PhUSE 2008

Paper TS01

J(u)mp to R: Using R functions in JMP Genomics

Patrick René Warnat, HMS Analytical Software GmbH, Heidelberg, Germany
Friedrich Schuster, HMS Analytical Software GmbH, Heidelberg, Germany
Andreas Mangold, HMS Analytical Software GmbH, Heidelberg, Germany

ABSTRACT

JMP is a software product that dynamically links statistics with graphics and offers an interactive and easy-to-handle user interface. The Genomics add-on for JMP comprises methods for the analysis of data generated in life sciences.

However, many more methods for data analysis in life sciences are available as source code for the R language. To use the large variety of methods that are readily available in R together with JMP Genomics, we provide a step by step description of how to integrate R functions into JMP Genomics. We show how SAS macros are created and integrated into the JMP user interface that use an external R installation. Results are read back into JMP Genomics and are ready for further analysis.

Thus, it is possible to combine the great user interface of JMP Genomics with the wealth of R implementations of algorithms used in life sciences.

INTRODUCTION

One of the most important challenges of modern biomedical research is a better understanding of cellular processes on a molecular level. A promising use of new knowledge generated in this context, is the development of new or more effective drugs by pharmaceutical researchers. Recent technological improvements have led to more frequent use of modern high-throughput molecular laboratory analysis methods in biomedical and pharmaceutical research. These laboratory methods produce an enormous amount of data, which can only be analyzed using adequate software tools.

JMP is a software product that dynamically links statistics with graphics and offers an interactive and easy-to-handle user interface. The Genomics add-on for JMP comprises methods for analysis of data generated in life sciences. Especially for laboratory researchers who want to perform basic analyses of their data, without continual help of a bioinformatics specialist, the JMP Genomics software with its easy-to-use interface is most interesting.

However, although the selection of analysis methods included in the Genomics add-on may be convenient, many more methods for data analysis in life sciences are available as source code for the R language [1, 2]. In particular, most recent algorithms conceived by scientists working in the field of bioinformatics are often published as implementations in R. In order to make these methods available to JMP Genomics users, the integration of R functions into JMP Genomics would be desirable. We therefore provide a description here of how to call R functions from within JMP Genomics.

The remainder of this paper is structured as follows: we begin by giving an overview on how to integrate R functions into JMP Genomics. Then, we provide a step by step description of the integration procedure. Next, a concrete example for the integration of an R function is described, including source code samples. Finally, the paper contains a discussion and conclusion section, as well as sections providing information about the versions of the software used, references, recommended reading and contact information.

INTEGRATION OF R FUNCTIONS INTO JMP GENOMICS

The technological basis of JMP Genomics is the collaboration between JMP as front-end with a SAS server as back-end. The main part of the Genomics add-on for JMP is a set of SAS macros, each representing a data transformation or analysis method. Parameterization and invocation of these macros is controlled by the JMP user interface. After invocation, a macro is executed on the SAS server, and its results are returned to JMP and presented to the user. A SAS macro that is executed from a JMP Genomics dialog is called an *Analytical Process* (AP).

The documentation provided with JMP Genomics describes how to integrate user-written SAS macros into JMP Genomics.

This extensibility of JMP genomics can be used to integrate SAS macros that use an external R installation, resulting in the integration of R functions in JMP Genomics.

The necessary steps are:

1. Create a new SAS macro wrapping the use of an external R installation.
   In this paper, we evaluate two approaches:
   a. Dynamic generation of R scripts and execution via batch calls to R.
   b. Usage of data step Java objects that interact with R via the RServe TCP/IP server.
2. Create a XML file describing the graphical interface for macro parameterization.
3. Generate the user interface dialog out of the XML definition.
4. Link the user interface dialog to the appropriate menu in the user interface.

Below, we explain each step in detail.

CREATE A NEW SAS MACRO WRAPPING THE USE OF AN EXTERNAL R INSTALLATION

A SAS macro that implements an Analytical Process in JMP Genomics should be structured in the following way:

1. Preamble
   An introductory comment block.
2. Process Variable Definitions
   Definition of input parameters by use of global macro variables. These variables receive the parameter values which the user specifies upon invocation of the analytic process.
3. Initialization and Argument Checking
   In this section, local macro variables are defined, libname and %include statements are stated and any necessary input parameter checking is performed.
4. SAS Data Step and Procedure Code
   Here, the business logic of the AP is implemented. The values of the input parameters are used to create a result by potentially using any of the SAS language elements.
5. JSL File Generation (optional)
   This is an optional part of the SAS macro that dynamically creates a JMP scripting language (JSL) file which is invoked by JMP Genomics after execution of the SAS macro. Such JSL code can be utilized to use components of the JMP system, e.g. to display an interactive data visualization as a result of the AP invocation.
6. Package Publication
   In this part of the macro code, all generated results that should be returned to the user (i.e. SAS data sets, PDF reports, …) are designated. JMP Genomics provides utility macros for this task.
7. Macro Conclusion and Invocation
   The final part of the macro contains final error handling, closing of the macro definition and invocation of the macro.

All of these steps are described in detail in the Programmer Guide shipped with JMP Genomics. We will therefore concentrate on section (4.), which is also the place where one defines the access to an external software system. For the realization of the use of R functions in this part of the AP macro, the concepts of two technically different approaches are discussed:

A) Dynamic generation of R scripts and execution via batch calls to R

Basically, this approach resembles the way real world users would run R to generate results, but in this case SAS code “interacts” with the R system using its batch mode.

For this method, the subsequent steps have to be implemented:

a) Write (preprocessed) data to the file system as an input to R
   In case the data you want to process with the help of R is not already stored in the file system, the data is written to disk using one of the many formats readable by R.
b) Generate the R program for processing the data using R functions
   Using elements of the SAS macro language, a text file with R source code is dynamically created. This source code file implements data import into R, calculation of results, and export of results into the file system.
c) Generate the batch file that calls R in batch mode with the program just generated as input
   Invocation options for R should be included in the batch file, as well as an instruction for the redirection of the R log into a text file.
d) Execute the batch file
   This is either done using the ‘x’ command of the SAS language or using a utility macro provided by JMP Genomics called ‘LaunchExternalProgram’.
e) Check whether results have been generated
   At this point, at the very least a check for the existence of the expected result files should be implemented. Further checks like a scan of the R log could also be considered.
f) Access the results
   The results are either accessed for further processing or are simply returned to the user. (In the latter case, they only have to be designated for that as stated in the description of AP macro code above, section 6).

B) Usage of Data Step Java objects that interact with R via the RServe TCP/IP server
This more elaborate approach corresponds to the concept of a client/server architecture. Using the RServe TCP/IP server, it is possible to use R from various programming languages via a TCP/IP connection. There are C++ and Java libraries that encapsulate functions for the communication of a client with an Rserve server. We use such a library (REngine) to create a custom Java API (application programming interface) for the R functions that are needed. This Java API is then called from within SAS using data step Java objects. Usage of Java objects in SAS data steps is an experimental feature in SAS 9.1.3, but is in production in SAS 9.2 .
This approach condenses steps b) – d) of the batch call approach into one data step. Such a data step can be structured in the following way:

a) Declaration of used java objects
b) Calls to methods of java objects including return value evaluation
c) Deletion of used java objects

After this data step, subsequent SAS code can either access the results of the data step for further processing or return them to the user.

CREATE A XML FILE DESCRIBING THE GRAPHICAL INTERFACE FOR MACRO PARAMETERIZATION

JMP Genomics provides a mechanism to create an interface dialog for a given AP macro, based on a XML file describing the interface. The interface dialog is the user front-end utilized for macro parametrization and invocation. Using a predefined set of XML tags, convenient interface dialogs can be defined, including a variety of widget types, default values and dependencies between widgets. Please refer to the JMP Genomics Programmers guide for a list of usable XML tags. The .xml file containing the interface definition has to be stored in the ProcessLibrary folder of your JMP Genomics installation. Apart from the suffix, the file should have the exact same name as the AP macro, in order to let JMP know which dialog definition belongs to which AP macro.

GENERATE THE USER INTERFACE DIALOG OUT OF THE XML DEFINITION

The Genomics menu of JMP comprises an item named ‘Generate Dialogs from XML’. Invocation of this item results in (re-)creation of JSL dialog definitions out of all XML files in the ProcessLibrary folder of the JMP Genomics installation. Consequently, the manually created XML files with interface specifications are translated into JSL dialog definitions executable by JMP.

LINK THE USER INTERFACE DIALOG TO THE APPROPRIATE MENU IN THE USER INTERFACE

The possibility to customize the menu items of the JMP user interface can be used to achieve this. The selection of Edit > Customize > Menus in the JMP menu bar launches a dialog in which the new menu item can be added to the appropriate menu in JMP and linked to the .jsl file of the new AP.

A SIMPLE EXAMPLE

Following, an example is given for an AP which integrates an implementation of Variance-stabilizing normalization (VSN) via a batch call to R. Variance-stabilizing normalization of measurement data is commonly used in microarray analysis. For more information about this method, please refer to Huber et al. [3].
The AP takes, as input, a SAS data set that contains the measurement data (e.g. gene expression data) to be normalized. The data has to be in tall format, i.e. feature values in rows and experiments in columns. Execution of the AP results in a SAS data set with normalized data values.
Using a XML-based definition, the user dialog shown in Figure 1 can be created for the AP.

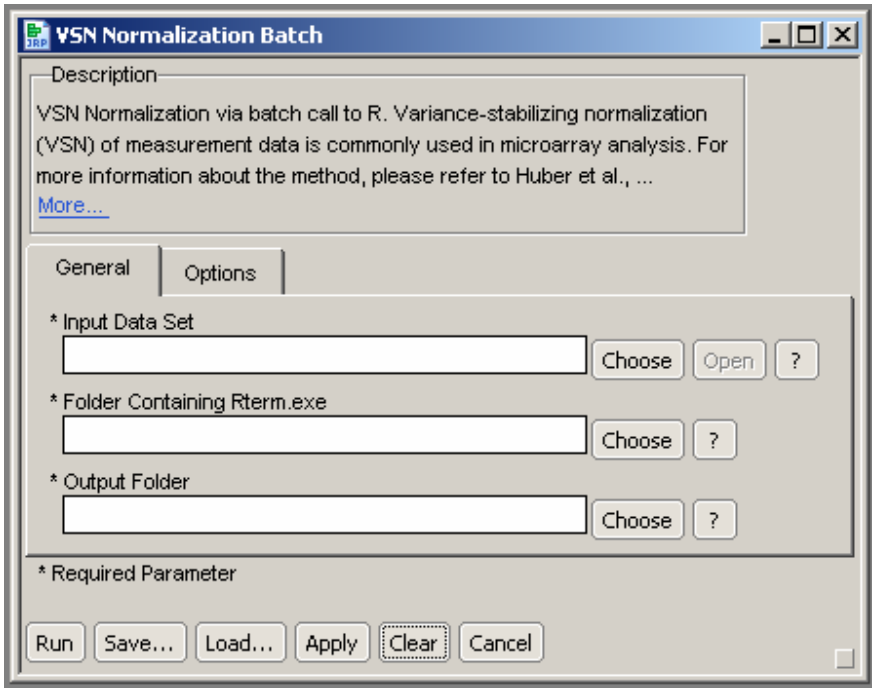

Figure 1. The user dialog for the Analytical Process described in the example.

The SAS macro for the AP that is invoked by the user dialog, takes the input data set and stores it as a text file in CSV format on disk. Then, an R script is dynamically created to implement data import into R, VSN normalization and data export to disk in CSV file format. The R script can be created in a simple data null step as follows:

```
/*Creation of R code, written to a script file*/
filename exp "&outPath.VSN_Normalization_Batch.r";
data _null_;
   file exp;

   /*the vsn library is loaded*/
   put 'library(vsn);';
   /*the csv file is loaded into the R workspace*/
   put 'inputData <- read.csv(file="' @@;
   put "&New_CSV_Input_Path" @@;
   put '", sep = ";");';
   /*Input Data is expected to contain three first columns with row
     information about the microarray probes which are removed for
     normalisation*/
   put 'datamatrix<- as.matrix(inputData[,c(-1,-2,-3)]);';

   /*Application of the justvsn function that implements an algorithm for
     variance-stabilizing normalization (VSN) of measurement data.
   put 'outmatrix<-justvsn(datamatrix);';

   /*Columns with row information about the microarray probes are attached
     to the file again*/
   put 'outdata<-data.frame(inputData[,c(1,2,3)],outmatrix);';

   /*resulting data is stored in csv format*/
   put 'write.csv(outdata, file="' @@;
   put "&outData..csv" @@;
   put '",row.names=FALSE, sep=";");';
run;
```

This R script is then executed via a batch call to R. For this, a batch file is created and afterwards executed by the utility macro `%LaunchExternalProgram`, which is provided by JMP Genomics.

```
/*creation of batch file with call to R, the R log file is redirected to a
  text file in the the output directory*/
```

```
filename bat "&outPath.rbatch.bat";
data _null_;
   file bat;
    put "cd &OutPath";
    put "path &RTermPath";
    put 'Rterm.exe --no-restore --no-save < %1 > %1.log 2>&1';
run;

/*The R code is executed via a batch call*/
%LaunchExternalProgram(&OutPath.rbatch.bat, ,&OutPath.VSN_Normalization_Batch.r,
&OutPath);
```

After that, the CSV file produced by R is read back to SAS and returned to the calling JMP Genomics environment, where it is presented to the user.

As an alternative, data step Java objects can be used that interact with R via the RServe TCP/IP server. For this, the Rserve library has to be installed in R and the RServe TCP/IP server has to be started. Then, at least one Java class has to be implemented that uses the Java library REngine to interact with the Rserve server. This java class encapsulates all R scripting completely into java object methods that can be called in SAS data steps. In the SAS AP macro, the above listed source code for generation and execution of an R script can be shortened to simple calls to a Java object:

```
data _null_;
   /*declaration of the java object used for access to R*/
   declare javaobj rc("sas2r/Normalization");
   /*establish connection to R*/
   rc.callVoidMethod("connectToR");
   /*vsn normalization performed in R, taking a stored CSV file as input
         and generating a stored CSV file as output.*/
   rc.callVoidMethod("vsn", "&New_CSV_Input_Path", "&New_CSV_Output_Path");
   /*close connection to R*/
   rc.callVoidMethod("closeConnection");
   /* delete java object*/
   rc.delete();
run;
```

In addition, Java methods that return values directly can be called in a SAS data step, e.g.:

```
/*variable for the return value of method getVersion*/
LENGTH R_VERSION $32;
/*get version of R*/
rc.callStringMethod("getVersion", R_VERSION);
/*write version of R to SAS LOG*/
put R_VERSION=;
```

Please note that your java classes have to be put into the class path of the java runtime environment that your SAS installation uses. This is done most conveniently by placing all your classes into a Java archive (.jar) file and copying it into one of the folders designated as a Java extension folder (current settings can be inspected by execution of PROC JAVAINFO in SAS and checking its output for the key ‘java.ext.dirs’).

DISCUSSION

COMPARISON OF BOTH IMPLEMENTATION APPROACHES FOR USAGE OF AN EXTERNAL R INSTALLATION

For single machine installations and simple analytical processes, the batch approach might be more appropriate, as it is adequate for such a scenario and much easier to implement. The second approach, using Rserve, is more complex to implement, but has the advantage of handling the interaction between SAS and R with a client/server approach, making it possible to run SAS and R on different server machines. In addition, the invocation of R code is encapsulated in a custom API that results in a better structuring of the implementation, allowing for the usage of features of the Java language. Another advantage is superior execution speed in comparison to the batch call approach, as the R system is started only once as a server and not every time an AP is executed. Thus, the constant start-up time of the R system is not part of the execution time of a SAS macro using the Rserve approach for access to R. In the example described above, this additional start-up time accrued for the batch call approach resulted in a difference of approximately 13 seconds in execution time. This time offset was measured by executing the same functionality with the two implementation approaches.

USAGE OF R FUNCTIONS IN JMP WITHOUT AN INSTALLATION OF THE GENOMICS ADD-ON

The benefit of being able to integrate existing R functions into JMP Genomics is quite obvious, particularly for the application domain of bioinformatics, since many more analysis methods are available as implementations in R. The usage of R and SAS implementations in combination with the user interface of JMP, though, could also be interesting in other application areas.

In principle, the JMP software allows for the execution of SAS programs without the installation of the Genomics add-on. It would also be thus possible to use an R installation with a basic installation of JMP, by either wrapping the access to R in an SAS program or using JSL directly. However, the Genomics add-on for JMP provides many useful utility programs simplifying the implementation and integration of SAS and R programs to be used in JMP.

CONCLUSION

To conclude, the technical solutions presented here make it possible to combine the great user interface of JMP Genomics with the wealth of R implementations of algorithms used in life sciences. Methods implemented in SAS or R can be accessed using an easy to use graphical interface, making the supplied functionality available to a broader audience of users.

VERSIONS OF USED SOFTWARE COMPONENTS

Software	Version
JMP	7
JMP Genomics add-on	3.2
SAS	9.1.3 SP 4
R	2.7.1
Rserve	0.5-2

REFERENCES

[1] R Development Core Team, R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org. (2006).

[2] Gentleman et al., Bioconductor: Open software development for computational biology and bioinformatics. Genome Biology, 5: R80. (2004).

[3] Huber et al., Variance stabilization applied to microarray data calibration and to the quantification of differential expression., Bioinformatics, 18 (Supplement 1): S96. (2002).

RECOMMENDED READING

JMP Genomics web site:
http://www.jmp.com/software/genomics/
JMP Genomics Programmer Guide:
SAS Institute Inc. 2007. JMP Genomics Programmer Guide. Cary, NC: SAS Press.
Documentation of the Rserve package:
http://www.rforge.net/Rserve/index.html
Documentation of SAS data step java objects:
http://support.sas.com/rnd/base/datastep/dot/javaobj.html

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Dr. Patrick R Warnat
HMS Analytical Software GmbH
Rohrbacher Str. 26
69115 Heidelberg
Germany
Fax: +49 6221 60 51 - 0
Email: patrick.warnat@analytical-software.de
Web: http://www.analytical-software.de