Package ‘ellipsis’

September 20, 2019

Version 0.3.0
Title Tools for Working with ...
Description The ellipsis is a powerful tool for extending functions. Unfortunately this power comes at a cost: misspelled arguments will be silently ignored. The ellipsis package provides a collection of functions to catch problems and alert the user.
License GPL-3
Encoding UTF-8
LazyData true
RoxygenNote 6.1.1
BugReports https://github.com/r-lib/ellipsis/issues
Depends R (>= 3.2)
Imports rlang (>= 0.3.0)
Suggests covr, testthat
NeedsCompilation yes
Author Hadley Wickham [aut, cre], RStudio [cph]
Maintainer Hadley Wickham <hadley@rstudio.com>
Repository CRAN
Date/Publication 2019-09-20 20:40:02 UTC

R topics documented:

check_dots_empty .................................................. 2
check_dots_unnamed ............................................. 2
check_dots_used .................................................. 3
safe_median ....................................................... 4

Index 5
check_dots_unnamed

Check that all dots are unnamed

Description

Named arguments in ... are often a sign of misspelled argument names.

Usage

check_dots_unnamed(env = parent.frame(), action = abort)

Arguments

- env: Environment in which to look for ....
- action: The action to take when the dots have not been used. One of `rlang::abort()`, `rlang::warn()`, `rlang::inform()` or `rlang::signal()`.

Examples

```r
f <- function(x, ..., foofy = 8) {
  check_dots_unnamed()
  x + foofy
}

try(f(1, foof = 4))
f(1, foofy = 4)
```

check_dots_empty

Check that dots are unused

Description

Sometimes you just want to use ... to force your users to fully name the details arguments. This function warns if ... is not empty.

Usage

check_dots_empty(env = parent.frame(), action = abort)

Arguments

- env: Environment in which to look for ....
- action: The action to take when the dots have not been used. One of `rlang::abort()`, `rlang::warn()`, `rlang::inform()` or `rlang::signal()`.

Examples

```r
f <- function(x, ..., foofy = 8) {
  check_dots_empty()
  x + foofy
}

try(f(1, foof = 4))
f(1, foofy = 4)
```
check_dots_used

Examples

```r
f <- function(..., foofy = 8) {
  check_dots_unnamed()
  c(...)
}

f(1, 2, 3, foofy = 4)
try(f(1, 2, 3, foof = 4))
```

check_dots_used               Check that all dots have been used

Description

Automatically sets exit handler to run when function terminates, checking that all elements of ... have been evaluated. If you use on.exit() elsewhere in your function, make sure to use add = TRUE so that you don’t override the handler set up by check_dots_used().

Usage

```r
check_dots_used(env = parent.frame(), action = abort)
```

Arguments

- `env` Environment in which to look for ... and to set up handler.
- `action` The action to take when the dots have not been used. One of rlang::abort(), rlang::warn(), rlang::inform() or rlang::signal().

Examples

```r
f <- function(...) {
  check_dots_used()
  g(...)
}

g <- function(x, y, ...) {
  x + y
}

f(x = 1, y = 2)
try(f(x = 1, y = 2, z = 3))
try(f(x = 1, y = 2, 3, 4, 5))
```
safe_median  Safe version of median

Description

safe_median() works stats::median() but warns if some elements of ... are never used.

Usage

safe_median(x, ...)

## S3 method for class 'numeric'
safe_median(x, ..., na.rm = TRUE)

Arguments

x  Numeric vector
...
Additional arguments passed on to methods.
na.rm For numeric method, should missing values be removed?

Examples

x <- c(1:10, NA)
safe_median(x, na.rm = TRUE)
median(x, na.rm = TRUE)

try(median(x, na.rm = TRUE))
try(safe_median(x, na.rm = TRUE))

try(median(1, 2, 3))
try(safe_median(1, 2, 3))
Index

check_dots_empty, 2
check_dots_unnamed, 2
check_dots_used, 3
on.exit(), 3
rlang::abort(), 2, 3
rlang::inform(), 2, 3
rlang::signal(), 2, 3
rlang::warn(), 2, 3
safe_median, 4
stats::median(), 4