

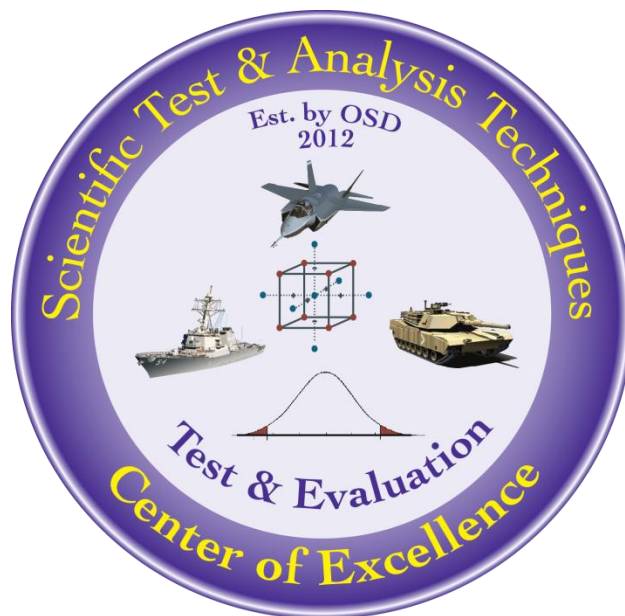
# MATLAB and R Integration with JMP

---

*Authored by: Francisco Ortiz, PhD*

*28 August 2017*

*Revised 11 October 2018*



**The goal of the STAT COE is to assist in developing rigorous, defensible test strategies to more effectively quantify and characterize system performance and provide information that reduces risk. This and other COE products are available at [www.afit.edu/STAT](http://www.afit.edu/STAT).**

## Table of Contents

Introduction .....	2
Background .....	2
Method .....	2
Sample Data Set .....	3
How to Create a Script in JMP.....	3
How to Open the Log Window in JMP .....	5
MATLAB and JMP Integration Setup.....	6
List of MATLAB Commands.....	9
JMP Script to Build Regression Tree in MATLAB.....	9
R and JMP Integration Setup .....	11
List of R Commands.....	12
JMP Script to Build a Regression Tree in R.....	13
Comparison of Different Models .....	14
Conclusion.....	16
References .....	16

*Revision 1, 11 Oct 2018: Minor formatting; updated image showing JMP menu; renamed associated .jmp file to match this best practice.*

## Introduction

A STAT practitioner will have to utilize some statistical and mathematics software to plan, design and analyze tests. One commonly used statistical software package in the T&E community is JMP. While JMP is a versatile tool capable of performing the majority of STAT tasks, there will be situations that call for utilizing other software programs (e.g., MATLAB and R). This paper will describe one such situation, which required analysis to be performed using JMP, MATLAB and R. This tutorial will demonstrate how to get MATLAB and R integrated with JMP.

Keywords: MATLAB, R, JMP, regression trees

## Background

The following is an example of a situation the STAT COE encountered in which we had to utilize both MATLAB, R and JMP for our analysis. An infrared counter measure system (IRCM) lab has created a MATLAB based digital simulation of threat engagements. To run the simulation, they still need Missile Warning System (MWS) timelines inputs (e.g., time to go, time of handoff, etc.). These data come from a statistical model (a regression tree) that predicts timeline performance based on various engagement factors. While JMP can help build this statistical model, it cannot create an output function that the MATLAB simulation can easily query. In this case, we will utilize MATLAB's Statistical toolbox to build and save a regression tree model and thus allowing the simulation to query the timeline prediction model to get its inputs. We still want to evaluate the regression tree model and build surface plots of the model using JMP.

## Method

In the following sections, we will:

- Introduce the data set used to build the prediction models.
- Show how to setup and write scripts in JMP.
- Provide step-by-step instructions on how to setup JMP to call MATLAB and R.
- Present the JMP scripts that creates the regression tree model in MATLAB and R.
- Perform a model comparison between all models in JMP.

For more information on regression trees and how to build them, see the "Using Regression Trees" best practice available on the STAT COE website. All data presented in this paper is notional and are for demonstration purposes.

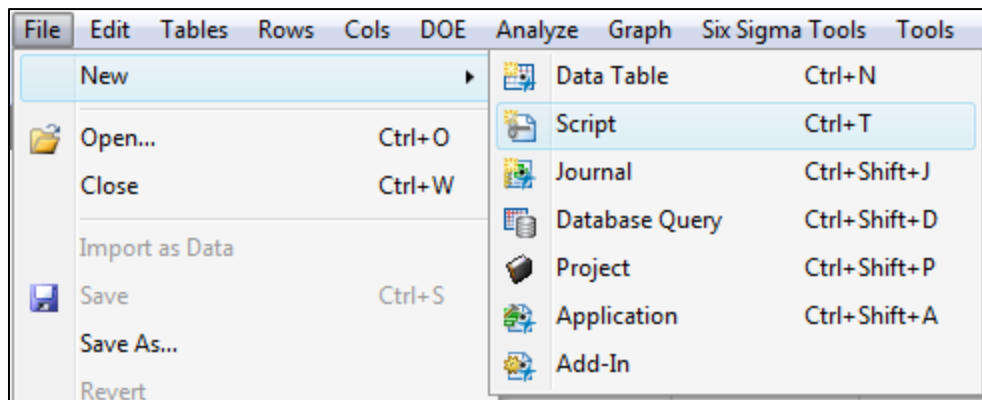
## Sample Data Set

Download and open “Matlab-and-R-Integration-with-JMP-DATA.jmp” file from the STAT COE website. The file contains 50000+ rows of data. There are six columns labeled X1 through X6 that represent the engagement factors. Note that the variables are coded and range from -1 to 1. The column labeled Y denotes the timeline response that needs to be modeled and provided as input to the IRCM simulation. The goal is to build and save regression tree models using MATLAB and R and return the results to JMP. In JMP, we will compare each model and determine which one provides the best prediction.

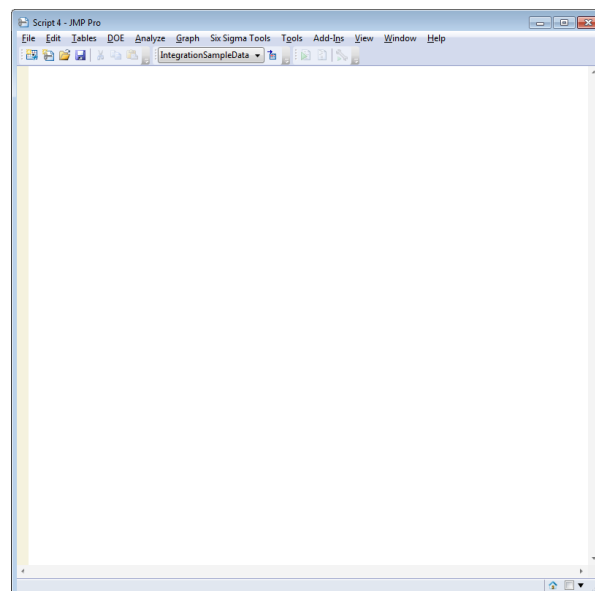
## How to Create a Script in JMP

In this section, we show you how to create a script and use the Log window in JMP. There are two ways to create a JMP script.

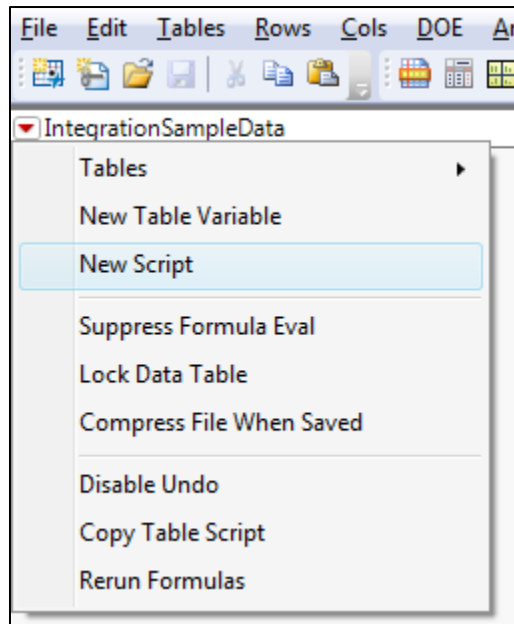
**Method 1:** Go to File>New>Script.



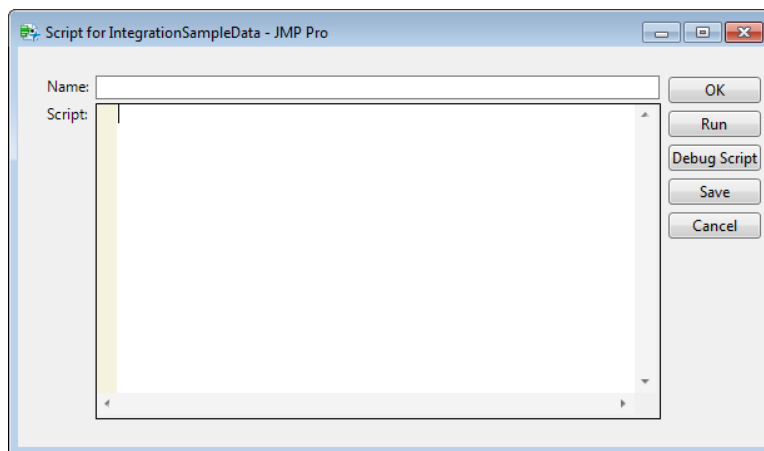
The following window should appear.



**Method 2:** A script could also be created within the actual data table by clicking the red drop-down arrow by the filename in the upper left-hand box of the data table.

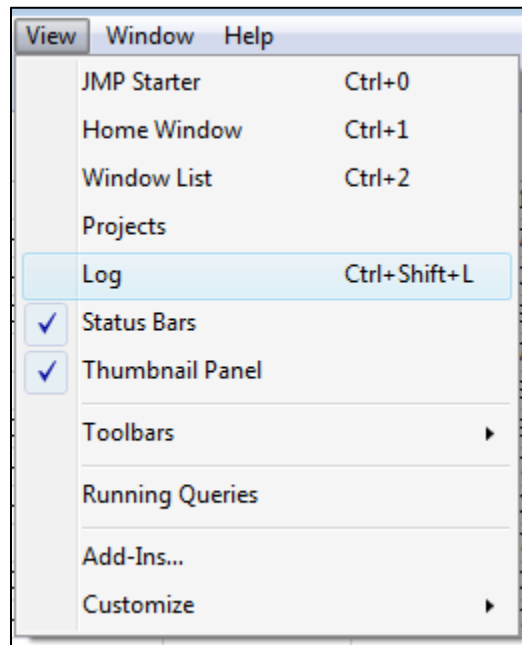


This will open up a script box that you can name and start inputting code.

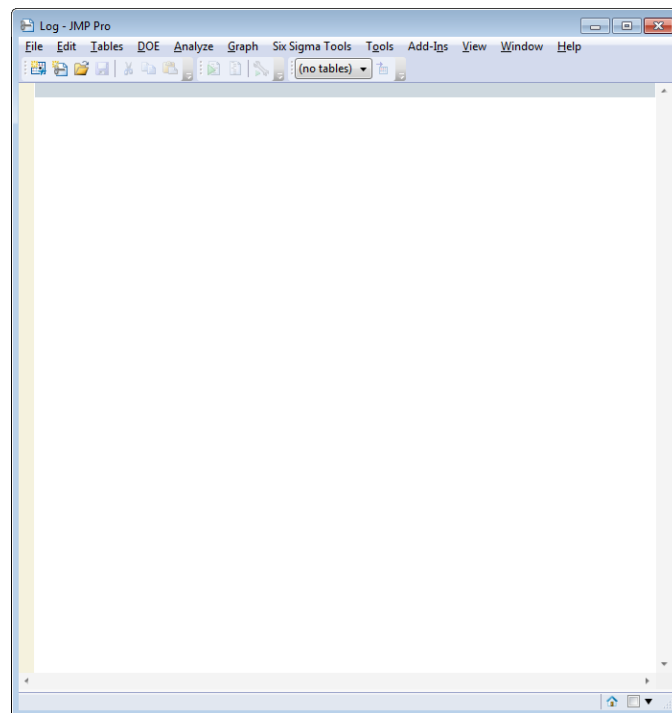


## How to Open the Log Window in JMP

Open the JMP log by going into the View menu and selecting Log. You can also use the shortcut Ctrl+Shift+L.



The following screen should appear.



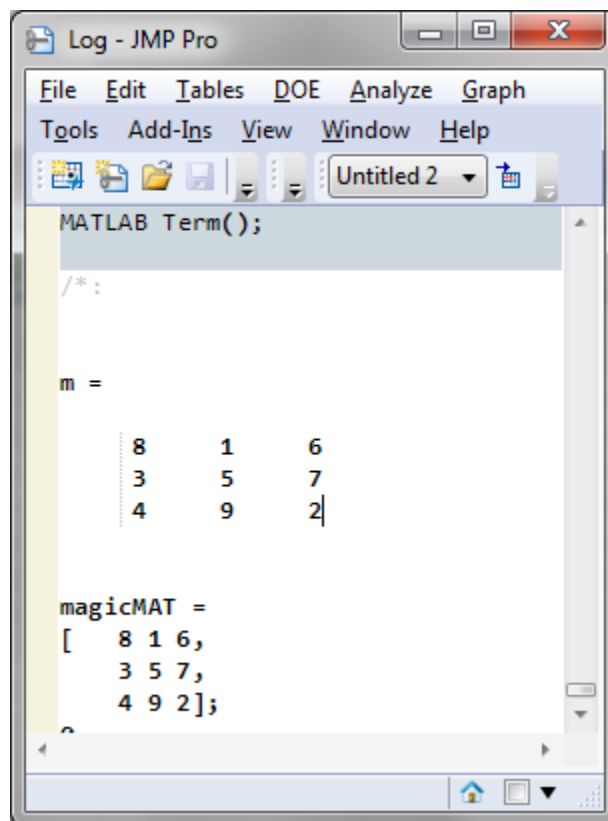
## MATLAB and JMP Integration Setup

MATLAB and its Statistics toolbox must be installed on the same computer as JMP. To check that JMP is properly integrated with MATLAB run the following script code.

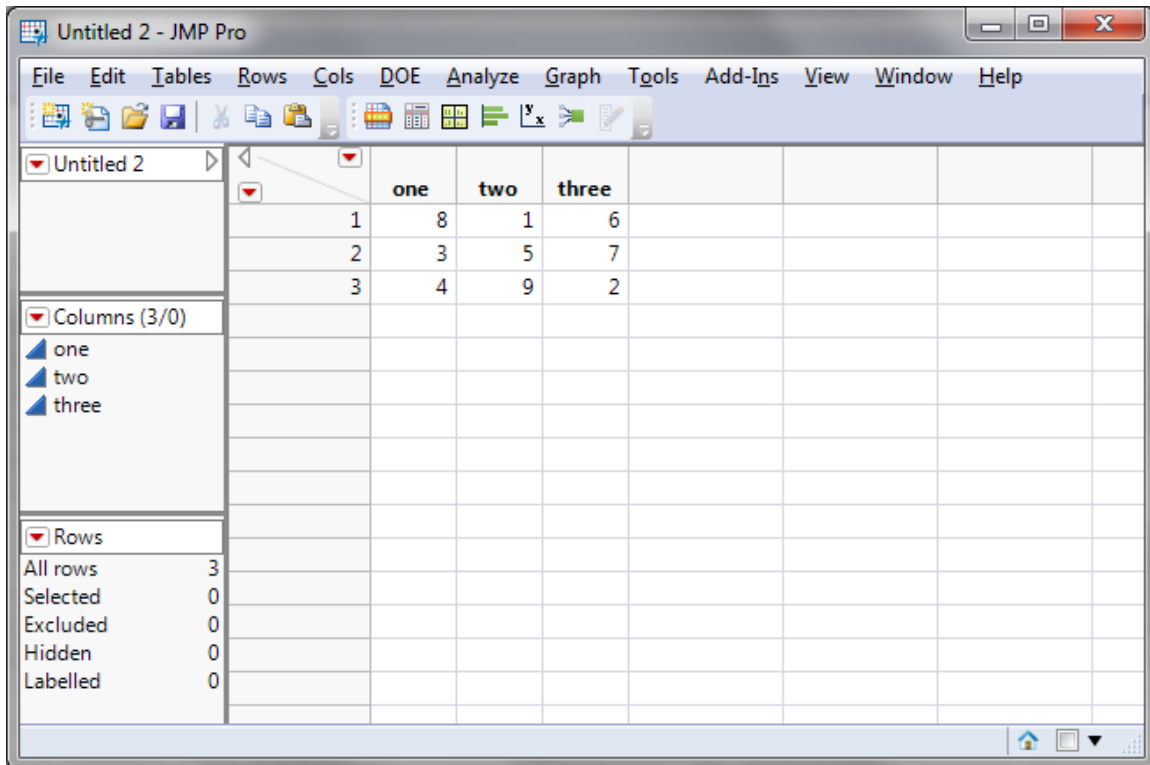
### Simple Code to Check Installation

```
MATLAB Init();  
//Makes MATLAB command window visible  
MATLAB Control(Visible(1));  
MATLAB Submit ("m=magic(3)");  
magicMAT=MATLAB Get(m);  
Show (magicMAT);  
dt=As Table(magicMAT, <<Column Names ({one,two,three}));  
MATLAB Term();
```

You should see the following output in the log window:



And the follow JMP table should appear:

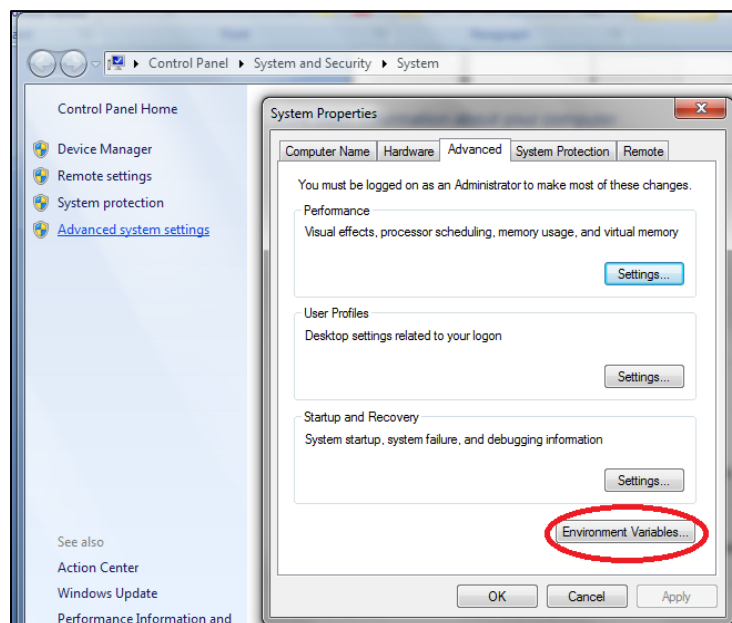


The screenshot shows the JMP Pro software window titled "Untitled 2 - JMP Pro". The interface includes a menu bar (File, Edit, Tables, Rows, Cols, DOE, Analyze, Graph, Tools, Add-Ins, View, Window, Help) and a toolbar. On the left, there is a sidebar with "Columns (3/0)" and "Rows". The main area displays a table with the following data:

	one	two	three
1	8	1	6
2	3	5	7
3	4	9	2

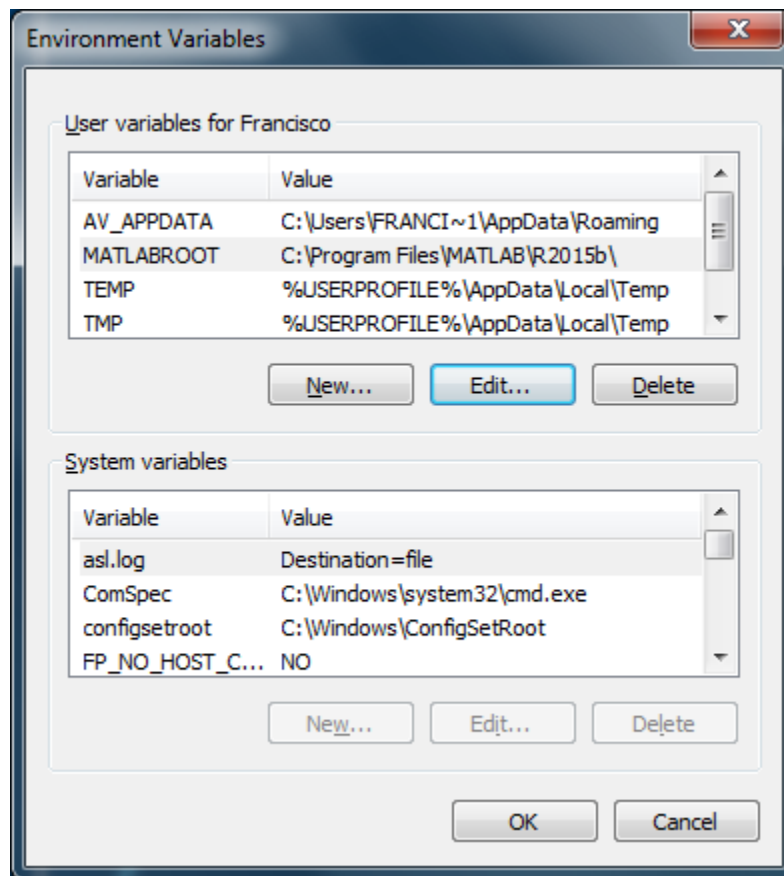
The "Columns (3/0)" section lists "one", "two", and "three". The "Rows" section shows "All rows" as 3, "Selected" as 0, "Excluded" as 0, "Hidden" as 0, and "Labelled" as 0.

If you got an error message instead you may need to create an environmental variable to point JMP to the MATLAB installation. To do so, go to **Control Panel>System and Security>System** in windows, select Advance system settings and then click on the Environmental Variables button.

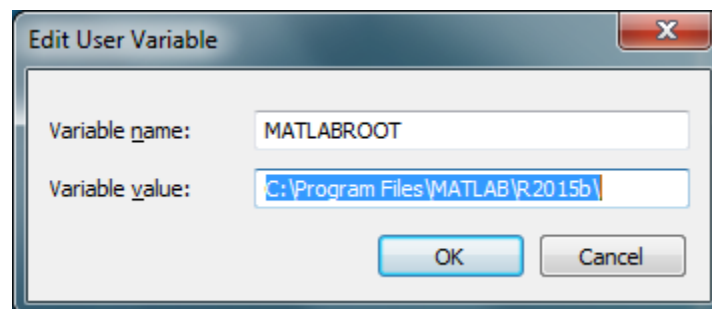




Select “New...” on the next dialog box.



For variable name type in “MATLABROOT” and for the Variable value enter the path name where MATLAB resides



If you need further help to get MATLAB and JMP working together, please see the following link and white papers on how to setup JMP to find your MATLAB installation.

- [http://www.jmp.com/support/help/Working\\_with\\_MATLAB.shtml](http://www.jmp.com/support/help/Working_with_MATLAB.shtml)
- [https://www.jmp.com/blind/whitepapers/wp\\_jmp\\_matlab\\_106667.pdf](https://www.jmp.com/blind/whitepapers/wp_jmp_matlab_106667.pdf)

## List of MATLAB Commands

The following table lists the available MATLAB integration function.

JSL	Function
MATLAB Connect ( );	Returns a MATLAB connection scriptable object.
MATLAB Control ( );	Changes the control options for MATLAB.
MATLAB Execute ( );	Sends a list of inputs, executes statements and returns a list of options.
MATLAB Get ( );	Returns data from MATLAB, where the name argument can represent any of the following MATLAB data types: numeric, string, matrix, list, data frame.
MATLAB Get Graphics ( );	Returns the last graphics object written to the MATLAB graph display window in a graphics format specified by the format argument.
MATLAB Init ( );	Initializes the MATLAB interface.
MATLAB Is Connected ( );	Returns 1 if there is an active MATLAB connection; otherwise, returns 0.
MATLAB JMP Name to MATLAB Name ( );	Maps a JMP variable name to a MATLAB variable name using MATLAB variables naming rules.
MATLAB Send ( );	Sends data to MATLAB, where the name argument can represent any of the following JMP data types: numeric, string, matrix, list, data table.
MATLAB Send File ( );	Sends a data file to MATLAB where the filename argument is a string specifying a pathname to the file to be sent to MATLAB.
MATLAB Submit ( );	Submit statements to MATLAB. Statements can be in the form of a string value or list of string values.
MATLAB Submit File ( );	Submit statements to MATLAB using a file specified by the path argument.
MATLAB Term ( );	Terminate the MATLAB interface.

## JMP Script to Build Regression Tree in MATLAB

The code below does the following:

- Setup a new column to store the regression tree prediction from MATLAB.
- Create two tables, one consisting of the factors, the other of the response. These tables will be sent to MATLAB for processing.
- Initiates MATLAB.
- Send both the factor and response tables to MATLAB.
- Convert the inputs into arrays for MATLAB to create the regression tree model.
- Build the model.
- Get predictions for every point in the matrix.
- Send those predictions back to JMP.

- Prompt the user to save the prediction model in the MATLAB format (.mat).
- Terminates MATLAB.

### MATLAB Code

```
//Setup column to store regression tree prediction
dt=current data table();
dt<<New Column ("MATLAB RT Prediction");
col=Column("MATLAB RT Prediction");

//Open up data and create two data tables (factors and responses)
dtfactors = Data Table( "Matlab-and-R-Integration-with-JMP-DATA" ) << Subset(All
rows,columns( :X1, :X2, :X3, :X4, :X5, :X6 ));
dtresponse= Data Table( "Matlab-and-R-Integration-with-JMP-DATA" ) << Subset(All
rows,columns( :Y ));

//Start MATLAB
MATLAB Init( );

//Makes MATLAB command window visible
MATLAB Control(Visible(0));

//Send tables to MATLAB (They are stored as structures)
MATLAB Send (dtfactors);
MATLAB Send (dtresponse);

//Close tables
Close(dtfactors);
Close(dtresponse);

//Convert structures into arrays in order to use regression tree function in MATLAB
MATLAB Submit ("F = struct2table(dtfactors)");
MATLAB Submit ("R = struct2table(dtresponse)");
MATLAB Submit ("F = table2array(F)");
MATLAB Submit ("R = table2array(R)");

//Build your Tree
MATLAB Submit("tree = fitrtree(F,R);");

//Collect Predicted values from regression tree
MATLAB Submit("Predictions = predict(tree,F)");

//Get data back to JMP
dtmatlab = MATLAB Get( Predictions );

//Move data into a data table from a matrix
col<<SetValues(dtmatlab);

//Change directory
MATLAB Submit("folder_name = uigetdir");
MATLAB Submit("cd(folder_name)");
```

```
//Save regression tree model
MATLAB Submit("save tree.mat");

//Terminate Connect
MATLAB Term( );
```

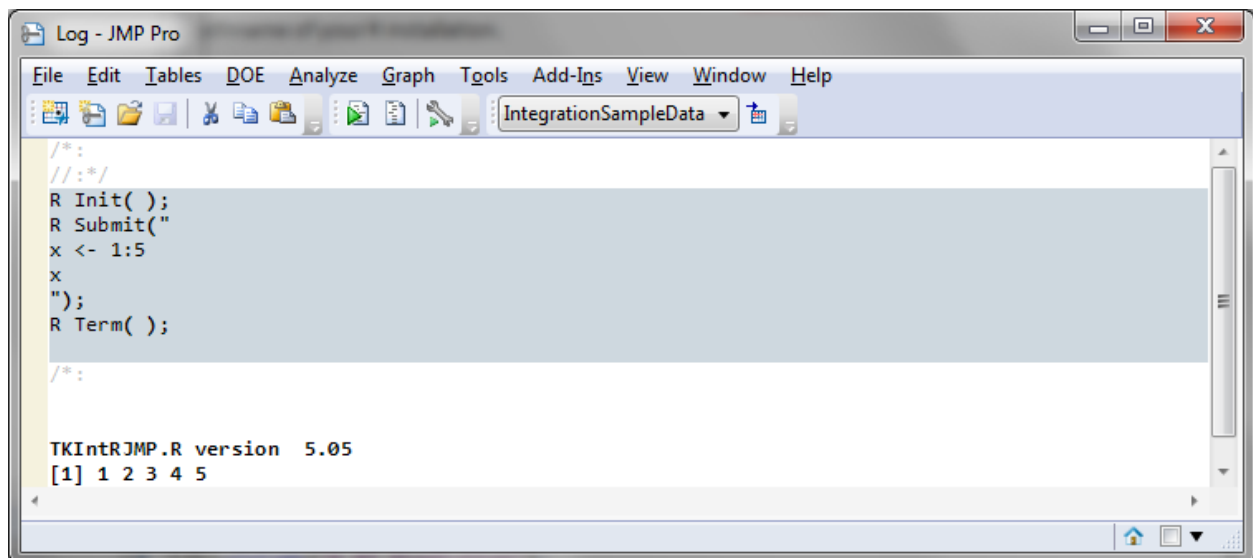
## R and JMP Integration Setup

Make sure R the package “rpart” is installed on the same computer as JMP. You can download R for free from the Comprehensive R Archive Network website: <http://cran.r-project.org> To check that JMP is integrated with R run the following script code.

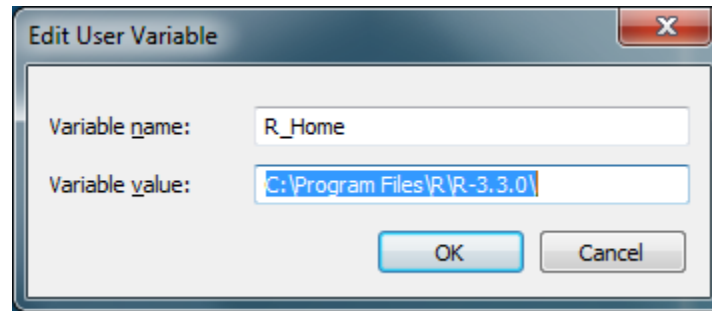
### Simple Code to Check Installation

```
R Init( );
R Submit("
x <- 1:5
x
");
R Term( );
```

You should see following in the log window:



If you got an error message instead, you might need to create an environmental variable to direct JMP to your R installation. To do so use the same procedure outlined previously for creating the environmental variable for MATLAB but this time the variable name is “R\_Home” and the variable name is the pathname of your R installation.



If you need further help to get R and JMP working together, please see the following link

[http://www.jmp.com/support/help/Working\\_with\\_R.shtml](http://www.jmp.com/support/help/Working_with_R.shtml)

[http://www.jmp.com/en\\_gb/events/ondemand/mastering-jmp/jmp-and-r-integration.html](http://www.jmp.com/en_gb/events/ondemand/mastering-jmp/jmp-and-r-integration.html)

### List of R Commands

The following table lists the available R integration function.

JSL	Description
R Connect( <named_arguments> )	Returns the current R connection object. If there is no connection to R, it initializes the R integration interfaces and returns an active R integration interface connection as a scriptable object.
R Execute( { list of inputs }, { list of outputs }, "rCode", <named_arguments> )	Submit the specified R code to the active global R connection given a list of inputs. On completion, the outputs are returned into the specified list.
R Get( variable_name )	Gets the named variable from R to JMP.
R Get Graphics( "format" )	Gets the last graphics object written to the R graph display window in the specified format.
R Init( named_arguments )	Initializes the R session.
R Is Connected()	Determines whether a connection to R exists.
R Send( name, <R Name( name )> )	Sends named variables from JMP to R.
R Send File( "pathname", <R Name("name")> )	Sends the specified data file from JMP to R.
R Submit( "rCode", <named_arguments> )	Submits the specified R code to the active global R connection.
R Submit File( "pathname" )	Submits statements to R using a file pointed in the specified pathname.
R Term()	Terminates the currently active R integration interface

## JMP Script to Build a Regression Tree in R

The code below does the following:

- Setup a new column to store the regression tree prediction from R.
- Initiates R.
- Send the data tables to R.
- Build the model using the rpart function.
- Get predictions for every point in the matrix.
- Send those predictions back to JMP.
- Prompt the user to save the prediction model in the R format (.RData).
- Terminates R.

```
//Open up data table a
dt= Data Table( "Matlab-and-R-Integration-with-JMP-DATA" );

//Creates new column to store prediction
dt<<New Column ("R RT Prediction");
col=Column("R RT Prediction");

//Start R
R Init();

// Sends the opened data table represented by dt to R
R Send(dt);

//Creates regression tree
R submit("\[

#Call regression tree library
library(rpart)

#Create fit
fit <- rpart(Y ~ X1 + X2 + X3 + X4 + X5 + X6, data=dt)

#Save predictions
yhat<-predict(fit)

#Choose your working directory
direc = choose.dir()
setwd(direc)

#Save workspace to working directory
save.image(file = "Tree.RData")
]\");

//Gets predicted values from R
YHAT=R Get(yhat);
```

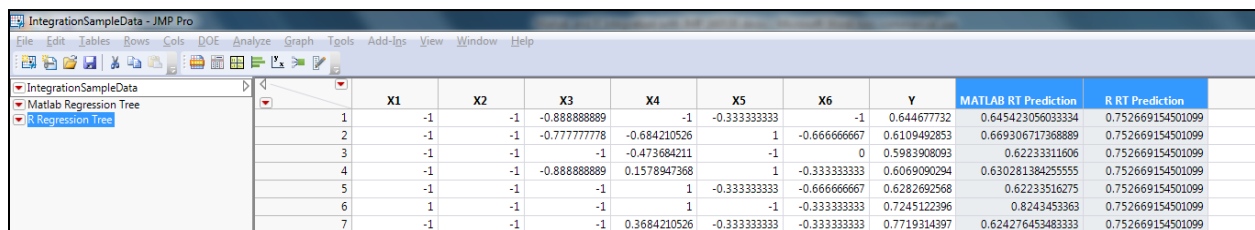
```
//Transposes data
YHAT=Transpose(YHAT);

//Move data into a data table from a matrix
col<<SetValues(YHAT);

//Terminate the connection with R
R Term();
```

## Comparison of Different Models

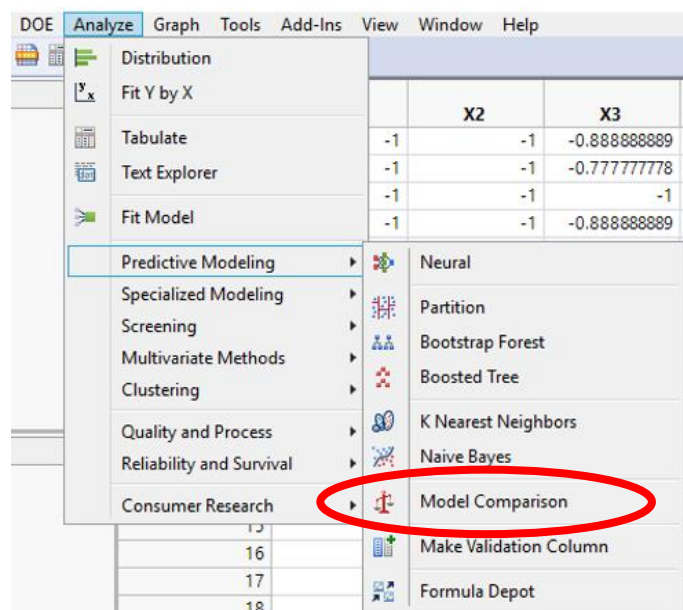
If you were successful in getting the scripts to work you should now have two new columns within the “IntegrationSampleData” table, “MATLAB RT Prediction” and “R RT Prediction”.



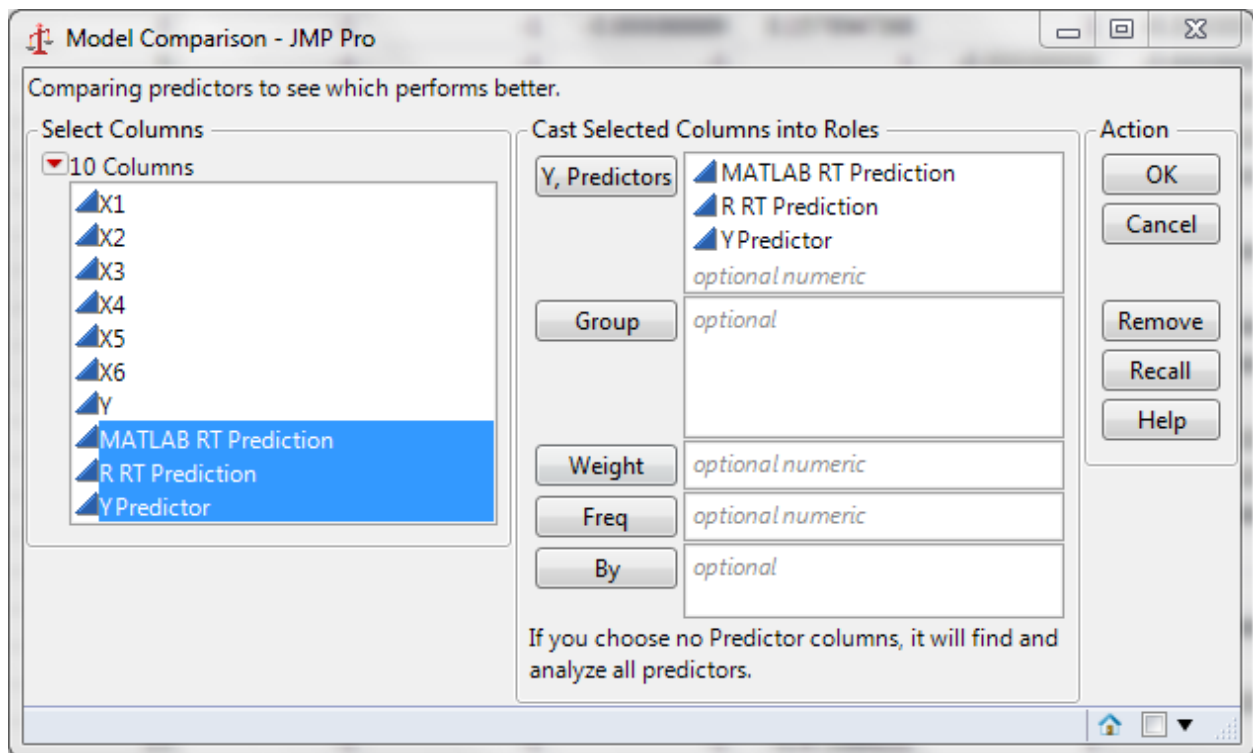
	X1	X2	X3	X4	X5	X6	Y	MATLAB RT Prediction	R RT Prediction
1	-1	-1	-0.888888889	-1	-0.333333333	-1	0.644677732	0.645423056033334	0.752669154501099
2	-1	-1	-0.777777778	-0.684210526	1	-0.666666667	0.6109492853	0.669306717368889	0.752669154501099
3	-1	-1	-0.473684211	-1	-1	0	0.5983908093	0.62233311606	0.752669154501099
4	-1	-1	-0.888888889	0.1578947368	1	-0.333333333	0.6069090294	0.630281384255555	0.752669154501099
5	-1	-1	-1	1	-0.333333333	-0.666666667	0.6282692568	0.62233516275	0.752669154501099
6	1	-1	-1	1	-1	-0.333333333	0.7245122396	0.8243453363	0.752669154501099
7	-1	-1	-1	0.3684210526	-0.333333333	-0.333333333	0.7719314397	0.624276453483333	0.752669154501099

Let’s say we want to compare these two models to the regression tree model JMP would create. Create a regression tree model in JMP using the instructions within the STAT COE best practice **Using Regression Trees**.

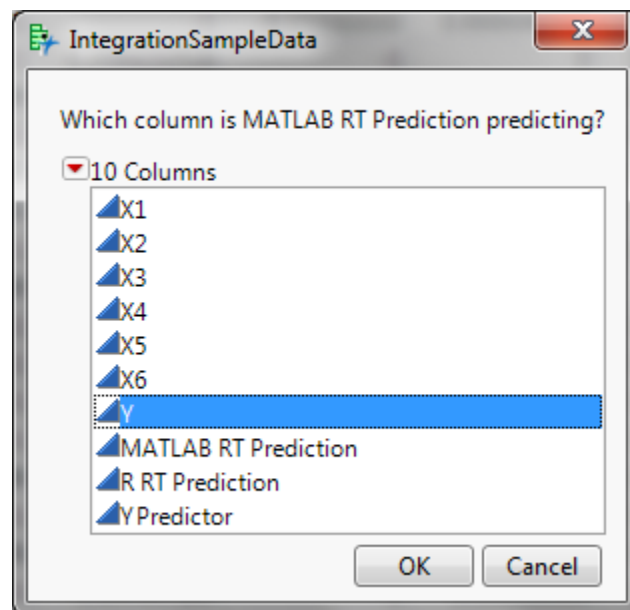
To make a direct comparison in JMP select the following from the top menu, Analyze>Modeling>Model Comparison.



Choose the regression tree models you've created.

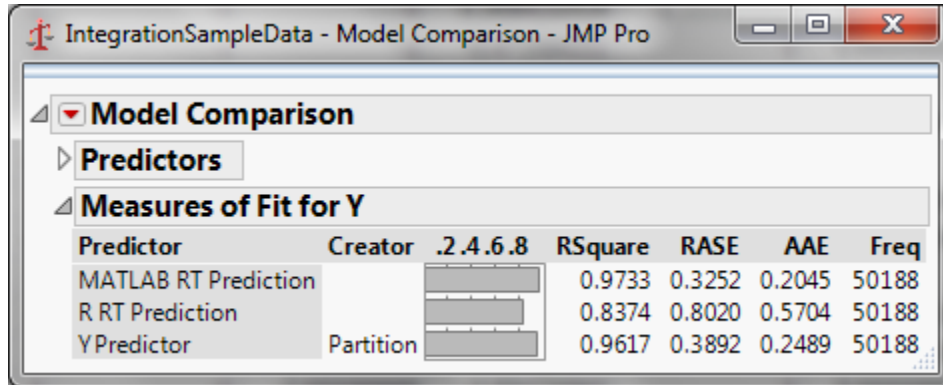


Select Y as the column these models are predicting.





The following dialog box should appear.



Predictor	Creator	.2	.4	.6	.8	RSquare	RASE	AAE	Freq
MATLAB RT Prediction						0.9733	0.3252	0.2045	50188
R RT Prediction						0.8374	0.8020	0.5704	50188
Y Predictor	Partition					0.9617	0.3892	0.2489	50188

You can see that in this case, the MATLAB model has the highest RSquare value, 0.9733, indicating that this model best fit the data. This isn't surprising since the default setting used by MATLAB to build a regression tends to grow very deep trees. The algorithm also tends to be the most computational expensive amongst these three statistical packages, something you may have noticed when you ran the MATLAB script. These results should not be interpreted to mean that MATLAB is a better statistical package for building regression trees. This just shows that the settings for the other two packages would need to be tweaked in order to produce as deep of a tree as the default settings for MATLAB.

## Conclusion

This paper provides a step-by-step tutorial on how to use MATLAB and R within the JMP environment. The tutorial is demonstrated on IRCM example based on a real life scenario encounter by the STAT COE. Integrating R and MATLAB increases the functionality of JMP. Advance statistical toolboxes and packages within MATLAB and R are now assessable without jumping between software packages and changing data formatting. Results from MATLAB and R can be imported back into JMP for easier graphing and further analysis. There may be situations in which labs will have access to MATLAB or R but not to JMP. With this approach, predictive models and results could be created in a language easier to share with the community.

## References

- SAS JMP (2016), Scripting Guide (Version 11). Retrieved from:  
[https://www.jmp.com/support/downloads/pdf/jmp\\_scripting\\_guide.pdf](https://www.jmp.com/support/downloads/pdf/jmp_scripting_guide.pdf)
- SAS JMP (2016), JMP Extensibility Synergy with MATLAB. Retrieved from:  
[http://www.jmp.com/en\\_gb/whitepapers/jmp/jmp-extensibility-matlab.html](http://www.jmp.com/en_gb/whitepapers/jmp/jmp-extensibility-matlab.html)