Paper 3089-2019

Working with Both Proprietary and Open-Source Software

Ernestynne Walsh, Nicholson Consulting Ltd

ABSTRACT

There is often debate around whether organizations should use proprietary or open source software. By utilizing both you can enjoy benefits that the enterprise values and gain access to a greater pool of talent that are used to working with a variety of tools. SAS Global Forum 2019 has a Hands-On Workshop that shows you how you can use R in SAS® Enterprise Miner™ and how R can interface with SAS® Viya™. This paper is the abridged version of the workshop that is publicly available on GitHub and has been reprinted and in some places modified with the permission of the author.

INTRODUCTION

In the past few years the number of approaches to integrating open source code into SAS has increased. Two such examples include the SAS® Enterprise Miner™ open source integration node and the Scripting Wrapper for Analytics Transfer (SWAT) package. This paper and Hands-On Workshop focuses on how to combine SAS with open-source tools, rather than teaching attendees how to code in R. For those wanting help with coding in R they can check out the references in the recommended readings. Examples of building predictive models in R using the open source integration node and the swat package are shown in the workshop along with tips and tricks. The data is publicly available so readers will be able to access the code on GitHub and run the code on their copy of SAS® software.

OVERVIEW OF TOOLS AND COMPONENTS

R

R is an open source programming language for statistical computing and machine learning. Being open source means that the source code is freely available and distributed. It is used a lot in academia and also in data science. There are over 10,000 packages available to perform specific tasks.

SAS® ENTERPRISE MINER™

This product is used to build predictive models and score them via a graphical user interface. The open source integration node allows you to create and score R models via SAS® Enterprise Miner™. You can find it under the Utilities tab. It handles the data and results transfer to and from R for you.

SAS® VIYA™

The SAS® Viya platform offers a variety of data mining and machine learning solutions. The runtime environment that does all the analytical work within SAS® Viya is called Cloud Analytic Services (CAS). The swat R package allows you to write R code that is submitted to CAS. Behind the scenes this code is converted into a series of CAS actions that run on CAS. Not all R functions have CAS actions (e.g. xgboost) but you can still run these R functions by bringing the data directly into R.

SAS® ENTERPRISE MINER™ AND OPEN SOURCE INTEGRATION

The two main elements to be aware of when working with the open source integration node are the useful helper macro variables and what options need changing. Once you have a basic understanding of these you can work through the workshop exercises.

THE MACRO VARIABLES

Macro variables allow you to modify text in code by substituting the macro variable with its specified string. They are great to use when passing data and results between SAS and R because:

- If you change your target or your input variables you don't need to rewrite the R code.
- R is case sensitive. The macro variables handle this.
- The target variable must be preceded by the letter r.
- Some of the macro variables must be defined so that the score code can be generated.

The macro variables and their definitions are specified below (Table 1).

Macro variable	Definition
&EMR_MODEL	This is the name of the model object in R
&EMR_IMPORT_DATA	Name of the dataset to be used in R to build the model (usually it is the training dataset)
&EMR_CLASS_TARGET	Name of the target variable (must be categorical)
&EMR_NUM_TARGET	Name of the target variable (must be numeric)
&EMR_CLASS_INPUT	Categorical variables used as inputs to the predictive model
&EMR_NUM_INPUT	Numeric/continuous variables used as inputs to the predictive model
&EMR_EXPORT_TRAIN	Name of the dataset containing the scored training data to be exported from the Open Source Integration node
&EMR_EXPORT_VALIDATE	Name of the dataset containing the scored validation data to be exported from the Open Source Integration node
&EMR_IMPORT_VALIDATE	Name of the validation dataset to be used in R for predictions

Table 1. List of Macro Variables used in the Open Source Integration Node and their Definitions

THE OPTIONS

Open Source Integration Node

Asides from the Code Editor ellipsis where you put your R code, the other option/property you will change is the output mode (Figure 1). The two main choices you are interested in here are PMML and Merge.

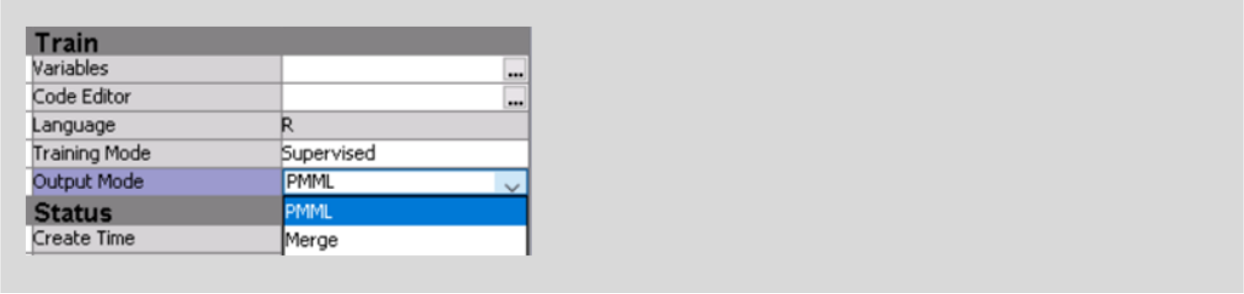

Figure 1: Output Mode Options

Most of the work is done for you with PMML. SAS score code is produced but PMML only works with the following R packages:

- base (linear models, generalized linear models, kmeans).
- rpart (decision trees).
- nnet (neural networks and multinomial log-linear models).

For a more complicated model like xgboost you will need to set the output mode to Merge mode. With merge mode:

- There is no SAS score code. Make use of R's predict() function.
- The Model Import node is required to compare the R model to the other models created in in SAS® Enterprise Miner™.

Model Import Node

The Model Import node can be found under the Model tab. The main option to change in the Model Import node is the Mapping Editor (Figure 2). It maps the R prediction variables to the SAS® Enterprise Miner™ prediction variables. This allows you to do model comparisons between SAS generated models and R generated models. You can bring up the mapping editor by clicking on the ellipsis.

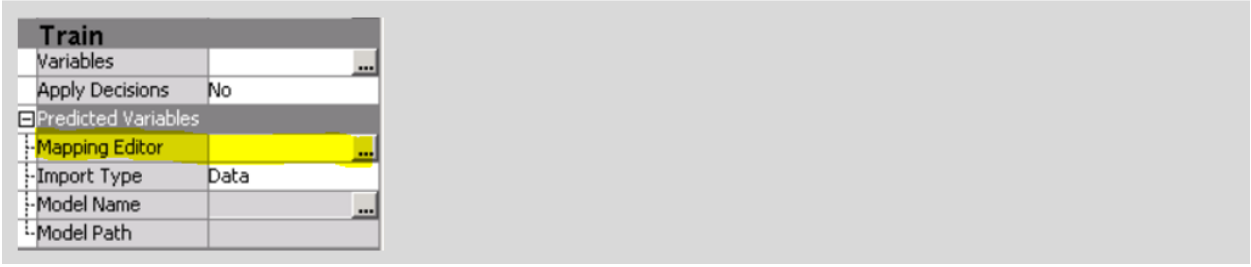

Figure 2: The Model Import Node and the Mapping Editor

You need to tell SAS® Enterprise Miner™ which variable represents the primary prediction and which one represents the secondary prediction. EMR_VAR1 represents level 0 and EMR_VAR2 represents level 1 (Figure 3).

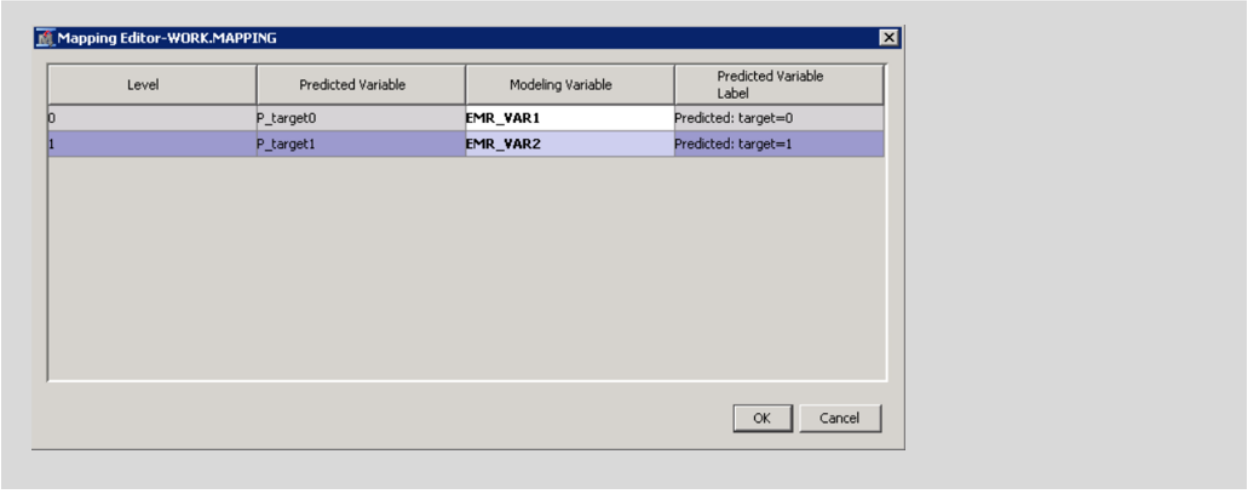

Figure 3: Mapping the Modelling Variables to the Predicted Variables

TIPS AND TRICKS FOR WORKING WITH R IN SAS® ENTERPRISE MINER™

These tips and tricks can be found in the workshop notes which also have some additional reference material.

MISSING VALUES

Watch out for missing values R and SAS handle missing values differently:

- R will either ignore them or encounter an error.
- Some SAS models like trees can use records with missing values.

WRITE SCRIPTS IN AN INTEGRATED DEVELOPMENT ENVIRONMENT LIKE R STUDIO

These are designed to speed up the development process by autocompleting variable names, highlighting syntax for easy error spotting and so on.

You are less likely to make mistakes compared to writing directly into the Code Editor.

USER A HELPER FILE FOR YOUR MACRO VARIABLES

Having all the macro variables that you need saved in a helper file and the default ones prepopulated saves you a few typos and a few errors.

GET THE ADMIN TEAM TO HELP WITH PACKAGE INSTALLATION

R needs to be on the SAS server. It is also a good idea to have the administrator install all the relevant packages ahead of time. When you install packages in R you can get a prompt asking what distribution site (i.e. what mirror) you would like to download the package from. A dialog box will not pop up in SAS® Enterprise Miner and the code will not run.

SAS® VIYA™ AND THE SWAT PACKAGE

The SAS® Enterprise MinerTM approach has a large graphical user interface component. The swat package is script based. The scripts can be developed using an Integrated Development Environment such as R Studio or a Jupyter notebook by using the SAS Kernel for Jupyter.

Once you have swat installed you can follow these steps:

1. Set up (loading packages, setting working directories and so on)
2. Creating a CAS connection
3. Load data into memory on CAS
4. Run CAS actions to perform data prep and modelling or pull data to R to run native models and data prep in R
5. Close the connection when you are finished

The quickest way to understand what the syntax looks like is to use an example:

```
#  step 1: setup
library(swat)

# step 2: connection
conn2cas <- CAS("server-name", port_num, protocol="http")

# step 3: load data into memory
tbl <- cas.read.csv(conn2cas,
"D:/Path/to/file/file.csv"
)

#also check out the dimension and names of the tables
dim(tbl)
names(tbl)

# step 4: work in CAS
# list and load action sets
listActionSets(conn2cas)
loadActionSet(conn2cas, 'decisionTree')

….

cas.decisionTree.forestTrain(conn,
table    = list(name = 'file' , where = '_PartInd_ = 1'),),
target   = tgt,
inputs   = inp,
nominals = nom,
nTree    = 20,
casOut   = list(name = 'rf_model', replace = TRUE)
)

# step 5: disconnect from CAS
cas.session.endSession(conn2cas)
```

TIPS AND TRICKS FOR WORKING WITH THE SWAT PACKAGE

TRADITIONAL R OBJECTS VS CAS OBJECTS

Sometimes you can lose track of whether your data/results are CAS objects or traditional R objects. You can make use of the class function to check

- If you get CASTable this resides on the server.
- If you get casDataFrame this resides on the server and the client (i.e. R).
- If you get some other object type (data frame, tibble etc) then you are on the client.

PROTECT YOUR PASSWORD

Having passwords in plain text and sharing them with others is a big no, no. It presents security and other risks. Often blogs show this as a quick and dirty way to get something up and running but not all of them note that this is not the way it should be done.

There are several ways you can protect your password when making connections such as using an authinfo file, an .REnviron file or using an advanced encryption standard. More details can be found in the workshop notes.

LOOK AT GITHUB

There are plenty of resources on the code sharing platform GitHub.

A more detailed example (sassoftware, 2016) of building several types of models in R that interface with CAS is available online.

There is also a book called SAS Viya: The R Perspective. The code snippets are available as R markdown files on GitHub (sassoftware, 2018).

CASTABLE OBJECT VS TABLE NAME

This is another area that can cause confusion. With a command like:

```
castbl_cas_crash <- cas.read.csv(conn2cas, "cas_crash.csv")
```

castbl_cas_crash is a CASTable object.

On the other hand, this is a character object:

```
table_name_str <- 'cas_crash'
```

When an action is asking for a table name it will want the string/character. It is ok to pass the object as long as the object is a string/character. Other times, a CASTable object is needed. For example, to.casDataFrame() requires a CASTable object.

For more details, refer to the documentation when you use a function from the swat package for the first time.

RUNNING THROUGH THE WORKSHOP

STEPPING THROUGH THE EXAMPLES

The full set of workshop notes (Walsh, 2019) is publicly available on GitHub. It is recommended that readers go through that material as well because it contains many useful references as well as material that would be relevant to Python users.

To go through the examples covered in the workshop:

1. Go to the GitHub project https://github.com/nicholson-consulting/sgf-workshop
2. Review notes-exercise-viya.pdf or notes-exercises-enterprise-miner.pdf which also contain references for installation instructions if you require them.
3. Review the README which contains instructions for the workshop. This includes instructions to download the New Zealand Transport Association Crash Analysis System dataset that is used in the workshops.

LOOKING AT THE SOLUTIONS

Those who do not have the required software but are still interested in viewing the code can look at the xml diagrams and R scripts in the solutions folder https://github.com/nicholson-consulting/sgf-workshop/tree/master/solutions

CONCLUSION

SAS offers many ways to integrate open source tools into your analysis. The open source integration node in SAS® Enterprise Miner™ and the swat package are two examples of how this is possible. The examples shown in the workshop provide a taster of how this can be carried out while the tips and tricks help readers avoid some of the common traps.

REFERENCES

sassoftware. 2016. "sas-viya-programming". Accessed 26 January 2019.

https://github.com/sassoftware/sas-viya-programming/blob/master/developerTrial/R/Basic%2BR%2Bnotebook%2Bexample.ipynb

sassoftware. 2018. "sas-viya-the-R-perspective". Accessed 26 January 2019. https://github.com/sassoftware/sas-viya-the-R-perspective

Walsh, E. 2019. "sgf-workshop". Accessed Feb 17, 2019. https://github.com/nicholson-consulting/sgf-workshop

ACKNOWLEDGMENTS

A big thank you to Paul Stone, Aimee Whitcroft and Amanda Hughes for suggesting an open source dataset that could be used for the workshop. Special thanks to the New Zealand Transport Agency for publicly sharing the Crash Analysis System data and the support they and Paul Stone have given me with troubleshooting the data.

RECOMMENDED READING

Stack Overflow is a great place to ask R specific questions (most of the time someone has already asked your question)
https://stackoverflow.com/

R-Bloggers is a great site with 'how to' tutorials.
https://www.r-bloggers.com/

CONTACT INFORMATION

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

Ernestynne Walsh
Nicholson Consulting Ltd
ernestynne@nicholsonconsulting.co.nz
https://www.nicholsonconsulting.co.nz/