

# The SASnRdisplay package

Lars Madsen\*

December 1, 2017

## Introduction

The SASnRdisplay package acts as a frontend to the versatile listings package in order assist the user in typesetting SAS or R code or output. The package replaces the similar SASdisplay package, which was only available to my local users.

Please be aware that SASnRdisplay is *not* fully compatible with SASdisplay, the default settings are different and some macros are named differently.

## Acknowledgements

I'd like to thank Ulrike Fischer for hints about some very useful features in the caption package, Heiko Oberdiek for his help with hyperref details. Plus a thanks to Jørgen Granfeldt and Preben Blæsild for answering my various questions about details within the R or SAS languages.

## Loaded package

The following packages will be loaded (without any options): listings, xkeyval, xcolor, etoolbox, caption, needspace. If you need to pass options to these packages, load them *before* the SASnRdisplay package.

## Contents

<b>1</b>	<b>Interface</b>	<b>3</b>
1.1	Environments	3
1.2	Input from external files	3
1.3	Snippets	3
1.4	Package options	3
<b>2</b>	<b>Configuration</b>	<b>4</b>
2.1	Titles	4
2.2	Handling listings configuration of our environments and macros	5
2.3	Configuration FAQ/Examples	5
2.3.1	Font	5
2.3.2	Colors	6
2.3.3	Frames	6
2.3.4	Captions	7
2.3.5	Keywords	8
2.3.6	Escape to L <sup>A</sup> T <sub>E</sub> X	8
2.3.7	Input encodings (e.g. UTF8)	9
2.3.8	Other	9

---

\*Email: daleif@math.au.dk, v0.95 2017/12/01

<b>3</b>	<b>Examples</b>	<b>11</b>
3.1	SAS . . . . .	11
3.2	R . . . . .	12
<b>4</b>	<b>Default style settings for R and SAS</b>	<b>13</b>
4.1	Style settings for R . . . . .	13
4.1.1	R configuration styles . . . . .	13
4.1.2	R user styles . . . . .	14
4.1.3	R collector styles . . . . .	14
4.2	Style settings for SAS . . . . .	15
4.2.1	SAS configuration styles . . . . .	15
4.2.2	SAS user styles . . . . .	16
4.2.3	SAS collector styles . . . . .	17
4.2.4	Extra SAS keywords . . . . .	18
	<b>Bibliography</b>	<b>19</b>

# 1 Interface

## 1.1 Environments

```
\begin{env}[\langle options \rangle]  
  ...  
\end{env}
```

Each environment support an optional [*options*] argument. The *options* part should be listings related configuration.

### **SAScode, SAScode\***

For typesetting SAS code. The starred version is not automatically numbered.

### **SASoutput, SASoutput\***

Similar for SAS output. Note that it might be an idea to decrease the width of the SAS output from within the SAS programme.

### **Rcode, Rcode\***

Typesets R code

### **Routput, Routput\***

Typesets R output

## 1.2 Input from external files

General syntax:

```
\macro[\langle options \rangle]{\langle filename \rangle}
```

Available macros:

### **\inputSAScode, \inputSAScode\***

Similar to the SAScode(\*) environment, but get the data from external file.

### **\inputSASoutput, \inputSASoutput\***

Similar to the SASoutput(\*) environment.

### **\inputRcode, \inputRcode\***

Similar to the Rcode(\*) environment.

### **\inputRoutput, \inputRoutput\***

Similar to the Rcode(\*) environment.

## 1.3 Snippets

To typeset inline snippets we provide

```
\SASinline
```

```
\Rinline
```

They both behave like `\verb`, thus one can write

```
\Rinline[\langle options \rangle]|x <- 34|
```

Note that for `\SASinline`, key words are marked, i.e. it is SAS aware. This is not the case for R.

## 1.4 Package options

```
\usepackage[\langle options \rangle]{SASnRdisplay}
```

Available options:

**danish**

Loads Danish translations for some keywords. (Executed by default)

**english**

Similar for English.

**grayscale**

Changes some build in colors to monochrome.

**countbysection**

Force counters to be dominated by the section counter.

**countbychapter**

Force counters to be dominated by the chapter counter. This is the default if `\chapter` exist, otherwise `countbysection` will be used.

**consecutive**

Use this if you just want consecutive numbering throughout, that is the number of say SAS code, is not reset at every new chapter or section.

**countbylistings**

Here we will leave the counters alone and just use the one that comes with `listings`. This (of course) disables `countbysection`, `countbychapter` and `consecutive`.

**noautotitles-r, noautotitles-sas**

Do not automatically add a number to any of the code and outputs. It can still be added manually by using the `caption={{text}}` option.

**needspace=<length>**

This issue a `\needspace{<length>}` before each code or output environment or inclusion macro. It will ensure that if there is less than `<length>` space left on the page, a page break is issued before the construction start.

This feature is enabled by default with a default `<length>` of `3\baselineskip`.

**noneedspace**

This disables the `needspace` feature.

**sweave**

Overloads the Sweave package, i.e. it makes the `Sinput` and `Scode` environments behave as the `Rcode` environment, and the `Soutput` like the `Routput` environment.

If `Sinput` should have a slightly different look than `Rcode`, then use the `Sinput` style to add your extra configuration. (The Sweave package typesets the contents of the `Sinput` env to be in the `typewriter/monospace` font plus italic, whereas we just set it in `typewriter/monospace`.)

**sasweave**

Similar for the `sasweave` package, adding this options, we will overwrite `SASinput`, `SASoutput` and `SAScode` environments with our versions. Note you will have to load `SASnRdisplay` after the `SasWeave` package.<sup>1</sup>

Please note: This option has *not* been thoroughly tested. Please let me know if it works as advertised.

**<other>**

Other options will be passed on to the `listings` package.

## 2 Configuration

As a frontend to `listings`, the configuration is based upon `listings styles`, i.e. collections of `listings` configurations. These are applied in left to right fashion, the last configuration loaded takes precedence.

### 2.1 Titles

These macros holds the titles for the four types of displays. English:

---

<sup>1</sup>Because `SasWeave` use the same environment names as we do.

```
\renewcommand*\SnRRcodename{R-code}
\renewcommand*\SnRSAScodename{SAS-code}
\renewcommand*\SnRRoutputname{R-output}
\renewcommand*\SnRSASoutputname{SAS-output}
```

And for Danish:

```
\renewcommand*\SnRRcodename{R-kode}
\renewcommand*\SnRSAScodename{SAS-kode}
\renewcommand*\SnRRoutputname{R-udskrift}
\renewcommand*\SnRSASoutputname{SAS-udskrift}
```

Note if you are using `babel` and the `babel` option to `SASnRdisplay`, then these names are added to the language setup, and thus if you want to change then you will have to add your changes to the language setup as well.

## 2.2 Handling listings configuration of our environments and macros

The listings configuration we use in this package is based on the listings concept of *styles*. A style is basically just giving a collection of listings key-value sets a more convenient name. At first glance, this may seem a tedious method for configuration instead of giving various features macro names and letting the user change those macros. Been there, done that. Given the sheer number of listings options, this would make configuration very un-flexible.

The general listings syntax for styles is

```
\lstdefinestyle{<name>}{
  <key-value set>
}
```

One drawback is that if `<name>` already exist, then you will replace the contents of this style. Currently there is *no* manner in which to *add* to a style.

Thus it is a rather bad idea to provide the configuration as one long style, because then changing a small thing, would require the user to retype the rest of the configuration. Instead we split the configuration into smaller themed pieces. The user then have a choice of either overwriting one of these pieces or override a special user style which is executed as the last style (and thus overwrites any former style).

The settings can be seen in Section 4. The styles are broken into smaller pieces. In some cases it make sense to change one of these smaller pieces, in other cases it is easier to add stuff to the provided `<name>-<type>-user` styles.

## 2.3 Configuration FAQ/Examples

Here follows a list of FAQs as to how one would make some configuration changes. Again we remind the user, that it is not possible to add to a listings style. Thus you will have to add all your setting for, say, `r-user`, into one single call to `\lstdefinestyle`.

### 2.3.1 Font

#### *I'd like to change the font size*

The general fontsize (not comments) is handled by the keyword `basicstyle`, so you can change the font size for SAS code in two ways,

```
\lstdefinestyle{sas-code-user}{
  basicstyle=\ttfamily\footnotesize
}
```

or by overwriting the default SAS code font setting:

```
\lstdefinestyle{sas-code-fonts}{
  basicstyle = \ttfamily\small,
}
```

### 2.3.2 Colors

The default colors are SnRFrame »« for the frame, and SnRBG »« for the background. If the grayscale option is used, the mentioned colored are mapped on to SnRFrameGray »« and SnRBGGray »«.

#### *I'd like other colors*

We automatically load the xcolor package for colors, so we refer that package for details. If you just want to change, say, the background color, try (after SASnRdisplay)

```
\definecolor{SnRBG}{gray}{0.8}
```

for a gray tone (1 means white). If one loads the xcolor package *before* SASnRdisplay, then one can pass certain options to it and get access to a lot of color names, a survey of these can be found in the xcolor manual, [1].

The two default colores are defined as

```
\definecolor{SnRBG} {rgb}{0.94,0.97,1}
\definecolor{SnRFrame}{rgb}{0.79,0.88,1}
```

#### *I'd like different colors for code and output*

Since the  $\langle name \rangle$ - $\langle type \rangle$ -user styles are executed at the very end of the configuration, it will be suitable to add them there. Here is how to make the SAS code have a blue background while leaving the SAS output with the default.

```
\lstdefinestyle{sas-code-user}{
  backgroundcolor = \color{blue},
}
```

Note: Remember that there will only be one sas-code-user, thus if you have several configurations to add to it, collect them in one such \lstdefinestyle.

#### *How about the color of the text*

This can be seen in three different ways: regular text, comments and keyword. (In our case keywords only apply to SAS code.) This means that colors will have to be inserted into say the basicstyle key to change the basic font color. Here is instead how to make all SAS comments green, note that we have to copy the rest of the font settings for SAS comments, as we cannot add to a setting.

```
\lstdefinestyle{sas-code-user}{
  commentstyle = \normalfont\slshape\ttfamily\footnotesize\color{green},
}
```

When dealing with SAS keywords one can even add different colors to separate groups of keywords, though this is a bit out of our scope in this manual.

### 2.3.3 Frames

Listings supports a number of different types of frames, see the manual ([2]) for details.

*I like the settings from the old SASdisplay package with the line above and below*

Here we choose to simply overwrite a frame style.

```
\lstdefinestyle{r-frame}{  
  frame      = lines,  
  framesep   = 0.5em,  
  framerule  = 1mm,    % thickness of the rule  
}
```

*No rules*

```
\lstdefinestyle{r-frame}{}
```

A user may want to experiment with the keys `x(left|right)margin` and `framex(left|right|top|bottom)margin`.

### 2.3.4 Captions

This is not a configuration as such but rather a hint to how one adds a caption.

If the `noautotitles` is not activated, all non-starred environments and input macros will get an automatic caption, including a number. If one wish to add extra text use the following to the options of the environment or input macro.

```
caption={⟨My text⟩}
```

Remember the `{}` pair around the text.

In this version of `SASnRdisplay` ‘list of ...’ are not supported due to technical difficulties.

If you want to configure captions related to listings, please use

```
\captionsetup[lstlisting]{⟨options⟩}
```

For example, in this document we use

```
\captionsetup[lstlisting]{  
  font=small,  
  labelfont=bf  
}
```

to make the label text bold, and the entire caption text in `\small`.

Note that numbered construction without a caption are typeset as `Name Num`, with a caption this change into `Name Num: Caption`.

*I do not like numbers, but I'd like to add some titling info for some of my code.*

If you do not want to use the auto numbering scheme, then use the `noautotitles-sas` or `noautotitles-r` package options. Then to add just a title, add the following to the `⟨options⟩`

```
title={My title text}
```

It is similar to the caption option, but has no numbers or preceding text.

*How do I refer to code or output?*

First of all, as with floats, it is the caption that provide the number that one can refer to. So as long as the code or output is numbered, then one can just add

```
label={keyname}
```

to the environment or inclusion macro `⟨options⟩`.

You can of course also add a label even if it is not numbered, then `\ref{⟨key⟩}` will just not be weldefined. But `\pageref{⟨key⟩}` will!

### 2.3.5 Keywords

*In a presentaion, I'd like to highlight a word*

See the `emph` and `emphstyle` keys. Here is an example.

```
\begin{SAScode*}[emph={INSIGHT},emphstyle=\color{red}\bfseries]
PROC INSIGHT DATA data=fisk;
\end{SAScode*}
```

resulting in

```
PROC INSIGHT DATA data=fisk;
```

*How do I disable the keyword marking?*

You could either specify an empty language, i.e. `language=` to either the *options* or to a global style.

Or you could redefine the keyword style:

```
keywordstyle=
```

*I typeset SAS code, but keywords are not being marked!?*

This is usually because the mono space font (i.e. the font behind `\ttfamily`) does not support boldface (as that is the default manner which we mark keywords).

One such example is the default  $\LaTeX$  font: Computer Modern. Its mono space has no bold version.

Solutions: see <http://www.tug.dk/FontCatalogue/typewriterfonts.html>, you will need to look for fonts that are shown to support `\bfseries`.

In this manual we use `beramono`. Another interesting solution is to use

```
\renewcommand\ttdefault{txtt}
```

### 2.3.6 Escape to $\LaTeX$

This is a very handy feature and can e.g. be used to get formatted  $\LaTeX$  code inside, say, a comment. For example using

```
escapeinside=||,
```

means that if one write `||$a_{ij}$|` in a comment one would get  $a_{ij}$  typeset in the output.

A feature like this is *not* enabled by default. Though a user can always add it globally to the settings of his/her document, say using

```
\lstdefinestyle{sas-code-user}{
  escapeinside=||,
}
```

It does not have to be `»|«` that is the escape character.

It can be locally disabled by adding

```
escapeinside={},
```

in the *options* for the environment or the input macro.

The listings manual, [2, Section 4.14] list other features related to escaping back to normal  $\LaTeX$  formatting.

### 2.3.7 Input encodings (e.g. UTF8)

*I keep getting errors when I include my program, something about undefined chars!?*

The problem here is that `listings` cannot cope with so-called two two-byte characters, `listings` needs to do a lot of parsing which may break when dealing with e.g. two-byte UTF8 chars (essentially all non-ascii chars).

At the moment, no-one have made a UTF8 compatible version of `listings`, so we need to cope somehow.

We will be assuming you are working with UTF8 files (both on the LaTeX side, and any including source code), UTF8 is recommended nowadays, so we will ignore working purely in latin1 (which `listings` has no issues with at all).

First scenario: Assuming all your `listings` presented material comes from external sources, e.g. they are inputted, and assuming your UTF8 sources code is compatible with the latin1 encoding (aka Western European languages), then you can simply additionally use the package `listingsutf8`. It extends the file inclusion feature and extend the input encoding syntax. Add `listingsutf8` to your preamble, and using

```
\lstdefinestyle{r-include-code-user}{
  inputencoding=utf8/latin1
}
```

will attempt to auto convert the included code into latin1 before handing it off to `listings`. See [4] for a bit more details.

Another solution, which works very well is using the the `literate` option. The value of this key is read in triples, and is basically saying, if you see this char, typeset this instead. Here is an example with the Danish æøå:

```
\lstdefinestyle{r-code-user}{
  literate={æ}{\ae}1%
           {Æ}{\AE}1%
           {ø}{\o}1%
           {Ø}{\O}1%
           {å}{\aa}1%
           {Å}{\AA}1,%
}
```

(add the comma after the last one). There are several examples on <http://tex.stackexchange.com> if you search for “`listings literate`”.

As for the syntax of the triple:  $\{\langle input \rangle\}\{\langle typeset output \rangle\}\{\langle length \rangle\}$ .

### 2.3.8 Other

*I'd like to have line numbers*

Here is how to add line numbers to all R code.

```
\lstdefinestyle{r-code-user}{
  numbers = left,
  numberstyle = \tiny
}
```

Line numbers can be configured further, see section 4.8 in the `listings` manual, [2].

Line numbers can be very handy when displaying source code. For output, it might not be that relevant.

It is possible to actually label and refer to specific lines in a piece of code, see section 7 in the `listings` manual, [2].

*I have many blank lines, can some be ignored?*

Yes with the `emptylines` key. It determine the number of consecutive blank lines to allow in the output. By default listings will already ignore blank lines at the end of what ever is shown. To show only one blank line in the output for R, try

```
\lstdefinestyle{r-user}{
  emptylines=1,
}
```

If you are also using line numbers, you may want to use

```
\lstdefinestyle{r-user}{
  emptylines=*1,
}
```

then the line numbers 'jump' correctly in regards to the blank lines.

*Can also blank space at the start of lines be ignored?*

Of course, that key is called `gobble`, its value will indicate the number of characters to eat (from the left). Note that it will not distinguish between spaces and non-spaces, it will just eat a set number of characters at the start of each line.

*By the way, can one control the width of the SAS output from within a SAS programme?*

Yes, try the `»OPTIONS LS=80;«` setting.

*Can I show snippets of code?*

Sure. See the `firstline` and `lastline` keys or the `linerange`. They cannot be set globally, so can only be added into environment or input macro options.

There is also an experimental feature where instead of line numbers one specify certain strings inside the external file. This can be quite handy if the contents of the external file may change.<sup>2</sup> Section 5.7 in the listings manual ([2]) has more details.

*The quotes look odd in my code listings, can it look more like keyboard keys?*

Of course. Add the `textcomp` package, and issue

```
\lstdefinestyle{r-user}{
  upquote=true,
}
\lstdefinestyle{sas-user}{
  upquote=true,
}
```

*I use the Sweave package and overload the look using `SASnRdisplay`. I'd like `Sinput` to look more like the default in Sweave.*

This can be done by using the extra `Sinput` style to overload the basic style:

```
\lstdefinestyle{Sinput}{
  basicstyle = \ttfamily\itshape
}
```

---

<sup>2</sup>Then one does not have to manually change the line number references all the time.

### 3 Examples

#### 3.1 SAS

inline: `\SASinline|RANGE xxx|` results in **RANGE xxx**.

Personally I often add »...« around inline snippets to indicate where they start and end. Sadly this is apparently not something one can add into the inline macro definition because of its \verb-like nature.

**SAS code 3.1: Test of caption**

```
PROC INSIGHT DATA data=fisk;
SCATTER x1 x2 x3 x4 x5 * dosis vgt;
RUN;
OUTPUT
QUIT; /* a standard SAS comment */
```

was typeset via

```
\begin{SAScode}[caption={Test of caption}]
PROC INSIGHT DATA data=fisk;
SCATTER x1 x2 x3 x4 x5 * dosis vgt;
RUN;
OUTPUT
QUIT; /* a standard SAS comment */
\end{SAScode}
```

whereas

**SAS output 3.1**

TABLE OF NIVEAU BY SUBJECT					
NIVEAU	SUBJECT				
Frequency					
Percent					
Row Pct					
Col Pct	kem	mat	mus	samf	Total
h	0	1	1	0	2
	0.00	20.00	20.00	0.00	40.00
	0.00	50.00	50.00	0.00	
	0.00	50.00	100.00	0.00	
m	0	1	0	1	2
	0.00	20.00	0.00	20.00	40.00
	0.00	50.00	0.00	50.00	
	0.00	50.00	0.00	100.00	
o	1	0	0	0	1
	20.00	0.00	0.00	0.00	20.00
	100.00	0.00	0.00	0.00	
	100.00	0.00	0.00	0.00	
Total	1	2	1	1	5
	20.00	40.00	20.00	20.00	100.00

comes from

```
\begin{SASoutput}
TABLE OF NIVEAU BY SUBJECT

NIVEAU SUBJECT

FrequencyPercent
Row Pct Col Pct kem mat mus samf Total
```

```

-----+-----+-----+-----+-----+
h      0      1 1      0 2  0.00  20.00  20.00  0.00  40.00
        0.00  50.00  50.00  0.00   0.00  50.00  100.00  0.00
-----+-----+-----+-----+-----+
m      0      1 0      1 2  0.00  20.00  0.00  20.00  40.00
        0.00  50.00  0.00  50.00   0.00  50.00  0.00  100.00
-----+-----+-----+-----+-----+
o      1      0 0      0 1  20.00  0.00  0.00  0.00  20.00
        100.00  0.00  0.00  0.00  100.00  0.00  0.00  0.00
-----+-----+-----+-----+-----+
Total      1      2      1      1      5
          20.00  40.00  20.00  20.00  100.00
\end{SASoutput}

```

### 3.2 R

\Rinline|x <- a| result in x <-a

**R code 3.1:** With just one blank line showing, plus line numbers. Note we only show the first 25 lines.

```

1 #####
2 ### chunk number 1:
3 #####
4 maxlike <- function(x) {
5   b <- 0.5*(sqrt((sum(x)/length(x))^2 + 4*(sum(x^2)/length(x))) - sum(x)/length(x))
6   b
7 }
10
11 #####
12 ### chunk number 2:
13 #####
14 a <- rep(0,200)
15 for(n in 1:200) {
16   a[n] <- maxlike(rnorm(n,1,1))
17 }
20
21 #####
22 ### chunk number 3:
23 #####
24 plot(a, type="l",xlab="stikprøvestørrelse_n",
25 main="mle_indrammet_af_.025_og_.975_fraktiler_i_asymptotisk_ford.",

```

was typeset via

```

\inputRcode[
emptylines=*1,
numbers=left,
numberstyle=\tiny,
caption={With just one blank line showing, plus line numbers. Note we
only show the first 25 lines.},
linrange=1-25
]{opg_G17.R}

```

**R output 3.1:** Copied from an Sweave result.

```
[1] 0.0495
```

is just simple use of the Routput environment.

## 4 Default style settings for R and SAS

### 4.1 Style settings for R

Note how there are three user styles. The style `r-user` apply to both R code and R output, whereas `r-code-user` and `r-output-user` apply only to code and output respectively. This means that if, say, the user want to change the framing to a line above and below instead of the default box, the user can either overwrite the `r-frame` style, or the `r-user` style.

The style settings are divided into three separate groups: (a) the settings themselves, (b) user styles and (c) collector styles, which are just a common name and loading order for other styles and

#### 4.1.1 R configuration styles

Style: `r-vskips`

```
\lstdefinestyle{r-vskips}{
  aboveskip      = 10pt plus 3pt minus 5pt,
  belowskip      = 10pt plus 3pt minus 5pt,
  belowcaptionskip = 7pt,
  lineskip       = 0pt plus 0.1em, % help with blank lines and page stretch
}
```

Style: `r-fonts`

```
\lstdefinestyle{r-fonts}{
  basicstyle = \small\ttfamily,
}
```

The font is the base font for the rest. If for example one make use of `emphstyle=`, then one will get the `basicstyle` plus bold face (if possible) for things that are emphasized.

Style: `r-chars-and-breaks`

```
\lstdefinestyle{r-chars-and-breaks}{
  columns      = fixed, % chars vertically aligned
  breaklines,  % lines can be broken
  breakatwhitespace, % at white space
  extendedchars = true, % special chars allowed, be aware of utf8
}
```

Style: `r-markup`

```
\lstdefinestyle{r-markup}{ % this only make sense for code
  language      = R, % R lang added 2017/12/01
  commentstyle = \normalfont\slshape\ttfamily\footnotesize,
}
```

R lang setting was added 2017/12/01 and requires a listings newer from 2015 or later.

Style: `r-frame`

```
\lstdefinestyle{r-frame}{
  frame      = single, % single frame all the way round, box broken at page break
  framesep = 0.5em, % sep from frame to text
}
```

Style: r-colors

```
\lstdefinestyle{r-colors}{  
  backgroundcolor = \color{SnRBG},  
  rulecolor      = \color{SnRFrame},  
}
```

The colors being used as standard have special names. By default they are in color. If the monochrome option is issued they are mapped onto SnRBGGray and SnRFrameGray, the later being black by default.

Style: r-inline

```
\lstdefinestyle{r-inline}{  
  basicstyle = \ttfamily,  
  language = R, % R lang added 2017/12/01  
}
```

Note that by default, the inline style for \Rinline, shares nothing with the rest of the R configuration and is loaded on its own.

R lang setting was added 2017/12/01 and requires a listings newer from 2015 or later.

#### 4.1.2 R user styles

These are all empty by default. There are two types of these: (i) styles applying to both types (code and output) in one go, and (ii) ones that are specific to either code or output. We also have a third level *file include* versus *in document*. The more specific the name, the later it will come in the collector styles (i.e. its settings will apply last).

Style: r-user – user stuff for everything R

```
\lstdefinestyle{r-user}{}
```

Style: r-code-user – only for R code

```
\lstdefinestyle{r-code-user}{}
```

Style: r-output-user – only for R output

```
\lstdefinestyle{r-output-user}{}
```

These three apply only to inclusion macros. Can be handy to specify an input encoding for, say, all included code files.

Style: r-include-user

```
\lstdefinestyle{r-include-user}{}
```

Style: r-include-code-user

```
\lstdefinestyle{r-include-code-user}{}
```

Style: r-include-output-user

```
\lstdefinestyle{r-include-output-user}{}
```

#### 4.1.3 R collector styles

Please note the calling sequence.

Style: r-style – all common code for R

```
\lstdefinestyle{r-style}{  
  style = r-vskips,           % vertical spacing  
  style = r-fonts,           % fonts  
  style = r-colors,           % colors  
  style = r-chars-and-breaks, % special chars and line breaks  
  style = r-frame,           % framing  
  style = r-user,            % user defined configuration  
}
```

Style: r-code – specific for R code

```
\lstdefinestyle{r-code}{  
  style = r-style,  
  style = r-markup,           % markup only make sense for code  
  style = r-code-user,  
}
```

Style: r-output – specific for R output

```
\lstdefinestyle{r-output}{  
  style = r-style,  
  style = r-output-user,  
}
```

Next we have the extra styles used for inclusion macros. They are the same as for R code and R output, with the addition of two extra styles.

Style: r-include-code

```
\lstdefinestyle{r-include-code}{  
  style = r-code,  
  style = r-include-user,  
  style = r-include-code-user,  
}
```

Style: r-include-output

```
\lstdefinestyle{r-include-output}{  
  style = r-output,  
  style = r-include-user,  
  style = r-include-output-user,  
}
```

## 4.2 Style settings for SAS

The structure is similar to the one used for R, though the font settings are split in two.

### 4.2.1 SAS configuration styles

Style: sas-inline

```
\lstdefinestyle{sas-inline}{  
  basicstyle = \ttfamily,  
  style      = sas-more-keywords,  
  language   = SAS,  
}
```

#### Style: sas-vskips

```
\lstdefinestyle{sas-vskips}{
  aboveskip      = 10pt plus 3pt minus 5pt,
  belowskip      = 10pt plus 3pt minus 5pt,
  belowcaptionskip = 7pt,
  lineskip       = 0pt plus 0.1em,    % help with blank lines and page stretch
}
```

#### Style: sas-colors

```
\lstdefinestyle{sas-colors}{
  backgroundcolor = \color{SnRGB},
  rulecolor       = \color{SnRFrame},
}
```

#### Style: sas-code-fonts

```
\lstdefinestyle{sas-code-fonts}{
  basicstyle = \ttfamily\small,
}
```

#### Style: sas-output-fonts

```
\lstdefinestyle{sas-output-fonts}{
  basicstyle = \ttfamily\footnotesize,
}
```

Note how we actually split the font settings in two. This is because it is common to have SAS code and SAS output in different font sizes. As R output is not that common, this split has not been made with R, it will be up to the user.

#### Style: sas-chars-and-breaks

```
\lstdefinestyle{sas-chars-and-breaks}{
  columns      = fixed,    % chars vertically aligned
  breaklines,  % lines can be broken
  breakatwhitespace, % at white space
  extendedchars = true,    % special chars allowed, be aware of utf8
}
```

#### Style: sas-markup

```
\lstdefinestyle{sas-markup}{
  language      = SAS,
  keywordstyle  = \bfseries,
  comment       = [s]{/*}{*/},
  commentstyle  = \slshape\footnotesize,
}
```

Note that only the `/*...*/` style SAS comments are supported for styling. The `*...;` syntax is *not* supported. Also note we had to change the SAS settings for listings otherwise we would be unable to style SAS comments.

#### Style: sas-frame

```
\lstdefinestyle{sas-frame}{
  frame      = single,
  framesep  = 0.5em,
}
```

## 4.2.2 SAS user styles

Style: sas-user – user stuff for everything SAS

```
\lstdefinestyle{sas-user}{}
```

Style: sas-code-user – only for SAS code

```
\lstdefinestyle{sas-code-user}{}
```

Style: sas-output-user – only for SAS output

```
\lstdefinestyle{sas-output-user}{}
```

These three apply only to inclusion macros. Can be handy to specify an input encoding for, say, all included code files.

Style: sas-include-user

```
\lstdefinestyle{sas-include-user}{}
```

Style: sas-include-code-user

```
\lstdefinestyle{sas-include-code-user}{}
```

Style: sas-include-output-user

```
\lstdefinestyle{sas-include-output-user}{}
```

### 4.2.3 SAS collector styles

Style: sas-style – all common code for SAS

```
\lstdefinestyle{sas-style}{  
  style = sas-vskips,  
  style = sas-frame,  
  style = sas-colors,  
  style = sas-chars-and-breaks,  
  style = sas-user,  
}
```

Style: sas-code – specific for SAS code

```
\lstdefinestyle{sas-code}{  
  style = sas-style,  
  style = sas-code-fonts,  
  style = sas-markup, % there is no markup of the output  
  style = sas-more-keywords, % has to come after markup when loading styles  
  style = sas-code-user,  
}
```

Style: sas-output – specific for SAS output

```
\lstdefinestyle{sas-output}{  
  style = sas-style,  
  style = sas-output-fonts,  
  style = sas-output-user,  
}
```

Style: sas-include-code

```
\lstdefinestyle{sas-include-code}{  
  style = sas-code,  
  style = sas-include-user,  
  style = sas-include-code-user,  
}
```

Style: sas-include-output

```
\lstdefinestyle{sas-include-output}{  
  style = sas-output,  
  style = sas-include-user,  
  style = sas-include-output-user,  
}
```

#### 4.2.4 Extra SAS keywords

Jørgen Granfeldt supplied extra SAS keywords to supplement those supported by listings. The keywords are found in `SASnRdisplay.cfg` and is labelled as the `sas-more-keywords` style. Note that even though not required by SAS, all supported keywords are written in upper case. JG explains that this is encouraged because that it makes it easier to tell the difference between build in SAS commands and user supplied (lower case) variables and procedure names.

Please note that we also change a list of *other* keywords, otherwise we will be unable to style SAS comments.

Here is the current list.

Style: sas-more-keywords, from `SASnRdisplay.cfg`

```
\lstdefinestyle{sas-more-keywords}{%  
morekeywords={SASAUTOS,LABEL},  
morekeywords={PROC,INSIGHT,SCATTER,QUIT,FORMAT,VALUE},  
morekeywords={DISCRIM,WCOV,WSSCP,METHOD,POOL},  
morekeywords={DATALINES,WITH,OPTIONS,GPLOT,LS,PS},  
morekeywords={SYSLIN,INSTRUMENTS,ENDOGENOUS,EXOGENOUS,IDENTITY,%  
WEIGHT,OLS,2SLS,LIML,SUR,ITSUR,3SLS,IT3SLS,FIML,MELO},  
morekeywords={MODEL,OUT,STDR,STDP,H,R,STUDENT,RSTUDENT,PRESS,%  
UCL,LCL,UCLM,LCLM,CL},  
morekeywords={FREQ,TABLES},  
morekeywords={GLM,CLASS,LSMEANS,MANOVA,MTEST,REG,PRINTE,%  
FILENAME,GOPTIONS,DEV,CTEXT,GACCESS,NOPRINT,CONTRAST,ESTIMATE,RANDOM},  
morekeywords={SS1,SS2,SS3,SSD,SS4,CLI,CLM,CLPARM},  
morekeywords={NOUNI,OUTPUT},  
morekeywords={E,E1,E2,E3,SOLUTION,TEST},%  
morekeywords={IML,USE,READ,ALL,INTO,PRINT,COLNAME,ROWNAME,CREATE,%  
FROM,APPEND},  
morekeywords={MIXED,DDFM,REPEATED,PARMS,PRIOR,ALPHA,TYPE},  
morekeywords={GREPLAY,NOFS,NOBYLINE,IGOUT,TC,TEMPLATE,TREPLAY,GOUT},  
morekeywords={GSFMODE,TARGETDEVICE,ROTATE,CBACK,GUNIT,HTITLE,HTEXT,%  
FTEXT,CSYMBOL,ANNOTATE},  
morekeywords={SYMBOL,SYMBOL1,SYMBOL2,SYMBOL3,SYMBOL4,SYMBOL5,SYMBOL6,%  
SYMBOL7,SYMBOL8},  
morekeywords={LEGEND1,LEGEND2,LEGEND3,ANGLE},  
morekeywords={INTERPOL,I},  
morekeywords={AXIS,AXIS1,AXIS2,AXIS3,AXIS4,AXIS5,AXIS6,AXIS7,AXIS8,%  
HAXIS,VAXIS,ORDER},  
morekeywords={MINOR,WIDTH,COLOR,GPLOT,PLOT,OVERLAY},  
morekeywords={I,V,L,H,C,ANGLE,NOLEGEND,USS,OF},  
morekeywords={TITLE,TITLE1,TITLE2,TITLE3,TITLE4,TITLE5,TITLE6},  
morekeywords={PRINCOMP,COV},  
morekeywords={GSFNAME,GSASFILE,INCLUDE},  
morekeywords={GENMOD,LINK,FWDLINK,INVLINK,ASSESS,ASSESSMENT,OBSTATS,%  
SCALE,DSCALE,PSCALE},  
morekeywords={TYPE1,TYPE3,WALD,WALDCI,XVARS},  
morekeywords={DIST,TOTAL,NOINT,OFFSET},  
morekeywords={ODS,LISTING,ParameterEstimates,RESDEV,STDRESDEV,%  
PREDICTED,RESCHI,RESLIK,STDRESCHI},  
morekeywords={XBETA,STDXBETA,LOWER,UPPER,HESSWGT},  
morekeywords={FWDLINK,INVLINK,VARIANCE,DEVIANCE},  
morekeywords=[7]{P},  
keywordstyle=[7]{\normalfont\ttfamily},
```

```
% Listings setup for SAS include / and * in the keyword list,  
% meaning we cannot style comments in SAS, we therefore remove  
% remove them from the keyword list  
otherkeywords={!, !=, ~, $, \&, _ , <, >=, = <, >},  
}
```

## Bibliography

- [1] Uwe Kern, *Extending L<sup>A</sup>T<sub>E</sub>X's color facilities: the xcolor package*, 2016. CTAN: [/macros/latex/contrib/xcolor/](#).
- [2] Various, *The listings Package*, 1996–2015. CTAN: [/macros/latex/contrib/listings/](#).
- [3] Philipp Lehman, Joseph Wright, *The etoolbox Package – An e-TeX Toolbox for Class and Package Authors*, 2017. CTAN: [/macros/latex/contrib/listings/etoolbox](#).
- [4] Heiko Oberdiek, *The listingsutf8 package*, 2016. CTAN: [/macros/latex/contrib/oberdiek/](#).
- [5] Axel Sommerfelt, *Customizing captions of floating environments using the caption package*, 2011. CTAN: [/macros/latex/contrib/caption](#).