Modeling in Revolution R Enterprise Module 6: Model Selection and Scoring

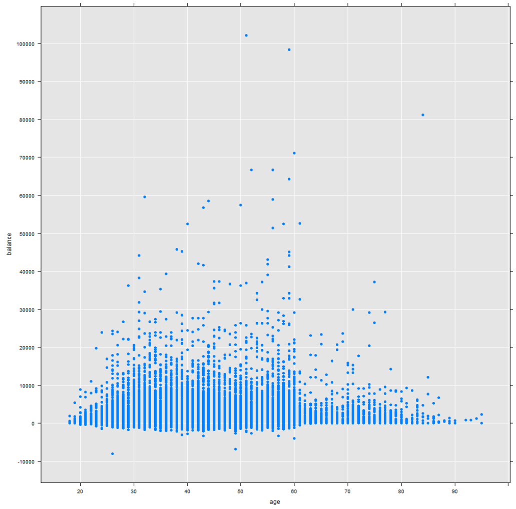
July 25, 2014

Model Selection and Scoring

In this module, we will consider model evaluation, building upon our previous discussion on regression but focusing less on model construction and more on model assessment.

Let's begin by once again considering two of our variables in the Bank data, age and balance. Let's repeat the plot that we produced in the linear regression module on the next slide:

Balance versus Age



Transformation

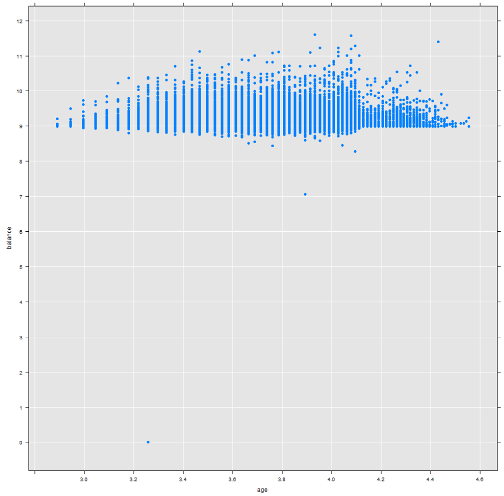
We can transform variables so that increments in value are proportional by taking a logarithmic transformation. We accomplished this in the Introduction to Revolution R Enterprise course, and the code is repeated below:

```
infile <- file.path("data", "BankXDF.xdf")
BankDS <- RxXdfData(file = infile)

outfile <- file.path("data", "BankSub.xdf")

BankSubDS <- rxDataStep(inData = BankDS, outFile = outfile, varsToKeep = c("balance",
    "age"), transforms = list(newBalance = balance + 8019 + 1, logBalance = log(newBalance)),
    overwrite = TRUE)
```

```
## Rows Read: 10000, Total Rows Processed: 10000, Total Chunk Time: 0.011 seconds
## Rows Read: 10000, Total Rows Processed: 20000, Total Chunk Time: 0.010 seconds
## Rows Read: 10000, Total Rows Processed: 30000, Total Chunk Time: 0.011 seconds
## Rows Read: 10000, Total Rows Processed: 40000, Total Chunk Time: 0.010 seconds
## Rows Read: 5211, Total Rows Processed: 45211, Total Chunk Time: 0.006 seconds
```

```
rxLinePlot(balance ~ age, type = "p", data = BankSubDS)
rxLinePlot(logBalance ~ age, type = "p", data = BankSubDS)
```

Transfomations



Outliers

We can clearly see two outliers in the data after the variable transformation: the two balance values which are distinctly separated from the rest of the data in the set. Let's remove those two values using the rxDataStep function.

We can accomplish the removal using a criterion evaluation for balance:

```
BankSubDS <- rxDataStep(inData = BankSubDS, outFile = BankSubDS, transforms = list(logBalance = ifelse(logBalan
    8, NA, logBalance)), overwrite = TRUE)
```

```
## Rows Read: 10000, Total Rows Processed: 10000, Total Chunk Time: 0.015 seconds
## Rows Read: 10000, Total Rows Processed: 20000, Total Chunk Time: 0.014 seconds
## Rows Read: 10000, Total Rows Processed: 30000, Total Chunk Time: 0.017 seconds
## Rows Read: 10000, Total Rows Processed: 40000, Total Chunk Time: 0.014 seconds
## Rows Read: 5211, Total Rows Processed: 45211, Total Chunk Time: 0.009 seconds
```

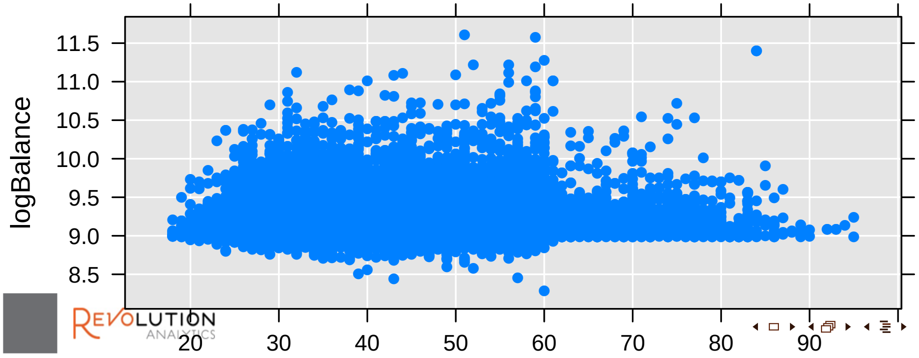
```
BankSubDS <- rxDataStep(inData = BankSubDS, outFile = BankSubDS, rowSelection = !is.na(logBalance),
    overwrite = TRUE)
```

```
## Rows Read: 10000, Total Rows Processed: 10000, Total Chunk Time: 0.013 seconds
## Rows Read: 10000, Total Rows Processed: 20000, Total Chunk Time: 0.012 seconds
## Rows Read: 10000, Total Rows Processed: 30000, Total Chunk Time: 0.012 seconds
## Rows Read: 10000, Total Rows Processed: 40000, Total Chunk Time: 0.011 seconds
## Rows Read: 5211, Total Rows Processed: 45211, Total Chunk Time: 0.008 seconds
```

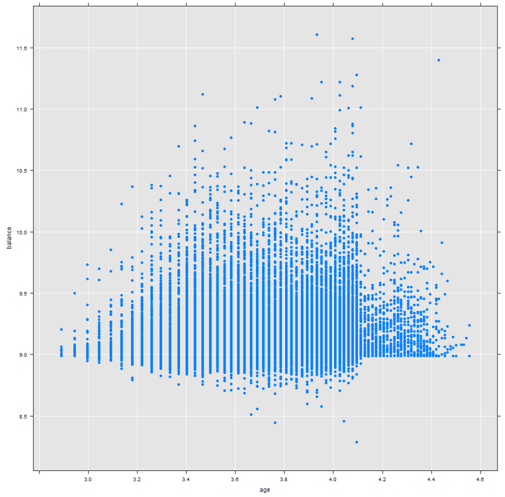
Outliers

Replotting the result:

```
rxLinePlot(logBalance ~ age, type = "p", data = BankSubDS)

## Rows Read: 10000, Total Rows Processed: 10000, Total Chunk Time: 0.006 seconds
## Rows Read: 9998, Total Rows Processed: 19998, Total Chunk Time: 0.005 seconds
## Rows Read: 10000, Total Rows Processed: 29998, Total Chunk Time: 0.005 seconds
## Rows Read: 10000, Total Rows Processed: 39998, Total Chunk Time: 0.004 seconds
## Rows Read: 5211, Total Rows Processed: 45209, Total Chunk Time: 0.004 seconds
```



Outliers



Residuals

In our model construction, the residual would be the difference between the observable data point and that predicted by the model. In this manner, a lot can be obtained about model rigidity through consideration of residuals.

Let's once again use linear regression to construct a statistical model for our data - only this time, we will compute the residuals along with the normal linear model.

Residuals

To do so, we have to construct the linear model and then use the rxPredict function, with the computeResiduals parameter set to true:

```
linMod <- rxLinMod(logBalance ~ age, data = BankSubDS)
```

```
## Rows Read: 10000, Total Rows Processed: 10000, Total Chunk Time: 0.001 seconds
## Rows Read: 9998, Total Rows Processed: 19998, Total Chunk Time: 0.002 seconds
## Rows Read: 10000, Total Rows Processed: 29998, Total Chunk Time: 0.002 seconds
## Rows Read: 10000, Total Rows Processed: 39998, Total Chunk Time: 0.002 seconds
## Rows Read: 5211, Total Rows Processed: 45209, Total Chunk Time: 0.002 seconds
## Computation time: 0.014 seconds.
```

```
linResult <- rxPredict(linMod, data = BankSubDS, computeResiduals = TRUE)
```

```
## Rows Read: 10000, Total Rows Processed: 10000, Total Chunk Time: Less than .001 seconds
## Rows Read: 9998, Total Rows Processed: 19998, Total Chunk Time: 0.008 seconds
## Rows Read: 10000, Total Rows Processed: 29998, Total Chunk Time: 0.008 seconds
## Rows Read: 10000, Total Rows Processed: 39998, Total Chunk Time: 0.008 seconds
## Rows Read: 5211, Total Rows Processed: 45209, Total Chunk Time: 0.008 seconds
```

Finally, let's get some information on the residuals using the rxGetVarInfo function:

```
rxGetVarInfo(linResult)
```

```
## Var 1: balance, Type: integer, Low/High: (-8019, 102127)
## Var 2: age, Type: integer, Low/High: (18, 95)
## Var 3: newBalance, Type: numeric, Low/High: (1.0000, 110147.0000)
## Var 4: logBalance, Type: numeric, Low/High: (0.0000, 11.6096)
## Var 5: logBalance_Pred, Type: numeric, Low/High: (9.0686, 9.2374)
```

Prediction Standard Errors

We can also obtain prediction standard errors from our model, provided that we have included the variance-covariance matrix in the original linear model fit. There are two types of prediction standard errors that we can compute:

- Confidence Intervals: for a given confidence level*, provides information on how confident we are that the expected value is within the given interval.
- Prediction Intervals: for a given confidence level*, provides how likely future observations are to fall within the interval given what has already been observed.
- The default confidence level is 0.95. This can be modified with the confLevel argument.

Example: Prediction Standard Errors

Let's obtain fitted values, prediction standard errors, and confidence intervals for our linear model including a second-degree term, by first including the computation of the variance-covariance matrix within our rxLinMod computation:

```
linMod <- rxLinMod(logBalance ~ age + I(age^2), data = BankSubDS, covCoef = TRUE)
```

```
## Rows Read: 10000, Total Rows Processed: 10000, Total Chunk Time: 0.003 seconds
## Rows Read: 9998, Total Rows Processed: 19998, Total Chunk Time: 0.003 seconds
## Rows Read: 10000, Total Rows Processed: 29998, Total Chunk Time: 0.002 seconds
## Rows Read: 10000, Total Rows Processed: 39998, Total Chunk Time: 0.003 seconds
## Rows Read: 5211, Total Rows Processed: 45209, Total Chunk Time: 0.003 seconds
## Computation time: 0.021 seconds.
```

Example: Prediction Standard Errors

Executing the rxPredict function, we include the terms computeStdErrors=TRUE, interval="confidence", writeModelVars = TRUE, as shown below:

```
predResult <- rxPredict(linMod, data = BankSubDS, computeStdErrors = TRUE, interval = "confidence",
    writeModelVars = TRUE, overwrite = TRUE)
```

```
## Rows Read: 10000, Total Rows Processed: 10000, Total Chunk Time: 0.011 seconds
## Rows Read: 9998, Total Rows Processed: 19998, Total Chunk Time: 0.012 seconds
## Rows Read: 10000, Total Rows Processed: 29998, Total Chunk Time: 0.010 seconds
## Rows Read: 10000, Total Rows Processed: 39998, Total Chunk Time: 0.009 seconds
## Rows Read: 5211, Total Rows Processed: 45209, Total Chunk Time: 0.010 seconds
```

```
summary(predResult)
```

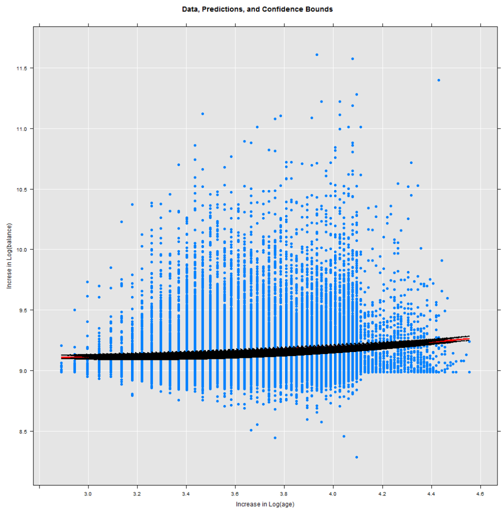
```
## Call:
## rxSummary(formula = form, data = object, byTerm = TRUE, reportProgress = 0L)
##
## Summary Statistics Results for: ~balance + age + newBalance +
##     logBalance + logBalance_Pred + logBalance_Resid +
##     logBalance_StdErr + logBalance_Lower + logBalance_Upper
...
```

Example: Prediction Standard Errors

Now, we can view the original data, the fitted prediction line, and the confidence intervals as follows:

```
rxLinePlot(logBalance + logBalance_Pred + logBalance_Upper + logBalance_Lower ~
    age, data = predResult, type = "b", lineStyle = c("blank", "solid", "dotted",
    "dotted"), lineColor = c(NA, "red", "black", "black"), symbolStyle = c("solid circle",
    "blank", "blank", "blank"), title = "Data, Predictions, and Confidence Bounds",
    xTitle = "Increase in Log(age)", yTitle = "Increse in Log(balance)", legend = FALSE)
```

Example: Prediction Standard Errors



Example: Prediction Standard Errors

Interestingly, we can see that while the maximum balances do not occur at later ages, the presence of less debt (recall, debt = negative balance) causes the average balance to actually increase on average at later ages.

We can also tell by the solid black lines the prediction boundaries for predictions made from our model. While the inclusion of the second-degree term provides a better-fitting model, the wide range of balances tells us that predictions based on this model would not be entirely accurate.

The red line, which is very close between the two solid black lines, shows the exact model we have constructed.

Exercise: Prediction Standard Errors

Produce a similar plot to the one in the last example, only this time modeling balance with the duration of the phone call in BankDS. Use polynomial regression, including a second-order term, and don't forget to include the variance-covariance computation in the function call as well.

Exercise: Solution

Constructing the linear model, using the Bank data source:

```
linMod <- rxLinMod(balance ~ duration + I(duration^2), data = BankDS, covCoef = TRUE)
```

```
## Rows Read: 10000, Total Rows Processed: 10000, Total Chunk Time: 0.003 seconds
## Rows Read: 10000, Total Rows Processed: 20000, Total Chunk Time: 0.003 seconds
## Rows Read: 10000, Total Rows Processed: 30000, Total Chunk Time: 0.006 seconds
## Rows Read: 10000, Total Rows Processed: 40000, Total Chunk Time: 0.003 seconds
## Rows Read: 5211, Total Rows Processed: 45211, Total Chunk Time: 0.003 seconds
## Computation time: 0.025 seconds.
```

Then, computing the standard errors:

```
predResult <- rxPredict(linMod, data = BankDS, computeStdErrors = TRUE, interval = "confidence",
    writeModelVars = TRUE, overwrite = TRUE)
```

```
## Rows Read: 10000, Total Rows Processed: 10000, Total Chunk Time: 0.004 seconds
## Rows Read: 10000, Total Rows Processed: 20000, Total Chunk Time: 0.010 seconds
## Rows Read: 10000, Total Rows Processed: 30000, Total Chunk Time: 0.009 seconds
## Rows Read: 10000, Total Rows Processed: 40000, Total Chunk Time: 0.008 seconds
## Rows Read: 5211, Total Rows Processed: 45211, Total Chunk Time: 0.009 seconds
```

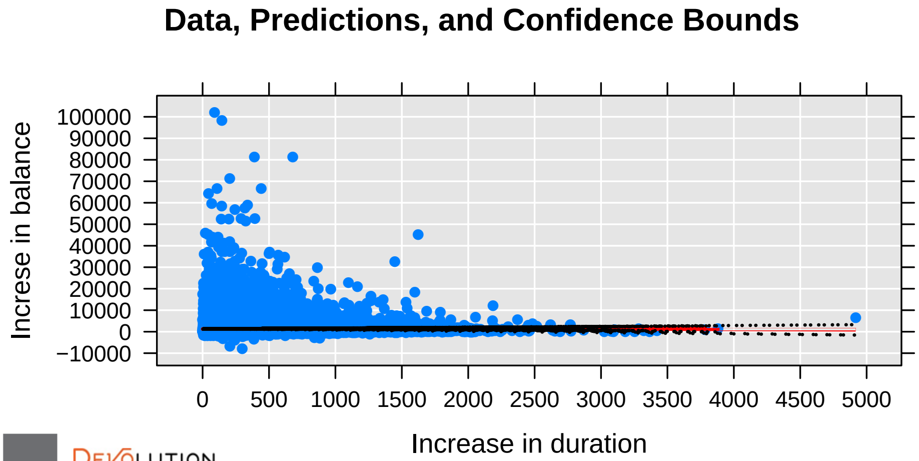
Exercise: Solution

Then, constructing the plot:

```
rxLinePlot(balance + balance_Pred + balance_Upper + balance_Lower ~ duration,
    data = predResult, type = "b", lineStyle = c("blank", "solid", "dotted",
        "dotted"), lineColor = c(NA, "red", "black", "black"), symbolStyle = c("solid circle",
        "blank", "blank", "blank"), title = "Data, Predictions, and Confidence Bounds",
    xTitle = "Increase in duration", yTitle = "Increse in balance", legend = FALSE)
```

Exercise: Solution

```
## Rows Read: 10000, Total Rows Processed: 10000, Total Chunk Time: 0.012 seconds
## Rows Read: 10000, Total Rows Processed: 20000, Total Chunk Time: 0.010 seconds
## Rows Read: 10000, Total Rows Processed: 30000, Total Chunk Time: 0.010 seconds
## Rows Read: 10000, Total Rows Processed: 40000, Total Chunk Time: 0.009 seconds
## Rows Read: 5211, Total Rows Processed: 45211, Total Chunk Time: 0.006 seconds
```



Recap

Let’s review some of the concepts covered in this module:

- What is a residual?
- What are the two types of prediction standard errors that you can compute using ScaleR?

Thank you

Revolution Analytics is the leading commercial provider of software and support for the popular open source R statistics language.

www.revolutionanalytics.com, 1.855.GET.REVO, Twitter: @RevolutionR