REVOLUTION ANALYTICS

Linear Regression with ScaleR

Revolution Analytics

Overview

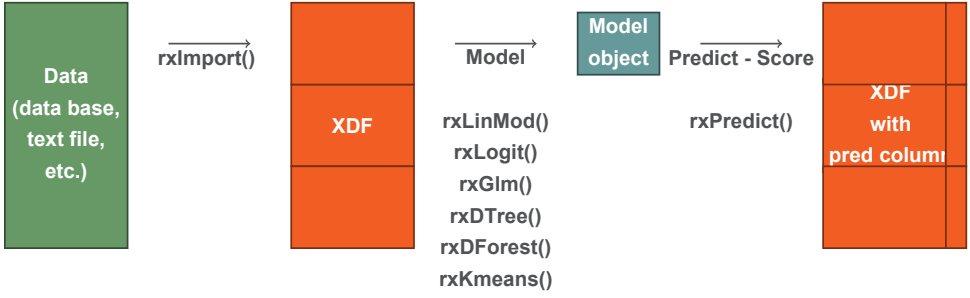
In this session you analyze big data with `RevoScaleR`.

We focus on techniques relevant to the linear model.

Outline

The RevoScaleR model life cycle

Import, model and predict:



Linear Modeling with RevoScaleR

```
help(rxLinMod)
```

In base R, you can do linear regression with `lm()`. The RevoScaleR function for big data linear regression is `rxLinMod()`

`rxLinMod()` uses an updating algorithm to estimate the regression model.

rxLinMod() Arguments

```
args(rxLinMod)
```

```
## function (formula, data, pweights = NULL, fweights = NULL, cube = FALSE,
##     cubePredictions = FALSE, variableSelection = list(), rowSelection = NULL,
##     transforms = NULL, transformObjects = NULL, transformFunc = NULL,
##     transformVars = NULL, transformPackages = NULL, transformEnvir = NULL,
##     dropFirst = FALSE, dropMain = rxGetOption("dropMain"), covCoef = FALSE,
##   covData = FALSE, covariance = FALSE, coefLabelStyle = rxGetOption("coefLabelStyle"),
##   blocksPerRead = rxGetOption("blocksPerRead"), reportProgress = rxGetOption("reportProgress"),
...
```

rxLinMod() Output

The object returned by `rxLinMod()` includes:

- the estimated model coefficients
- the call used to generate the model
- other information that allows RevoScaleR to recompute the model.

This ability to recreate the model without having to rerun it on the full dataset is key. You see this in action later when we use the small number of returned objects to generate a large number of predicted values and residuals.

Folder configuration

```
## dir config
big.data.path <- Sys.getenv("ACADEMYR_BIG_DATA_PATH")
data.path <- "../data"
output.path <- "../output/xdf"
if (!file.exists(output.path)) dir.create(output.path, recursive = TRUE)
sample.data.dir <- rxGetOption("sampleDataDir")
```

Import Description

The `colInfo` argument in `rxLinMod()` allows us to explicitly specify the levels of a factor variable upon import. Alternatively, if we did not care about the order of the levels, we could just set the argument `stringsAsFactors` to `TRUE`.

The input csv file uses the letter `M` to represent missing values, rather than the default `NA`, so we specify this with the `missingValueString` argument.

Import Code

```
airline.csv <- file.path(sample.data.dir, "AirlineDemoSmall.csv")
airline.xdf <- file.path(output.path, "airline.xdf")

colInfo <- list(DayOfWeek = list(type = "factor", levels = c("Monday",
  "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")))

if (!file.exists(airline.xdf)) {
  rxImport(inData = airline.csv, outFile = airline.xdf, colInfo = colInfo,
    missingValueString = "M", overwrite = TRUE)
}
```

Airline Data Information

Examining the data…

```
rxGetInfo(airline.xdf, getVarInfo = TRUE, numRows = 5)
```

```
## File name: /home/jeremy.reynolds/github/AcademyR/Revolution_Course_Materials/modules/Revolution
## Number of observations: 600000
## Number of variables: 6
## Number of blocks: 2
## Compression type: zlib
## Variable information:
## Var 1: ArrDelay, Type: integer, Low/High: (-86, 1490)
...
```

Airline Data Summary

```
rxSummary(~., data = airline.xdf)
```

```
## Call:
## rxSummary(formula = ~., data = airline.xdf)
##
## Summary Statistics Results for: ~.
## Data: airline.xdf (RxXdfData Data Source)
## File name: ../output/xdf/airline.xdf
## Number of valid observations: 600000
...
```

Delay by Day-of-week

Use `rxSummary()`

```
(delaySumm <- rxSummary(ArrDelay ~ DayOfWeek, data = airline.xdf))
```

```
## Call:
## rxSummary(formula = ArrDelay ~ DayOfWeek, data = airline.xdf)
##
## Summary Statistics Results for: ArrDelay ~ DayOfWeek
## Data: airline.xdf (RxXdfData Data Source)
## File name: ../output/xdf/airline.xdf
## Dependent variable(s): ArrDelay
...
```

Delay by Day-of-week

Use `rxSummary()`

```
(delayCube <- rxCube(ArrDelay ~ DayOfWeek, data = airline.xdf, means = TRUE))
```

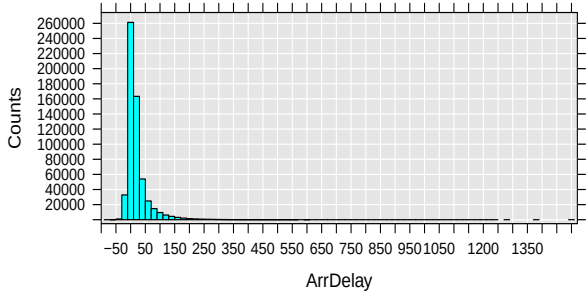
```
## Call:
## rxCube(formula = ArrDelay ~ DayOfWeek, data = airline.xdf, means = TRUE)
##
## Cube Results for: ArrDelay ~ DayOfWeek
## File name: ../output/xdf/airline.xdf
## Dependent variable(s): ArrDelay
## Number of valid observations: 582628
...
```

Exercise

Extract the mean arrival delay for Monday.

Distribution of Arrival Delays

```
rxHistogram(~ArrDelay, data = airline.xdf)
```



Outline

Simple Linear Model

Let's start off with a simple linear regression… Arrival Delay on Day of Week using `airline.xdf` data and `rxLinMod()`.

```
model <- rxLinMod(ArrDelay ~ DayOfWeek, data = airline.xdf)
```

Extracting Results

Operate on the result of the function call

```
model
```

```
## Call:
## rxLinMod(formula = ArrDelay ~ DayOfWeek, data = airline.xdf)
##
## Linear Regression Results for: ArrDelay ~ DayOfWeek
## Data: airline.xdf (RxXdfData Data Source)
## File name: ../output/xdf/airline.xdf
## Dependent variable(s): ArrDelay
...
```

Summarizing Results: Default

```
(mod.summ <- summary(model))
```

```
## Call:
## rxLinMod(formula = ArrDelay ~ DayOfWeek, data = airline.xdf)
##
## Linear Regression Results for: ArrDelay ~ DayOfWeek
## Data: airline.xdf (RxXdfData Data Source)
## File name: ../output/xdf/airline.xdf
## Dependent variable(s): ArrDelay
...
```

Extracting Coefficients

```
coef(model)
```

```
##          (Intercept)    DayOfWeek=Monday   DayOfWeek=Tuesday
##           10.3318058           1.6937981           0.9620019
## DayOfWeek=Wednesday  DayOfWeek=Thursday    DayOfWeek=Friday
##           -0.1752668          -1.6737983           4.4725290
##  DayOfWeek=Saturday    DayOfWeek=Sunday
##            1.5435207                  NA
```

Summarizing Coefficients Only

```
names(mod.summ)
```

```
## [1] "ArrDelay"
```

```
coef(mod.summ$ArrDelay)
```

```
##                         Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)          10.3318058  0.1330168 77.6729612 2.220446e-16
## DayOfWeek=Monday      1.6937981  0.1871726  9.0493894 2.220446e-16
## DayOfWeek=Tuesday     0.9620019  0.2000524  4.8087499 1.519149e-06
## DayOfWeek=Wednesday -0.1752668  0.1980254 -0.8850724 3.761179e-01
## DayOfWeek=Thursday  -1.6737983  0.1963991 -8.5224331 2.220446e-16
## DayOfWeek=Friday      4.4725290  0.1957367 22.8497179 2.220446e-16
...
```

Other Important Values

Condition Number, R-squared, etc.

```
names(model)
```

```
##  [1] "coefficients"      "residual.squares"  "condition.number"
##  [4] "rank"              "aliased"           "coef.std.error"
##  [7] "coef.t.value"      "coef.p.value"      "total.squares"
## [10] "y.var"             "sigma"             "residual.variance"
## [13] "r.squared"         "f.pvalue"          "df"
## [16] "y.names"           "deviance"          "aic"
## [19] "params"            "formula"           "call"
...
```

Using Model Objects

What is the estimated mean arrival delay for Monday?

Add intercept and Monday coefficient and compare to Monday average from cube

```
coef(model)[1] + coef(model)[2]
```

```
## (Intercept)
##     12.0256
```

```
rxCube(ArrDelay ~ DayOfWeek, data = airline.xdf, means = TRUE)["1",
  "ArrDelay"]
```

```
## [1] 12.0256
```

Better Performance with `cube=TRUE`

When the first independent variable is a factor, we can use `cube=TRUE` to run the model as a partitioned inverse regression, which may be faster and use less memory than the standard computation.

```
help(rxLinMod)
```

```
model_cubed <- rxLinMod(ArrDelay ~ DayOfWeek, data = airline.xdf,
  cube = TRUE)
```

Cube Model Summary

```
print(summary(model_cubed), header = FALSE)
```

```
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)     | Counts
## DayOfWeek=Monday      12.0256     0.1317   91.32 2.22e-16 *** |  95298
## DayOfWeek=Tuesday     11.2938     0.1494   75.58 2.22e-16 *** |  74011
## DayOfWeek=Wednesday   10.1565     0.1467   69.23 2.22e-16 *** |  76786
## DayOfWeek=Thursday     8.6580     0.1445   59.92 2.22e-16 *** |  79145
## DayOfWeek=Friday      14.8043     0.1436  103.10 2.22e-16 *** |  80142
...
```

Differences with cube=TRUE

No Intercept estimated

Although F-statistic is calculated as though one exists.

Factors with an Intercept

$k - 1$ regressors are used to code for a factor with k levels.

`dummy` or `treatment` codes are used

Example

```
mydf <- data.frame(y = rnorm(30), x = gl(3, 1, 30, labels = LETTERS[1:3]))
str(mydf)
```

```
## 'data.frame':    30 obs. of  2 variables:
##  $ y: num  -0.626 0.184 -0.836 1.595 0.33 ...
##  $ x: Factor w/ 3 levels "A","B","C": 1 2 3 1 2 3 1 2 3 1 ...
```

```
myModelMat <- model.matrix(lm(y ~ x, data = mydf))
head(myModelMat)
```

```
##   (Intercept) xB xC
## 1           1  0  0
## 2           1  1  0
## 3           1  0  1
## 4           1  0  0
## 5           1  1  0
## 6           1  0  1
```

Baselines

By default `rxLinMod` will select the last level as your baseline group (consistent with SAS behavior)

NOTE: The open source R function `lm` will select the *first* level as the baseline group

You can mirror the open source behavior by setting the `dropFirst` argument to TRUE.

Exercise

The prior example uses Sunday as the baseline group, and shows that almost all days have different delays from Sunday. Rerun the example, and determine which days are different from Monday. Also, investigate teh coefficients and examine how they relate back to the prior estimation.

rxLinMod() continued

You can add another variable into the mix: running a regression of Arrival delay on two predictors, separated by plus signs (+).

Try this with DayOfWeek and CRSDepTime:

```
model <- rxLinMod(ArrDelay ~ DayOfWeek + CRSDepTime, data = airline.xdf,
  cube = TRUE)
```

Additive Model Summary

```
print(summary(model), header = FALSE)
```

```
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)     | Counts
## DayOfWeek=Monday    -1.11358    0.19988  -5.571 2.53e-08 *** |  95298
## DayOfWeek=Tuesday   -1.85442    0.21192  -8.751 2.22e-16 *** |  74011
## DayOfWeek=Wednesday -3.04303    0.21045 -14.459 2.22e-16 *** |  76786
## DayOfWeek=Thursday  -4.51759    0.20874 -21.642 2.22e-16 *** |  79145
## DayOfWeek=Friday     1.60584    0.20832   7.708 2.22e-16 *** |  80142
...
```

What does the formula mean?

$$z \sim x + y$$

This means model `z` as a function of `x` and `y`

Symbol	Example	Meaning
~	`z ~ x`	"as a function of"
+	`z ~ x + y`	additive terms
:	`z ~ x:y`	interaction
*	`z ~ a*b`	equivalent to `z ~ x + y + x:y`
-1	`z ~ x - 1`	fit a model and drop the intercept

Formulas with Interactions

You can include interaction terms by joining two or more variables with a colon (`:`).

```
# add departure time into the mix: interact with days
model <- rxLinMod(ArrDelay ~ DayOfWeek:F(CRSDepTime), data = airline.xdf,
  cube = TRUE)
```

Interactive Model Summary

```
print(summary(model), header = FALSE)
```

```
## Coefficients:
##                                          Estimate Std. Error t value Pr(>|t|) | Counts
## DayOfWeek=Monday, F_CRSDepTime=0          7.43609    3.49568   2.127 0.033402 |    133
## DayOfWeek=Tuesday, F_CRSDepTime=0         7.00000    3.89731   1.796 0.072478 |    107
## DayOfWeek=Wednesday, F_CRSDepTime=0       3.78571    4.07234   0.930 0.352570 |     98
## DayOfWeek=Thursday, F_CRSDepTime=0        3.00971    3.97227   0.758 0.448643 |    103
## DayOfWeek=Friday, F_CRSDepTime=0          4.47525    4.01140   1.116 0.264580 |    101
...
```

Formulas with Transformations

We can do transformations on the fly, here regressing on the log of the absolute value of `ArrivalDelay`.

```
# can use transformations
model <- rxLinMod(log(abs(ArrDelay)) ~ DayOfWeek:F(CRSDepTime), data = airline.xdf,
  cube = TRUE)
```

In-line transform Model Summary

```
print(summary(model), header = FALSE)
```

```
## Coefficients:
##                                      Estimate Std. Error t value Pr(>|t|) | Counts
## DayOfWeek=Monday, F_CRSDepTime=0      2.31253    0.09967  23.202 2.22e-16 |    131
## DayOfWeek=Tuesday, F_CRSDepTime=0     2.12193    0.11240  18.878 2.22e-16 |    103
## DayOfWeek=Wednesday, F_CRSDepTime=0   2.36748    0.11829  20.014 2.22e-16 |     93
## DayOfWeek=Thursday, F_CRSDepTime=0    2.32445    0.11523  20.172 2.22e-16 |     98
## DayOfWeek=Friday, F_CRSDepTime=0      2.17078    0.11465  18.934 2.22e-16 |     99
...
```

Formulas with transformations

We can use more elaborate transformations by specifying a `transforms` argument.

```
model <- rxLinMod(ArrDelay ~ DayOfWeek + depTimeCat, data = airline.xdf,
  cube = TRUE, transforms = list(depTimeCat = cut(CRSDepTime, breaks = seq(from = 5,
    to = 23, by = 2))))
```

transforms Model Summary

```
print(summary(model), header = FALSE)
```

```
## Coefficients: (1 not defined because of singularities)
##                      Estimate Std. Error t value Pr(>|t|)     | Counts
## DayOfWeek=Monday      14.9488     0.2707  55.216 2.22e-16 *** |  94594
## DayOfWeek=Tuesday     14.1960     0.2799  50.726 2.22e-16 *** |  73495
## DayOfWeek=Wednesday   13.0368     0.2782  46.859 2.22e-16 *** |  76209
## DayOfWeek=Thursday    11.5274     0.2770  41.619 2.22e-16 *** |  78528
## DayOfWeek=Friday      17.6799     0.2763  63.978 2.22e-16 *** |  79517
...
```

You can also use the transformFuncs argument.

Modeling Without XDF Files

This functionality is not limited to xdf files. We can run the previous regression on a text file, too…

```
airline_rxdata <- RxTextData(file = airline.csv, colInfo = colInfo,
  missingValueString = "M")

model <- rxLinMod(ArrDelay ~ DayOfWeek + depTimeCat, data = airline_rxdata,
  cube = TRUE, transforms = list(depTimeCat = cut(CRSDepTime, breaks = seq(from = 5,
    to = 23, by = 2))), dropFirst = TRUE)
```

textData Model Summary

```
print(summary(model), header = FALSE)
```

```
## Coefficients: (1 not defined because of singularities)
##                      Estimate Std. Error t value Pr(>|t|)     | Counts
## DayOfWeek=Monday       3.6622     0.2101  17.429 2.22e-16 *** |  94594
## DayOfWeek=Tuesday      2.9094     0.2215  13.136 2.22e-16 *** |  73495
## DayOfWeek=Wednesday    1.7502     0.2201   7.951 2.22e-16 *** |  76209
## DayOfWeek=Thursday     0.2407     0.2185   1.102     0.27     |  78528
## DayOfWeek=Friday       6.3933     0.2179  29.345 2.22e-16 *** |  79517
...
```

Outline

Another dataset

Let's start thinking about this in the context of another dataset.

S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014

The Data: Bank Marketing Data

The Bank Marketing Data Set, which we will refer to as the `Bank` data, concerns the relationship between direct marketing campaigns and subscription to a term deposit for a Portuguese bank.

17 variables are recorded.

Examining the Bank Marketing Data

Let's examine a portion of this data to get a sense of these variables…

```
##   age           job marital education default balance housing loan
## 1  58    management married  tertiary      no    2143     yes   no
## 2  44    technician  single secondary      no      29     yes   no
## 3  33 entrepreneur married secondary      no       2     yes  yes
## 4  47  blue-collar married   unknown      no    1506     yes   no
## 5  33       unknown  single   unknown      no       1      no   no
## 6  35    management married  tertiary      no     231     yes   no
```

More Variables

```
##   contact day month duration campaign pdays previous poutcome  y
## 1 unknown   5   may      261        1    -1        0  unknown no
## 2 unknown   5   may      151        1    -1        0  unknown no
## 3 unknown   5   may       76        1    -1        0  unknown no
## 4 unknown   5   may       92        1    -1        0  unknown no
## 5 unknown   5   may      198        1    -1        0  unknown no
## 6 unknown   5   may      139        1    -1        0  unknown no
```

A Single Row

So, for instance, our third contact had the following properties:

- is 33 years old
- works as an entrepreneur
- is married
- has a secondary level of education
- does not have credit in default
- has an average yearly balance of 2 euros
- has both a housing and personal loan

A Single Row (continued)

- was contacted using an unknown source
- was last contacted on the fifth of the month, which was May
- the duration of contact lasted 76 seconds
- 1 contacts performed during this campaign for this client
- was not previously contacted by a campaign
- 0 contacts performed before this campaign for this client
- unknown outcome of previous campaign
- did not subscribe to a term deposit

Exercise

In the bank dataset, estimate a linear model predicting a customer's balance based on their age.

What other variables might be important?

Exercise 2

This time, estimate the customer bank balance against multiple variables: the age of the customer, the duration of time the customer has maintained the account, whether or not the customer is married, and whether or not the customer owns a house.

Exercise 3

This time, add interactions, and test whether any of these variables interact with one another.

Questions?

Getting additional outputs

The function `rxLinMod()` has been engineered for big data.

Certain outputs that may be too big to fit in memory must be asked for explicitly at the time the model is run.

Suppose we want to return a variance-covariance matrix of the main parameters in a regression of arrival delay on an interaction of day of week and departure time.

Attempt 1: Covariance Data

How big would the covariance matrix be?

```
model <- rxLinMod(ArrDelay ~ DayOfWeek:F(CRSDepTime), data = airline.xdf)
length(model$coefficients)
```

```
## [1] 169
```

Model Covariance Data

Outputs such as variance-covariance matrix for coefficients or residuals may be too big to fit in memory and are not returned by default.

Notice this doesn't return the covariance matrix.

```
model$covCoef
```

```
## NULL
```

Getting Covariance Data

We fit the same linear model with `rxLinMod()`, this time setting `covCoef=TRUE` to ensure we have the variance-covariance matrix in our model object.

```
# force return of covCoef
model <- rxLinMod(ArrDelay ~ DayOfWeek:F(CRSDepTime), data = airline.xdf,
  covCoef = TRUE)
model$covCoef
```

```
##                                      (Intercept)
## (Intercept)                             2.759299
## DayOfWeek=Monday, F_CRSDepTime=0       -2.759299
## DayOfWeek=Tuesday, F_CRSDepTime=0      -2.759299
## DayOfWeek=Wednesday, F_CRSDepTime=0    -2.759299
## DayOfWeek=Thursday, F_CRSDepTime=0     -2.759299
## DayOfWeek=Friday, F_CRSDepTime=0       -2.759299
...
```

View the Covariance Matrix

```
# extract covariance matrix
cov.matrix <- model$covCoef
dimnames(cov.matrix) <- NULL
cov.matrix[1:5, 1:5]
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,]  2.759299 -2.759299 -2.759299 -2.759299 -2.759299
## [2,] -2.759299 14.979053  2.759299  2.759299  2.759299
## [3,] -2.759299  2.759299 17.948339  2.759299  2.759299
## [4,] -2.759299  2.759299  2.759299 19.343251  2.759299
## [5,] -2.759299  2.759299  2.759299  2.759299 18.538205
```

Exercise 4

Go back to your banking data, and extract the covariance matrix of the coefficients for the additive model. What are the dimensions on that matrix?

Additional Topics

- Creating Predictions and Performing Out-of-Sample Validation (`rxPredict()`)
- Model Selection (`rxStepControl()`)
- Generalized Linear Models (`rxGlm()`)

Questions?

REVOLUTION ANALYTICS

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