

SAS[®] Viya[®]

The R Perspective



Yue Qi
Kevin D. Smith
Xiangxiang Meng

The correct bibliographic citation for this manual is as follows: Qi, Yue, Kevin D. Smith, and Xiangxiang Meng. 2018. *SAS® Viya: The R Perspective*. Cary, NC: SAS Institute Inc.

SAS® Viya: The R Perspective

Copyright © 2018, SAS Institute Inc., Cary, NC, USA

ISBN 978-1-63526-704-4 (Hardcopy)

All rights reserved. Produced in the United States of America.

For a hard-copy book: No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

For a web download or e-book: Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

The scanning, uploading, and distribution of this book via the Internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

U.S. Government License Rights; Restricted Rights: The Software and its documentation is commercial computer software developed at private expense and is provided with RESTRICTED RIGHTS to the United States Government. Use, duplication or disclosure of the Software by the United States Government is subject to the license terms of this Agreement pursuant to, as applicable, FAR 12.212, DFAR 227.7202-1(a), DFAR 227.7202-3(a) and DFAR 227.7202-4 and, to the extent required under U.S. federal law, the minimum restricted rights as set out in FAR 52.227-19 (DEC 2007). If FAR 52.227-19 is applicable, this provision serves as notice under clause (c) thereof and no other notice is required to be affixed to the Software or documentation. The Government's rights in Software and documentation shall be only those set forth in this Agreement.

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513-2414.

August 2018

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

Contents

Chapter 1: Installing R, SAS SWAT, and CAS

Chapter 2: The Ten-Minute Guide to Using CAS from R

Chapter 3: The Fundamentals of Using R with CAS

Chapter 4: Managing Your Data in CAS

Chapter 5: First Steps with the CASTable Object

Chapter 6: Working with CAS Tables

Chapter 7: Data Exploration and Summary Statistics

Chapter 8: Modeling Continuous Variables

Chapter 9: Modeling Categorical Variables

Chapter 10: Advanced Topics

About This Book

Description

SAS® Viya®: The R Perspective is a general-purpose introduction to using R with the SAS Viya platform. SAS Viya is a high-performance, fault-tolerant analytics architecture that can be deployed on both public and private cloud infrastructures. This book introduces an entirely new way of using SAS statistics from R, taking users step-by-step from installation and fundamentals to data exploration and modeling

Author These Authors

Yue Qi, PhD, is a data scientist at SAS. He has deep expertise in predictive modeling and machine learning. His current work includes interactive visualization and modeling of big data, providing innovative insights for big data analytics, and parallel and distributed computing platform development. His research interests include predictive modeling and machine learning for big data and high dimensional data, distributed and parallel computing, and anomaly detection. Yue received his PhD in Statistics from the University of Missouri and is currently based in Cary, NC.

Kevin D. Smith has been a software developer at SAS since 1997. He has been involved in the development of PROC TEMPLATE and other underlying ODS technologies for most of his tenure. He has spoken at numerous SAS Global Forum conferences, as well as at regional and local SAS users groups with the "From Scratch" series of presentations that were created to help users of any level master various ODS technologies. More recently, he has been involved in the creation of the scripting language interfaces to SAS Cloud Analytic Services on the SAS Viya platform.

Xiangxiang Meng, PhD, is a Senior Product Manager at SAS. The current focus of his work is on SAS® Visual Statistics, cognitive computing, the Python interface to SAS Cloud Analytic Services, and other new product initiatives. Previously, Xiangxiang worked on SAS® LASRTM Analytic Server, SAS® In-Memory Statistics for Hadoop, SAS Recommendation Systems, and SAS® Enterprise Miner™. His research interests include decision trees and tree ensemble models, automated and cognitive pipelines for business intelligence and machine learning, and parallelization of machine learning algorithms on distributed data. Xiangxiang received his PhD and MS from the University of Cincinnati.

Chapter 2: The Ten-Minute Guide to Using CAS from R

Loading SWAT and Getting Connected	5
Running CAS Actions.....	8
Loading Data	8
Executing Actions on CAS Tables	10
Data Visualization	11
Closing the Connection	11
Conclusion.....	12

If you are already familiar with R, have a running CAS server, and just can't wait to get started, we've written this chapter just for you. This chapter is a very quick summary of what you can do with CAS from R. We don't provide a lot of explanation of the examples; that comes in the later chapters. This chapter is here for those who want to dive in and work through the details in the rest of the book as needed.

Loading SWAT and Getting Connected

The only thing that you need to know about the CAS server in order to get connected is the host name, the port number, your user name, and your password. The last two items might even be optional if you are using an Authinfo file, which is explained in detail in Chapter 3. The SWAT package contains the CAS class that is used to talk to the server. The arguments to the CAS class are host name, port, user name, and password, in that order.¹ Note that you can use the REST interface by specifying the HTTP port that is specified by the CAS server. The CAS class can auto detect the port type for the standard CAS port and HTTP. However, if you use HTTPS, you must specify `protocol='https'` as a keyword argument when you start a CAS connection. You can also specify `'cas'` or `'http'` to explicitly override auto detection.

```
> library('swat')
```

```
SWAT 0.1.4
```

```
> conn <- CAS('server-name.mycompany.com', 5570, 'username', 'password')
```

```
Connecting to CAS and generating CAS action functions for loaded action sets...  
To generate the functions with signatures (for tab completion), add  
'genActSyntax=TRUE' to your connection parms.
```

When you connect to CAS, it creates a session on the server. By default, all resources (CAS actions, data tables, options, and so on) are available only to that session. Some resources can be promoted to a global scope, which we discuss later in the book.

6 SAS Viya: The R Perspective

To see what CAS actions are available, use the help method on the CAS connection object, which calls the help action in builtins action set on the CAS server.

```
> out <- cas.builtins.help(conn)
```

```
NOTE: Available Action Sets and Actions:
NOTE:   accessControl
NOTE:     assumeRole - Assumes a role
NOTE:     dropRole - Relinquishes a role
NOTE:     showRolesIn - Shows the currently active role
NOTE:     showRolesAllowed - Shows the roles that a user is a member
NOTE:                          of
NOTE:     isInRole - Shows whether a role is assumed
NOTE:     isAuthorized - Shows whether access is authorized
NOTE:     isAuthorizedActions - Shows whether access is authorized to
NOTE:                          actions
NOTE:     isAuthorizedTables - Shows whether access is authorized to
NOTE:                          tables
NOTE:     isAuthorizedColumns - Shows whether access is authorized to
NOTE:                          columns
NOTE:     listAllPrincipals - Lists all principals that have explicit
NOTE:                          access controls
NOTE:     whatIsEffective - Lists effective access and explanations
NOTE:                          (Origins)
NOTE:     ...
NOTE:     partition - Partitions a table
NOTE:     shuffle - Randomly shuffles a table
NOTE:     recordCount - Shows the number of rows in a Cloud Analytic
NOTE:                          Services table
NOTE:     loadDataSource - Loads one or more data source interfaces
NOTE:     update - Updates rows in a table
```

The return values from all actions are in the form of the R list class. To see a list of names of all of the list members, use the names() function just as you would with any R list. In this case, the object names correspond to the names of the CAS action sets.

```
> names(out)
 [1] "accessControl" "builtins"      "configuration"
 [4] "dataPreprocess" "dataStep"     "percentile"
 [7] "search"        "session"      "sessionProp"
[10] "simple"         "table"
```

Printing the contents of the return value shows all of the top-level list members as sections. The builtins.help action returns the information about each action set in a table. These tables are stored in the output as casDataFrames.

```
> out
$accessControl
```

	Name	Description
1	assumeRole	Assumes a role
2	dropRole	Relinquishes a role
3	showRolesIn	Shows the currently active role
4	showRolesAllowed	Shows the roles that a user is a member of
5	isInRole	Shows whether a role is assumed

	Name	Description
6	isAuthorized	Shows whether access is authorized
7	isAuthorizedActions	Shows whether access is authorized to actions
8	isAuthorizedTables	Shows whether access is authorized to tables
9	isAuthorizedColumns	Shows whether access is authorized to columns
10	listAllPrincipals	Lists all principals that have explicit access controls
...
20	partition	Partitions a table
21	shuffle	Randomly shuffles a table
22	recordCount	Shows the number of rows in a Cloud Analytic Services table
23	loadDataSource	Loads one or more data source interfaces
24	update	Updates rows in a table

Since the output is based on R's list object, you can access each list member individually as well.

```
> out$builtins
```

	Name	Description
1	addNode	Adds a machine to the server
2	removeNode	Remove one or more machines from the server
3	help	Shows the parameters for an action or lists all available actions
4	listNodes	Shows the host names used by the server
5	loadActionSet	Loads an action set for use in this session
6	installActionSet	Loads an action set in new sessions automatically
7	log	Shows and modifies logging levels
8	queryActionSet	Shows whether an action set is loaded
9	queryName	Checks whether a name is an action or action set name
10	reflect	Shows detailed parameter information for an action or all actions in an action set
11	serverStatus	Shows the status of the server
12	about	Shows the status of the server
13	shutdown	Shuts down the server
14	userInfo	Shows the user information for your connection
15	actionSetInfo	Shows the build information from loaded action sets
16	history	Shows the actions that were run in this session
17	casCommon	Provides parameters that are common to many actions
18	ping	Sends a single request to the server to confirm that the connection is working
19	echo	Prints the supplied parameters to the client log
20	modifyQueue	Modifies the action response queue settings
21	getLicenseInfo	Shows the license information for a SAS product
22	refreshLicense	Refresh SAS license information from a file
23	httpAddress	Shows the HTTP address for the server monitor

Running CAS Actions

Just like the `builtins.help` action, all of the actions are available as R functions. You need to specify the fully qualified name of the action, which includes both the action set name and the action name. For example, the `userInfo` action is contained in the `builtins` action set. To call it, you have to use the full name `cas.builtins.userInfo`. Note that both the action set name and the action name are always written in camelCase.

For example, the `userInfo` action is called as follows.

```
> cas.builtins.userInfo(conn)
```

```
$userInfo
$userInfo$anonymous
[1] FALSE

$userInfo$groups
$userInfo$groups[[1]]
[1] "users"

$userInfo$hostAccount
[1] TRUE

$userInfo$providedName
[1] "username"

$userInfo$providerName
[1] "Active Directory"

$userInfo$uniqueId
[1] "username"

$userInfo$userId
[1] "username"
```

The result this time is still a list object, and the contents of that object is another list (`userInfo`) that contains information about your user account. Although all actions return a list object, there are no strict rules about what member names and values are in that object. The returned values are determined by the action and they vary depending on the type of information returned. Analytic actions typically return one or more `casDataFrames`.

Loading Data

The easiest way to load data into a CAS server is by using the `as.casTable()` function. This function uploads the data from an R `data.frame` to a CAS table. We use the classic Iris data set in the following data-loading example.

```
> iris_ct <- as.casTable(conn,iris)
> attributes(iris_ct)
```

```
$conn
```

```

CAS(hostname=server-name.mycompany.com, port=5570, username=username,
session=60c6e0fc-d690-ea48-9dbc-9692e7205455, protocol=http)

$name
[1] "iris"

$caslib
[1] ""

$where
[1] ""

$orderby
[1] ""

$groupby
[1] ""

$gbmode
[1] ""

$computedOnDemand
[1] FALSE

$computedVars
[1] ""

$computedVarsProgram
[1] ""

$names
[1] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" "Species"

$class
[1] "CASTable"
attr(,"package")
[1] "swat"

```

The output from the `as.casTable()` function is a `CASTable` object. The `CASTable` object contains the connection information, name of the created table, the `caslib` that the table was created in, and other information. The `CASTable` objects also support many of the operations that are defined by R `data.frame` so that you can operate on them as if they were local data.²

You can use actions such as `tableInfo` and `columnInfo` in the table action set to access general information about the table itself and its columns.

```

# Call the tableInfo action on the CASTable object.
> cas.table.tableInfo(conn)

```

```

  Name Rows Columns Encoding CreateTimeFormatted  ModTimeFormatted
1 IRIS  150      5  utf-8  17Apr2017:02:17:40 17Apr2017:02:17:40
  JavaCharSet CreateTime  ModTime Global Repeated View SourceName
1      UTF8 1808014660 1808014660      0      0      0
  SourceCaslib Compressed Creator Modifier
1              0 username

```

```

# Call the columnInfo action on the CASTable.

```

```
> cas.table.columnInfo(iris_ct)
$ColumnInfo
      Column ID      Type RawLength FormattedLength NFL NFD
1 Sepal.Length 1  double         8             12  0  0
2 Sepal.Width  2  double         8             12  0  0
3 Petal.Length 3  double         8             12  0  0
4 Petal.Width  4  double         8             12  0  0
5 Species      5  varchar        10            10  0  0
```

Now that we have some data, let's run some more interesting CAS actions on it.

Executing Actions on CAS Tables

The simple action set that comes with CAS contains some basic analytic actions. Let's run the summary action from the simple action set on our CAS table.

```
> summ <- cas.simple.summary(iris_ct)
> summ
```

```
$Summary
      Column Min Max   N NMiss      Mean   Sum      Std
1 Sepal.Length 4.3 7.9 150     0 5.843333 876.5 0.8280661
2 Sepal.Width  2.0 4.4 150     0 3.057333 458.6 0.4358663
3 Petal.Length 1.0 6.9 150     0 3.758000 563.7 1.7652982
4 Petal.Width  0.1 2.5 150     0 1.199333 179.9 0.7622377
      StdErr      Var      USS      CSS      CV      TValue
1 0.06761132 0.6856935 5223.85 102.16833 14.17113 86.42537
2 0.03558833 0.1899794 1430.40  28.30693 14.25642 85.90830
3 0.14413600 3.1162779 2582.71 464.32540 46.97441 26.07260
4 0.06223645 0.5810063  302.33  86.56993 63.55511 19.27060
      ProbT
1 3.331256e-129
2 8.004458e-129
3 2.166017e-57
4 2.659021e-42
```

The summary action displays summary statistics in a form that is familiar to SAS users. If you want them in a form that is similar to what R users are used to, you can use the `summary()` method (just like on R `data.frame` objects).

```
> summary(iris_ct)
```

```
 Sepal.Length      Sepal.Width      Petal.Length      Petal.Width
Min.      :4.300   Min.      :2.000   Min.      :1.000   Min.      :0.100
1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
Median :5.800   Median :3.000   Median :4.350   Median :1.300
Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500

Species
setosa      :50
versicolor:50
virginica   :50
```

Note that when you call the `summary()` function on a `CASTable` object, it calls various CAS actions in the background to do the calculations. This includes the `cas.table.columnInfo`, `cas.simple.summary`,

`cas.percentile.percentile`, and `cas.fedsql.execDirect` actions. The output of those actions is combined into a `data.frame` in the same form that the real R `summary()` function returns. This enables you to use `CASTable` objects and R `data.frame` objects interchangeably in your workflow to work on the result tables from CAS.

Data Visualization

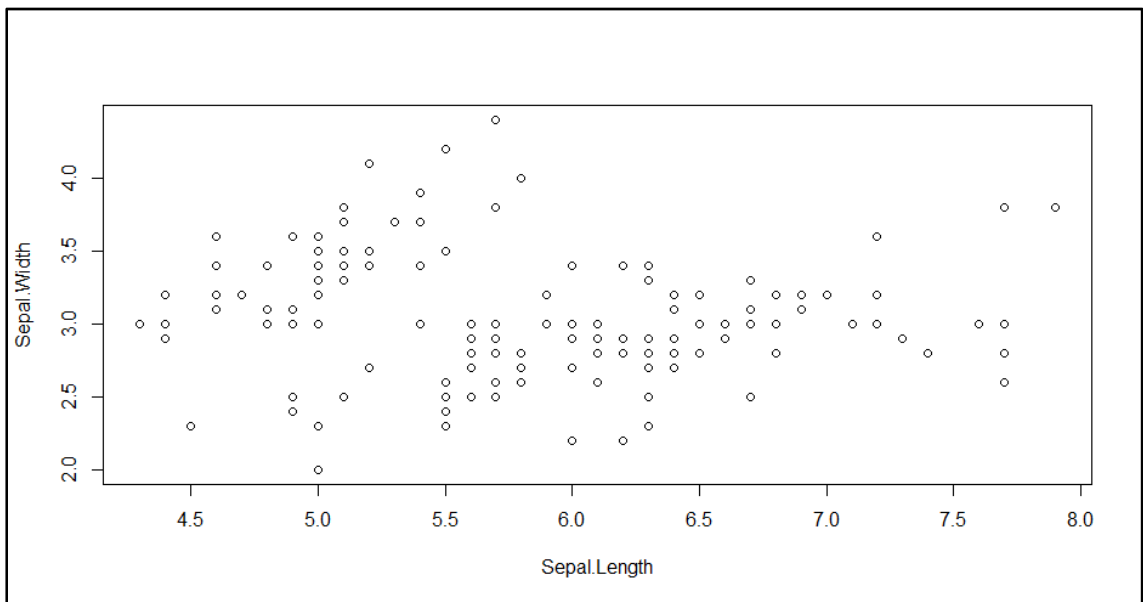
Since the tables that come back from the CAS server are subclasses of an R `data.frame`, you can do anything to them that works on a `data.frame`. You can plot the results of your actions using the `plot` function or use them as input to more advanced packages, such as `ggplot2`, which are covered in more detail in a later section.

The following example uses the `plot` method to download the data set and plot it using the default options.³

```
> plot(iris_ct$Sepal.Length, iris_ct$Sepal.Width)
```

The output that is created by the `plot` function is shown in Figure 2.1.

Figure 2.1: Scatter Plot of Sepal.Width versus Sepal.Length



Closing the Connection

As with any network or file resource in R, you should close your CAS connections when you are finished. They time out and disappear eventually if left open, but it's always a good idea to clean them up explicitly.

```
> cas.terminate(conn)
```

Conclusion

Hopefully, this ten-minute guide was enough to give you an idea of the basic workflow and capabilities of the R CAS client. In the following chapters, we dig deeper into the details of the R CAS client and how to blend the power of SAS analytics with the tools that are available in the R environment.

¹ Later in the book, we show you how to store your password so that you do not need to specify it in your programs.

² However, until you explicitly fetch the data or call a function that returns data from the table (such as head or tail), all operations are simply combined on the client side (essentially creating a client-side view) until they are needed for the call to the CAS server for data.

³ To prevent downloading very large data sets to the client, a maximum of only 10,000 rows can be randomly sampled and downloaded when the data set has more than 10,000 rows.

Ready to take your SAS[®] and JMP[®] skills up a notch?



Be among the first to know about new books,
special events, and exclusive discounts.

support.sas.com/newbooks

Share your expertise. Write a book with SAS.

support.sas.com/publish

 sas.com/books
for additional books and resources.


THE POWER TO KNOW.®

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies. © 2017 SAS Institute Inc. All rights reserved. M1588358 US.0217