WPS Proc R
user guide and reference

Programming in the languages of SAS and R

Version: 4.0.3

www.worldprogramming.com

Contents

Introduction

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

This document provides the information you will need to configure and use the R procedure (`PROC R`) from within WPS.

Motivation

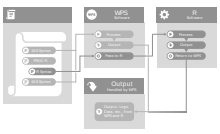
The R language is popular in significant segments of the data analysis community for the following reasons:

- It has established a strong position within the open source ecosystem.
- A large body of statistical functionality is available to the language.
- It may offer some statistical features not available in WPS.
- Many people are already competent R programmers and want to use these skills.

Putting R together with the language of SAS creates a solution where the whole is greater than the sum of its parts. This inter-operability allows the bulk of data processing and analytics to be written in the industrial strength and high-performing language of SAS, whilst exploiting novel and specialist statistical features that are present in the world of R.

The big picture

We recommend that programs are mainly coded in the language of SAS, dropping into R where specialist statistics are required:



It is simple to transfer data quickly and efficiently between WPS and the R environment using the `PROC R` syntax. Once data has been transferred, regular R syntax is used to write an R program in exactly the same way an R program would be written in a non-WPS environment.

Once an R program is complete, data can be readily transferred back to the WPS environment to continue processing.

This user guide and reference

To get started as quickly as possible, begin with the *Quick start* (page 5) section. For more background and detailed configuration information, read *Setup and configuration* (page 9). To understand how to perform typical computing operations, refer to *Basic operation* (page 14). For detailed information about the syntax and semantics of WPS' integration with R, study the *Reference* section towards the back.

Quick start

To get started quickly, you should have experience of both the language of SAS and the R language.

Before you start, check that:

- WPS version 3.1 or above and R (either version 2.15.x or version 3.x) are installed on your system. This short tutorial assumes a Windows installation of WPS, but the implementations on other operating systems are very similar.
- Your R installation includes the shared library API. This is included in default installations on Windows and macOS, but is not included by default on other platforms such as Linux. Refer to the *Setup and configuration* (page 9) section for more information.

  Warning:

  The R procedure will not operate correctly if your R installation does not include the shared library API.

- Automatic generation and management of ODS HTML is enabled within WPS Workbench. Set this via **Window ➤ Preferences ➤ WPS ➤ Result Options**.

The following example program:

- Specifies the location of the R installation.
- Creates a WPS dataset called `source`.
- Transmits the source dataset to the R environment.
  - Prints the dataset from within the R language.
  - Performs some simple processing and plotting of source data with the R language.
  - Creates a simple dataset called `x` in the R language.
- Within the WPS environment, retrieves the dataset `x` from the R environment.
- Prints the dataset `x` using the language of SAS

1. Launch WPS and create a new program (also known as a *script*).

2. Copy and paste the following code into the file and ensure that the path in the first line points to your local R installation, for example:

```
options SET=R_HOME "C:\Program Files\R\R-2.15.3";

data source;
  do x=1 to 10;
    y=ranuni(-1);
  output;
  end;

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;

proc print data=x;
run;
```

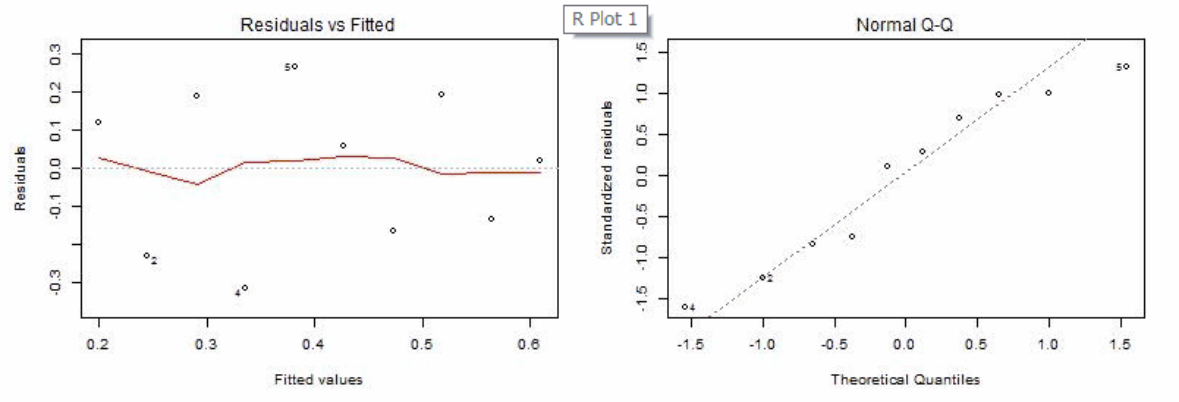
3. Save the program and run it by clicking the toolbar **Run** icon .
4. Examine the generated log and ODS output.

The standard R source echo is routed to the WPS log file:

```
NOTE: Submitting statements to R:

>     str(source)
>     print(source)
>
>     model <- lm(source$y ~ source$x)
>     print(model)
>     par(mfrow=c(2, 2))
>     plot(model)
>
>     x <- (1:10)

NOTE: Processing of R statements complete
```

The HTML output for the local server contains the output from the `print` and `plot` statements:

```
'data.frame':   10 obs. of  2 variables:
 $ x: num  1 2 3 4 5 6 7 8 9 10
 $ y: num  0.319 0.0167 0.4796 0.0208 0.6477 ...
    x          y
1   1 0.31896910
2   2 0.01672716
3   3 0.47964911
4   4 0.02080863
5   5 0.64773289
6   6 0.48497394
7   7 0.30747265
8   8 0.70942907
9   9 0.42861953
10 10 0.62938010
Call:
lm(formula = source$y ~ source$x)
Coefficients:
(Intercept)     source$x
    0.15361      0.04559
```



Obs	x
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10

Your WPS and R environments are now integrated, and you can explore the full range of possibilities available through having simultaneous access to both languages. The key concept is that you can place R code between the `submit` and `endsubmit` statements, employing other `PROC R` statements to manage data transfer betwen the two environments.

Setup and configuration

This section covers the setup and configuration of the WPS and R environment.

Required software

The R procedure is delivered with WPS version 3.1 and later, and both the 32-bit and 64-bit versions of WPS can be used. You do not have to install extra modules, and there are no special licensing requirements as the R procedure is part of the WPS core licensed software. However, the WPS software is not shipped with a copy of R, and to use the R procedure, you will need an installation of R on your computer.

Note:
At the time of writing, the two major versions of R in use are 2.15.x and 3.x. WPS software can be used with either version.

Refer to the *Platform-specific notes* (page 12) for how to install R on different platforms.

Setting the *R_HOME* environment variable

In order to be able to locate the installed version of R, WPS requires that the *R_HOME* environment variable is set.

This is not necessary on the Windows platform if the default R installation location is used, since WPS can retrieve the information from the Windows registry. If you are running WPS with R on a Unix or Linux platform, you will need to set the *R_HOME* variable to point to the folder containing `libr.so`. If you have multiple instances of R installed, *R_HOME* must point to the folder containing either `libr.dll` on Windows platforms or `libr.so` on UNIX or Linux platforms.

The ways in which you can set the *R_HOME* environment variable are dependent on your particular installation and usage scenario:

Installation and Usage Scenario	OPTIONS statement	Server start up environment variable	UNIX shell profile script	System level variable*
Standalone Workstation	Yes	Yes	Yes	Yes
Standalone Server	Yes	Yes	Yes	Yes
Remote Desktop Connection to Server	Yes	Yes	Yes	Yes

Installation and Usage Scenario	OPTIONS statement	Server start up environment variable	UNIX shell profile script	System level variable*
WPS Link Between Workstation and Server	Yes	N/A	Yes	Yes

* Please refer to your particular Operating System documentation to find out how to set a system level variable.

When setting the *R_HOME* environment variable externally on the platform of your choice, you can verify the variable using a SAS program containing the following:

```
%let EnvVar = %sysget( R_HOME ); %put "R_HOME is set to &EnvVar";
```

Setting *R_HOME* using the OPTIONS statement (all operating systems)

The simplest way to set the *R_HOME* environment variable is to use the OPTIONS statement in a WPS program.

1. Invoke the following code before the R procedure, substituting the path to your own R installation:

   ```
   options SET=R_HOME "C:\Program Files\R\R-2.15.3";
   ```

   This will set the *R_HOME* environment variable for the remainder of the WPS session. A key advantage of this method is that it enables you to use multiple versions of R within a single WPS session:

   ```
   OPTIONS SET=R_HOME "C:\Program Files\R\R-2.15.3";
   PROC R;
     SUBMIT;
       R.version
     ENDSUBMIT;
   RUN;

   OPTIONS SET=R_HOME "D:\Program Files\R\R-3.0.3";
   PROC R;
     SUBMIT;
       R.version
     ENDSUBMIT;
   RUN;
   ```

2. The output from this program resemble the following:

```
platform       x86_64-w64-mingw32
arch           x86_64
os             mingw32
system         x86_64, mingw32
status
major          2
minor          150.3
year           2013
month          03
day            01
svn rev        62090
language       R
version.string R version 2.15.3 (2013-03-01)
nickname       Security Blanket
```

and

```
platform       x86_64-w64-mingw32
arch           x86_64
os             mingw32
system         x86_64, mingw32
status
major          3
minor          0.3
year           2014
month          03
day            06
svn rev        6512
language       R
version.string R version 3.0.3 (2014-03-06)
nickname       Warm Puppy
```

Setting *R_HOME* via a WPS Server startup environment variable (all Operating Systems)

An alternative method for specifying the location of the R installation is to set the *R_HOME* environment variable using the **Startup** option of the WPS Server where both WPS and R are installed.

1. Ensure that you have logged onto the workstation or physical server where both WPS and R are installed.
2. Launch the WPS Workbench.
3. Right-click on the local WPS Server in the **WPS Server Explorer** tab and select **Properties.**

   The **Properties for the Local Server** dialog appears.
4. Ensure that the **Environment Variables** item is selected on the left-hand side of this dialog box, under **Startup**.
5. Click the **New** button, add the new environment variable, and click **OK** on the **Properties** dialog.
6. You are prompted to restart the server to apply your changes.

In a few seconds, the server will have restarted and the **WPS Server Explorer** will indicate that a set of **Libraries** and **Filerefs** are once again associated with it. WPS will use the new environment variable when it needs to locate the local R installation.

Setting *R_HOME* in a UNIX shell profile script

A final option for UNIX platforms is to set the value of *R_HOME* in a shell profile script.

For example, the environment variable could be set in a user's `~/.bashrc` file.

Note:
Due to the way in which applications are launched on macOS, this technique has no effect on the environment for WPS Workbench when launched, and so will not work on that platform.

Platform-specific notes

Windows

The standard Windows binary distribution from the R project website should be installed.

The package that can be downloaded at *http://www.r-project.org/* includes both 32-bit and 64-bit versions of R, so the same bundle works 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 will be the version of R most recently installed, and no special configuration of WPS is required for WPS to locate it. You can, however, set the *R_HOME* environment variable using an `OPTIONS` statement (page 10) or as a WPS Server startup option (page 11) if you want to use a specific version of R.

UNIX or Linux platforms

The required shared library is not included with the R binary distribution for UNIX platforms by default, and it is currently not possible to download a suitable pre-built binary distribution of R from the R project website.

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.

You require a minimal set of pre-installed libraries before you can build R from source code. These are equivalent to *build essentials* plus a JDK on Ubuntu. For more information, see the R documentation at *http://www.r-project.org/*

To install R from source code:

1. Download the R source code from *http://www.r-project.org/*.
2. Unpack the source code bundle.

   ```
   tar -xzf R-<version>.tar.gz
   ```

   This creates a directory called `R-<version>`.
3. Change to the new directory and configure the build process.

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

   Ensure you use the --enable-R-shlib option when running configure as this builds the `libr.so` shared library.
4. Compile and link the R software.

   ```
   make
   ```
5. Install the R software.

   ```
   make install
   ```

Once installed, set the *R_HOME* variable to point to the folder containing the `libr.so` file,

If you have installed R using your systems package management system, you need to locate `libr.so` on your system and set *R_HOME* to point to that folder.

Further instructions for installing R from source are available in the R documentation.

macOS

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

To use R with WPS, the *R_HOME* variable must to be set to point to the installation directory containing the `libr.so` shared library. The correct setting is: `/Library/Frameworks/R.framework/Resources`. This will use the default version of R.

Note:

Due to the way in which 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/R.framework/Versions/3.0/Resources`

Basic operation

Testing the installation and configuration of WPS and R

To test the installation and configuration of WPS and R, you can create and run a short program. It will also serve as an introduction to the `PROC R` syntax.

1. Create a new program file, and paste and save the following code:

```
proc r;
  submit;
    x <- (1:10)
    print(x)
  endsubmit;
run;
```

2. Run the program by clicking on the toolbar **Run** icon  .

   When the program has finished executing, the local server log should contain the elements shown in the following example:

```
4          proc r;
5          submit;
6              x <- (1:10)
7              print(x)
8              endsubmit;
NOTE: Using R version 3.0.3 (2014-03-06) from D:\Program Files\R\R-3.0.3

NOTE: Submitting statements to R:

>      x <- (1:10)
>      print(x)

NOTE: Processing of R statements complete

9          run;
NOTE: Procedure r step took :
      real time : 0.406
      cpu time  : 0.015
```

   The WPS software prints out the R version number and the location of the version of R that it is using. This will happen for each invocation of `PROC R`. The standard R source echo is also included in the WPS execution log, and HTML output for the local server contains output from the `print(x)` statement:

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

Passing data between WPS and R

The EXPORT and IMPORT statements allow you to pass data back and forth between WPS and R.

Using the EXPORT statement

The EXPORT statement of the R procedure is used to transmit data from the WPS environment to the R environment.

This might typically happen when some data that has been generated in prior steps using the language of SAS needs to be made available in an R context.

1. Create a new program file, paste the following code, and save the file:

```
data source;
  do x=1 to 10;
    y=ranuni(-1);
    output;
  end;

proc r;
  export data=source;
  submit;
    str(source)
    print(source)
  endsubmit;
run;
```

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

   By default, the EXPORT statement creates an R data frame with the same name as the WPS dataset. The data frame also has the same columns as the original WPS dataset.

```
'data.frame': 10 obs. of  2 variables:

 $ x: num  1 2 3 4 5 6 7 8 9 10

 $ y: num  0.67611 0.15623 0.00499 0.76839 0.71394 ...

    x           y

1   1 0.676114940
2   2 0.156228246
3   3 0.004988594
4   4 0.768386004
5   5 0.713939278
6   6 0.004663629
7   7 0.714222821
8   8 0.453278570
9   9 0.613537647
10 10 0.072553599
```

Using the IMPORT statement

The `IMPORT` statement of the R procedure is used to retrieve data from the R environment back into the WPS environment.

The `IMPORT` statement accepts vectors, matrices and R data frames and converts them into suitable WPS datasets.

1. Create a new program file, paste the following code and save the file:

```
proc r;
  submit;
    x <- (1:10)
  endsubmit;
import R=x;

proc print data=x;
run;
```

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

   HTML output from the `proc print` statement is shown below.

Note:

The R vector called `x` has been converted into a dataset called `x` in the language of SAS, containing a single column, also called `x`.

```
Obs x
1   1
2   2
3   3
4   4
5   5
6   6
7   7
8   8
9   9
10  10
```

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.

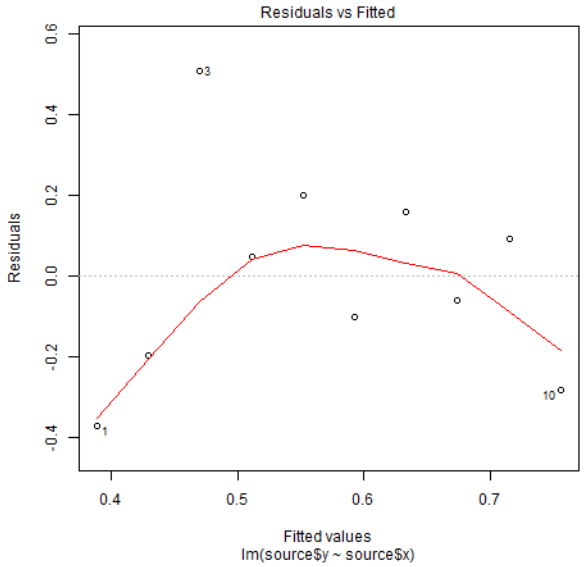
1. Create a new program file, paste the following code, and save the file:

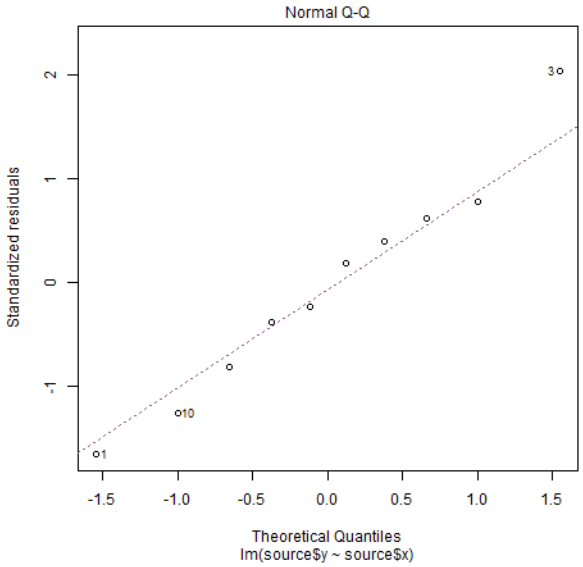
```
data source;
  do x=1 to 10;
    y=ranuni(-1);
    output;
  end;

PROC R;
  export data=source;
  submit;
    model <- lm(source$y ~ source$x)
    print(model)
    par(mfrow=c(2, 2))
    plot(model)
endsubmit;
run;
```

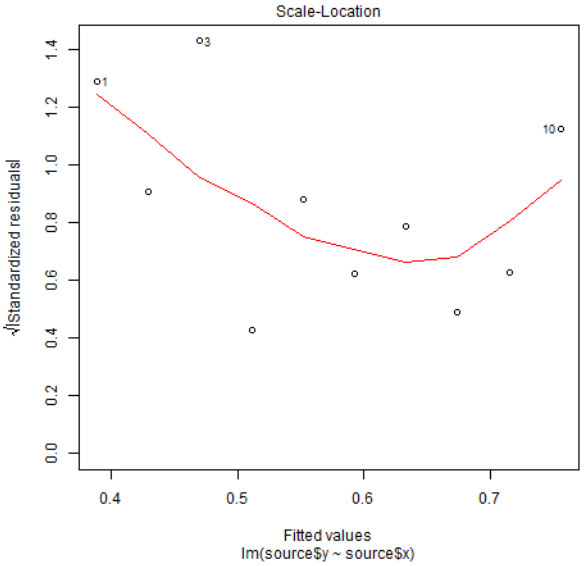
2. Run the program by clicking on the toolbar **Run** icon, and examine the HTML output.

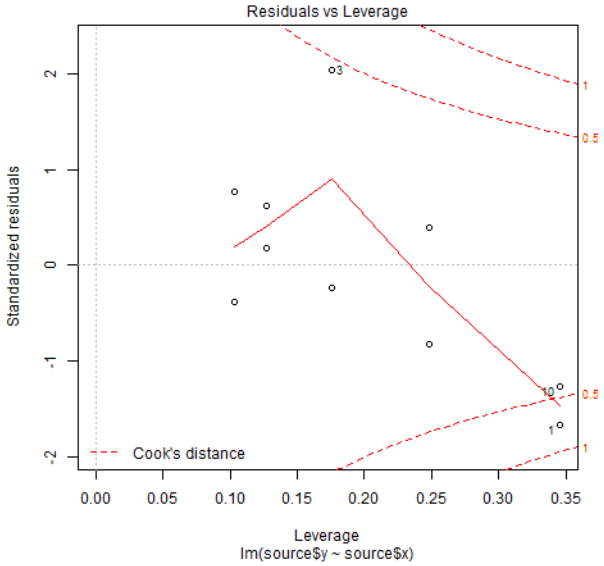
   The output includes the printed R results together with a single graphic generated within the R session and routed into the WPS output.

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


Residuals vs Fitted
Residuals
Fitted values
lm(source$y ~ source$x)


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(source$y ~ source$x)


Scale-Location
√|Standardized residuals|
Fitted values
lm(source$y ~ source$x)


Residuals vs Leverage
Standardized residuals
Cook's distance
Leverage
lm(source$y ~ source$x)

Using additional R packages

To use additional packages that are not included in the regular R installation, we recommend that the interactive R command line environment is used to perform the installation and to check the basic operation of these packages, before any attempt is made to use them within the WPS R procedure.

Note:

An R session launched by WPS inherits the list of environment variables from the WPS process. When run from the WPS Workbench on your computer, the local WPS server process in turn inherits its list of environment variables from the Workbench process. If third party software is installed for use within R that requires, for example, additional entries in the PATH environment variable, then WPS Workbench will need to be restarted to pick up the changes. Just restarting the local server in the Workbench is not sufficient for environment variable changes to take effect.

Reference

EBNF (Extended Backus-Naur Form) definitions are notations that help to explain the syntax of programming languages, and they are used in this guide to describe the language syntax.

How to read EBNF

EBNF is a textual notation that accompanies significant language structures such as procedures, statements and so on.

The description of each language concept commences with its EBNF representation .

Entering text

Text that should be entered exactly as displayed is shown in a typewriter font :

```
OUTPUT ;
```

This example describes a fragment of syntax in which the keyword `OUTPUT` is followed by a semi-colon character: `;`. The EBNF version of this fragment is simply the characters: `OUTPUT;`.

Generally the case of the text is not significant, but in this reference, it is the convention to use upper-case for keywords.

Placeholder items

Placeholders that should be substituted with relevant, context-dependent text are rendered in lower case, surrounded by a box :

`OUTPUT` *data-set-name* `;`

Here, the keyword `OUTPUT` should be entered literally, but *data-set-name* should be replaced by something appropriate to the program – in this case, the name of a dataset to add an observation to.

Optionality

When items are optional, they appear in square brackets `[ ]` in EBNF :

`OUTPUT` **[** *data-set-name* **]** `;`

Repetition

In EBNF, repetition is denoted by curly braces and ellipsis `{ … }`. An optional separator for multiple instances is shown just before the ellipsis.

```
OUTPUT { data-set-name ... } ;
```

Above, the keyword `OUTPUT` should be entered literally, and it should be followed by one or more repetitions of *data-set-name* - in this case, no separator other than a space has been asked for.

The example below shows the use of a separator.

```
function-name ( { argument, ... } ) ;
```

Choices

In EBNF, the choice is represented by a logical `OR` symbol | between each option.

```
GETNAMES [ YES | NO ] ;
```

In the above example, the keyword `GETNAMES` should be entered literally, and then either the keyword `YES` or the keyword `NO`.

Fragments

When the syntax is too complicated to fit in one definition, it might be broken into fragments:

```
PROC PRINT { option ... }
```

option

```
[ DATA = data-set-name | LABEL ]
```

Above, the whole syntax is split into separate EBNF fragments. The first indicates that `PROC PRINT` should be followed by one or more instances of an *option*, each of which must adhere to the syntax given in the second diagram.

Help with reading EBNF

The table below summarises the EBNF conventions used in this reference:

Convention	Description
`[]`	Optional elements
`{...}`	Repeated group of elements
`{,...}`	Separators that enclose iterations of repeated items are shown just before the ellipsis
|	Either/or within a selection group enclosed in `[ ]`. For example: • `[A|B]` – choose either `A` or `B` • `[A|B|]` – choose `A` or `B` or nothing (i.e. omit the group altogether)
`" "`	Comment – for example, a group title

Convention	Description
`monospaced`	Keywords and separators
italics	variables
bold	Elements presented in their own diagrams, like fragments
underlined	Default selection

R Procedure

The R procedure is invoked by the `PROC R` statement and allows execution of program code written in R.

Supported statements

PROC R

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

option

```
[  GMTOFFSET = "+/-HH:MM" |
   TIMESASCHRON |
   LIB = default-library |
   KEEP |
   TERMINATE |
   TERM
]
```

GMTOFFSET="+/-HH:MM"

Sets the offset to GMT to be applied when moving date-time values between WPS and R using the `ASSIGN`, `EXPORT` or `IMPORT` statements. Date-time values in WPS do not have an implied time-zone, whereas date-time values in R are represented in UTC (Coordinated Universal Time) with an associated time zone.

TIMESASCHRON

Controls whether time values are represented in R using the `chron` class. By default, time values are represented in R using a count of seconds from midnight, but optionally the `chron` package can be used. However, this package is not part of the standard R installation.

The option affects the `ASSIGN` and `EXPORT` statements.

LIB=default-library

Specifies the default library used by the `EXPORT`, `IMPORT`, `LOAD` and `SAVE` statements. The descriptions of these statements provide more details of the impact of this option.

KEEP

Specifies that the R application should be kept alive at the end of the procedure step and used for subsequent `PROC R` invocations. The default behaviour is to terminate the R application at the end of the procedure, although this can be altered using the `RKEEP` system option.

TERMINATE

Specifies that the R application should be terminated at the end of the procedure step. This is the default behaviour, although it can be changed using the `RKEEP` system option.

TERM

`TERM` is an alias for `TERMINATE`.

An example invocation of `PROC R`

```
proc r;
  submit;
    R.version
  endsubmit;
run;
```

ASSIGN

The ASSIGN statement can be used to assign a value to an R vector value.

```
ASSIGN r-object-name =
  [  value |
     ( { value, ... } )
  ] ;
```

value

```
[  int-literal |
   float-literal |
   date-literal |
   datetime-literal |
   time-literal |
   string-literal
]
```

Note:
All of the values on the right hand side of the assignment must be of the same type. For example, they must be all integers, or all strings.

The type of R vector created depends on the types of the values supplied:

Types of values supplied	Type of R vector created
Integers	Integer
Floating point	Real
Date	Real with a class of Date
Datetime	Real with a class of POSIXct, adjusted according to the GMTOFFSET option supplied to the PROC R statement
Time	Real with a class of times
String	String

Typically, this statement would be used to pass parameters into an R program. Often the values on the right of the assignment would be generated using macro variable expansion, or macro execution. With this facility, the need to perform macro expansion within the SUBMIT block itself is greatly reduced.

The name of the R object is specified as a normal identifier in the WPS language. Case is preserved when creating the R object. If necessary, a name literal can be used (as in "r.object.name"n) to create R objects having names that would not otherwise be valid in the language of SAS.

An example assignment of a value to an R object

```
%let parm=15;
proc r;
  assign parm=&parm;
  submit;
    x<-sample(1:3, parm, replace=TRUE)
    print(x);
  endsubmit;
run;
```

EXECUTE

The EXECUTE statement of the R procedure allows the execution of an R program stored in a file.

EXECUTE *"filename"* ;

Note:

Using the EXECUTE statement is an alternative to using the SUBMIT statement. It allows the R code to be placed into a separate file. This is useful because it allows the same program code to be executed directly in an interactive R environment. Where relevant, any relative path names are resolved relative to the current directory of the WPS process.

An example of executing an R program stored in a file

Contents of model.r source file:

```
    model <- lm(source$y ~ source$x)
    print(model)
    par(mfrow=c(2, 2))
    plot(model)
```

Sample R procedure invocation:

```
data source;
  do x=1 to 10;
    y=ranuni(-1);
    output;
  end;

PROC R;
  export data=source;
  execute "model.r";
run;
```

EXPORT

The EXPORT statement of the R procedure creates an R data frame from a WPS dataset.

```
EXPORT { option ... } ;
```

option

```
[   DATA = wps-data-set  |
    [ R | FRAME ] = r-object-name
]
```

DATA

Specifies the name of the dataset to export, along with any required dataset options. This option is required.

R or FRAME

Optionally specifies the name to give the object in the R environment. If omitted, the name of the R object is taken from the member name of the dataset.

Dataset options can be specified on the input dataset in the normal way, but it might be useful to apply a WHERE clause or a DROP list to the dataset before exporting it to R.

Note:

Applying a WHERE clause, or exporting a dataset from a sequential library or view (in other words, when the number of observations in the dataset is not known) requires additional resources as the dataset must first be spooled so as to calculate the exact number of observations before the R data frame can be created.

There are two types of variable in a WPS dataset: *numeric* and *character*. In addition, a numeric column can have a format associated with it that may be used to infer further type information. The type of R vector created is as follows:

WPS variable type	Type of R vector created
Character	Standard string.
Numeric variable with date format applied	Real R vector assigned a class of Date.
Numeric variable with a datetime format applied	A real R vector will be created and assigned a class of POSIXct. Values of this class represent a count of seconds since 1st Jan 1970 (Coordinated Universal Time). The values are adjusted based on the GMTOFFSET option passed to the PROC R invocation to take account of the fact that datetime values in the language of SAS are local time, whereas the values of the POSIXct class in R have to be in UTC.

WPS variable type	Type of R vector created
Numeric variable with a time format applied	There are two options based on whether the TIMESASCHRON option is specified by the `PROC R` invocation. By default, a normal real R vector is created and no special class is assigned to the vector. However, if the TIMESASCHRON option is specified, then the vector is assigned a class of `times`. The R `chron` package (that provides the `times` class) is not part of a standard R installation but provides utilities for handling time-of-day values.
Other numeric variables	A normal real R vector is created and no special class assigned.

When passing numeric values from WPS to R, the `EXPORT` statement interprets the special missing values `.I` and `.M` and creates the R values `Inf` and `-Inf` accordingly. Any other missing value is passed to R as the `NaN` value.

An example of exporting data from WPS to R

This example creates a dataset containing two numeric columns and exports it to R.

```
data source;
  do x=1 to 10;
    y=ranuni(-1);
    output;
  end;

proc r;
  export data=source;
  submit;
    str(source)
  endsubmit;
run;
```

The resulting data frame:

```
'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

The `IMPORT` statement of the R procedure creates a WPS dataset from an R object.

`IMPORT` **{** ***option*** **...** **}** ;

option

[`DATA` = *wps-data-set* **|**
[`R` **|** `FRAME` **]** = *r-object-name*
]

DATA

Optionally specifies the location into which the dataset should be saved. This can include dataset options. If omitted, the dataset is saved in the default library (either `WORK`, or the library named on the `LIB=` option on the `PROC R` invocation).

R or FRAME

Specifies the name of the R object to import. This must be in the form of an identifier, not a quoted string literal. A name literal can be used here to specify a name that is not normally valid for an identifier in the language of SAS. This option is required.

IMPORT conversion rules

Any object can be imported that can be coerced into a data frame using the `as.data.frame` R function.

If the specified R object cannot be coerced into a data frame then an error is produced. WPS can import columns that have the R logical, integer, real, and character types. In addition it can import factors. Columns with type logical, integer or real are converted into numeric columns in the WPS dataset. Columns of type character, and factors, are converted into string columns. The following notes provide more detail on these conversion rules.

Logical values

The values of vectors of type logical are converted as follows:

R Value	WPS Value
TRUE	1
FALSE	0
NA	.

Integer values

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

Real values

There are three special real numeric values in the R language, `NA`, `NaN`, and `Inf`. In R, `NA` is used to represent an absent value (Not Available), `Inf` denotes infinity (divide by zero for example), and `NaN` represents not-a-number (the result of 0/0 for example). These values are converted as follows:

R Value	WPS Value
NA	.
NaN	.
+Inf	.I
-Inf	.M

Date values

Integer or real columns that have an R class of `Date` have special processing applied to them. The variable in the WPS dataset is given a format of `DATE9` and when imported the values are adjusted to take account of the difference in the epoch used in R and WPS. Values of class `Date` in R are represented as a count of days since 1st Jan 1970, whereas, in the language of SAS, the epoch is 1st Jan 1960.

Date-time values

WPS will apply special handling to real columns that have class `POSIXct`. Values of this class in R represent a count of seconds since 1st Jan 1970 in UTC. When columns of this class are imported, the values are adjusted to take account of the difference in epoch between SAS and R. The column in the WPS dataset is given the format `DATETIME19`. The values are also adjusted according to the value of the GMTOFFSET option on the `PROC R` invocation to take into account that the values in R are in UTC, whereas, in the language of SAS, datetime values are in local time.

Time values

WPS will apply special handling to real columns that have class `times`. The column in the WPS dataset is given the format `TIME8`.

Character values

WPS will scan the values in the character column to find the longest value, and will assign the length of the WPS column to that value. Individual values in a character column can be Not Available (`NA`) in R, and these will be converted to the missing character value in WPS (that is the value will consist of all blanks). There will be no difference therefore between the values `" "` and `NA` when imported into WPS.

Factor values

A factor in R is a special form of integer column, where the values of the integer in the column are indexes into a list of unique values that is stored as an attribute on the column (these are called the levels in R). When imported into WPS these are converted into character variables in the dataset. The column is given a length equal to the longest string in the levels list.

An example of importing data from R to WPS

```
proc r;
  submit;
    x<-sample(1:3, 15, replace=TRUE)
  endsubmit;
import r=x data=demo_import;
run;
proc print data=demo_import;
run;
```

LOAD

The LOAD and SAVE statements allow R objects to be serialised and stored temporarily or permanently in a WPS data library and later deserialised in the same or a subsequent WPS session. The LOAD statement deserialises an R object that was previously saved with the SAVE statement.

LOAD { ***option*** ... } ;

option

[[CATALOG | CAT | C] = [*libname* .] *catalog* . *entry* |
[R | FRAME] = *r-object-name*
]

CATALOG or CAT or C

Gives the location in which the R object will be saved. This option is required. If the library isn't specified on the CATALOG option, then the default library is given by the LIB= option on the PROC R invocation, or otherwise, the USER or WORK library is used as normal.

R or FRAME

The R object name can be specified using name literal syntax (for example, "r.object.name"n) if the name of the R object doesn't comply with the normal rules for identifiers in the language of SAS. The case of the name is preserved when creating the R object. This option is required.

An example of using the LOAD statement

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

SAVE

The LOAD and SAVE statements allow R objects to be serialised and stored temporarily or permanently in a WPS data library and later deserialised in the same or a subsequent WPS session. The SAVE statement serialises an R object and stores it in an entry in a catalog.

```
SAVE { option ... } ;
```

option

```
[   [ CATALOG | CAT | C ] = [ libname . ] catalog . entry |
    [ R | FRAME ] = r-object-name |
    DESCRIPTION = "Catalog entry description"
]
```

CATALOG or CAT or C

This option, which is mandatory, gives the location in which the R object will be saved. If the library is not specified on the CATALOG option, then the default library is supplied by the LIB= option on the PROC R invocation : otherwise the USER or WORK library is used as normal.

R or FRAME

Specifies the name of the R object to save. This can be specified using name literal syntax (for example "r.object.name"n) if the name of the R object doesn't comply with the normal rules for identifiers in the SAS language. Since R is a case sensitive language, the case of the name must match that of the R object. This option is required.

DESCRIPTION

Gives a description string that is saved in the catalog entry. This description will be displayed in the output from the PROC CATALOG CONTENTS statement.

Note:

The catalog entry will have a type of ROBJECT.

An example of saving an R object to a WPS catalog

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

SUBMIT

The SUBMIT statement for the R procedure allows in-line program code in the R language to be executed.

```
SUBMIT
  [  { symbol = "substitution-value" ... }
  ] ; { R-language-statement ... } ENDSUBMIT ;
```

Note:

No changes to R program code are necessary. It is possible to copy and paste normal R language program code, surround it with SUBMIT and ENDSUBMIT statements, and invoke it from within the R procedure.

The R source code 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 SUBMIT blocks can exist within a single PROC R invocation. Each SUBMIT block is executed as it is encountered. SUBMIT blocks can be interleaved with other statements as required.

Macro processing

The nature of the R language means that the lines between the SUBMIT and ENDSUBMIT statements are copied verbatim and passed to the R environment. Macro processing is suspended between the SUBMIT and ENDSUBMIT statements. There are a number of reasons why this is the case:

- The R language uses the & and % characters as part of its syntax. Attempting to macro process the R source code may result in legitimate R syntax being misinterpreted as language of SAS macro invocations or macro variable wps-references.
- The R language allows line-end style comments, the contents of which may, for example, contain unmatched apostrophes. This would make it difficult to tokenise the R syntax using the regular language of SAS parsing rules, which is what would be necessary to allow macro processing of the R source code.

Also, due to the way the macro processor works and the way it handles source lines, it is not possible to generate a SUBMIT block using a macro. That is, a SUBMIT block cannot appear within a language of SAS macro. However, it is permitted that the SUBMIT block can appear in a file that is identified via an %INCLUDE statement. So, if it is necessary to generate a PROC R, invocation with the macro processor, it is necessary to either use the EXECUTE statement or put the contents of the SUBMIT block in a separate file that is then identified via an %INCLUDE statement.

Text substitution

In place of the macro processor, a simple text substitution facility is provided. Before being passed to R, the lines between SUBMIT and ENDSUBMIT can have a limited set of substitutions applied. The substitutions are given on the SUBMIT statement. The syntax for the substitutions is similar to that for normal macro variable substitution. However, there is no rescanning, and only simple single-level macro variable style syntax is allowed.

```
&symbol
&symbol.
```

Even with this simple syntax and only substituting symbols explicitly listed on the SUBMIT statement, it is possible that unintended substitutions may occur. To avoid this, it is recommended that symbol names are chosen that are different from any R object names wps-referenced in the submitted R code. The ampersand symbol (& and &&) is used in R as the logical *and* operator. Consider the case of a SUBMIT statement containing the following R code:

```
a>b&c<d
```

If *c* is defined as a substitution symbol, this will result in the unintended substitution of *c* within this expression. Including space around the ampersand will prevent this, as will choosing the substitution symbol names so that they are less likely to clash with R object names.

There is no escape syntax and there is no way to prevent a symbol from being substituted. Consider the following case:

```
proc r;
  submit Goodbye="hello";
    A <- "Hello&Goodbye"
  endsubmit;
run;
```

It is not possible to prevent substitution in this case, other than by choosing a different name for the symbol: the ampersand is not a special symbol as it is used as a logical operator within R.

It is possible to generate substitution values using normal macro processor facilities, as in the following example:

```
proc r;
  submit Goodbye="hello";
    A<-"&sym"
endsubmit;
run;
```

Example of submitting in-line R code

```
proc r;
  submit;
    x <- (1:10)
    print(x)
  endsubmit;
run;
```

Further reading

A suggested starting point for further information, including syntax, semantics and many add-on packages that extend the utility of the basic language features, is the R project website at *http://www.r-project.org*.

Legal Notices

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.