Package 'Rd'

May 23, 2019

Type Package

Title Rd Manipulation and Creation

Version 0.2.0

Maintainer Andrew Redd <andrew.redd@hsc.utah.edu>

License GPL-2

Description Creation and manipulation of R documentation (Rd) as R objects. Also contains functions for checking consistency of Rd and S4 generics and methods for converting objects to Rd formatted output.

Imports assertthat, methods, pkgcond, postlogic, purrr, rlang, testthat, tools, utils

Suggests covr, htmltools, rmarkdown (>= 1.8), stringi

Encoding UTF-8

RoxygenNote 6.1.1

Language en-US

Collate 'setup-set_old_classes.R' 'util-aliases.R' 'Class-Rd.R' 'Fun-toRd.R' 'Rd_lines.R' 'S3_extensions.R' 'canon.R' 'construction.R' 'convenience.R' 'globals.R' 'indent.R' 'keyword.db.R' 'testextra.R' 'util-Rd.R'

URL https://github.com/RDocTaskForce/Rd

BugReports https://github.com/RDocTaskForce/Rd/issues

NeedsCompilation no

Author Andrew Redd [aut, cre] (<https://orcid.org/000-0002-6149-2438>), R Documentation Task Force [ctb] (https://rdoctaskforce.github.io/), R Consortium [fnd, cph] (https://www.r-consortium.org)

Repository CRAN

Date/Publication 2019-05-23 04:10:27 UTC

R topics documented:

`aliases` *Internal Utilities These utilities are used internally and not exported. They are however documented for completeness*

Description

Internal Utilities

These utilities are used internally and not exported. They are however documented for completeness

Usage

```
s(x, ...)

cl(x, new)

undim(x)

named(...)

clean_Rd(obj, ...)

get_attr(x, which, default = NULL, exact = TRUE)

forward_attributes(x, obj)

fwd(x, obj)
```

```
is_whitespace(x)
```

Arguments

`x, obj`	An object, any object.
`...`	passed on to other function(s).
`new`	the new class(es) to append.
`which`	name of the attribute to extract.
`default`	the default value to return if not found.
`exact`	exact or partial matching?

Functions

- `s`: Alias for structure, but also adds automatic naming when unnamed,
- `cl`: Specify an additional class for an object, or class when none is set.
- `undim`: Remove the `dim` attribute.
- `named`: Create a named list with names inferred if needed.
- `clean_Rd`: Alias for `tools::toRd.default()`
- `get_attr`: Alias for `attr(x, which) %||% default`.
- `forward_attributes`: Forward attributes from object to value.
- `fwd`: Alias for `forward_attributes()`.
- `is_whitespace`: Check if a string is composed of only whitespace, with regex pattern `"^\s+$"`.

See Also

`purrr::%||%()`

`base::attributes()`

`is_Rd_newline` *Check if an element is a newline*

Description

Check if an element is a newline

Usage

```
is_Rd_newline(x)
```

Arguments

`x`	object to check

pieces *Rd Pieces*

Description

These objects are provided to help construct Rd documents.

Usage

```
Rd.newline
Rd.code.newline
Rd.break
```

Rd *Construct an Rd container*

Description

An Rd container is a list which contains other Rd objects.

Usage

```
Rd(..., content = list(...), .check = NA,
  verbose = getOption("Rd::verbose", getOption("verbose", FALSE)))
```

Arguments

`..., content`	Content specified in individual or list form. Either may be used but only one. All elements should be unnamed.
`.check`	Should the content be checked for valid Rd and if options are valid? A value of FALSE indicates no checking, TRUE strict checking and NA convert where possible, with messages and warnings.
`verbose`	Print informational messages.

Details

An empty Rd vector can be created with a `Rd()` call. A call to Rd with only one argument will not create a double nested list, but will encapsulate tags and strings. For example, calling `Rd(Rd('test'))` has the same effect as `Rd('test')`

A character vector may be passed to Rd where it is collapsed and normalized to a `Rd` container of Rd string.

Value

Will always return valid Rd in canonical form.

See Also

Other construction: Rd_string_creation, Rd_tag

Examples

```
## Recreating the first few lines of the `example` help file.
R <- Rd_tag("\\R")
Rd( Rd_name('example'), '\n'
  , Rd_alias('example'), '\n'
  , Rd_title("Run an Examples Section from the Online Help"), '\n'
  , Rd_description( "Run all ", R, " code from the "
                  , Rd_tag("\\bold", "Examples"), " part of \n"
                  , R, "'s online help topic ", Rd_code("topic")
                  , " with possible exceptions \n"
                  , Rd_code("dontrun"), ", "
                  , Rd_code("dontshow"), ", and "
                  , Rd_code("donttest"), ",\n see "
                  , Rd_tag("\\sQuote", "Details"), " below."
                  )
  )
```

Rd-extraction — *Extract elements from Rd containers These functions can be used to extract or subset elements of Rd lists, either bare* Rd *containers or* Rd_tag *objects. The* [[*operator performs classification of output.* Rd_get_element *acts as* [[*but will also accept tags to search for. The* [*operator wraps* Rd_subset*, respectively.*

Description

Extract elements from Rd containers

These functions can be used to extract or subset elements of Rd lists, either bare Rd containers or Rd_tag objects. The [[operator performs classification of output. Rd_get_element acts as [[but will also accept tags to search for. The [operator wraps Rd_subset, respectively.

Usage

```
Rd_get_element(x, ..., drop = TRUE)

## S3 method for class 'Rd_tag'
x[[...]]

## S3 method for class 'Rd'
x[[...]]

Rd_subset(x, i, ..., drop = FALSE)
```

```
## S3 method for class 'Rd'
x[...]

## S3 method for class 'Rd_tag'
x[...]
```

Arguments

x	Either an Rd or Rd_tag object.
...	Must be empty.
drop	For matrices and arrays. If TRUE the result is coerced to the lowest possible dimension (see the examples). This only works for extracting elements, not for the replacement. See drop for further details.
i	indices specifying elements to extract or replace. Indices are numeric or character vectors or empty (missing) or NULL. Numeric values are coerced to integer as by as.integer (and hence truncated towards zero). Character vectors will be matched to the names of the object (or for matrices/arrays, the dimnames): see 'Character indices' below for further details. For [-indexing only: i, j, ... can be logical vectors, indicating elements/slices to select. Such vectors are recycled if necessary to match the corresponding extent. i, j, ... can also be negative integers, indicating elements/slices to leave out of the selection. When indexing arrays by [a single argument i can be a matrix with as many columns as there are dimensions of x; the result is then a vector with elements corresponding to the sets of indices in each row of i. An index value of NULL is treated as if it were integer(0).

Value

- Rd_get_element **and** [[**:** the return value is not necessarily an Rd. The class of the element returned will inferred if it does not have a class. Lists will be classed as Rd unless the Rd_tag attribute is set in which case a Rd_tag will be returned
- Rd_subset **and** [**:** will always return a value of the same class as is given. Even if what is returned is an empty list, an empty list with the Rd_tag attribute set is valid, and denotes a tag without any content such as the "\item" tag.

See Also

base::Extract

Rd_c *Combine Rd elements*

Description

Combine Rd elements

Usage

```
Rd_c(...)
```

Arguments

`...`	Rd elements.

Note

The special considerations for Rd elements necessitate a special method for combining the traditional generic c cannot be used due to problems that occur, particularly in the RStudio GUI.

`Rd_canonize` *Rd Canonical Form*

Description

Canonical form is simply described as that which would come out from reading an Rd file via, `tools::parse_Rd()`.

Usage

```
Rd_canonize(rd, ..., .check = TRUE)

Rd_canonize_text(rd, .check = TRUE, ...)

Rd_canonize_code(rd, .check = TRUE, ...)
```

Arguments

`rd`	the Rd container object to put in canonical form.
`...`	Arguments passed on to `is_valid_Rd_object` **x** object to test **strict** if the class must be set. A value of NA indicates that the first level need not be classed but all subsequent must be. **tags** the type of tag(s) allowed in the `Rd_tag` attribute. **deep** should contained elements also be checked for validity?
`.check`	Perform input checks?

Details

Canonical Rd Text has:

- One line per element, with `attr(., 'Rd_tag')=='TEXT'`
- The indents are merged with content if the first content is text.
- Newlines are contained with the content provided the content is 'TEXT', but the newline must be the last character in the string and cannot appear anywhere else.
- Comments are a separate class and do not include the newline.

Canonical R code follows the following rules:

- One element per line of code.
- newline is included at the end of the line string, not as a separate element.
- if there are multiple lines they are bound together in an Rd or Rd_tag list.

Functions

- `Rd_canonize_text`: Put text in canonical form.
- `Rd_canonize_code`: Put R code in canonical form.

Examples

```
## Rd_c does not guarantee canonical code.
x <- Rd_c(Rd('Testing'), Rd('\n'))
str(x)
str(Rd_canonize(x))
```

Rd_clean_indent *Check and clean an indent string*

Description

Check and clean an indent string

Usage

```
Rd_clean_indent(indent.with, type = get_Rd_tag(indent.with))
```

Arguments

`indent.with`	The string to use for indentation.
`type`	The type of string the indent is to be.

Details

Indents must have the following characteristics:

- They must be all whitespace, i.e. spaces.
- Newlines are not allowed.
- Tabs, are allowed with a warning as they violate the guidelines.
- May only be Rd_strings of tag type `TEXT` or `RCODE`. Code type should only be used to indent other text.

Value

An Rd_string, of type 'TEXT' or 'CODE', see details, wrapped in an Rd container.

Note

When put in canonical form the indents element will often be merged with other elements.

See Also

`https://developer.r-project.org/Rds.html`

`Rd_compact` *Compact a list into an Rd vector*

Description

When creating Rd from other objects it is common to iterate over a vector or list and convert each to Rd then combine the results. However `toRd()` can return an Rd_string, Rd_tag, or an Rd container. to combine these together intelligently, use `Rd_compact` which converts each to an Rd container, placing strings and tags in a container, then concatenating all together.

Usage

```
Rd_compact(l)
```

Arguments

`l`	list of Rd objects.

Rd_indent *Indent Rd*

Description

Indent Rd

Usage

```
Rd_indent(rd, indent.with = getOption("Rd::indent.with", "  "),
  recursive = TRUE, ..., no.first = is(rd, "Rd_tag"), .check = TRUE)
```

Arguments

rd	an Rd container or an Rd tag.
indent.with	What to indent with. See Rd_clean_indent().
recursive	Indent recursively?
...	Ignored but included for forward compatibility and to force full names on subsequent parameters.
no.first	if the first element should be indented.
.check	check for valid Rd?

Examples

```
(x <- Rd_description("line 1\n", "line 2\n"))
Rd_indent(x)
```

Rd_lines *Combine with line breaks*

Description

Combine a list of Rd objects with a line break between each.

Usage

```
Rd_lines(l, ...)
```

Arguments

l	A list of valid Rd objects.
...	Arguments passed on to Rd_canonize **rd** the Rd container object to put in canonical form. **.check** Perform input checks?

Rd_split *Split a rd object into relevant lines.*

Description

Split a rd object into relevant lines.

Usage

```
Rd_split(rd)
```

Arguments

rd	an rd list.

Rd_string_creation *Rd String Construction*

Description

Construct a Rd string type.

Usage

```
Rd_string(content, type = c("TEXT", "RCODE", "VERB", "COMMENT",
  "UNKNOWN", "LIST"))

Rd_text(content)

Rd_rcode(content)

Rd_verb(content)

Rd_symb(content)

Rd_comment(content)
```

Arguments

content	a string to type as an Rd string type
type	the type of text is is.

Functions

- `Rd_text`: Type as plain text.
- `Rd_rcode`: Type as R Code.
- `Rd_verb`: Type as R symbol or 'verb'.
- `Rd_symb`: Type as R symbol or 'verb'.
- `Rd_comment`: type as Rd comment. Comments must start with the comment character '

See Also

Other construction: `Rd_tag`, `Rd`

Examples

```
## Plain text
Rd_text("Plain text")
## R Code
Rd_rcode("code()")
## Symbols, i.e. verbs
Rd_verb("verb")
Rd_symb("symbol")
## Rd Comments
Rd_comment("% Rd/LaTeX comment")
```

`Rd_tag` *Create an Rd tag container*

Description

Rd_tag containers can contain, strings, Rd containers and other tags.

Usage

```
Rd_tag(tag, ..., content = Rd(...), opt = NULL,
  indent = getOption("Rd::indent", FALSE),
  indent.with = getOption("Rd::indent.with", "  "), .check = NA,
  verbose = getOption("Rd::verbose", getOption("verbose", FALSE)))
```

Arguments

`tag`	The Rd/LaTeX tag to use. Must start with the backslash.
`..., content`	The content specified in individual form or list form. Either may be used but not both.
`opt`	Options for the tag which as placed in square brackets.
`indent`	Should content inside the tag be indented?
`indent.with`	Amount to indent with, defaults to four spaces.

.check	Should the content be checked for valid Rd and if options are valid? A value of FALSE indicates no checking, TRUE strict checking and NA convert where possible, with messages and warnings.
verbose	Print informational messages.

See Also

Other construction: Rd_string_creation, Rd

Examples

```
Rd_tag("\\bold", Rd_text("Bolded text."))
```

---

shortcuts *Convenience Construction Shortcuts.*

Description

These functions are provided to construct Rd structure. In some cases additional compliance checks are included.

Usage

```
Rd_alias(alias)

Rd_aliases(aliases)

Rd_author(author)

Rd_arguments(..., items = list(...), indent = getOption("Rd::indent",
  TRUE), indent.with = getOption("Rd::indent.with", "  "))

Rd_code(code)

Rd_concept(concept)

Rd_concepts(concepts)

Rd_description(..., content = Rd(...))

Rd_enumerate(..., items = list(...), indent = getOption("Rd::indent",
  TRUE), indent.with = getOption("Rd::indent.with", "  "))

Rd_examples(..., content = list(...))

Rd_item(item, description = NULL)
```

```
Rd_itemize(..., items = list(...), indent = getOption("Rd::indent",
  TRUE), indent.with = getOption("Rd::indent.with", "  "))

Rd_keyword(key, .check = TRUE)

Rd_keywords(keys, .check = TRUE)

Rd_name(name)

Rd_title(title)

Rd_usage(..., usages = list(...))

Rd_value(value)
```

Arguments

alias	an alias as a plain string.
aliases	a character vector, each element a separate alias.
author	a string, a person object, or another Rd object.
..., content	Rd elements to be contained in the tag.
items	arguments each should be an 'item' tag, newlines to separate items are added automatically.
indent	indent content
indent.with	string to use for indent.
code	a string to be converted to RCODE then wrapped in the 'code' tag.
concept	the concept name, as a plain string.
concepts	a vector of concepts each to be put in a concept tag.
item	the item text
description	an optional description that if provided changes the 'item' tag into a two parameter, rather than a single item tag followed by the item text.
key	A string denoting a valid Rd keyword.
.check	perform validity checks?
keys	A character vector denoting valid Rd keywords.
name	A string for a name of the Rd document.
title	A string giving the title.
usages	lines of usage, all should be bare strings or RCODE strings.
value	The return value, must be a correctly formatted Rd object.

Functions

- Rd_alias: Create an alias tag.
- Rd_aliases: Create multiple aliases
- Rd_author: Create author tags
- Rd_arguments: create an arguments tag
- Rd_code: Create a code tag.
- Rd_concept: Create a 'concept' tag.
- Rd_concepts: Create multiple concepts.
- Rd_description: Create a description tag.
- Rd_enumerate: Create an enumerated list.
- Rd_examples: Create an examples tag.
- Rd_item: Create an item tag.
- Rd_itemize: Create an itemized list.
- Rd_keyword: Create a keyword tag.
- Rd_keywords: Create multiple keyword tags.
- Rd_name: Create a name tag.
- Rd_title: Create a title tag.
- Rd_usage: Create a usage tag.
- Rd_value: Create a value tag section.

Examples

```
Rd_enumerate(Rd_item("first"), Rd_item("second"))
Rd_enumerate( Rd_item("first", "comes before second.")
            , Rd_item("second", "comes after first."))
Rd_item("an item")
Rd_item('a', 'the first letter of the alphabet.')
Rd_itemize(Rd_item("first"), Rd_item("second"))
Rd_itemize( Rd_item("first", "comes before second.")
          , Rd_item("second", "comes after first."))
Rd_keyword('documentation')
```

---

testing-Rd *Testing Rd types These provide methods for testing if an object is valid.*

Description

Testing Rd types

These provide methods for testing if an object is valid.

Usage

```
is_Rd_string(x, tags = NULL, strict = FALSE, reason = TRUE)

are_Rd_strings(x, tags = NULL, strict = FALSE)

is_Rd_tag(x, tags = NULL, strict = FALSE, reason = TRUE)

are_Rd_tags(x, tags = NULL, strict = FALSE)

is_Rd(x, strict = FALSE)

is_valid_Rd_list(x, tags = NULL, strict = FALSE,
  deep = !isTRUE(strict) || !missing(tags))

is_valid_Rd_object(x, tags = NULL, strict = FALSE,
  deep = !isTRUE(strict) || !missing(tags))
```

Arguments

x	object to test
tags	the type of tag(s) allowed in the Rd_tag attribute.
strict	if the class must be set. A value of NA indicates that the first level need not be classed but all subsequent must be.
reason	should the reason for failure be included. Used in assertthat::assert_that() and related functions.
deep	should contained elements also be checked for validity?

Functions

- is_Rd_string: test if object is a valid Rd string type.
- are_Rd_strings: Vector version of is_Rd_string.
- is_Rd_tag: Check if a list is a valid Rd_tag object.
- are_Rd_tags: Vector version of is_Rd_tag
- is_Rd: check if a list is an Rd container object.
- is_valid_Rd_list: Check that an object is a valid Rd list, an Rd_tag or Rd, but not an Rd_string
- is_valid_Rd_object: Check that an object is valid

Examples

```
is_Rd_string('nope')
is_Rd_string(structure('Valid but not strict', Rd_tag='TEXT'))
is_Rd_string(structure('Valid but not strict', Rd_tag='TEXT'), strict=TRUE)
is_Rd_string(Rd_text('Valid and strict'), strict=TRUE)
are_Rd_strings(Rd( Rd_alias('example'), '\n'
                 , Rd_name('example'), '\n'
```

```
                ))
is_Rd_tag(Rd('text'))
is_Rd_tag(Rd_alias('alias'))
are_Rd_tags(Rd( Rd_alias('example'), '\n'
              , Rd_name('example'), '\n'
              ))
is_Rd(list())
is_Rd(list(), strict=TRUE)
is_Rd(Rd(), strict=TRUE)
is_valid_Rd_list(Rd_name('name'))
is_valid_Rd_list(Rd('text'))
is_valid_Rd_list(Rd_text('text'))
is_valid_Rd_object(Rd_name('name'))
is_valid_Rd_object(Rd('text'))
is_valid_Rd_object(Rd_text('text'))
```

---

toRd *Convert an object to Rd*

Description

This provides the generic for converting objects to Rd. It extends the core function toRd.

Usage

```
toRd(obj, ...)

\S3method{toRd}{NULL}(obj, ...)

\S3method{toRd}{list}(obj, ..., unnest=NA)

\S3method{toRd}{Rd_string}(obj, ...)

\S3method{toRd}{Rd_tag}(obj, ...)

\S3method{toRd}{Rd}(obj, ...)

\S3method{toRd}{person}(obj, ..., include = c('given', 'family', 'email'))

\S3method{toRd}{name}(obj, ...)
```

Arguments

obj	object to convert
...	passed on to methods
unnest	Should the results be unlisted to remove nesting? A value of FALSE indicates that nesting should never be removed, TRUE implies always remove nested elements with class Rd, and when NA, the default, nesting will be removed only if all elements are Rd.

include The parts of the person object to include.

Examples

```
str(toRd(NULL))
toRd(person('John' , 'Doe', email="john@email.com"))
toRd(c( person('John' , 'Doe', email="john@email.com")
      , person('Jane' , 'Poe', email="jane@email.com")
      ))
```

Index