Chapter 11
Calling Functions in the R Language

Contents

Introduction to Calling R Functions ................................................. 73
Submit R Statements ................................................................. 74
Transfer between SAS and R Data Structures ................................ 75
  Transfer from a SAS Source to an R Destination ..................... 75
  Transfer from an R Source to a SAS Destination ..................... 76
Call an R Analysis from IMLPlus ................................................. 76
Call R Packages and Graphics from IMLPlus ............................... 79

Introduction to Calling R Functions

R is a freely available language and environment for statistical computing and graphics. Like IMLPlus, 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 for creating statistical graphs, and the capability to extend the basic function library by writing user-defined functions. There are also a large number of R packages that implement specialized computations.

SAS/IML Studio has an interface to the R language that enables you to submit R statements from within your IMLPlus program. In previous chapters you learned how to transfer data between SAS/IML Studio and a SAS server, how to call SAS procedures, and how to read the results back into SAS/IML Studio. This chapter describes how to transfer data to R, how to call R functions, and how to transfer the results to a number of SAS data structures.

The program statements in this chapter are distributed with SAS/IML Studio. To open the program that contains the statements:

1 Select File ▶ Open ▶ File from the main menu.

2 Click Go to Installation directory near the bottom of the dialog box.

3 Navigate to the Programs\Doc\STATGuide folder.

4 Select the R.sx file.

5 Click Open.
In order to run the examples in this chapter, you must first install R on the same PC that runs SAS/IML Studio. For details on how to install R and which versions of R are supported, see the chapter “Accessing R” in the SAS/IML Studio online Help.

---

**Submit R Statements**

Submitting R statements is similar to submitting SAS statements. You use a SUBMIT statement, but add the R option: SUBMIT / R. All statements in the program prior to the next ENDSUBMIT statement are sent to R for execution.

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.

```sas
/* 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};   /* 3x3 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) # 3x3 matrix
  rq <- rm %*% t(rx) # matrix multiplication
  print(rq)
endsubmit;
```

---

**Figure 11.1** Output from SAS/IML and R

<table>
<thead>
<tr>
<th>SAS/IML Results</th>
<th>R Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>q</td>
<td><img src="https://example.com/sasiml_vs_r.png" alt="Output" /></td>
</tr>
</tbody>
</table>

---
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.

Transfer between SAS and R Data Structures

Many research statisticians take advantage of special-purpose functions and packages written in the R language. To 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
- an IMLPlus DataObject

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

The following table summarizes the frequently used methods that copy from a SAS source to an R destination. Several of these modules and methods are used in the program in the next section. For details of the transfer process and a full list of methods that transfer data, see the “Accessing R” chapter in the online Help.

<table>
<thead>
<tr>
<th>Method or Module</th>
<th>SAS Source</th>
<th>R Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExportDataSetToR</td>
<td>SAS data set</td>
<td>R data frame</td>
</tr>
<tr>
<td>ExportMatrixToR</td>
<td>SAS/IML matrix</td>
<td>R matrix</td>
</tr>
<tr>
<td>DataObject.ExportToR</td>
<td>DataObject</td>
<td>R data frame</td>
</tr>
</tbody>
</table>

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.
run ExportDataSetToR("Sashelp.Class", "df");
submit / R;
  names(df);
endsubmit;

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

Figure 11.2  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, a DataObject, or a SAS/IML matrix. The following table summarizes the frequently used methods that copy from an R source to a SAS destination.

<table>
<thead>
<tr>
<th>Method or Module</th>
<th>R Source</th>
<th>SAS Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataObject.AddVarFromR</td>
<td>R expression</td>
<td>DataObject variable</td>
</tr>
<tr>
<td>DataObject.CreateFromR</td>
<td>R expression</td>
<td>DataObject</td>
</tr>
<tr>
<td>ImportDataSetFromR</td>
<td>R expression</td>
<td>SAS data set</td>
</tr>
<tr>
<td>ImportMatrixFromR</td>
<td>R expression</td>
<td>SAS/IML matrix</td>
</tr>
</tbody>
</table>

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 and into variables in a DataObject.

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 can not be coerced into data frames. As the example in the next section shows, you can use R statements to extract simpler objects and transfer the simpler objects.

### Call an R Analysis from IMLPlus

The example in Chapter 4, “Calling SAS Procedures,” submits SAS statements to call the REG procedure. The example performs a linear regression of the `wind_kts` variable by the `min_pressure`
variable of the Hurricanes data. The following program repeats the same analysis, but does it by submitting statements to R:

```
declare DataObject dobj;
dobj = DataObject.CreateFromFile("Hurricanes");
dobj.GetVarData( "wind_kts", w ); /* Step 1 */
dobj.GetVarData( "min_pressure", p );

/* send matrices to R */
run ExportMatrixToR( w, "Wind" ); /* Step 2 */
run ExportMatrixToR( p, "Pressure" );

print "-------------- In R ---------------"; /* Step 3 */
submit / R;
    Model <- lm(Wind~Pressure, na.action="na.exclude") # 3a
    ParamEst <- coef(Model) # 3b
    Pred <- fitted(Model)
    Resid <- residuals(Model)
    print (ParamEst) # 3c
endsubmit;

print "----------- In SAS/IML ------------"; /* Step 4 */
run ImportMatrixFromR( pe, "ParamEst" );
print pe[r={"Intercept" "min_pressure"}];

/* add variables to the DataObject */
dobj.AddVarFromR( "R_Pred", "Pred" ); /* Step 5 */
dobj.AddVarFromR( "R_Resid", "Resid" );
ScatterPlot.Create(dobj, "min_pressure", "R_Resid");
```

The output from this program is shown in Figure 11.3. The program consists of the following steps:

1. The GetVarData method of the DataObject class copies the data for the wind_kts and min_pressure variables into SAS/IML vectors named w and p.

2. These vectors are sent to R by the ExportMatrixToR module. The names of the corresponding R vectors that contain the data are Wind and Pressure.

3. The SUBMIT statement with the R option is used to send statements to R. Note that comments in R begin with a hash mark (#, also called a number sign or a pound sign).
   a) The `lm` function computes a linear model of Wind as a function of Pressure. 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.
   b) Various information is retrieved from the linear model and placed into R vectors named ParamEst, Pred, and Resid.
   c) The parameter estimates are printed in R, as shown in Figure 11.3.

4. The ImportMatrixFromR module transfers the ParamEst vector from R into a SAS/IML vector named pe. This vector is printed by the SAS/IML PRINT statement.
5. The `Pred` and `Resid` vectors are added to the DataObject. The new variables are given the names `R_Pred` and `R_Resid`. A scatter plot of the residual values versus the explanatory variable is created, similar to Figure 6.1.

**Figure 11.3** Calling an R Analysis

<table>
<thead>
<tr>
<th>In R</th>
<th>In SAS/IML</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>1333.3549</td>
</tr>
<tr>
<td>Pressure</td>
<td>-1.291374</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>pe</td>
<td></td>
</tr>
<tr>
<td>Intercept</td>
<td>1333.3549</td>
</tr>
<tr>
<td>min_pressure</td>
<td>-1.291374</td>
</tr>
</tbody>
</table>

Note that you cannot directly transfer the contents of the `Model` object. Instead, various R functions were used to extract portions of the `Model` object, and those pieces were transferred.

As an alternative to steps 1 and 2, you can call the ExportToR method in the DataObject class. The ExportToR method writes an entire DataObject to an R data frame. For example, after creating the DataObject you could use the following statements to create an R data frame named `Hurr`:

```r
dobj.ExportToR("Hurr");
submit / R;
   Model <- lm(wind_kts~min_pressure, data=Hurr, 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.

The SUBMIT statement for R supports parameter substitution from SAS/IML matrices, just as it does for SAS statements. For example, you can substitute the names of analysis variables into a SUBMIT block by using the following statements:

```r
YVar = "wind_kts";
XVar = "min_pressure";
submit XVar YVar / R;
   Model <- lm(&YVar ~ &XVar, data=Hurr, 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 = wind_kts ~ min_pressure, data = Hurr, na.action = "na.exclude")
```
Call R Packages and Graphics from IMLPlus

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.

Similarly, you do not need to do anything special to call R graphics. The graph appears in the standard R graphics window.

The example in this section calls an R package and creates a graph in R.

In Chapter 6, “Adding Curves to Graphs,” you called the KDE procedure to compute a kernel density estimate for the `min_pressure` variable in the Hurricanes data set. The following program reproduces that analysis by calling functions in the KernSmooth package and creating a histogram in R:

```r
declare DataObject dobj;
dobj = DataObject.CreateFromFile("Hurricanes");
dobj.GetVarData("min_pressure", p);
run ExportMatrixToR( p, "Pressure" );

submit / R;
library(KernSmooth)
idx <- which(!is.na(Pressure)) # must exclude missing values (NA)
p <- Pressure[idx] # from KernSmooth functions
h = dpik(p) # Sheather-Jones plug-in bandwidth
est <- bkde(p, bandwidth=h) # est has 2 columns
hist(p, breaks="Scott", freq=FALSE, col="lightyellow") # histogram
lines(est) # kde overlay
endsubmit;
```

The program creates an R matrix `Pressure` from the data in the `min_pressure` variable. Because the functions in the KernSmooth package do not handle missing values, the nonmissing values in `Pressure` must be copied to a matrix `p`. 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.

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 11.5. You can compare the histogram and density estimate created by R with the IMLPlus graph shown in Figure 6.4.
Figure 11.5  R Graphics

Histogram of p