How to work with SweaveListingUtils

Peter Ruckdeschel*

Fraunhofer ITWM
Fraunhofer Platz 1
67663 Kaiserslautern
Germany

e-Mail: Peter.Ruckdeschel@itwm.fraunhofer.de

Version control information:

Head URL:	svn+ssh://ruckdeschel@svn.r-forge.r-project.org/svnroot/distr/pkg/SweaveListingUtils/vignettes/ExampleSweaveListingUtils.Rnw
Last changed date:	2011-11-18 13:52:03 +0100 (Fr, 18 Nov 2011)
Last changes revision:	770
Version:	Revision 770
Last changed by:	Peter Ruckdeschel (ruckdeschel)

November 18, 2011

Abstract

In this vignette, we give short examples how to use package "SweaveListingUtils".

1 What package SweaveListingUtils is for

Package "SweaveListingUtils" provides utilities for combining Sweave, confer Leisch (2002a,b, 2003), with functionality of TEX-package-listings, confer Heinz and Moses (2007). In particular, we define R / Rd as TEX-package-listings "language" and functionality to include R / Rd source file (sniplets) copied from an url, by default from the svn server at R-forge, confer R-Forge Administration and Development Team (2008) in its most recent version, thereby avoiding inconsistencies between vignette and documented source code. In this respect it supports (and to some extent enhances) Sweave.

2 Preparations: Preamble

You should include into the preamble of your .Rnw file something like

*Fraunhofer ITWM, Kaiserslautern

```
% ------------------------------------------------------------------------
\RequirePackage{fancyvrb}
\RequirePackage{listings}
%%% important: keep the following comment in
%%% (see https://mailman.stat.ethz.ch/pipermail/r-help/2009-July/204747.html)
% this comment persuades Sweave not to insert \usepackage{Sweave}
%
% ------------------------------------------------------------------------
\SweaveOpts{keep.source=TRUE}
% ------------------------------------------------------------------------
<<SweaveListingsPreparations, results=tex, echo=FALSE, strip.white=FALSE>>=
require(SweaveListingUtils)
SweaveListingPreparations()
## setToBeDefinedPkgs(....) line just necessary for the example;
##     may be skipped in general
##     you may also wish to pass arguments to SweaveListingPreparations()
##     see ?SweaveListingPreparations
setToBeDefinedPkgs(pkgs = c("SweaveListingUtils","distr"),
                   keywordstyles = c("\\bf\\color{blue}","\\bf\\color{red}"))
@
```

Actually, after Sweave-ing the `.Rnw` file to a corresponding `.tex` file, will expand to a rather long form (depending on which packages you have attached), but you should not worry about your document getting very long, as the inserted TEXcommands (or more precisely `listings`-package commands) only declare the list(s) of registered keywords (for later markup).
In our case this should expand to something like

```
\RequirePackage{fancyvrb}
\RequirePackage{listings}
%\usepackage{Sweave}
% ------------------------------------------------------------------------
% ------------------------------------------------------------------------
%-------------------------------------------------------------------------%
%Preparations for Sweave and Listings
%-------------------------------------------------------------------------%
%
\RequirePackage{color}
\definecolor{Rcolor}{rgb}{0, 0.5, 0.5}
\definecolor{RRecomdcolor}{rgb}{0, 0.6, 0.4}
\definecolor{Rbcolor}{rgb}{0, 0.6, 0.6}
\definecolor{Routcolor}{rgb}{0.461, 0.039, 0.102}
\definecolor{Rcommentcolor}{rgb}{0.101, 0.043, 0.432}
%-------------------------------------------------------------------------%
\lstdefinelanguage{Rd}[common]{TeX}%
{moretexcs={acronym,alias,arguments,author,bold,cite,%
            code,command,concept,cr,deqn,describe,%
            description,details,dfn,doctype,dots,%
            dontrun,dontshow,donttest,dQuote,%
            email,emph,enc,encoding,enumerate,env,eqn,%
            examples,file,format,item,itemize,kbd,keyword,%
            ldots,link,linkS4class,method,name,note,%
            option,pkg,preformatted,R,Rdopts,Rdversion,%
            references,S3method,S4method,Sexpr,samp,section,%
            seealso,source,sp,special,%
            sQuote,strong,synopsis,tab,tabular,testonly,%
```

```
          title,url,usage,value,var,verb},
   sensitive=true,%
   morecomment=[l]\%% 2008/9 Peter Ruckdeschel
}[keywords,comments]%%
%-------------------------------------------------------------------------------%
\lstdefinestyle{RstyleO1}{fancyvrb=true,escapechargğ=,language=R,%
       % ğ just for convenience replacing the accent grave
       % (which would escape the rest here, which we do not want for
       % printing out the original code)
                           basicstyle={\color{Rcolor}\small},%
                           keywordstyle={\bf\color{Rcolor}},%
                           commentstyle={\color{Rcommentcolor}\ttfamily\itshape},%
                           literate={<-}{{$\leftarrow$}}2{<<-}{{$\twoheadleftarrow$}}2%
                                    {~}{{$\sim$}}1{<=}{{$\leq$}}2{>=}{{$\geq$}}%
                                    2{^}{{$\scriptstyle\wedge$}}1,%
                           alsoother={$},%
                           alsoletter={.<-},%
                           otherkeywords={!,!=,~,$,*,\&,\%/\%,\%*\%,\%\%,<-,<<-,/},%
                           escapeinside={(*}{*)}}%
% note: we define styles RstyleO<num> incrementally, i.e.
%       \lstdefinestyle{RstyleO<num>}{style = RstyleO<num-1>,
%                                     <further definitions for RstyleO<num> > }
% and then overwrite each time style Rstyle by
%     \lstdefinestyle{Rstyle}{style=RstyleO<num>}
\lstdefinestyle{Rstyle}{style=RstyleO1}
\lstdefinestyle{Rdstyle}{fancyvrb=true,language=Rd,keywordstyle={\bf},%
                          basicstyle={\color{black}\footnotesize},%
                          commentstyle={\ttfamily\itshape},%
                          alsolanguage=R}%
%-------------------------------------------------------------------------------%
\global\def\Rlstset{\lstset{style=Rstyle}}%
\global\def\Rdlstset{\lstset{style=Rdstyle}}%
%-------------------------------------------------------------------------------%
\global\def\Rinlstset{\lstset{style=Rinstyle}}%
\global\def\Routlstset{\lstset{style=Routstyle}}%
\global\def\Rcodelstset{\lstset{style=Rcodestyle}}%
%-------------------------------------------------------------------------------%
\Rlstset
%-------------------------------------------------------------------------------%
%copying relevant parts of Sweave.sty
%-------------------------------------------------------------------------------%
%
\RequirePackage{graphicx,fancyvrb}%
\IfFileExists{upquote.sty}{\RequirePackage{upquote}}{}%

\RequirePackage{ifthen}%
\newboolean{Sweave@gin}%
\setboolean{Sweave@gin}{true}%
\setkeys{Gin}{width=0.8\textwidth}%
\newboolean{Sweave@ae}
\setboolean{Sweave@ae}{true}%
\RequirePackage[T1]{fontenc}
\RequirePackage{ae}
%
\newenvironment{Schunk}{}{}

\newcommand{\Sconcordance}[1]{%
\ifx\pdfoutput\undefined%
```

```
\csname newcount\endcsname\pdfoutput\fi%
\ifcase\pdfoutput\special{#1}%
\else\immediate\pdfobj{#1}\fi}

%-------------------------------------------------------------------------------%
% ---- end of parts of Sweave.sty
%-------------------------------------------------------------------------------%
%
\lstdefinestyle{RinstyleO}{style=Rstyle,fancyvrb=true,%
                           basicstyle=\color{Rcolor}\small}%
\lstdefinestyle{Rinstyle}{style=Rstyle,fancyvrb=true,%
                          basicstyle=\color{Rcolor}\small}%
\lstnewenvironment{Sinput}{\Rinlstset}{\Rlstset}
\lstdefinestyle{RoutstyleO}{fancyvrb=false,basicstyle=\color{Routcolor}\small}%
\lstdefinestyle{Routstyle}{fancyvrb=false,basicstyle=\color{Routcolor}\small}%
\lstnewenvironment{Soutput}{\Routlstset}{\Rlstset}
\lstdefinestyle{RcodestyleO}{style=Rstyle,fancyvrb=true,fontshape=sl,%
                             basicstyle=\color{Rcolor}}%
\lstdefinestyle{Rcodestyle}{style=Rstyle,fancyvrb=true,fontshape=sl,%
                            basicstyle=\color{Rcolor}}%
\lstnewenvironment{Scode}{\Rcodelstset}{\Rlstset}
%-------------------------------------------------------------------------------%
\let\code\lstinline
\def\Code#1{{\tt\color{Rcolor} #1}}
\def\file#1{{\tt #1}}
\def\pkg#1{{\tt "#1"}}
\newcommand{\pkgversion}{{\tt 2.2}}
\lstdefinestyle{RstyleO2}{style=RstyleO1,%
% --------------------------
% Registration of package SweaveListingUtils
% --------------------------
morekeywords={[2]taglist,SweaveListingPreparations,SweaveListingOptions,%
SweaveListingoptions,SweaveListingMASK,setToBeDefinedPkgs,setBaseOrRecommended,%
readSourceFromRForge,readPkgVersion,lstsetRout,lstsetRin,lstsetRd,lstsetRcode,%
lstsetR,lstsetLanguage,lstset,lstinputSourceFromRForge,isBaseOrRecommended,%
getSweaveListingOption,copySourceFromRForge,changeKeywordstyles%
},%
keywordstyle={[2]{\bf}},%
%
% --------------------------
% Registration of package startupmsg
% --------------------------
morekeywords={[3]suppressStartupMessages,startupType,startupPackage,%
StartupMessage,startupMessage,startupEndline,readVersionInformation,%
readURLInformation,pointertoNEWS,onlytypeStartupMessages,%
NEWS,mystartupMessage,mySMHandler,infoShow,buildStartupMessage%
},%
keywordstyle={[3]{\bf}},%
%
% --------------------------
%
...
snipped expanded (TeX) (see how the original ((`\TeX`)) gets escaped!)
code for registration of packages
        tools, stats, graphics, grDevices, utils, datasets, methods, base
...
%
```

```
},%
keywordstyle={[11]{\bf\color{RRecomdcolor}}}%
%
}%
\lstdefinestyle{Rstyle}{style=RstyleO2}
%
%-------------------------------------------------------------------------------%
```

Finally to clean up things (in particular unmask the base functions **require** and **library** again) at the end of your document you should append something like

```
<<cleanup, echo=FALSE>>=
unloadNamespace("SweaveListingUtils")
@
```

Remark: As suggested by Andrew Ellis, ETH Zürich, SweaveListingPreparations from version `0.3` has two more options: First, by setting argument withOwnFileSection (default FALSE), one can have one's own definition of LaTeX environments for `Sinput`, `Soutput`, `Scode`, be it in an extra file or in a section in one's `.Rnw` file. Second, and this is Andrew's suggestion, by means of argument withVerbatim (default FALSE), you may from now on use LaTeX environments for `Sinput`, `Soutput`, `Scode` using `listings`-command `\lstnewenvironment` instead of the original `fancyvrb` definitions provided in the original `Sweave.sty` file by Fritz Leisch. This way we also solve the escaping problem (as noted by Frank E. Harrell): the escaping mechanisms provided by `listings` command `lstset` (as e.g. `escapechar`, `escapeinline`) as described in detail in (Heinz and Moses , 2007, section 4.14) are now available; in particular one can place LaTeX references `\ref{...}`, `\label{...}` within comments.
Just to show some little example:

```
\lstdefinestyle{Rinstyle}{style=RinstyleO,frame=trBL,backgroundcolor=\color{gray90},%
        numbers=left,numberstyle=\tiny,stepnumber=1,numbersep=7pt}
\lstdefinestyle{Routstyle}{style=RoutstyleO,frame=trBL,frameround=fttt,%
        backgroundcolor=\color{gray95},numbers=left,numberstyle=\tiny,%
        stepnumber=1,numbersep=7pt}
\begin{quotation}
<<exam00, eval = TRUE>>=
x <- rnorm(3) # define random numbers -> insert a label <here> (*\label{comment}*)
print(round(x,2))
f <- function(x) sin(x) ## here is a ref: section(*~\ref{PrepSec}*)
a <- 2; b <- 3
# compute (*$\int_a^b f(x)\,dx$*)
integrate(f, lower=a, upper=b)$value
@
Note that line~\ref{comment} contains a comment.
\end{quotation}
\lstdefinestyle{Rinstyle}{style=RinstyleO,frame=none,backgroundcolor=\color{white},%
        numbers=none}
\lstdefinestyle{Routstyle}{style=RoutstyleO,frame=none,backgroundcolor=\color{white},%
        numbers=none}
```

becomes

```
> x <- rnorm(3) # define random numbers -> insert a label <here>
> print(round(x,2))
```

```
[1] 0.27 0.13 1.00
```

```
> f <- function(x) sin(x) ## here is a ref: section 2
> a <- 2; b <- 3
> # compute ∫_a^b f(x) dx
> integrate(f, lower=a, upper=b)$value
```

```
[1] 0.5738457
```

Note that line 1 contains a comment.

Note that for easier reference later on, we produce copies of the original `listings` styles `Rinstyle`, `Routstyle`, `Rcodestyle`, named `RinstyleO`, `RoutstyleO`, `RcodestyleO`, respectively. If you really want to have background colors, frames and the like, however, you might wish to specify the corresponding options in the preparational chunk to have something like.

```
% ------------------------------------------------------------------------------
\RequirePackage{fancyvrb}
\RequirePackage{listings}
% ------------------------------------------------------------------------------
\SweaveOpts{keep.source=TRUE}
% ------------------------------------------------------------------------------
<<SweaveListingsPreparations, results=tex, echo=FALSE, strip.white=FALSE>>=
require(SweaveListingUtils)
 ## we are only appending new options so use
 ## c(getSweaveListingOption(.),<newoptions>)
SweaveListingPreparations(#
        Rin = c(getSweaveListingOption("Rin"),
                frame = "trBL",
                backgroundcolor = "\\color{gray90}",
                numbers = "left",
                numberstyle = "\\tiny",
                stepnumber = "1",
                numbersep = "7pt"),
        Rout = c(getSweaveListingOption("Rout"),
                 frame = "trBL",
                 frameround = "fttt",
                 backgroundcolor = "\\color{gray95}",
                 numbers = "left",
                 numberstyle = "\\tiny",
                 stepnumber="3",
                 numbersep = "5pt"))
@
```

3 Listings markup

3.1 Example of code coloring

Any keyword of some new R package "loaded in" by **require** or **library** which is on the **search** list item of this package afterwords when used in `\lstinline{ .... }` or `\begin{lstlisting} .... \end{lstlisting}` or in some Sweave chunk is typeset in style keywordstyle. More specifically, with argument keywordstyles of functions setToBeDefinedPkgs or lstsetLanguage all packages may obtain their own style; in the preamble, for instance, package `"SweaveListingUtils"` is colored blue, and package `"distr"` (to be attached just now) will be colored red. Also, comments are set in a different style (by default using color Rcommentcolor). Of course, instead of colors, you may use any other markup, like different font shapes, fonts, font sizes or whatever comes into your mind. For this purpose, commands setToBeDefinedPkgs and changeKeywordstyles are helpful.

Note that in order to define these new keywords correctly, they must not be included into a `\begin{Schunk} .... \end{Schunk}` environment, so we use

```
<<Prepa, echo=FALSE, results=tex, strip.white=FALSE>>=
require(distr)
## preparation: load package distr and register its keywords
@
```

The next example takes up package `"distr"`, confer Ruckdeschel et al. (2006), to illustrate particular markup for a particular package.

Example (note the different colorings):

```
<<exam1, eval=TRUE >>=
require(distr)
N <- Norm(mean = 2, sd = 1.3)
P <- Pois(lambda = 1.2)
Z <- 2*N + 3 + P
Z
p(Z)(0.4)
q(Z)(0.3)
@
```

which gives

```
> require(distr)
> N <- Norm(mean = 2, sd = 1.3)
> P <- Pois(lambda = 1.2)
> Z <- 2*N + 3 + P
> Z

Distribution Object of Class: AbscontDistribution

> p(Z)(0.4)

[1] 0.002415387

> q(Z)(0.3)

[1] 6.705068
```

Remark: `.Rd` keywords will be taken from file `Rdlistings.sty` in the `TeX` subfolder of this package, which is according to Murdoch (2010).

3.2 Changing the markup

Triggered by an e-mail by David Carslaw, this subsection lists some possibilities howto change the (default) markup of code.

Changing the global settings: The default markup for R code is set in a global option Rset to be inspected by

```
> getSweaveListingOption("Rset")

$fancyvrb
[1] "true"

$escapechar
[1] "`"

$extendedchars
[1] "false"

$language
[1] "R"

$basicstyle
[1] "{\\color{Rcolor}\\small}"

$keywordstyle
[1] "{\\bf\\color{Rcolor}}"

$commentstyle
[1] "{\\color{Rcommentcolor}\\ttfamily\\itshape}"

$literate
[1] "{\"}{\\texttt{\"}}1{←}{{$\\leftarrow$}}2{↞}{{$\\twoheadleftarrow$}}2{~}{{$\\sim$}}1{≤}

$alsoother
[1] "{$}"

$alsoletter
[1] "{.←}"

$otherkeywords
[1] "{!,!=,~,$,*,\\&,\\%/\\%,\\%*\\%,\\%\\%,←,↞,/}"

$escapeinside
[1] "{*)"
```

Similarly, default markup for `Rd` code is set in a global option Rset to be inspected by

```
> getSweaveListingOption("Rdset")

$fancyvrb
[1] "true"

$language
[1] "Rd"

$keywordstyle
[1] "{\\bf}"

$basicstyle
[1] "{\\color{black}\\footnotesize}"

$commentstyle
[1] "{\\ttfamily\\itshape}"

$alsolanguage
[1] "R"
```

The inspection / modification mechanism for these global options is the same as for the R global options, i.e., instead of the functions **options**, **getOption**, we have functions SweaveListingoptions, getSweaveListingOption; see also ?getSweaveListingOption.

Some comments are due:

The items of this list are just the tagged `name = value` list items to be passed as arguments to (TEX-)`listings` command `lstset`, and you may include any `name = value` pair allowed for . For details confer the documentation of the `listings` package, Heinz and Moses (2007).

As usual in R, backslashes have to be escaped, giving the double appearance in the examples listed above.

For cooperation of `listings` with `Sweave`, it is necessary, however, to use the tagged pair `"fancyvrb" = "true"`.

The colors used in the default setting are also set as global (`SweaveListing`-)options — i.e.; Rcolor, Rcommentcolor, Rdcolor.

Item `"literate"` will be discussed in the next subsection.

Using the escape character defined as item `"escapechar"`, you may force TEXto typeset (parts of) your comments in TEXstyle, which is handy for mathematical formula.

In an e-mail, Frank Harrell suggested to use R color names to assign markup colors as Rcolor. As in TEXwe are just using the command \color which expects a comma-separated list of the three rgb coordinates (scaled to be in $[0, 1]$), a good way to do this is, as Frank suggested, to use col2rgb(...)/255 to transform them to TEX-digestible format.

Changing the markup settings without changing defaults at startup: Alternatively, you may change global markup without modifying (`SweaveListing`-)option `"Rset"`. To this end you may build up your own (local) `"Rset"`-list, say Rset0. This is most easily done by first copying the global default list and then by modifying some items by simple R list operations. This might give the following alternative preparatory chunk to be inserted at the beginning of your `.Rnw` file.

```
<<SweaveListingsPreparations, results=tex, echo=FALSE, strip.white=FALSE>>=
require(SweaveListingUtils)
### just want to modify 1 entry of option Rset
### so first copy the default settings:
Rset0 <- getSweaveListingOption("Rset")
### change item "basicstyle" in the local copy
Rset0$"basicstyle" <- "{\\color{Rcolor}\\footnotestyle}"
SweaveListingoptions(intermediate = FALSE)
SweaveListingPreparations(Rset=Rset0)
@
```

Changing the markup locally: If you want to change the markup style within some `.Rnw` file, use something like:

```
<<changeStyle, results=tex, echo=FALSE, strip.white=FALSE>>=
lstsetR(Rset = list("basicstyle" = "{\\tiny}"))
@
```

This will add/replace item `"basicstyle"` to/in the existing items. For `Rd`-style you may use a respective call to lstsetRd(), and if you only want to modify the `Sinput`, `Soutput`, or `Scode` environments, you may use respective calls to lstsetRin, lstsetRout, lstsetRcode, respectively.

3.3 Using literate programming

This is —to some degree— a matter of taste: R has two assignment operators, which when typed look like `<-` and `<<-`; now literally these are interpreted as one token; the same goes for comparison operators like `<=`. One idea of literate programming is to replace these tokens by special symbols like ←, ↞, ≤ for printing to enhance readability — think of easy confusions arising between `<-` and `< -`.

TEX-package `listings`, confer Heinz and Moses (2007), to this end has the directive `literate`, and in our default setting for R markup, we use it at least for the replacement of the assignments.

Note that the `.Rnw` file still contains valid R code in the chunks; stangle will work just fine — the chunks are just output by TEX in a somewhat transformed way.

A considerable part of R users would rather prefer to see the code output "as you type it"; if you tend to think you like this, you are free of course to change the default markup as described in the previous section.

4 Including Code Sniplets from R Forge

When documenting code, which is not necessarily of the same package, and be it R code or `.Rd`-code, we provide helper functions to integrate code sniplets from an url (by default, we use the svn server at R-forge in its most recent version). This can be useful to stay consistent with the current version of the code without having to update vignettes all the time. To this end, besides referencing by line numbers, lstinputSourceFromRForge also offers referencing by matching regular expressions.

For instance, to refer to some code of file `R/AllClasses.R` in package `"distr"`, we would use:

```
<<AllClass, results=tex, echo=FALSE, strip.white=FALSE>>=
lstinputSourceFromRForge("distr","R","AllClasses.R","distr",
```

```
                    "## Class: BinomParameter", "#-")
@
```

which returns
lines 185–194

```
## Class: BinomParameter
setClass("BinomParameter",
         representation = representation(size = "numeric", prob = "numeric"),
         prototype = prototype(size = 1, prob = 0.5, name =
                     gettext("Parameter of a Binomial distribution")
                     ),
         contains = "Parameter"
         )

#-
```

Note the referencing with regular expressions instead of line numbers, which helps if you later on add/delete (other) code in this file.

To refer to a whole `.Rd` file, use something like the following chunk:

```
<<BinomParam, results=tex, echo=FALSE, strip.white=FALSE>>=
lstinputSourceFromRForge("distr","man","BinomParameter-class.Rd","distr")
@
```

giving

```
\name{BinomParameter-class}
\docType{class}
\alias{BinomParameter-class}
\alias{initialize,BinomParameter-method}

\title{Class "BinomParameter"}
\description{ The parameter of a binomial distribution, used by Binom-class}
\section{Objects from the Class}{
Objects can be created by calls of the form
      \code{new("BinomParameter", prob, size)}.
Usually an object of this class is not needed on its own, it is generated
automatically when an object of the class Binom
is instantiated.
}
\section{Slots}{
  \describe{
    \item{\code{prob}}{Object of class \code{"numeric"}:
          the probability of a binomial distribution }
    \item{\code{size}}{Object of class \code{"numeric"}:
          the size of a binomial distribution }
    \item{\code{name}}{Object of class \code{"character"}:
          a name / comment for the parameters }
  }
}
\section{Extends}{
Class \code{"Parameter"}, directly.
}
\section{Methods}{
  \describe{
    \item{initialize}{\code{signature(.Object = "BinomParameter")}:
```

```
        initialize method }
    \item{prob}{\code{signature(object = "BinomParameter")}: returns the slot
        \code{prob} of the parameter of the distribution }
    \item{prob<-}{\code{signature(object = "BinomParameter")}: modifies the slot
        \code{prob} of the parameter of the distribution }
    \item{size}{\code{signature(object = "BinomParameter")}: returns the slot
        \code{size} of the parameter of the distribution }
    \item{size<-}{\code{signature(object = "BinomParameter")}: modifies the slot
        \code{size} of the parameter of the distribution}
  }
}

\author{
  Thomas Stabla \email{statho3@web.de},\cr
  Florian Camphausen \email{fcampi@gmx.de},\cr
  Peter Ruckdeschel \email{Peter.Ruckdeschel@itwm.fraunhofer.de},\cr
  Matthias Kohl \email{Matthias.Kohl@stamats.de}
  }


\seealso{
\code{\link{Binom-class}}
\code{\link{Parameter-class}}
}

\examples{
   W <- new("BinomParameter",prob=0.5,size=1)
   size(W) # size of this distribution is 1.
   size(W) <- 2 # size of this distribution is now 2.
}
\keyword{distribution}
\concept{parameter}
\concept{Binomial distribution}
\concept{S4 parameter class}
```

References

Heinz, C and Moses, B (2007) The Listings package. Manual for TEX package `listings` version 1.4. http://www.ctan.org/get/macros/latex/contrib/listings/listings.pdf. 1, 5, 9, 10

Leisch, F (2002a) Sweave: Dynamic generation of statistical reports. In: Härdle, W and Rönz, B (eds): *Compstat 2002 - Proceedings in Computational Statistics*. pp. 575–580. Physika Verlag, Heidelberg. http://www.statistik.lmu.de/~leisch/Sweave/. 1

Leisch, F (2002b) Sweave, Part I: Mixing R and LATEX. *R-News*, **2**(3): 28–31. http://cran.r-project.org/doc/Rnews/Rnews_2002-3.pdf. 1

Leisch, F (2003) Sweave, Part II: Package Vignettes. *R-News*, **3**(2): 21–24. http://cran.r-project.org/doc/Rnews/Rnews_2003-2.pdf. 1

Murdoch, D (2010) Parsing Rd Files. Technical Report on `developer.r-project.org` as of Jan. 1 2010. http://developer.r-project.org/parseRd.pdf. 8

R-Forge Administration and Development Team (2008). *R-Forge User's Manual*. http://download.R-Forge.R-project.org/R-Forge.pdf. 1

Ruckdeschel P, Kohl M, Stabla T, and Camphausen F (2006) S4 Classes for Distributions. *R-News*, **6**(2): 10–13. http://cran.r-project.org/doc/Rnews/Rnews_2006-2.pdf. 7