Chapter 11

Calling Functions in the R Language

Contents

Overview of Calling Functions in the R Language

R is a freely available language and environment for statistical computing and graphics. Like the SAS/IML language, the R language has features suitable for developers of statistical algorithms: the ability to manipulate matrices and vectors, a large number of built-in functions for computing statistical quantities, and the capability to extend the basic function library by writing user-defined functions. There are also a large number of user-contributed packages in R that implement specialized computations.

In 2009, the SAS/IML Studio application introduced a mechanism for calling R functions from programs written in the IMLPlus language. As of SAS/IML 9.22, this feature is available in PROC IML. This chapter shows you how to call R functions from PROC IML by using the SUBMIT and ENDSUBMIT statements.

This chapter describes how to configure the SAS system so that you can call functions in the R language. The chapter also decribes how to do the following:

- transfer data to R
- call R functions from PROC IML
- transfer the results from R to a number of SAS data structures

Installing the R Statistical Software

SAS does not distribute R software. In order to call R software, you must first install R on the same computer that runs SAS software. If you access a SAS workspace server through client software such as SAS® Enterprise Guide®, then R must be installed on the SAS server.

You can download R from The Comprehensive R Archive Network Web site: `http://cran.r-project.org`. If you experience problems installing R, consult the R FAQ: `http://cran.r-project.org/doc/FAQ/R-FAQ.html`. SAS Technical Support does not provide support for installing or configuring third-party software.

In SAS/IML, the interface to R is supported on computers that run a 32-bit or 64-bit Windows operating system or Linux operating systems. If you are using SAS software in a 64-bit Linux environment, you must download a 64-bit binary distribution of R. Otherwise, download a 32-bit binary distribution.

The document “Installing R on Linux Operating Systems” is available on support.sas.comand includes pointers for installing R on Linux that it works with the SAS interface to R.

The RLANG System Option

The RLANG system option determines whether you have permission to call R from the SAS system. You can determine the value of the RLANG option by submitting the following SAS statements:

```
proc options option=RLANG;
run;
```

The result is one of the following statements in the SAS log:

NORLANG   Do not support access to R language interfaces
If the SAS log contains this statement, you do not have permission to call R from the SAS system.

RLANG Support access to R language interfaces
If the SAS log contains this statement, you can call R from the SAS system.

The RLANG option can be changed only at SAS start-up. In order to call R, the SAS system must be launched with the -RLANG option. (It is often convenient to insert this option in a SASV9.CFG file.) For security reasons, some system administrators configure the SAS system to start with the -NORLANG option. The RLANG option is similar to the XCMD option in that both options enable SAS users to potentially write or delete important data and system files.

If you attempt to submit R statements on a system that was not launched with the -RLANG option, you get the following error message:

```
ERROR: The RLANG system option must be specified in the SAS configuration file or on the
SAS invocation command line to enable the submission of R language statements.
```

Some operating systems do not support the RLANG system option. The RLANG system option is currently supported for the Windows and Linux operating systems. If you attempt to submit R statements on a host that does not support the RLANG option, you get the following warning message:

```
WARNING: SAS option RLANG is not supported on this host.
```

Submit R Statements

In order to call R from the SAS system, the R statistical software must be installed on the SAS workspace server and the RLANG system option must be enabled. (See "The RLANG System Option" on page 190.)

Chapter 10, "Submitting SAS Statements," describes how to submit SAS statements from PROC IML. Submitting R statements is similar. You use a SUBMIT statement, but add the R option: SUBMIT / R. All statements in the program between the SUBMIT statement and the next ENDSUBMIT statement are sent to R for execution. The ENDSUBMIT statement must appear on a line by itself.

The simplest program that calls R is one that does not transfer any data between the two environments. In the following program, SAS/IML is used to compute the product of a matrix and a vector. The result is printed. Then the SUBMIT statement with the R option is used to send an equivalent set of statements to R.

```
proc iml;
/* Comparison of matrix operations in IML and R */
print "----------  SAS/IML Results  -----------------";
x = 1:3;                                /* vector of sequence 1,2,3 */
m = {1 2 3, 4 5 6, 7 8 9};              /* 3 x 3 matrix */
q = m * t(x);                           /* matrix multiplication */
print q;
print "-------------  R Results  --------------------";
submit / R;
  rx <- matrix( 1:3, nrow=1)             # vector of sequence 1,2,3
  rm <- matrix( 1:9, nrow=3, byrow=TRUE) # 3 x 3 matrix
  rq <- rm %*% t(rx)                     # matrix multiplication
  print(rq)
endsubmit;
```

The printed output from R is automatically routed to the SAS/IML Studio output window, as shown in Figure 11.1. As expected, the result of the computation is the same in R as in SAS/IML.

Figure 11.1 Output from SAS/IML and R

```
                 ----------  SAS/IML Results  -----------------

                              q

                                 14
                                 32
                                 50

                 -------------  R Results  --------------------


     [,1]
[1,]   14
[2,]   32
[3,]   50
```

Transferring Data between SAS and R Software

Many research statisticians take advantage of special-purpose functions and packages written in the R language. When you call an R function, the data must be accessible to R, either in a data frame or in an R matrix. This section describes how you can transfer data and statistical results (for example, fitted values or parameter estimates) between SAS and R data structures.

You can transfer data to and from the following SAS data structures:

- a SAS data set in a libref
- a SAS/IML matrix

In addition, you can transfer data to and from the following R data structures:

- an R data frame
- an R matrix

Transfer from a SAS Source to an R Destination

Table 11.1 summarizes the subroutines that copy data from a SAS source to an R destination. For more information, see the section "Details of Data Transfer" on page 200.

Table 11.1 Transferring from a SAS Source to an R Destination

Subroutine	SAS Source	R Destination
ExportDataSetToR	SAS data set	R data frame
ExportMatrixToR	SAS/IML matrix	R matrix

As a simple example, the following program transfers a data set from the Sashelp libref into an R data frame named df. The program then submits an R statement that displays the names of the variables in the data frame.

```
proc iml;
call ExportDataSetToR("Sashelp.Class", "df" );
submit / R;
   names(df)
endsubmit;
```

The R **names** function produces the output shown in Figure 11.2.

Figure 11.2 Result of Sending Data to R

```
[1] "Name"   "Sex"    "Age"    "Height" "Weight"
```

Transfer from an R Source to a SAS Destination

You can transfer data and results from R data frames or matrices to a SAS data set or a SAS/IML matrix. Table 11.2 summarizes the frequently used methods that copy from an R source to a SAS destination.

Table 11.2 Transferring from an R Source to a SAS Destination

Subroutine	R Source	SAS Destination
ImportDataSetFromR	R expression	SAS data set
ImportMatrixFromR	R expression	SAS/IML matrix

The next section includes an example of calling an R analysis. Some of the results from the analysis are then transferred into SAS/IML matrices.

The result of an R analysis can be a complicated structure. In order to transfer an R object via the previously mentioned methods and modules, the object must be coercible to a data frame. (The R object **m** can be coerced to a data frame provided that the function **as.data.frame(m)** succeeds.) There are many data structures that cannot be coerced into data frames. As the example in the next section shows, you can use R statements to extract and transfer simpler objects.

Call an R Analysis from PROC IML

You can use the techniques in Chapter 10, "Submitting SAS Statements," to perform a linear regression by calling a regression procedure (such as REG, GLM, or MIXED) in SAS/STAT software. This section presents examples of submitting statements to R to perform a linear regression. The first example performs a linear regression on data that are transferred from SAS/IML vectors. The second example performs an identical analysis on data that are transferred from a SAS data set.

Using R to Analyze Data in SAS/IML Matrices

The program in this section consists of four parts:

1. Read the data into SAS/IML vectors.
2. Transfer the data to R.
3. Call R functions to analyze the data.
4. Transfer the results of the analysis into SAS/IML vectors.

1 Read the data. The following statements read the Weight and Height variables from the Sashelp.Class data set into SAS/IML vectors with the same names:

```
proc iml;
use Sashelp.Class;
read all var {Weight Height};
close Sashelp.Class;
```

2 Transfer the data to R. The following statements run the ExportMatrixToR subroutine in order to transfer data from a SAS/IML matrix into an R matrix. The names of the corresponding R vectors that contain the data are **w** and **h**.

```
/* send matrices to R */
call ExportMatrixToR(Weight, "w");
call ExportMatrixToR(Height, "h");
```

3 Call R functions to perform some analysis. The SUBMIT statement with the R option is used to send statements to R. Comments in R begin with a hash mark (#, also called a number sign or a pound sign).

```
submit / R;
   Model    <- lm(w ~ h, na.action="na.exclude")         # a
   ParamEst <- coef(Model)                               # b
   Pred     <- fitted(Model)
   Resid    <- residuals(Model)
endsubmit;
```

The R program consists of the following steps:

a. The **lm** function computes a linear model of w as a function of h. The **na.action=** option specifies how the model handles missing values (which in R are represented by NA). In particular, the **na.exclude** option specifies that the **lm** function should not omit observations with missing values from residual and predicted values. This option makes it easier to merge the R results with the original data when the data contain missing values.

b. Various information is retrieved from the linear model and placed into R vectors named **ParamEst**, **Pred**, and **Resid**.

4 Transfer the data from R. The ImportMatrixFromR subroutine transfers the **ParamEst** vector from R into a SAS/IML vector named **pe**. This vector is printed by the SAS/IML PRINT statement. The predicted values (**Pred**) and residual values (**Resid**) can be transferred similarly. The parameter estimates are used to compute the predicted values for a series of hypothetical heights, as shown in Figure 11.3.

```
call ImportMatrixFromR(pe, "ParamEst");
print pe[r={"Intercept" "Height"}];

ht = T( do(55, 70, 5) );
A = j(nrow(ht),1,1) || ht;
pred_wt = A * pe;
print ht pred_wt;
```

Figure 11.3 Results from an R Analysis

```
                    pe

          Intercept -143.0269
          Height    3.8990303

                 ht   pred_wt

                 55 71.419746
                 60 90.914898
                 65 110.41005
                 70  129.9052
```

You cannot directly transfer the contents of the **Model** object. Instead, various R functions are used to extract portions of the **Model** object, and those simpler pieces are transferred.

Using R to Analyze Data in a SAS Data Set

As an alternative to the data transfer statements in the previous section, you can call the ExportDataSetToR subroutine to transfer the entire SAS data set to an R data frame. For example, you could use the following statements to create an R data frame named Class and to model the Weight variable:

```
call ExportDataSetToR("Sashelp.Class", "Class");
submit / R;
   Model <- lm(Weight ~ Height, data=Class, na.action="na.exclude")
endsubmit;
```

The R language is case-sensitive so you must use the correct case to refer to variables in a data frame. You can use the CONTENTS function in the SAS/IML language to obtain the names and capitalization of variables in a SAS data set.

Passing Parameters to R

The SUBMIT statement supports parameter substitution from SAS/IML matrices as detailed in "Passing Parameters from SAS/IML Matrices" on page 182. For example, you can substitute the names of analysis variables into a SUBMIT block by using the following statements:

```
YVar = "Weight";
XVar = "Height";
submit XVar YVar / R;
   Model <- lm(&YVar ~ &XVar, data=Class, na.action="na.exclude")
   print (Model$call)
endsubmit;
```

Figure 11.4 shows the result of the `print(Model$call)` statement. The output shows that the values of the `YVar` and `XVar` matrices were substituted into the SUBMIT block.

Figure 11.4 Parameter Substitutions in a SUBMIT Block

```
lm(formula = Weight ~ Height, data = Class, na.action = "na.exclude")
```

Call R Packages from PROC IML

You do not need to do anything special to call an R package. Provided that an R package is installed, you can call `library(`*package*`)` from inside a SUBMIT block to load the package. You can then call the functions in the package.

The example in this section calls an R package and imports the results into a SAS data set. This example is similar to the example in "Creating Graphics in a SUBMIT Block" on page 185, which calls the UNIVARIATE procedure to create a kernel density estimate. The program in this section consists of the following steps:

1. Define the data and transfer the data to R.

2. Call R functions to analyze the data.

3. Transfer the results of the analysis into SAS/IML vectors.

1 Define the data in the SAS/IML vector **q** and then transfer the data to R by using the ExportMatrixToR subroutine. In R, the data are stored in a vector named **rq**.

```
proc iml;
q = {3.7, 7.1, 2, 4.2, 5.3, 6.4, 8, 5.7, 3.1, 6.1, 4.4, 5.4, 9.5, 11.2};
RVar = "rq";
call ExportMatrixToR( q, RVar );
```

2 Load the KernSmooth package. Because the functions in the KernSmooth package do not handle missing values, the nonmissing values in **q** must be copied to a matrix **p**. (There are no missing values in this example.) The Sheather-Jones plug-in bandwidth is computed by calling the **dpik** function in the KernSmooth package. This bandwidth is used in the **bkde** function (in the same package) to compute a kernel density estimate.

```
submit RVar / R;
   library(KernSmooth)
   idx <-which(!is.na(&RVar))        # must exclude missing values (NA)
   p <- &RVar[idx]                   #    from KernSmooth functions
   h = dpik(p)                       # Sheather-Jones plug-in bandwidth
   est <- bkde(p, bandwidth=h)       # est has 2 columns
endsubmit;
```

3 Copy the results into a SAS data set or a SAS/IML matrix, and perform additional computations. For example, the following statements use the trapezoidal rule to numerically estimate the density that is contained in the tail of the density estimate of the data:

```
call ImportMatrixFromR( m, "est" );
/* estimate the density for q >= 8 */
x = m[,1];                /* x values for density */
idx = loc( x>=8 );        /* find values x >= 8 */
y = m[idx, 2];            /* extract corresponding density values */

/* Use the trapezoidal rule to estimate the area under the density curve.
   The area of a trapezoid with base w and heights h1 and h2 is
   w*(h1+h2)/2. */
w = m[2,1] - m[1,1];
h1 = y[1:nrow(y)-1];
h2 = y[2:nrow(y)];
Area = w * sum(h1+h2) / 2;
```

```
   print Area;
```

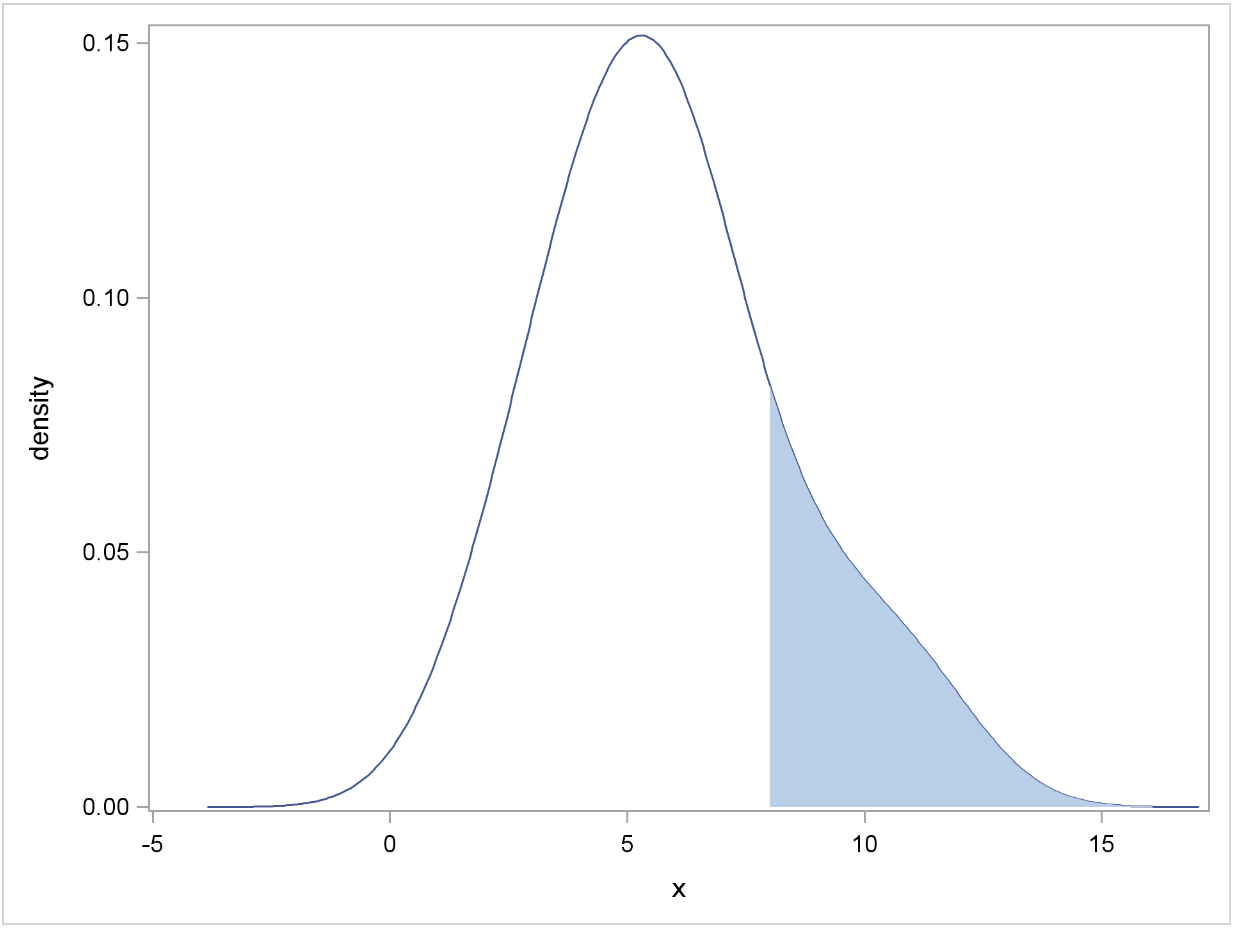
The numerical estimate for the conditional density is shown in Figure 11.5. The estimate is shown graphically in Figure 11.6, where the conditional density corresponds to the shaded area in the figure. Figure 11.6 was created by using the SGPLOT procedure to display the density estimate computed by the R package.

Figure 11.5 Computation That Combines SAS/IML and R Computations

```
                                  Area

                                0.2118117
```

Figure 11.6 Estimated Density for $x \geq 8$



Call R Graphics from PROC IML

R can create graphics in a separate window which, by default, appears on the same computer on which R is running. If you are running PROC IML and R locally on your desktop or laptop computer, you can display R graphics. However, if you are running client software that connects with a remote SAS server that is running PROC IML and R, then R graphics might be disabled.

The following statements describe some common scenarios for running a PROC IML program:

- If you run PROC IML through a SAS Display Manager Session (DMS), you can create R graphics from your PROC IML program. The graph appears in the standard R graphics window.
- If you run PROC IML through SAS Enterprise Guide, the display of R graphics is disabled because, in general, the SAS server (and therefore R) is running on a different computer than the SAS Enterprise Guide application.
- If you run PROC IML from interactive line mode or from batch mode, then R graphics are disabled.

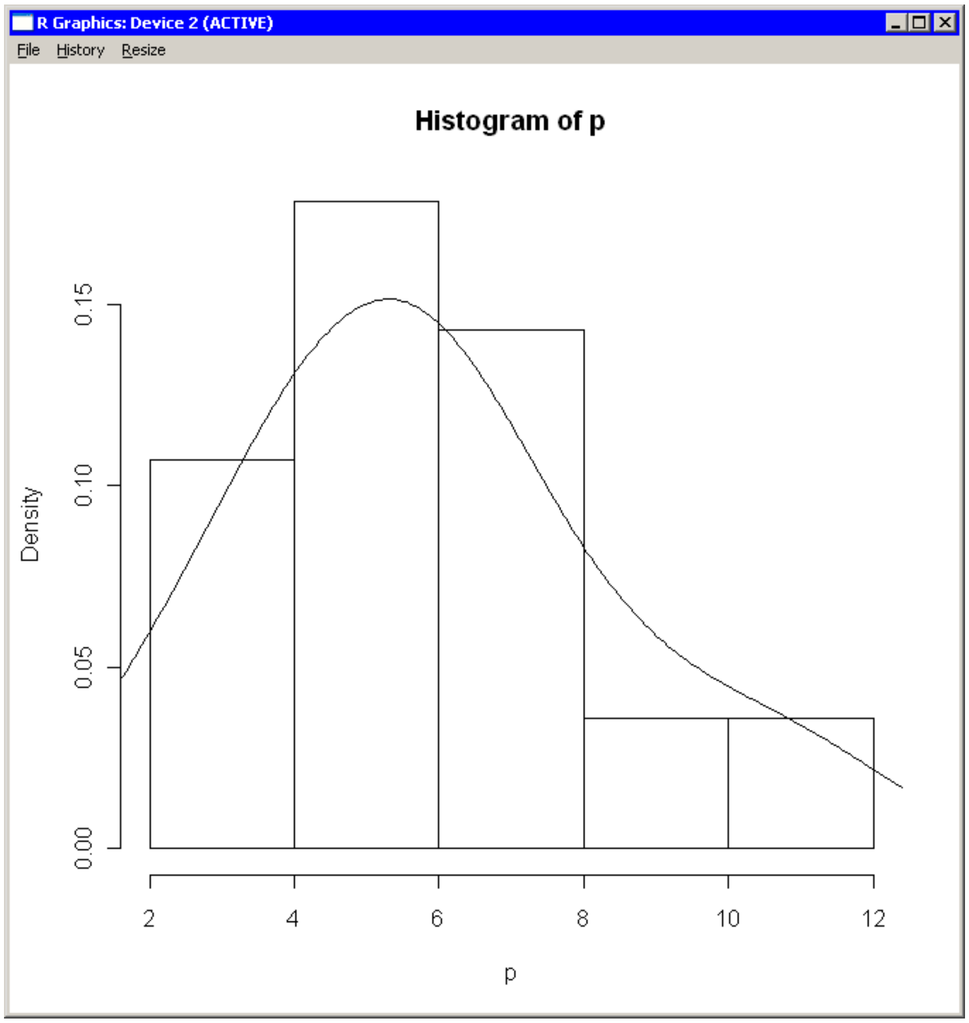
You can determine whether R graphics are enabled by calling the **`interactive`** function in the R language.

For example, the previous section used R to compute a kernel density estimate for some data. If you are running PROC IML through SAS DMS, you can create a histogram and overlay the kernel density estimate by using the following statements:

```
submit / R;
   hist(p, freq=FALSE) # histogram
   lines(est)          # kde overlay
endsubmit;
```

The **`hist`** function creates a histogram of the data in the **`p`** matrix, and the **`lines`** function adds the kernel density estimate contained in the **`est`** matrix. The R graphics window contains the histogram, which is shown in Figure 11.7.

Figure 11.7 R Graphics



Handling Errors from R

If you submit R code that causes an error, you can attempt to handle the error by using the OK= option in the SUBMIT statement, as described in "Handling Errors in a SUBMIT Block" on page 187.

Details of Data Transfer

This section describes how data are transferred between SAS and R software. It includes a discussion of numerical data types, missing values, and data that represent dates and times.

Numeric Data Types

R can store numeric data in either an integer or a double-precision data type. When transferring R data to a SAS data type, integers types are converted to double precision.

Logical Data Types

R provides a logical data type for storing the values **TRUE** and **FALSE**. When logical data are transferred to a SAS data type, the value **TRUE** is converted to the number 1 and the value **FALSE** to the number 0.

Unsupported Data Types

R provides two data types that are not converted to a SAS data type: complex and raw. It is an error to attempt to transfer data stored in either of these data types to a SAS data type.

Special Numeric Values

The R language has four symbols that are used to represent special numerical values.

- The symbol **`NA`** represents a missing value.
- The symbol **`Inf`** represents positive infinity.
- The symbol **`-Inf`** represents positive infinity.
- The symbol **`NaN`** represents a "NaN," which is a floating-point value that represents an undefined value such as the result of the division $0/0$.

The SAS language has 28 symbols that are used to represent special numerical values.

- The symbol `.` represents a generic missing value.
- The symbols **`.A`**–**`.Z`** and `._` are also missing values. Some applications use **`.I`** to represent positive infinity and use **`.M`** to represent negative infinity.

The following table shows how special numeric values in R are converted to SAS missing values:

Value in R	SAS Missing Value
Inf	.I
–Inf	.M
NA	.
NaN	.

The following table shows how SAS missing values are converted when data are transferred to R:

SAS Missing Value	Value in R
.I	Inf
.M	–Inf
All others	NA

Date, Time, and Datetime Values

R supports date and time data differently than does SAS software. In SAS software, variables that represent dates or times are assigned a format such as DATE9. or TIME5. In R, classes are used to represent dates and times.

When a variable in a SAS data set is transferred to R software, the variable's format is examined and the following occurs:

- If the format is in the family of date formats (for example, DATE*w.d*), the variable in R is assigned the "Date" class.
- If the format is in the family of datetime formats (for example, DATETIME*w.d*) or time formats (for example, TIME*w.d*), the variable in R is assigned the "POSIXct" and "POSIXt" classes.
- In all other cases, the variable in R is assigned the "numeric" class.

When a variable in an R data frame is transferred to SAS software, the variable's class is examined and the following occurs:

- If the variable's class is "Date," the corresponding SAS variable is assigned the DATE9. format.
- If the variable's class is "POSIXt," the corresponding SAS variable is assigned the DATETIME19. format.
- In all other cases, the SAS variable is not assigned a format.

Time Series Data

In SAS, the sampling times for time series data are often stored in a separate variable. In R, the sampling times for a time series object are specified by the `tsp` attribute. When a time series object in R is transferred to SAS software, the following occurs:

- The R **`time`** function is used to generate a vector of the times at which the time series is sampled.
- A new variable named *VarName*_ts is created, where *VarName* is the name of the time series object in R. The variable contains sampling times for the time series.

No special processing of time series data is performed when data are transferred from SAS to R software.

Data Structures

R provides a wide range of built-in and user-defined data structures. When data are transferred from R to SAS software, the data are coerced to a data frame prior to the transfer. If the coersion fails, the data are not transferred.

The section "Using R to Analyze Data in SAS/IML Matrices" on page 194 presents an example of an R object that cannot be directly imported to SAS software and shows how to use R functions to extract simpler data structures from the R object.

Differences from SAS/IML Studio

This section lists differences between the R option in the SUBMIT statement as implemented in SAS/IML Studio and the same option in PROC IML:

- In PROC IML, R must be installed on the computer that runs the SAS server. In SAS/IML Studio, R must be installed on the computer that runs the SAS/IML Studio application.
- If R is installed on a SAS workspace server and is accessed through SAS Enterprise Guide, everyone that connects to that server uses the same version of R and the same set of installed packages. In SAS/IML Studio, R is installed locally on the client computer, so each user can potentially have a different version of R and different packages.