Execute R Script

Updated: July 11, 2015

Executes an R script from an Azure Machine Learning experiment

Category: R Language Modules (https://msdn.microsoft.com/en-us/library/azure/dn905920.aspx)

Module Overview

You can use the **Execute R Script** module to embed R code into experiments in Azure Machine Learning and execute them using R 3.1.0.

By adding R code to this module, you can perform a variety of customized tasks that are not available in Studio, including:

- Create custom data transformations
- Make your own metrics for evaluating predictions
- Build models using algorithms that are not implemented as standalone modules in Studio

The **Execute R Script** module contains sample R code that can be used as a template for developing new code.

> **Note**
>
> R code that runs in other tools, such as R Studio, might need small changes to run it in Studio. For example, input data that you provide in CSV format must be explicitly converted to a dataset before you can use it in your code.
>
> The data and column types used in the R language are not exactly the same as the data and column types used in Azure Machine Learning. For details, see the Technical Notes section.

R Packages Available in Azure Machine Learning Studio

The R environment in Azure Machine Learning has over 400 R packages installed. To get a list of installed packages as a dataset, run the **Execute R Script** module with the following R code:

```
data.set <- data.frame(installed.packages())
maml.mapOutputPort("data.set")
```

How to Use Execute R Script

To configure the **Execute R Script** module, you provide a set of optional inputs and the R code that is to be run in the workspace.

Inputs

The **Execute R Script** module supports the following inputs:

- **Dataset1**: Attach an optional input dataset.

  The input dataset should be in CSV, TSV, ARFF, or the internal Dataset format.

- **Dataset2**: Attach a second optional input dataset.

  The input dataset should be in CSV, TSV, ARFF, or the internal Dataset format.

- **Script Bundle**: Specify an optional ZIP file that can include additional R code, R objects, R packages, and datasets.

Before you can configure the **Script Bundle** input, the ZIP file must be present in the Studio workspace. To upload a ZIP file to your workspace, click **New**, click **Dataset**, and then select **From local file** and the **Zip file** option.

After you have uploaded the zipped package to Studio, verify that the zipped file is available in the **Saved Datasets** list, and then connect the dataset to the **Script Bundle** input port. If your zipped file contains an R package, you need to install the R package as part of the custom code in the **Execute R Script** module.

Any file that was contained in your ZIP file will be available for use during run time. If there was a directory structure present, it will be preserved. The only change is that you must prepend a directory called **src** to the path.

Note

Typically, R packages are provided as downloadable ZIP files. If you have already downloaded and extracted the R package that you are using in your code, you will need to re-zip the package or include the original ZIP file before you can upload the R package to Studio.

Outputs

This module returns the following outputs:

- **Results Dataset**: Provides a dataset that contains the results of any computations performed by the embedded R code.

  The dataset uses the internal Data Table (https://msdn.microsoft.com/en-us/library/azure/dn905851.aspx) format.

Warning
On this output, you can return only objects where `is.data.frame` is true. To return other R objects, try serializing the object into a byte array, or use a function that returns the desired data as a **data.frame**.

- **R Device**: Supports console output and display of PNG graphics using the R interpreter.

Parameters

The **Execute R Script** module supports customization by using embedded R script.

R Script
: You type R script into the **R Script** text box.

  The **R Script** text box is prepopulated with the following sample code, which you can edit or replace.

```
# Map 1-based optional input ports to variables
dataset1 <- maml.mapInputPort(1) # class: data.frame
dataset2 <- maml.mapInputPort(2) # class: data.frame

# Contents of optional Zip port are in ./src/
# source("src/yourfile.R")
# load("src/yourData.rdata");

# Sample operation
data.set = rbind(dataset1, dataset2)

# You'll see this output in the R Device port.
# It'll have your stdout, stderr and PNG graphics device(s).
plot(data.set)

# Select data.frame to be sent to the output Dataset port
maml.mapOutputPort("data.set")
```

Random Seed

Define a random seed value for use inside the R environment. This parameter is equivalent to calling `set.seed(value)` in R code.

Examples

You can see examples of how this module is used by exploring these sample experiments in the Model Gallery (http://gallery.azureml.net/):

- The Student Performance sample (http://go.microsoft.com/fwlink/?LinkId=525727) uses custom R script to combine the results of evaluations for multiple models into a single dataset. This sample also uses R code in the **Execute R Script** module to compute 16 time-dependent columns.

- The Breast Cancer sample (http://go.microsoft.com/fwlink/?LinkId=525726) uses custom code in the **Execute R Script** module to replicate positive examples and to combine metrics.

- The Time Series Forecasting (http://go.microsoft.com/fwlink/?LinkId=525273) sample uses **Execute R Script** to generate custom metrics, and then combines them into a single table by using the Add Rows (https://msdn.microsoft.com/en-us/library/azure/dn905871.aspx) module.

There are many ways that you can extend your experiment by using custom R script.

To get started using R in the **Execute R Script** module, see this video: Using R in Azure Machine Learning Studio (http://channel9.msdn.com/Blogs/Windows-Azure/R-in-Azure-ML-Studio)

This following sections provide simple walkthroughs that demonstrate common scenarios for use of custom R code, including reading files, manipulating data, writing graphics files, and using custom learners.

Reading from Input and Writing to Output

The following script takes the input table and appends a copy of the table to itself, effectively doubling the size of the table. The result is then sent to the output port.

```
# Map existing dataset to first input port
dataset1 <- maml.mapInputPort(1) # class: data.frame
# Concatenate dataset1 to dataset 1
newdataset = rbind(dataset1, dataset1)
# Send the combined dataset to the output port
maml.mapOutputPort("newdataset");
```

Reading a ZIP File as Input

This scenario assumes that you have uploaded a data file in CSV format, named "mydatafile.csv", as part of your ZIP file. Next, you connected the resulting dataset to the **ScriptBundle** input of your **Execute R Script** module. Using the following code, you can read a CSV data from the ZIP file as a dataset as part of processing in the **Execute R Script** module.

```
# Load data from attached ZIP file, optionally specifying the
encoding
mydataset=read.csv("src/newdata.csv",encoding="UTF-8");
nrow(mydataset);
ncol(mydataset);
# Map new dataset to the first output port
maml.mapOutputPort("mydataset");
```

Warning

If you pass an existing dataset to **Execute R Script**, the underlying Data Table format will be passed to your R function in the `data.frame` format. Any other data that is compatible with the Data Table format (CSV files, ARFF files, etc.) will be automatically converted to the `data.frame` format.

Replicate Rows

In R, replicating rows is easy. This example demonstrates how you can replicate the positive samples in a dataset by a factor of 20 to balance the sample.

```
dataset <- maml.mapInputPort(1)

data.set <- dataset[dataset[,1]==-1,]
pos <- dataset[dataset[,1]==1,]
for (i in 1:20) data.set <- rbind(data.set,pos)
row.names(data.set) <- NULL

maml.mapOutputPort("data.set")
```

Call a Custom Learner – R Arules Package

This example calls a learner that implements the *a priori* association rules algorithm provided by the R language package **Arules**. The learner is then used in a market basket analysis task.

```
library("arules")
library("arulesViz")
dataset <- read.transactions(file="src/SalesReport.csv",
rm.duplicates= TRUE, format="single",sep=",",cols =c(1,2))
#dataset <- sapply(dataset,as.factor)
basket <- apriori(dataset,parameter = list(sup = 0.5, conf =
0.9,target="rules"));
inspect(basket)
plot(basket)  # if this is not NULL i.e. if there are rules
```

Call a Custom Learner – Naïve Bayes

The following code illustrates how you can call an R library to build a model by using a machine learning algorithm that is not included in Studio. This example implements a Naïve Bayes learner provided by the R **e1071** library.

```
library(e1071)
features <- get.feature.columns(dataset)
labels   <- get.label.column(dataset)
train.data <- data.frame(features, labels)
feature.names <- get.feature.column.names(dataset)
names(train.data) <- c(feature.names, "Class")
model <- naiveBayes(Class ~ ., train.data)
```

Create a Custom Scorer

The following code demonstrates how you can call a scorer from an existing R library by using custom code in the **Execute R Script** module.

```
library(e1071)
    features <- get.feature.columns(dataset)
    scores <- predict(model, features)
```

Write a Graphics File

Suppose you are running an experiment that uses **Execute R Script** and your script generates a chart as a PDF file. Although Studio supports the display of PNG files using the **R Device** output port, you might want to generate the results as a PDF file in a blob in Azure Storage to use for reporting.

To do this, you can create the basic PDF file, and then use your **Execute R Script** module to return the Base64-encoded string of the PDF file. You can then pass this output to a Writer (https://msdn.microsoft.com/en-us/library/azure/dn905984.aspx) module, which can send the value to a blob.

The following code illustrates this approach:

```
d <- maml.mapInputPort(1)
d$dteday <- as.numeric(d$dteday)
pdf()
plot(d)
dev.off()
library(caTools)
b64ePDF <- function(filename) {
                maxFileSizeInBytes <- 5 * 1024 * 1024 # 5 MB
                return(base64encode(readBin(filename, "raw", n =
maxFileSizeInBytes)))
}

d2 <- data.frame(pdf = b64ePDF("Rplots.pdf"))

maml.mapOutputPort("d2");
```

Technical Notes

Optimizing R Performance in Azure Machine Learning Studio

The current default memory is 14 GB. You might encounter an out-of-memory error message if you attempt to manipulate very large data frames by using the **Execute R Script** module.

To increase the amount of memory that is used by R script, you can use a line similar to this at the beginning of the script:

```
memory.limit(56000)
```

User-specified R code is run by a 64-bit R interpreter (version 3.1.0) that runs in Azure using an A8 virtual machine with 56 GB of RAM. To increases the speed of your R code, you can use the just-in-time compiler provided in the preinstalled **Compiler** package.

Converting Column Data Types Between R and Studio

The following table shows how the data types in R correspond to the data types in Azure Machine Learning:

R type	Studio type
Integer	Integer
Double	Double
Complex	Complex This type is supported by only a subset of modules.
Logical	Boolean
Character	String
Raw	Not supported
Difftime	TimeSpan
factor	categorical
data.frame	dataset

Warning

Columns of data type **lists** in R cannot be converted because the elements in such columns potentially are of different types and sizes. For example, the following valid R code will fail if used in the **Execute R Script** module:

```
data.set <- data.frame(r=I(list(list(1,2,3),list(4,5))))
maml.mapOutputPort("data.set")
```

Converting DateTime Values

Azure Machine Learning Studio uses different datetime types than does R. If the data that you are analyzing contains date or time data, you should be aware of the following conversion requirements when porting existing R code into Studio:

To convert from Azure Machine Learning Studio to R:

DateTime columns are converted to POSIXct vectors. However, each individual element of the resulting vector is a number of seconds since 1970-01-01T00:00:00. No time zone adjustments are made in this conversion.

To convert from R to Azure Machine Learning Studio:

POSIXct vectors are converted to DateTime columns in the UTC time zone.

For example, 2011-03-27 01:30:00 PDT will be converted to 2011-03-27T08:30:00Z, where the Z indicates that the time is in UTC.

> **Tip**
>
> When using times inside the **Execute R Script** module, you must specify time stamps explicitly. The R interpreter hosted in the **Execute R Script** module does not have access to local time zone definitions.

Passing R Objects Between Execute R Script Modules

You can pass R objects between instances of the **Execute R Script** module by using the internal serialization mechanism. For example, the following R code demonstrates how to move the R object named **A** between two **Execute R Script** modules.

In the first **Execute R Script** module, add the following code to create a serialized object `A` as a column in the module's output Data Table:

```
serialized <- as.integer(serialize(A,NULL))
data.set <- data.frame(serialized,stringsAsFactors=FALSE)
maml.mapOutputPort("data.set")
```

The explicit conversion to integer type is required because the serialization function outputs data in the R **Raw** format, which is not supported by Azure Machine Learning.

Use the following code in the second **Execute R Script** module to extract object `A` from the input Data Table:

```
dataset <- maml.mapInputPort(1)
A <- unserialize(as.raw(dataset$serialized))
```

Networking

For security reasons, all networking from or to R code in **Execute R Script** modules is blocked by Azure. Also, with very few exceptions, access to local ports from the **Execute R Script** is blocked.

Parallel Execution

Currently parallel execution with multiple threads is not supported.

Expected Inputs

Name	Type	Description
Dataset1	Data Table (https://msdn.microsoft.com/en-us/library/azure/dn905851.aspx)	Input dataset 1
Dataset2	Data Table (https://msdn.microsoft.com/en-us/library/azure/dn905851.aspx)	Input dataset 2
Script Bundle	Zip	Set of R sources

Module Parameters

Name	Range	Type	Default	Description
R Script	Any	StreamReader		Specify a **StreamReader** that points to the R script sources.
Random Seed	>=0	Integer		Define a random seed value for use inside the R environment. Equivalent to `\"set.seed(value)\"`. This parameter is optional.

Outputs

Name	Type	Description
Result Dataset	Data Table (https://msdn.microsoft.com/en-us/library/azure/dn905851.aspx)	Output dataset
R Device	Data Table (https://msdn.microsoft.com/en-us/library/azure/dn905851.aspx)	Console output and PNG graphics device from the R interpreter

See Also

R Language Modules (https://msdn.microsoft.com/en-us/library/azure/dn905920.aspx)
Create R Model (https://msdn.microsoft.com/en-us/library/azure/dn955435.aspx)
Machine Learning Module Descriptions (https://msdn.microsoft.com/en-us/library/azure/dn906013.aspx)
Python Language Modules (https://msdn.microsoft.com/en-us/library/azure/dn927167.aspx)
A-Z List of Machine Learning Studio Modules (https://msdn.microsoft.com/en-us/library/azure/dn906033.aspx)

© 2015 Microsoft