Package 'sig'

October 14, 2022

Type Package

Title Print Function Signatures

Version 0.0-6

Date 2022-04-19

Author Richard Cotton [aut, cre]

Maintainer Richard Cotton <richierocks@gmail.com>

Description Print function signatures and find overly complicated code.

URL https://bitbucket.org/richierocks/sig

BugReports https://bitbucket.org/richierocks/sig/issues

Depends R (>= 2.15.0)

License Unlimited

Acknowledgments Development of this package was partially funded by the Proteomics Core at Weill Cornell Medical College in Qatar <http://qatar-weill.cornell.edu>. The Core is supported by 'Biomedical Research Program' funds, a program funded by Qatar Foundation.

Collate 'sig.R' 'list_sigs.R' 'as.R' 'is.R' 'sig_report.R' 'indexing.R' 'utils.R' 'write_sigs.R'

RoxygenNote 7.1.2

NeedsCompilation no

Repository CRAN

Date/Publication 2022-04-21 12:50:02 UTC

R topics documented:

`as.list.sig` *Convert to list*

Description

Strips class attributes to return a list.

Usage

```
## S3 method for class 'sig'
as.list(x, ...)

## S3 method for class 'siglist'
as.list(x, ...)

## S3 method for class 'sigreport'
as.list(x, ...)
```

Arguments

`x`	`sig`, `siglist` or `sigreport` object.
`...`	Passed from other `as.list` methods.

Value

A list.

Examples

```
as.list(sig(read.csv))
head(as.list(list_sigs(pkg2env(stats))))
as.list(sig_report(baseenv()))
```

`as.sig` *Coerce object to be a sig*

Description

Coerces an object to be a `sig`.

Usage

```
as.sig(x, ...)

## Default S3 method:
as.sig(x, ...)

## S3 method for class 'siglist'
as.sig(x, ...)

## S3 method for class 'list'
as.sig(x, ...)

## S3 method for class 'sig'
as.sig(x, ...)
```

Arguments

`x`	Object to coerce.
`...`	Passed to other `as.sig` methods.

Value

An object of class `sig`.

See Also

`as.siglist`

Examples

```
as.sig(
  list(name = "fun", alist(x =,y = 1))
)
```

`as.siglist` *Coerce object to be a siglist*

Description

Coerces an object to be a `siglist`.

Usage

```
as.siglist(x, ...)

## S3 method for class 'sig'
as.siglist(x, ...)

## S3 method for class 'list'
as.siglist(x, ...)

## S3 method for class 'siglist'
as.siglist(x, ...)
```

Arguments

x	Object to coerce.
...	Passed to other `as.siglist` methods.

Value

An object of class `siglist`.

See Also

`as.sig`

Examples

```
as.siglist(list(
  sig(mean),
  list(name = "fun", alist(x =,y = 1))
))
```

backquote *Wrap in backquotes*

Description

Wraps strings in backquotes.

Usage

```
backquote(x)
```

Arguments

x	A character vector.

Value

A character vector.

Note

Existing backquote characters are escaped with a backslash.

See Also

sQuote

Examples

```
## Not run:
backquote(c("foo bar", "a`b`c"))

## End(Not run)
```

exponential_cut *Cut with exponential breaks*

Description

Wrapper to cut for positive integers.

Usage

```
exponential_cut(x)
```

Arguments

x	A vector of positive integers.

Value

A factor.

Note

The breaks are 1, 2, 3 to 4, 5 to 8, etc. No input checking is done; use at your peril.

See Also

cut

Examples

```
## Not run:
exponential_cut(c(1:10, 500))

## End(Not run)
```

---

fix_fn_names *Fix names for sigs*

Description

Make anonymous functions and special functions safe.

Usage

```
fix_fn_names(fn_name)
```

Arguments

fn_name	A character vector.

Value

A character vector.

Note

Strings beginning with "function" are given the value "..anonymous..".

Special function names are wrapped in backquotes.

Examples

```
## Not run:
fix_fn_names(c("%foo%", "?", "foo bar", "repeat", "function"))

## End(Not run)
```

`is.sig` *Is the input a sig?*

Description

Does the input inherit from "sig"?

Usage

```
is.sig(x)
```

Arguments

`x`	Object to test.

Value

`TRUE` if the object inherits from class "sig", and FALSE otherwise.

Examples

```
stopifnot(
  is.sig(sig(with)),
  !is.sig(with)     #functions are not their signatures.
)
```

`is.siglist` *Is the input a siglist?*

Description

Does the input inherit from "siglist"?

Usage

```
is.siglist(x)
```

Arguments

`x`	Object to test.

Value

TRUE if the object inherits from class "siglist" and is.sig returns TRUE for each element of the input, and FALSE otherwise.

Examples

```
stopifnot(
  !is.siglist(sig(with))     #1 sig is not a siglist.
)
```

---

list_sigs *List the signatures of all functions*

Description

Lists the signatures of all functions in an environment or file.

Usage

```
list_sigs(x, pattern = NULL, ...)

## Default S3 method:
list_sigs(x, pattern = NULL, ...)

## S3 method for class 'sig'
list_sigs(x, pattern = NULL, ...)

## S3 method for class 'character'
list_sigs(x, pattern = NULL, ...)
```

Arguments

x	An environment or the the path to a file.
pattern	An optional regular expression. Only names matching pattern are returned.
...	Currently ignored

Value

An object of class siglist, which is a list of sig objects.

Examples

```
#From a package
list_sigs(pkg2env(graphics))
#Just functions beginning with 'a'.
list_sigs(pkg2env(graphics), pattern = "^a")
#From a file
list_sigs(system.file("extdata", "sample.R", package = "sig"))
```

pkg2env *Get environment of a package.*

Description

Utility function to get the environment of a package on the search path.

Usage

```
pkg2env(pkg)

pkg2env_(pkg)
```

Arguments

pkg	A package.

Value

the environment corresponding to pkg.

See Also

list2env

Examples

```
# Non-standard evaluation version
pkg2env(graphics)

# Standard evaluations versions
pkg2env_("tools")
pkg2env_(~ utils)
pkg2env_(quote(stats))
```

print_engine *Workhorse of the print methods*

Description

Wraps toString methods with cat.

Usage

```
print_engine(x, ...)
```

Arguments

x	Object to print
...	Passed to toString.

Value

The input is invisibly returned, but the function is mostly invoked for the side effect of printing the object.

Note

Not intended for general consumption. This function is only exported because of package build requirements.

---

sig *Generate a function signature object*

Description

Generates a signature object for a function.

Usage

```
sig(fn, name_override, ...)

## Default S3 method:
sig(fn, ...)

## S3 method for class 'character'
sig(fn, ...)

## S3 method for class 'call'
sig(fn, ...)

## S3 method for class 'formula'
sig(fn, ...)

## S3 method for class 'name'
sig(fn, ...)
```

Arguments

fn	A function.
name_override	Override the default function name. See examples.
...	For possible additional future arguments, currently unused.

Value

A list, with the elements

- nameThe name of the function.
- argsThe arguments of the function.

Note

Anonymous functions are given the name "..anonymous..".

Nonstandard names ("foo bar"), assignment fns ("foo<-"), operators (" in backquotes.

Examples

```
sig(R.Version)               #no args
sig(scan)                    #lots of args
sig(function(x, y) {x + y})  #anonymous
sig(sum)                     #primitive

sig("sd")                   #string input
sig("function(x, y) {x + y}")
sig(~ prod)                 #formula input
sig(~ function(x, y) {x + y})
sig(quote(paste0))          #name input
sig(quote(function(x, y) {x + y}))

fn_list <- list(
  mean = mean,
  var = var
)
lapply(fn_list, sig)         #names are a mess
Map(                         #use Map for lists
  sig,
  fn_list,
  names(fn_list)             #Map mangles names, so override
)
```

---

sig_report *Summarise function complexity of a file or environment*

Description

Summarise function complexity of a file or environment

Usage

```
sig_report(x, ...)

## Default S3 method:
sig_report(x, ...)

## S3 method for class 'environment'
sig_report(
  x,
  too_many_args = 10,
  too_many_lines = 50,
  length_metric = c("deparse", "body"),
  ...
)

## S3 method for class 'character'
sig_report(x, ...)

## S3 method for class 'sigreport'
print(x, ...)
```

Arguments

x	A path to an R file or an environment.
...	Passed to sig_report.environment.
too_many_args	Upper bound for a sensible number of args.
too_many_lines	Upper bound for a sensible number of lines.
length_metric	Either "deparse" or "body". See note.

Details

sig_report summarises the number of input arguments and the number of lines of each function in an environment of file, and identifies problem files, in order to help you refactor your code. If the input is a path to an R file, then that file is sourced into a new environment and and the report is generated from that. The number of lines of code that a function takes up is subjective in R; this function gives you a choice of length(deparse(fn)) or length(body(fn)), depending upon the value of length_metric. The body metric tends to give smaller values than deparse, so you may want to reduce the too_many_lines argument.

Value

An object of class "sigreport" with the elements:

- n_varsNumber of variables.
- n_fnsNumber of functions.
- n_argsTable of the number of args of each function.
- too_many_argsUpper bound for a sensible number of args.

- fns_with_many_argsNames of each function with more args than `too_many_args`.
- n_linesTable of the number of lines of each function body.
- too_many_linesUpper bound for a sensible number of lines.
- long_fnsNames of each function with more lines than `too_many_lines`.

Examples

```
#Summarize function complexity in an environment
sig_report(pkg2env(stats))
#Summarize function complexity in a file

# From a file
tmp <- tempfile(fileext = ".R")
dump("scan", tmp)
sig_report(tmp)

# From an environment, adjusting the cutoff for reporting
sig_report(
  baseenv(),
  too_many_args  = 20,
  too_many_lines = 100
)
# Alternate length metric
sig_report(baseenv(), length_metric = "body")
```

`source_to_new_env` *Source a file into a new environment.*

Description

Silently sources a file into a new environment, returning that environment.

Usage

```
source_to_new_env(file, encoding = getOption("encoding"))
```

Arguments

`file`	a file to source.
`encoding`	character encoding of that file.

Value

An environment containing the sourced variables.

toString.sig *Print a sig object*

Description

Prints a function signature object.

Usage

```
## S3 method for class 'sig'
toString(x, width = getOption("width"), exdent = nchar(x$name), ...)

## S3 method for class 'sig'
print(x, width = getOption("width"), exdent = nchar(x$name), ...)
```

Arguments

x	An object of class sig.
width	Width of string to display.
exdent	Non-negative integer specifying the indentation of subsequent lines in the string.
...	Passed to toString

Value

toString creates a string representation of a function signature. print is mostly invoked for the side effect of printing a function signature, invisibly returning its input.

Examples

```
print_default_sig <- sig(print.default)
print(print_default_sig)
print(print_default_sig, width = 40)
print(print_default_sig, width = 40, exdent = 2)
toString(print_default_sig)
```

toString.siglist *Print a siglist object*

Description

Prints a list of function signature objects.

Usage

```
## S3 method for class 'siglist'
toString(x, width = getOption("width"), ...)

## S3 method for class 'siglist'
print(x, width = getOption("width"), ...)
```

Arguments

x	An object of class siglist.
width	Width of string to display.
...	Passed to the equivalent sig method.

Value

toString creates a string representation of a function signature. print is mostly invoked for the side effect of printing a function signature, invisibly returning its input.

Examples

```
method_sigs <- list_sigs(pkg2env(methods))
print(method_sigs)
print(method_sigs, width = 40)
print(method_sigs, width = 40, exdent = 2)
toString(method_sigs)
```

---

write_sigs *Write sigs to file*

Description

Writes a list of function signatures to a file.

Usage

```
write_sigs(x, file = stdout(), ...)

## Default S3 method:
write_sigs(x, file = stdout(), ...)

## S3 method for class 'siglist'
write_sigs(x, file = stdout(), ...)

## S3 method for class 'list'
write_sigs(x, file = stdout(), ...)

## S3 method for class 'environment'
```

```
write_sigs(x, file = stdout(), ...)

## S3 method for class 'character'
write_sigs(x, file = stdout(), ...)
```

Arguments

x	A list of function signatures. See details.
file	A file path or connection to write the output to (stdout by default).
...	passed to toString.siglist.

Details

Where x is an object of class siglist, the function essentially calls writeLines(tostring(x). If the input is a single function signature (of class sig), then it is coerced into a siglist. If the input is an environment or path to a file, then list_sigs is called on the input before writing.

Value

A character vector of the lines that were written to file is invisibly returned. Mostly invoked for the side effect of writing function signatures to a file.

Examples

```
#Step by step:
#First, list some sigs.
utils_sigs <- list_sigs(pkg2env(utils))
#Without a file argument, sigs are just printed to the console.
head(write_sigs(utils_sigs))
#Write to a file
tmpf <- tempfile("sig", fileext = ".R")
write_sigs(utils_sigs, tmpf)
#Open the file we've just written
readLines(tmpf, n = 6)
#Can also list and write in one line.
tmpf2 <- tempfile("sig", fileext = ".R")
write_sigs(pkg2env(grDevices), tmpf2)
#Single sigs are coerced to siglists
write_sigs(sig(stats::var))
```

[*Indexing for siglists*

Description

Get or set a subset of a siglist.

Usage

```
## S3 method for class 'siglist'
x[i, ...]

## S3 method for class 'siglist'
x[[i, ...]]

## S3 replacement method for class 'siglist'
x[...] <- value

## S3 replacement method for class 'siglist'
x[[...]] <- value
```

Arguments

x	A siglist object.
i	An integer vector index.
...	Passed from other index methods.
value	A value to set the subset to.

Value

A siglist.

See Also

Extract

Examples

```
methods_sigs <- list_sigs(pkg2env(methods))
methods_sigs[1:5]
methods_sigs[[1]]
```

Index