Package 'oshka'

July 22, 2025

Title Recursive Quoted Language Expansion

Description Expands quoted language by recursively replacing any symbol that points to quoted language with the language it points to. The recursive process continues until only symbols that point to non-language objects remain. The resulting quoted language can then be evaluated normally. This differs from the traditional 'quote'/'eval' pattern because it resolves intermediate language objects that would interfere with evaluation.

Version 0.1.2

Depends R (>= 3.3.2)

License GPL (>= 2)

LazyData true

URL https://github.com/brodieG/oshka

BugReports https://github.com/brodieG/oshka/issues

VignetteBuilder knitr

Imports utils

Suggests knitr, rmarkdown, unitizer, covr

RoxygenNote 6.0.1

NeedsCompilation no

Author Brodie Gaslam [aut, cre]

Maintainer Brodie Gaslam <brodie.gaslam@yahoo.com>

Repository CRAN

Date/Publication 2017-10-14 13:00:34 UTC

Contents

oshka-package *Recursive Quoted Language Expansion*

Description

Expands quoted language by recursively replacing any symbol that points to quoted language with the language it points to. The recursive process continues until only symbols that point to non-language objects remain. The resulting quoted language can then be evaluated normally. This differs from the traditional 'quote'/'eval' pattern because it resolves intermediate language objects that would interfere with evaluation.

expand *Recursively Expand Symbols in Quoted Language*

Description

Finds symbols in quoted R language objects and recursively replaces them with any language objects that those symbols point to. This leads to an expanded language object that can be evaluated. Language objects are objects of type "symbol", "language", or "expression", though only unclassed language is expanded by default.

Usage

```
expand(expr, envir = parent.frame(), enclos = if (is.list(envir) ||
  is.pairlist(envir)) parent.frame() else baseenv(),
  class.shield = getOption("oshka.class.shield"),
  name.shield = getOption("oshka.name.shield"))
```

Arguments

`expr`	an object to be evaluated. See 'Details'.
`envir`	the `environment` in which `expr` is to be evaluated. May also be `NULL`, a list, a data frame, a pairlist or an integer as specified to `sys.call`.
`enclos`	Relevant when `envir` is a (pair)list or a data frame. Specifies the enclosure, i.e., where R looks for objects not found in `envir`. This can be `NULL` (interpreted as the base package environment, `baseenv()`) or an environment.
`class.shield`	TRUE, FALSE, or character, determines what portions of quoted language are shielded from expansion. TRUE, the default, means that any any classed language (e.g. formula) will be left unexpanded. If FALSE all language will be expanded, irrespective of class. If character, then any classed objects with classes in the vector will be left unexpanded, and all others will be expanded.
`name.shield`	character names of symbols that should not be expanded, which by default is `c("::", ":::")`. If position 1 in a call (i.e. the function name) is a name in this list , then the entire call is left unexpanded.

Details

For more general documentation `browseVignettes('oshka')`.

Value

If the input is a language object, that object with all symbols recursively expanded, otherwise the input unchanged.

Programmable NSE

The expansion can be used to implement programmable Non-Standard Evaluation (NSE hereafter). Users can create complex quoted language expressions from simple ones by combining them as they would tokens in standard R expressions. Then, a programmable NSE aware function can use `expand` to turn the quoted language into usable form. See examples.

Expansion mechanics

During the recursive expansion, symbols are looked up through the search path in the same way as standard R evaluation looks up symbols. One subtlety is that if symbol A expands to a language object B, the symbols in language object B are looked for starting from the environment that A is bound to, not the initial evaluation environment. Expansion stops at symbols that point to non-language objects.

Symbols at the first position in calls (e.g. `fun` in `fun(x, y)`) are expanded slightly differently: they will continue to be expanded until an object of mode "function" is found. This is to follow the semantics of symbol searches in R where a symbol pointing to a non-function object will not mask a symbol pointing to a function object when it is used as the name of the function in a call.

You can prevent expansion on portions of language via shielding. Some language is not expanded by default (see next section).

Shielding

There are two mechanisms for shielding language from expansion. The first one is to give language a class. This is why formulas are not expanded by default. Be careful though that you do not give a symbol a class as that is bad practice and will become an R runtime error in the future.

The second mechanism is to specify symbol names that should not be expanded. This is easier to specify than the class based mechanism, but it is less precise as it applies to all instances of that name. By default the symbols "::" and ":::" are not expanded. If a function call has a shielded symbol for function name the *entire* call will be shielded.

See the `class.shield` and `name.shield` parameters, and examples.

Examples

```
xzw <- uvt <- NULL  # make sure not lang objects
aaa <- quote(xzw > 3)
bbb <- quote(xzw < 10)
ccc <- quote(aaa & bbb)
expand(ccc)
```

```
## You can place list like objects in the search path
l <- list(bbb=quote(uvt < 9999))
expand(ccc, l)

## But notice what happens if we use `quote(ccc)` instead of
## just `ccc`.  This is because in this case `expand` must
## look for the `ccc` symbol in the search path, and once
## it finds it it looks for `aaa` and `bbb` starting from the
## environment `ccc` is bound to, so the `bbb` defined
## inside `l` is skipped.
expand(quote(ccc), l)

## Implementing an NSE fun (see vignettes for detailed
## examples)
subset2 <- function(x, subset) {
  subset <- expand(substitute(subset), x, parent.frame())
  eval(bquote(base::subset(.(x), .(subset))), parent.frame())
}
subset2(iris, Sepal.Width > 4.3)
iris.sub <- quote(Sepal.Width > 4.3)
subset2(iris, iris.sub)

## You can shield all instances of a symbol from expansion.
## Note we append existing name shield list.

expand(ccc, name.shield=c(getOption('oshka.name.shield'), 'bbb'))

## You can also shield by attaching classes to language
## objects or portions thereof
expand(I(ccc))  # add the `AsIs` class to `ccc` with `I`
expand(ccc)

## If you wish to shield a symbol with this method you
## cannot do so directly.  Note the `quote((bbb))` as
## otherwise we would attach attributes to a symbol:
ccd <- bquote(aaa & .(I(quote((bbb)))))
expand(ccd)

## Equivalently
cce <- ccc
cce[[3]] <- I(quote((bbb)))
expand(cce)

## Formulas not expanded by default, but can be forced
## to expand by setting `class.shield` to FALSE
expand(aaa ~ bbb)
expand(aaa ~ bbb, class.shield=FALSE)
```

Index