

Package ‘SweaveListingUtils’

November 4, 2009

Title Utilities for Sweave together with TeX listings package

Encoding latin1

Version 0.4

Depends R(>= 2.10.0), startupmsg

Suggests distr

Imports stats

LazyLoad yes

Author Peter Ruckdeschel

Description provides utilities for defining R / Rd as Tex-package-listings “language” and including R / Rd source file (snippets) copied from R-forge in its most recent version (or another url) thereby avoiding inconsistencies between vignette and documented source code

Maintainer Peter Ruckdeschel <Peter.Ruckdeschel@itwm.fraunhofer.de>

License LGPL-3

Date 2009-10-16

LastChangedDate {\$LastChangedDate: 2009-10-16 05:53:10 +0200 (Fr, 16. Okt 2009) \$}

LastChangedRevision {\$LastChangedRevision: 610 \$}

Repository CRAN

Date/Publication 2009-11-03 13:47:37

R topics documented:

SweaveListingUtils-package	2
changeKeywordstyles	4
copySourceFromRForge	6
library	7
lstinputSourceFromRForge	11
lstset	13
lstsetLanguage	15
readPkgVersion	17
readSourceFromRForge	18
setToBeDefinedPkgs	19
SweaveListingMASK	20
SweaveListingOptions	20
SweaveListingPreparations	23
taglist	26
Index	28

SweaveListingUtils-package
Package SweaveListingUtils

Description

Package **SweaveListingUtils** provides utilities for defining R / ‘Rd’ as TeX-package-‘listings’ "language" and including R / ‘.Rd’ source file (snippets) copied from an url, by default from svn server at R-forge in its most recent version, thereby avoiding inconsistencies between vignette and documented source code

Details

Package: SweaveListingUtils
Version: 0.4
Date: 2009-09-04
Depends: R(>= 2.10.0), startupmsg
LazyLoad: yes
License: LGPL-3

TeX-package ‘listings’, confer <http://www.ctan.org/tex-archive/macros/latex/contrib/listings/>, <http://www.ctan.org/tex-archive/macros/latex/contrib/listings/listings.pdf> offers a very powerful setup for typesetting [program] code in TeX.

For quite some time there has already been a corresponding language definition for R. We enhance this definition and also introduce a corresponding "language" definition file to typeset ‘.Rd’ code (file ‘Rdlistings.sty’ in the ‘TeX’ subfolder of this package, which is according to Duncan Murdoch’s “Parsing Rd Files” as of Nov. 4 2008.).

In recent versions ‘listings’ also cooperates with TeX package ‘fancyvrb’, so it can be configured to enhance Sweave typesetting.

Just as a first simple example, comments are recognized automatically and hence typeset in particular format.

For pretty printing, or moreover literate programming, with **SweaveListingUtils**, assignment operators `<-` and `<<-` get typeset by one symbol each.

For a corresponding TeX preamble combining Sweave and ‘listings’, we provide command [SweaveListingPreparations](#).

As for the R language definition, we allow for different keywordstyles to typeset symbols from different packages. This is useful to distinguish mark-up for newly defined functions and already existing ones.

More specifically, whenever in some R code snippet in some Sweave chunk, there is some `library` or `require` command, the corresponding symbols found by `ls()` afterwords in some `search()` entry position, are registered as `<comma-separated keywordlist>` (printed as 5 items per line) as a new "higher order" group of keywords by corresponding

```
\lstdefinestyle{RstyleO<numi>}% RstyleO<numi> is the current order of Rstyle
  {style = RstyleO<numi-1>,
   morekeywords=[<order number>]{ <comma-separated keywordlist> },%
   sensitive=true,%
   keywordstyle=[<order number>]<keywordstyle as format string>,%
   % [ possibly more
   morekeywords=[<order number + 1>]{ <comma-separated keywordlist> },%
   sensitive=true,%
   keywordstyle=[<order number + 1>]<keywordstyle as format string>,%
   % .... ]
  }
\lstdefinestyle{Rstyle}{style = RstyleO<numi>}
```

TeX directives in the automatically generated ‘.tex’ file where `<order number>` is incremented (resp. gets looked up from the global, non-exported variable `.alreadyDefinedPkgs` in the package name space) at each instance of a `library` or `require` command; an analogue incrementation present in `<numi>` is done for subsequent (incremental) redefinitions of style `Rstyle`, which is controlled by the again non-exported global variable `.numberOfRequires`. `<order number>` and `<numi>` will in general differ, as in one call to `require/library`, several packages may be registered at once.

To this end commands `library` and `require` are masked. See also [lstsetLanguage](#), [changeKeywordstyles](#), and [setToBeDefinedPkgs](#)

As for the integration of code snippets 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.

Functions

```
lstset
lstsetR
```

```
lstsetRd
SweaveListingPreparations
readSourceFromRForge
copySourceFromRForge
lstinputSourceFromRForge
readPkgVersion
SweaveListingOptions
getSweaveListingOption
SweaveListingoptions
SweaveListingMASK
setToBeDefinedPkgs
lstsetLanguage
changeKeywordstyles
library
require
```

S3 classes and methods

```
taglist
print.taglist
```

Example - .Rnw vignette file

An example '.Rnw' file, 'ExampleSweaveListingUtils.Rnw', may be found in the 'doc' folder of this package.

Note

This version already uses 'new-style' Rd-format version 1.1. For versions installable for <R-2.10.0, try and get a version 0.3.x of this package from a suitable repository.

Author(s)

Peter Ruckdeschel <Peter.Ruckdeschel@itwm.fraunhofer.de>
Maintainer: Peter Ruckdeschel <Peter.Ruckdeschel@itwm.fraunhofer.de>

```
changeKeywordstyles
      changeKeywordstyles
```

Description

sets up / updates a table of keywordstyles to different packages

Usage

```
changeKeywordstyles(pkgs, keywordstyles)
```

Arguments

`pkgs` character; the packages for which keywordstyle information is to be changed

`keywordstyles` character or missing; the corresponding keywordstyle format strings; if missing the corresponding option `Keywordstyle` is read off by using `getSweaveListingOption("Keywordstyle")`. Internally, it is being cast to the same length as `pkgs` by `rep(keywordstyles, length.out = length(pkgs))`.

Details

Before changing the keywordstyles, we first check whether the corresponding package is registered at all —by looking up the (non-exported) vector object `.alreadyDefinedPkgs`, which is hidden in the namespace of this package.

For changing the keywordstyle, we write out a

```
\lstdefinlanguage{R}%
  {keywordstyle=[<order number>]<keywordstyle as format string>,%
  }
```

directive to standard out, where `<keywordstyle as format string>` is a string containing any sequence of TeX formatting commands like `"\\bfseries\\footnotesize"`. Note that backslashes have to be escaped. and `<order number>` is just `num+2` where `num` is the index of the package in the `.alreadyDefinedPkgs` vector.

For use in an `.Rnw` file, the call to `lstsetlanguage` should be wrapped into a corresponding Sweave chunk in the form

```
<< /chunkname/, results=tex, echo=FALSE>>=
changeKeywordstyles( . . . . . )
@
```

for example

```
<<distrRegisterKeywords, results=tex, echo=FALSE>>=
changeKeywordstyles(pkgs = "distr",
                    keywordstyles = "
bfseries
color{blue}")
@
```

Value

```
invisible()
```

Author(s)

Peter Ruckdeschel <Peter.Ruckdeschel@itwm.fraunhofer.de>

See Also[lstsetLanguage](#)**Examples**

```
require(MASS)
lstsetLanguage(pkgs = c("MASS", "stats"),
               keywordstyles = paste("\\bfseries\\color{", c("blue", "red"), "}",
                                     sep="", collapse=""))
changeKeywordstyles(pkgs = c("distr", "distrEx"),
                    keywordstyles = paste("\\bfseries\\color{", c("green", "blue"), "}",
                                           collapse="", sep = ""))
```

```
copySourceFromRForge
```

```
copySourceFromRForge
```

Description

copies lines of a source file (usually '.R' oder '.Rd') from R forge repository

Usage

```
copySourceFromRForge(PKG, TYPE, FILENAME, PROJECT, from, to,
                     offset.before = 0, offset.after = 0,
                     fromRForge = getSweaveListingOption("fromRForge"),
                     base.url = getSweaveListingOption("base.url") )
```

Arguments

PKG	character; name of package to be downloaded
TYPE	character; style of the source code — "man" or "R"
FILENAME	character; the name of the source file to be downloaded
PROJECT	character; the name of the R-Forge project
from	single character or single numeric or missing; if character, the starting string being searched (by <code>grep</code> , hence as regular expression); if numeric, the starting line number, if missing we begin with the first line of the file
to	single character or single numeric or missing; if character, the ending string being searched (by <code>grep</code> , hence as regular expression); if numeric, the ending line number, if missing we end with the last line of the file
offset.before	numeric; number of lines to be included before the first match; defaults to 0
offset.after	numeric; number of lines to be included after the first match; defaults to 0
fromRForge	logical; shall code be downloaded from an R-Forge mirror? Defaults to the corresponding global option
base.url	character; base url from where to download the code snippet

Details

produces a vector of characters where each component is one line of the original source file; arguments `from`, `to` may be

- `missing`: then the whole file (resp. from the beginning or to the end) is used
- `numbers`: then the limits are just given as line numbers
- `characters`: then file is searched for the first occurrence of a passage framed by `from`, `to`; uses `grep`; hence [regular expressions](#) may be used which involves masking of `(, {, \`, etc. as described in the cited reference;

with offsets, additional lines may be pasted before and after the search result

Value

the character content of the filtered source file, if nothing is found it returns `invisible()`.

Author(s)

Peter Ruckdeschel <Peter.Ruckdeschel@itwm.fraunhofer.de>

Examples

```
copySourceFromRForge("distr", "R", "AllClasses.R", "distr", from =2, to =3,
                      offset.after=2)
copySourceFromRForge("distr", "R", "AllClasses.R", "distr", from ="setClass",
                      to ="\}")
```

library

Loading Packages with registering symbols for TeX package 'listing'

Description

`library` and `require` load add-on packages. In the masked versions of package **SweaveListingUtils** documented here, they also register corresponding symbols. Besides this registration, they behave identically to the original versions of package **base**. Hence much of this help is also just copied from the original help page.

Usage

```
library(package, help, pos = 2, lib.loc = NULL,
         character.only = FALSE, logical.return = FALSE,
         warn.conflicts = TRUE,
         keep.source = getOption("keep.source.pkgs"),
         verbose = getOption("verbose"),
         version, inSweave,
         keywordstyles, interm.keywordstyles, overwrite, intermediate)
```

```
require(package, lib.loc = NULL, quietly = FALSE,
        warn.conflicts = TRUE,
        keep.source = getOption("keep.source.pkgs"),
        character.only = FALSE, version, save = TRUE, inSweave,
        keywordstyles, interm.keywordstyles, overwrite, intermediate)
```

Arguments

`package`, `help` the name of a package, given as a [name](#) or literal character string, or a character string, depending on whether `character.only` is `FALSE` (default) or `TRUE`).

`pos` the position on the search list at which to attach the loaded package. Note that `.First.lib` may attach other packages, and `pos` is computed *after* `.First.lib` has been run. Can also be the name of a position on the current search list as given by [search\(\)](#).

`lib.loc` a character vector describing the location of R library trees to search through, or `NULL`. The default value of `NULL` corresponds to all libraries currently known. Non-existent library trees are silently ignored.

`character.only` a logical indicating whether `package` or `help` can be assumed to be character strings.

`version` A character string denoting a version number of the package to be loaded, for use with *versioned installs*: see the section later in this document.

`logical.return` logical. If it is `TRUE`, `FALSE` or `TRUE` is returned to indicate success.

`warn.conflicts` logical. If `TRUE`, warnings are printed about [conflicts](#) from attaching the new package, unless that package contains an object `.conflicts.OK`. A conflict is a function masking a function, or a non-function masking a non-function.

`keep.source` logical. If `TRUE`, functions ‘keep their source’ including comments, see argument `keep.source` to [options](#). This applies only to the named package, and not to any packages or name spaces which might be loaded to satisfy dependencies or imports.
This argument does not apply to packages using lazy-loading. Whether they have kept source is determined when they are installed (and is most likely false).

`verbose` a logical. If `TRUE`, additional diagnostics are printed.

`quietly` a logical. If `TRUE`, no message confirming package loading is printed.

`save` logical or environment. If `TRUE`, a call to `require` from the source for a package will save the name of the required package in the variable `".required"`, allowing function [detach](#) to warn if a required package is detached. See section ‘Packages that require other packages’ below.

`inSweave` shall the command show Sweave behaviour (no startup messages; instead issuing new symbols for `morekeywords` tag in language definition of R in TeX package ‘listings’)? By default argument is taken from [getSweaveListingOption](#).

- `keywordstyles` character or missing or NULL; added argument in masked versions of `library` and `require` of package **SweaveListingUtils**; if given, the keywordstyle to be used by TeX package 'listings' for the symbols attached in this package (to be found with `ls()`). Remember that special characters like `\` have to be escaped. If missing, a special strategy is used to get sensible value (see details).
- `interm.keywordstyles` character or missing or NULL; added argument in masked versions of `library` and `require` of package **SweaveListingUtils**; if given, the keywordstyle to be used by TeX package 'listings' for the symbols to be found by automatically loaded intermediate packages (mentioned in 'depends' field of the corresponding DESCRIPTION file). This will only be used if effective argument `intermediate` is TRUE. Remember that special characters like `\` have to be escaped. If missing, a special strategy is used to get sensible value (see details).
- `overwrite` logical or missing or NULL; added argument in masked versions of `library` and `require` of package **SweaveListingUtils**; if missing, the corresponding global option is read out by `getSweaveListingOption`. Let us call the corresponding possibly imputed value *effective* value. If this effective value is TRUE, before registering a new symbol for TeX package 'listings', we first check whether this symbol had already been registered in the original 'listings'-language definition of R in file '`lstlang3.sty`' [which we have copied to the internal (un-exported) constant `.keywordsR`] and if so do not include the symbol once again; if the value is FALSE, no matter of the original language definition all symbols to be found by `ls()` are registered and hence typeset in the corresponding package-keywordstyle.
- `intermediate` logical or missing or NULL; added argument in masked versions of `library` and `require` of package **SweaveListingUtils**; if missing, the corresponding global option is read out by `getSweaveListingOption`. If the effective value is TRUE, also the symbols to be found in newly, automatically loaded packages (loaded, because the required/"library"-ed package features them in its corresponding Depends field in its DESCRIPTION file) are registered for TeX package 'listings'.

Details

`library` and `require` are masked versions of the original versions in package **base**. This masking is necessary to allow for extra arguments `keywordstyles`, `interm.keywordstyles`, `overwrite`, `intermediate`. Hence please confer [library](#), [require](#).

If argument `inSweave` is FALSE or if argument `inSweave` is missing and the corresponding global option `getSweaveListingOption("inSweave")` is FALSE, then `library` and `require` behave exactly as the **base** package versions; only if the effective argument is TRUE, the corresponding startup-messages are suppressed, and the TeX code for registering all new symbols to be found with `ls` as `morekeywords` for the R language definition in TeX package 'listings' is issued (and hence can be used in '.Rnw' files).

The strategy to fill arguments `keywordstyles` and `interm.keywordstyles` if missing is as follows: First we check whether for the corresponding package, there is an entry in the

global option variable `.tobeDefinedPkgs` (which is a matrix residing in the package namespace with one column of package names and one column of corresponding keywordstyle format strings) and if so use its corresponding keywordstyle; else we use option `Keywordstyles` resp. `intern.Keywordstyles` (which we get by `getSweaveListingOption`). New entries in the global variable `.tobeDefinedPkgs` may be appended by `setToBeDefinedPkgs`.

The masked versions work as follows: first the call is written to an object and split into a **base-package-part** and into the extra arguments mentioned above. Then we call `base::<fct>` where `<fct>` is either `require` or `library`. By noting the list of attached packages before and after this call we also get hand on the intermediate packages, which are automatically attached packages by dependence. After this call we register the new symbols by corresponding calls to `lstsetLanguage`.

For use in an `.Rnw` file, the call to `require` or `library` should be wrapped into a corresponding Sweave chunk in the form

```
<< /chunkname/, results=tex, echo=TRUE>>=
require( ..... )
@
```

,for example

```
<< distrRequire, results=tex, echo=TRUE>>=
require(distr)
@
```

In order to unmask the original versions of `library` and `require` again after Sweave-ing your `.Rnw` file, at the end of your `.Rnw` file you should include an Sweave chunk like

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

or, if you want to keep the namespace of **SweaveListingUtils** loaded

```
<<cleanup2, echo=FALSE>>=
SweaveListingUtils(inSweave = FALSE)
@
```

Value

as the unmasked versions.

See Also

[library](#).

Examples

```
require(survival)
```

```
lstinputSourceFromRForge
      lstinputSourceFromRForge
```

Description

copies lines of a source file (usually ‘.R’ oder ‘.Rd’) from R forge repository

Usage

```
lstinputSourceFromRForge(PKG, TYPE, FILENAME, PROJECT, from, to,
      offset.before = 0, offset.after = 0,
      LineLength = getOption("width"),
      withLines = ifelse(TYPE=="R", TRUE, FALSE),
      fromRForge = getSweaveListingOption("fromRForge"),
      base.url = getSweaveListingOption("base.url"))
```

Arguments

PKG	character; name of package to be downloaded
TYPE	character; style of the source code — "man" or "R"
FILENAME	character; the name of the source file to be downloaded
PROJECT	character; the name of the R-Forge project
from	vector of characters or vector of numerics or missing; beginnings of the code snippets; for details see copySourceFromRForge
to	vector of characters or vector of numerics or missing; endings of the code snippets; for details see copySourceFromRForge
offset.before	numeric; numbers of lines to be included before the code snippets; defaults to 0
offset.after	numeric; numbers of lines to be included after the the code snippets; defaults to 0
LineLength	numeric number of characters per line; defaults to <code>getOption("width")</code> ;
withLines	logical; shall line-numbers be issued
fromRForge	logical; shall code be downloaded from an R-Forge mirror? Defaults to the corresponding global option
base.url	character; base url from where to download the code snippet

Details

includes [downloaded] code snippets in ‘.R’ ‘.Rd’ format in some TeX-`lstlistings` environment; output is issued on stdout, hence included in ‘.Rnw’ file if wrapped to

```
<< /chunkname/, results=tex, echo=FALSE>>=
... R code ...
@
```

For example

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

<<skew, results=tex, echo=FALSE>>=
lstinputSourceFromRForge("distrEx", "R", "Skewness.R", "distr",
                          from = "\"skewness\"", signature
(x = \"Binom\"",
                          to = "\ }
) ")
@
```

(CAVEAT: the space between backslash and right brace in the line with "to =" is *not* intended; I simply did not find another work-around)

Within ‘.Rd’ code, examples are type-set in ‘.R’ style; vectors are allowed for arguments from, to; lstinputSourceFromRForge uses [copySourceFromRForge](#) for download / reading from cache; line numbers in the downloaded source may be printed out

Value

```
invisible()
```

Author(s)

Peter Ruckdeschel <Peter.Ruckdeschel@itwm.fraunhofer.de>

Examples

```
lstinputSourceFromRForge("distr", "R", "AllClasses.R", "distr",
                          "## Class: BinomParameter", "#-")

lstinputSourceFromRForge("distr", "R", "AllClasses.R", "distr",
                          from = "## Class: binomial distribution",
                          to = "contains = \"LatticeDistribution\"", offset.after = 1)
lstinputSourceFromRForge("distr", "man", "Binom-class.Rd", "distr")

lstinputSourceFromRForge("distr", "R", "BinomialDistribution.R", "distr",
                          from = c("## Access Methods", "## wrapped access methods"),
                          to = c("setReplaceMethod\\(\"prob\", \"BinomParameter\",
                                \"size = value\\\")\"),
                          offset.after = c(1,1))
lstinputSourceFromRForge("distr", "R", "BinomialDistribution.R", "distr",
                          from = c(8,43,45), to = c(16,53,45))
lstinputSourceFromRForge("distr", "R", "BinomialDistribution.R", "distr",
                          from = c("## Access Methods", "## wrapped access methods"),
                          to = c("setReplaceMethod\\(\"prob\", \"BinomParameter\",
                                \"size = value\\\")\"),
                          offset.after = c(1,1))
```

lstdset

*lstdset and friends***Description**

Functions for defining how listings prints R and Rd source code

Usage

```
lstdset(taglist, LineLength = getOption("width"), startS = "\\lstdset{")
lstdsetR(Rset = NULL, LineLength = getOption("width"),
        add = getSweaveListingOption("addRset"),
        startS = "\\lstdset{", append = TRUE, withRstyle = FALSE)
lstdsetRd(Rdset = NULL, LineLength = getOption("width"),
        add = getSweaveListingOption("addRdset"),
        startS = "\\lstdset{",
        append = TRUE)
lstdsetRin(Rinset = NULL, LineLength = getOption("width"),
        add = getSweaveListingOption("addRinset"),
        startS = "\\lstdefinestyle{Rinstyle}\\{",
        append = TRUE)
lstdsetRout(Routset = NULL, LineLength = getOption("width"),
        add = getSweaveListingOption("addRoutset"),
        startS = "\\lstdefinestyle{Routstyle}\\{",
        append = TRUE)
lstdsetRcode(Rcodeset = NULL, LineLength = getOption("width"),
        add = getSweaveListingOption("addRcodeset"),
        startS = "\\lstdefinestyle{Rcodestyle}\\{",
        append = TRUE)
lstdsetRall(Rallset = NULL, LineLength = getOption("width"),
        add = c("in" = getSweaveListingOption("addRinset"),
              "out" = getSweaveListingOption("addRoutset"),
              "code" = getSweaveListingOption("addRcodeset")),
        startS = c("in" = "\\lstdefinestyle{Rinstyle}\\{",
                  "out" = "\\lstdefinestyle{Routstyle}\\{",
                  "code" = "\\lstdefinestyle{Rcodestyle}\\{"),
        append = c("in" = TRUE, "out" = TRUE, "code" = TRUE),
        withOptionsDefAppend = TRUE)
lstdsetRstyle(Rset = NULL, LineLength = getOption("width"),
        add = TRUE)
```

Arguments

LineLength	numeric number of characters per line for lstdset and friends; defaults to <code>getOption("width")</code>
taglist	S3-object of class taglist; arguments for <code>\lstdset</code> of TeX package <i>listings</i> .

Rset	object of S3-class <code>taglist</code> or named list of characters; the listings settings for R-code (for <code>\lstset</code>); defaults to NULL.
Rinset	object of S3-class <code>taglist</code> or named list of characters; the listings settings for R-code in environment <code>Sinput</code> ; defaults to NULL.
Routset	object of S3-class <code>taglist</code> or named list of characters; the listings settings for R-code in environment <code>Soutput</code> ; defaults to NULL.
Rcodeset	object of S3-class <code>taglist</code> or named list of characters; the listings settings for R-code in environment <code>Scode</code> ; defaults to NULL.
Rallset	object of S3-class <code>taglist</code> or named list of characters; the listings settings for R-code in environments <code>Sinput</code> , <code>Soutput</code> , <code>Scode</code> , simultaneously; defaults to NULL.
Rdset	object of S3-class <code>taglist</code> or named list of characters; the listings settings for Rd-code (for <code>\lstset</code>); defaults to NULL.
add	boolean; defaults to TRUE; if TRUE, argument list <code>Rset</code> resp. <code>Rdset</code> will be appended to default value lists <code>getSweaveListingOption("Rset")</code> resp. <code>getSweaveListingOption("Rdset")</code> (see below), overwriting respective entries of the default value lists; in case of <code>lstsetRall</code> may be named vector of length 3 with names <code>c("in", "out", "code")</code> .
startS	character; defaults to <code>"\lstset{"</code> ; what to do by default we use <code>\lstset</code> ; an alternative is to use <code>\lstdefinestyle</code> which amounts to <code>"\lstdefinestyle{"</code> ; in case of <code>lstsetRall</code> may be named vector of length 3 with names <code>c("in", "out", "code")</code> .
append	logical; if TRUE (default) the new settings are appended to the old ones; otherwise they are prepended; in case of <code>lstsetRall</code> may be named vector of length 3 with names <code>c("in", "out", "code")</code> .
withRstyle	logical; if TRUE <code>"style = Rstyle, "</code> is prepended.
withOptionsDefAppend	logical: shall definitions from global options (see SweaveListingoptions) be included?

Details

`lstset` writes out to stdout a call to TeX command `\lstset{arg1 = val1, arg2 = val2,}` and doing so respects a maximal number of characters per line and does not break `arg=val` tags.

`lstsetR` and `lstsetRd` expect either objects of S3 class `taglist`, or lists of named characters as first arguments, which in the latter case are then converted to `taglist`; both `lstsetR` and `lstsetRd` use particular default values to define R resp. Rd output format. More specifically for R code, it uses [getSweaveListingOption\("Rset"\)](#), and for Rd code, it [getSweaveListingOption\("Rdset"\)](#); `lstsetRin`, `lstsetRout`, and `lstsetRcode` are corresponding specialized commands for 'listings' environments `Sinput`, `Soutput`, and `Scode`, respectively.

`lstsetRall` simultaneously sets/modifies settings for 'listings' environments `Sinput`, `Soutput`, and `Scode`.

`lstsetdefRstyle` redefines listings style definition for `Rstyle`.

The output to stdout can be captured in an `.Rnw` file as

```
<< lstsetR, results=tex, echo=FALSE>>=
lstsetR()
@
```

to insert the corresponding `\lstset` command to the produced TeX file.

Value

```
invisible()
```

Author(s)

Peter Ruckdeschel <Peter.Ruckdeschel@itwm.fraunhofer.de>

Examples

```
lstset(taglist(A="H", b=2, 3),30)
lstset(taglist(A="H", b=2, 3),30, startS = "\\lstdefinestyle{Rstyle}\{")
lstsetR()
lstsetRd()
```

<code>lstsetLanguage</code>	<i>l<code>stsetLanguage</code></i>
-----------------------------	------------------------------------

Description

registers the symbols of a package or a position in the search list as *morekeywords* for the TeX-package ‘listings’ language definition of R

Usage

```
lstsetLanguage(pkgs, posIdx, keywordstyles, overwrite = FALSE)
```

Arguments

<code>pkgs</code>	character; the packages the symbols of which are to be registered.
<code>posIdx</code>	numeric; positions in the <code>search()</code> list from which the symbols are to be registered.
<code>keywordstyles</code>	character or missing; the corresponding keywordstyle format strings; if missing the corresponding option <code>Keywordstyle</code> is read off by using <code>getSweaveListingOption("Keywordstyle")</code> . Internally, it is being cast to the same length as <code>pkgs</code> by <code>rep(keywordstyles, length.out = length(pkgs))</code> .
<code>overwrite</code>	logical; before registering the new symbols shall we check if there already is a registration of this symbol in the original R language definition for TeX package ‘listings’ of file ‘l <code>stlang3.sty</code> ’ provided by Robert Denham; in package SweaveListingUtils , this information is available in the non-exported global object <code>.keywordsR</code> in the namespace of this package; if TRUE we overwrite existing registrations; default is FALSE.

Details

Arguments `pkgs` and `posIdx` can be used independently from each other: If there is an argument `pkgs`, after checking whether these packages are already on the search list, we unite the corresponding search list positions with those of argument `posIdx` (if the latter is given); the positions corresponding to packages already in the `.alreadyDefinedPkgs` vector (see below), are filtered out, however. If argument `pkgs` is missing, by default the whole list of attached packages gained from `.packages()` is taken in the beginning. For registering the new symbols, we write out a

```
\lstdefinelanguage{R}%
  {morekeywords=[<order number>]{ <comma-separated keywordlist> },%
   sensitive=true,%
   keywordstyle=[<order number>]<keywordstyle as format string>,%
  }
```

directive to standard out, where `<comma-separated keywordlist>` is a comma-separated list of the keywords to be registered printed out as five items per line; we get this list by a corresponding `ls(pos=<position>)` command. If argument `overwrite` is `FALSE`, before registration, we filter out the keywords already in the original ‘listings’ R language definition. `<keywordstyle as format string>` may be a string containing any sequence of TeX formatting commands like `"\bfseries\footnotesize"`. Note that backslashes have to be escaped.

To be able to distinguish/manage several keyword format styles on R-side, we append the name of each package, the symbols of which are registered, to the (non-exported) vector object `.alreadyDefinedPkgs`, which is hidden in the namespace of this package.

On TeX/‘listings’-side, the different keyword format styles are managed by the corresponding `<order number>` information in the `morekeywords` tag; it is identified with `num+2` where `num` is the index of the package in the `.alreadyDefinedPkgs` vector.

The settings of these format styles may afterwards be overwritten using [changeKeywordstyles](#).

For use in an `.Rnw` file, the call to `lstsetlanguage` should be wrapped into a corresponding Sweave chunk in the form

```
<< /chunkname/, results=tex, echo=FALSE>>=
lstsetLanguage( ..... )
@
```

for example

```
<<distrRegisterKeywords, results=tex, echo=FALSE>>=
lstsetLanguage("distr", keywordstyles = "
bfseries
color{green}")
@
```

Value

```
invisible()
```

Author(s)

```
Peter Ruckdeschel <Peter.Ruckdeschel@itwm.fraunhofer.de>
```

Examples

```
require(MASS)
lstsetLanguage(pkgs = c("MASS", "stats"),
               keywordstyles = paste("\\bfseries\\color{", c("blue", "red"), "}",
                                     sep="", collapse=""))
### not to be used:
print(SweaveListingUtils:::.alreadyDefinedPkgs)
print(SweaveListingUtils:::.keywordsR)
```

```
readPkgVersion      readPkgVersion
```

Description

reads the package version out of the 'DESCRIPTION' file

Usage

```
readPkgVersion(package, lib.loc = NULL)
```

Arguments

package	character; name of the package
lib.loc	location of a local library in which the described package resides

Value

package version as character if existing else ""

Author(s)

```
Peter Ruckdeschel <Peter.Ruckdeschel@itwm.fraunhofer.de>
```

Examples

```
readPkgVersion(package = "distr")
```

```
readSourceFromRForge
      readSourceFromRForge
```

Description

reads a source file (usually '.R' oder '.Rd') from R forge repository

Usage

```
readSourceFromRForge(PKG, TYPE, FILENAME, PROJECT,
                     fromRForge = getSweaveListingOption("fromRForge"),
                     base.url = getSweaveListingOption("base.url"))
```

Arguments

PKG	character; name of package to be downloaded
TYPE	character; style of the source code — "man" or "R"
FILENAME	character; the name of the source file to be downloaded
PROJECT	character; the name of the R-Forge project
fromRForge	logical; shall code be downloaded from an R-Forge mirror? Defaults to the corresponding global option
base.url	character; base url from where to download the code snippet

Details

reads source file from R-forge [from most recent revision in trunc], but caches the result in a global list in the namespace of package **SweaveListingUtils**, so if the URL has already been cached, the cached copy is used. The cached copies are stored in list `.CacheFiles`; the current length of this list is stored in `.CacheLength`.

Value

the character content of the source file

Author(s)

Peter Ruckdeschel <Peter.Ruckdeschel@itwm.fraunhofer.de>

Examples

```
readSourceFromRForge("distr", "R", "AllClasses.R", "distr")
```

```
setToBeDefinedPkgs setToBeDefinedPkgs
```

Description

sets up / updates a table of keywordstyles to different packages

Usage

```
setToBeDefinedPkgs(pkgs, keywordstyles)
```

Arguments

`pkgs` character; the packages for which keywordstyle information is to be stored

`keywordstyles` character or missing; the corresponding keywordstyle format strings; if missing the corresponding option `Keywordstyle` is read off by using `getSweaveListingOption("Keywordstyle")`. Internally, it is being cast to the same length as `pkgs` by `rep(keywordstyles, length.out = length(pkgs))`.

Details

The corresponding table is stored globally in the (non-exported) object `.tobeDefinedPkgs`, which is hidden in the namespace of this package.

It is used afterwards by the masked versions of `require` and `library` of this package to allow for defining a set of keywordstyle formats for different packages right in the preamble of a `.Rnw` file.

This transfer of information to `require` and `library` clearly is a deviation from the functional programming paradigm but is necessary at this place, as otherwise (although this is still allowed) `require` and `library` would have to be called with non-standard (i.e. package **base**-) arguments, which is *not* the goal of including R code snippets by Sweave.

Value

```
invisible()
```

Author(s)

```
Peter Ruckdeschel <Peter.Ruckdeschel@itwm.fraunhofer.de>
```

Examples

```
setToBeDefinedPkgs(pkgs = c("distr", "distrEx"),
  keywordstyles = paste("\bfseries\color{", c("blue", "red"), "}",
    sep="", collapse=""))
### not to be used:
print(SweaveListingUtils::.tobeDefinedPkgs)
```

SweaveListingMASK *Masking off/by other functions in package "SweaveListingUtils"*

Description

Provides information on the (intended) masking of and (non-intended) masking by other other functions in package **SweaveListingUtils**

Usage

```
SweaveListingMASK(library = NULL)
```

Arguments

`library` a character vector with path names of R libraries, or `NULL`. The default value of `NULL` corresponds to all libraries currently known. If the default is used, the loaded packages are searched before the libraries

Value

no value is returned

Author(s)

Peter Ruckdeschel <Peter.Ruckdeschel@itwm.fraunhofer.de>

Examples

```
SweaveListingMASK()
```

SweaveListingOptions

Function to change the global options of the package 'SweaveListingUtils'

Description

With `SweaveListingOptions` you can inspect and change the global variables of the package **SweaveListingUtils**.

Usage

```
SweaveListingOptions(...)  
SweaveListingoptions(...)  
getSweaveListingOption(x)
```

Arguments

- ... any options can be defined, using name = value or by passing a list of such tagged values.
- x a character string holding an option name.

Value

SweaveListingOptions() returns a list of the global variables.
 SweaveListingOptions(x) returns the global variable x.
 getSweaveListingOption(x) returns the global variable x.
 SweaveListingOptions(x=y) sets the value of the global variable x to y.

SweaveListingoptions

For compatibility with spelling in package **dist**, SweaveListingoptions is just a synonym to SweaveListingOptions.

Global Options

Rset: default setting for \lstset-definition of R-code, default value is

```
list("fancyvrb" = "true", "language" = "R", "escapechar" = "\\",
     "basicstyle" = "{\\color{Rcolor}\\footnotesize}",
     "keywordstyle" = "{\\bf\\color{Rcolor}}",
     "commentstyle" = "{\\color{Rcomment}\\ttfamily\\itshape}",
     "literate" = "{<-}{\\$\\leftarrow$}2{<<-}{\\$\\twoheadleftarrow$}2",
     "alsoother" = "{\\$}", "alsoletter" = "{.<-}"
     "otherkeywords" = "{!, !=, ~, $, *, \\&, \\%/\\%, \\%*\\%, \\%\\%, <-, <<-
, /}")
```

Rdset: default setting for \lstset-definition of Rd-code, default value is

```
list("fancyvrb" = "true", "language" = "Rd",
     "basicstyle" = "{\\color{black}\\tiny}",
     "keywordstyle" = "{\\bf}",
     "commentstyle" = "\\ttfamily\\itshape",
     "alsolanguage" = "R")
```

Rin: default setting for \lstdefinestyle-definition of Rinstyle (for R input), default value is

```
list("style" = "Rstyle", "fancyvrb" = "true",
     "basicstyle" = "\\color{Rcolor}\\small")
```

Rout: default setting for \lstdefinestyle-definition of Routstyle (for R output), default value is

```
list("fancyvrb" = "false", "basicstyle" = "\\color{Rout}\\small")
```

Rcode: default setting for \lstdefinestyle-definition of Rcodestyle (for R code), default value is

```
list("style" = "Rstyle", "fancyvrb" = "true",
     "basicstyle" = "\\color{Rcolor}\\small") "fontshape" = "sl", "basicstyle"
= "\\color{Rcolor}")
```

Rcolor: default setting for color of R-code in rgb-coordinates; defaults to c(0, 0.5, 0.5)

RRcomdcolor: default setting for color of R-code from recommended packages in rgb-coordinates; defaults to c(0, 0.6, 0.4)

- Rbcolor:** default setting for color of R-code symbols of intermediate packages in rgb-coordinates; defaults to `c(0, 0.6, 0.6)`
- Routcolor:** default setting for color of R-output in rgb-coordinates; defaults to `c(0.461, 0.039, 0.102)`
- Rcommentcolor:** default setting for color of R-comments in rgb-coordinates; defaults to `c(0.101, 0.043, 0.432)`
- pkv:** default setting for package version as character; defaults to `"2.0.2"`
- inSweave:** logical; default setting for masked functions `library` and `require`; shall they show Sweave-behaviour? defaults to `FALSE`
- Keywordstyle:** default setting for R-symbols as character; defaults to `"{\bf}"`
- interm.Keywordstyle:** default setting for R-symbols in an intermediate package as character; defaults to `"{\bf\color{Rbcolor}}"`
- Recomd.Keywordstyle:** default setting for R-symbols in package `base` or another recommended package as character; defaults to `"{\bf\color{Recomdcolor}}"`
- intermediate:** default setting: shall symbols from automatically loaded intermediate packages also be registered for listings-printing? defaults to `TRUE`
- overwrite:** default setting: shall R symbols already defined in the original R language definition in `'lstlang3.sty'` be overwritten if they reappear in a required package (to be printed in a different keywordstyle)? defaults to `FALSE`
- fromRForge:** default setting: logical: shall code snippets be drawn from `"r-forge.r-project.org"` (or a mirror) or not; defaults to `TRUE`.
- base.url:** default setting for the base url to download code snippets from (character); defaults to `"http://r-forge.r-project.org/plugins/scmsvn/viewcvs.php/*checkout*/pkg/"`
- addRset:** logical; default for argument `add` in command `Rset`; defaults to `TRUE`
- addRdset:** logical; default for argument `add` in command `Rdset`; defaults to `TRUE`
- addRinset:** logical; default for argument `add` in command `Rinset`; defaults to `TRUE`
- addRoutset:** logical; default for argument `add` in command `Routset`; defaults to `TRUE`
- addRcodeset:** logical; default for argument `add` in command `Rcodeset`; defaults to `TRUE`
- fileCommand:** character; the TeX code to define TeX command `\file`; defaults to `"\def\file#1{\tt #1}"`
- pkgCommand:** character; the TeX code to define TeX command `\pkg`; defaults to `"\def\pkg#1{\tt \"#1\"}"`

Author(s)

Peter Ruckdeschel <Peter.Ruckdeschel@itwm.fraunhofer.de>

See Also

[options](#), [getOption](#)

Examples

```

SweaveListingOptions()
SweaveListingOptions("Rout")
SweaveListingOptions("Rout" = c(0,0,0))
# or
SweaveListingOptions(Rout = c(0,1,0))
getSweaveListingOption("Rout")

```

```

SweaveListingPreparations
SweaveListingPreparations

```

Description

helping tool for writing the corresponding TeX preamble commands to integrate Sweave and package listings

Usage

```

SweaveListingPreparations (
  withOwnFileSection = FALSE,
  withVerbatim = FALSE,
  withSchunkDef = TRUE,
  gin = TRUE,
  ae = TRUE,
  LineLength = getOption("width"),
  Rset = getSweaveListingOption("Rset"),
  Rdset = getSweaveListingOption("Rdset"),
  Rin = getSweaveListingOption("Rin"),
  Rout = getSweaveListingOption("Rout"),
  Rcode = getSweaveListingOption("Rcode"),
  Rcolor = getSweaveListingOption("Rcolor"),
  RRecomdcolor = getSweaveListingOption("RRecomdcolor"),
  Rbcolor = getSweaveListingOption("Rbcolor"),
  Routcolor = getSweaveListingOption("Routcolor"),
  Rcommentcolor = getSweaveListingOption("Rcommentcolor"),
  pkg = getSweaveListingOption("pkg"),
  pkv = getSweaveListingOption("pkv"),
  fileCommand = getSweaveListingOption("fileCommand"),
  pkgCommand = getSweaveListingOption("pkgCommand"),
  lib.loc = NULL)

```

Arguments

`withOwnFileSection`

logical: Does one want to use an own definition file/ section to define `Sinput`, `Soutput`, `Scode` environments; if `TRUE` you should write some `\include` directive to include your own definitions / or write them in your `'\Rnw'` file.

<code>withVerbatim</code>	logical of length 3 (filled by recycling if of shorter length) either named ("Sinput", "Soutput", "Scode") or taken in order (Sinput, Soutput, Scode): Should we use Verbatim from TeX package 'fancyvrb' (i.e. the original " <code>\DefineVerbatimEnvironment{Si}</code> " by Fritz Leisch) or just TeX package 'listings' (i.e. we define our own Sinput environment based on 'listings' command 'lstnewenvironment'; this option is due to a suggestion by Andrew Ellis (thank you!)).
<code>withSchunkDef</code>	logical: shall environment Schunk be defined?
<code>gin</code>	logical: shall we use gin
<code>ae</code>	logical: shall we use ae
<code>LineLength</code>	numeric; defaults to 80
<code>Rset</code>	list or taglist; defaults to global option "Rset "
<code>Rdset</code>	list or taglist; defaults to global option "Rdset "
<code>Rin</code>	list or taglist; format string for R input code; defaults to global option "Rin "
<code>Rout</code>	list or taglist; format string for R output code; defaults to global option "Rout "
<code>Rcode</code>	list or taglist; format string for R code; defaults to global option "Rcode "
<code>Rcolor</code>	numeric of length 3; [color for R-input-code] defaults to global option "Rcolor"; rgb coordinates of the color in which to print R-code
<code>RRecomdcolor</code>	numeric of length 3; [color for R-symbols from recommended packages] defaults to global option "RRecomdcolor"; rgb coordinates of the color in which to print R-code
<code>Rbcolor</code>	numeric of length 3; [color for R-symbols from intermediate packages] defaults to global option "Rbcolor"; rgb coordinates of the color in which to print R keywords induced by intermediate packages.
<code>Routcolor</code>	numeric of length 3; [color for R-output-code] defaults to global option "Rout "; rgb coordinates of the color in which to print R output
<code>Rcommentcolor</code>	numeric of length 3; [color for R-comments] defaults to global option "Rcomment "; rgb coordinates of the color in which to print comments in R-code
<code>pkg</code>	character; name of the package to be described (e.g. in a vignette); defaults to global option "pkg"
<code>pkv</code>	character; package version to be described (e.g. in a vignette); defaults to global option "pkv"
<code>fileCommand</code>	character; the TeX code to define TeX command <code>\file</code>
<code>pkgCommand</code>	character; the TeX code to define TeX command <code>\pkg</code>
<code>lib.loc</code>	location of a local library in which the described package resides

Details

`SweaveListingPreparations` writes a corresponding preamble to the '.Rnw'-file for the simultaneous use of Sweave and package listings; note that so far, even if you do not want to use the default style file 'Sweave.sty', the lines

```
%\usepackage{Sweave}
\SweaveOpts{keep.source=TRUE}
```

still have to appear in the `.Rnw` file — before the corresponding `SweaveListingPreparations-` chunk.

The TeX code inserted by `SweaveListingPreparations` sets colors, requires TeX-package `'listings'`, defines listings-settings for `'R'`- and `'Rd'`-code. For `'Rd'` files it uses a new `'listings'`-language definition file, `'Rdlisting.sty'` to be found in subfolder `'TeX'` in the **SweaveListingUtils** package folder. `SweaveListingPreparations` is to be called in an `.Rnw` file and [re]defines `Sinput`, `Soutput`, `Scode` environments for use with TeX-package `'listings'`, defines commands `\code`, `\file`, `\pkg` and sets the corresponding package version.

The default values are taken from [SweaveListingOptions](#). The output to stdout can be captured in an `.Rnw` file as

```
<< lstPreamble, results=tex, echo=FALSE>>=
require(SweaveListingUtils)
SweaveListingPreparations()
@
```

to insert the corresponding preamble parts to the produced TeX file.

If you one want to use distinct keywordstyles for packages loaded by `require` or `library` without adding extra arguments to these commands (and hence displaying the actual R code), you have to set up a global formatting matrix `.tobeDefinedPkgs` by [setToBeDefinedPkgs](#).

Value

```
invisible()
```

Acknowledgement

The author wants to thank Frank E. Harrel and Andrew Ellis for very valuable suggestions to enhance this package.

Author(s)

```
Peter Ruckdeschel <Peter.Ruckdeschel@itwm.fraunhofer.de>
```

Examples

```
SweaveListingPreparations()
SweaveListingPreparations(pkv="2.1")
```

taglist

*S3 class taglist***Description**

Generating function and print method for S3 class taglist

Usage

```
taglist(..., list = NULL, defname = "v")
## S3 method for class 'taglist':
print(x, LineLength = getOption("width"), offset.start = 0,
      withFinalLineBreak = TRUE, first.print = NULL,
      ErrorOrWarn = "warn", ...)
```

Arguments

...	arbitrary number of arguments in taglist, respectively, in print.taglist, further arguments to be passed to or from other methods. They are ignored in this function.
list	list; to be prepended to the ... argument; useful to take in predefined default lists;
defname	character; defaults to "v"; prefix of names given to a priori unnamed list elements; see below
x	an object of class taglist
LineLength	numeric; defaults to getOption("width"); number of characters per line; i.e.; print will make a linebreak <i>between</i> two tags before this number is reached; it will not break a tag, however, and only issue an error if this single tag is longer than <code>max(3*LineLength, getOption("width"))</code> .
offset.start	numeric; number of characters to indent in printing;
withFinalLineBreak	logical; shall there be a final line break; defaults to TRUE.
first.print	character; something to be printed immediately before the first list item; defaulting to NULL.
ErrorOrWarn	shall we issue a warning (if partially matched to "warn" or else an error

Details

taglist is an S3 class inheriting from class list. It requires all elements to be uniquely named. Objects of this class can be generated by a call to the generating function taglist(); there is a particular print method for this class. The generating function taglist() accepts an arbitrary number of (not necessarily named) arguments and a list of (again not necessarily named) elements; if in the arguments of taglist() some list items do not have a name a priori, all of these get named in the order of appearance as `<defname><position number in arglist>`.

Value

```
taglist      an object of S3 class taglist
print.default
invisible()
```

Author(s)

Peter Ruckdeschel <Peter.Ruckdeschel@itwm.fraunhofer.de>

Examples

```
TL <- taglist("HA"=8,"JUI"=7,"butzi", list=list("HU"="AHAL","HA"="BETA","BUZ"))
print(TL)
print(TL, LineLength=10, first.print="myList=", offset.start=4,
      withFinalLineBreak = FALSE)
```

Index

*Topic **documentation**

SweaveListingMASK, 20

*Topic **misc**

SweaveListingOptions, 20

*Topic **package**

SweaveListingUtils-package, 2

*Topic **programming**

SweaveListingMASK, 20

*Topic **utilities**

changeKeywordstyles, 4

copySourceFromRForge, 6

library, 7

lstinputSourceFromRForge, 11

lstset, 13

lstsetLanguage, 15

readPkgVersion, 17

readSourceFromRForge, 18

setToBeDefinedPkgs, 19

SweaveListingOptions, 20

SweaveListingPreparations, 23

SweaveListingUtils-package, 2

taglist, 26

addRdset (*SweaveListingOptions*),
20

addRset (*SweaveListingOptions*), 20

base.url (*SweaveListingOptions*),
20

changeKeywordstyles, 3, 4, 16

conflicts, 8

copySourceFromRForge, 6, 11, 12

detach, 8

fileCommand
(*SweaveListingOptions*), 20

fromRForge
(*SweaveListingOptions*), 20

getOption, 22

getSweaveListingOption, 8, 9

getSweaveListingOption
(*SweaveListingOptions*), 20

getSweaveListingOption (Rdset), 14

getSweaveListingOption (Rset), 14

inSweave (*SweaveListingOptions*),
20

interm.Keywordstyle
(*SweaveListingOptions*), 20

intermediate
(*SweaveListingOptions*), 20

Keywordstyle
(*SweaveListingOptions*), 20

library, 3, 7, 9, 10, 19

lstdefRstyle (*lstset*), 13

lstinputSourceFromRForge, 3, 11

lstset, 13

lstsetLanguage, 3, 5, 10, 15

lstsetR (*lstset*), 13

lstsetRall (*lstset*), 13

lstsetRcode (*lstset*), 13

lstsetRd (*lstset*), 13

lstsetRin (*lstset*), 13

lstsetRout (*lstset*), 13

MASKING (*SweaveListingMASK*), 20

name, 8

options, 8, 22

overwrite (*SweaveListingOptions*),
20

pkg (*SweaveListingOptions*), 20

pkgCommand
(*SweaveListingOptions*), 20

pkv (*SweaveListingOptions*), 20

`print.taglist(taglist)`, 26

`Rbcolor(SweaveListingOptions)`, 20

`Rcode(SweaveListingOptions)`, 20

`Rcolor(SweaveListingOptions)`, 20

`Rcommentcolor`
(*SweaveListingOptions*), 20

`Rdset(SweaveListingOptions)`, 20

`readPkgVersion`, 17

`readSourceFromRForge`, 18

`Recomd.Keywordstyle`
(*SweaveListingOptions*), 20

regular expressions, 7

`require`, 3, 9, 19

`require(library)`, 7

`Rin(SweaveListingOptions)`, 20

`Rout(SweaveListingOptions)`, 20

`Routcolor(SweaveListingOptions)`,
20

`RRecomdcolor`
(*SweaveListingOptions*), 20

`Rset(SweaveListingOptions)`, 20

search, 8

`setToBeDefinedPkgs`, 3, 9, 19, 25

`SweaveListingMASK`, 20

`SweaveListingOptions`, 20, 25

`SweaveListingoptions`, 14

`SweaveListingoptions`
(*SweaveListingOptions*), 20

`SweaveListingPreparations`, 2, 23

`SweaveListingUtils`
(*SweaveListingUtils-package*),
2

`SweaveListingUtils-package`, 2

`taglist`, 26