Viya, R and the swat package

What is the swat package?

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.

To learn more about action sets refer to the folowing link
https://documentation.sas.com/?docsetId=pgmdiff&docsetTarget=p06ibhzb2bklaon1a86ilj3wpil9.htm&docsetVersion=3.4&locale=en#n038r32gkh6lw9n1nehq3dttefg3

What about Python?
There is a python equivalent that allows you interface with the CAS server. You can find more details about it on the GitHub page
https://github.com/sassoftware/python-swat

SAS also has a book called: SAS Viya: The Python Perspective available to purchase. All the code snippets are available as Jupyter Notebooks
https://github.com/sassoftware/sas-viya-the-python-perspective

How do you set up the swat package?

Your administrator needs to install a 64 bit version of R (>3.1.0) on a 64 bit machine. Once that is done the swat package and its dependencies can be installed. More details can be found here https://github.com/sassoftware/R-swat

Some things that can trip people up

- The package is not available on CRAN you will need to point to the release on the GitHub website if you use install.packages
- Release names are case sensitive. The package you are downloading is **R-swat** (i.e. make sure the R is upper case)
- Make sure you have the dependencies installed

Using CAS actions to run your code in SAS

Once you have all the relevant packages 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

An example is shown below

```
#  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)
```

Another example can be found in this blog by Joe Furbee
https://blogs.sas.com/content/sgf/2018/07/25/using-rstudio-with-sas-viya/

Exercise

Hint: You can highlight code in R and hit Ctrl + Enter to run the highlighted section or run line by line by hitting Ctrl + Enter while you are within the script.

Ex-4a Connecting to CAS

1. Start up R Studio. You can find it under the Windows Start button Scroll down to the letter R and you will see it.
2. Open the script **Ex-4.R** it has most of the script completed so you just need to focus on filling in the blanks
3. Use the library function to load the swat package
4. Create an object called conn2cas that is a connection to the CAS server
5. Run the code and confirm that you get the following message

```
NOTE: Connecting to CAS and generating CAS action functions for loaded
      action sets...
NOTE: To generate the functions with signatures (for tab completion), set
      options(cas.gen.function.sig=TRUE).
```

Ex-4b Finding and loading CAS actions

Continue editing the script from Exercise 4a

1. Add to your script the line **listActionSets(conn2cas)$actionset** to see some of the other action sets available
2. Load the **'decisionTree'** action set to access the random forest functions

Ex-4c Building a model

Two actions have been done already one that creates a partition variable called _PartInd_ and the other that performs the imputation

1. After these two functions use the example in the notes to create a random forest

Ex-4d Bonus pulling the data to R so that you could run R models like xgboost

1. Figure out how to create a data frame that can be used to run models natively in R. Hint: you will need to experiment with the function **to.casDataFrame** and pass it the name of the CAS Table object.

2. Use the **class** function to determine what type of object is produced from **to.casDataFrame** . What type of object is **castbl_cas_crash**?

3. How many rows are in the data frame? Does it differ from how many were in the CAS table? Review the documentation on cas.max.download.rows and the obs argument for to.casDataFrame for more information.

Tips and Tricks

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 following code to check

```
class(<object_name>)
```

- 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.

This information could be stored in an authinfo file. More can be read about them here https://documentation.sas.com/api/docsets/authinfo/9.4/content/authinfo.pdf?locale=en

There are also ways of protecting login information in R ranging from simple things like an .REnviron file.
https://blog.revolutionanalytics.com/2015/11/how-to-store-and-use-authentication-details-with-r.html

to more secure ways using an advanced encryption standard
https://blog.revolutionanalytics.com/2015/12/securely-storing-your-secrets-in-r-code.html

Look at GitHub blogs

There are plenty of good resources available online. In addition to the resources already mentioned above:

There is also a book called SAS Viya: The R Perspective. The code snippets are available as R markdown files on GitHub
https://github.com/sassoftware/sas-viya-the-R-perspective

A more detailed example of building several types of models in R that interface with CAS can be found here
https://github.com/sassoftware/sas-viya-programming/blob/master/developerTrial/R/Basic%2BR%2Bnotebook%2Bexample.ipynb

Note if you do prefer in person training SAS does offer a course called **SAS Open Source Series: SAS Viya and Open Source Languages** (OSSVIYAOSL).

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,
table_name_str <- 'cas_crash'

is a character object.

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.

References

de Vries, A. (2015). *How to store and use webservice keys and authentication details with R*. Retrieved 26 January 2019 from https://blog.revolutionanalytics.com/2015/11/how-to-store-and-use-authentication-details-with-r.html

de Vries, A. (2015). *Securely storing your secrets in R code*. Retrieved 26 January 2019 from https://blog.revolutionanalytics.com/2015/12/securely-storing-your-secrets-in-r-code.html

Furbee, J (2018). *Using RStudio with SAS Viya*. Retrieved 26 January 2019 from https://blogs.sas.com/content/sgf/2018/07/25/using-rstudio-with-sas-viya/

SAS (2018). *An Introduction to SAS Viya 3.4 Programming*. Retrieved 26 January 2019 from https://documentation.sas.com/?docsetId=pgmdiff&docsetTarget=p06ibhzb2bklaon1a86ili3wpil9.htm&docsetVersion=3.4&locale=en#n038r32gkh6lw9n1nehq3dttefg3

SAS (2017). *Client Authentication Using an Authinfo File*. Retrieved 26 January 2019 from https://documentation.sas.com/api/docsets/authinfo/9.4/content/authinfo.pdf?locale=en

sassoftware (2016). *sas-viya-programming*. Retrieved 26 January 2019 from https://github.com/sassoftware/sas-viya-programming/blob/master/developerTrial/R/Basic%2BR%2Bnotebook%2Bexample.ipynb

sassoftware (2017) *python-swat*. Retrieved 26 January 2019 from https://github.com/sassoftware/python-swat

sassoftware (2017) *R-swat*. Retrieved 26 January 2019 from https://github.com/sassoftware/R-swat

sassoftware (2018). *sas-viya-the-R-perspective*. Retrieved 26 January 2019 from https://github.com/sassoftware/sas-viya-the-R-perspective