The StatRep User's Guide

Tim Arnold and Warren F. Kuhfeld
SAS Institute Inc.

June 10, 2014

Contents

1 Synopsis

The StatRep system consists of a LaTeX package and a suite of SAS macros that support SAS users who want to create documents with reproducible results.

The LaTeX package provides two environments and two tags that work together to display your SAS code and results and to generate the SAS program that produces those results. The two environments (`Datastep` and `Sascode`) display SAS code. The two tags (`\Listing` and `\Graphic`) display SAS output.

The generated SAS program includes calls to the StatRep macros that use the SAS Output Delivery System (ODS) document to capture the output as external files. These SAS macros are included in the package file `statrep_macros.sas`. The package is available at http://support.sas.com/StatRepPackage

Bundled with the StatRep package is the `longfigure` package for multipage figures.

2 Introduction

At the end of a research project, one of the most difficult tasks remains: documentation. The task is especially difficult with computational research because you must ensure that the displayed program code works as expected and exactly produces the displayed output.

The StatRep package, a single-source document system, is an open-source software project that you can use for your own research documentation to ensure that the results you display can easily be reproduced by your readers. The StatRep package is based on the LaTeX typesetting system. You write your paper using both the usual LaTeX markup and the customizations and SAS macros that this package provides. The system reads the code and markup from the single source (your document) and creates a SAS program. This automatically generated SAS program produces the results that are displayed in your document.

Comparable projects such as Sweave (Leisch 2002) and SASweave (Lenth 2007) address the problem of reproducibility through the use of a special intermediate language. Although similar in spirit to those systems, StatRep differs in that it is a normal LaTeX package; no special steps are needed to create the LaTeX file or the SAS program. In addition, StatRep provides both a complete, customizable system for automatic handling of multiple outputs and page breaking and an easy-to-use, flexible method for output selection.

When you use the StatRep LaTeX package, you follow a four-step process to create an executable document that enables you to create reproducible research results:

1. Create your LaTeX source file so that it contains your text, data, and SAS code.

2. Compile the document with pdfLaTeX. You can use a LaTeX-aware editor such as TeXworks, or you can use the command-line command `pdflatex`. This step generates the SAS program that is needed to produce the results.

3. Execute the SAS program to capture your output.

   For each code block in your document, SAS creates a SAS Output Delivery System (ODS) document that contains the resulting output. For more information about ODS documents, see the *SAS Output Delivery System User's Guide*. For each output request in your document, SAS replays the specified output objects to external files. All of your requested output is generated and captured when you execute the generated SAS program.

4. Recompile the document with pdfLaTeX. This step compiles your document to PDF, this time including the SAS results that are generated in the preceding step.

   In some cases listing outputs may not be framed properly after this step. If your listing outputs are not framed properly, repeat this step so that LaTeX can remeasure the listing outputs.

When you need to make a change in your data or SAS code, you make the change in one place (the LaTeX source file) and repeat steps 2 through 4. Your changes are automatically displayed in your code and in your results. You perform the steps only as needed—when you change your data or code.

You can share your LaTeX source with colleagues and be sure that your results are reproducible. Any SAS user can reproduce your analysis with your LaTeX document and the supplemental files that are described in this manual.

2.1 Requirements for the StatRep Package

To use the StatRep package, you need SAS 9.2 or later, the LaTeX typesetting system (the pdfLaTeX typesetting engine must be version 1.30 or later), and the StatRep package itself.

For complete step-by-step instructions for installation, see section 7.1.

2.2 Package Usage

To use the StatRep package, include it in your document preamble after you declare the `documentclass`. Figure 1 displays an example of how you can use the StatRep package.

```
\documentclass{book}
\usepackage[figname=output,resetby=chapter]{statrep}
```

Figure 1: Example of Using the StatRep Package

The StatRep package supports the following options:

- `generate` specifies whether a SAS program is generated at compile time. It can have a value of `true` or `false`; the default is `true`.
- `figname=` specifies the name of a LaTeX counter that is used for numbering outputs. The default is `figure`. If you specify a value for the `figname` option for which no counter exists, a counter is created.
- `resetby=` specifies that the counter for output numbering be reset with each change in the specified counter value. For example, if `resetby=chapter`, all output numbering is reset when the chapter value changes.

The options `figname=` and `resetby=` are not used directly by the StatRep package but are passed to the `longfigure` package, which is provided with the StatRep package. The `longfigure` package supports display and page breaking within a stream of outputs, and it can be used independently of the StatRep package. See section 7.2 for more information.

3 Getting Started

This section provides a simple example of how you can use the StatRep package to produce a document with reproducible results.

Two code environments (`Datastep`, shown in Figure 2, and `Sascode`, shown in Figure 3) and two output tags (`\Listing` and `\Graphic`, shown in Figure 4) are used to generate a SAS program that produces the necessary output files.

The code from the `Datastep` environment is passed unchanged to the generated SAS program.

```
\begin{Datastep}
proc format;
   value $sex 'F' = 'Female' 'M' = 'Male';
data one;
   set sashelp.class;
   format sex $sex.;
run;
\end{Datastep}
```

Figure 2: Example of `Datastep` Environment

The code in the `Sascode` environment is parsed by the StatRep package before it is written to the generated SAS program.

```
\begin{Sascode}[store=class]
proc reg;
    model weight = height age;
run;
\end{Sascode}
```

Figure 3: Example of `Sascode` Environment

The `\Listing` and `\Graphic` tags convey information to LaTeX and to SAS. The tags specify the names of the output files to insert into the document and the captions for the output. Additionally, they specify the names of the output files to create and which ODS objects to capture.

```
\Listing[store=class,
         caption={Regression Analysis}]{rega}

\Graphic[store=class, scale=0.9,
         caption={Graphs for Regression Analysis}]{regb}
```

Figure 4: Example of `Listing` and `Graphic` Tags

Figure 5 shows the SAS code that is generated from the preceding LaTeX source when you compile the document.

generated from **Datastep Block**

```
proc format;
   value $sex 'F' = 'Female' 'M' = 'Male';
data one;
   set sashelp.class;
   format sex $sex.;
run;
```

generated from **Sascode Block**

```
%output(class)
proc reg;
    model weight = height age;
run;
%endoutput(class)
```

generated from **Listing & Graphic Tags**

```
%write(rega,store=class,type=listing)

%write(regb,store=class,type=graphic)
```

Figure 5: Generated SAS Code

When you generate the SAS program by compiling your LaTeX document, the lines in the `Datastep` environment are passed unchanged to the program and the lines in the `Sascode` environment are wrapped between two SAS macros (`%output` and `%endoutput`), whose definitions accompany this package (`statrep_macros.sas`). The macros and their options are discussed in detail in section 4.2.

The `\Listing` tag results in a call to the `%write` macro that selects all notes and tables from the ODS document. The `\Graphic` tag results in a call to the `%write` macro that selects all graphs from the ODS document.

When you execute the generated SAS program that is displayed in Figure 5, the SAS results created in the `Sascode` block are contained in the ODS document `class`. The `%write` macro writes the requested results from the ODS document to the specified external files.

When you compile your LaTeX document again, the `\Listing` and `\Graphic` tags

insert the requested SAS results, handling page breaks automatically.

4 Syntax

4.1 StatRep LaTeX Environments and Tags

The StatRep package provides two environments and two tags that work together to display your SAS code and results and generate the SAS program that produces those results.

The environments:

- The `Datastep` environment contains SAS code blocks that produce no output. Its purpose is to read in data.
- The `Sascode` environment contains SAS code that generates output to be captured. It supports line-based commands to identify code lines that should only be displayed, only passed to the generated program, or both displayed and passed to the generated program.

The tags:

- The `\Listing` tag provides information to the generated program about which tabular SAS output should be captured. It also provides information to LaTeX about how that output should be displayed.
- The `\Graphic` tag provides information to the generated program about which graphical SAS output should be captured. It also provides information to LaTeX about how that output should be displayed.

The environments and tags are described in detail in the following sections.

Datastep Environment

The purpose of the `Datastep` environment is to read in data. It produces no SAS results.

The `Datastep` contents are passed unchanged to the generated program. The `Datastep` block is indented by three spaces in the PDF file. You can adjust the amount that the block is indented; see section 5.1 for details. The block indent is provided automatically so that your data and program lines can begin in the first column in your LaTeX source.

Because you begin the `Datastep` data lines in the first column, formatted or column input statements will work correctly when pasted into a SAS session.

Although the purpose of the `Datastep` environment is to read in data, it can contain any SAS code that does not generate output to be captured. Additional statements

typically include TITLE and OPTIONS statements and PROC FORMAT steps. See Figure 10 on page 27 for an example.

Table 1 summarizes the `Datastep` environment options.

Table 1: Commonly Used `Datastep` Environment Options

Option	Action
	By default, all lines are displayed and written to the program.
`program`	Specifies that all lines in the environment be written to the generated program only (that is, no lines are displayed). This option is useful when you need to produce a data set that is not central to the topic being discussed and does not need to be displayed.
`display`	Specifies that all lines in the environment be displayed only (that is, no lines are written to the program). This option is useful when you need to show code fragments that will not run as is or example code that is not needed for later output generation. A `Datastep` environment that specifies the `display` option is similar to a plain `verbatim` environment except that it is automatically indented when displayed.
`first=`*n*	Specifies that the first *n* lines in the environment be displayed. The option affects only the displayed code block. This option is useful when you have many data lines that do not need to be displayed, but that must be available to the program. After the *n*th line is displayed, the following text line is written in the displayed code block: `... more data lines ...` You can specify different text to be used; see section 5.1 for details.
`last=`*m*	Specifies that the last *m* lines in the environment be displayed. The option affects only the displayed code block. This option is used in conjunction with the `first=` option to show the ending lines of the `Datastep` environment. Without the `first=` option, the `last=` option has no effect.
`fontsize=`	Specifies the LaTeX font size used to display the code block. For exampe, `fontsize=small` or `fontsize=footnotesize`.

See the section Using the Datastep Environment for an example.

Sascode Environment

The purpose of the `Sascode` environment is to generate output. In addition to the environment options, it supports line commands that enable you to specify certain lines as display-only or program-only.

The Sascode environment is parsed for line commands, and the appropriate lines are passed to the program and displayed. The displayed code block is indented by three spaces. You can adjust the amount the block should be indented; see section 5.1 for details. The block indent is provided automatically so that your program lines can begin in the first column in your LaTeX source.

Because all line commands are valid SAS statements, you can copy Sascode blocks and paste them directly into a SAS session.

Table 2 summarizes the Sascode environment options.

Table 2: Commonly Used Sascode Environment Options

Option	Action
	By default, all lines are displayed and written to the program.
store=	Specifies the name of the ODS document to contain the SAS output. When you specify the store= option, the StatRep package writes the appropriate %output and %endoutput macros to the generated file. If you are writing the macros yourself, you must omit the store= option and call the %output (and optionally the %endoutput) macros from within the Sascode environment. When you want to display output from the ODS document, you refer to the store= name from a \Listing tag, a \Graphic tag, or by an explicit call to the %write macro. You can refer to the name in \Listing and \Graphic tags anywhere in your document, even in sections before, after, and far removed from the Sascode block that generated the output.
program	Specifies that all lines in the environment be written to the program only (that is, no lines are displayed). This option is useful when you need to execute code that is not central to the topic being discussed and need not be displayed.
display	Specifies that all lines in the environment be displayed only (that is, no lines are written to the program). This option is useful when you need to show example code fragments that will not run as is or that are not needed for later output generation. A Sascode environment that specifies the display option is similar to a plain verbatim environment except that it is automatically indented when displayed.
fontsize=	specifies the LaTeX font size used to display the code block (for example, fontsize=small or fontsize=footnotesize).

The Sascode environment also supports a finer degree of control with line-based commands to identify lines that should be only displayed or only passed to the generated program.

Table 3 summarizes the line commands you can use in the `Sascode` environment.

Table 3: `Sascode` Line Commands

Option	Action
`%* program` *n* `;`	The next *n* lines are only written to the program and not displayed.
`%* display` *n* `;`	The next *n* lines are only displayed and not written to the program.
`%*;` *code line*	The current line is only written to the program and not displayed.

The `Sascode` environment is parsed for line commands before being written to the generated program file.

See the section Using the Sascode Environment for an example.

By using a combination of environment options and line commands, you have complete control over the displayed code and the generated program contents.

Listing and Graphic Tags

The `\Listing` and `\Graphic` tags specify the outputs to be displayed. The purpose of the `\Listing` tag is to display tabular output and notes. The purpose of the `\Graphic` tag is to display graphical output.

All figures are centered. If the figure width is narrower than the text block, the figure is centered with respect to the text block. Otherwise, the figure is centered with respect to the page.

The `\Listing` and `\Graphic` tags support a set of options and have one mandatory argument, which specifies the filename prefix for the output to be generated and displayed. The prefix must be unique; otherwise the output from one example will overwrite another.

Furthermore, the prefix must not end in a numeral so that the prefix name does not interfere with SAS-generated output file names. When SAS generates a set of files from one ODS selection, it follows a pattern: the first file that is generated is identical to the filename, the next file that is generated has the same name with a "1" appended to it, the next file has the same name with a "2" appended, and so on.

The options supported by the `\Listing` and `\Graphic` tags are used by the StatRep LaTeX package and by the StatRep SAS macros.

Options Used by the StatRep LaTeX Package

The following options are used by the StatRep LaTeX package.

caption= specifies the caption to use for an output.

fontsize= specifies the LaTeX font size to use to display an output (for example, `fontsize=small` or `fontsize=footnotesize`).

linesize= specifies the line size used to generate and display `Listing` output. By default, the value is 80 columns. This specification lasts for the duration of this step. The current line size is restored at the end. Typical values are `80`, `96`, or `120`.

For extremely wide output tables, you can use the `linesize` and `fontsize` options together (for example, `linesize=120` and `fontsize=scriptsize`). The `linesize` option affects how SAS captures the table. The `fontsize` option specifies how LaTeX displays the table.

scale= specifies a factor by which to scale a `Graphic` image. For example, specify `scale=0.5` to scale the image to half its original size, or specify `scale=2` to scale it to double its original size.

store= specifies the name of the ODS document that is created in a `Sascode` environment. When you specify the `store=` option, the StatRep package adds the appropriate SAS macro calls to the generated program.

width= specifies the width to generate and display `Graphic` output The default is 6.4 inches, which is the standard width for ODS graphs.

Options Passed to the StatRep SAS Macros

The `store=`, `linesize=`, and `width=` options described in the previous section are passed to the StatRep SAS macros.

In addition, the following options are passed to the StatRep SAS macros:

dpi= specifies dots per inch (DPI) to use in generating graphs. The default is `dpi=300`. A typical alternative is `dpi=100`.

firstobj= specifies the first data object to capture in an output stream. All objects after and including the specified object are displayed, up to the final object (or optionally up to the object specified in `lastobj=`). You can use `options=skipfirst` to begin with the object after the one specified in `firstobj=`. See section 5.4 for details.

height= specifies the height of graphs. The default is 0.75 times the width.

lastobj= specifies the last data object to capture in an output stream. All objects starting with the first object (or optionally the object specified in `firstobj=`) are displayed up to and including the specified object. You can use `options=skiplast` to end with the object before the one specified in `lastobj=`.

linesize= specifies the line size used to generate and display `Listing` output. By default, the value is 80 columns. This specification lasts for the duration of this step. The current line size is restored at the end. Typical values are `80`, `96`, or `120`.

objects= specifies a space-separated list of ODS objects to capture in an output stream. The names that are used for selection come from the ODS document. If you specify `objects=`, then you can also specify object breaking rules (where page breaks can occur). See section 5.4 for details.

options= specifies binary options. Specify one value or a space-separated list of values (for example, `options=skipfirst skiplast`). You can specify the following values (the default is `options=autopage`):

autopage specifies that the first `\Listing` command or `%write` macro start a new output stream with titles, procedure titles, and so on. Page breaks also occur at other places where the procedure explicitly sets a page break. The `autopage` value is the default. See also the `nopage` and `newpage` values.

graph specifies that only graphs be selected. You can alternatively specify `type=graph`.

list specifies that the contents of the ODS document be listed in the SAS log. This value does not run PROC DOCUMENT to replay the output.

newpage specifies that SAS force a new page for the first object.

nopage suppresses page breaks.

onebox groups all tables, notes, reports, and so on into a single piece of SAS output. You cannot specify this option to group graphs. See section 5.4 for more information about grouping.

skipfirst modifies the `firstobj=` option so that the first object in the list is not selected. This enables you to select all objects after the one specified in `firstobj=`.

skiplast modifies the `lastobj=` option so that the last object in the list is not selected. This enables you to select all objects before the one specified in `lastobj=`.

table selects all objects (tables, notes, reports, and so on) except graphs. You can alternatively specify `type=listing`.

pagesize= specifies the page size. The default is the page size currently in effect. This specification lasts for the duration of this step. The current page size is restored at the end.

If you have not changed the page size, the default page size set by the StatRep package is 500. This large page size is the default so that output is generated with minimal new pages caused by page boundaries. For large tables, you can specify a smaller page size to force more page breaks. See section 5.1 for

information about how to change the StatRep default. See section 6.7 for information about how to use the `pagesize=` option with large tables.

`pattern=` provides an optional and additional selection criterion. Specify part of a path (for example, a group name). Only objects whose name includes the specified value are selected.

`store=` specifies the name of the ODS document that is created in a `Sascode` environment. When you specify the `store=` option, the StatRep package adds the appropriate SAS macro calls to the generated program.

`style=` specifies the ODS style to use in generating output. The default is `style=Statistical`. You can change the default style (for example, to `HTMLBlue`) by inserting the following line into a `Sascode` or `Datastep` environment:

```
*; %let defaultstyle=HTMLBlue;
```

This option affects ODS graphs only when used in a `\Graphic` tag. You can specify this option in the `%output` macro to set the style for GRSEG graphs (graphs that are produced by legacy SAS/GRAPH procedures such as the GPLOT, GMAP, and GCHART procedures). GRSEG graphs are stored in catalogs and cannot be changed after they are generated. In contrast, style, DPI, and so on for ODS graphis can be changed after the graph is initially created. See section 5.5 for more information.

`type=`*`listing|graph`* specifies that only listings or only graphs be selected. You can alternatively specify `options=table` or `options=graph`.

`width=` specifies the width to generate and display `Graphic` output The default is 6.4 inches, which is the standard width for ODS graphs.

4.2 StatRep SAS Macros

SAS programs created by the StatRep package run SAS macros to capture output. The macros depend on the ODS document. The ODS document is a destination or repository for the results (tables, notes, titles, and graphs) that come from SAS procedures. Each procedure step is run only once, and the results are captured in an ODS document. Then the parts of the ODS document are replayed using PROC DOCUMENT.

When StatRep encounters a `Sascode` environment, it generates a macro call to create an ODS document from the environment. When StatRep encounters a `\Listing` or `\Graphic` tag, it generates a macro call to replay output from that ODS document into an external file.

For the output that is generated in each `Sascode` block, the SAS macros provide you with a list of all of the objects in the ODS document and a table that displays

the objects selected for display. You can review this list in the SAS log. It is important to check these lists to ensure that either all output is included somewhere or any omissions are deliberate.

The following macros are defined:

- The `%output` and `%endoutput` macros open and close an ODS document, respectively. When you use StatRep to automatically generate your program, these macros are called at the beginning and end of a `Sascode` block, respectively. You can manually call the macros at any time within a `Sascode` block by prefixing the call with a null SAS macro comment (`%*;`).
- The `%write` macro writes ODS objects that are contained in an ODS document to one or more external files. When you use StatRep to automatically generate your program, this macro is called when a `\Listing` or `\Graphic` tag is encountered. You can manually call the macro at any time within a `Sascode` block by prefixing the call with a null SAS macro comment (`%*;`).
- The `%startlist` and `%endlist` macros capture printed content (for example, a PROC PRINT or DATA step) to an external file. To use these macros, you must manually call the macro within a `Sascode` block and prefix the call with a null SAS macro comment (`%*;`).
- The `%startlog` and `%endlog` macros capture content from the SAS log to an external file. To use these macros, you must manually call the macro within a `Sascode` block and prefix the call with a null SAS macro comment (`%*;`).

The `%output` and `%endoutput` Macros

The `Sascode` environment writes the `%output` and `%endoutput` macros to the generated program whenever the `store=` option is specified. However, you can call the macros yourself by omitting the `store=` option in the `Sascode` environment and call the macros within the `Sascode` environment. Each call must be prefixed with a null SAS macro comment (`%*;`).

The `%output` macro supports two other options (`style=` and `dpi=`) that are used to set parameters for GRSEG graphs. These options are not supported in the `Sascode` environment options. If you want to change the style or DPI for GRSEG graphs, you must call the `%output` and `%endoutput` macros manually.

The following options are supported by the `%output` macro:

`store=` specifies the name of the ODS document. This name is used in the `store=` option in the `\Listing` and `\Graphic` tags or in the `%write macro`.

`style=` specifies the style used for GRSEG graphs. The default is `HTMLBlue`. See section 5.5 for details.

`dpi=` specifies the dots per inch (DPI) setting used for GRSEG graphs. The default is 300 DPI. See section 5.5 for details.

The `%write` Macro

The `%write` macro supports the same key-value options as the `\Listing` and `\Graphic` tags support. The StatRep package generates the `%write` macro in the SAS program file whenever the `store=` option is specified in the output tag. However, you can call the macros yourself by omitting the `store=` option in the `\Listing` or `\Graphic` tag and call the macro within a `Sascode` environment. Each call must be prefixed with a null SAS macro comment (`%*;`).

See the section Options Passed to the StatRep SAS Macros for more information about the options used in the `\Listing` and `\Graphic` tags that are passed to the `%write` macro.

Note: If you use the macros interactively, be aware that these macros open and close ODS destinations, enable and disable ODS Graphics, and change ODS options. Output capture uses the LISTING destination, and when the `%write` macro finishes, only the LISTING destination remains open. If you need other ODS destinations for your work, you need to reset them when you are done with a section of output capture. For example, if you are using the HTML destination in the SAS windowing environment, then you need to close the LISTING destination and reopen the HTML destination when you finish capturing output.

The `%startlist` and `%endlist` Macros

The `%startlist` and `%endlist` macros capture printed information from the ODS listing destination. The macros are used when you create output with a procedure that does not support ODS. They are also used when you use the `DATA _NULL_` and `PUT _ODS_` SAS statements to manually capture output.

These macros are not automatically generated by the StatRep package. You must call them manually within a `Sascode` environment (with each call preceded by a null SAS macro comment).

In SAS 9.2, PROC PRINT is not fully integrated into the ODS document, unless you specify PROC PRINT with no options. If you specify PROC PRINT with options, you must use the `%startlist` and `%endlist` macros to capture output.

If you use PROC PRINT in SAS 9.3 or later, you can use the `%output` and `%write` macros as you would with any SAS procedure.

The `%startlist` macro has one mandatory argument, the filename prefix of the file to contain the output. The argument is also used in the `\Listing` tag to insert the output.

The `%startlist` macro supports the following options after the filename argument, separated with a comma (,):

`linesize=` specifies the line size. The default is the line size currently in effect. This specification lasts for the duration of this step. The current line size is restored at the end. When you specify the `linesize=` option in the `%startlist`

macro, be sure to make the same specification in the `\Listing` tag you use to insert the output.

`pagesize=` specifies the size of the output page. The default is the page size currently in effect. This specification lasts for the duration of this step. The current page size is restored at the end. This option is useful for breaking up long listings into smaller parts to allow for page breaks. If a listing output spans more than one SAS page, the output is automatically split into parts and the page breaks can occur only between parts of output. For more information about capturing large outputs, see section 6.7.

The `%startlog` and `%endlog` Macros

The `%startlog` and `%endlog` macros capture SAS notes or error messages from the SAS log. They also capture output from some SAS/IML functions that write to the SAS log rather than using ODS.

These macros are not automatically generated by the StatRep package. To capture content from the SAS log, you must call these macros manually within a `Sascode` environment (with each call preceded by a null SAS macro comment).

The `%startlog` macro has one mandatory argument, the filename prefix of the file to contain the output. The argument is also used in the `\Listing` tag to insert the output.

The `%endlog` macro supports the following options, separated by a comma (,):

`code=` specifies whether program code in the SAS log is included. By default, code is captured (`code=1`). Set `code=0` to exclude code.

`range=` specifies a Boolean expression to select certain observations. For example, you can specify `range=_n_ <= 5` to select the first five lines. You can specify `range=not index(line, 'ERROR')` to select all lines that do not contain the string 'ERROR'. Selection must be based on `_n_` or the variable `line`, which contains a single line of the log.

Macro Variable Defaults

The SAS macro defaults are set globally in the file `statrep.cfg`. See section 5.1 for details. You can also reset the defaults within your document by specifying new settings in a `Sascode` environment.

Table 4 shows the description and default values for each macro variable.

Table 4: Default Values for Macro Variables

Macro Variable	Default	Description
defaultlinesize	80	Line size for tabular ODS output
defaultpagesize	500	Page size for tabular ODS output
defaultstyle	Statistical	ODS style for graphical output
defaultdpi	300	Dots per inch (DPI) for graphical output
graphtype	png	graphics file format ('png' or 'pdf')
odsgraphopts		string containing ODS graphics options

You can edit the `statrep.cfg` file to globally reset the defaults, or you can specify commands to change the default anywhere in your document. For example, the following lines change all of the macro variable default settings for the duration of the program. The `program` option specifies that the code be written only to the generated program and not displayed.

```
\begin{Sascode}[program]
%let defaultlinesize=96;
%let defaultpagesize=50;
%let defaultstyle=statistical;
%let defaultdpi=100;
%let graphtype=pdf;
%let odsgraphopts=antialiasmax=10000;
\end{Sascode}
```

Figure 6: Reset SAS Macro Defaults within Document with SAS

When you change the `defaultlinesize` in the `statrep.cfg` file, the same value is automatically used by SAS and by the StatRep package. When you change the `defaultlinesize` inside your document, you change the line size used by SAS in generating outputs; You must also set the line size in the `\Listing` tag to match.

The setting of `defaultpagesize=500` produces a large virtual page so that SAS does not break ODS objects into smaller pieces. When a stream of outputs is typeset, page breaks can occur only between ODS objects or when SAS forces a page break inside an ODS table.

The macro options and default macro variables work as follows: If an option is specified in a macro, its value is used regardless of the specification in the default macro variables. If an option is not specified in a macro, the default macro variables provide the values.

In summary, a direct option specification in a macro takes precedence over the default settings, and you can change the default settings by resetting the default macro

variables in your document or by editing the `statrep.cfg` file.

5 Details

The following sections provide details about how you can use the StatRep system to capture complex output, customize its behavior, and take complete control of the generated SAS program.

In most cases, the default behavior and settings satisfy the typical usage requirements. In exceptional cases you might find that you need to make changes in how StatRep behaves. The following sections provide you with this information.

5.1 Customizing StatRep

You can modify the configuration file `statrep.cfg` to change the following settings used by the StatRep package. See section 4.2 for more information about macro variable defaults.

`\SRcaptionfont` specifies the font for the output captions. The default is `\sffamily` (sans serif).

`\SRcaptioncontinuedfont` specifies the font for the `continued` name for outputs that break across pages. The default is `\sffamily\itshape` (*sans serif, italic*).

`\SRcontinuedname` specifies the name that indicates that an output block is continued. The name is used when an output stream breaks across a page. The default is `continued`.

`\SRdpi` specifies the default dots per inch (DPI) for SAS to use in generating graphical output. The default is `300`.

`\SRgraphicdir` specifies the name of the directory that contains the SAS generated graphical output files. The default is `png`.

`\SRgraphtype` specifies the format of the SAS generated graphical output. You can specify either `png` or `pdf`. The default is `png`.

`\SRodsgraphopts` specifies a string that is passed as ODS GRAPHICS statement options. For a complete explanation of all available options, see the documentation of the ODS GRAPHICS statement in *SAS Output Delivery System: User's Guide*.

`\SRintertext` specifies the text to insert in `Datastep` environments that specify the `first=` option. The default is `... more data lines ...`

`\SRlinesize` specifies the default line size to use in generating tabular output and centering it for display. The default is `80`.

\SRlistingdir specifies the name of the directory that contains the SAS generated listing (tabular) output files. The default is `lst`.

\SRmacropath specifies the path to the location of the SAS macros that are bundled with the `StatRep` package. For example, if you installed the `statrep_macros.sas` file to a directory named `C:\mymacros`, then define macro `\SRmacropath` as follows:

```
\def\SRmacropath{c:/mymacros/statrep_macros.sas}
```

Use the forward slash in the definition as the directory name delimiter instead of the backslash, which is a special character in LaTeX. If you want to use a backslash character (\), you must insert it with the LaTeX command, `\@backslashchar`.

The default value is the current path. That is, the default definition for the `\SRmacropath` macro is the filename itself, `statrep_macros.sas`.

\SRmacroinclude specifies the line used in the generated SAS program to include the SAS macros that are bundled with the `StatRep` package. The default is `%include \SRmacropath /nosource;`

\SRpagesize specifies the default page size for SAS to use in generating tabular output. The default is `500`.

\SRparindent specifies the amount of space to indent `Datastep` and `Sascode` environments. The argument is a dimension. The default is `3em` and is measured according to the font currently in use.

\SRprogramline specifies the first lines to include in the generated SAS program after the `\SRmacroinclude` line.

The following default value calls a macro (from `statrep_macros.sas`) that removes the contents of the listing and graphic directories to ensure that the generated graphs and listings from the SAS program are current. The directories are created with each SAS run that includes the macros themselves (via `x` commands).

```
%hostdel;
```

\SRprogramname specifies the filename for the generated SAS program. The default is `\jobname_SR.sas`, where `\jobname` is usually the stem name of the LaTeX source file.

\SRstyle specifies the default ODS style for SAS to use to generate graphical output. The default is `Statistical`.

\SRtempfilename specifies the name of a temporary file that is used as a scratch file in the current working directory. The default is `sr.tmp`.

\SRverbfont specifies the font to use for code within `Datastep` and `Sascode` blocks. The default is `\ttfamily\bfseries` (**typewriter text, bold**).

5.2 The Program Preamble

The StatRep package automatically writes a preamble to the generated program and a preamble file.

The preamble settings are split into two parts to support users who prefer to manually write the calls to the StatRep macros and work interactively between the LaTeX source document and a SAS session. For this use, you can include the external preamble file once in your SAS session and all the necessary settings are made for you.

If you do not manually write calls to the StatRep macros (you use the default, automated method), there is nothing you need to do—your generated program contains the lines that specify your settings.

The preamble in the generated program includes the preamble file and deletes the contents of the output directories (`lst` and `png`, by default) so that obsolete files are not included in the document. Figure 7 shows an example of the preamble lines that are written to the generated program.

```
/*
 This file is auto-generated by the statrep package.
 Do not edit this file or your changes will be lost.
 Edit the LaTeX file instead.

 See the statrep package documentation and the file
 statrep.cfg for information on these settings.
 */


%include "report_SR_preamble.sas" /nosource;
/* Remove all output files. */
%hostdel;

/* Start program with a null title. */
title;
```

Figure 7: Generated SAS Program Preamble

The external preamble file sets defaults, includes the output-capture macros, and creates the output directories if they do not exist. You can customize the preamble; see section 5.1 for details. Figure 8 shows the default file preamble.

```
/*
 This file is auto-generated by the statrep package.
 Do not edit this file or your changes will be lost.
 Edit the LaTeX file instead.

 See the statrep package documentation and the file
 statrep.cfg for information on these settings.
 */


/* Set and invoke macro variable defaults. */
%let defaultlinesize=80;
%let defaultpagesize=500;
%let defaultdpi=300;
%let defaultstyle=statistical;
%let listingdir=lst;
%let graphicdir=png;
%let graphtype=png;
%let odsgraphopts=;

options nodate nonumber
ls=&defaultlinesize ps=&defaultpagesize
formchar='|----|+|---+=|-/\<>*';

/* Include SAS macro definitions. */
%include "statrep_macros.sas" /nosource;
```

Figure 8: Generated Preamble File

5.3 Two Methods of Writing

To maximize flexibility, the StatRep package provides two methods of writing code in your LaTeX document.

When you create your LaTeX document, you can use either the automatic method described in section 3 (in which the SAS macro calls are generated automatically) or a manual method (in which you write the `%output`, `%endoutput`, and `%write` macros yourself).

In the automatic method, each `Sascode` code block generates an `%output` macro call at the beginning of the block and an `%endoutput` macro call at the end of the block. Each `\Listing` and `\Graphic` tag generates the `%write` macro to replay the selected output objects to external files.

In the manual method, you decide where and when to make the macro calls. It is only in this respect that the method is manual: the StatRep package still generates your

SAS program and displays your code and results.

The StatRep package uses the SAS macro comment (`%* comment ;`) to provide line commands within a `Sascode` block. Furthermore, any line of code that begins with a null macro comment (`%*;`) in the first column is written to the SAS program and is not displayed.

You can use the manual method when you want to do one or more of the following:

- capture specialized or complicated output
- capture print output with SAS 9.2 (see the `%startlist` macro in section 4.2)
- capture output from the SAS log (see the `%startlog` macro in section 4.2)
- work interactively when writing (you can interactively develop or debug a certain section of your document by copying code from your LaTeX document and pasting it into a SAS session)

You can use either method, and you can mix the methods in a single document. The manual method is provided for cases in which the automatic method is too inflexible. By using the line commands in a `Sascode` environment, you are free to write your program as you want, while retaining control of the code that is displayed in your final PDF document.

Continuing with example shown in Figure 2 and Figure 3, you can write the code yourself within your LaTeX document as shown in Figure 9 and obtain the identical code display and capture.

Datastep is identical

```
\begin{Datastep}
proc format;
   value $sex 'F' = 'Female' 'M' = 'Male';
data one;
   set sashelp.class;
   format sex $sex.;
run;
\end{Datastep}
```

%output added manually

```
\begin{Sascode}
 *; %output(class)
proc reg;
    model weight = height age;
run;
```

%write macros added manually

```
 *; %write(rega, type=listing)
 *; %write(regb, type=graphic)
\end{Sascode}
```

Listing and Graphic tags only caption and insert output

```
\Listing[caption={Regression Analysis}]{rega}
\Graphic[caption={Graphs for Regression Analysis}]{regb}
```

Figure 9: Using the SAS Macros Manually

The `Datastep` environment in Figure 9 is identical to that shown in the Figure 2. However, the `Sascode` environment makes an explicit call to the `%output` macro to create the ODS document that contains all the results from the code block. Because this line begins with the null SAS macro comment (`%*;`), the line is passed directly to the generated SAS program and is not displayed.

The `%endoutput` macro is not necessary when you are processing only one ODS document. It is implicitly specified by the first `%write` macro.

Next are two explicit calls to the `%write` macro, which specify the ODS objects to capture and the ODS document that the objects should be taken from. Both `%write` macros use the minimum number of options. The first `%write` macro selects all notes and tables from the last ODS document created. The second `%write` macro selects all

graphs from the last ODS document created. Because the `store=` option is omitted in both cases, output from the most recently created ODS document is displayed.

Finally, the `\Listing` and `\Graphic` tags request the outputs. In this method, you do not place the options that are related to SAS in the the `\Listing` and `\Graphic` tags. You need to specify only the caption and filename prefix.

To summarize, the `Datastep` environment is handled identically in either method, the `Sascode` environment can optionally produce the `%output` and `%endoutput` macros, and the `\Listing` and `\Graphic` tags can optionally produce the `%write` macros.

5.4 ODS Object Selection

To select and display ODS objects, you specify options in the `\Listing` tag, `\Graphic` tag, or the `%write` macro. By default, when you omit object selection options, the `%write` macro selects all ODS objects, the `\Listing` tag selects all ODS tables and notes, and the `\Graphic` tag selects all ODS graphs.

Table 5 summarizes how you select ODS objects.

Table 5: ODS Object Selection Options

Option	Action
`options=table`	Select all tables and notes
`options=graph`	Select all graphs
`pattern=` *pattern*	Select all objects with a name matching a pattern. When an ODS object name has more than two levels, the middle level name is a group name. You can specify the `pattern=` option to select all ODS objects in the specified ODS group. More generally, you can specify any pattern to select all objects whose path contains the pattern.
`firstobj=`	Specifies the first object in the output stream to capture. The specified and subsequent objects are captured.
`lastobj=`	Specifies the last object in the output stream to capture. The first object in the stream to capture is the first object produced by the `Sascode` code block or the object specified in the `firstobj=` option.
`objects=` option	Specifies a space-separated list of objects to capture.

The `firstobj=` and `lastobj=` options can be modified with the option `options=skiplast` and `options=skipfirst`. For more information about how to use these options, see page 12.

Page Breaks

By default, a page break can occur between any two objects in the output stream. However, you can use left and right angle brackets, <>, to delineate a set of objects in which to suppress breaks. You use the symbols in the `objects=` option list in a `\Listing` tag, a `\Graphic` tag, or a `%write` macro.

For example, you can use the symbols to prevent a break between a "Parameter Estimates" table and the "Fit Statistics" table that follows it with the following option: `objects = < ParameterEstimates FitStatistics >`

After the < symbol, breaking is suppressed until the > symbol is encountered. After the > symbol, a break is introduced and normal breaking continues.

In summary:

- You can use the <> symbols in pairs to keep ODS objects together.
- You can use the > symbol (unpaired with a matching <) to create a break between tables.
- You can use the < symbol (unpaired with a matching >) to suppress all breaks.

A break is always allowed before and after a graph.

See page 12 for an alternate method of controlling breaks with the `options=nopage` and `options=onebox` options.

5.5 ODS Graphics

In SAS 9.3 and later, ODS Graphics is enabled by default in the SAS windowing environment. ODS Graphics is not enabled by default in batch mode and in the SAS windowing environment in SAS 9.2. When ODS Graphics is not enabled by default, you can enable ODS Graphics by specifying the following statement:

```
ods graphics on;
```

You can enable ODS Graphics in StatRep for all steps by providing this code block at the beginning of your LaTeX document:

```
\begin{Sascode}[program]
   ods graphics on;
\end{Sascode}
```

ODS Graphics and GRSEG Graphics

When you create a graph with ODS Graphics, the style and dots per inch (DPI) can be changed after the graph is created. The style and DPI are set when the graph is written to the external file. This enables you to specify the options in the `\Graphic` tag or in the `%write` macro.

On the other hand, when you create a GRSEG graph, the style and DPI are set when the graph is created. That is why you must specify the options in the `%output` macro. See section 4.2 for details about the `%output` macro options.

Table 6 summarizes the methods you can use to modify the style and DPI settings when you create a graph.

Table 6: Methods to Change Graph Properties

Method	ODS Graph	GRSEG Graph
`\Graphic` tag options	Yes	No
`%write` macro options	Yes	No
`%output` macro options	No	Yes
Reset global default	Yes	Yes

The `\Graphic` tag, the `%write` macro, and the `%output` macro have `style=` and `dpi=` options. For more information about these options, see section 4.1. Also, see section 4.2 for details about using the SAS macro variables to reset global defaults.

For GRSEG graphs, there are only two choices for DPI: 300 (the default) and 96. When the DPI is set to anything other than 300, then 96 is automatically used instead for GRSEG graphs.

6 Examples

6.1 Using the Datastep Environment

Figure 10 displays an example `Datastep` environment. The left margin for the environment is in the first column, which is where the data lines themselves begin. This ensures that the variables will be read correctly.

```
\begin{Datastep}[first=9, last=3]
title 'Probit Analysis, Newspaper Survey';
proc format;
   value subscrib 1 = 'accept' 0 = 'reject';
run;
data news;
   input sex $ 1-6 age 12-13 subs 18 ;
   datalines;
Female     35    0
Male       45    1
Female     51    0
Male       54    1
Female     35    0
Female     48    0
Male       46    1
Female     46    1
Male       38    1
Male       49    1
Male       50    1
Female     47    0
Female     39    0
Female     45    0
Male       39    1
Female     39    0
Female     52    1
Male       58    1
Female     32    0
Female     35    0
;
\end{Datastep}
```

Figure 10: `Datastep` Environment with Options

In Figure 10, the options to the `Datastep` environment specify that only a portion of the code block be displayed. All lines in the environment are written to the generated program.

The option `first=9` specifies that the displayed code block contain the TITLE, the PROC FORMAT code, and the DATA step block through the second line of data (the first nine input lines). After these lines, the following text is displayed:

```
...  more data lines ...
```

The option `last=3` specifies that the displayed code block will contain the last three lines of the environment.

Figure 11 shows the display resulting from the preceding `Datastep` environment.

```
   title 'Probit Analysis, Newspaper Survey';
   proc format;
      value subscrib 1 = 'accept' 0 = 'reject';
   run;
   data news;
      input sex $ 1-6 age 12-13 subs 18 ;
      datalines;
   Female     35    0
   Male       45    1

      ... more data lines ...

   Female     32    0
   Female     35    0
   ;
```

Figure 11: Displayed `Datastep` Environment with Options

6.2 Using the Sascode Environment

Figure 12 displays an example `Sascode` environment.

```
\begin{Sascode}[store=mdoc]
proc reg data=h38 plots=predictions(X=Year);
   model Population = Year Yearsq;
quit;
\end{Sascode}
```

Figure 12: Sascode Block

The code displayed in Figure 12 contains SAS code that performs a regression analysis. Because no line commands are given, the code block is written as-is to the generated SAS program, as shown in Figure 13.

```
%output(mdoc);
proc reg data=h38 plots=predictions(X=Year);
   model Population = Year Yearsq;
quit;
%endoutput(mdoc);
```

Figure 13: Generated Code from Sascode Block

6.3 Using the Sascode Environment with Line Commands

Figure 14 displays an example `Sascode` environment that contains line commands.

```
\begin{Sascode}[store=mdoc]
 * program  2;
libname mylib 'c:/mylibs';
filename in1 'h38.ssp';
 * display 2;
libname mylib 'path to your library directory';
filename in1 'path to data directory/h38.ssp';
proc reg data=mylib.h38 plots=predictions(X=Year);
   model Population = Year Yearsq;
quit;
\end{Sascode}
```

Figure 14: Sascode Block with Line Commands

The code displayed in Figure 14 contains two line commands that delineate two specifications for the `libname` and `filename` SAS statements. The line command `%* program 2;` specifies that the location-specific definitions be passed to the generated program, as shown in Figure 15.

```
 output(mdoc);
libname mylib 'c:/mylibs';
filename in1 'h38.ssp';
proc reg data=mylib.h38 plots=predictions(X=Year);
   model Population = Year Yearsq;
quit;
 endoutput(mdoc);
```

Figure 15: Generated Code from Sascode Block with Line Commands

The line command `%* display 2;` in Figure 14 specifies that the generic version of the `libname` and `filename` statements be displayed, as shown in Figure 16.

```
   libname mylib 'path to your library directory';
   filename in1 'path to data directory/h38.ssp';
   proc reg data=mylib.h38 plots=predictions(X=Year);
      model Population = Year Yearsq;
   quit;
```

Figure 16: Displayed Code from Sascode Block with Line Commands

6.4 Selecting ODS Objects by Default

When you use the `\Graphic` tag, all graph objects are automatically selected. When you use the `\Listing` tag, all non-graph objects such as tables and notes are automatically selected. When you use the `%write` macro, you can specify the `options=graph` option to select graphs or the `options=table` to select tables and notes.

The following statements select all of the tables for the `\Listing` display and all of the graphs for the `\Graphic` display:

```
\begin{Sascode}[store=docgs1]
ods graphics on;
proc corresp data=PhD short;
   var y1973-y1978;
   id Science;
run;
\end{Sascode}

\Listing[store=docgs1,
   caption={Inertia and Chi-Square Decomposition}]{crsi1a}

\Graphic[store=docgs1,
   caption={Correspondence Analysis of Ph.D. Data}]{crsi1b}
```

Figure 17: Object Selection with the `\Listing` and `\Graphic` Tags

The log information tables display the selected objects for each block of output.

The first information table corresponds to the ODS selection that is produced by the `\Listing` tag. All ODS objects of type 'Table' are selected (more precisely, all objects that are not of type 'Graph'). Each object is contained in its own selection group, so a page break might occur between any of the tables.

```
Objects                              Type  Status   Group

Corresp.Inertias                     Table Selected   1
Corresp.Rows.RowCoors                Table Selected   2
Corresp.Columns.ColCoors             Table Selected   3
Corresp.Configuration.ConfigPlot     Graph            .
```

Figure 18: SAS Log Information Table from the Listing Tag

The second information table corresponds to the ODS selection that is produced by the `\Graphic` tag. The single ODS object of type 'Graph' is selected.

```
Objects                              Type  Status   Group

Corresp.Inertias                     Table            .
Corresp.Rows.RowCoors                Table            .
Corresp.Columns.ColCoors             Table            .
Corresp.Configuration.ConfigPlot     Graph Selected   1
```

Figure 19: SAS Log Information Table from the Graphic Tag

6.5 Specifying and Capturing ODS Objects by Name

To capture particular ODS objects or ODS group output, you must specify the appropriate names in the `\Listing` tag, the `\Graphic` tag, or the `%write` macro. The options that support specific ODS names are the `pattern=`, `firstobj=`, `lastobj=`, or `objects=` options.

If an object appears more than once in a particular ODS document (which typically means in one `Sascode` block), you must specify additional name levels to differentiate the objects. The log information table displays the fully qualified ODS names; you use the information from the log to specify the appropriate name for the ODS objects to capture.

For example, if there are multiple residual panels, you must specify the additional level to select a particular ODS object.

```
objects=residualplot
```

```
objects=residualplot\#2
```

Note: When you have a pound character (#) in a pattern or object name, you must escape it in LaTeX tags. The pound character is a special LaTeX control character and

must be escaped with a backslash. In other words, specify `Group\#2` instead of `Group#2` in a LaTeX tag. Do not escape the # when you use the `%write` macro.

Comparisons are not case sensitive. For example, if you specify `pattern=fit`, the following objects will be selected if they occur in the output stream:

```
Fit.Population.ANOVA
MODEL1.Fit.Population.ANOVA
Reg.MODEL1.Fit.Population.ANOVA
reg#1.model1#1.fit#1.population#1.anova#1
reg#1.model1.fit.population#1.anova#1
reg.model1.fit.population.anova
```

Typically[1], you need only to specify the last level of an ODS name. For example, for one model and one ANOVA table, all of the following specifications for the ANOVA object are equivalent.

```
anova
ANOVA
ANOVA\#1
Fit.Population.ANOVA
MODEL1.Fit.Population.ANOVA
Population.ANOVA
Reg.MODEL1.Fit.Population.ANOVA
reg.model1.fit.population.anova
reg\#1.model1.fit.population\#1.anova\#1
reg\#1.model1\#1.fit\#1.population\#1.anova\#1
```

When you run the SAS program that is generated by the StatRep package, the SAS log contains a table with information about each ODS object. For example, Figure 20 shows a `Sascode` environment that is parsed and written to the generated SAS program when the LaTeX document is compiled.

[1] When deciding on names to specify, be sure to consult the table of names from the ODS document that appears in the SAS log. It contains the proper pattern of # characters. See Figure 21 for an example.

```
\begin{Sascode}[store=Ex31]
proc probit data=news;
   class subs sex;
   model subs=sex age / d=logistic itprint;
run;
\end{Sascode}

\Listing[store=Ex31,
  objects=ClassLevels IterHistory ModelInfo
          LastGrad LastHess ParameterEstimates,
  caption={Logistic Regression of Subscription Status}]{prb31a}
```

Figure 20: Example of Capturing Listing Output

The `\Listing` tag results in a call to the `%write` macro in the automatically generated SAS program. When you execute the program, the `%write` macro generates the log information table shown in Figure 21.

```
   Objects                       Type  Status   Group

   Probit.IterHistory            Table Selected   1
   Probit.ModelInfo              Table Selected   2
   Probit.NObs                   Table            .
   Probit.ClassLevels            Table Selected   3
   Probit.ParmInfo               Table            .
   Probit.ResponseProfile        Table            .
   Probit.Note                   Note             .
   Probit.LastGrad               Table Selected   4
   Probit.LastHess               Table Selected   5
   Probit.ConvergenceStatus      Table            .
   Probit.Type3Analysis          Table            .
   Probit.ParameterEstimates     Table Selected   6
```

Figure 21: SAS Log Information Table

The table of information displays the fully qualified name for each generated ODS object, its type, whether it is selected, and its selection group.

Page breaks can occur only between selection groups. You can control the grouping as described in the section 5.4 on page 25. For example, if you specify `LastGrad LastHess` as `<LastGrad LastHess>`, the two tables would be in the same group.

The order in which objects are created is determined by the order in which they are generated, not the order in which they are specified in the `objects=` option.

6.6 Capturing PRINT Output

Figure 22 shows simple use of the `%startlist` and `%endlist` macros.

```
\begin{Sascode}
 *; %startlist(myprtlabel);
   proc print data=sashelp.class(obs=10) noobs;
   run;
 *; %endlist;
\end{Sascode}

\Listing[caption={Mass Analysis}]{myprtlabel}
```

Figure 22: Capturing Print Output with the `%startlist` and `%endlist` Macros

The `%startlist` macro opens the ODS listing destination for writing. The PRINT procedure code is executed, and the `%endlist` macro closes the ODS destinations. The result is an output file called `myprtlabel.lst`, which is inserted into the document with the `\Listing` tag.

The following section describes how to use the `%startlist` and `%endlist` macros to capture and display part of a table.

6.7 Capturing Large Tables

The StatRep package automatically takes care of all page breaks in the output. Pages are allowed to break between groups of ODS objects or wherever there is a new page in the listing output.

In some cases, an ODS object is too large to fit on a page. There are two ways to handle such large tables:

- Set the page size to a smaller size so that a single ODS object is broken into pieces by using the SAS system option to control the size of the output parts. By default, a large page size is in effect (`defaultpagesize=500`) and each table appears as a complete output block. However, when you have tables that are too long to fit on a page (that is, single ODS objects that will not fit on a page), you must specify a smaller page size (for example, `pagesize=50`). The tables will automatically split using the normal SAS rules for splitting tables.
- Save the ODS object to a SAS data set and manipulate the data set so that it contains only part of the original output. Use PROC PRINT or DATA step statements to generate and capture the modified data set.

Figure 23 shows an example of using a SAS data set to manipulate the ODS object resulting from a problem that is very slow to converge. The output from this step includes a long iteration history table. The goal is to display only the first few and last few lines of the iteration history table.

```
\begin{Sascode}[store=doc]
proc prinqual data=X n=1 maxiter=2000
              plots=transformation out=results;
 *; ods output mtv=m convergencestatus=c;
   title 'Linearize the Scatter Plot';
   transform spline(X1-X3 / nknots=9);
run;
\end{Sascode}
```

Figure 23: Capturing Output to an ODS Output Data Set

Any arbitrary SAS statement can be sent to the generated program by preceding it with a null SAS macro comment (%*;). With the output generated and the ODS document `doc` created, the `Sascode` block shown in Figure 24 is passed directly to the generated program and not displayed.

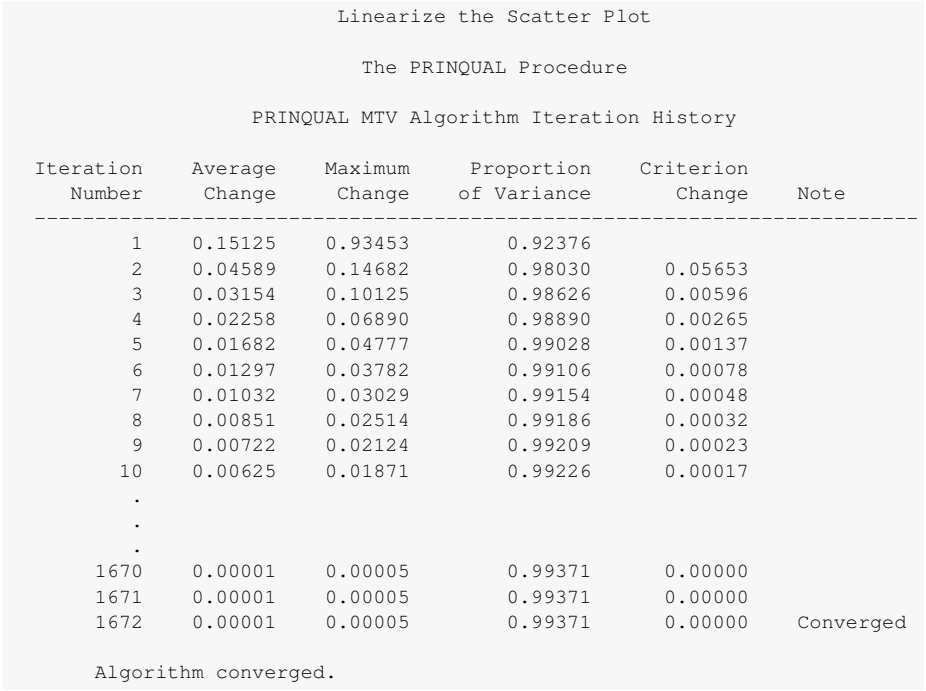
```
\begin{Sascode}[program]
%startlist(prqdb)
title3 'The PRINQUAL Procedure';
title5 'PRINQUAL MTV Algorithm Iteration History';
title6 '  ';

data _null_;
  set m(rename=(variance=rsquare));
  if _n_ le 13 or _n_ gt 1669;
  if 11 le _n_ le 13 then do;
     iternum   = .; avechange = ._; criterionchange = ._;
     maxchange = ._; rsquare = ._;
     end;
  file print ods=(template='Stat.Transreg.MTV');
  put _ods_;
run;
title;

data _null_;
   set c;
   file print;
   put @8 reason;
run;
%endlist;
\end{Sascode}
```

Figure 24: Manipulating the ODS Output Data Set for Printing

The `%startlist` macro opens the file `prqdb` to contain the printed results of the code block. The DATA step reads, manipulates, and prints the data set `m` that was just created by PROC PRINQUAL, and the printed content is written to the output file `prqdb.lst`, which is shown in Figure 25.



Linearize the Scatter Plot

The PRINQUAL Procedure

PRINQUAL MTV Algorithm Iteration History

Iteration Number	Average Change	Maximum Change	Proportion of Variance	Criterion Change	Note
1	0.15125	0.93453	0.92376		
2	0.04589	0.14682	0.98030	0.05653	
3	0.03154	0.10125	0.98626	0.00596	
4	0.02258	0.06890	0.98890	0.00265	
5	0.01682	0.04777	0.99028	0.00137	
6	0.01297	0.03782	0.99106	0.00078	
7	0.01032	0.03029	0.99154	0.00048	
8	0.00851	0.02514	0.99186	0.00032	
9	0.00722	0.02124	0.99209	0.00023	
10	0.00625	0.01871	0.99226	0.00017	
.					
.					
.					
1670	0.00001	0.00005	0.99371	0.00000	
1671	0.00001	0.00005	0.99371	0.00000	
1672	0.00001	0.00005	0.99371	0.00000	Converged

Algorithm converged.

Figure 25: Result of Printing ODS Output Data Set

Similarly, Figure 26 creates an output data set in the first step. The second step reads in the data set and uses PROC PRINT to display the first 10 rows of a table.

```
\begin{Sascode}
   title2 'Binary Table';

   *---Perform Multiple Correspondence Analysis---;
   proc corresp data=Cars binary;
 *;   ods output rowcoors=rc;
      ods select RowCoors;
      tables Origin Size Type Income Home Marital Sex;
   run;
\end{Sascode}
\begin{Sascode}[program]
%startlist(crse1n)
proc print data=rc(obs=10) noobs label;
  label label = '00'x;
  title4 'The CORRESP Procedure';
  title6 'Row Coordinates';
run;
%endlist;
\end{Sascode}
```

Figure 26: Capturing PROC PRINT Output from an ODS Data Set

6.8 Capturing Log Output

The SAS/IML example shown in Figure 27 illustrates the use of the macros that capture log output. In this example, the goal is to display SAS/IML error messages[2].

[2]The example code generates an error because of a variable scoping issue. Module `Mod7` is called from module `Mod8`. Therefore, the variables available to `Mod7` are those defined in the scope of `Mod8`. Because no variable named `x` is in the scope of `Mod8`, an error occurs on the PRINT statement in `Mod7`. An error would not occur if `Mod7` was called from the main scope, because `x` is defined at main scope.

```
\begin{Sascode}
 *; %startlog(psmodb)
proc iml;
   x = 123;

   start Mod7;
      print "In Mod7:" x;
   finish;

   start Mod8(p);
      print "In Mod8:" p;
      run Mod7;
   finish;
   run Mod8(x);

 *; %endlog;
\end{Sascode}
\Listing[caption={Error Message When a Variable is
                  Not Defined in a Module}]{psmodb}
```

Figure 27: Capturing Log Output

The captured log output is shown in Figure 28.

```
1599 +proc iml;
NOTE: IML Ready
1600 +   x = 123;
1601 +
1602 +   start Mod7;
1603 +      print "In Mod7:" x;
1604 +   finish;
NOTE: Module MOD7 defined.
1605 +
1606 +   start Mod8(p);
1607 +      print "In Mod8:" p;
1608 +      run Mod7;
1609 +   finish;
NOTE: Module MOD8 defined.
1610 +
1611 +   run Mod8(x);
ERROR: Matrix x has not been set to a value.

 statement : PRINT at line 1603 column 7
 traceback : module MOD7 at line 1603 column 7
             module MOD8 at line 1608 column 7

NOTE: Paused in module MOD7.
NOTE: Exiting IML.
NOTE: The SAS System stopped processing this step because of errors.
```

Figure 28: Complete Log Output Capture

The `%startlog` macro opens the specified file for writing, the IML procedure code is executed, and the `%endlog` macro closes the file.

To omit the PROC IML code from the captured log output, specify the `code=0` option in the `%endlog` macro as follows:

```
   %*; %endlog(code=0);
```

With the `code=0` option specified, the captured log output is displayed as shown in Figure 29.

```
NOTE: IML Ready
NOTE: Module MOD7 defined.
NOTE: Module MOD8 defined.
ERROR: Matrix x has not been set to a value.

 statement : PRINT at line 1651 column 7
 traceback : module MOD7 at line 1651 column 7
             module MOD8 at line 1656 column 7

NOTE: Paused in module MOD7.
NOTE: Exiting IML.
NOTE: The SAS System stopped processing this step because of errors.
```

Figure 29: Log Output Capture: Procedure Code Omitted

You can further refine the captured log output with the `range=` option in the `%endlog` macro as follows:

```
   %*; %endlog(code=0, range=_n_ > 6 and _n_ le 24);
```

With the both options specified, the captured log output is displayed as shown in Figure 30.

```
NOTE: Module MOD7 defined.
NOTE: Module MOD8 defined.
ERROR: Matrix x has not been set to a value.

 statement : PRINT at line 1722 column 7
 traceback : module MOD7 at line 1722 column 7
             module MOD8 at line 1727 column 7

NOTE: Paused in module MOD7.
```

Figure 30: Abbreviated Log Output Capture: Procedure Code Omitted

The `range=_n_ > 6 and _n_ le 24` option specifies that only the lines between 6 and 24 be included in the captured output. The value of `_n_` refers to the original line numbers before any filtering is done.

6.9 Capturing Output with Interactive Procedures

You can use the StatRep package to capture output from interactive procedures. Figure 31 shows how to capture output from the interactive IML procedure. It displays

three `Sascode` environments with `\Listing` outputs interleaved.

```
\begin{Sascode}
 *; %output(doc)
   proc iml;
      x = 1;
      print x;
\end{Sascode}

\Listing[caption={x = 1}]{x1a}

\begin{Sascode}
      x = 2;
      print x;
\end{Sascode}

\Listing[caption={x = 2}]{x2a}

\begin{Sascode}
      x = 3;
      print x;
 *; %write(x1a, objects=x)
 *; %write(x2a, objects=x#2)
 *; %write(x3a, objects=x#3)
\end{Sascode}

\Listing[caption={x = 3}]{x3a}
```

Figure 31: Capture Output with an Interactive Procedure

When you need to interleave `Sascode` blocks to make a single ODS document, as in Figure 31, you cannot rely on the StatRep package to automatically write the macros. You must write the output capture macros manually.

When the StatRep package automatically writes the macros, there is a one-to-one correspondence between a `Sascode` block and an ODS document. The interactive procedure terminates when the ODS document is closed.

Figure 31 uses three `Sascode` blocks to create a single ODS document. The contents of that ODS document are displayed in three separate listings. The ODS document is created by the `%output(doc)` statement in the first `Sascode` environment, and it is closed by the first `%write` macro in the third `Sascode` environment. The three `%write` macros create the output for the three output listings. The fact that two of the `\Listing` lines come before any of the output is created does not pose a problem.

6.10 Capturing and Displaying Numerical Results in Text

This example shows how you can capture SAS output values and display them in text. For example, suppose you want to capture the R-square values from a PROC REG step and a PROC GLM step and then display those values (but not display the steps) in your document.

Use the following steps to capture the values and display them in text:

1. In your document source file, anywhere before you want to display the captured values, include the following command to input a LaTeX file (named `myconstants.tex` in this example) to contain the LaTeX definitions of the captured values:

   ```
   \input{myconstants}
   ```

   The `myconstants.tex` file is actually generated in step 3, but you can include the preceding command before the file is generated because the SAS output capture program will create the file and write the definitions before pdfLaTeX generates the final PDF file.

2. Include one or more procedure steps, such as the following PROC REG and PROC GLM steps, which produce tables of fit statistics that are named `fsr` and `fsg`, respectively:

   ```
   \begin{Sascode}
   proc reg data=sashelp.class;
      model weight = height;
   %*;ods output fitstatistics=fsr;
   run; quit;

   proc glm data=sashelp.class;
      class sex;
      model weight = sex | height;
   %*;ods output fitstatistics=fsg;
   run; quit;
   \end{Sascode}
   ```

   Include a null macro comment at the beginning of the ODS OUTPUT statement in each step to cause the step to be run in the capture program but not be displayed in the text.

3. Near the end of your document source file, include code such as the following to generate the `myconstants.tex` file, extract the R-square values from the `fsr` and `fsg` tables of fit statistics that were generated in step 2, and write those values to `myconstants.tex`. The `Sascode` environment option, `program`, runs the code without displaying it in the text.

```
\begin{Sascode}[program]
data _null_;
   file 'myconstants.tex' termstr=nl;
   set fsr(keep=label2 nvalue2 where=(label2='R-Square'));
   put '\def \regrsq{' nvalue2 6.4 '}';
   set fsg(keep=rsquare);
   put '\def \glmrsq{' rsquare 6.4 '}';
run;
\end{Sascode}
```

4. Include the tags that are contained in the `myconstants.tex` file when you want to display the values of those tags, as follows:

   ```
   The R-square values are \regrsq\ and \qlmrsq, respectively.
   ```

   Include a backslash after the tag when you need to include a space after the value.

When you execute the generated SAS program, SAS writes the following tags in the `myconstants.tex` file:

```
\def \regrsq{0.7705}
\def \glmrsq{0.7930}
```

When pdfLaTeX processes the file, it replaces the `\regrsq` and `\glmrsq` tags with the values that were written to the `myconstants.tex` file and produces the following:

The R-square values are 0.7705 and 0.7930, respectively.

7 Appendix

7.1 Installation and Requirements

You can install the StatRep package by downloading `statrep.zip` from `support.sas.com/StatRepPackage`.

Table 7 shows the contents:

Table 7: Contents of the `statrep.zip` File

Filename	Description
`statrepmanual.pdf`	The *StatRep User's Guide* (this manual)
`statrep.ins`	The LaTeX package installer file
`statrep.dtx`	The LaTeX package itself
`statrep_macros.sas`	The StatRep SAS macros
`quickstart.tex`	A template and tutorial sample LaTeX file

Unzip the file `statrep.zip` to a temporary directory and perform the following steps:

Step 1: Install the StatRep SAS Macros

Copy the file `statrep_macros.sas` to a local directory. If you have a folder where you keep your personal set of macros, copy the file there. Otherwise, create a directory such as `C:\mymacros` and copy the file into that directory.

Step 2: Install the StatRep LaTeX Package

These instructions show how to install the StatRep package in your LaTeX distribution for your personal use.

a) **MikTeX users**: If you do not have a directory for your own packages, choose a directory name to contain your packages (for example, c:\localtexmf).

   In the following instructions, this directory is referred to as the "root directory".

   TeXLive users: If you maintain a system-wide LaTeX distribution and you want to make StatRep available to all users, see more detailed information about how to install LaTeX packages at: http://www.tex.ac.uk/cgi-bin/texfaq2html?label=what-TDS

   Determine the location that LaTeX uses to load packages. At a command-line prompt, enter the following command:
   `kpsewhich -var-value=TEXMFHOME`

   The command returns the root directory name in which LaTeX can find your personally installed packages.

   In the following instructions, this directory is referred to as the "root directory".

b) Create the directory if it does not exist, and create the additional subdirectories `tex/latex/statrep`. Your directory tree will have the following structure:

```
---------------
root directory/
   tex/
      latex/
        statrep/
```

c) Copy the files `statrep.dtx`, `statrep.ins`, `statrepmanual.pdf`, and `statrepmanual.tex` to the `statrep` subdirectory. Your directory tree will have the following structure:

```
---------------
root directory/
   tex/
      latex/
           statrep/
                 statrep.dtx
                 statrep.ins
                 statrepmanual.pdf
                 statrepmanual.tex
---------------
```

d) Change to the `statrep` directory and enter the following command:
`pdftex statrep.ins`

The command creates several files, one of which is the configuration file, `statrep.cfg`.

MikTEX users: Add the root directory name from Step 2a according to these instructions for installing packages for MikTeX (*Register a user-managed TEXMF directory*): `http://docs.miktex.org/manual/localadditions.html`

Step 3: Tell the StatRep Package the Location of the StatRep SAS Macros

Edit the `statrep.cfg` file that was generated in Step 2d so that the macro `\SRmacropath` contains the correct location of the macro file from step 1. For example, if you copied the `statrep_macros.sas` file to a directory named `C:\mymacros`, then you define macro `\SRmacropath` as follows:

```
\def\SRmacropath{c:/mymacros/statrep_macros.sas}
```

Use the forward slash as the directory name delimiter instead of the backslash, which is a special character in LaTeX.

You can now test and experiment with the package. Create a working directory, and copy the file `quickstart.tex` into it.

To generate the quick-start document:

1. Compile the document with pdfLATEX. You can use a LATEX-aware editor such as TEXworks, or use the command-line command `pdflatex`. This step generates the SAS program that is needed to produce the results.
2. Execute the SAS program quickstart_SR.sas, which was automatically created in the preceding step. This step generates the SAS results that are requested in the quick-start document.

3. Recompile the document with pdfLaTeX. This step compiles the quick-start document to PDF, this time including the SAS results that were generated in the preceding step.

   In some cases listing outputs may not be framed properly after this step. If your listing outputs are not framed properly, repeat this step so that LaTeX can remeasure the listing outputs.

You can make changes to the file with a LaTeX-aware editor or with any plain-text editor such as NotePad or emacs.

If you ever want to uninstall the StatRep package, delete the `statrep` directory that you created in the installation step 2d and remove the SAS macro file `statrep_macros.sas` that you copied in installation step 1. MikTeX users must additionally update the file-name database.
`(MikTeX Options dialog:  General-> Refresh FNDB)`

7.2 The `longfigure` Package

The `longfigure` package uses and relabels components of the well-known `longtable` package, written by David Carlisle, to provide a table-like environment that can display a stream of subfigures as a single figure that can break across pages.

The `longtable` package defines a `longtable` environment, which produces tables that can be broken by TeX's standard page-breaking algorithm. Similarly, the `longfigure` package defines a `longfigure` environment, which produces figures that can be broken by TeX's standard page-breaking algorithm. The internal structure of a long figure is similar to a long table. Rows might contain (for example) tables or graphics. Page breaks can occur only between rows.

The `longfigure` package differs from the `longtable` package in the following ways:

- The `longfigure` package supports two additional key-value options:
  - The `figname=` option specifies the counter for numbering `longfigure` environments. You can specify any string; the default is `figure`. When you specify a `figname=` value for which no counter already exists, the `longfigure` package loads the `tocloft` package and creates the counter.
  - The `resetby=` option specifies a counter (for example, `resetby=chapter`) such that output numbering is reset each time the counter value changes. If a counter is specified that does not exist, the `tocloft` package is loaded to create the new counter. For information about how the lists are typeset, see the `tocloft` package documentation.

- The counters and macros that start with `\LT` in the `longtable` package are renamed to start with `\LF` in the `longfigure` pacakge to avoid namespace conflicts when the two packages are used together. The generic macros that are defined in the `longtable` package (`\endfirsthead`, `\endhead`, `\endfoot`, and `\endlastfoot`) are also renamed with `\LF` as a prefix in the `longfigure` package.
- The `\LF@name` macro is based on the `\fnum@table` macro from the `longtable` package. The `\LF@name` macro returns the capitalized counter name and value. For example, if the counter is `figure` and the macro is processing the second `longfigure`, the `\LF@name` macro would contain the value "Figure 2."

You can use the `longfigure` package defaults to produce a *List of Figures* by inserting the following tag in your document at the point where you want the list to appear:

```
\listoffigures
```

The default counter used to display figures is the `figure` counter, but you can specify a different counter. For example, if you want your figures to be labeled as "Display," specify `figname=display` when you load the `longfigure` package; to display a *List of Displays*, insert the following command in your document at the point where you want the list to appear:

```
\listofdisplay
```

Note: If you specify a counter that does not exist, an auxiliary file with extension `.lft` is created to contain the information needed to create the list.

If you want to use more advanced features of the `tocloft` package, load it before you load the |longfigure| package so that the `longfigure` package sees that the counters specified by the `figname=` and `resetby=` options are already defined and does not attempt to create them.

Example

The following lines produce a single figure that contains three images and one tabular environment. Each element is a row of the `longfigure` environment. Page breaks can occur between rows.

```
\documentclass{book}
\usepackage{graphicx}
\usepackage{longfigure}

\begin{document}
```

```
  \begin{longfigure}{c}
    \caption{My Long Figure}\label{mlfig}\\
    \includegraphics[width=3in]{myfig1}\\
    \includegraphics[width=3in]{myfig2}\\
    \includegraphics[width=3in]{myfig3}\\
    \begin{tabular}{ll}
      one & two \\
      three & four\\
    \end{tabular}
  \end{longfigure}
\end{document}
```

In this example, the `{c}` argument in the `\begin{longfigure}` command specifies only a single centered column. You can also specify multiple columns and, if needed, use the `\multicolumn` command for more flexibility.