Package 'tidyselect'

October 11, 2018

Title Select from a Set of Strings

Version 0.2.5

Description A backend for the selecting functions of the 'tidyverse'. It makes it easy to implement select-like functions in your own packages in a way that is consistent with other 'tidyverse' interfaces for selection.

Depends R (>= 3.1)

Imports glue (>= 1.3.0), purrr, rlang (>= 0.2.2), Rcpp (>= 0.12.0)

Suggests covr, dplyr, testthat

LinkingTo Rcpp (>= 0.12.0),

License GPL-3

Encoding UTF-8

LazyData true

ByteCompile true

RoxygenNote 6.1.0

NeedsCompilation yes

Author Lionel Henry [aut, cre],
Hadley Wickham [aut],
RStudio [cph]

Maintainer Lionel Henry <lionel@rstudio.com>

Repository CRAN

Date/Publication 2018-10-11 13:00:03 UTC

R topics documented:

tidyselect-package *tidyselect: Select from a Set of Strings*

Description

A backend for the selecting functions of the 'tidyverse'. It makes it easy to implement select-like functions in your own packages in a way that is consistent with other 'tidyverse' interfaces for selection.

Author(s)

Maintainer: Lionel Henry <lionel@rstudio.com>

Authors:

- Hadley Wickham <hadley@rstudio.com>

Other contributors:

- RStudio [copyright holder]

poke_vars *Replace or get current variables*

Description

Variables are made available to select helpers by registering them in a special placeholder.

- `scoped_vars()` changes the current variables and sets up a function exit hook that automatically restores the previous variables once the current function returns.
- `with_vars()` takes an expression to be evaluated in a variable context.
- `poke_vars()` changes the contents of the placeholder with a new set of variables. It returns the previous variables invisibly and it is your responsibility to restore them after you are done. This is for expert use only.
- `peek_vars()` returns the variables currently registered.
- `has_vars()` returns `TRUE` if a variable context has been set, `FALSE` otherwise.

Usage

```
poke_vars(vars)

peek_vars()

scoped_vars(vars, frame = caller_env())

with_vars(vars, expr)

has_vars()
```

Arguments

vars	A character vector of variable names.
frame	The frame environment where the exit hook for restoring the old variables should be registered.
expr	An expression to be evaluated within the variable context.

Value

For poke_vars() and scoped_vars(), the old variables invisibly. For peek_vars(), the variables currently registered.

Examples

```
poke_vars(letters)
peek_vars()

# Now that the variables are registered, the helpers can figure out
# the positions of elements within the variable vector:
one_of(c("d", "z"))

# In a function be sure to restore the previous variables. An exit
# hook is the best way to do it:
fn <- function(vars) {
  old <- poke_vars(vars)
  on.exit(poke_vars(old))

  one_of("d")
}
fn(letters)
fn(letters[3:5])

# The previous variables are still registered after fn() was
# called:
peek_vars()


# It is recommended to use the scoped variant as it restores the
# state automatically when the function returns:
fn <- function(vars) {
  scoped_vars(vars)
  starts_with("r")
}
fn(c("red", "blue", "rose"))

# The with_vars() helper makes it easy to pass an expression that
# should be evaluated in a variable context. Thanks to lazy
# evaluation, you can just pass the expression argument from your
# wrapper to with_vars():
fn <- function(expr) {
  vars <- c("red", "blue", "rose")
  with_vars(vars, expr)
```

```
}
fn(starts_with("r"))
```

`select_helpers` *Select helpers*

Description

These functions allow you to select variables based on their names.

- `starts_with()`: Starts with a prefix.
- `ends_with()`: Ends with a suffix.
- `contains()`: Contains a literal string.
- `matches()`: Matches a regular expression.
- `num_range()`: Matches a numerical range like x01, x02, x03.
- `one_of()`: Matches variable names in a character vector.
- `everything()`: Matches all variables.
- `last_col()`: Select last variable, possibly with an offset.

Usage

```
starts_with(match, ignore.case = TRUE, vars = peek_vars())

ends_with(match, ignore.case = TRUE, vars = peek_vars())

contains(match, ignore.case = TRUE, vars = peek_vars())

matches(match, ignore.case = TRUE, vars = peek_vars())

num_range(prefix, range, width = NULL, vars = peek_vars())

one_of(..., .vars = peek_vars())

everything(vars = peek_vars())

last_col(offset = 0L, vars = peek_vars())
```

Arguments

`match`	A string.
`ignore.case`	If `TRUE`, the default, ignores case when matching names.
`vars, .vars`	A character vector of variable names. When called from inside selecting functions like `dplyr::select()` these are automatically set to the names of the table.

prefix	A prefix that starts the numeric range.
range	A sequence of integers, like 1:5.
width	Optionally, the "width" of the numeric range. For example, a range of 2 gives "01", a range of three "001", etc.
...	One or more character vectors.
offset	Set it to n to select the nth var from the end.

Value

An integer vector giving the position of the matched variables.

Examples

```
nms <- names(iris)
vars_select(nms, starts_with("Petal"))
vars_select(nms, ends_with("Width"))
vars_select(nms, contains("etal"))
vars_select(nms, matches(".t."))
vars_select(nms, Petal.Length, Petal.Width)
vars_select(nms, everything())
vars_select(nms, last_col())
vars_select(nms, last_col(offset = 2))

vars <- c("Petal.Length", "Petal.Width")
vars_select(nms, one_of(vars))
```

---

vars_select_helpers *List of selection helpers*

Description

This list contains all selection helpers exported in tidyselect. It is useful when you want to embed the helpers in your API without having to track addition of new helpers in tidyselect.

Usage

```
vars_select_helpers
```

Format

An object of class list of length 8.

Examples

```
# You can easily embed the helpers by burying them in the scopes of
# input quosures. For this example we need an environment where
# tidyselect is not attached:
local(envir = baseenv(), {
  vars <- c("foo", "bar", "baz")
  helpers <- tidyselect::vars_select_helpers

  my_select <- function(...) {
    quos <- rlang::quos(...)
    quos <- lapply(quos, rlang::env_bury, !!! helpers)

    tidyselect::vars_select(vars, !!! quos)
  }

  # The user can now call my_select() with helpers without having
  # to attach tidyselect:
  my_select(starts_with("b"))
})
```

Index