R Procedure

The R procedure enables WPS programs written in the SAS language to include code written in the R language.

Combining the R language and the SAS language enables the bulk of a data processing and analytics solution to be written in the industrial strength and high-performing SAS language, while also exploiting features present in the R language.

The following image shows how a SAS language program using the R procedure is processed.

Introduction .................................................. 4
The R procedure enables SAS language programs to include code written in the R language.

Setup and configuration ........................................ 4
Setting the R environment for WPS.

Using R with WPS .................................................. 6
Using R in a SAS language program enables you to use features that might not be available in WPS.

Example ................................................................ 10
Demonstrates how to use a SAS language dataset in the R procedure to create a scatter plot diagram.

R procedure reference ............................................. 11
Describes the syntax and options for PROC R and its contained statements.
Introduction

The R procedure enables SAS language programs to include code written in the R language.

By combining R and the SAS language you can:

• Use the SAS language to bulk process and prepare data, and pass the processed data to R.
• Use R packages you have previously developed for data analysis.
• Use R data analysis packages or solutions that may not be available in the SAS language.

We recommend that programs are mainly coded in the language of SAS, using R where specialist statistics are required.

Data is passed between the SAS language and R language environments using the **EXPORT** statement. Once data has been transferred, that data is made available as a **data.frame** object to an R program. On completion of the R program data can, if required, be returned to the SAS language environment using the **IMPORT** statement.

Setup and configuration

Setting the R environment for WPS.

When using R with WPS:

• The 32-bit R interpreter is required for the 32-bit version of WPS, and the 64-bit R interpreter is required for the 64-bit version of WPS.
• You do not have to install extra modules, and there are no special licensing requirements to use the R procedure.

The R procedure can be used with R version 2.15.x and later, and is currently supported on Microsoft Windows, Linux-based systems, and macOS.

WPS software is not shipped with a copy of R. To use the R procedure, you need a separate installation of R.

Following R installation, set the **R_HOME** environment variable to point to the folder containing either **libr.dll** on Windows platforms or **libr.so** on UNIX or Linux platforms.
Installing the R interpreter

Windows platforms

The Windows installer package can be downloaded from http://www.r-project.org/ and includes both 32-bit and 64-bit versions of R. The same installer can be used for both 32-bit and 64-bit versions of the WPS software.

By default, the R installation saves the installation location in the Windows registry which is where WPS looks to identify the currently installed version. This is the version of R most recently installed, and no special configuration of WPS is required for WPS to locate it.

UNIX or Linux platforms

WPS requires the shared R library libr.so to interact with the R interpreter. This library is not included with the R binary distribution for UNIX platforms by default. On UNIX or Linux platforms, you need to either build R from source to include the required shared library, or install R using your systems package management system.

To build from source code:

1. You require a minimal set of pre-installed libraries before you can build R from source code. These are equivalent to the build-essentials package plus a JDK on Ubuntu.
2. Download the R source code from http://www.r-project.org/ and follow the instructions included with the download. Ensure you use the -re-establishment option when running configure to build the libr.so shared library, for example:

   ./configure --enable-R-shlib --prefix=$HOME/R

For more information, see the R documentation at http://www.r-project.org/.

macOS platforms

The binary distribution of R can be installed directly from the R project website.

To use R with WPS, the R_HOME variable must be set to point to the installation directory containing the libr.so shared library. The default setting is: /Library/Frameworks/Framework/Resources. Because of the way applications are launched on macOS, it is not possible to set R_HOME in a shell profile script.

To use a specific version of R, you can modify the setting appropriately, for example: /Library/Frameworks/Framework>Versions/3.0/Resources
Setting the \textit{R\_HOME} environment variable

To locate the installed version of R, the \textit{R\_HOME} environment variable must be set.

On Windows platforms, the \textit{R\_HOME} environment variable must point to the folder containing the \textit{libr.dll} file. On UNIX or Linux platforms, the \textit{R\_HOME} environment variable must point to the folder containing the \textit{libr.so} file.

If you are running WPS with R on a Unix or Linux platform, set the \textit{R\_HOME} variable to point to the folder containing \textit{libr.so}.

If you have multiple installations of R, set \textit{R\_HOME} variable as part of the SAS language program you run in WPS.

\textbf{Set \textit{R\_HOME} before running WPS}

The \textit{R\_HOME} variable can be defined as a system variable, and used by all applications on your device that run R. It is not necessary to set the variable on the Windows platform if the default R installation location is used.

\textbf{Set \textit{R\_HOME} in a SAS language program}

The \textit{R\_HOME} variable can be set in a SAS language program using the \texttt{SET} system option, for example:

\begin{verbatim}
OPTIONS SET = R\_HOME 'C:\Program Files\R\R-3.5.0';
\end{verbatim}

This will set the \textit{R\_HOME} environment variable for duration of the execution of the SAS language program.

\textbf{Using R with WPS}

Using R in a SAS language program enables you to use features that might not be available in WPS.

Installed R packages can be imported and used within the in-line R code using the library statement, for example:

\begin{verbatim}
PROC R;
  SUBMIT;
    library(datasets)
data(iris)
  summary(iris)
  plot(iris)
ENDSUBMIT;
RUN;
\end{verbatim}

Each subsequent use of the R procedure in a SAS language program can use the same R environment. This means any global variables or imported packages used in an R procedure invocation are available to all subsequent R procedure invocations.
Each R procedure invocation can include multiple blocks of in-line R language code, and use a combination of in-line R language code, and R programs run using the `EXECUTE` statement.

**Data type conversion**

Describes the correspondence between SAS language formats and R data types.

When you write data to a `data.frame` using the R procedure `EXPORT` statement, formatted data is converted to an equivalent R data type. WPS has many formats that affect the output and display of data. Formats that only affect the layout of data output, such as adding currency symbols or comma separators, have no effect when writing data.

A `data.frame` is imported using the R procedure `IMPORT` statement. Any object that can be coerced into a `data.frame` using the `as.data.frame` R function is imported into the WPS dataset.

**Logical values**

Logical values are converted into numeric values in the WPS dataset. The values of vectors of type logical are converted as follows:

<table>
<thead>
<tr>
<th>R Value</th>
<th>WPS Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>1</td>
</tr>
<tr>
<td>FALSE</td>
<td>0</td>
</tr>
</tbody>
</table>

**Integer values**

The R language value `NA`, which is represented as the minimum integer value (-2147483648) is converted to a SAS language missing value.

**Real values**

There are three special real numeric values in the R language:

- `NA` (Not Available). Represents an absent value.
- `NaN` (Not a Number). Represents an undefined value, or a value that cannot be displayed, for example the result of zero divided by zero.
- `Inf` (Infinity). Represent positive and negative infinite values. For example the result of trying to divide any value by zero.

These values are converted from the R language representation to the SAS language representation as follows:

<table>
<thead>
<tr>
<th>R Value</th>
<th>WPS Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>. (missing value)</td>
</tr>
<tr>
<td>NaN</td>
<td>. (missing value)</td>
</tr>
<tr>
<td>+Inf</td>
<td>.I</td>
</tr>
<tr>
<td>-Inf</td>
<td>.M</td>
</tr>
</tbody>
</table>
Date values

Integer or real variables in the R language that have a `Date` class are formatted as `DATE9` when imported into WPS.

R language `Date` values represent a count of days from the Unix epoch of 1 January 1970 UTC. Imported variables are adjusted to take account of the SAS language epoch of 1 January 1960.

Datetime values

Real variables in the R language that have a `POSIXct` class are formatted as `DATETIME19` when imported into WPS.

R language `POSIXct` values represent a count of seconds from the Unix epoch of 1 January 1970 UTC. Imported variables are adjusted to take account of the SAS language epoch of 1 January 1960 and also use the value specified in the `GMTOFFSET` option of the `PROC R` statement.

Real variables in the R language that have a `times` class are formatted as `TIME8` when imported into WPS.

Character values

On import, WPS scans a character variable and assigns a format that is the length of the longest string. Individual values in a character variable that are `NA` (Not Available) are converted to the SAS language missing character value (".")

Factor values

An R language `factor` is a form of integer variable, where the values index a list of categorical variables (the list is known as a level in the R language). When imported into WPS these are converted into character variables in the dataset. The variable is given a length equal to the longest string in the levels list.

Using R graphics

When launching an R session, WPS configures R so that any graphics generated with the default graphics device are captured and included in the WPS session's standard ODS HTML output.

The following program extends the previous example to include simple linear regression analysis and graphics.
1. Create a new program file, paste the following code, and save the file:

```r
data source;
do x=1 to 10;
   y=ranuni(-1);
output;
end;
PROC R;
   export data=source;
submit;
   model <- lm(source$y ~ source$x)
   print(model)
   par(mfrow=c(2, 2))
   plot(model)
endsubmit;
run;
```

2. Run the program by clicking on the toolbar Run icon, and examine the HTML output.

The output includes the printed R results together with a single graphic generated within the R session and routed into the WPS output.

```
Call:  
   lm(formula = source$y ~ source$x)  
Coefficients:  
(Intercept)     source$x  
   0.5344       0.0241  
```

### Using additional R packages

To use additional packages that are not included in your R installation, we recommend you install and check the basic operation of these packages in the interactive R environment. Installed packages can be used in an R program using the `library()` function.

An R environment launched by WPS inherits the environment variables from the WPS process. If third-party software is installed for use in R that requires, for example, additional entries in the `PATH` environment variable, Workbench must be restarted to register the changes.

### SAS language macro processing

Using SAS language macros with in-line R programs.

When an in-line R program is run, the code between the `SUBMIT` and `ENDSUBMIT` statements is passed verbatim to the R interpreter. Macro processing is, therefore, suspended between the `SUBMIT` and `ENDSUBMIT` statements because:
• The R language uses the & and % characters as part of its syntax. Attempting to macro process the R source code might result in valid R syntax being misinterpreted as SAS language macro statements.

• The R language allows line-end style comments. The contents might contain, for example, unmatched apostrophes; tokenising of the R syntax using the regular SAS language parsing rules could not then occur.

Example

Demonstrates how to use a SAS language dataset in the R procedure to create a scatter plot diagram.

The following example creates a dataset in a SAS language DATA step, and then uses the EXPORT statement to pass that dataset to the R environment. The dataset is converted to a data.frame as part of the export, and the data.frame is used to create plots in a grid two plots wide and two plots deep.

An output PDF file destination is created using the SAS language Output Delivery System (ODS). Adding PDF to the output destinations includes the printed data.frame content and returned plot image file in the PDF output. The PDF is saved and the output can be viewed in a PDF viewer.

```sas
ODS PDF FILE='scatter_plot.pdf';
DATA SOURCE;
  DO X=1 TO 10;
    Y=RANUNI(-1);
    OUTPUT;
  END;
RUN;

PROC R;
  EXPORT DATA=source;
  SUBMIT;
    str(source)
    print(source)
    model <- lm(source$Y ~ source$X)
    print(model)
    par(mfrow=c(2, 2))
    plot(model)
    x <- (1:10)
  ENDSUBMIT;
  IMPORT R=x;
RUN;
ODS PDF CLOSE;
```

This creates the following plot in the ODS PDF output:
R procedure reference

Describes the syntax and options for PROC R and its contained statements.

PROC R ................................................................. 12
Invokes the R environment that enables the execution of in-line or external R language programs.

ASSIGN .............................................................. 14
The ASSIGN statement can be used to assign SAS language variable values to an R vector.

ENDSUBMIT ......................................................... 15
Specifies the end of an in-line R language program.

EXECUTE ............................................................ 16
Runs an R program stored in a separate file.

EXPORT ............................................................ 17
Enables a SAS language dataset to be converted to an R data.frame and referenced in an R program.
IMPORT Enables an R language data.frame to be converted to a SAS language dataset and referenced in a SAS language program.

LOAD The LOAD statement deserialises an R object stored in a SAS language catalog.

SAVE Enables R objects to be serialised and stored in a SAS language catalog.

SUBMIT Specifies the start of an in-line R language program.

**PROC R**

Invokes the R environment that enables the execution of in-line or external R language programs.

```
PROC R [{ R_options ... } ] ;
```

Datasets created in WPS can be made available to the R program using the EXPORT statement, and a dataset imported from the R program into WPS using the IMPORT statement.

An R program can be either written in-line in the R procedure, or run from a separate file:

- To run an in-line R program, use the SUBMIT and ENDSUBMIT statements.
- To run an R program stored in an external file use the EXECUTE statement.

The R environment is exited using a RUN statement.

**Options**

The following options are available.

**GMTOFFSET**

Specifies the offset to UTC applied when moving datasets between the SAS language and R language environments to take account the current time zone.

```
GMTOFFSET = "+/-HH:MM"
```

Date and time values in the R language are represented in UTC (Coordinated Universal Time) with an associated time zone. In the SAS language date and time values have no implied time zone. The specified GMTOFFSET is applied when using the ASSIGN, EXPORT, or IMPORT statements.

**KEEP**

Specifies that the current R environment is not terminated when the procedure exits.

KEEP
When specified, the current R environment is kept active when the current procedure exits, and the environment is used in the next invocation of the R procedure in the same program. If that invocation does not specify KEEP, the environment is terminated when the procedure exits.

The default behaviour is to terminate the R environment at the end of the procedure. Specifying KEEP keeps the current R environment, including any modules loaded during the execution of an R program, to be used in the next invocation of PROC R.

You can specify the RKEEP system option to use the same R environment for the duration of the execution of the SAS language program.

LIB

Specifies the default library location for the procedure step. The default location is the WORK library.

LIB = [default-library]

The LIB location is used:

- When exporting a dataset, and libname is not specified as part of the path for the DATA option of the EXPORT statement.
- When importing a dataset, and libname is not specified as part of the path for the DATA option of the IMPORT statement.
- When saving an R object to a SAS language catalog, and libname is not specified as part of the path for the CATALOG option of the SAVE statement.
- When loading an R object to a SAS language catalog, and libname is not specified as part of the path for the CATALOG option of the LOAD statement.

TERMINATE

Specifies that the R environment is terminated when the procedure exits.

[ TERMINATE ]

Specifying TERMINATE stops the current R environment even if the RKEEP system option has been specified.

TIMESASCHRON

Specifies whether time values are represented in R using the chron class.

TIMESASCHRON

By default, time values are stored in the R POSIXct type, representing a count of seconds from midnight. When TIMESASCHRON is specified, time values are stored in R as chron.times types.

To use the TIMESASCHRON option, you must include the chron package in your R environment using the R library() statement.
Example

The following example shows how to use PROC R to find the version of the R interpreter used with WPS. Version information is written to ODS output.

```
PROC R;
    SUBMIT;
        print(R.version)
    ENDSUBMIT;
RUN;
```

ASSIGN

The ASSIGN statement can be used to assign SAS language variable values to an R vector.

```
ASSIGN robject-name = [ variable-value |
    ( { variable-value } ... ) ];
```

The ASSIGN statement is used to pass parameters to an R program, and the specified robject-name can be used in an in-line program or a program run using the EXECUTE command. robject-name is case sensitive and, unlike SAS language variables, must be referred to in the R program using the same case as in the ASSIGN definition.

Variables can be generated using SAS language macro variable expansion or execution. This enables you to preprocess variables using the SAS language and pass the result to an R program.

robject-name

Specifies one or more variable values to pass to an R program. The robject-name specifies the variable name by which the variable-value is referenced in the R program.

The robject-name can be defined using a SAS language name literal (‘robject.name’N) to create an R object that would not be valid in the SAS language. For example, to assign a value Peter to an R object employee.firstname, the ASSIGN statement would be:

```
ASSIGN ‘employee.firstname’N = ‘Peter’;
```

Multiple variable-values can be assigned to a single R language object. In this case, all values must be of the same type and in a comma-separated list in parenthesis.
Passing multiple variables to R

In this example, multiple vector variables are passed from a SAS language program to an R program that converts the vectors into a data frame.

```r
PROC R;
  ASSIGN Nu = (1, 2, 3, 4, 5);
  ASSIGN Ch = ('Cyan', 'Magenta', 'Yellow', 'Black', 'Green');
SUBMIT;
  DFrame <- data.frame(Nu, Ch)
  print (DFrame)
ENDSUBMIT;
RUN;
```

This produces the following output:

<table>
<thead>
<tr>
<th>Nu</th>
<th>Ch</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cyan</td>
</tr>
<tr>
<td>2</td>
<td>Magenta</td>
</tr>
<tr>
<td>3</td>
<td>Yellow</td>
</tr>
<tr>
<td>4</td>
<td>Black</td>
</tr>
<tr>
<td>5</td>
<td>Green</td>
</tr>
</tbody>
</table>

Assigning a SAS language macro variable to an R object

In this example, the SAS language macro variable `PARM` is passed to an R program to determine the sample size in a randomly-generated sample.

```sas
%LET PARM=15;
PROC R;
  ASSIGN parm=&PARM;
SUBMIT;
  x<-sample(1:3, parm, replace=TRUE)
  print(x);
ENDSUBMIT;
RUN;
```

This produces the following output:

```
[1] 1 1 3 3 2 1 1 3 1 1 3 3 1 1
```

ENDSUBMIT

Specifies the end of an in-line R language program.

ENDSUBMIT ;

The `ENDSUBMIT` statement must be entered at the start of a new line after the R language program.
EXECUTE

Runs an R program stored in a separate file.

\[
\text{EXECUTE}\ *\ \text{filename}\ *\ [\ {\ \text{cmd-argument}\ ...\ }\ ]\ ;
\]

The EXECUTE statement is an alternative to using the SUBMIT statement. It allows the R code to be placed in a separate file and enables you to run the same program in both WPS and an interactive R environment.

**filename**

A quoted string containing the path of the R program file. filename can be either an absolute path or a relative path.

When using Workbench to run an R language program, if a relative path is specified the root of the path is the Workspace. For example, to run a file named math.r from a project named calculate, the relative path is calculate/math.r.

**cmd-argument**

Specifies a command line argument passed to the R program.

An example of executing an R program stored in a file

In this example, an R program stored in an external file model.r is executed in the R procedure. The contents of model.r source file:

```r
model <- lm(source$Y ~ source$X)
print(model)
par(mfrow=c(2, 2))
plot(model)
```

The following program creates a dataset. The dataset is passed to the R program using the EXPORT statement before the program is run using the model.r file:

```r
DATA SOURCE;
DO X=1 TO 10;
\ Y=RANUNI(-1);
\ OUTPUT;
END;

PROC R;
\ EXPORT DATA=source;
\ EXECUTE "model.r";
RUN;
```
EXPORT

Enables a SAS language dataset to be converted to an R data.frame and referenced in an R program.

EXPORT { EXPORT_option ... } ;

Export options

The following options are available with the EXPORT statement.

DATA
Specifies the name of the WPS dataset.

DATA = [ libname . ] dataset

The library location can be specified using either libname in the DATA option, or the LIB option of the PROC R statement:

• If libname is specified, that location is always used.
• If the LIB option of the PROC R statement is specified and libname is not specified, the location in the LIB option is used.
• If neither libname or the LIB option on the PROC R statement are specified, the WORK location is used.

R
Specifies the name of the data.frame as used in the R environment.

R = r-dataframe

If this option is not specified, the r-dataframe default is the dataset name specified in the DATA option. If you use the default dataset name in an in-line R language program, the variable name must match the case used in the DATA option.

An example of exporting data from WPS to R

The following example creates a dataset containing two numeric columns. The dataset is exported to the R environment and the content of the data.frame printed to ODS output.

DATA SOURCE;
  DO X=1 TO 10;
    Y=RANUNI(-1);
    OUTPUT;
  END;

PROC R;
  EXPORT DATA=SOURCE;
  SUBMIT;
    str(source)
  ENDSUBMIT;
RUN;
This produces the following output:

```
'data.frame': 10 obs. of  2 variables:
$ x: num  1 2 3 4 5 6 7 8 9 10
$ y: num  0.371 0.924 0.59 0.434 0.962 ...
```

## IMPORT

Enables an R language `data.frame` to be converted to a SAS language dataset and referenced in a SAS language program.

```
IMPORT { IMPORT_option ... };
```

### Import options

The following options are available with the `IMPORT` statement.

### DATA

Specifies the dataset location and name as used in the WPS Analytics SAS language environment.

```
DATA = [ libname . ] dataset
```

The library location can be specified using either `libname` in the `DATA` option, or the `LIB` option of the `PROC R` statement:

- If `libname` is specified, that location is always used.
- If the `LIB` option of the `PROC R` statement is specified and `libname` is not specified, the location in the `LIB` option is used.
- If neither `libname` nor the `LIB` option on the `PROC R` statement are specified, the `WORK` location is used.

If this option is not specified, the `dataset` default is the `r-dataframe` name specified in the R option.

### R

Specifies the name of the `data.frame` as used in the R environment. Must be specified

```
R = r-dataframe
```

`r-dataframe` is case sensitive and must match the case used for the imported `data.frame` in the R program.

## LOAD

The `LOAD` statement deserialises an R object stored in a SAS language catalog.

```
LOAD { LOAD_option ... };
```
The `SAVE` and `LOAD` statements enable an R object to be serialised and stored in a SAS language catalog and later deserialised in a WPS session. The `SAVE` statement serialises an R object and stores it in an entry in a catalog. The `LOAD` statement deserialises an R object from a catalog.

Load options

The following options are available with the `LOAD` statement.

### CATALOG

Specifies the catalog from which the stored R object is loaded.

\[
\text{CATALOG} = \left\{ \begin{array}{ll}
\text{CAT} & \text{libname}.
\end{array} \right\} \text{ catalogue } \text{ entry}
\]

A SAS language catalog is defined using:

- **libname**: Specifies the name of the library in which the catalog is stored. The library location can be specified using either `libname` or the `LIB` option of the `PROC R` statement.
  - If `libname` is specified, that location is always used.
  - If the `LIB` option of the `PROC R` statement is specified and `libname` is not specified, the location in the `LIB` option is used.
  - If neither `libname` nor the `LIB` option on the `PROC R` statement are specified, the `WORK` location is used.

- **catalog**: Specifies the name of the catalog.

- **entry**: Specifies the name of the R object in the catalog.

### R

Specifies the variable name for the loaded R object as used in the R language program.

\[
R = \text{robject-name}
\]

Because R is case-sensitive, `robject-name` must match the case used of the R variable name. The name can be specified using name literal syntax (for example "r.object.name") if the name of the R object does not follow the normal rules for identifiers in the SAS language.

An example of using the `LOAD` statement

```r
proc r;
    load cat=catalog.entry r='target.object'n;
run;
```
SAVE

Enables R objects to be serialised and stored in a SAS language catalog.

SAVE { SAVE_option ... } ;

The SAVE and LOAD statements enable an R object to be serialised and stored a SAS language catalog and later deserialised in a WPS session. The SAVE statement serialises an R object and stores it in an entry in a catalog. The LOAD statement deserialises an R object from a catalog.

Save options

The following options are available with the SAVE statement.

CATALOG
Species the location in which the R object will be saved.

[ CATALOG | CAT | C ] = [ libname ] catalog . entry

The library location can be specified using either libname or the LIB option of the PROC R statement:

libname

Specifies the name of the library in which the catalog is stored. The library location can be specified using either libname or the LIB option of the PROC R statement.

• If libname is specified, that location is always used.
• If the LIB option of the PROC R statement is specified and libname is not specified, the location in the LIB option is used.
• If neither libname nor the LIB option on the PROC R statement are specified, the WORK location is used.

catalog

Specifies the name of the catalog.

entry

Specifies the name of the R object in the catalog.

R

Specifies the name of the R object to save to the catalog. Must be specified.

R = robject-name

Because R is case-sensitive, the case of robject-name must match the object name in the R program. The name can be specified using name literal syntax (for example "r.object.name") if the name of the R object does not follow the normal rules for identifiers in the SAS language.

DESCRIPTION

Specifies a description string saved with the catalog entry.
DESCRIPTION = "Catalog entry description"

The description is displayed in the output from the PROC CATALOG statement.
The catalog entry will have a type of ROBJECT.

An example of saving an R object to a WPS catalog

```r
proc r;
  save cat=catalog.entry r='source.object'n;
run;
```

**SUBMIT**

Specifies the start of an in-line R language program.

```r
SUBMIT[{{ SUBMIT_option ... }}] ;
```

An in-line R language program is defined as part of the R procedure in a SAS language program. The SUBMIT statement marks the start of the program, and the ENDSUBMIT statement marks the end.

An R language program must start on a new line after the SUBMIT statement, and the ENDSUBMIT statement must appear at the beginning of a line on its own.

Multiple in-line R language programs can exist in a single R procedure environment. Each R language program is executed as it is encountered. Variables defined in one R language program can be used in subsequent in-line programs in the same R procedure environment.

**SUBMIT options**

The following option can be used with the SUBMIT statement.

**r-symbol**

Enables the replacement of a symbol in the R program with a string.

```r
r-symbol = 'substitution-value'
```

Before being passed to the R environment, an in-line R program is pre-processed to replace any r-symbol defined in the program with the content of the substitution-value. If the substitution-value is a string, it must be surrounded by quotation marks. If substitution-value is a SAS language macro variable, it must be prepended by an ampersand (&), but quotation marks are not required.
The r-symbol name is case sensitive and must be referred to in the R program using the same case as in the SUBMIT statement definition.

```
PROC R;
SUBMIT greeting = 'Hello World';
r.welcome <- "&greeting."
print (r.welcome)
ENDSUBMIT;
RUN;
```

The syntax for replacing a defined r-symbol is the same as that used for SAS language macro variable substitution, the name of the defined r-symbol is prepended by an ampersand (&) in the inline code.

**SUBMIT statement – basic example**

The following example shows how to incorporate an R-language program into a SAS language program:

```
PROC R;
SUBMIT;
  x <- (1:10)
  print(x)
ENDSUBMIT;
RUN;
```

Which outputs the following:

```
[1]  1  2  3  4  5  6  7  8  9 10
```

**SUBMIT statement – using a macro variable in an R program**

The following example shows how to pass a variable from a SAS language program to an R language program. The variable welcome is defined using the SAS language %LET macro. The specified macro variable is assigned to the r-symbol of the SUBMIT statement. The r-symbol is then referenced in the inline R language program:

```
%LET welcome = 'Hello World';
PROC R;
SUBMIT greeting = &welcome;
r.welcome <- "&greeting"
print (toupper(r.welcome))
ENDSUBMIT;
RUN;
```

Which outputs the following:

```
[1] "HELLO WORLD"
```
Legal Notices


All rights reserved. This information is confidential and subject to copyright. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system.

Trademarks

WPS and World Programming are registered trademarks or trademarks of World Programming Limited in the European Union and other countries. (r) or ® indicates a Community trademark.

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.

All other trademarks are the property of their respective owner.

General Notices

World Programming Limited is not associated in any way with the SAS Institute.

WPS is not the SAS System.

The phrases "SAS", "SAS language", and "language of SAS" used in this document are used to refer to the computer programming language often referred to in any of these ways.

The phrases "program", "SAS program", and "SAS language program" used in this document are used to refer to programs written in the SAS language. These may also be referred to as "scripts", "SAS scripts", or "SAS language scripts".

The phrases "IML", "IML language", "IML syntax", "Interactive Matrix Language", and "language of IML" used in this document are used to refer to the computer programming language often referred to in any of these ways.

WPS includes software developed by third parties. More information can be found in the THANKS or acknowledgments.txt file included in the WPS installation.