Revolution R Enterprise 6.0 for README

Revolution R Enterprise 6.0 for 32-bit and 64-bit Windows and 64-bit Red Hat Enterprise Linux (RHEL 5.x and RHEL 6.x) features an updated release of the RevoScaleR package that provides fast, scalable data management and data analysis: the same code scales from data frames to local, high-performance .xdf files to data distributed across a Windows HPC Server cluster or IBM Platform Computing LSF cluster. RevoScaleR also allows distribution of the execution of essentially any R function across cores and nodes, delivering the results back to the user.

Installation instructions and instructions for getting started are provided in your confirmation e-mail.

What's New in Revolution R Enterprise 6.0

RevoScaleR

New Distributed and Parallel Computing Functionality

RevoScaleR provides built-in distributed computing capabilities, beginning in Revolution R Enterprise 5 with support for Microsoft HPC Server clusters. This means that you can develop complex analysis scripts on your local computer, create one or more compute contexts for use with distributed computing resources, and then seamlessly move between executing scripts on the local computer and in a distributed context. These full-featured distributed compute contexts provided automated distribution of high performance data analysis functions as well as more traditional high performance computing (HPC) functionality. This release provides new full-featured distributed compute contexts, as well as additional HPC functionality.

- ***IBM Platform LSF Cluster support [Linux only].*** The new RevoScaleR function, **RxLsfCluster**, allows you to create a distributed compute context for the Platform LSF workload manager. See the help file and the *RevoScaleR Distributed Computing Guide* for more information.
- ***Azure Burst support added for Microsoft HPC Server [Windows only].*** The new RevoScaleR function, **RxAzureBurst**, allows you to create a distributed compute context to have computations performed in the cloud using Azure Burst. See the help file, the RevoScaleR *HPC Administrator's Guide* (provided separately) and the *RevoScaleR Distributed Computing Guide* for more information.
- ***New high performance parallel computing functionality.*** The **rxExec** function allows distributed execution of essentially any R function across cores and nodes, delivering the results back to the user. It can be used to perform parallel computations on cores of a workstation, across nodes/cores of a simple network of workstations, or across cores or nodes of a Microsoft HPC Server, Azure Burst, or LSF Linux cluster. The same code is used in all cases; the only change required is to set the compute context. These new functions are provided for changing and managing compute contexts:
    - New functions RxLocalParallel and RxLocalSeq allow you to create compute context objects for local parallel and local sequential computation, respectively. The

RxLocalParallel function uses the 'parallel' package that is included with R 2.14, which will run code in parallel across cores of your local computer.

- New function RxForeachDoPar allows you to create a compute context using the currently registered foreach parallel backend (doParallel, doSNOW, doMC, etc.). To execute rxExec calls, simply register the parallel backend as usual, then set your compute context as follows: rxSetComputeContext(RxForeachDoPar())
- rxSetComputeContext and rxGetComputeContext simplify management of compute contexts.

Big Data Generalized Linear Models (GLM)

The new RevoScaleR function, **rxGlm**, provides a fast, scalable, distributable implementation of generalized linear models. This expands the list of full-featured high performance analytics functions already available: summary statistics (rxSummary), cubes and cross tabs (rxCube, rxCrossTabs), linear models (rxLinMod), covariance and correlation matrices (rxCovCor), binomial logistic regression (rxLogit), and k-means clustering (rxKmeans).

- See impressive speed-ups relative to glm on in-memory data frames. For example: a Tweedie family with 1 million observations and 78 estimated coefficients (categorical data) took 17 seconds with ***rxGlm*** compared with 377 seconds for ***glm*** on a quadcore laptop.
- Scale your analysis to tens or hundreds of millions of observations without increasing memory requirements by storing your data in a native .xdf file,
- Automatically distribute computations across nodes of a cluster (Microsoft HPC Server for Windows or Platform LSF for Red Hat Enterprise Linux) for even faster analysis
- For examples using ***rxGlm***, see Chapter 9 of the *RevoScaleR User's Guide*.

Ability to use RevoScaleR analysis functions directly with external data sources

RevoScaleR high-performance analysis functions will now conveniently work directly with a variety of external data sources (delimited and fixed format text files, SAS files, SPSS files, and ODBC data connections). New functions are provided to create data source objects to represent these data sources (RxTextData, RxOdbcData, RxSasData, and RxSpssData), which in turn can be specified for the 'data' argument for these RevoScaleR analysis functions: rxHistogram, rxSummary, rxCube, rxCrossTabs, rxLinMod, rxCovCor, rxLogit, and rxGlm. For example, you can analyze a SAS file directly as follows:

```
# Create a SAS data source with information about variables and
# rows to read in each chunk
sasDataFile <- file.path(rxGetOption("sampleDataDir"),
     "claims.sas7bdat")
sasDS <- RxSasData(sasDataFile, stringsAsFactors = TRUE,
     colClasses = c(RowNum = "integer"),
     rowsPerRead = 50)
```

```
# Compute and draw a histogram directly from the SAS file
rxHistogram( ~cost|type, data = sasDS)

# Compute summary statistics
rxSummary(~., data = sasDS)

# Estimate a linear model
linModObj <- rxLinMod(cost~age + car_age + type, data = sasDS)
summary(linModObj)

# Import a subset into a data frame for further inspection
subData <- rxImport(inData = sasDS, rowSelection = cost > 400,
varsToKeep = c("cost", "age", "type"))
subData
```

New methods for .xdf file data sources

RxXdfData objects are light-weight, in-memory objects that represent a possibly very large .xdf file on disk; objects of this type are returned from data manipulation functions that work on .xdf files. New *data.frame*-like methods are now provided for these RxXdfData objects: head, tail, names, dim, colnames, length, str, and formula. For example:

```
xdfFileName <- file.path(rxGetOption("sampleDataDir"),
     "claims.xdf")
xdfDS <- RxXdfData( xdfFileName )
head(xdfDS)
names(xdfDS)
```

New 'big data' functions for summarizing and plotting

- Compute approximate quantiles quickly by summing binned data rather than sorting. (See the `rxQuantile` help file for more information and examples.)
- Compute and plot an ROC (Receiver Operating Characteristic) curve using actual and predicted values from binary classifier system such as a logistic regression. (See the `rxRoc` help file for more information and examples.)
- Compute and plot a Lorenz curve and compute the Gini coefficient. (See the `rxLorenz` help file for more information and examples.)

R Engine Update to R 2.14.2 and Accompanying Packages

- Base and recommended packages are now byte-code compiled.
- New package parallel included in base packages
- New package doParallel included with Revolution R Enterprise (foreach parallel backend using the new parallel package)
- Enhanced foreach parallel backends doNWS, doSNOW, doSMP, and doParallel to support use with rxExec

R Productivity Environment (Windows only)

- The currently active compute context is now displayed on the status bar.

- The q() function can now be used to close the RPE.

Changes from Previous Versions:

- Distributed computing compute contexts and job objects have changed; compute contexts and jobs created with Revolution R Enterprise Version 5.0.x are not compatible with Revolution R Enterprise 6.0. If you have older jobs on your HPC Server cluster, extract all data from them and delete them before installing Revolution R Enterprise 6.0.
- The rxLogit function now returns p-values based on the Z-statistic rather than on the t-statistic for consistency with rxGlm.

Known Issues:

- ***Known Issues in Revolution R Enterprise 6.0***