

Package ‘where’

July 22, 2025

Type Package

Title Vectorised Substitution and Evaluation

Version 1.0.0

Description Provides a clean syntax for vectorising the use of
Non-Standard Evaluation (NSE), for example in 'ggplot2', 'dplyr', or
'data.table'.

License MIT + file LICENSE

URL <https://github.com/KiwiMateo/where>

BugReports <https://github.com/KiwiMateo/where/issues>

Suggests data.table, dplyr, ggplot2, knitr, rmarkdown, testthat (>=
3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Matt Hendtlass [aut, cre]

Maintainer Matt Hendtlass <m.hendtlass@gmail.com>

Repository CRAN

Date/Publication 2024-11-19 09:40:02 UTC

Contents

.	2
run	2
%with%	4
Index	5

.	<i>Capture expressions</i>
---	----------------------------

Description

Capture expressions

Usage

`.(...)`

Arguments

... code

Value

a list

Examples

```
.(a = 1, b = x^2, c = filter(iris, Species == "vericolor"))
```

run	<i>Run interpolated code</i>
-----	------------------------------

Description

Vectorised substitution of expressions into a large code block and execution.

Usage

```
run(expr, ..., e = parent.frame())
```

expr %for% x

expr %where% pars

Arguments

- | | |
|------|--|
| expr | the code to run |
| ... | named values to be substituted by name into 'expr' |
| e | environment, for evaluation; defaults to 'parent.frame()' |
| x | list of expressions to be substituted for 'x' in 'expr' |
| pars | a named list of values to be substituted by name into 'expr' |

Details

‘ ‘

Value

A list.

Examples

```
library(dplyr)

subgroups = .(all = TRUE,
               long_sepal = Sepal.Length > 6,
               long_petal = Petal.Length > 5.5)
functions = .(mean, sum, prod)

run(
  iris %>%
    filter(subgroup) %>%
    summarise(across(Sepal.Length:Petal.Width,
                     summary),
              .by = Species),
  subgroup = subgroups,
  summary = functions
)

library(data.table)
df <- as.data.table(iris)

run(df[subgroup, lapply(.SD, functions), keyby = "Species",
  .SDcols = Sepal.Length:Petal.Width],

    subgroup = subgroups,
    functions = functions)

library(ggplot2)

plots <- run(
  ggplot(filter(iris, subgroup),
           aes(Sepal.Length, Sepal.Width)) +
    geom_point() +
    theme_minimal(),
  subgroup = subgroups
)
Map(function(plot, name) plot + ggtitle(name), plots, names(plots))

(
  iris %>%
    filter(subgroup) %>%
    summarise(across(Sepal.Length:Petal.Width,
                     summary),
              .by = Species)
```

```

) %where%
  list(subgroup = subgroups,
        summary  = functions)

library(ggplot2)
(
  ggplot(filter(iris, x),
    aes(Sepal.Length, Sepal.Width)) +
    geom_point() +
    theme_minimal()
) %for% subgroups

```

<i>%with%</i>	<i>Posterior variable declaration</i>
---------------	---------------------------------------

Description

Posterior variable declaration

Usage

`expr %with% variables`

Arguments

<code>expr</code>	expression to evaluate
<code>variables</code>	expression with variable assignments

Value

The value of the evaluated expression.

Examples

```

(a + b) %with% {
  a = 1
  b = 2
}

```

Index

..[2](#)
%for%(run), [2](#)
%where%(run), [2](#)
%with%, [4](#)

run, [2](#)