

Package: ellipsis (via r-universe)

June 7, 2024

Version 0.3.2.9000

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 MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

URL <https://ellipsis.r-lib.org>, <https://github.com/r-lib/ellipsis>

BugReports <https://github.com/r-lib/ellipsis/issues>

Depends R (>= 3.2)

Imports rlang (>= 0.3.0)

Suggests covr, testthat

Repository <https://r-lib.r-universe.dev>

RemoteUrl <https://github.com/r-lib/ellipsis>

RemoteRef HEAD

RemoteSha 55fe5b861529bad3e9fe93650b48064cb2345c50

Contents

check_dots_empty	2
check_dots_unnamed	2
check_dots_used	3
safe_median	3

Index	5
--------------	----------

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.

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

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

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

check_dots_unnamed *Check that all dots are unnamed*

Description

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

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

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

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

check_dots_used	<i>Check that all dots have been used</i>
-----------------	---

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()`.

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 <code>rlang::abort()</code> , <code>rlang::warn()</code> , <code>rlang::inform()</code> or <code>rlang::signal()</code> .

Examples

```
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	<i>Safe version of median</i>
-------------	-------------------------------

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

<code>x</code>	Numeric vector
<code>...</code>	Additional arguments passed on to methods.
<code>na.rm</code>	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`, 3
`stats::median()`, 3