Package ‘globals’

December 7, 2019

Version 0.12.5

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, e.g. conservative or liberal. The objective of this package is to make it as simple as possible to identify global objects for the purpose of exporting them in distributed compute environments.

License LGPL (>= 2.1)

LazyLoad TRUE

ByteCompile TRUE

URL `https://github.com/HenrikBengtsson/globals`

BugReports `https://github.com/HenrikBengtsson/globals/issues`

RoxygenNote 7.0.2

NeedsCompilation no

Author Henrik Bengtsson [aut, cre, cph]

Maintainer Henrik Bengtsson <`henrikb@braju.com`>

Repository CRAN

Date/Publication 2019-12-07 21:10:02 UTC

R topics documented:

cleanup.Globals *Drop certain types of globals*

Description

Drop certain types of globals

Usage

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

Arguments

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

findGlobals *Get all global objects of an expression*

Description

Get all global objects of an expression

Usage

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

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

Arguments

`expr`	An R expression.
`envir`	The environment from where to search for globals.
`...`	Not used.
`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.
`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, globals that are closures (functions) and that exist outside of namespaces ("packages"), will be recursively scanned for globals.
`skip`	(internal) A list of globals not to be searched for additional globals. Ignored unless `recursive` is TRUE.

Details

There 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 (which the `"conservative"` method would).

The `method = "conservative"` search method tries to keep the number of false positive 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.

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, ...)
```

Arguments

`object`	A named list.
`...`	Not used.

Value

An object of class `Future`.

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.

packagesOf.Globals — *Identify the packages of the globals*

Description

Identify the packages of the globals

Usage

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

Arguments

`globals`	A Globals object.
`...`	Not used.

Value

Returns a character vector of package names.

Index