

Package ‘tribe’

October 14, 2022

Type Package

Title Play with the Tribe of Attributes

Version 0.1.8

Description Functions to make manipulation of object attributes easier.

It also contains a few functions that extend the 'dplyr' package for data manipulation, and it provides new pipe operators, including the pipe '%@>%' similar to the 'magrittr' '%>%', but with the additional functionality to enable attributes propagation.

License MIT + file LICENSE

LazyData TRUE

ByteCompile TRUE

Depends R (>= 3.2)

Imports dplyr, lazyeval, magrittr, rlang, rstudioapi, utils

VignetteBuilder knitr

Suggests knitr, testthat

URL <https://github.com/paulponcet/tribe>

BugReports <https://github.com/paulponcet/tribe/issues>

RoxygenNote 7.0.0

NeedsCompilation no

Author Paul Poncet [aut, cre],
Stefan Milton Bache [aut] (for functions copied or modified from the
'magrittr' package),
Hadley Wickham [aut] (for functions copied or modified from the
'magrittr' package)

Maintainer Paul Poncet <paulponcet@yahoo.fr>

Repository CRAN

Date/Publication 2019-11-23 06:20:03 UTC

R topics documented:

at_mutate	2
make_pipe	3
shield	4
stick_to	6
tribe	7

Index	8
--------------	----------

at_mutate	<i>Manipulate attributes in a dplyr fashion</i>
-----------	---

Description

The function `at_mutate` adds or changes attributes to `obj`.

The function `at_select` selects attributes of `obj`, and removes the others.

The function `at_rename` renames attributes of `obj`.

The function `at_slice` chooses a specific attribute and returns it.

Usage

```
at_mutate(obj, ...)
```

```
at_mutate_(obj, ..., .dots)
```

```
at_select(obj, ...)
```

```
at_select_(obj, ..., .dots)
```

```
at_rename(obj, ...)
```

```
at_rename_(obj, ..., .dots)
```

```
at_slice(obj, at)
```

```
at_slice_(obj, at)
```

Arguments

<code>obj</code>	An object.
<code>...</code>	Comma separated list of unquoted expressions.
<code>.dots</code>	Used to work around non-standard evaluation.
<code>at</code>	Attribute to be obtained.

Value

at_slice returns the attribute chosen. The other functions return obj with possibly modified attributes.

See Also

[structure, attributes](#)

Examples

```
library(dplyr)
df <- data.frame(x = sample(10, 5, rep = TRUE),
                 y = sample(10, 5, rep = TRUE)) %>%
  at_mutate(example = "yes",
            package = "dplyr")
tribe(df)

at_slice(df, names)
at_slice_(df, "class")
at_slice_(df, ~ package)

df <- df %>%
  at_mutate_(package = ~ NULL,
            example = ~ "no")
tribe(df)

df <- df %>%
  at_mutate_(.dots = list(x = ~ 2, y = ~ c(3,4)))
tribe(df)
```

make_pipe

Create a pipe operator.

Description

This function is used to create magrittr like pipe operators.

Usage

```
make_pipe(propagate, keep_also = NULL, try = FALSE)
```

```
lhs %@>% rhs
```

```
lhs %<@>% rhs
```

```
lhs %try>% rhs
```

Arguments

propagate	character. See the eponymous argument in shield .
keep_also	character. See the eponymous argument in shield .
try	logical. If TRUE and the pipe <code>x > f</code> generates an error, then the pipe <code>x try> f</code> returns <code>x</code> unchanged silently.
lhs	Left-hand side of the pipe.
rhs	Right-hand side of the pipe.

Author(s)

Stefan Milton Bache and Hadley Wickham for the original pipe function in package **magrittr**; Paul Poncet for the modifications introduced.

See Also

[shield](#) in this package.

Examples

```
library(dplyr)
df <- data.frame(x = sample(10, 5, rep = TRUE),
                y = sample(10, 5, rep = TRUE)) %>%
  at_mutate(example = "yes",
            package = "dplyr",
            class = c("my_tbl", "data.frame"))
tribe(df)

# Attributes just created are lost when the object
# passes through dplyr verbs
tribe(df %>% mutate(z = 3))

# With the pipe '%@>%', most attributes are kept
tribe(df %@>% mutate(z = 3))

# One can create a new pipe to adjust attributes propagation settings
"%newpipe%" <- make_pipe(propagate = "none", keep_also = "example")
tribe(df %newpipe% mutate(z=3))
```

 shield

Attributes protection

Description

The function `shield` is made to facilitate the propagation of attributes of an object `obj` through R operations.

Usage

```
shield(obj, at, propagate = "some", keep_also = NULL)
```

Arguments

obj	An object.
at	A named list, the attributes to be possibly added to obj.
propagate	character. The method to be applied, one of "all", "most", "some", "none", "many". If propagate="some" (the default), the attributes of obj are kept unchanged (up to the value of keep_also). If propagate="all" (not advised), the attributes of the returned object are exactly at (up to the value of keep_also). If propagate="none" (not advised either), the attributes of the returned object are NULL (up to the value of keep_also). If propagate="most", new attributes taken from at will be added to obj; however, attributes found in at that have the same name as attributes of obj are not considered.
keep_also	character. A vector of named attributes to be added to the final result.

Value

The object obj with possibly different attributes.

Examples

```
library(dplyr)
df <- data.frame(x = sample(10, 5, rep = TRUE),
                 y = sample(10, 5, rep = TRUE)) %>%
  at_mutate(example = "yes",
            package = "dplyr",
            class = c("my_tbl", "data.frame"))
tribe(df)

# Attributes are lost when the object passes through dplyr verbs
df2 <- df %>%
  mutate(z = 3)
tribe(df2)

# Most attributes are kept
df3 <- shield(df2, tribe(df), propagate = "most")
tribe(df3)

# To keep the class, use 'keep_also'
df4 <- shield(df2, tribe(df), propagate = "most", keep_also = "class")
tribe(df4)
```

`stick_to`*Work on a specific attribute within a pipeline*

Description

The functions `stick_to` and `unstick` enable to select an attribute within a pipe and work on it. It must be combined with the `%@>%` pipe to work properly, see the example below.

Usage

```
stick_to(obj, at)
```

```
stick_to_(obj, at)
```

```
unstick(x)
```

Arguments

<code>obj</code>	An object with an <code>at</code> attribute.
<code>at</code>	The name of the attribute to be considered.
<code>x</code>	An object to be unsticked. Must have <code>".obj_stick"</code> and <code>".at_stick"</code> attributes.

Value

`stick_to` basically inverses the roles of `.data` and `at`, meaning that `.data` becomes an attribute of the selected attribute. `unstick` makes the inverse operation.

Examples

```
## Not run:
library(dplyr)
library(observer)

df <- ggplot2::diamonds
  mutate(depth2 = 100*2*z/(x+y))
  observe_if(abs(depth-depth2) < 1)

observations(df)

df
  stick_to(observations)
  mutate(Id = 2)
  select(Id, Status)
  unstick()

observations(df)
```

```
## End(Not run)
```

tribe	<i>Object attribute list</i>
-------	------------------------------

Description

The function `tribe` is identical to `attributes`, expect that it always returns a named list (thus, when `attributes` will return `NULL`, `tribe` will return an empty named list).

Usage

```
tribe(obj, keep_obj = FALSE)
```

```
tribe(obj) <- value
```

```
untribe(x)
```

Arguments

<code>obj</code>	An object.
<code>keep_obj</code>	logical. If <code>TRUE</code> , <code>obj</code> is passed as an attribute to the result (useful in combination of <code>untribe</code>).
<code>value</code>	An appropriate named list of attributes, or <code>NULL</code> .
<code>x</code>	A list (of attributes) to be untribed.

Value

A named list, the attributes of `obj`.

See Also

[attributes](#), [attributes<-](#), [mostattributes<-](#).

Examples

```
## Not run:
library(lplyr)
A <- c(x = 1, y = 2, z = 3)
at_mutate(package = "trib?")
A
  tribe(keep_obj = TRUE)
  mutate(package = "tribe")
  untribe()

## End(Not run)
```

Index

`%try>%(make_pipe)`, 3

`at_mutate`, 2

`at_mutate_(at_mutate)`, 2

`at_rename(at_mutate)`, 2

`at_rename_(at_mutate)`, 2

`at_select(at_mutate)`, 2

`at_select_(at_mutate)`, 2

`at_slice(at_mutate)`, 2

`at_slice_(at_mutate)`, 2

`attributes`, 3, 7

`make_pipe`, 3

`shield`, 4, 4

`stick_to`, 6

`stick_to_(stick_to)`, 6

`structure`, 3

`tribe`, 7

`tribe<- (tribe)`, 7

`unstick(stick_to)`, 6

`untribe(tribe)`, 7