R Language Modules

Updated: May 3, 2015

Support for the R language in Azure Machine Learning makes it easier than ever to publish R models in production and to use the experience of the R language community to solve real-world problems.

Requirements when Using R

Before using R script in Azure Machine Learning Studio, be sure to understand the following requirements:

- If you imported data that uses CSV or other formats, you must convert the data to a dataset before using the data in an R module.
- When you attach a dataset as input to an R module, the dataset is automatically loaded into the R workspace as a data frame with the variable name, **dataset**.

  You can define additional data frames or change the name of the workspace variable within your R script.
- The R modules run in a sandbox within your private workspace. Within your workspace, you can create data frames and variables for use by multiple modules.

  However, you cannot load R data frames from a different workspace or read variables created in a different workspace, even if that workspace is open in an Azure session.
- The implementation of R in the Azure Machine Learning Studio and workspace environment includes two principal components: one that coordinates running the script, and one that provides high-speed data access and scoring.
- Scoring has been optimized to enhance scalability and performance. The R implementation in Azure Machine Learning Studio supports two types of scoring tasks, each optimized for different requirements: experimentation and scoring on a file-by-file basis, and request response scoring via a web service.

Which R Packages Are Supported ?

Azure Machine Learning Studio includes over 400 of the most popular R packages. However, the packages that are supported in the experiment environment can change, so if you have any doubts about whether an R package is supported, use the following code to get the complete list of packages in the current environment:

```
data.set = rbind(dataset, dataset)
print(rownames(installed.packages()))
```

The list of installed R packages is printed to the output log. To view the output log, run the experiment, select the **Execute R Script** module, and click the **View output log** link near the bottom of the module parameter pane.

Tip

To help R users understand the packages that are available, the R documentation site also provides a categorized list of packages, which you can search by keywords: http://www.rdocumentation.org/ (http://www.rdocumentation.org/)

Additional R Resources

For additional R code samples and help with R and its applications, see these resources:

- R Project (http://www.r-project.org/): The official site for the R language
- Rseek (http://www.rseek.org/): A search engine for R resources
- R-bloggers (http://www.r-bloggers.com/): An aggregation of blogs in the R community
- CRAN (http://cran.r-project.org/web/views/): The largest repository of R packages
- Quick-R (http://www.statmethods.net/): A good R tutorial
- Webinar: Learn How to Get Faster End Results from Your R Models (http://channel9.msdn.com/blogs/Cloud-and-Enterprise-Premium/Learn-How-to-Get-Faster-End-Results-from-Your-R-Models)
- Bioconductor (http://bioconductor.org/): Large repository of R packages in bioinformatics
- Quick Start Guide for R (http://go.microsoft.com/fwlink/?LinkId=524954): Provides a detailed walkthrough of a time series forecasting example and tips about working with R in Azure Machine Learning Studio.

Extending Experiments Using the R Language

There are many ways that you can extend your experiment by using custom R script or by adding R packages, including:

- Adding packages to perform custom math operations. For example, there are R packages to solve differential equations, generate random numbers, or run Monte Carlo simulations.

- Defining custom transformations for data, or adding packages to read new data. For example, there are R packages to flatten hierarchical data into a flat data table, import data from Excel or export it to HTML, or perform interpolation on time series data.

- Specifying optional arguments for the underlying R functions.

- Creating custom metrics for evaluation. For example, you can use an R package to evaluate models with functions used by other statistical packages.

Splitting Columns by using R

Sometimes the data requires extensive manipulation to extract features. For example, you might get a text file that contains an ID followed by values and notes, all separated by spaces or by characters that are unsupported by Studio.

There are several R packages that provide specialized functions for this task. For example, the splitstackshape library (http://cran.r-project.org/web/packages/splitstackshape/index.html) package contains several useful functions for splitting multiple columns, even if each column has a different delimiter.

The following sample illustrates how to install the needed packages and split apart columns.

```
#install dependent packages
install.packages("src/concat.split.multiple/data.table_1.9.2.zip",
lib=".", repos = NULL, verbose = TRUE)
(success.data.table <- library("data.table", lib.loc = ".",
logical.return = TRUE, verbose = TRUE))

install.packages("src/concat.split.multiple/plyr_1.8.1.zip",
lib=".", repos = NULL, verbose = TRUE)
(success.plyr <- library("plyr", lib.loc = ".", logical.return =
TRUE, verbose = TRUE))

install.packages("src/concat.split.multiple/Rcpp_0.11.2.zip",
lib=".", repos = NULL, verbose = TRUE)
(success.Rcpp <- library("Rcpp", lib.loc = ".", logical.return =
TRUE, verbose = TRUE))

install.packages("src/concat.split.multiple/reshape2_1.4.zip",
lib=".", repos = NULL, verbose = TRUE)
(success.reshape2 <- library("reshape2", lib.loc = ".",
logical.return = TRUE, verbose = TRUE))

#install actual packages
install.packages("src/concat.split.multiple/splitstackshape_1.2.0.zi
 lib=".", repos = NULL, verbose = TRUE)
(success.splitstackshape <- library("splitstackshape", lib.loc =
".", logical.return = TRUE, verbose = TRUE))

#Load installed library
library(splitstackshape)

#Use library method to split & concat
data <- concat.split.multiple(maml.mapInputPort(1),
c("TermsAcceptedUserClientIPAddress", "EmailAddress"), c(".", "@"))

#Print column names to console
colnames(data)

#Redirect data to output port
maml.mapOutputPort("data")
```

List of Modules

The Modules References.R Language Modules category includes the following modules:

Module	Description

Execute R Script (https://msdn.microsoft.com/en-us/library/azure/dn905952.aspx)	Executes an R script from an Azure Machine Learning experiment
Create R Model (https://msdn.microsoft.com/en-us/library/azure/dn955435.aspx)	Creates an R model using custom resources

See Also

Python Language Modules (https://msdn.microsoft.com/en-us/library/azure/dn927167.aspx)
Machine Learning Module Descriptions (https://msdn.microsoft.com/en-us/library/azure/dn906013.aspx)