R - Regression and Forecast Example

Installing Packages

For this example we require two packages. One that is used to connect to the PI Web API is the `RCurl` which was used in the R - Basic Example. The other is a package that is popular in data science called `forecast`. To install `forecast` in RStudio run the following command in the console and press `Enter`

```
> install.packages("forecast")
```

Now we can load the library in our script by adding the following line at the beginning the of our program.

```
library(forecast)
```

Creating Time Series Object

After retrieving our data from the PI Web API and formatting into a data frame as shown in the R - Basic Example, we can use it create a model and prediction forecast.

To create a model and forecast using the `forecast` package we have to convert our data frame into a time-series object. We can use R's built-in `ts` function to do this.

```
time.series.df <- ts(DataFrame, start = as.numeric(DataFrame$Timestamp[1]))
```

We have to indicate where our time series begins, so we pass a numerical value, using `as.numeric`, of our first timestamp as our start parameter.

Creating Linear model

A function that the `forecast` package offers it the `tslm` which creates linear model from a time series object. We just have to give the function a formula which it will use to create the model. In the example the steam flow is a function of time, and forecast uses the keyword `trend` for time.

```
model <- tslm(Steam.Flow ~ trend , data = time.series)
```

Forecasting Data

Now, we can pass the model we created into the `forecast` and it will use the model to predict future values of the steam flow. We have to tell the `forecast` function where the steam flow data begins, so we pass the steam flow data as the start parameter.

```
f <- forecast(model, start = time.series[,"Steam.Flow"])
```

How far out the `forecast` function predicts depends on the number of periods for forecasting parameter. To change this value you can set the `h` parameter. In this case the default was calculated to be 10 periods for forecasting. The period length is dependent on the interval of the timestamps. In this example the data was returned at 10 minute intervals, so the `forecast` function will forecast up to ten ten-minute periods or 100 minutes since the last timestamp. If, say you wanted to forecast out to 2 hours that would equate to 12 periods and we would call `forecast` like so

```
f <- forecast(model, start = time.series[,"Steam.Flow"], h=12)
```

The `forecast` function also stores a linear fit(the fit depends on the model that was used) of the data. We can get this data using R's `fitted` function.

```
linear.fit <- fitted(f)
```

Plotting the Forecast and Fit

We can plot the forecast easily using the `plot` function, but the marks on the x-axis will end up being large numeric values of the timestamps. To fix this we can initially plot the forecast with any markings on the x-axis. By setting the `xaxt` parameter equal to `"n"` we can remove the markings.

```
plot(f, xaxt="n")
```

Editing the X-Axis Marks

Then, we can get the x-axis marks using the `axTicks` function and converting those values to a date-time class.

```
dates <- as.POSIXct(axTicks(1), tz = "UTC", origin = "1970-01-01")
```

The `1` is passed into the `axTicks` function because the `1` corresponds to the 'bottom' axis. For more information see R's documentation for the `axis` function.

After converting the numeric values into date-time objects we can assign them to the x-axis labels.

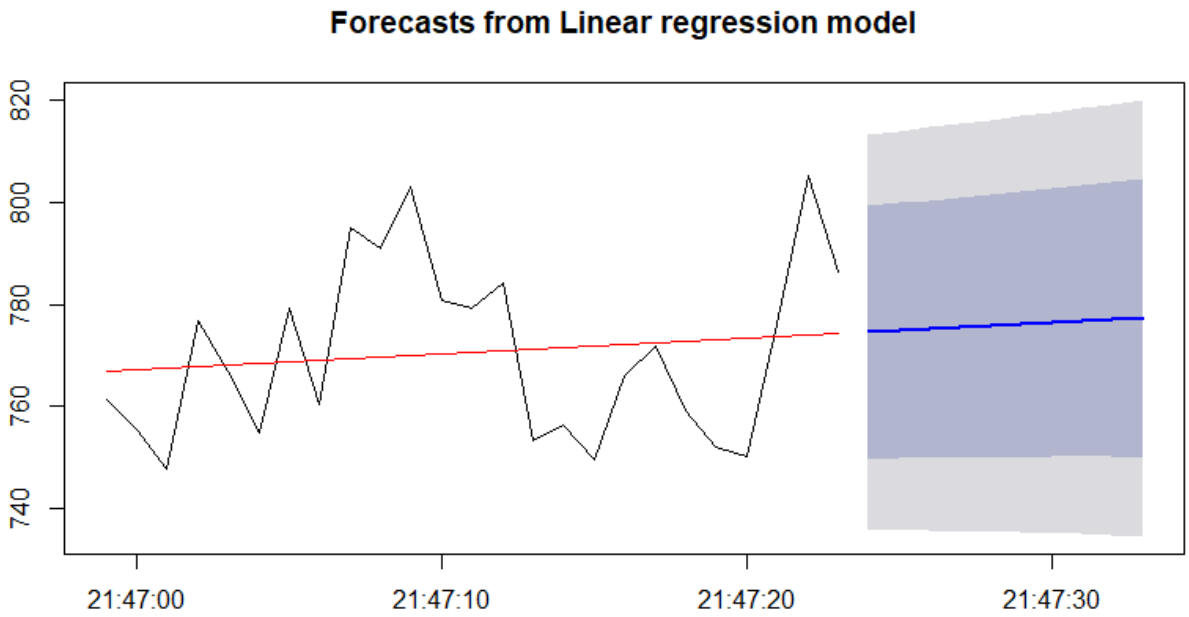
```
axis.POSIXct(side = 1, dates, format = "%H:%M:%S")
```

Adding Linear fit to the Plot

Finally, we can add the linear fit created by the `foreacst` function to the plot.

```
lines(linear.regression, col = "red")
```

This should result in the following plot.


Forecasts from Linear regression model
820
800
780
760
740
21:47:00
21:47:10
21:47:20
21:47:30