An Introduction to SAS® Viya® 3.4 Programming

What's Included in SAS Viya?

This software supports analytical data preparation, variable transformations, exploratory analysis, analytical modeling, integrated model comparison, and scoring. Here are the main components:

SAS Viya	The third generation of high-performance in-memory analytics.
SAS Studio	The integrated programming environment.
SAS Cloud Analytic Services (CAS)	The analytic engine. CAS uses high-performance, multithreaded analytic code to rapidly process requests against data of any size.
SAS Visual Analytics	Programming tools that provide baseline functionality, including reporting and basic analytics. The following functionality is provided: ▪ analytical data preparation ▪ variable transformations ▪ exploratory analysis ▪ descriptive statistics
SAS Visual Statistics	An additional set of advanced analytic functionality that builds on SAS Visual Analytics, providing the following capabilities: ▪ ability to build predictive models ▪ integrated model comparison
SAS Visual Data Mining and Machine Learning	An additional set of advanced analytic functionality that builds on SAS Visual Statistics and that enables you to do the following: ▪ tune machine learning algorithm hyperparameters ▪ perform advanced statistical operations ▪ analyze complex data
SAS Econometrics	A set of functionality that provides techniques to model complex business and economic scenarios and to analyze the dynamic impact that specific events might have over time.

SAS Visual Forecasting	Provides automatic variable, event, and model selection. It then automatically generates your forecasts.
SAS Visual Text Analytics	A text analytics framework that combines text mining, contextual extraction, categorization, sentiment analysis, and search capabilities.
SAS Optimization	A set of procedures for exploring models of distribution networks, production systems, resource allocation problems, and scheduling problems, using the tools of operations research.

Servers and Sessions

SAS Cloud Analytic Services

The high-performance processing power of SAS Viya is SAS Cloud Analytic Services (CAS). CAS is a server that provides the run-time environment for data management and analytics with SAS. (Run-time environment refers to the combination of hardware and software where data management and analytics take place.)

Note: Your site must license and install one or more of the SAS Viya products to use this functionality.

The CAS server has the following characteristics:

- Data being transmitted to and from the CAS server and stored data is secure. See these publications:
  - Encryption in SAS Viya: Data in Motion

    Note: If you are connecting the CAS server from SAS 9.4, see Configure SAS 9.4 Clients to Work with SAS Viya in this publication.
  - Encryption in SAS Viya: Data at Rest
- Authentication is used to control access to the CAS server and its resources. Your identity must be successfully authenticated before your session is created.

  For SAS Studio and for instances where SAS Enterprise Guide uses the SAS Workspace Server, your user credentials authenticate you to the CAS server. If your SAS Enterprise Guide server is local to your computer and is not a remote server, an attempt is made to find an authinfo file (.authinfo is the default filename on UNIX, _authinfo on Windows). The authinfo file provides a user name and password to CAS for operating system authentication. For more information, see Authentication Mechanisms and *Client Authentication Using an Authinfo File*.

  The SAS windowing environment requires an authinfo file for authentication if your site does not use kerberos for authentication. If authentication to the CAS server fails, see *Client Authentication Using an Authinfo File* or contact your site administrator.
- The CAS server session encoding is UTF-8.
- The server can run on a single machine or as a distributed server on multiple machines. For both architectures, the server is multi-threaded for high-performance analytics.
  - The distributed server consists of one controller and one or more workers. This architecture is often referred to as a massively parallel processing (MPP) architecture.

    Using single-server symmetric multi-processing (SMP), the threaded processing of the CAS server is shared by multiple CPUs or is shared between multiple cores of a single CPU.

    Note: On Windows, the CAS server uses SMP processing.

  - The distributed server has a communication layer that supports fault tolerance. A distributed server can continue to process requests even after losing connectivity to some nodes. The communication layer also enables you to remove or add worker nodes from a server while it is running.
  - If you have a distributed server, one host can be designated as a backup controller. If the primary controller fails, client connections automatically fail over to the backup controller. New sessions can be started by connecting to the backup controller.
  - The CAS language (CASL) enables you to run CAS actions and operate on the results of an action. Using CASL, you can build an analytic pipeline. From SAS, you can use CASL with the CAS procedure. Requests and responses are exchanged between SAS and the server. You can also use CASL to define your own user-defined action set. A user-defined action can enhance processing by performing the analytic processing on the server and reduce the number of messages exchanged between SAS and the server.
  - Deep learning CAS actions can take advantage of Nvidia GPUs in both SMP and MPP environments.
- The CAS server processes in-memory tables. The source data for the in-memory tables can come from SAS data sets, server-side files, event stream processing, and database tables. Database tables can be loaded serially or in parallel.
- The server can manage all of your data and easily share data with multiple users.
- You can program using new high-performance SAS Viya procedures, most SAS 9.4 procedures and language elements, or CAS actions. CAS actions are the smallest unit of work for the CAS server. To program using CAS actions, you use the CAS procedure.
- The DATA step, DS2, and FedSQL languages can run in the CAS server.
- The CAS server organizes user-defined formats in format libraries. You can use the FORMAT procedure to specify a CAS format library to store a new user-defined format. Because the CAS server does not support SAS catalogs, you can use PROC FMTC2ITM to move user-defined formats that are stored in SAS format catalogs to a CAS format library.
- The programming interfaces include all programming interfaces that you use with SAS as well as the open-source languages Python, Lua, Java, and R.

See Also

- For an overview of SAS Viya and architectural diagrams, see SAS® Viya 3.4: Overview
- *SAS Cloud Analytic Services: Fundamentals*
- SAS Viya 3.4 Administration

SAS Workspace Server and SAS Compute Server

SAS Viya provides two programming run-time servers for processing data that is not performed by the CAS server. Which server is used is determined by your SAS environment. When your SAS environment includes the SAS Viya visual and programming environments, your SAS administrator determines the server. The SAS Workspace Server and the SAS Compute Server support the same SAS code and produce the same results. The following table shows which server can be used based on the SAS environment and the SAS programming client:

SAS Environment	SAS Viya 3.4		SAS 9.4M5 or SAS 9.4M6		
	SAS Studio 4.4	**SAS Studio 5.1**	**SAS Studio 3.71** **SAS Studio 3.8**	**SAS Enterprise Guide**	**SAS Windowing Environment**
SAS 9.4	Not Applicable	Not Applicable	SAS Workspace Server	SAS Workspace Server	SAS Workspace Server
SAS Viya visual and programming environments	SAS Workspace Server	SAS Compute Server	Not Applicable	Not Applicable	Not Applicable
SAS Viya programming-only environment	SAS Workspace Server	Not Applicable	Not Applicable	Not Applicable	Not Applicable

The SAS Compute Server uses the SAS Viya Compute service, which manages server sessions using SAS Viya microservices.

Note: The SAS Compute Server does not support the X command, which enables execution of operating system commands from within SAS.

The DATA step runs on the SAS Workspace Server, the SAS Compute Server, as well as on the CAS server. If it is determined that the DATA step cannot run on the CAS server, the DATA step runs on the SAS Workspace Server or the SAS Compute Server.

The data management and utility procedures and macros run on the SAS Workspace Server or the SAS Compute Server. SAS procedures that do not run on the CAS server can transfer data from in-memory tables on the CAS server to the workspace server or compute server and then run the procedure.

FedSQL and DS2 run on the SAS Workspace Server or the SAS Compute Server when they access SAS libraries. They run on the CAS server when a CAS LIBNAME engine libref is used to access in-memory tables.

The SAS Workspace Server and the SAS Compute Server support SAS catalogs.

High-performance SAS Viya procedures and other supporting procedures use a CAS LIBNAME engine libref to identify in-memory tables and then use CAS actions to perform data analysis and processing on the CAS server. The SAS Workspace Server or the SAS Compute Server do not perform data analysis for high-performance SAS Viya procedures.

File access using SAS Studio is different on the two servers:

- SAS Studio 4.4 can access files on the server and the network that the user has permission to access.
- SAS Studio 5.1 access to files is restricted to only SAS files using the SAS Viya Files Service. For more information, see "SAS Viya Files Service" on page 43.

Connecting to the CAS Server and Creating a CAS Session

Connection with the CAS server is automatically created when you use SAS Studio 4.4 on Linux or SAS Studio 5.1 on Linux and Windows, which are the web programming interfaces for SAS Viya. The connection information, the host name and port, are set during installation. You can view those values in the CASHOST and CASPORT system options using this code:

```
proc options group=cas;
run;
```

After logging on to the client, you submit the CAS statement to create a session. This CAS statement starts the session named Casauto:

```
cas casauto;
```

If your CAS client is SAS 9.4M5 or later, or SAS Enterprise Guide, you must explicitly connect to the CAS server and start a session:

```
options cashost="cloud.example.com" casport=5570;
cas casauto;
```

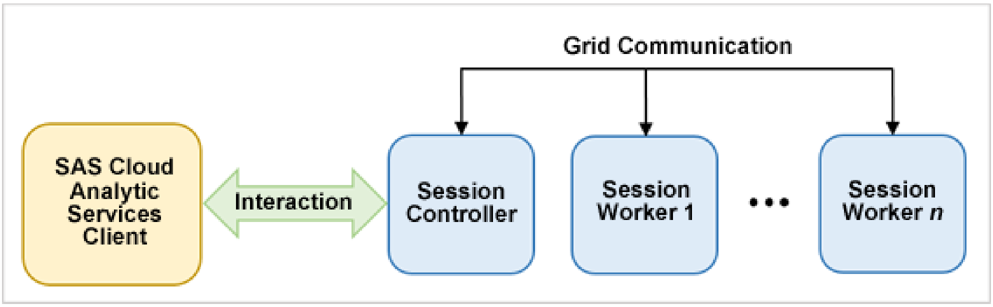
When you create a CAS session, the server completes the following process:

1 The server authenticates your identity.

2 The server listens for a session request.

3 The server starts a session between the client process and the session process, as shown in Figure 1.

Figure 1 Session Processes in a Distributed System



Sessions provide the following functionality:

- user identification.
- fault isolation for each session. If a problem occurs in your session, it does not impact other sessions or the server. The client might be affected if the session that it connects to has a problem.
- efficiency. Resources that are visible only to your session (session-scoped tables and caslibs) do not require checks for concurrent access, and they run faster.
- resource tracking, which is enabled through the use of resource metrics.

You can add the CAS statement to the autoexec.sas file so that your CAS session starts when you start SAS or SAS Studio.

In the SAS windowing environment, you can use the AUTHINFO= option in the CAS statement to provide the connection information. For more information about the AUTHINFO= option, see “AUTHINFO="authentication-info-file"” in *SAS Cloud Analytic Services: User’s Guide*.

You can view sessions by submitting the `CAS _ALL_ LIST;` statement.

See Also

- Editing the Autoexec File using SAS Studio 4.4
- Editing the Autoexec File using SAS Studio 5.1
- “Sessions” in *SAS Cloud Analytic Services: Fundamentals*
- “CAS Statement” in *SAS Cloud Analytic Services: User’s Guide*
- Working with Code Snippets using SAS Studio 4.4
- Working with Code Snippets using SAS Studio 5.1

Batch and Interactive Line-Mode Processing

You can submit code in batch mode and interactive line mode from the command line as you did in previous releases of SAS, with these differences:

- Authentication to CAS is accomplished either with Kerberos or with an authinfo file.
- Only Zip, SFTP, and URL FILENAME access methods are available in SAS Viya.
- You set the CASHOST= and CASPORT= system options in the SAS command line, a configuration file, or in the autoexec file. For example:

  ```
  -cashost "cloud.myorg.com" -casport 5570
  ```

- The default path to the SAS Viya install directory on UNIX is **`/opt/sas/spre`**.

  The default path to the SAS Viya install directory on Windows is **`c:\Program Files\SAS\Viya`**.

> **TIP** You can submit batch programs from SAS Studio 4.4 using the background submit feature.

See Also

- "Batch Mode in UNIX Environments" in *SAS Companion for UNIX Environments*
- "Interactive Line Mode in UNIX Environments" in *SAS Companion for UNIX Environments*
- "Connecting to the CAS Server" in *SAS Companion for UNIX Environments*
- Using the Background Submit Feature

Open-Source Code for Python, Lua, R, and Java

SAS open-source code provides an interface to the CAS server using CAS actions from Python, Lua, R, and Java. A CAS action is the smallest unit of work in the server. Related CAS actions are grouped in an action set. An example of an action is the table.loadtable action that loads an in-memory table from a server-side file.

The following versions of the scripting languages are compatible with the SAS interfaces:

- Python 2.7.x or Python 3.4.x
- Lua 5.2 or Lua 5.3
- Oracle JDK SE Java 8
- OpenJDK version 1.8.0.65
- R 3.1.0 or later.
- Install the httr and jsonlite packages. These packages have additional dependencies that are automatically installed from CRAN when you run install.packages().

The client machine that runs Python, Lua, R, or Java applications must use the following:

For Python, R, and Lua, you use the SAS Scripting Wrapper for Analytics Transfer (SWAT) interface to connect to the CAS server. For Java, you use the SAS Java Interface for SAS Viya to connect to the CAS server. You can download the client interfaces from the following locations

Python	http://support.sas.com/downloads/package.htm?pid=1977 https://sassoftware.github.io/python-swat/install.html
Lua	http://support.sas.com/downloads/package.htm?pid=1975

Java	http://support.sas.com/downloads/package.htm?pid=1976
R	https://support.sas.com/downloads/package.htm?pid=2061 https://github.com/sassoftware/R-swat

This Python example connects to the CAS server, starts a session, and executes the listSessOpts action to list the name, type, and current value for all of the session options.

```
import swat

# Create session CASAUTO. Specify the CAS host and port for your site.
s = swat.CAS("cloud.example.com", 5570)

# Get the current session options.
res = s.sessionProp.listSessOpts()

# Print the option name and value for each option.
print "Current session options:"
display(res.SessOpts.ix[:,['Name','Type','Value']])
```

Many of the action examples in the Help Center show examples in the new CAS language (CASL), Lua, R, and Python. For example, the Aggregation Action Set examples provide CASL, Lua, R, and Python links below the navigation buttons. You can find additional examples in the SAS Viya community.

The REST APIs are available at developer.sas.com.

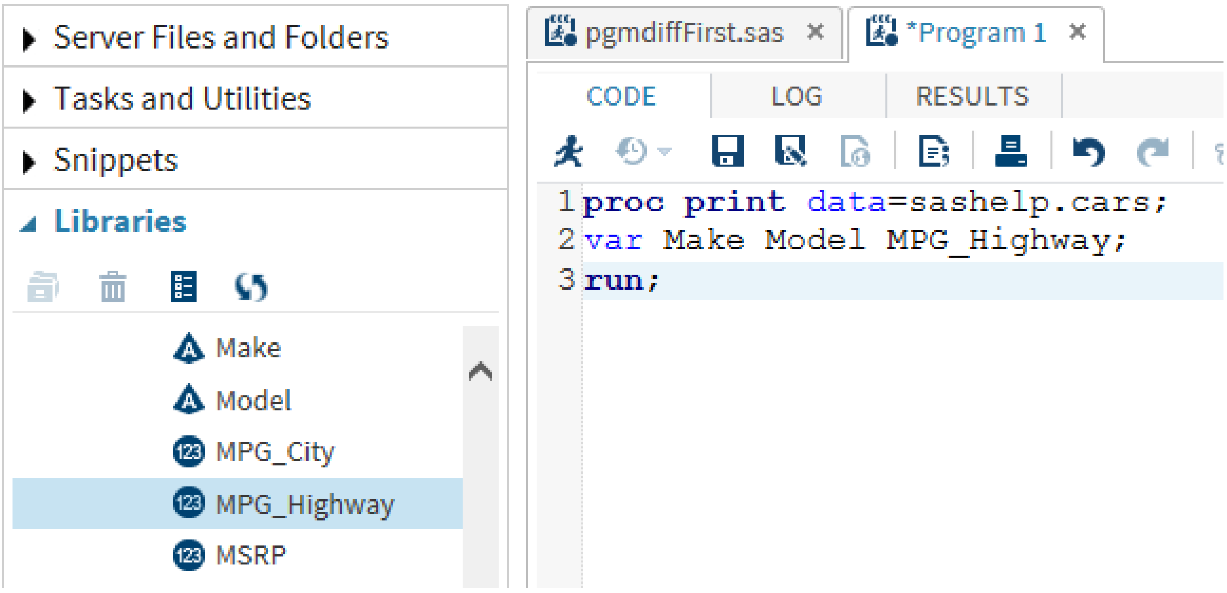
SAS Studio

SAS Studio is a browser-based SAS coding interface. It is the only graphical editor that is installed with a SAS Viya deployment. SAS Viya 3.4 and later include two versions of SAS Studio, SAS Studio 4.4 and SAS Studio 5.1. Here are some important differences:

- The SAS server is different. For more information, see "SAS Workspace Server and SAS Compute Server" on page 3.
- SAS Studio 5.1 has an improved user interface, including different tasks.
- File access is different. For more information, see "SAS Workspace Server and SAS Compute Server" on page 3.

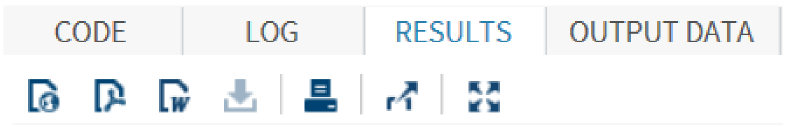
SAS Studio is designed to help you write your SAS programs as quickly and as accurately as possible. You can program using the point-and-click interface, or you can enter your code directly on the **CODE** tab. The following instructions are for both SAS Studio 4.4 and SAS Studio 5.1, unless information has been added specifically for one of those releases.

- Click **Libraries** (SAS Studio 4.4) or (SAS Studio 5.1) in the Navigation pane to access all of your libraries and the tables in those libraries.
- Save time when you are writing a program by dragging items from the **Libraries** section to your program. SAS Studio adds the code for those items to your program. In the following figure, the variables Make, Model, and MPG_highway were dragged to the program on the **CODE** tab:



```
proc print data=sashelp.cars;
var Make Model MPG_Highway;
run;
```

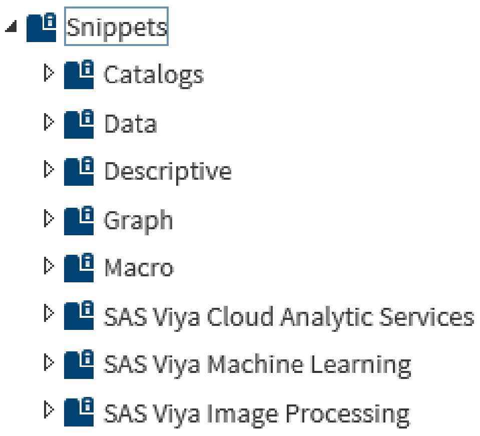
- Each time that you submit code, the **LOG** tab displays messages that indicate whether the submission was successful. Table output from your program appears on the **RESULTS** tab. By default, SAS Studio creates HTML5 output. In the SAS Studio 4.4 **RESULTS** tab, you can click one of the first three buttons to download the results in HTML5, PDF, or RTF formats:

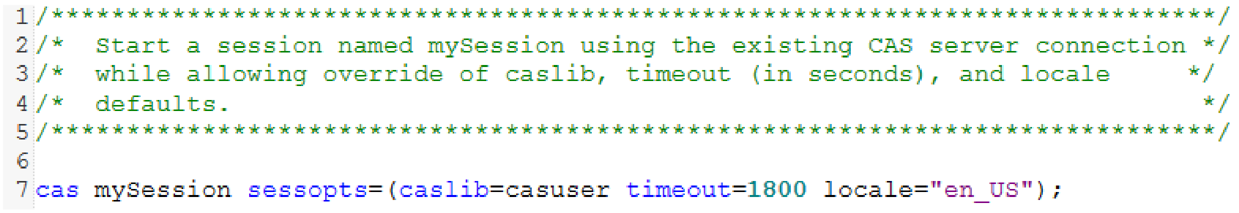
  To download the results in other output formats in SAS Studio 5.1, click ⋮ and select **Download**.

To assist you in coding, SAS Studio provides snippets and tasks that automatically generate code when you drag them to the **CODE** tab. A snippet is code that you might use frequently. The following example explains how to use snippets in your programming.

1 In the Navigation pane, expand **Snippets** (SAS Studio 4.4) or select and **SAS Snippets** (SAS Studio 5.1) for a list of code snippets for use with SAS Viya Cloud Analytic Services, SAS Viya Machine Learning, and SAS Viya Image Processing. This is the list of snippets in SAS Studio 4.4:

  
  Snippets
  Catalogs
  Data
  Descriptive
  Graph
  Macro
  SAS Viya Cloud Analytic Services
  SAS Viya Machine Learning
  SAS Viya Image Processing

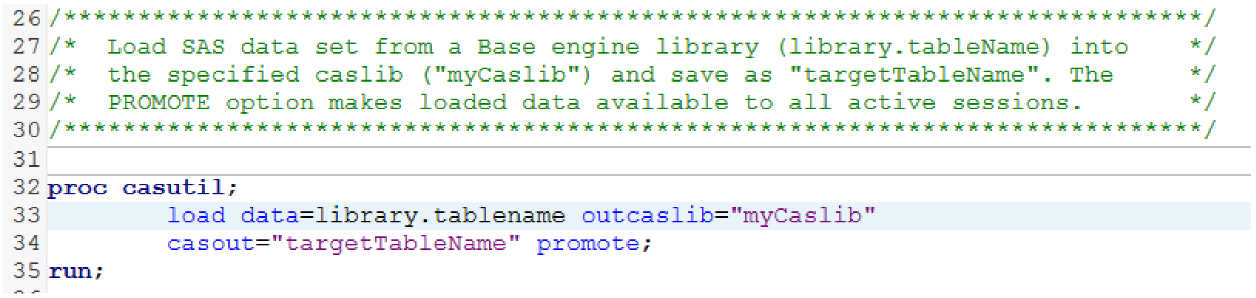
2 Expand **SAS Viya Cloud Analytic Services**, drag the **New CAS Session** snippet to the **CODE** tab, and submit the code. A CAS session named mySession that uses the initially active caslib casuser is established.



```
/***************************************************************************/
/*  Start a session named mySession using the existing CAS server connection */
/*  while allowing override of caslib, timeout (in seconds), and locale     */
/*  defaults.                                                               */
/***************************************************************************/

cas mySession sessopts=(caslib=casuser timeout=1800 locale="en_US");
```

3 Drag the **Load data to caslib** snippet to the **CODE** tab. Delete the code above and below the PROC CASUTIL code that loads a SAS data set from a Base engine library.



```
/**************************************************************************/
/*  Load SAS data set from a Base engine library (library.tableName) into   */
/*  the specified caslib ("myCaslib") and save as "targetTableName". The    */
/*  PROMOTE option makes loaded data available to all active sessions.      */
/**************************************************************************/

proc casutil;
        load data=library.tablename outcaslib="myCaslib"
        casout="targetTableName" promote;
run;
```

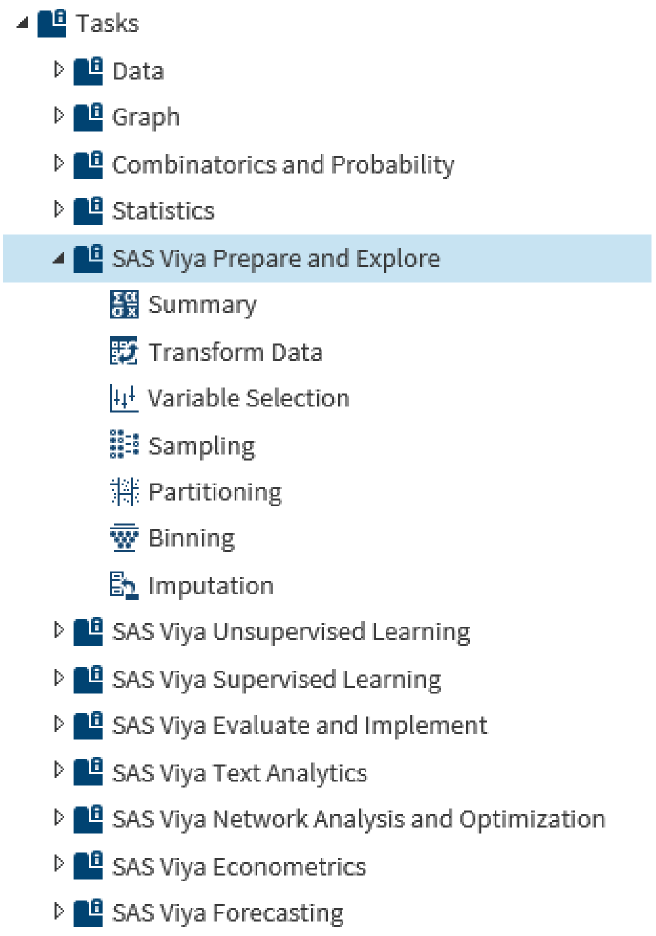
4 Customize the code:

- Replace `library.tablename` with `sashelp.cars`.
- Replace `myCaslib` with `casuser`.
- Replace `targetTableName` with `cars`.

5 Submit the code. SAS runs the CASUTIL procedure code to perform the client-side load of the Cars data set from the Sashelp library to the Cars table in the Casuser caslib.

> **TIP** This technique is appropriate for small data sets and convenient for a getting-started experience. More efficient techniques are available for larger data sets.

6 Now that you have data in a caslib, you can use the **Machine Learning** snippets or tasks to explore, model, and score data. To use the tasks, expand **Tasks** (SAS Studio 4.4) or select and **SAS Tasks** (SAS Studio 5.1) to see the types of tasks that assist you by generating code. Here are some of the SAS Studio 4.4 tasks:


Tasks
Data
Graph
Combinatorics and Probability
Statistics
SAS Viya Prepare and Explore
Summary
Transform Data
Variable Selection
Sampling
Partitioning
Binning
Imputation
SAS Viya Unsupervised Learning
SAS Viya Supervised Learning
SAS Viya Evaluate and Implement
SAS Viya Text Analytics
SAS Viya Network Analysis and Optimization
SAS Viya Econometrics
SAS Viya Forecasting

See Also

- Getting Started with Programming in SAS Studio 4.4
- Getting Started with Programming in SAS Studio 5.1
- SAS Studio 4.4: User’s Guide
- SAS Studio 5.1: User’s Guide
- “Caslibs” on page 12

Data Migration to UTF-8 Encoding

The CAS server supports only UTF-8 encoding. UTF-8 is variable-width, multi-byte encoding. One UTF-8 character can be 1–4 bytes in length. If your SAS or SAS Viya session encoding is not UTF-8, then the data that is read to a CAS table must be transcoded to UTF-8. You can set your SAS or SAS Viya session to UTF-8 in order to avoid the transcoding of data. If you want to change your session encoding, see SAS Viya 3.4 Administration: General Servers and Services.

If your SAS or SAS Viya session is not UTF-8, the data must be transcoded to UTF-8 when it is loaded to a caslib. When double-byte character set (DBCS) characters and some single-byte character set (SBCS) characters are transcoded to UTF-8, they require additional bytes to represent a character.

CAUTION! Data truncation can occur during transcoding. If a data set column or a database column is not wide enough to store the results that are encoded as UTF-8 when data transcoding occurs, then data might be truncated. To avoid truncation, specify the appropriate option for the CAS LIBNAME engine, the CASUTIL procedure LOAD statement, or CAS data connectors. If a variable width is associated with a format, the format width is not increased. You must re-create the format using the FORMAT procedure. If there is the possibility of truncated data, an error message appears in the log.

When the SAS client encoding is not UTF-8 and the data must be transcoded for the CAS server, the client estimates the number of bytes that are needed to transcode the data to UTF-8:

- For the CAS LIBNAME engine and the CASUTIL procedure LOAD statement, environments that use an SBCS encoding estimate 1 byte in UTF-8 for every 1 byte in the local encoding. Environments that use a DBCS encoding estimate 1.5 bytes in UTF-8 for every 1 byte in the local encoding.
- For CAS data connectors, all environments estimate 3 bytes in UTF-8 for every 1 byte in the local encoding.

The CAS LIBNAME statement and the CAS data connectors provide options for you to replace the estimates for the number of bytes that are needed to transcode data from SBCS and DBCS environments to UTF-8:

- For CAS data connectors, use the CHARMULTIPLIER= option to modify the multiplier when the data originates from SBCS environments.
- Use the NCHARMULTIPLIER= option to modify the multiplier in these instances:
  - when you use the CAS LIBNAME engine and the data originates from SBCS or DBCS environments
  - when you use CAS data connectors and the data originates from DBCS environments

These options apply to the CAS LIBNAME engine:

- NCHARMULTIPLIER= CAS engine LIBNAME option
- NCHARMULTIPLIER= data set option
- CASNCHARMULTIPLIER= system option can specify a default value for CAS LIBNAME engine transcoding

  Note: The NCHARMULTIPLIER= option overrides the value of the CASNCHARMULTIPLIER= system option. CAS data connectors do not recognize the value of the CASNCHARMULTIPLIER= system option.

The NCHARMULTIPLIER= LOAD statement option is available for the CASUTIL procedure. PROC CASUTIL recognizes the CASNCHARMULTIPLIER= system option if you set a default value using that system option.

The CHARMULTIPLIER= and the NCHARMULTIPLIER= data connector options are available for all CAS data connectors with these exceptions:

- Transcoding of Hive data using SAS Data Connector to Spark and SAS Data Connector to Hadoop can be performed only by using the NCHARMULTIPLIER= option.
- Transcoding of data using SAS Data Connector to SPD Engine Files can be performed only by using the CHARMULTIPLIER= option.

A best practice is to perform a test reading of data by the CAS engine before the data is used in a production environment.

Because ASCII encoding is compatible with UTF-8 encoding, no transcoding is necessary.

Indexes and format catalogs must also be compatible with UTF-8 session encoding.

For additional information about transcoding SAS data sets to UTF-8, see these publications:

- *Migrating Data to UTF-8 for SAS Viya*
- SAS Viya FAQ for Processing UTF-8 Data

See Also

"Avoiding Character Data Truncation By Using the CVP Engine" in *SAS National Language Support (NLS): Reference Guide*

Caslibs

All data is available to the CAS server through caslibs and all operations in the CAS server that use data are performed using caslibs. A caslib provides access to files in a data source, such as a database or a file system directory, and to in-memory tables. Access controls are associated with caslibs to manage access to data. You can think of a caslib as a container where the container has two areas where data is referenced: a physical space that includes the source data or files, and an in-memory space that makes the data available for CAS action processing.

Caslibs can be of the following types:

- personal and automatically available for a given user (analogous to the Sasuser library, but in-memory)
- pre-defined by an administrator
- manually added

Authorized users can add or manage caslibs with the CASLIB statement. Caslib authorization is set by your administrator. In some instances, such as when you copy native CAS tables that are not in-memory, a caslib is required although data is not copied to the caslib.

When you start a CAS session, your personal caslib is added by default. It is the active caslib and is named Casuser(*my-user-ID*). When you specify the caslib in your code, you use only the name Casuser, you do not need to specify Casuser(*my-user-ID*). Your personal caslib is available only to you. To share tables in your caslib, make your table available in another caslib that is accessible by other users. Your site administrator can add caslibs with global scope for anyone to use.

The following output lists the metadata that is associated with a personal caslib. The code to obtain this information is `caslib casuser list;`.

```
57         caslib casuser list;
NOTE: Session = MYSESSION Name = CASUSER(my-user-ID)
         Type = PATH
         Description = Personal File System Caslib
         Path = /home/myuserID/
         Definition =
         Subdirs = Yes
         Local = No
         Active = Yes
         Personal = Yes
```

This caslib is a path-based caslib that enables the user specified by *my-user-ID* to save in-memory tables as files in the specified path and to perform a server-side load of data from the path.

You use the CASLIB statement to add a new caslib if you are authorized to do so. Only one caslib can be active in a session at a time. When you use the CASLIB statement, that caslib becomes the active caslib. If an additional caslib is added, that one becomes the active caslib. If that second one is dropped, the session returns to use the initially active caslib.

The active caslib is used as the default data source if you do not override it. You can override the active caslib using the CASUTIL procedure INCASLIB and OUTCASLIB= options or the CASLIB= LIBNAME option.

To view all caslibs, enter `caslib _all_ list;`.

SAS Studio provides a **Cloud Analytic Services** snippet that adds a new caslib:

```
/****************************************************************************/
/*  Create a CAS library (myCaslib) for the specified path ("/filePath/")    */
/*  and session (mySession).  If "sessref=" is omitted, the caslib is        */
/*  created and activated for the current session.  Setting global makes     */
/*  myCaslib visible in all active sessions. The default is local.  Setting */
/*  subdirs extends the scope of myCaslib to subdirectories of "/filePath".  */
/****************************************************************************/

caslib myCaslib datasource=(srctype="path") path="/filePath/" sessref=mySession global subdirs;
```

Specify the values that are particular for your environment.

In the following example, the CASLIB statement adds the caslib Myvapublic for a Hadoop Distributed File System (HDFS) data source.

```
caslib Myvapublic path="/vapublic"
                          datasource=(srctype="hdfs");
```

Alternatively, you can set the active caslib for your session with the CASLIB session option in the CAS statement:

```
cas casauto sessopts=(caslib="casuser");
```

Video: Accessing, Manipulating, and Saving Data Using SAS Cloud Analytic Services.

See Also

- "Caslibs" in *SAS Cloud Analytic Services: Fundamentals*
- "CASLIB Statement" in *SAS Cloud Analytic Services: User's Guide*
- "CASLIB= LIBNAME Option" in *SAS Cloud Analytic Services: User's Guide*
- "CASUTIL Procedure" in *SAS Cloud Analytic Services: User's Guide*

The CAS LIBNAME Engine

The CAS LIBNAME engine performs client/server communication. The engine is part of the SAS client and connects to the CAS server. The CAS libref is your handle to communicate from SAS with the specific session.

The CAS LIBNAME engine provides two important uses for SAS programming:

- The CAS libref and table name are used to identify in-memory tables that can then be used as input data for SAS Viya procedures.
- The engine enables data transfer between the SAS session and the CAS session.

When you assign a CAS engine libref, the CAS LIBNAME engine associates the libref with the active caslib unless you specify the LIBNAME statement CASLIB= option. If you use the CASLIB= option, the CAS engine libref is bound to the caslib. Binding the CAS engine libref with the active caslib prevents the CAS engine libref from switching to a subsequently added caslib.

Here are two examples of the CAS LIBNAME statement:

```
/* The libref mycas is associated with the active caslib.                */
/* If a new caslib is defined, mycas is associated with the new caslib.  */

libname mycas cas;


/* The libref mycas is bound to the testTables caslib,                   */
/* even if the active caslib changes to a caslib other than testTables.  */

libname mycas cas caslib=testTables;
```

For a complete example that uses the CAS LIBNAME engine, see "Use the SAS Windowing Environment to Connect to the CAS Server and Run the FACTMAC Procedure" on page 18.

Note: Downloading a CAS table with millions of rows can cause network degradation. You can use the DATALIMIT= LIBNAME option, the DATALIMIT= DATA set option, or the CASDATALIMIT= system option to limit the amount of data that transfers at one time. For more information about these options, see *SAS Cloud Analytic Services: User's Guide*.

See Also

"CAS LIBNAME Statement" in *SAS Cloud Analytic Services: User's Guide*

Load Data to a Caslib

You can load a SAS data set, database tables, and more to a caslib. After the data is in a caslib, you can use a DATA step, procedures, CAS actions, PROC DS2, or PROC FEDSQL, operations on the CAS table. Tables are not automatically saved when they are loaded to a caslib. You can use PROC CASUTIL to save tables. Native CAS tables have the file extension .sashdat. For information about file types that are supported in CAS, see "Path-Based Data Source Types and Options" in *SAS Cloud Analytic Services: User's Guide*.

Use the DATA Step to Load Data

This example uses the DATA step to perform client-side load of a data set from the Sashelp library to the caslib Mycaslib. PROC CASUTIL saves the table.

```
cas casauto;                                           /* 1 */

caslib mycaslib datasource=(srctype="path")
   path="/myCasTables/" sessref=casauto;               /* 2 */

libname mycas cas caslib=mycaslib;                     /* 3 */

data mycas.heart;
   set sashelp.heart;
   keep status BP_status weight_status
        smoking_status chol_status deathcause;         /* 4 */
run;

proc print data=mycas.heart(obs=5) ; run;              /* 5 */

proc casutil  incaslib="mycaslib" outcaslib="mycaslib"; /* 6 */
   save casdata="heart" replace;
run;
```

1 Start the CAS session and name it CASAUTO.

2 Add the caslib Mycaslib. The data source for this caslib is a server directory path, and the caslib is associated with the CASAUTO session. If you omit the SESSREF option, the caslib is associated with the most recently started session.

3 Assign a CAS LIBNAME libref to load the data to, and to bind the libref Mycas to the caslib Mycaslib.

4 Keep a limited number of variables from the original data set.

5 View the results. The table Mycas.Heart is an in-memory table. This table exists in memory for the duration of your CAS session. PROC PRINT performs data transfer of the five rows from the mycas.heart table on the CAS server to the SAS session.

Note: Printing large SAS Cloud Analytic Services tables can cause a large amount network traffic. You can use the OBS= system option to limit the number of rows the PRINT procedure processes.

6 Save the table to the **`/myCasTables`** directory on the server by using the SAVE CASDATA statement in PROC CASUTIL. To save a table in CAS, you specify the caslib and the table.

Video: Understanding Caslibs and Loading Data in SAS Viya

See Also

- "CASUTIL Procedure" in *SAS Cloud Analytic Services: User's Guide*
- "Tables Action Set Details" in *SAS Viya: System Programming Guide*
- "Data Lifecycle" in *SAS Cloud Analytic Services: Fundamentals*

Use PROC CASUTIL to Load Data

Loading tables into the server from a caslib's data source is the most efficient way to load data. In this example, a table is read from Oracle, and the in-memory table is kept in the same caslib. This is a server-side load. The server accesses Oracle rather than SAS.

```
caslib oralib datasource=(                                       /* 1 */
    srctype="oracle",
    uid="DBUSER",
    pwd="secret",
    path="//dbserver.example.com:1521/dbname",
    schema="DBUSER"
    );


proc casutil;
   list files;                                                   /* 2 */

   droptable casdata="sales" quiet;                              /* 3 */
   contents casdata="sales";                                     /* 4 */

   load casdata="sales" casout="sales" promote                   /* 5 */
       label="Fact table for User-to-Item Analysis"
       varlist=(
            (name="USERID"  label="User ID"),
            (name="ITEMID"  label="Item ID")
       );

quit;

run;
```

1 Add a caslib that uses Oracle as the data source. Oralib becomes the active caslib for the session and the subsequent programming statements use it for input and output.

2 The LIST FILES statement displays the tables that are available in the Oracle database.

3 The DROPTABLE statement includes the QUIET option. Running this statement is useful on repeated runs because it ensures that no table named Sales can be in-memory to interfere with the subsequent LOAD CASDATA= statement.

4 Because the first CONTENTS statement follows the DROPTABLE statement, the table information and column information from Oracle are read.

5 The CASDATA= argument in the LOAD statement indicates that the Sales table is read from the caslib's data source (Oracle) into SAS Cloud Analytic Services. Options are specified to add labels to the table and columns.

Video: Loading Data

See Also

"CASUTIL Procedure" in *SAS Cloud Analytic Services: User's Guide*

Use CAS Actions to Load Data

The CAS procedure interacts with the CAS server by using CAS actions. An advantage to using CAS actions for table management is that the table actions have more features that you can use in PROC CAS than are available in PROC CASUTIL. When you are working with large data sets, it is more efficient to use a server-side load with the loadTable action to access data. Here is an example that loads the iris.sashdat table from the server to the active caslib and names the table IRIS:

```
proc cas;
   table.loadtable /path="iris.sashdat" casOut={name="IRIS"};
run;
```

Note: CAS server file names that you specify are case sensitive.

Some of the features available in the Tables action set that are not part of PROC CASUTIL are the following:

- managing caslibs
- creating user-defined table attributes
- fetching table rows from a table
- updating table rows

See Table Action Set: Syntax for a complete list of table actions.

See Also

- "CAS Procedure" in *SAS Cloud Analytic Services: CASL Reference*
- "CAS Actions"

Use the PROC CAS UPLOAD Statement to Load Data

The PROC CAS UPLOAD statement loads client-side data to the server. The following example uses the HTTP procedure to download the Iris data set locally, and then uses the PROC CAS UPLOAD statement to load the file into CAS. Loading data from a client-side file is appropriate for small data sets and when you are learning to program with CAS actions. When the data sets become larger, it is more efficient to use a server-side load with the loadTable action to access data.

```
filename data "file-path/iris.csv";                    /* 1 */
proc http method="get"
   url="http://support.sas.com/documentation/onlinedoc/viya/exampledatasets/iris.csv"
   out=data;
run;

proc cas;
   upload path="file-path/iris.csv"                    /* 1 */
        casOut={
        name="iris"
        caslib=casuser
      }
     importOptions={fileType="csv"}
;
run;
```

1 The HTTP procedure makes a GET request to support.sas.com and downloads the Iris.csv file to a local directory.

2 The UPLOAD statement transfers the file Iris.csv to the Casuser caslib on the CAS server. The CASOUT option uses the name parameter to specify the table name Iris. The IMPORTOPTIONS= “CSV” option specifies the file type.

Output 1 *SAS Log*

```
 NOTE: Active Session now CASAUTO.
 NOTE: Cloud Analytic Services made the uploaded file available as table IRIS in caslib CASUSER(casdemo).
 NOTE: The table IRIS has been created in caslib CASUSER(casdemo) from binary data uploaded to Cloud Analytic
Services.
 {caslib=CASUSER(casdemo),tableName=IRIS}
```

See Also

- “UPLOAD Statement” in *SAS Cloud Analytic Services: CASL Reference*
- “HTTP Procedure” in *Base SAS Procedures Guide*

Programming Examples Using SAS Cloud Analytic Services

Example Data

Some examples use data from the Sashelp library. If a data set is not in the Sashelp library, you can install the sample data in the Sashelp library if you have sudo privileges. The instructions for installing the sample data sets are in the deployment guide at support.sas.com. Contact your administrator for assistance.

To download CSV files, see SAS Viya Example Data Sets.

Use SAS Studio to Connect to the CAS Server and Run a Gradient Boosting Model Procedure

The SAS system options that configure the CAS server, CASHOST= and CASPORT=, are set during deployment.

After you are signed in to SAS Studio, initiate a session with the server by using this CAS statement:

```
cas myCasSess;
```

This example starts a session with the CAS server, loads data to the server, and creates a gradient boosting model using the GRADBOOST procedure.

```
cas casauto;                                                      /* 1 */
libname mycas cas;                                                /* 2 */

data mycas.junkmail;                                              /* 3 */
   set sashelp.junkmail;
run;

proc gradboost data=mycas.junkmail outmodel=mycas.gradboost_model;   /* 4 */
   input Address Addresses All Bracket Business CS CapAvg CapLong
         CapTotal Conference Credit Data Direct Dollar Edu Email
         Exclamation Font Free George HP HPL Internet Lab Labs
         Mail Make Meeting Money Order Original Our Over PM Paren
         Parts People Pound Project RE Receive Remove Semicolon
         Table Technology Telnet Will You Your _000 _85 _415 _650
         _857 _1999 _3D / level = interval;
   target class /level=nominal;
   output out=mycas.score_at_runtime;                             /* 5 */
   ods output FitStatistics=fit_at_runtime;
run;
```

1 The CAS statement connects to the CAS server and creates a session with the default name CASAUTO on that server. This step also sets Casuser as the active caslib for the CASAUTO session.

2 The LIBNAME statement creates a new SAS libref, Mycas, and assigns it to the CAS engine.

3 The DATA step loads the Sashelp.Junkmail data set to the server-side table Junkmail in the Casuser caslib.

4 Execute the GRADBOOST procedure. The input data comes from the Casuser caslib. The same is true for the output model table Gradboost_Model.

5 The OUTPUT statement scores the training data and saves the results to a new table named fit_at_runtime.

To further explore this example and to view the output, see the GRADBOOST procedure.

See Also

- “Connecting to the CAS Server and Creating a CAS Session” on page 4
- “Caslibs” on page 12
- “The CAS LIBNAME Engine” on page 13
- “Load Data to a Caslib” on page 14

Use the SAS Windowing Environment to Connect to the CAS Server and Run the FACTMAC Procedure

After connecting to the CAS server and establishing a session using the CAS statement, the CAS LIBNAME engine libref is your handle to communicate with the specific session from a SAS programming client.

Note:

Authentication is used to control access to the CAS server and its resources. Your identity must be successfully authenticated before your session is created.

For SAS Studio and for instances where SAS Enterprise Guide uses the SAS Workspace Server, your user credentials authenticate you to the CAS server. If your SAS Enterprise Guide server is local to your computer and is not a remote server, an attempt is made to find an authinfo file (.authinfo is the default filename on UNIX, _authinfo on Windows). The authinfo file provides a user name and password to CAS for operating system authentication. For more information, see Authentication Mechanisms and *Client Authentication Using an Authinfo File*.

The SAS windowing environment requires an authinfo file for authentication if your site does not use kerberos for authentication. If authentication to the CAS server fails, see *Client Authentication Using an Authinfo File* or contact your site administrator.

The example below demonstrates the following concepts:

- Connection information is required to connect to CAS from the SAS windowing environment.
- You must explicitly start a CAS session from the SAS windowing environment.
- The CAS LIBNAME engine is required when you use a SAS procedure to transfer data from the CAS server.
- SAS Viya analytic procedures use CAS processing, so the engine provides the way to identify the tables to use for analysis.

You can download the compressed archive file from the website at http://files.grouplens.org/datasets/movielens/ml-100k.zip and use any third-party unzip tool to extract all the files in the archive to the destination directory of your choice. The file that contains the ratings is **`u.data`**. The following CASUTIL procedure loads the data table from the directory into your CAS session:

```
option casport=5570 cashost="cloud.example.com";                          /* 1 */
cas casauto sessopts=(caslib=casuser);                                    /* 2 */
libname mycas cas;                                                        /* 3 */

proc casutil;                                                             /* 4 */
   load file="your-file-path/data/u.data"
   casout="movlens"
   importoptions=(filetype="CSV" delimiter="TAB" getnames="FALSE"
               vars=("userid" "itemid" "rating" "timestamp"));
run;

proc factmac data=mycas.movlens nfactors=10 learnstep=0.15                /* 5 */
               maxiter=20 outmodel=mycas.factors;

   input userid itemid /level=nominal;
   target rating /level=interval;
   output out=mycas.out1 copyvars=(userid itemid rating);
run;

proc print data=mycas.factors(obs=10);                                    /* 6 */
run;
cas casauto terminate;                                                    /* 7 */
```

1 Specify your connection information. The CASHOST= and CASPORT= system options specify the connection information. As an alternative, your administrator might have set the default host and port for your server in a configuration file.

2 Start a session and set the active caslib to Casuser. You can start a session manually with the CAS statement. In this example, the CAS statement starts the CAS session named CASAUTO.

3 Specify the CAS engine LIBNAME statement and use the libref with the input table name. The LIBNAME statement assigns a CAS engine libref named Mycas that you use to connect to the session CASAUTO. In subsequent procedure steps, input table names must begin with the CAS engine libref named Mycas.

Some procedures require output table names to also begin with the CAS engine libref. See the documentation for your procedure for more information. PROC CASUTIL require a CAS engine libref on both the input and output table names. For documentation about the CAS engine, see *SAS Cloud Analytic Services: User's Guide*.

4 Add the u.Data data set as an in-memory table named Movlens.

5 Execute the FACTMAC procedure. The input data comes from the Casuser caslib. Use the Mycas libref to identify the input table for a SAS Viya procedure.

6 View ten rows from the new data. The engine performs data transfer from the CAS server to SAS.

7 At the end of your program, when you no longer need to access data in CAS, you can terminate your session to preserve resources. If you do not terminate your session at the end of your program, the session has a time-out interval and eventually terminates on its own.

A Factorization Machine Model Using the Factmac CAS Action

You can also do factorization by using the factmac CAS action. You can use the movie recommendation example in "Use the SAS Windowing Environment to Connect to the CAS Server and Run the FACTMAC Procedure" to compare programming using CAS actions and procedures.

This example uses the factmac action, which uses observations in a data table to train a factorization machine model. In this example, the data is derived from companies that provide movies for online viewing. A company wants to offer its customers recommendations of movies that they might like. These recommendations are based on ratings that are provided by users.

```
/*In the SAS Windowing Environment, you must specify the connection information*/
option casport=your-port-number cashost="cloud.example.com";
cas casauto sessopts=(caslib=casuser);                        /* 1 */

proc casutil;                                                 /* 2 */
    load file="u.data"
    casout="movlens"
    importoptions=(filetype="CSV" delimiter="TAB" getnames="FALSE"
                   vars=("userid" "itemid" "rating" "timestamp"));
run;

proc cas;                                                     /* 3 */
      factmac result=R / table={name="movlens"},              /* 4 */
      outModel={name="factors_out", replace=true},            /* 5 */
      inputs={"userid", "itemid"},                            /* 6 */
      nominals={"userid", "itemid"},                          /* 7 */
      target="rating",                                        /* 8 */
      maxIter=20, nFactors=10, learnStep=0.15,                /* 9 */
      output={casout={name="score_out", replace="TRUE"},      /* 10 */
      copyvars={"userid","itemid","rating"}};
run;

print r;
```

1 Create a session with the CAS server and use the Casuser caslib. Casuser is the personal caslib that is set up for your user ID by the administrator.

2 Use the CASUTIL procedure to load the data to the in-memory table Movlens using the import options that are specified in the IMPORTOPTIONS= option. The file **`u.data`** contains the movie ratings.

PROC CASUTIL is a data management procedure for managing CAS server tables.

3 Execute the CAS actions with the CAS procedure.

PROC CAS executes CAS actions. In addition, you can code by using the new CAS language (CASL) within PROC CAS.

4 Execute the factmac action from the factmac action set, and place the results in the variable R. The table parameter names the input data table to be analyzed. An action sends a request to the server and returns a result. The results are usually a data table or a metadata table.

5 The `outModel` parameter creates output for the fitted parameter estimates to the factors_out data table.

6 The `inputs` parameter specifies the input variables.

7 The `nominals` parameter specifies that the *userid* and *itemid* variables are nominal.

8 The `target` parameter specifies the target variable.

9 The `maxIter` parameter specifies the number of iterations for the optimization. The `nFactors` parameter specifies the number of factors to be estimated for the factorization machine model. The `learnStep` parameter specifies the learning step size for the optimization.

10 The `output` parameter names the in-memory table in the `casout` parameter and specifies that an existing table is to be replaced.

For more information about this example, see the MovieLens data example for the Factorization Machine action set.

See Also

- "CAS Actions" on page 39
- *SAS Cloud Analytic Services: CASL Reference*

Generate Sentiment Results from Input Documents

This example executes the sentimentAnalysis action set to generate sentiment results for documents in the input table AMAZON.

```
/*If not already done, create session Casauto. Specify a host and port that are valid for your site.*/
/*options cashost="cloud.example.com" casport=5570;*/
/*cas casauto;*/

libname mycas cas sessref=mysess;                     /* 1 */

proc cas;
   session mysess;
   loadactionset "sentimentAnalysis";                 /* 2 */

   action sentimentAnalysis.applySent;                /* 3 */
      param
         docId="_document_"
         text="reviewbody"
         table={name="amazon"}
         casOut={name="outsent", replace=TURE}
      ;
run;
quit;

proc sort data=mycas.outsent out-sorted;              /* 4 */
   by _document_;
```

```
run;
```

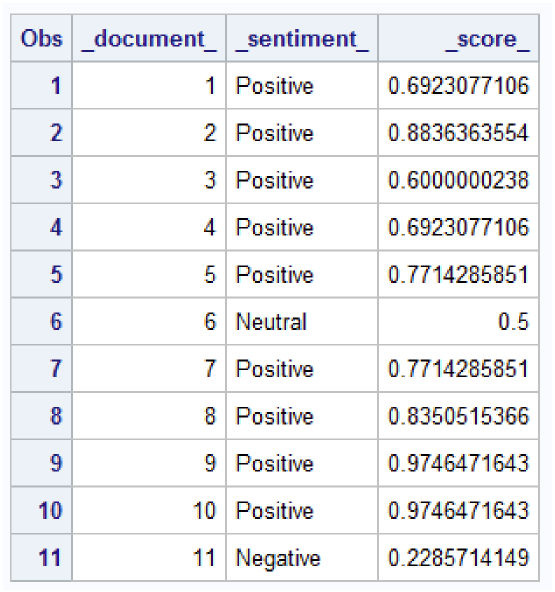
1 Associate the Mycas libref with the CAS engine.

2 Load the sentimentAnalysis action set.

3 Use the applySent action to get the sentiment results from the table.

4 Sort the results.

Here are the results of the sentiment scoring:



Obs	_document_	_sentiment_	_score_
1	1	Positive	0.6923077106
2	2	Positive	0.8836363554
3	3	Positive	0.6000000238
4	4	Positive	0.6923077106
5	5	Positive	0.7714285851
6	6	Neutral	0.5
7	7	Positive	0.7714285851
8	8	Positive	0.8350515366
9	9	Positive	0.9746471643
10	10	Positive	0.9746471643
11	11	Negative	0.2285714149

Getting Started with Machine Learning

Getting Started with SAS Visual Data Mining and Machine Learning is a case study that demonstrates the new high-performance analytic procedures. You can explore and prepare data, create and assess models, and score new data. Follow along with this case study by downloading the data and copying the code to the SAS Studio programming interface.

Data Types

The CAS server supports the VARCHAR, INT32, INT64, and VARBINARY data types, in addition to the CHARACTER and NUMERIC data types, which are traditionally supported by SAS. You use the VARBINARY data type for image, audio, and document files.

Variables that are created using the VARCHAR data type are varying width and use character semantics, rather than being fixed-width and using the byte semantics of the traditional CHARACTER data type. Using the VARCHAR data type in the DATA step in the CAS server has some restrictions. For more information, see "Restrictions for the VARCHAR Data Type in the CAS Engine" in *SAS Cloud Analytic Services: User's Guide*.

Variables that are created or loaded using the INT32 or INT64 data types support more digits of precision than the traditional NUMERIC data type. All calculations that occur on the CAS engine maintain the INT32 or INT64 data type. Calculations in DATA steps or procedures that run on the SAS 9 engine are converted to NUMERIC values.

The CHARACTER and NUMERIC data types continue to be the supported data types for processing in the SAS Workspace Server.

The DS2 language supports several additional data types. On the CAS server, DS2 converts non-native data types to CHARACTER, NUMERIC, or VARCHAR. For information about data types that are supported for specific data sources, see "Data Type Reference" in *SAS DS2 Language Reference*.

The CAS language (CASL) determines the data type of a variable when the variable is assigned.

In the following table, the letter Y indicates the data types that are supported for programming on the CAS server. In the last column, Y indicates data types that are supported on the SAS Workspace Server.

Data Type	**CAS Actions**	**CASL**	**Data Connectors*** **	**Procedures and DATA Step**	**DS2**	**FedSQL**	**Workspace Server Processing***
BIGINT			Y		Y		
BOOLEAN		Y	Y				
CHARACTER (CHAR)	Y	Y	Y	Y	Y	Y	Y
DATE		Y	Y		Y		
DATETIME		Y	Y				
DOUBLE**	Y	Y	Y	Y	Y	Y	
FLOAT			Y		Y		
INTEGER			Y		Y		
INT32		Y	Y		Y	Y	
INT64		Y	Y		Y	Y	
ITEMS		Y					
LISTS		Y					
NCHAR			Y		Y		
NUMERIC (NUM)			Y				Y **
NVARCHAR			Y		Y		
SMALLINT			Y		Y		
STRING UTF-8		Y					
TABLE		Y					
TIME		Y	Y		Y		
TIMESTAMP			Y		Y		
TINYINT			Y		Y		
VARBINARY	Y	Y					

Data Type	CAS Actions	CASL	Data Connectors ***	Procedures and DATA Step	DS2	FedSQL	Workspace Server Processing*
VARCHAR	Y	Y	Y	Y	Y	Y	

* Some of the SAS utility procedures read CAS tables or metadata about the tables, but do not run on the CAS server. For example, the PRINT procedure reads and prints CAS tables, including VARCHAR columns. For a list of utility procedures that support CAS data, see "SAS Viya Foundation Procedures".

** The SAS NUMERIC data type is the same as a DOUBLE data type.

*** Additional data types are supported by the data connectors. These data types are first converted to data types that can be processed on the CAS server. Check the data connector documentation for your data source to ensure that a data type is supported. For more information, see "Data Connectors" in *SAS Cloud Analytic Services: User's Guide*.

Note: The CVP LIBNAME engine converts the CHAR data type to VARCHAR if you specify the CVPVARCHAR=YES LIBNAME option.

SAS/IML Procedure

SAS/IML software gives you access to a powerful and flexible programming language in a dynamic, interactive environment. The acronym IML stands for "interactive matrix language." You must have SAS/IML licensed and installed in order to use these procedures. See SAS/IML for more information.

SAS Visual Statistics Procedures

The statistical procedures available on SAS Viya run only on in-memory CAS tables. You must license and install SAS Visual Statistics to use these procedures.

For examples, see the individual procedures listed in the table.

Procedure	Description
ASSESS	Assesses and compares supervised learning models and calculates fit statistics, such as average square error, mean square logarithmic error, mean absolute error, mean consequential error, and multiclass log loss. For a supervised learning model that has a nominal target, the ASSESS procedure produces lift information and receiver operating characteristic (ROC) information. For a regression model, the ASSESS procedure performs a quantile binning of the predictions, and then returns the summary statistics of the response variable for each bin.
BINNING	Performs binning. The BINNING procedure supports several binning methods and can calculate the weight of evidence (WOE) and information value (IV), based on binning results. Video: Transforming Data
CARDINALITY	Determines a variable's cardinality or limited cardinality. Video: Selecting Features

Procedure	Description
CORRELATION	Computes Pearson correlation coefficients and probabilities.
FREQTAB	Produces one-way to *n*-way frequency and crosstabulation (contingency) tables and provides a variety of tests and measures to analyze frequency and crosstabulation tables.
GAMMOD	Fits generalized additive models that are based on low-rank regression splines.
GENSELECT	Fits and performs model selection for generalized linear models.
ICA	Performs independent component analysis.
KCLUS	Performs clustering, which is a common step in data exploration. Video: K-Means Clustering
LMIXED	Fits a variety of mixed linear models to data and enables you to use these fitted models to make statistical inferences about the data.
LOGSELECT	Fits and performs model selection for logistic regression models, including binary, binomial, and multinomial response models.
MBC	Fits mixtures of multivariate Gaussian and uniform distributions to achieve unsupervised and semi-supervised clustering of data.
MDSUMMARY	Computes basic descriptive statistics in parallel for CAS tables.
MODELMATRIX	Creates the design matrix that is associated with a specified data set and MODEL statement.
NLMOD	Fits nonlinear regression models with standard or general distributions.
PARTITION	Performs simple random sampling, stratified sampling, oversampling, or *k*-fold partitioning to produce a table that contains a subset of observations or that contains partitioned observations. Video: Exploring Data
PCA	Performs principal component analysis, which is a multivariate technique for examining relationships among several quantitative variables. Video: Principal Component Analysis
PHSELECT	Fits the Cox proportional hazards regression models for survival data and performs variable selection.
PLSMOD	Fits reduced-rank linear models by using any one of a number of linear predictive methods, including partial least squares (PLS).
QTRSELECT	Fits and performs model selection for quantile regression models.
REGSELECT	Fits and performs model selection for ordinary linear least squares models.

Procedure	Description
SPC	Performs Shewhart control chart analysis.
TREESPLIT	Builds tree-based statistical models for classification and regression. Video: Auto-tuning: Decision Tree
VARIMPUTE	Performs numeric variable imputation. Video: Transforming Data
VARREDUCE	Performs both supervised and unsupervised variable selection. Video: Selecting Features

SAS Visual Data Mining and Machine Learning Procedures

CAS procedures enable you to have the familiar experience of coding SAS procedures, but behind each procedure statement is one or more CAS actions that run across multiple machines. You must have SAS Visual Data Mining and Machine Learning licensed and installed to use these procedures.

For examples, see the individual procedures listed in the table.

Procedure	Description
ASTORE	Enables you to save the state from a predictive analytic procedure and to use the saved state to score new data. The state is created using the results from the training phase of model development.
BNET	Learns a Bayesian network from an input data set.
BOOLRULE	Enables you to extract Boolean rules from large-scale transactional data. The procedure can automatically generate a set of Boolean rules by analyzing a text corpus that has been processed by the TEXTMINE procedure and that is represented in a transactional format.
FACTMAC	Implements the factorization machine model, which has applications in predictive modeling and recommendation. Video: Factorization Machines
FASTKNN	Implements the *k*-nearest neighbor (*k*-NN) search algorithm, which has many applications, including recommendation systems, image search, and fingerprint recognition.
FISM	Performs frequent item-set mining, which looks for frequent patterns in a large database by using the FP-growth (frequent pattern growth) algorithm of Han, Pei, and Yin (2000).

Procedure	Description
FOREST	Creates a predictive model called a forest, which consists of several decision trees. Video: Autotuning: Forest Video: Ensemble Methods: Forest
GMM	Creates a Gaussian mixture model, which is a probabilistic model for soft clustering. In this model, all the data points are assumed to be from a mixture of Gaussian distributions where the number of clusters is inferred nonparametrically by using the Dirichlet process.
GRADBOOST	Creates a predictive gradient boosting model, which consists of multiple decision trees. The procedure samples the original data without replacement to create the training data for an individual tree, and performs the action of sampling multiple times throughout a run. Video: Ensemble Methods: Gradient Boosting
GVARCLUS	Performs variable clustering and graphical modeling. The procedure divides a set of variables into disjoint clusters and creates tables that contain the edge and vertex information for defining an undirected graph.
MBANALYSIS	Performs association rule mining on a transaction database.
MTLEARN	Implements the multitask learning technique for least squares loss with L-1 and graph structure penalizations. The procedure solves multiple related sparse linear regression problems simultaneously by using the graph structure to encode the relationships between the problems.
MWPCA	Implements moving windows robust principal component analysis. You can use this procedure to capture changes in principal components over time by using sliding windows. Also, you can choose to perform robust principal component analysis (RPCA) on each window. In RPCA, outliers and noise are excluded from each window before the analysis is performed.
NETWORK	Provides a number of network analysis algorithms that use an abstract graph or network as input, help explain network structure, and compute important network measures. Depending on the application, this type of network analysis can stand on its own to provide independent value, or it can support machine learning models. Video: Community Detection in Networks
NNET	Trains a multilayer perceptron neural network. The procedure can also use a previously trained network to score a data table; this is referred to as stand-alone scoring. Or, it can generate SAS DATA step statements that can be used to score a data table. Video: Autotuning: Neural Network

Procedure	Description
RPCA	Implements the robust principal component analysis. RPCA has applications in many areas, including image processing, latent semantic indexing, ranking, and matrix completion.
SEMISUPLEARN	Implements graph-based semisupervised learning based on the label spreading algorithm. The procedure predicts the labels for unlabeled data by computing the similarity measure between the labeled data and the unlabeled data in an iterative way. The graph-based semisupervised learning algorithm has important applications in natural language processing, computer vision, financial data analysis, and text analytics.
SVDD	Implements the support vector data description (SVDD), a machine learning algorithm. SVDD is a one-class classification technique that is useful in applications where data that belongs to one class is abundant but data about any other class is scarce or missing. Fraud detection and process control are examples of application areas where the majority of the data belongs to one class.
SVMACHINE	Implements the support vector machines (SVM) algorithm, which computes support-vector machine-learning classifiers for the binary pattern recognition problem. This method has been broadly used in fields such as image classification, handwriting recognition, financial decision-making, and text mining.
TEXTMINE	Integrates natural language processing and statistical analysis to analyze large-scale textual data. Video: Text Mining
TMSCORE	Scores textual data. In text mining, scoring is the process of applying parsing and singular value decomposition (SVD) projections to new textual data. Video: Text Mining
TSNE	Implements the t-distributed stochastic neighbor embedding algorithm for dimension reduction and visualization of high-dimensional data. The procedure computes two- or three-dimensional representations by optimizing the Kullback-Leibler divergence between joint probabilities in low and high dimensions.

SAS Visual Text Analytics

CAS procedures enable you to have the familiar experience of coding SAS procedures, but behind each procedure statement is one or more CAS actions that run across multiple machines. You must have SAS Visual Text Analytics licensed and installed to use these procedures.

Procedure	Description
BOOLRULE	Enables you to extract Boolean rules from large-scale transactional data. The procedure can automatically generate a set of Boolean rules by analyzing a text corpus that has been processed by the TEXTMINE procedure and that is represented in a transactional format.
TEXTMINE	Integrates natural language processing and statistical analysis to analyze large-scale textual data. Video: Text Mining
TMSCORE	Scores textual data. In text mining, scoring is the process of applying parsing and singular value decomposition (SVD) projections to new textual data. Video: Text Mining

SAS Econometrics Procedures and Packages

SAS Econometrics provides a set of procedures that provide techniques to model complex business and economic scenarios and to analyze the dynamic impact that specific events might have over time. You must have SAS Econometrics licensed and installed in order to use these procedures.

Procedure	Description
CARIMA	Analyzes and forecasts univariate time series or transactional data by using the autoregressive integrated moving average (ARIMA) model. It supports seasonal, subset, and factored ARIMA models and allows for missing values in time series.
CCDM	Prepares an empirical estimate of the probability distribution of S, which is the sum of N continuous, IID random variables X. Conducts scenario and perturbation analyses to assess the effects of external factors (regressors) and uncertainty in the parameters of the models.
CCOPULA	Fits multivariate distributions by using the copula framework (which separates marginal distributions and dependence structure) and performs large-scale multivariate simulations from estimated or provided models.
CESM	Generates forecasts by using exponential smoothing models with optimized smoothing weights for time series data or transactional data.
CNTSELECT	Analyzes regression models in which the dependent variable takes nonnegative integer values (count values).
CPANEL	Analyzes a class of linear econometric models that arise when time series and cross-sectional data are combined.

Procedure	Description
CQLIM	Analyzes univariate qualitative and limited dependent variable models in which dependent variables take discrete values or are observed only in a limited range.
CSPATIALREG	Analyzes a class of linear spatial econometric models for cross-sectional data whose observations are spatially referenced.
ECM	Prepares an empirical estimate of the probability distribution of the sum of multiple continuous, dependent random variables by combining a simulated sample of their joint probabilities with the empirical samples of their respective marginal probability distributions, which does not need to be identical.
HMM	Supports hidden Markov models (HMMs), which have been widely applied in economics, finance, science, and engineering.
SEVSELECT	Estimates parameters of any predefined or user-defined continuous probability distribution that is used to model the magnitude (severity) of a continuous-valued event of interest, while accounting for censoring and truncation, and estimates the effect of regressors on the scale parameter of the distribution by selecting the best subset of regression effects.
TSMODEL	Provides a new distributed time series analysis and scripting environment, which is supported by the time series analysis (TSA) package, the time series modeling (TSM) package, and the utility package.
Time Series Analysis Package	Provides tools for efficient statistical analysis of time series, including transformations, decompositions, statistical tests for intermittency, seasonality, stationarity, and forecast bias.
Time Series Model Package	Provides tools for flexible time series modeling and forecasting. This package enables you to create custom time series models from families such as intermittent demand (IDM), ARIMA, exponential smoothing (ESM), and unobserved component (UCM) models.
Utility Package	Provides tools for statistical analysis of forecast series (that is, pairs of actual and predicted series) and programming utilities. This package contains programming utilities that facilitate time series code development.

SAS/QC Procedures

SAS/QC software, a component of SAS, provides a comprehensive set of tools for statistical quality improvement. You must have SAS/QC licensed and installed in order to use these procedures.

Procedure	Description
ANOM	Analysis of means (ANOM) is a graphical and statistical method for simultaneously comparing *k* treatment means with their overall mean at a specified significance level. You can use the ANOM procedure to create ANOM charts for various types of response data, including continuous measurements, proportions, and rates.
CAPABILITY	You can use the CAPABILITY procedure to compute summary statistics and process capability indices.
CUSUM	The CUSUM procedure creates cumulative sum control charts, also known as cusum charts, which display cumulative sums of the deviations of measurements or subgroup means from a target value.
FACTEX	The FACTEX procedure generates various designs, such as factorial designs, fractional designs, and so on.
ISHIKAWA	The ISHIKAWA procedure provides a highly interactive graphics environment for creating and modifying Ishikawa diagrams.
MACONTROL	The MACONTROL procedure creates moving average control charts, which are tools for deciding whether a process is in a state of statistical control and for detecting shifts in a process average.
MVPDIAGNOSE	The MVPDIAGNOSE procedure is used in conjunction with the MVPMODEL and MVPMONITOR procedures to monitor multivariate process variation over time, to determine whether the process is stable, and to detect and diagnose changes in a stable process. The MVPDIAGNOSE procedure produces graphs that can provide insight into the variation in a process.
MVPMODEL	The MVPMODEL procedure is used in conjunction with the MVPMONITOR and MVPDIAGNOSE procedures to monitor multivariate process variation over time, to determine whether the process is stable, and to detect and diagnose changes in a stable process. The MVPMODEL procedure provides computational and graphical tools for building a principal component model from multivariate process data in which the measured variables are continuous and correlated.
MVPMONITOR	The MVPMONITOR procedure is used in conjunction with the MVPMODEL and MVPDIAGNOSE procedures to monitor multivariate process variation over time, to determine whether the process is stable, and to detect and diagnose changes in a stable process. The MVPMONITOR procedure produces control charts for multivariate process data.
OPTEX	The OPTEX procedure searches for optimal experimental designs.
PARETO	The PARETO procedure creates Pareto charts, which display the relative frequencies of quality-related problems in a process or operation.

Procedure	Description
RAREEVENTS	The RAREEVENTS procedure produces control charts for rare events.
RELIABILITY	The RELIABILITY procedure provides tools for reliability and survival data analysis and for recurrent events data analysis.
SHEWHART	The Shewhart control chart is a graphical and analytical tool for deciding whether a process is in a state of statistical control.

SAS Visual Forecasting Procedures

SAS Visual Forecasting procedures provide automatic variable, event, and model selection. You must have SAS Visual Forecasting licensed and installed in order to use these procedures.

Procedure	Description
SMCALIB	Calibrates one or more project definitions that are generated by the SMPROJECT procedure, performs holdout analysis for each model that is specified for each project, and generates output for selecting one or more models for scoring and for the scoring itself.
SMPROJECT	Generates a data table (called a project repository) that contains a full description of the stability monitoring (SM) project.
SMSCORE	Scores one or more SM projects for a specified repository and one or more specified models. For each model, it generates forecast values for a target variable, its prediction limits, and the difference between the forecast and the actual value.
SMSELECT	Selects the best model for stability monitoring.
SMSPEC	Generates an ordinary least squares (OLS) regression or ARIMA model specification (or both) to be used by the SMPROJECT procedure.
TSINFO	Evaluates a variable in an input data table for its suitability as a time ID variable in SAS procedures and solutions that are used for time series analysis.
TSMODEL	Analyzes timestamped transactional data with respect to time and accumulates the data into a time series format. The procedure forms time series vectors from timestamped data, and then provides these vectors as array variables for subsequent processing by your program statements.

Procedure	Description
TSRECONCILE	Reconciles forecasts of timestamped data at two different levels of a hierarchy in a top-down fashion when the input data are contained in CAS tables.

SAS Optimization Procedures

SAS Optimization software is a set of procedures for exploring models of distribution networks, production systems, resource allocation problems, and scheduling problems. These procedures use the tools of operational research. You must have SAS Optimization licensed and installed in order to use these procedures.

Procedure	Description
Mathematical Optimization Procedures	
CLP	The CLP procedure is a finite-domain constraint programming solver for constraint satisfaction problems (CSPs) with linear, logical, and global constraints.
OPTLP	Solves linear programming problems that are submitted in a SAS data table that uses a mathematical programming system (MPS) format.
OPTMILP	Solves general and mixed integer linear programming (MILP) problems in which a subset of the decision variables are constrained to be integers. The OPTMILP procedure solves MILP problems with an LP-based branch-and-bound algorithm, augmented by advanced techniques such as cutting planes and primal heuristics.
OPTQP	Solves quadratic problems using the interior point algorithm. These are problems with a quadratic objective function and a collection of linear constraints, including general linear constraints along with lower or upper bounds (or both) on the decision variables.
OPTMODEL	Includes the OPTMODEL modeling language and provides a modeling environment for building, solving, and maintaining optimization models.
Network Optimization Procedures	
OPTNETWORK	Includes a number of graph theory and network optimization algorithms that can augment more generic mathematical optimization approaches.

SAS Viya Foundation Procedures

The core product of SAS Viya is SAS Visual Analytics. If you install SAS Visual Analytics only, then you have access to the Base SAS procedures in the following table. If you have SAS 9.4M5 installed, all SAS 9.4 Base procedures run on SAS Viya. If you have SAS Viya with any offering in addition to SAS Visual Analytics licensed and installed, you also have access to all SAS 9.4 Base procedures. For all Base procedure documentation, see *Base SAS Procedures Guide*.

Most of these procedures run on the SAS Workspace Server or the SAS Compute Server and not on the CAS server. The data or metadata that these procedures use transfer data from the CAS server to the SAS Workspace Server or the SAS Compute Server. To perform operations on CAS tables, the CAS LIBNAME engine can connect a SAS 9.4 session to an existing CAS session through the CAS session name or through the CAS session UUID. Large tables might cause performance degradations.

Beginning with SAS 9.4M5, some Base SAS procedures are enhanced to process data using CAS actions. For more information, see "CAS Processing of Base Procedures" in *Base SAS Procedures Guide*.

Table 1   SAS Foundation Procedures That Are Available for Sites with Only SAS Viya and SAS Visual Analytics Installed

Procedure	Description	Uses CAS Processing
APPEND	Adds rows from a CAS table to the end of a SAS data set, and adds rows from a SAS data set to the end of a CAS table.	Yes
CAS	Provides a programming environment that is based on the CASL language specification. This environment enables you to interact with SAS Cloud Analytic Services (CAS) from the SAS client.	Yes
CASUTIL	Works with tables in SAS Cloud Analytic Services, SAS data sets in SAS libraries, and external files.	Yes
CATALOG	Manages entries in SAS catalogs.	No
COMPARE	Compares the contents of two SAS data sets, selected variables in different data sets, or variables within the same data set.	No
CONTENTS	Shows the contents of a CAS table and prints the directory of the caslib.	Yes
COPY	Copies tables to and from libraries.	Yes
DATASETS	Manages CAS tables.	Yes
DELETE	Deletes SAS data sets and CAS tables.	Yes
DOWNLOAD	In SAS/CONNECT, copies to the client SAS files that are stored on the server.	No
DS2	Runs DS2 code.	Yes

Procedure	Description	Uses CAS Processing
DSTODS2	Converts DATA step code to DS2 code.	No
EXPORT	Writes a SAS data set to delimited files or to JMP files.	No SAS/ACCESS to PC Files must be purchased to add support for other file types, such as Microsoft Access databases for DBMS=ACCESS, Microsoft Excel workbooks, dBase database (DBF) files, and IBM Lotus spreadsheets.
FCMP	Enables you to create, test, and store SAS functions, CALL routines, and subroutines before you use them in other SAS procedures or in DATA steps.	Yes
FEDSQL	Submits FedSQL code.	Yes, with limitations on page 44 for the CAS server
FMTC2ITM	Converts one or more format catalogs into a single item store.	No
FORMAT	Creates user-defined formats.	Yes PROC FORMAT stores user-defined formats in a CAS format library, which is associated with a CAS session. CAS user-defined formats must be associated with a variable before CAS processes a table.
HADOOP	Reads and writes Hadoop data.	No
HDMD	Generates XML-based metadata that describes the contents of files that are stored in HDFS.	No
HTTP	Processes data from the web.	No
IMPORT	Reads external data into a SAS data set.	No
JAVAINFO	Shows information about the version of Java on your system.	No
JSON	Reads data from a SAS data set and writes it to an external file in JSON representation.	No
LUA	Enables you to run statements from the Lua programming language within SAS code.	Yes
MAPIMPORT	Produces an output map data set, which can be used with the ODS Graphics SGMAP procedure or other mapping procedures in SAS.	No
MDSUMMARY	Computes basic descriptive statistics for variables across all observations or within groups of observations in parallel for data tables that are stored in SAS Cloud Analytic Services (CAS).	Yes

Procedure	Description	Uses CAS Processing
MEANS	Provides data summarization tools to compute descriptive statistics for variables across all observations and within groups of observations.	Yes, if the DATA= option specifies a data set with a CAS LIBNAME engine libref.
OPTIONS	Lists the current settings of SAS system options in the SAS log.	No
PRINT	Prints SAS data sets and CAS tables.	No. Provides data transfer if the DATA= option specifies a data set with a CAS LIBNAME engine libref.
PRINTTO	Redirects output.	No
PRODUCT_STATUS	Returns a list of the SAS Foundation products that are installed on your system, along with the version numbers of those products.	No
PWENCODE	Encodes passwords.	No
REGISTRY	Maintains the SAS registry.	No
REPORT	Combines features of the PRINT, MEANS, and TABULATE procedures with features of the DATA step in a single report-writing tool that can produce a variety of reports.	Yes, if the DATA= option specifies a data set with a CAS LIBNAME engine libref.
S3	Performs object management for objects in Amazon Simple Storage Service (Amazon S3).	No
SCOREACCEL	Provides an interface to the CAS server for DATA step and DS2 model publishing and scoring.	Yes
SGPANEL	Creates a panel of graph cells for the values of one or more classification variables.	No Graphing large tables can cause performance degradations, You can summarize or reduce large amounts of data before attempting to graph it. You can limit the number of rows to graph by using the OBSMAX= option to indicate the maximum number of rows to process.
SGPLOT	Creates statistical graphics such as histograms and regression plots, in addition to simple graphics such as scatter plots and line plots.	No Graphing large tables can cause performance degradations, You can summarize or reduce large amounts of data before you attempt to graph it. You can limit the number of rows to graph by using the OBSMAX= option to indicate the maximum number of rows to process.

Procedure	Description	Uses CAS Processing
SGRENDER	Produces graphical output from templates that are created with the Graph Template Language (GTL).	No Graphing large tables can cause performance degradations, You can summarize or reduce large amounts of data before you attempt to graph it. You can limit the number of rows to graph by using the OBSMAX= option to indicate the maximum number of rows to process.
SGSCATTER	Creates a paneled graph of scatter plots for multiple combinations of variables.	No Graphing large tables can cause performance degradations, You can summarize or reduce large amounts of data before attempting to graph it. You can limit the number of rows to graph by using the OBSMAX= option to indicate the maximum number of rows to process.
SORT	Sorts SAS data sets.	No
SQL	Runs SAS SQL.	No
STREAM	Enables you to process an input stream that consists of arbitrary text and that can contain SAS macro specifications.	No
SUMMARY	Provides data summarization tools that compute descriptive statistics for variables across all observations or within groups of observations.	Yes, if the DATA= option specifies a data set with a CAS LIBNAME engine libref.
TABULATE	Displays descriptive statistics in tabular format, using some or all of the variables in a data set.	Yes, if the DATA= option specifies a data set with a CAS LIBNAME engine libref.
TEMPLATE (see *SAS Output Delivery System: Procedures Guide*)	Enables you to customize the appearance of your SAS output.	No
TRANSPOSE	Creates a CAS table by restructuring the values in a CAS table and transposing selected columns into rows.	If the DATA= and OUT= options point to CAS, the transposition is performed within CAS by invoking the CAS transpose action.

PROC CASUTIL

The CASUTIL procedure works with tables in SAS Cloud Analytic Services, SAS data sets in SAS libraries, and external files. Here is a list of the table functions and caslib management functions that are part of PROC CASUTIL:

- display table metadata
- delete files from a data source

- unload a table from a caslib
- lists tables in a caslib
- load tables to a caslib
- promote a table to global scope
- save a table in a caslib to a data source

See Also

- "Use PROC CASUTIL to Load Data" on page 15
- "CASUTIL Procedure" in *SAS Cloud Analytic Services: User's Guide*

CAS Language (CASL) and CAS Actions

CAS Language (CASL)

CASL is a scripting language that you use to prepare arguments for execution of CAS actions, submit actions to the CAS server, and then process action results. Action results can be evaluated, manipulated, and used as an argument to another action. You can also use CASL to create user-defined actions. For detailed information about CASL programming, see *CASL Programmer's Guide* and *SAS Cloud Analytic Services: CASL Reference*.

CASL code can be submitted to the CAS server in various ways:

- from a SAS client by using PROC CAS
- from Python, Lua, and R using the runCASL action
- from the CAS server when a user-defined action is executed

This simple program uses PROC CAS to load the file Iris.sashdat to the active caslib (default) to perform a correlation:

```
proc cas;                                                                       /* 1 */
   table.loadtable / path="iris.sashdat" casOut={name="IRIS"};                  /* 2 */
   simple.correlation result=x
      / table={groupBy={"Species"}, name="IRIS", orderBy={"SepalLength"}};      /* 3 */
run;
   print x;                                                                     /* 4 */
run;
```

1 The PROC CAS statement enables SAS to interpret and execute CASL statements.

2 The loadTable action loads the file iris.sashdat to the in-memory table IRIS by using the active caslib. To define parameters such as casOut, use {}.

3 The simple.correlation action measures the linear correlation between two variables. The groubBy parameter enables you to group information in the output according to the values of one or more variables. The orderBy parameter specifies the columns by which to order the data within each group. The results are placed in the CASL variable that is named x. By placing the results in a variable, you can use the results later in the program by using the x variable.

4 The PRINT statement prints the output data sets from variable x. If you did not assign the results to variable x, the output would appear in SAS Studio on the **RESULTS** tab without the PRINT statement.

Here is a simple use of PROC CAS with CASL that uses the CASAUTO session to list CSV files from the active caslib:

```
proc cas;
   session casauto;
   table.fileinfo / path="%.csv";
run;
```

The runCASL action enables you to submit a CASL program to the server as an action. The code is run in a single subsession that is started on the server or in parallel subsessions that run at the same time. By using the CREATE_PARALLEL_SESSION function, the CASL program runs faster using parallel sessions. For details and an example of using the runCASL action, see "CASL Server Action Set Details" in *SAS Viya: System Programming Guide*.

By using CASL in user-defined actions, a CASL program can be written once and run by many users. This CASL server-side processing reduces the need to share or access files that store common code. For more information, see "Writing User-Defined Actions" in *CASL Programmer's Guide*.

See Also

- "CAS Procedure" in *SAS Cloud Analytic Services: CASL Reference*
- table.loadtable action

CAS Actions

CAS actions perform a single task. CAS actions are organized with other actions in an action set and usually contain actions that are based on common functionality. You specify an action using the PROC CAS ACTION statement, in which only the action and action parameters are required. The keyword ACTION and the action set name are optional, as shown in the following examples of valid syntax for the fileInfo action in "CAS Language (CASL) ":

- `fileinfo / path="%.csv";`
- `table.fileinfo / path="%.csv";`
- `action table.fileinfo / path="%.csv";`

This example investigates how Major League Baseball (MLB) players' salaries depend on performance measures for the players' previous season and MLB career. To address a concern that some players who are outliers could dominate your least squares analysis, you can use the quantreg action to obtain a median regression model.

```
/*In the SAS Windowing Environment, you must specify the connection information*/
option casport=your-port-number cashost="cloud.example.com";
cas casauto sessopts=(caslib="casuser");                         /* 1 */

proc casutil;
   load data=sashelp.baseball;                                   /* 2 */
quit;

proc cas;                                                        /* 3 */
   quantreg.quantreg                                             /* 4 */
      table={name='baseball'},
      class={'league','division'},
      model={depvars='Salary',
             effects={'nAtBat', 'nHits', 'nHome', 'nRuns',
                      'nRBI', 'nBB', 'yrMajor', 'crAtBat',
                      'crHits', 'crHome', 'crRuns', 'crRbi',
                      'crBB', 'league', 'division', 'nOuts',
                      'nAssts', 'nError'}};
run;
```

1 Starts a CAS session named Casauto. The SESSOPTS= option is used to set the active caslib to Casuser explicitly.

2 The CASUTIL procedure transfers the Baseball data set from the Sashelp library to the server. The server makes the data available as an in-memory table that is named Baseball in the active caslib.

3 CAS action programming using CASL begins with the PROC CAS statement.

4 Use the quantreg action to fit a quantile regression model. When you refer to a table in a caslib within action code, you specify only the table name, not the libref. The class variables are League and Division, and the dependent variable is Salary.

To continue exploring this example and to view the example output, see the quantreg action.

To begin learning about programming with CAS actions, you can review one of the following Getting Started documents:

- *Getting Started with CASL Programming*
- *Getting Started with SAS Viya for Python*
- *Getting Started with SAS Viya for Lua*
- *Getting Started with SAS Viya for R*

Action sets are documented by product or component in the following publications:

Document	**Action Set Types**
SAS Visual Data Mining and Machine Learning: Programming Guide	Machine Learning
SAS Visual Data Mining and Machine Learning: Deep Learning Programming Guide	Deep Learning
SAS Visual Text Analytics: Programming Guide	Text Mining
SAS Visual Statistics: Programming Guide	Statistics
SAS Viya: System Programming Guide	Access Control FCMP Language: DATA Step, DS2, FedSQL, CASL Server Properties Streaming Data Session Methods Session Properties System Table Transpose

Document	Action Set Types
SAS Visual Analytics: Programming Guide	Aggregation Decision Tree Hypergroup Data Preprocessing Analytics
SAS Econometrics 8.4: Programming Guide	Regression models Dependency structure of multivariate distributions Linear econometric models Univariate limited dependent variable models Estimation of parameters of probability distributions
SAS Visual Forecasting 8.4: Programming Guide	Reconcile forecasts Short-time Fourier transform of a time series Smoothing spectra Automatic time series model identification and forecasting Scripted time series analysis Time-stamped data accumulation for time series
SAS Optimization 8.4: Network Optimization Programming Guide	Graph theory and network optimization algorithms Clique subgraphs to create a complete graph Connected graph components Cycle nodes Combinatorial optimization Sending flow over a network at minimal cost
SAS Optimization 8.4: Mathematical Optimization Programming Guide	Solving linear problems General mixed integer linear programs (MILPS) Quadratic programs Solver techniques and algorithms

See Actions and Action Sets by Name and Product for quick access to CAS action documentation.

DATA Step

The DATA step can run in your SAS session as well as in the CAS server. In CAS, the DATA step runs in a CAS server session on a single server (called symmetric multiprocessing, or SMP) or across multiple computers in parallel (called massively parallel processing, or MPP). By default, the DATA step runs in multiple threads. When the DATA step runs in multiple threads, the same DATA step program runs in multiple threads on all nodes in the environment. For more information, see "Processing Modes" in *SAS Cloud Analytic Services: DATA Step Programming*.

When in-memory CAS tables are used exclusively, the DATA step takes advantage of the in-memory speeds. Some language element restrictions and VARCHAR data type restrictions apply when the DATA step runs in the CAS server.

To process data in CAS, you load the data to a caslib.

The following example demonstrates how to use the DATA step to load data into CAS. :

```
/* In the SAS Windowing Environment, you must specify the connection information */
option casport=5570 cashost="cloud.example.com";
cas casauto;                              /* 1 */
libname mycas cas caslib=casuser;         /* 2 */
data mycas.cars (where=(weight>6000));    /* 3 */
   set sashelp.cars;
   keep make model type weight MPG_City;
run;
```

Note that the example DATA step above runs in SAS and not in CAS server. When the DATA step is used to load data to CAS, it runs in SAS. For more information about what it means for the DATA step to run in CAS, see "Running the DATA Step in CAS" in *SAS Cloud Analytic Services: DATA Step Programming*.

The following DATA step runs in the CAS server. When the input and output librefs use the CAS LIBNAME engine, the DATA step runs in the CAS server.

```
 data mycas.cars2;                        /* 4 */
   set mycas.cars;
   if mpg_city > 25 then eff='Y';
      else eff='N';                       /* 5 */
   put 'Thread number: '  _threadid_      /* 6 */
       'on worker node '  _hostname_;
run;
```

1 Start a session with the CAS server.

2 Create the libref Mycas and bind it to the Casuser caslib. The CASLIB= option in the LIBNAME statement for the CAS engine binds the MyCas library to the Casuser caslib.

3 Load a subset of the sashelp.cars data set to the Casuser caslib.

4 Create the table cars2 in the Casuser caslib from the CAS table Cars. The SET statement runs in SAS. The SET statement loads a subset of the Sashelp.Cars data set (cars for which the weight is greater than 6000 pounds) to an in-memory table.

5 Specify whether the cars in the cars2 table run efficiently.

6 Write the thread and worker node where the data was processed.

See "DATA Step Basics" in *SAS Cloud Analytic Services: DATA Step Programming* for an overview of running the DATA step in CAS.

Not all DATA step language elements that you used in previous releases of SAS are appropriate for distributed processing in the CAS environment. For a list of language elements that run on the CAS server, see the category table in the respective documentation:

- Hash objects and Java objects
- Data Set Options
- Formats
- Functions
- Statements: DATA step and global

If the language element specifies a category of CAS in the documentation, it can run in the CAS server.

Some restrictions also apply when you use the DATA step in the CAS server. For a list of these restrictions, see the following topics:

- "Restrictions" in *SAS Cloud Analytic Services: DATA Step Programming*

- “Restrictions for the VARCHAR Data Type in the CAS Engine” in *SAS Cloud Analytic Services: User’s Guide*

The BY statement in a SAS DATA step divides table data into groups of rows that share the same values for the BY variables. When you use the BY statement in a distributed server, CAS groups the rows based on the first BY variable. When CAS distributes the table across multiple nodes, it keeps the BY groups intact. That is, rows that share the same BY variables are stored together on a single machine. For information about BY processing behavior in CAS, see the following topics:

- “How CAS Groups Data with BY Variables” in *SAS Cloud Analytic Services: DATA Step Programming*
- “How BY Variables Affect Multithreaded DATA Step Execution” in *SAS Cloud Analytic Services: DATA Step Programming*

Note: You can also submit DATA step code by using the CAS action dataStep.runCode in PROC CAS.

Video: Using the DATA Step in SAS Viya

Video: Processing Data in Groups with the DATA Step in SAS Viya

SAS Viya Files Service

The SAS Viya Files service enables you to store, retrieve, and delete files maintained in the SAS Infrastructure Data Server database repository. The repository is not considered a complete 'file system'. Rather, the repository contains individual files that are directly accessible by their file identifier. This file identifier contains a universally unique identifier (UUID) generated by the Files service when a file is created.

The Files service also assigns a unique name to each file in the repository, but it is not human-friendly name. A user can change the name, but there is a risk that the name might not be unique within the repository.

You can access a file in the Files service using the file identifier. The file identifier is contained in the URI stored in the file information. Use the system-generated name or user-assigned name to find the file URI and the file identifier. Once you have the file identifier, you can then use it to access the file directly in the Files service.

The Files service does not have a concept of 'folders' in its repository. However, you can associate files using a PARENTURI= option. A PARENTURI= option specifies a relative URI for any object in SAS Viya. You can create a collection of files by specifying the same value of the PARENTURI= option for each file.

This example creates the file class.csv and attaches the file to a job named in the fileref Jobout.

```
filename jobout filesrvc
   parenturi='/jobExecution/jobs/5a308aa7-1c3a-4465-a14c-fd69a9091926';
data _null_;
   set sashelp.class;
   file jobout('class.csv');
   put name "," sex "," age "," height "," weight;
run;
```

For more information, see “FILENAME Statement, FILESRVC Access Method” in *SAS Global Statements: Reference*.

Macro Language

Macros are supported in SAS Viya, but only in the SAS session. Macros can be useful in generating code and submitting the code to the CAS server to run. However, the macro language itself does not run on the CAS server.

See Also

- *SAS Macro Language: Reference*

DS2

DS2 runs in SAS Viya as it has in previous releases of SAS.

A DS2 program can perform manipulations on multiple data rows concurrently, thus reducing the time that is required to process big data sets. Based on the structure of the DS2 program, the DS2 compiler determines which observations can be processed sequentially and which observations can be processed in parallel. A DS2 program is classified as either a serial program, parallel program, or parallel-serial program.

During execution of a DS2 program on a CAS server, nodes might be active or passive. An active node processes observations. Active nodes read, manipulate, and write data. A passive node does not process observations. When serial operations are processed, one and only one node processes each observation sequentially. This special node is the principal worker. The controller cannot be the principal, because the controller is unable to read or write data. Therefore, a worker is selected as the principal worker. What occurs when a DS2 program executes in CAS depends on the classification of the DS2 program.

For more information, see “How DS2 Runs in CAS” in *SAS DS2 Programmer’s Guide*.

Using a FedSQL query within your DS2 program is not currently supported on the CAS server, with one exception. A FedSQL query is allowed in the SET statement in a DS2 action or PROC DS2.

In addition to using PROC DS2 to run DS2 code, you can also use the ds2.runDS2 action by using PROC CAS or a supported third-party language. You can also specify FedSQL statements in the DS2 SET statement in SAS Viya 3.4.

Also, in SAS Viya 3.4, DS2 has the ability to publish and run DATA step and DS2 models in CAS or in Hadoop or Teradata by using CAS actions or PROC SCOREACCEL.

In SAS Viya 3.4, scoring models in Hadoop can be run with either MapReduce or the Spark2 engine.

DS2 supports the same data sources in SAS Viya as it does in SAS 9.4, with two exceptions. Beginning in SAS Viya 3.4, DS2 supports Apache Spark and JDBC-compliant databases. You can access the data sources through the SAS Workspace Server or the SAS Compute Server by using SAS/ACCESS software. You can access the data sources from the CAS server with SAS data connectors.

See Also

- *SAS DS2 Programmer’s Guide*
- *SAS DS2 Language Reference*

FedSQL

SAS FedSQL is a SAS proprietary implementation of the ANSI SQL:1999 core standard. It provides support for new data types and other ANSI 1999 core compliance features and proprietary extensions. FedSQL provides a scalable, threaded, high-performance way to access, manage, and share relational data from multiple data sources. When possible, FedSQL queries are optimized with multithreaded algorithms in order to resolve large-scale operations. For applications, FedSQL provides a common SQL syntax across all data sources. That is, FedSQL is a vendor-neutral SQL dialect that accesses data from various data sources without having to submit queries in the SQL dialect that is specific to the data source. In addition, a single FedSQL query can target data in several data sources and return a single result set.

When FedSQL requests are submitted through the SAS Workspace Server or the SAS Compute Server, FedSQL supports the same statements in SAS Viya as it does in previous releases of SAS. FedSQL uses its extended data type support to process the third-party data. FedSQL supports full and partial implicit SQL pass-through. It supports explicit SQL pass-through in the CONNECTION TO component of the SELECT statement and via the EXECUTE statement.

On the CAS server, only the following statements are supported: CREATE TABLE with the AS expression, SELECT, and DROP TABLE. FedSQL supports full-query implicit SQL pass-through in CAS. That is, the entire request must be capable of being passed to the data source for processing. Beginning with SAS Viya 3.4, FedSQL supports limited explicit SQL pass-through. You can submit native SQL syntax in a CONNECTION TO component of the FedSQL SELECT statement FROM clause. Data processed on the CAS server uses CAS data types.

PROC FEDSQL submits requests to the SAS Workspace Server or the SAS Compute Server by default. You assign a SAS library reference (libref) to access SAS and third-party data. You reference tables in FedSQL statements by using a two-part name in the form libref.table-name. If you refer to tables simply by table-name, the procedure assumes that you are using the SAS Work library, which persists only for the duration of the SAS session.

To submit a FedSQL request to the CAS server, use PROC FEDSQL with the SESSREF= procedure option. The SESSREF= procedure option directs the request to a CAS session. The CAS session must have been previously established. You assign a CAS library reference (caslib) to access CAS, SAS, and third-party data. You can use a one-part table name or a two-part table name in the form caslib.table-name to reference tables. In CAS, when you use a one-part name, FedSQL looks for a table of that name in the active caslib, and uses it if one is found. A two-part name in the form caslib.table-name is necessary when accessing tables from multiple caslibs in a single request.

For detailed information about the FedSQL functionality that is supported for SAS libraries, see *SAS FedSQL Language Reference*. For detailed information about the FedSQL functionality that is supported in CAS libraries, see *SAS Viya: FedSQL Programming for SAS Cloud Analytic Services*.

FedSQL supports the same data sources in SAS Viya as it does in SAS 9.4, with an exception. Beginning in SAS Viya 3.4, FedSQL supports Apache Spark. You can access the data sources through the SAS Workspace Server or the SAS Compute Server by using SAS/ACCESS software. You can access the data sources from the CAS server with SAS data connectors.

See Also

- "CAS Procedure" in *SAS Cloud Analytic Services: CASL Reference*
- "FEDSQL Procedure" in *Base SAS Procedures Guide*
- "FedSQL Action Set" in *SAS Viya: System Programming Guide*
- "Dynamically Executing FedSQL Statements from DS2" in *SAS DS2 Programmer's Guide*

User-Defined Formats

You can store user-defined formats in SAS Viya in catalogs to use in a SAS session, or you can store them in a format library on the CAS server. Format libraries are associated with a CAS session, or they can be promoted to global scope to be available to all CAS sessions. User-defined formats in a format library are server-side formats that the server uses when an analysis is performed according to formatted values. You can migrate existing user-defined formats from SAS to SAS Viya. For more information, see *SAS Cloud Analytic Services: User-Defined Formats*.

You still use PROC FORMAT to create formats. A new option, CASFMTLIB=, names the CAS format library for the caslib where the format is stored. Without the CASFMTLIB= option, formats continue to be stored in a format catalog. PROC FMT2ITM converts one or more format catalogs into a single item store that can be used as input to the addFmtLib action, which adds a CAS format library.

It is a best practice to assign a format to a variable before the table is loaded into the server. After a table is in memory, some procedures cannot assign a format. You can assign a format with the FORMAT statement in a DATA step or with the CASUTIL procedure.

This example adds and saves a user-defined format:

```
/*In the SAS Windowing Environment, you must specify the connection information*/
options casport=5570 cashost="cloud.example.com";
cas casauto sessopts=(caslib="casuser");
proc format library=work.formats casfmtlib="casformats";               /* 1 */
  value enginesize
    low - <2.7 = "Very economical"
    2.7 - <4.1 = "Small"
    4.1 - <5.5 = "Medium"
    5.5 - <6.9 = "Large"
    6.9 - high = "Very large"
;

cas casauto savefmtlib fmtlibname=casformats                           /* 2 */
   table=enginefmt replace;

proc casutil;
  format enginesize enginesize.;                                       /* 3 */
  load data=sashelp.cars casout="cars" replace;
  contents casdata="cars";
quit;
libname mycas cas;                                                     /* 4 */

proc mdsummary data=mycas.cars;
  var mpg_highway;
  groupby enginesize / out=mycas.mpg_hwy_by_size;                      /* 5 */
run;

proc print data=mycas.mpg_hwy_by_size;
  var enginesize--_mean_;
run;
```

1 The FORMAT procedure creates a format that is named Enginesize. On the SAS client, the format is temporary and is stored in the Work library. The CASFMTLIB= option adds the same format to your CAS session in a format library that is named Casformats.

2 The CAS statement with the SAVEFMTLIB option persists the format library with member Enginesize as a SASHDAT file in the data source that is associated with the active caslib. The Casuser caslib is typically a PATH type and uses a file system. The TABLE= option specifies the name and results in a file that is named enginefmt.sashdat. This enables you to load a format library from a table in future sessions.

3 The FORMAT statement with the CASUTIL procedure assigns the Enginesize format to the Enginesize variable. The format is applied to the in-memory instance of the Cars table.

4 A CAS LIBNAME engine libref is assigned so that SAS procedures can access the Cars table that is in memory on the server. The assignment does not specify a caslib, so the active caslib is used.

5 The MDSUMMARY procedure provides descriptive statistics for the Cars table. The results are grouped by the formatted values of the Enginesize variable (five values) instead of by the numeric value. As a result, the output table, Mpg_hwy_by_size, has five rows.

Video: Creating User-Defined Formats with the FORMAT Procedure in SAS Viya

See Also

- *SAS Cloud Analytic Services: User-Defined Formats*