Paper SAS3331-2019

Using Custom, User-Defined SAS® Formats with SAS® Visual Analytics on SAS® Viya®

Andrew Fagan SAS Institute Inc.

ABSTRACT

Are formats associated with your data? Formats that you created in SAS® using the FORMAT procedure? If you want to use SAS® Visual Analytics on SAS® Viya® with your data, you need to load your formats so that they can be found. In SAS®9, this meant adding the catalog that contained your formats to the format search path. On SAS Viya, there is an analogous task but the process is different. First, SAS Viya cannot read a SAS catalog, so you need to convert your catalog to a structure that SAS Viya understands. Next, you need to place the formats in a location where SAS Viya can find them. And finally, you need to tell SAS Viya to load the formats whenever a new session is started. This paper explores how you, as a user of SAS Visual Analytics on SAS Viya, can accomplish these steps and make your formats available. You can do this by using SAS® Environment Manager, SAS® Studio, or shell scripts. Each of these methods is described in detail, including sample code, and the benefits and limitations of each are presented.

INTRODUCTION

Formats are commonly used in SAS to map one value to another. Data is often stored as key values or in a shorthand form, but those same values are not always meaningful in a report. One of the simplest examples is date values: SAS stores a date as the number of days since January 1, 1960. It is unlikely many people reading a report would like to see today's date displayed as 21,600 rather than May 1, 2019. So SAS supplies a format that will take as input the number of days and convert it to a number of different friendly date displays.

But SAS can't supply formats for data values that are specific to your data. For that, SAS supplies a procedure (PROC FORMAT) that allows you to create your own formats, mapping one value to another. Many SAS users have made extensive use of this procedure and as a result have a large library of formats associated with their data. In some cases, the data is physically bound to an associated format, so the format must be present in order for the data to be used.

In SAS 9, formats are stored in a catalog and these catalogs can be added to a format search path. The search path is unique to each SAS session, and when SAS encounters data that requests a format, it searches the catalogs in this search path to find it. As long as the format is in a catalog on the search path, SAS can find it and everything works as planned.

SAS Viya can use the same formats used in SAS 9. But making them accessible to SAS Viya is not done in the same way. This paper explains three different options for making your formats accessible to SAS Viya.

THE DATA

The data used in this paper is based on values in the sashelp.cars table that is available with all SAS 9 deployments. The data itself is not an important part of the paper, but it provides several good examples of where you might use formats and for what purpose. Examples of a format on a character field, on a numeric field, formats that map values on-to-one, and formats that are used to categorize values are shown.

Both the data and the formats need to be stored in a permanent location (you cannot use the saswork library). This is because a copy of each will need to be loaded into SAS Viya. For illustrative purposes, the paper assumes they are stored in a directory named C:/temp/SGF2019 on Windows or /tmp/SGF2019 on Linux.

In order to generate the data used, you can submit this DATA step (but don't do it yet, the formats need to be defined first):

```
options fmtsearch=(SGF2019.carFormats) ;

data sgf2019.carData (keep=make model mfgCountry type MSRP msrp_range mpg
mfgAvail) ;
    attrib make length=$13
    model length=$40
    type length=$8
    mfgCountry length=$32 format=$mfgCountry. label="Country of origin"
    mfgAvail length=$32 format=$mfgCurrent. label="Available in US?"
    mpg length=8. format=4.1 label="Combined city/hwy mileage"
    MSRP length=8. format=DOLLAR8. label="Suggested Retail Price in USD"
    msrp_range length=8. format=priceRange. label="Price Range"
    ;
    set sashelp.cars (keep=make model type MSRP mpg_city mpg_highway) ;
    mfgCountry = make ;
    msrp_range = MSRP ;
    mfgAvail = make ;
    mpg = ((.55*mpg_city) + (.45*mpg_highway)) ;
run ;
```

THE FORMATS

The formats associated with the data are simple but illustrate some common usages.

Format name	DB2 Data Type
$mfgCountry.	Maps the manufacturer to the country of origin
$mfgCurrent.	Maps the manufacturer to a simple yes/no on whether it is currently available in the US.
priceRange.	Classifies the price of a vehicle into a category

Table 1. Formats associated with the data used in the paper

These formats are created by using PROC FORMAT and submitting the code to run in SAS.

In order to generate the formats used, you can submit this code:

```
libname sgf2019 "C:\temp\sgf2019" ;
proc format library=sgf2019.carFormats ;
    value $mfgCountry
          "Acura" = "Japan"
          "Audi" = "Germany"
          "BMW" = "Germany"
          "Buick" = "USA"
          "Cadillac" = "USA"
          "Chevrolet" = "USA"
          "Chrysler" = "USA"
          "Dodge" = "USA"
          "Ford" = "USA"
          "GMC" = "USA"
          "Honda" = "Japan"
          "Hyundai" = "Korea"
          "Infiniti" = "Japan"
          "Isuzu" = "Japan"
          "Jaguar" = "United Kingdom"
          "Jeep" = "USA"
          "Kia" = "Korea"
          "Land Rover" = "United Kingdom"
          "Lexus" = "Japan"
          "Lincoln" = "USA"
          "MINI" = "United Kingdom"
          "Mazda" = "Japan"
          "Mercedes-Benz" = "Germany"
          "Mercury" = "USA"
          "Mitsubishi" = "Japan"
          "Nissan" = "Japan"
          "Oldsmobile" = "USA"
          "Pontiac" = "USA"
          "Porsche" = "Germany"
          "Saab" = "Sweden"
          "Saturn" = "USA"
          "Scion" = "Japan"
          "Subaru" = "Japan"
          "Suzuki" = "Japan"
          "Toyota" = "Japan"
          "Volkswagen" = "Germany"
          "Volvo" = "Sweden"
          other = "_unknown_"
    ;

    value $mfgCurrent
          "Isuzu" = "No"
          "Mercury" = "No"
          "Oldsmobile" = "No"
          "Pontiac" = "No"
          "Saab" = "No"
          "Saturn" = "No"
          "Suzuki" = "No"
```

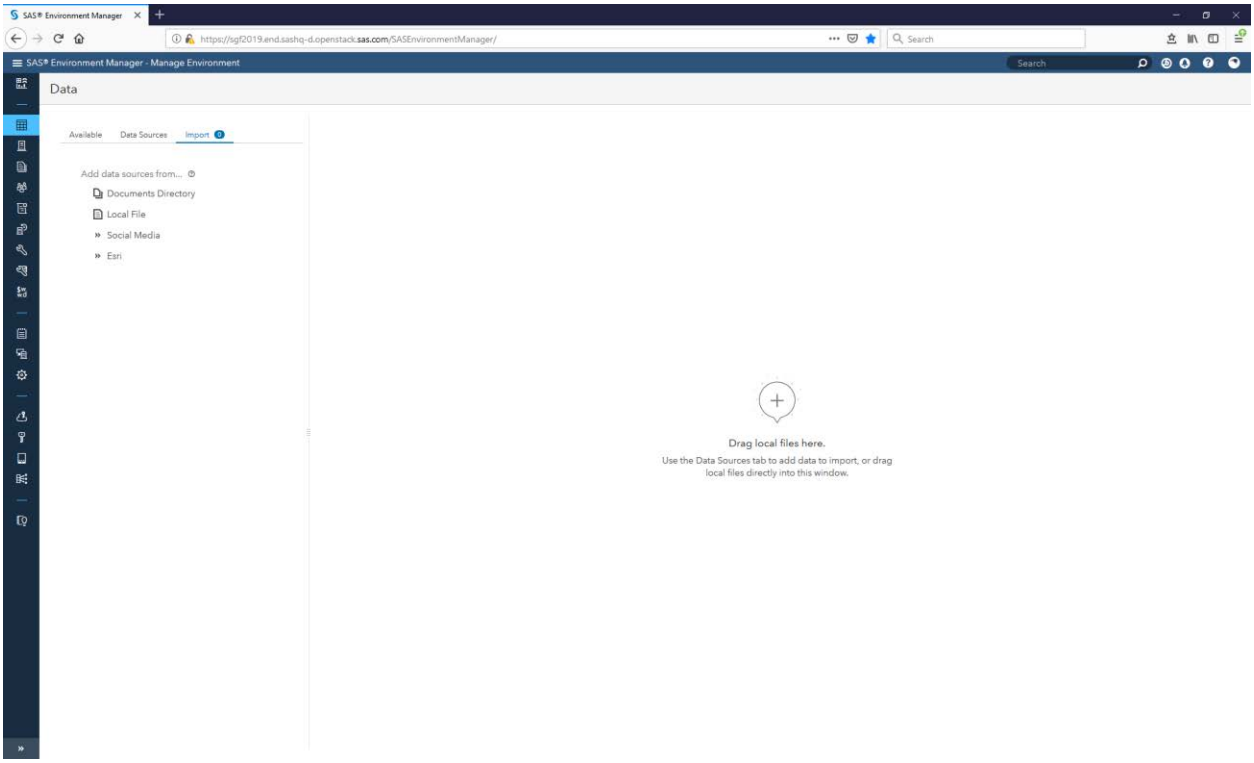
```
            other = "Yes"
      ;

      value priceRange
            0-15000 = "Budget"
            15001-30000 = "Medium"
            30001-40000 = "High"
            other = "Luxury"
      ;
  run ;
```

With both the data and the formats defined, we can try to use them in SAS Viya.

LOADING THE DATA INTO SAS VIYA

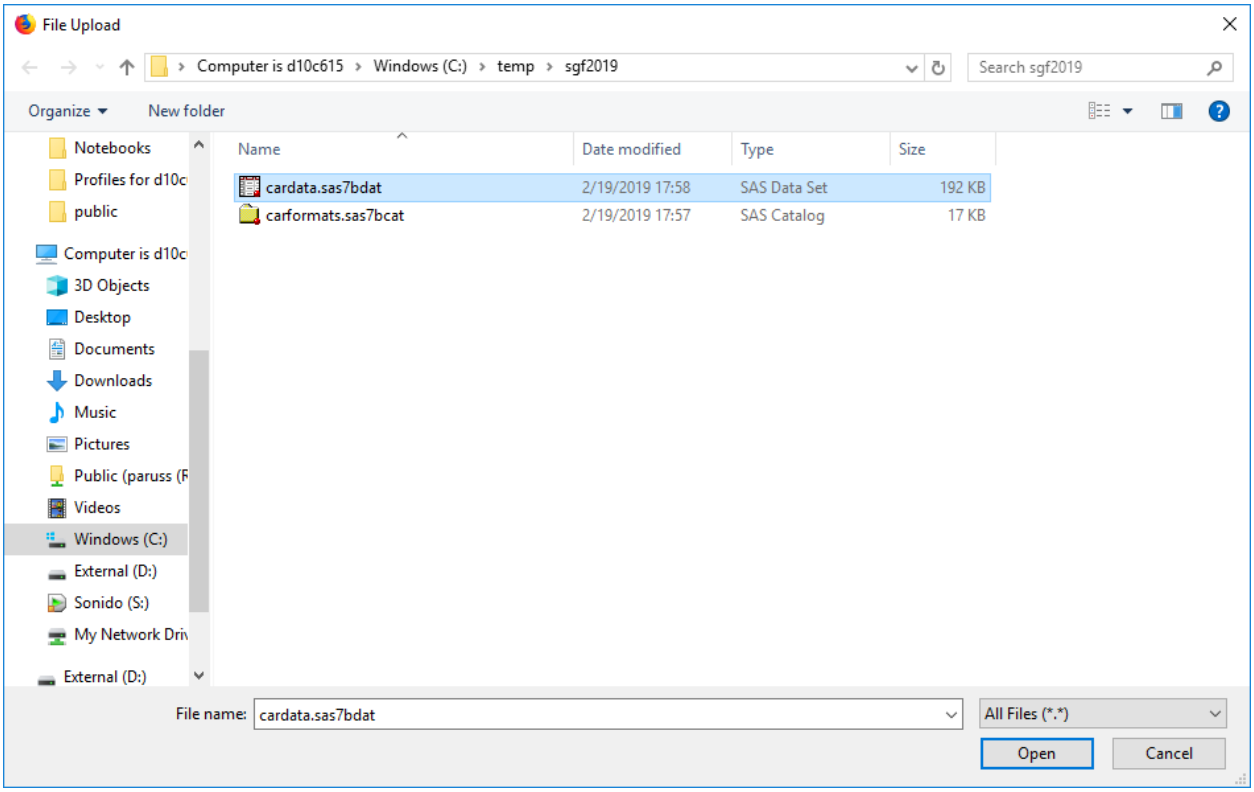
There are a number of methods available to load data into SAS Viya. For this example, we will use the Data Import feature available in SAS Environment Manager. The data will be loaded into a library named SGF2019_Data. The exact location of the data is not important to this example: once the formats are defined to SAS Viya, they will be available to any table in any library.

In SAS Environment Manager, choose Data then Import.



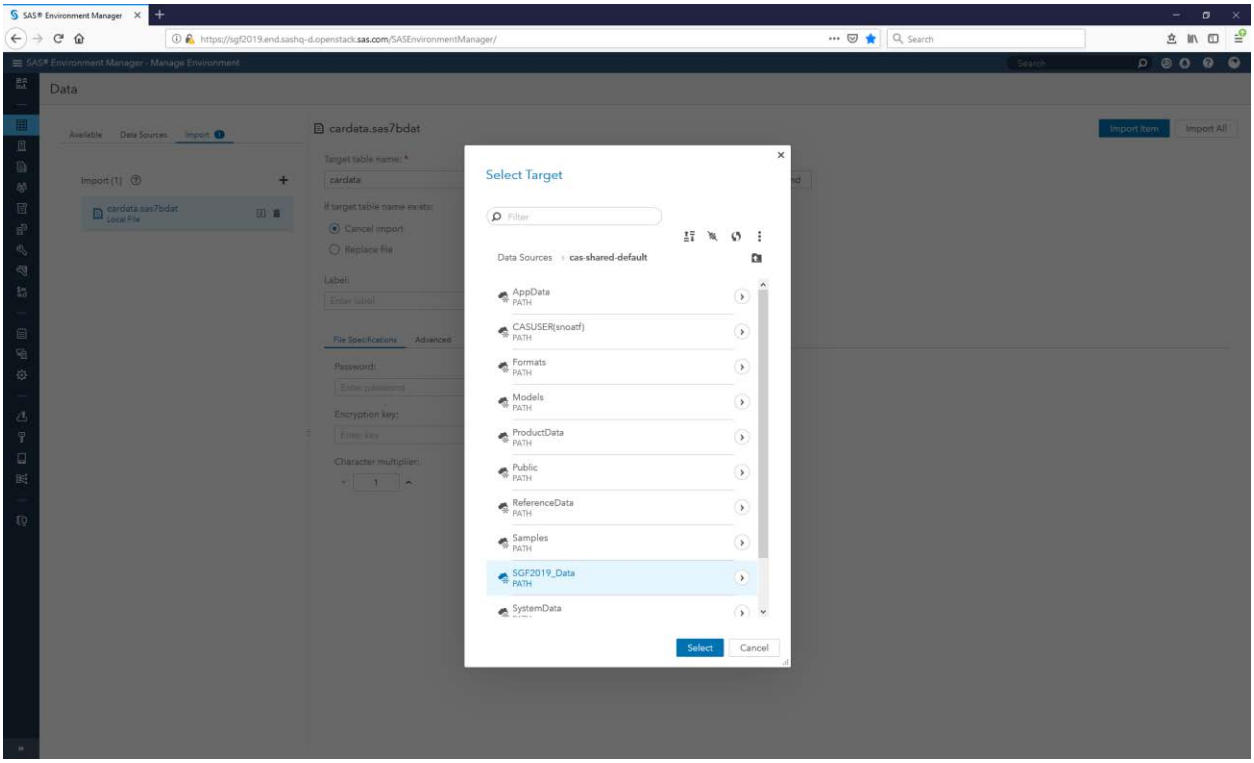
Display 1. SAS Environment Manager Data Import

Select Local File and navigate to the location on disk where you saved the carData table.



Display 2. SAS Environment Manager Data Import Local File selection

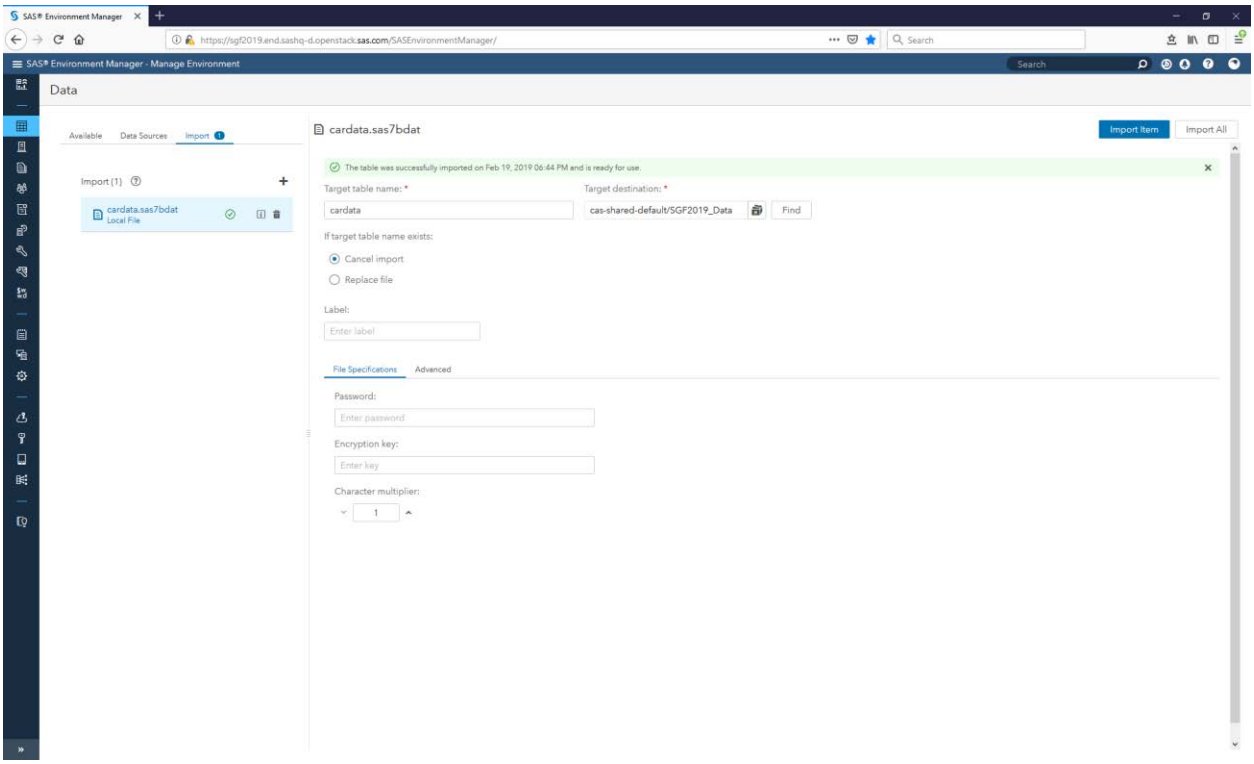
Next, you want to specify the library to load the data into.



Display 3. SAS Environment Manager Data Import Library selection

The exact library is not important. For this example, I have created a library specifically for the data but it could have just as easily been added to the existing Public library or any other available library that you have the appropriate access to.

Choose Import Item in the top, right corner and your table will be imported.

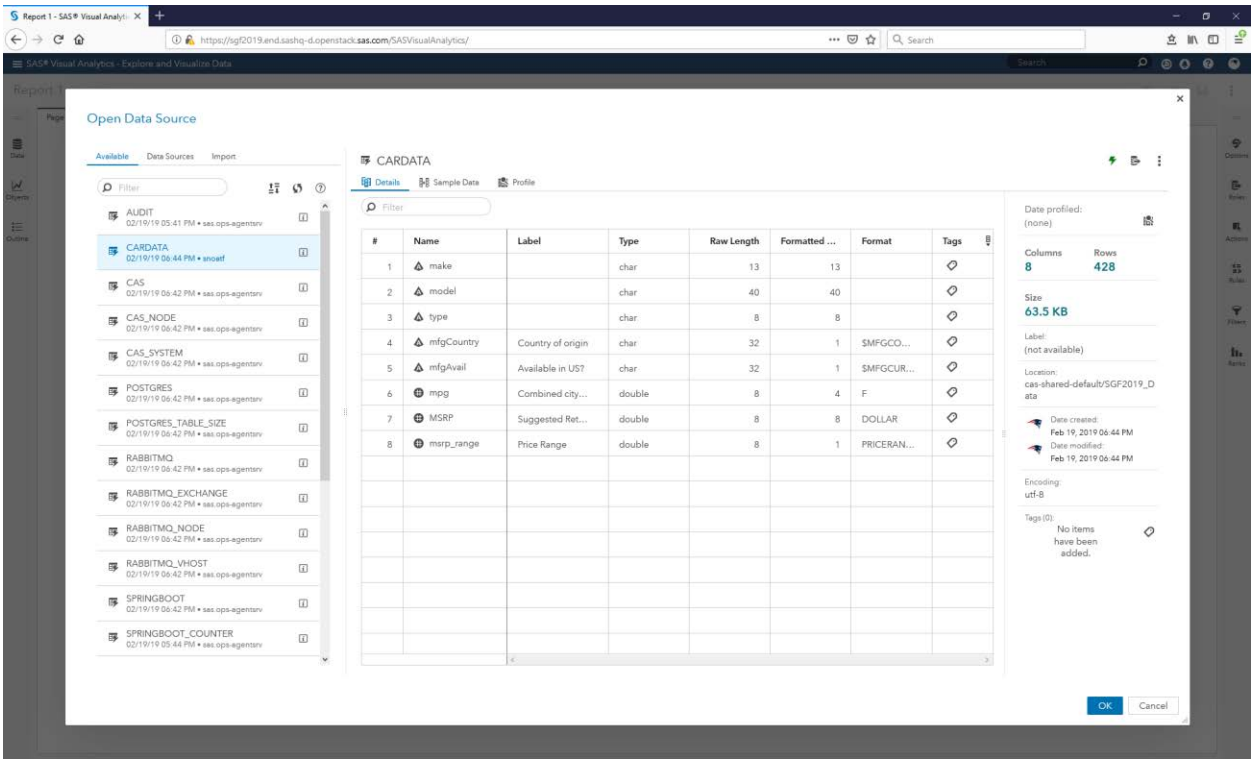


Display 4. SAS Environment Manager Data Import complete

ATTEMPTING TO ACCESS THE DATA WITHOUT THE FORMATS

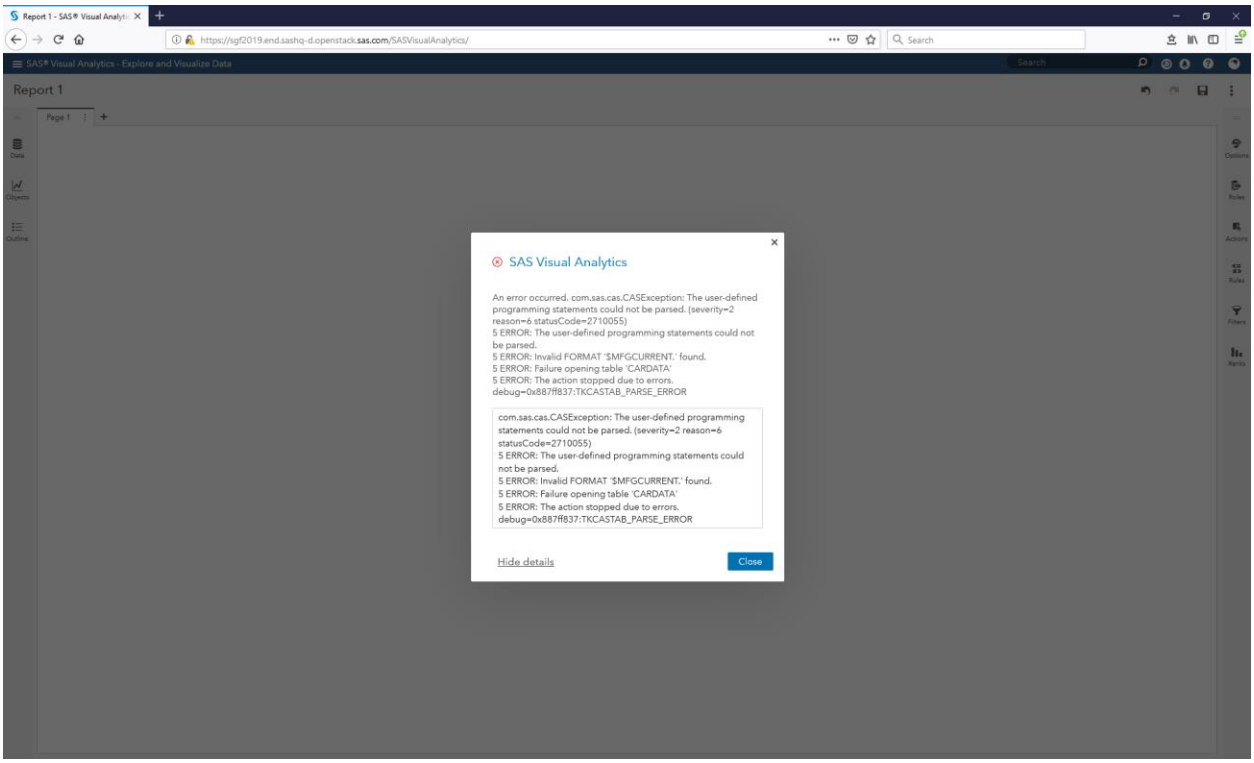
With the data loaded, you can now try to use the data in SAS Visual Analytics.

In Explore and Visualize Data, choose your carData table to work with.



Display 5. Explore and Visualize Data table selection

Everything looks great until you choose OK. You then get this lovely error message that in its own way is trying to tell you that it can't find the formats associated with your data.



Display 6. Explore and Visualize Data error when opening table

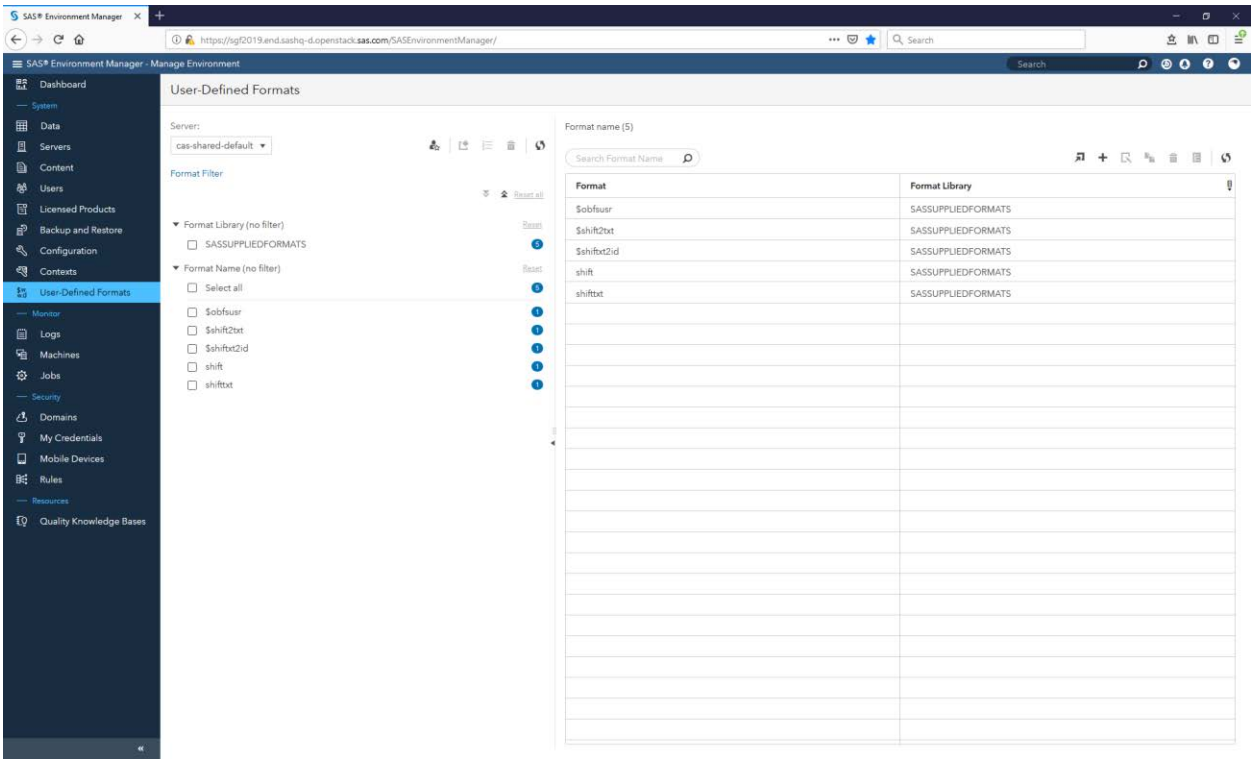
Since the formats cannot be found, the data is inaccessible and any analysis you attempt to perform will fail. There is currently no way to tell SAS Visual Analytics to ignore the formats and proceed (similar to the NOFMTERR option in SAS 9).

Unless you want to go back to your SAS programming environment and re-create the table without referencing any formats, you will need to make the formats available.

METHOD 1: SAS ENVIRONMENT MANAGER

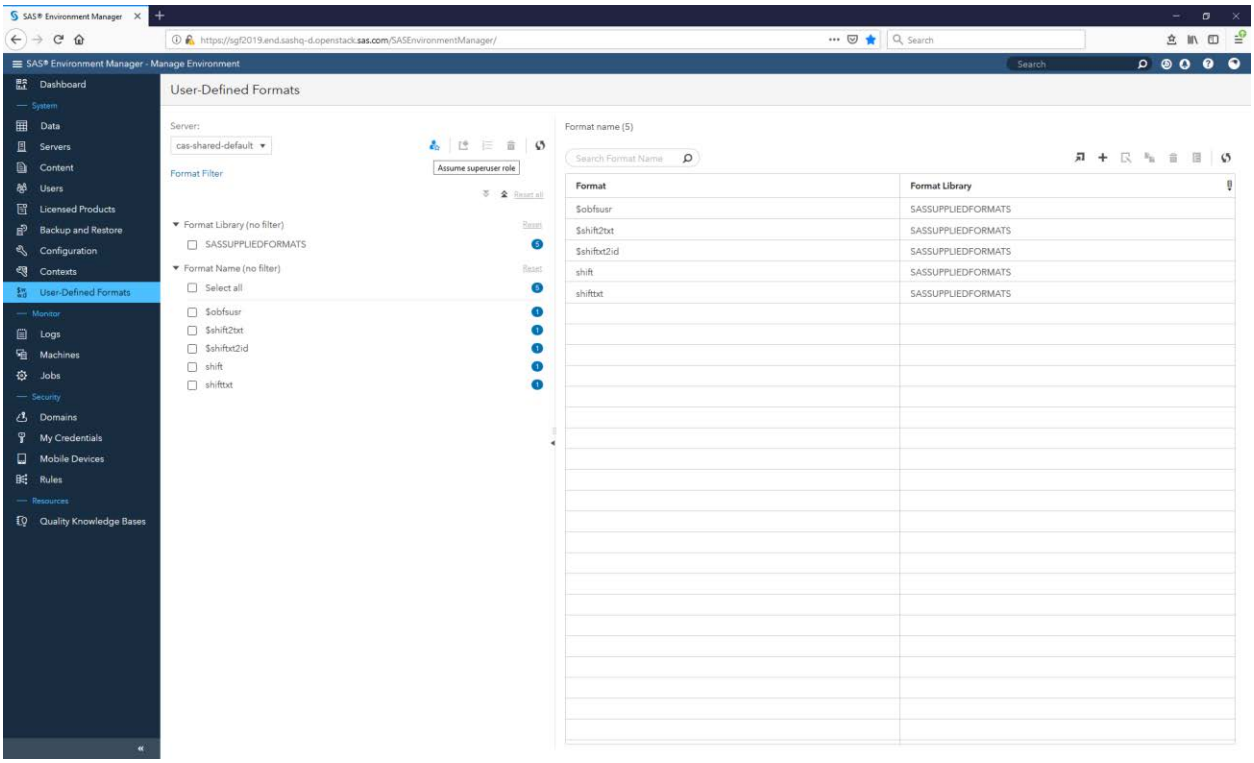
Often the easiest way to make a limited number of formats available to SAS Viya is by using the point and click interface in SAS Environment Manager. This feature is only available to members of the SAS Administrators security group since it impacts the overall configuration for all users.

If you are a member of the SAS Administrators group, you will see a menu option for User-Defined Formats (the very last choice in the System group). The first time you use this feature, the only formats that will be present are those supplied by SAS, stored in a format library aptly named SASSUPPLIEDFORMATS. Once you have added your own formats, they will also be visible on this screen to other administrators as well.



Display 7. Initial User Defined Formats screen in SAS Environment Manager

We need to create a new Format Library where our formats will be stored. This is a special task that requires you to Assume the Superuser role. Click on the corresponding icon at the top left of the screen to temporarily become a Superuser.

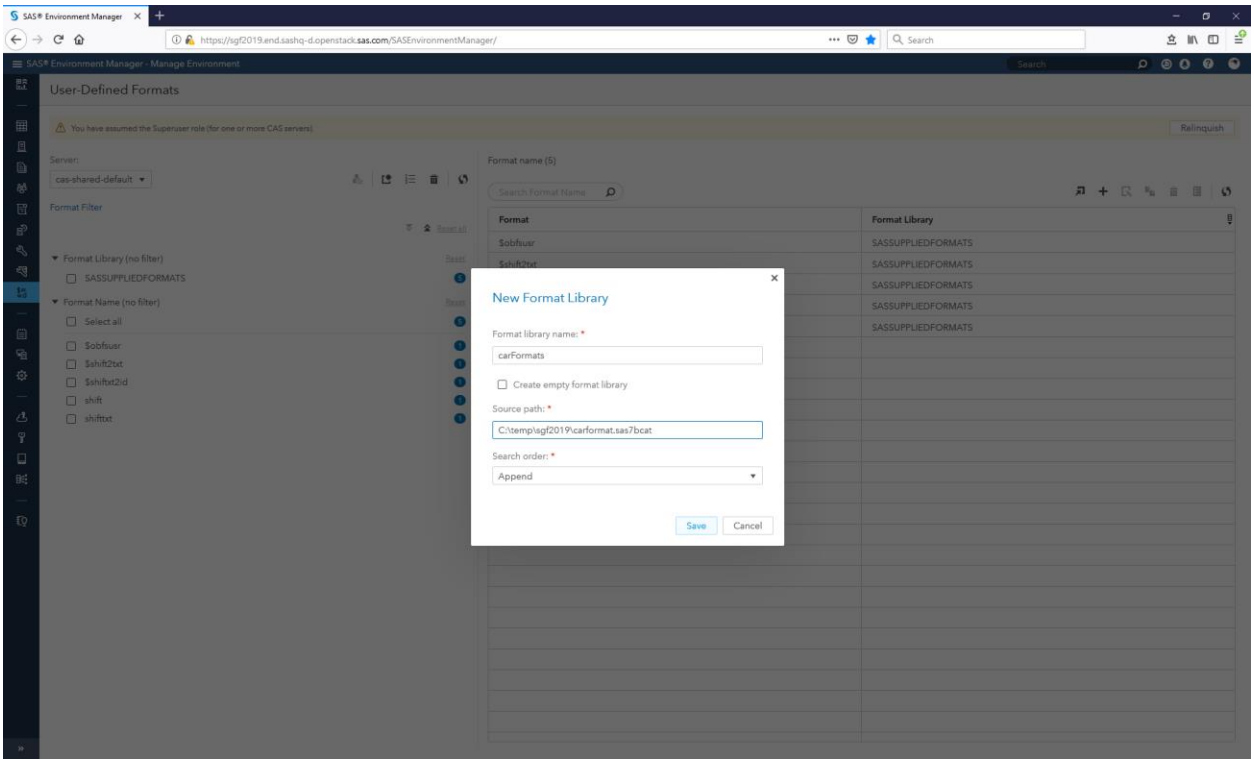


Display 8. Assume the superuser role

Once you have assumed the superuser role, the icon representing the creation of a new format library is available. Select it and you can supply the parameters for the new format library. While it is possible to create an empty format library and add formats to it later, we are going to use the shortcut to create the library and populate it with our formats in one step.

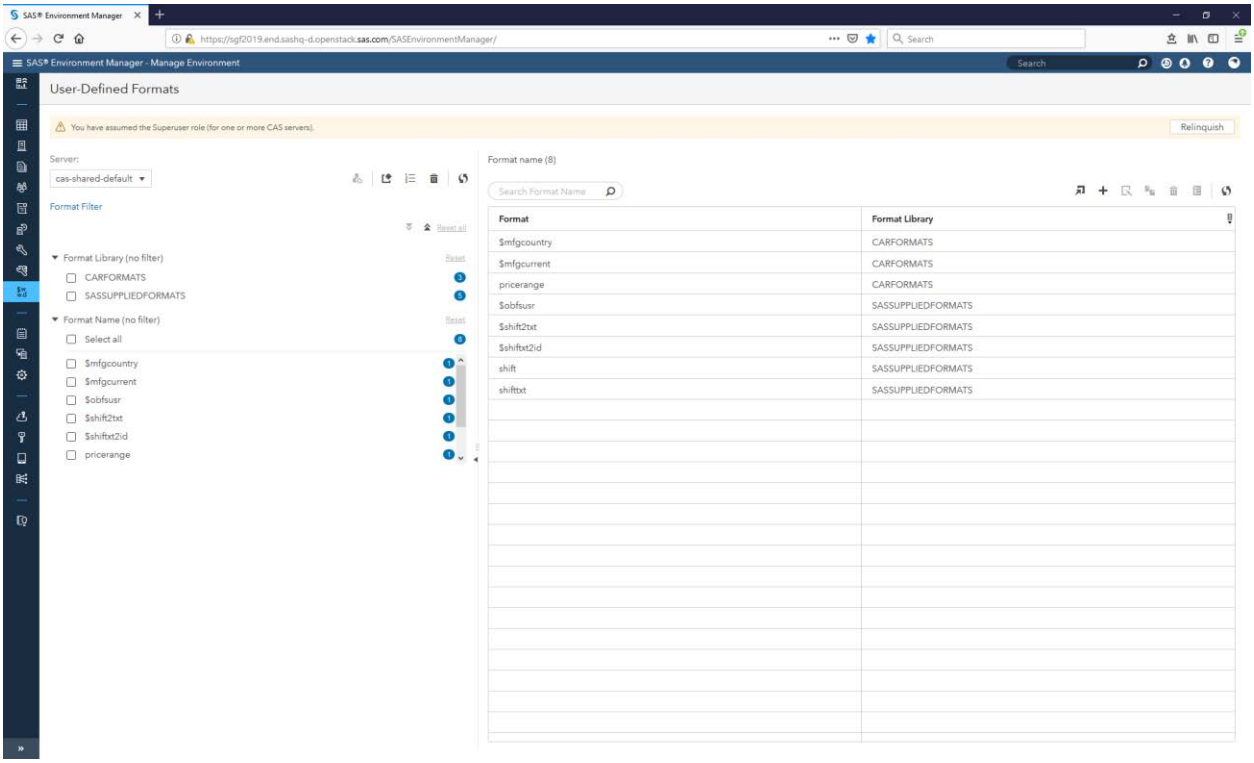
The name can be any valid SAS library name. Typically, you want to make this something meaningful, so another administrator will have an idea of what type of formats it contains. The source path field is the OS path to the SAS catalog containing the formats. Note that there is no Browse button for this field, so you must enter it in correctly (cut and paste is your friend). The search order allows you to specify how these formats relate to other format libraries that are already defined. You can choose to put them at the start of the search order, at the end, to replace the existing search order completely, or to add the formats but not put them in the search order at all. Append is typically what you want to do. Be very careful with Prepend or Replace as you will be impacting, and potentially changing in an unforeseen way, existing format usage.

Enter the fields and choose Save.



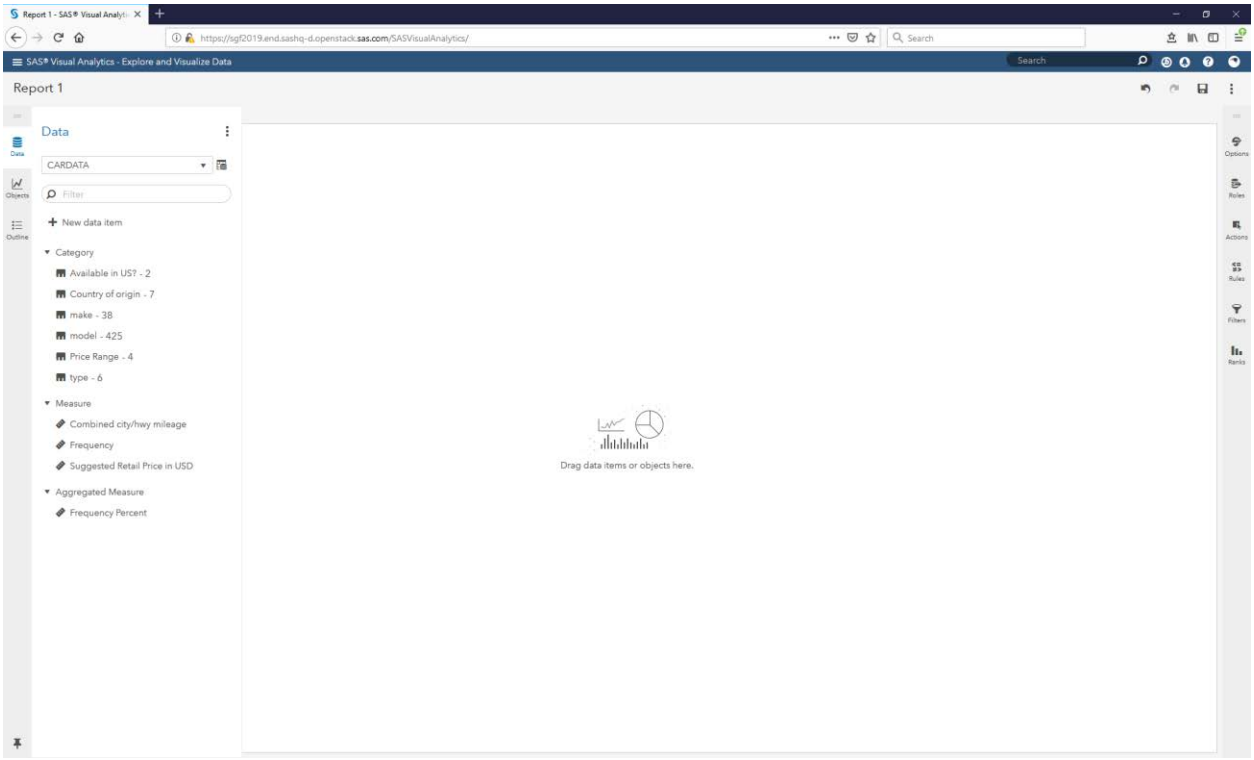
Display 9. Create a new format library

The new format library is created and your formats are added to the search path.



Display 10. New format library created and formats added to the search path

If you return to Explore and Visualize Data, your report will now open successfully.



Display 11. Data available for analysis once the formats are loaded

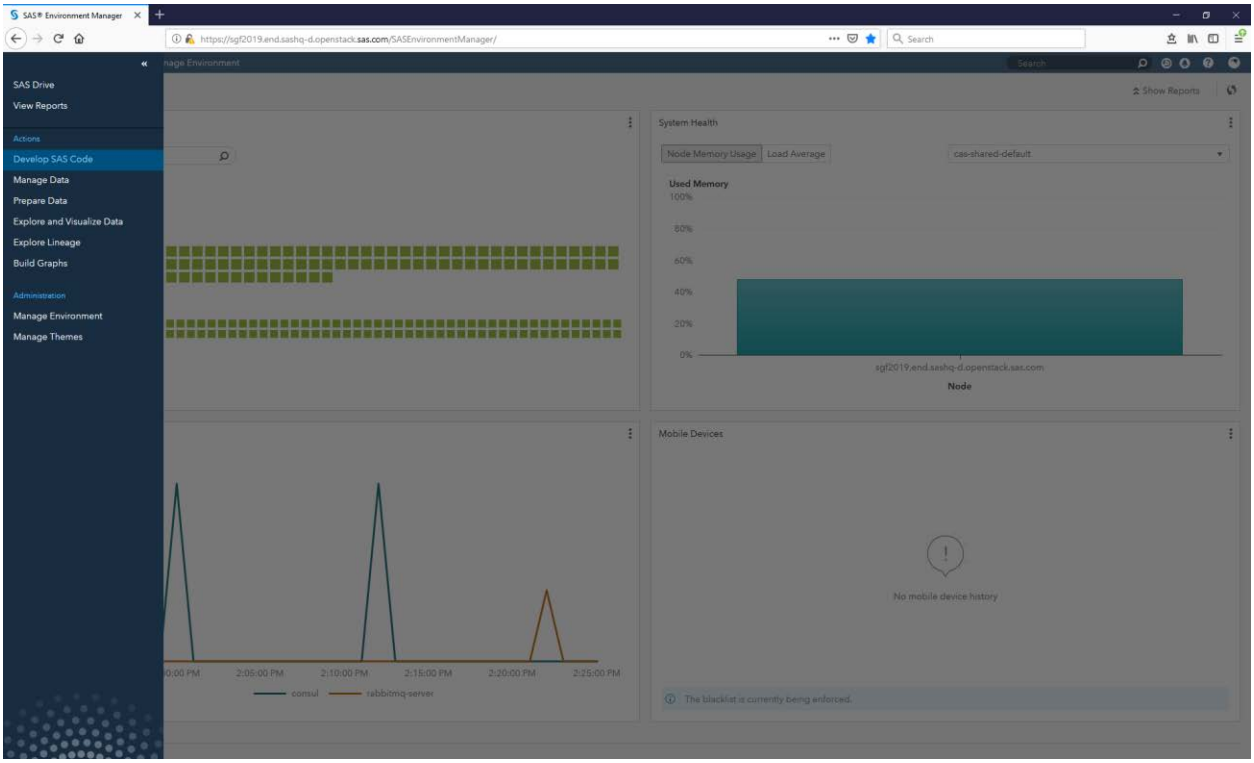
Success. The formats will remain available to all users until you remove the format library or alter the search path to not include then.

METHOD 2: SAS STUDIO

SAS Environment Manager is an easy way to add a set of formats to your SAS Viya environment. But sometimes the formats change frequently (maybe they are generated based on data values). Or you have a large number of format libraries you want to add. Or maybe you are just one of those people who hates user interfaces. A programmatic approach is often better suited in these cases.

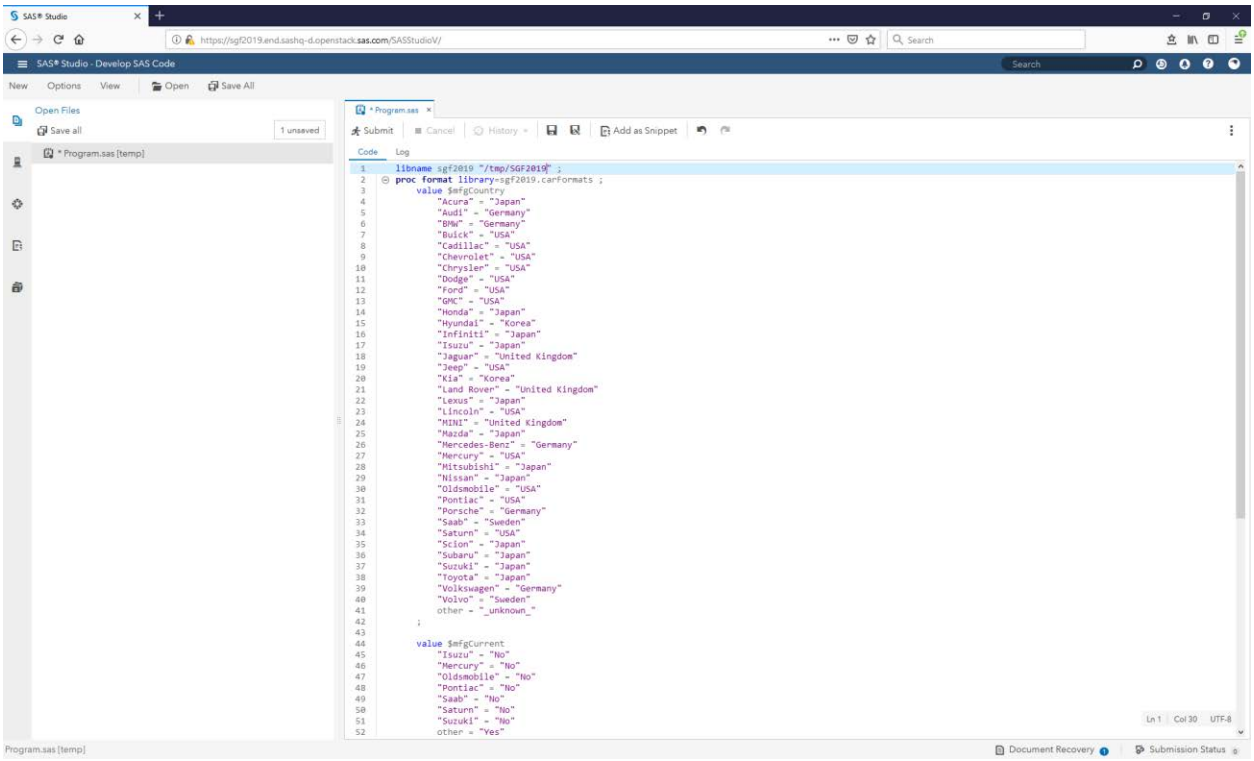
SAS Studio is the programming environment component of SAS Viya. Not only does it provide a familiar looking SAS development environment, but it can communicate with CAS as well. CAS is the underlying server for SAS Viya that needs to have access to the formats. So SAS Studio gives you a single place where you can run the SAS code to create the data and formats, and a programmatic way to make both available to SAS Visual Analytics on SAS Viya.

To invoke SAS Studio, choose Develop SAS Code.



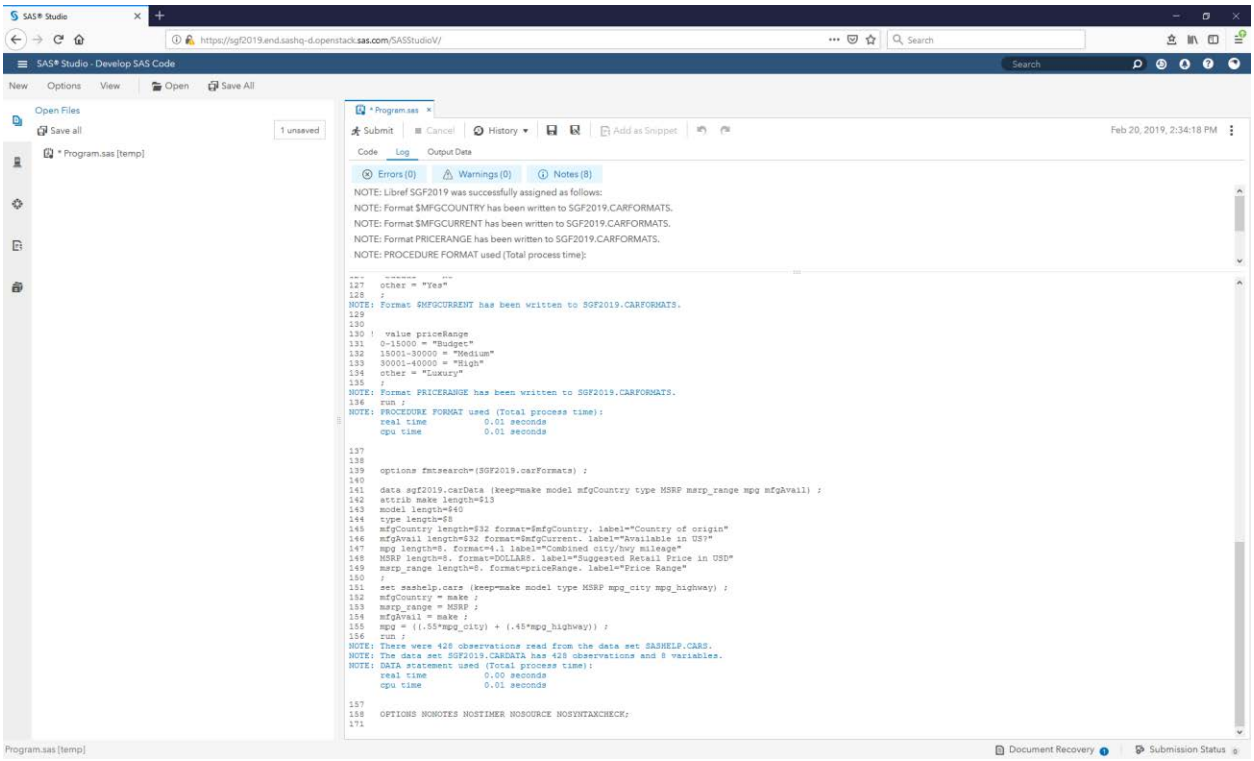
Display 12. Invoke SAS Studio

Initially you get a blank program area. Enter the same code used in the SAS Environment Manager section to create the formats and the data. You can enter the code in or just use cut and paste. Note that the example here is running on Linux rather than Windows. So, the example directory names are slightly different from those used in the SAS Environment Manager section.



Display 13. SAS Studio code to create formats and data

Choose Submit at the top of the program window to have SAS run the code. The log will be displayed, and the formats and data are created.



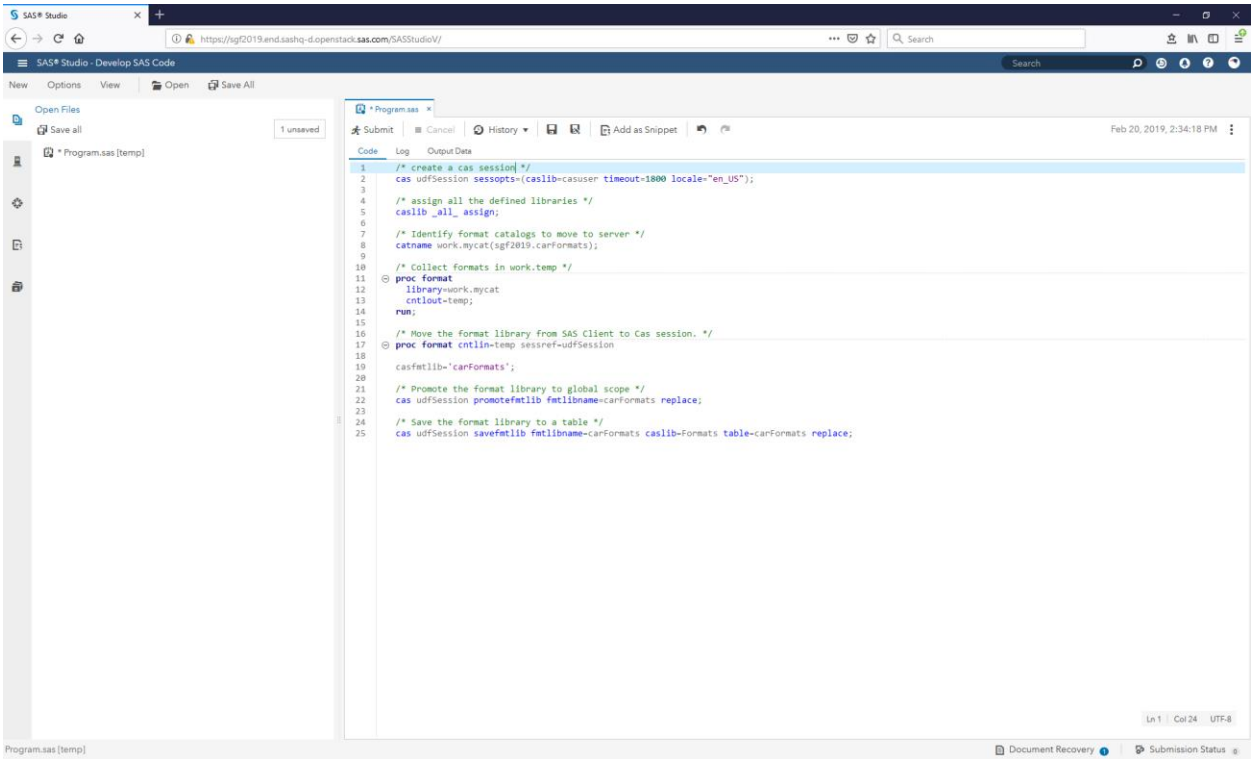
Display 14. Formats and Data created using SAS Studio

With the data and formats created, we need to make the formats available to SAS Viya. This is done by creating a CAS session in SAS Studio, and then using PROC FORMAT to load the formats into SAS Viya. Note that although I am treating these steps as different programs in order to explain the details, they can all be submitted at the same time as one program.

Back in the SAS Studio program window, submit the following code.

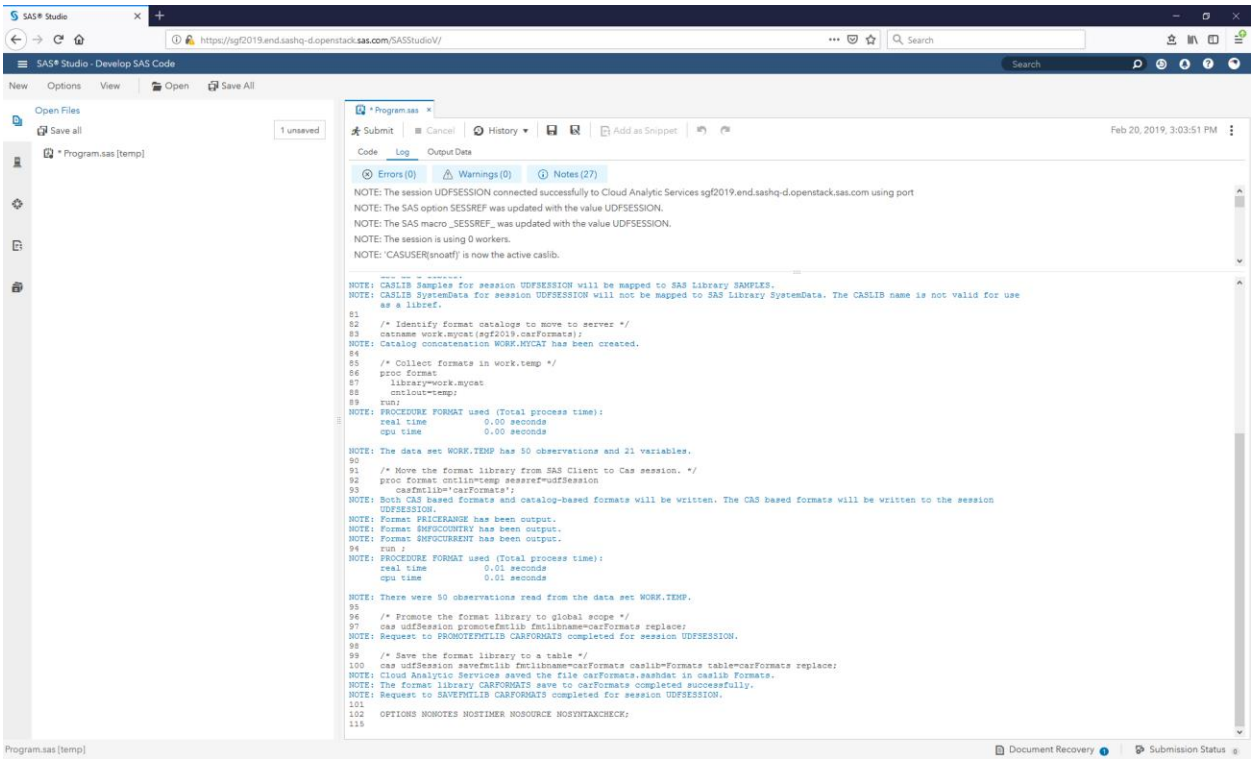
```
/* create a cas session */
cas udfSession sessopts=(caslib=casuser timeout=1800 locale="en_US");

/* assign all the cas libraries */
caslib _all_ assign;

/* Identify format catalogs to move to server */
catname work.mycat(sgf2019.carFormats) ;

/* Collect formats in work.temp */
proc format
    library=work.mycat
    cntlout=temp;
run;

/* Move the format library from SAS Client to Cas session. */
proc format cntlin=temp sessref=udfSession
    casfmtlib='carFormats';
run ;
```

```
/* Promote the format library to global scope */
cas udfSession promotefmtlib fmtlibname=carFormats replace;

/* Save the format library to a table */
cas udfSession savefmtlib fmtlibname=carFormats caslib=Formats
table=carformats replace;
```



Display 15. SAS Studio code to make the formats available

When the code completes, the log will contain information about what was done with the formats.



Display 16. Formats available to SAS Viya after submitting code in SAS Studio

Without a full explanation of what you can do in SAS Studio to interact with CAS, this code does a few things that you should understand.

First, a CAS session is created. This is necessary for any work you want to do with SAS Viya.

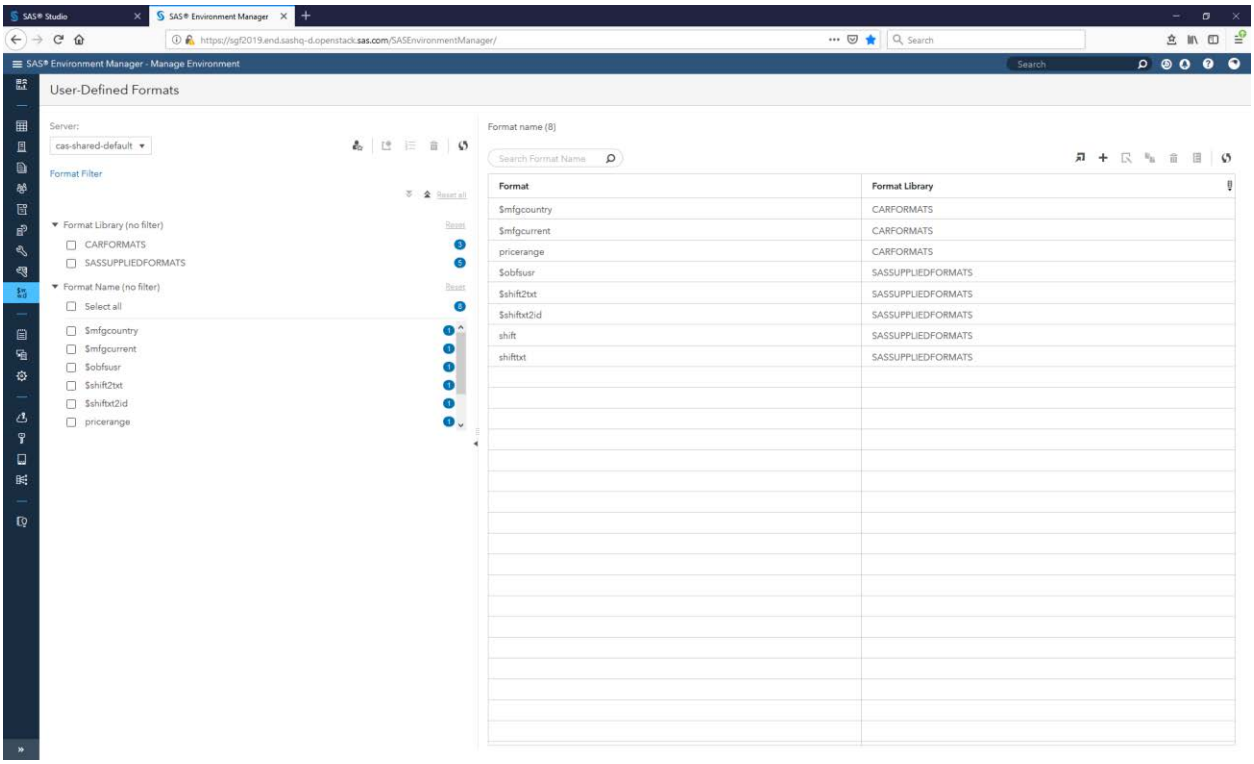
Next, all the libraries defined in the SAS Viya environment are mapped to SAS librefs. The one that we are primarily concerned with is the Formats library. This library is predefined for you in every SAS Visual Analytics environment and is used to store a backup copy of any formats that are loaded. This library will have an entry for the sassuppliedformats by default. After we complete the process to make our formats available to SAS Viya, it will also have an entry for carFormats (regardless of whether you use SAS Environment Manager or SAS Studio). The entries in this library are solely for backup purposes in the event the server goes down. They are not used during regular SAS Viya usage and can be deleted at any time if it is determined there is no need for a backup.

The next three statements take the existing formats from the SAS catalog and copy them to SAS Viya using PROC FORMAT. The casfmtlib name (in the example "carFormats") can be anything you want. It helps to make it meaningful and related to the formats it contains but you can follow whatever naming convention you wish.

Finally, the SAS Viya format library is promoted to global scope and the backup copy is made. SAS Viya allows format libraries to have either SESSION or GLOBAL scope. By default, when created, all format libraries have session scope. This means that the formats are available to all actions in the current session (remember we created a CAS session at

the start of this program?). When the session ends, the formats are no longer available. With global scope, the formats are available to all sessions for as long as the server is running. Since we want our formats to be available to anyone who tries to use the data, we need to make sure they have global scope. In CAS speak, this is done by promoting the formats from session to global scope. When loading the formats in either SAS Environment Manager or using the Command-Line Interface, this promotion to global scope is done for you.

After running this code, if you jump back over to SAS Environment Manager and choose User-Defined Formats, you will see that the format library carFormats is now defined and the formats it contains are available.



Display 17. Formats visible in SAS Environment Manager after loading using SAS Studio

If you are a SAS programmer, or typically create your data and formats using SAS programs, SAS Studio provides an environment for you to run your SAS code and also make the formats available to SAS Viya in a single program.

METHOD 3: SAS COMMAND-LINE INTERFACE

The final method described to load a format library to SAS Viya is arguably the easiest. The SAS Command-Line interface is included with SAS Viya and provides a direct process for calling the underlying services used by SAS Viya. No user interface, no programming environment, you just issue the commands directly at an operating system command. And

while it is possible using the command-line interface to create formats, in the example here we will only take an existing SAS 9 catalog of formats and load them.

This is especially convenient if you want to load or update your formats as part of an automated process. A script that contains the commands could even be scheduled or included as part of a work flow.

How easy is it? Specify the URL for your system, log on, and issue a single command to load the formats.

This example uses the same format catalog as the previous two. This example is being run on Linux but the command-line interface is available on most hosts. The command-line interface is typically installed in this directory:

- /opt/sas/viya/home/bin

Navigate to that directory. The first thing you must do is create a profile to define the system that you want to access. Then log on. Once you have defined a profile, you can use it in subsequent sessions. Your session will last for a set period, typically a few hours, depending on your system settings.

The commands to create a profile and log on are:

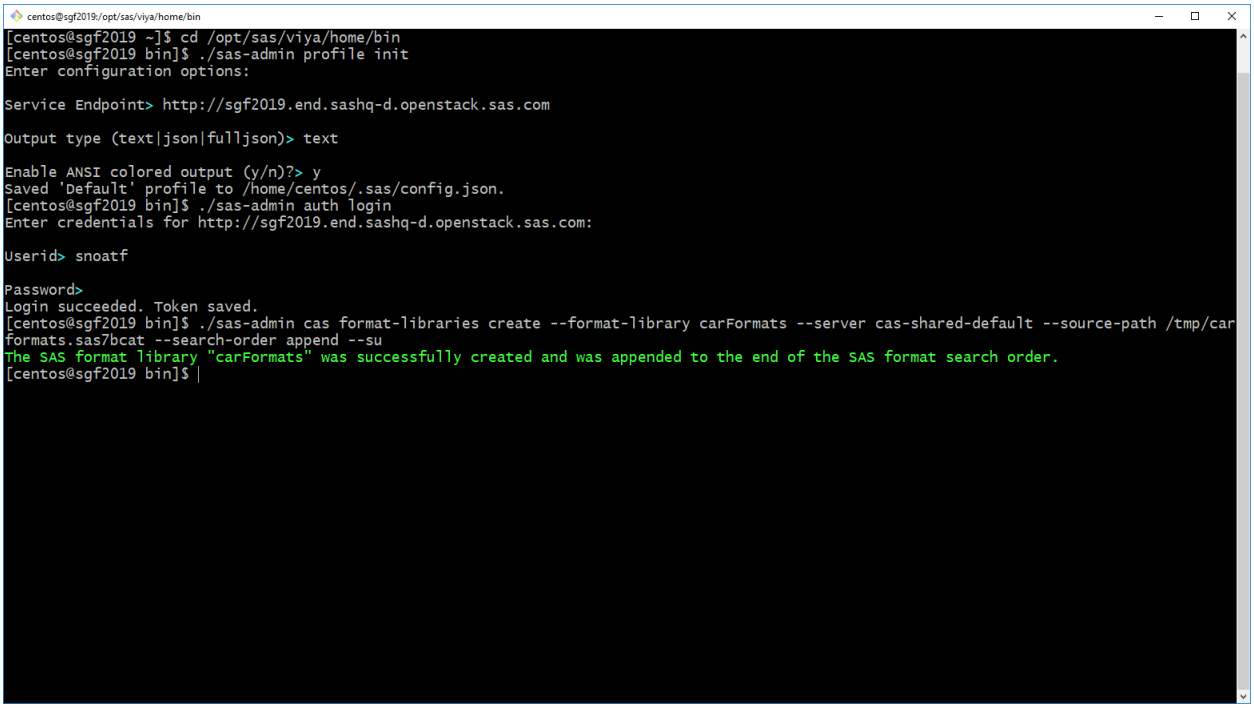
- ./sas-admin profile init
- ./sas-admin auth login

These commands will prompt you to supply the necessary values. The Service Endpoint is the fully qualified URL for your system, including “http” or “https”. The other values should be self-explanatory.

Once you have logged in successfully, a single command will take your format catalog as input and create a corresponding format library, also adding it to the existing search order:

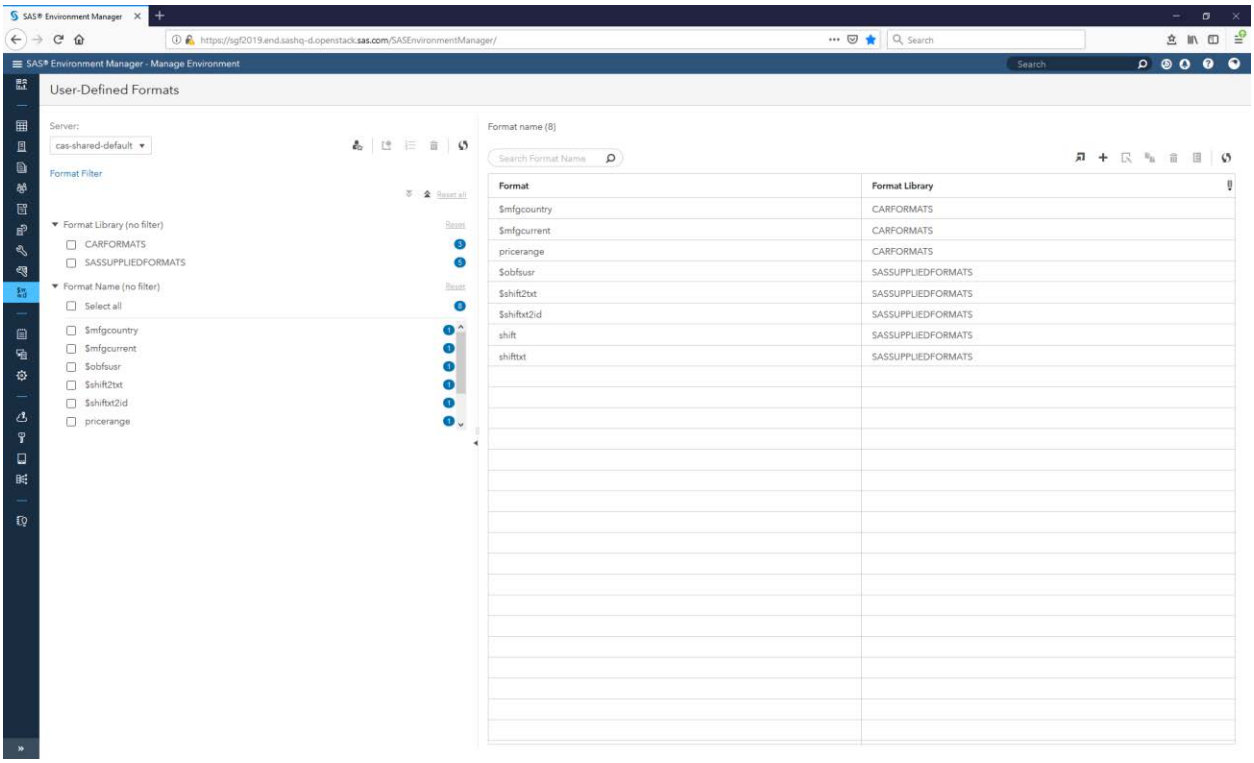
- ./sas-admin cas format-libraries create –format-library <formatLibraryName> --server <serverName> --source-path <formatCatalogLocationOnDisk> --search-order <formatLibrarySearchOrder> --su

Supply the values for this command and you will get a message telling you the format library has been created and where it has been placed in the search order. A sample session showing the expected responses follows.



Display 17. Command line interface to load formats

To verify the results, check in SAS Environment Manager and you will see your new format library defined.



Display 17. Formats visible in SAS Environment Manager after loading using command line interface

It really is that easy. These commands can easily be included as part of a scripted process to load new formats on demand. And since it can be scripted, even a very large volume of format libraries can be loaded quickly using this process.

If you are comfortable using OS level commands and have direct access to a machine where the SAS Command Line Interface is installed, this is most likely the approach you will choose.

CONCLUSION

There are a number of different ways to take formats created in SAS 9 and make them available to SAS Visual Analytics on SAS Viya. Each has advantages and limitations and which you choose will be based on your specific requirements. The variety of options available should cover the needs for any situation.

All SAS code used in this paper is included as snippets in the References section in a format that enables easy cut and paste.

Documentation for SAS Formats, SAS Viya Administration, SAS Environment Manager, SAS Studio, and the SAS Command-Line Interface are all available online.

REFERENCES

- *Code snippet to create the formats used in this paper*

```
libname sgf2019 "/tmp/sgf2019" ;
proc format library=sgf2019.carFormats ;
    value $mfgCountry
          "Acura" = "Japan"
          "Audi" = "Germany"
          "BMW" = "Germany"
          "Buick" = "USA"
          "Cadillac" = "USA"
          "Chevrolet" = "USA"
          "Chrysler" = "USA"
          "Dodge" = "USA"
          "Ford" = "USA"
          "GMC" = "USA"
          "Honda" = "Japan"
          "Hyundai" = "Korea"
          "Infiniti" = "Japan"
          "Isuzu" = "Japan"
          "Jaguar" = "United Kingdom"
          "Jeep" = "USA"
          "Kia" = "Korea"
          "Land Rover" = "United Kingdom"
          "Lexus" = "Japan"
          "Lincoln" = "USA"
          "MINI" = "United Kingdom"
          "Mazda" = "Japan"
          "Mercedes-Benz" = "Germany"
          "Mercury" = "USA"
          "Mitsubishi" = "Japan"
          "Nissan" = "Japan"
          "Oldsmobile" = "USA"
          "Pontiac" = "USA"
          "Porsche" = "Germany"
          "Saab" = "Sweden"
          "Saturn" = "USA"
          "Scion" = "Japan"
          "Subaru" = "Japan"
          "Suzuki" = "Japan"
          "Toyota" = "Japan"
          "Volkswagen" = "Germany"
          "Volvo" = "Sweden"
          other = "_unknown_"
    ;

    value $mfgCurrent
          "Isuzu" = "No"
          "Mercury" = "No"
          "Oldsmobile" = "No"
          "Pontiac" = "No"
          "Saab" = "No"
          "Saturn" = "No"
```

```
          "Suzuki" = "No"
          other = "Yes"
    ;

    value priceRange
          0-15000 = "Budget"
          15001-30000 = "Medium"
          30001-40000 = "High"
          other = "Luxury"
    ;
run ;
```

- *Code snippet to create the data used in this paper*

```
options fmtsearch=(SGF2019.carFormats) ;

data sgf2019.carData (keep=make model mfgCountry type MSRP msrp_range mpg
mfgAvail) ;
    attrib make length=$13
    model length=$40
    type length=$8
    mfgCountry length=$32 format=$mfgCountry. label="Country of origin"
    mfgAvail length=$32 format=$mfgCurrent. label="Available in US?"
    mpg length=8. format=4.1 label="Combined city/hwy mileage"
    MSRP length=8. format=DOLLAR8. label="Suggested Retail Price in USD"
    msrp_range length=8. format=priceRange. label="Price Range"
    ;
    set sashelp.cars (keep=make model type MSRP mpg_city mpg_highway) ;
    mfgCountry = make ;
    msrp_range = MSRP ;
    mfgAvail = make ;
    mpg = ((.55*mpg_city) + (.45*mpg_highway)) ;
run ;
```

- *Code snippet to load the formats using SAS Studio*

```
/* create a cas session */
cas udfSession sessopts=(caslib=casuser timeout=1800 locale="en_US");

/* assign all the cas libraries */
caslib _all_ assign;

/* Identify format catalogs to move to server */
catname work.mycat(sgf2019.carFormats) ;

/* Collect formats in work.temp */
proc format
    library=work.mycat
    cntlout=temp;
run;
```

```
/* Move the format library from SAS Client to Cas session. */
proc format cntlin=temp sessref=udfSession
    casfmtlib='carFormats';
run ;

/* Promote the format library to global scope */
cas udfSession promotefmtlib fmtlibname=carFormats replace;

/* Save the format library to a table */
cas udfSession savefmtlib fmtlibname=carFormats caslib=Formats
table=carformats replace;
```

- *Code snippet to load the formats using the command line interface*

```
cd /opt/sas/viya/home/bin
./sas-admin profile init
./sas-admin auth login
./sas-admin cas format-libraries create --format-library carFormats --
server cas-shared-default –-source-path /tmp/carformats.sas7bvat –-search-
order append –su
```

ACKNOWLEDGMENTS

Rick Langston and Denise Poll of SAS deserve the credit for SAS Formats, the FORMAT Procedure, and the process for making formats available in SAS Viya.

Additional work, including making the command-line interface available, was done by Jeff Schrilla and James Holman, also from SAS.

Mark Fulp of SAS implemented the User-Defined Formats section of SAS Environment Manager.

CONTACT INFORMATION

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

Andrew Fagan
SAS
Andrew.Fagan@sas.com