Package: globals (via r-universe)

November 5, 2026

Version 0.20.0-9000

Depends R (>= 3.1.2)

Imports codetools

Title Identify Global Objects in R Expressions

Description Identifies global (``unknown'' or ``free'') objects in R expressions by code inspection using various strategies (ordered, liberal, conservative, or deep-first search). The objective of this package is to make it as simple as possible to identify global objects for the purpose of exporting them in parallel, distributed compute environments.

License LGPL (>= 2.1)

LazyLoad TRUE

ByteCompile TRUE

Language en-US

Encoding UTF-8

URL https://globals.futureverse.org,
https://github.com/futureverse/globals

BugReports https://github.com/futureverse/globals/issues

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.0

Repository https://futureverse-devel.r-universe.dev

Date/Publication 2026-11-05 15:57:29 UTC

RemoteUrl https://github.com/futureverse/globals

RemoteRef develop

RemoteSha 843fa76e14e1048d921f872e97215f5b0f568f86

Contents

cleanup 2
findGlobals 2
Globals 5
globalsByName 5
packagesOf 7
print.Globals 7

Index **8**

`cleanup` *Drop certain types of globals*

Description

Drop certain types of globals

Usage

```
cleanup(...)

## S3 method for class 'Globals'
cleanup(globals, drop = c("missing", "base-packages", "nativesymbolinfo"), ...)
```

Arguments

`globals`	A Globals object.
`drop`	A character vector specifying what type of globals to drop.
`...`	Not used

Value

Return A Globals object.

`findGlobals` *Get all global objects of an expression*

Description

Get all global objects of an expression

Usage

```
findGlobals(
  expr,
  envir = parent.frame(),
  ...,
  attributes = TRUE,
  tweak = NULL,
  dotdotdot = c("warning", "error", "return", "ignore"),
  method = c("ordered", "conservative", "liberal", "dfs"),
  substitute = FALSE,
  unlist = TRUE,
  trace = FALSE
)

globalsOf(
  expr,
  envir = parent.frame(),
  ...,
  method = c("ordered", "conservative", "liberal", "dfs"),
  tweak = NULL,
  locals = NA,
  substitute = FALSE,
  mustExist = TRUE,
  unlist = TRUE,
  recursive = TRUE,
  skip = NULL,
  ignore = NULL
)
```

Arguments

`expr`	An R expression.
`envir`	The environment from where to search for globals.
`attributes`	If TRUE (default), attributes of `expr` are also searched. If FALSE, they are not. If a character vector, then attributes with matching names are searched. Note, the attributes of the attributes' elements are not searched, that is, attributes are not searched recursively. Also, attributes are searched with `dotdotdot = "ignore"`.
`tweak`	An optional function that takes an expression and returns a tweaked expression.
`dotdotdot`	TBD.
`method`	A character string specifying what type of search algorithm to use.
`substitute`	If TRUE, the expression is `substitute()`:ed, otherwise not.
`unlist`	If TRUE, a list of unique objects is returned. If FALSE, a list of `length(expr)` sublists.
`trace`	TBD.
`locals`	Should globals part of any "local" environment of a function be included or not?

mustExist	If TRUE, an error is thrown if the object of the identified global cannot be located. Otherwise, the global is not returned.
recursive	If TRUE, found globals are searched for additional globals. For example, a closure (function) that exists outside a package namespace may contain additional globals. Similarly, a formula may depend on globals.
skip	(internal) A list of globals not to be searched for additional globals. Ignored unless recursive is TRUE.
ignore	An optional character vector of names of globals to drop. These are excluded before they are searched for and the do not need to exist.
...	Not used.

Details

There are currently three strategies for identifying global objects.

The method = "ordered" search method identifies globals such that a global variable preceding a local variable with the same name is not dropped (unlike the "conservative" method).

The method = "conservative" search method tries to keep the number of false positives to a minimum, i.e. the identified objects are most likely true global objects. At the same time, there is a risk that some true globals are not identified (see example). This search method returns the exact same result as the findGlobals() function of the **codetools** package.

The method = "liberal" search method tries to keep the true-positive ratio as high as possible, i.e. the true globals are most likely among the identified ones. At the same time, there is a risk that some false positives are also identified.

The method = "dfs" search method identifies globals in the abstract syntax tree (AST) using a depth-first search, which better emulates how the R engine identifies global variables.

With recursive = TRUE, globals part of locally defined functions will also be found, otherwise not.

Value

findGlobals() returns a character vector.

globalsOf() returns a Globals object.

See Also

Internally, the **codetools** package is utilized for code inspections.

Examples

```
b <- 2
expr <- substitute({ a <- b; b <- 1 })

## Will _not_ identify 'b' (because it's also a local)
globalsC <- globalsOf(expr, method = "conservative")
print(globalsC)

## Will identify 'b'
globalsL <- globalsOf(expr, method = "liberal")
print(globalsL)
```

Globals *A representation of a set of globals*

Description

A representation of a set of globals

Usage

```
Globals(object, ...)

as.Globals(x, ...)
```

Arguments

object	A named list.
x	An object to be coerced to a Globals object.
...	Not used.

Value

An object of class Globals, which is a *named* list of the value of the globals, where the element names are the names of the globals. Attribute where is a named list of the same length and with the same names.

See Also

The globalsOf() function identifies globals from an R expression and returns a Globals object.

globalsByName *Locates and retrieves a set of global variables by their names*

Description

Locates and retrieves a set of global variables by their names

Usage

```
globalsByName(names, envir = parent.frame(), mustExist = TRUE, ...)
```

Arguments

names	A character vector of global variable names.
envir	The environment from where to search for globals.
mustExist	If TRUE, an error is thrown if the object of the identified global cannot be located. Otherwise, the global is not returned.
...	Not used.

Value

A Globals object of named elements and an attribute `where` with named elements. Both sets have names according to `names`.

Special "argument" globals

If `names` specifies `"..."`, `"..1"`, `"..2"`, ..., then they are interpreted as arguments `...`, `..1`, `..2`, ..., respectively. If specified, then the corresponding elements in the results are lists of class `DotDotDotList` comprising the value of the latter. If the special argument does not exist, then the value is `NA`, and the corresponding `where` attribute is `NULL`.

Examples

```
f <- function(x = 42, ...) {
  globalsByName("x")
}

globals <- f()
str(globals)

globals <- f(3.14)
str(globals)


g <- function(x = 42, ...) {
  globalsByName("...")
}

globals <- g()
str(globals)

globals <- g(3.14)
str(globals)

globals <- g(3.14, 1L, b = 2L, c = 3L)
str(globals)


h <- function(x = 42, ...) {
  globalsByName("..2")
}

globals <- h(x = 3.14, a = 1, b = 2)
str(globals)

globals <- g(3.14)
str(globals)

globals <- g(3.14, 1L, b = 2L, c = 3L)
str(globals)
```

packagesOf *Identify the packages of the globals*

Description

Identify the packages of the globals

Usage

```
packagesOf(...)

## S3 method for class 'Globals'
packagesOf(globals, ...)
```

Arguments

globals	A Globals object.
...	Not used.

Value

Returns a character vector of package names.

print.Globals *Print a Globals Object*

Description

Print a Globals Object

Usage

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

Arguments

x	A Globals object.
...	Not used.

Value

(invisibly) the Globals object itself.

Index

[.Globals *(*Globals*)*, 5

as.Globals *(*Globals*)*, 5

cleanup, 2

findGlobals, 2, *4*

Globals, *4*, 5, *5–7*
globalsByName, 5
globalsOf, *5*
globalsOf *(*findGlobals*)*, 2

names *(*Globals*)*, 5

packagesOf, 7
print.Globals, 7