Package ‘collectArgs’

October 14, 2017

Title Quickly and Neatly Collect Arguments from One Environment to Pass to Another

Version 0.4.0

Description We often want to take all (or most) of the objects in one environment (such as the parameter values of a function) and pass them to another. This might be calling a second function, or iterating over a list, calling the same function. These functions wrap often repeated code.
Current stable version (committed on October 14, 2017).

Depends R (>= 3.0.2)

License MIT + file LICENSE

Encoding UTF-8

LazyData TRUE

Imports magrittr, stats

RoxygenNote 6.0.1.9000

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

NeedsCompilation no

Author Rick Saporta [aut, cre]

Maintainer Rick Saporta <RickSaporta@gmail.com>

Repository CRAN

Date/Publication 2017-10-14 15:06:49 UTC

R topics documented:

collectArgs-and-iterateWithArgs

collectArgs and iterateWithArgs

Description

Functions to cleanly collect arguments from within one function or environment (to then pass to another or to iterate over)

Usage

```
collectArgs(except = c(), incl.dots = TRUE, all.names = TRUE,
  envir = parent.frame())

iterateWithArgs(arg_to_iterate_over, FUNC,
  nm.arg_to_iterate_over = as.character(substitute(arg_to_iterate_over)),
  except = c(), incl.dots = TRUE, envir = parent.frame())
```

Arguments

`except`	A vector of string values. Objects to *NOT* include in the collection Generally, the user will not want to pass objets created inside the function and hence will pass to except _NOTE_ pass the quoted string-name of the object, not the object itself.
`incl.dots`	A single logical value. Should the `...` be collected as well? *NOTE: Has no effect in functions without dots argument* Default is `TRUE`.
`all.names`	A single logical value. Passed to `ls()`. When `FALSE`, then objects whose name begins with a '.' are omitted from the collection
`envir`	An `environment` object. Passed to `ls()`. The environment from which to collect the objects. Defaults to `parent.frame`
`arg_to_iterate_over`	Object, not the string-name of the object.
`FUNC`	function or string of length 1. function to iterate over. Normally the same function in which `iterateWithArgs` is being called
`nm.arg_to_iterate_over`	The string-name of the object. Default is `as.character(substitute(arg_to_iterate_over))`

Details

`collectArgs()` colects objects from an envrionment into a single list. Generally, the list will then be passed to other functions (usually with `do.call`)

`iterateWithArgs()` similarly collects the objects in an environment, with the difference that one specific object is selected to iterate over. For each iteration, the given value is passed along with all the other objects to `FUNC`.

Value

for `collectArgs`: A list of all of the objects in `envir` (less any objects excluded via the parameters). The names of the list are the names of object in `envir`.

for `iterateWithArgs`: A list of the return values of `FUNC`, the length of `arg_to_iterate_over`. Naming of the list will be handled by `do.call`

Examples

```
sample_function <- function(x, base, thresh=500, verbose=TRUE) {

  some_object    <- is.na(x) ## an example of an object that we will exclude
  another_object <- 1:10     ## an example of an object that we will exclude

  if (length(x) > 1) {
   return(iterateWithArgs(x, FUNC=sample_function, except=c("some_object", "another_object")))
  }

  ret <- (base ^ x)

  if (verbose)
   cat(base, "^", x, " is ", ifelse(ret > thresh, "", "NOT "), "larger than ", thresh, "\n")

  return(ret)
}

sample_function(5, base=2)
sample_function(5:10, base=2)


 some_function <- function(x, param1, param2, etc, ...) {

   ARGS <- collectArgs(except="x")
   return(
           lapply(x, function(x_i)
              do.call(some_function, c(ARGS, x=x_i))
           )
         )
 }
```

Index