks

1. Repeat this analysis on the built-in Nile dataset. Create appropriate graphs and describe what you see. Test several ARIMA type models and sub-models to determine the model with the best fit. Discuss your findings in depth.

Next week: final project presentations

Resources:

1. https://rpubs.com/odenipinedo/time-series-analysis-in-R
2. https://www.dominodatalab.com/blog/time-series-with-r
3. https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/Nile.html