Package 'safeframe'

July 23, 2025

Title Generic Data Tagging and Validation Tool

Version 1.0.0

Description Provides tools to help tag and validate data according to user-specified rules. The 'safeframe' class adds variable level attributes to 'data.frame' columns. Once tagged, these variables can be seamlessly used in downstream analyses, making data pipelines clearer, more robust, and more reliable.

License MIT + file LICENSE

URL https://epiverse-trace.github.io/safeframe/, https://github.com/epiverse-trace/safeframe

BugReports https://github.com/epiverse-trace/safeframe/issues

Depends R (>= 4.1.0)

Imports checkmate, lifecycle, rlang, tidyselect

Suggests callr, dplyr, knitr, magrittr, rmarkdown, spelling, testthat, tibble

Config/Needs/website r-lib/pkgdown, epiverse-trace/epiversetheme

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

Language en-GB

RoxygenNote 7.3.2

NeedsCompilation no

Author Chris Hartgerink [cre, aut] (ORCID: <https://orcid.org/0000-0003-1050-6809>), Hugo Gruson [rev] (ORCID: <https://orcid.org/0000-0002-4094-1476>), data.org [cph]

Maintainer Chris Hartgerink <chris@data.org>

Repository CRAN

Date/Publication 2025-06-27 13:00:02 UTC

Contents

`has_tag` *A selector function to use in* **tidyverse** *functions*

Description

A selector function to use in **tidyverse** functions

Usage

```
has_tag(tags)
```

Arguments

`tags`	A character vector of tags you want to operate on

Value

A numeric vector containing the position of the columns with the requested tags

Examples

```
## create safeframe
x <- make_safeframe(cars,
  mph = "speed",
  distance = "dist"
)
head(x)

if (require(dplyr) && require(magrittr)) {
  x %>%
    select(has_tag(c("mph", "distance"))) %>%
    head()
```

```
}
```

`lost_tags_action` *Check and set behaviour for lost tags*

Description

This function determines the behaviour to adopt when tagged variables of a `safeframe` are lost for example through subsetting. This is achieved using `options` defined for the `safeframe` package.

Usage

```
lost_tags_action(action = c("warning", "error", "none"), quiet = FALSE)

get_lost_tags_action()
```

Arguments

`action`	a `character` indicating the behaviour to adopt when tagged variables have been lost: "error" (default) will issue an error; "warning" will issue a warning; "none" will do nothing
`quiet`	a `logical` indicating if a message should be displayed; only used outside pipelines

Details

The errors or warnings generated by safeframe in case of tagged variable loss has a custom class of `safeframe_error` and `safeframe_warning` respectively.

Value

returns `NULL`; the option itself is set in `options("safeframe")`

Examples

```
# reset default - done automatically at package loading
lost_tags_action()

# check current value
get_lost_tags_action()

# change to issue errors when tags are lost
lost_tags_action("error")
get_lost_tags_action()

# change to ignore when tags are lost
lost_tags_action("none")
get_lost_tags_action()

# reset to default: warning
```

```
lost_tags_action()
```

`make_safeframe` *Create a safeframe from a data.frame*

Description

This function converts a `data.frame` or a `tibble` into a `safeframe` object, where data are tagged and validated. The output will seem to be the same `data.frame`, but `safeframe`-aware packages will then be able to automatically use tagged fields for further data cleaning and analysis.

Usage

```
make_safeframe(.data, ...)
```

Arguments

`.data`	a `data.frame` or a `tibble`
`...`	<dynamic-dots> A series of tags provided as `tag_name = "column_name"`

Value

The function returns a `safeframe` object.

See Also

- An overview of the safeframe package
- `tags()`: for a list of tagged variables in a `safeframe`
- `set_tags()`: for modifying tags
- `tags_df()`: for selecting variables by tags

Examples

```
x <- make_safeframe(cars,
  mph = "speed",
  distance = "dist"
)

## print result - just first few entries
head(x)

## check tags
tags(x)

## tags can also be passed as a list with the splice operator (!!!)
my_tags <- list(
  mph = "speed",
```

```
  distance = "dist"
)
new_x <- make_safeframe(cars, !!!my_tags)

## The output is strictly equivalent to the previous one
identical(x, new_x)
```

`print.safeframe` *Printing method for safeframe objects*

Description

This function prints safeframe objects.

Usage

```
## S3 method for class 'safeframe'
print(x, ...)
```

Arguments

`x`	a `safeframe` object
`...`	further arguments to be passed to 'print'

Value

Invisibly returns the object.

Examples

```
## create safeframe
x <- make_safeframe(cars,
  mph = "speed",
  distance = "dist"
)

## print object - using only the first few entries
head(x)

# version with a tibble
if (require(tibble) && require(magrittr)) {
  cars %>%
    tibble() %>%
    make_safeframe(
      mph = "speed",
      distance = "dist"
    )
}
```

set_tags *Change tags of a safeframe object*

Description

This function changes the tags of a safeframe object, using the same syntax as the constructor make_safeframe().

Usage

```
set_tags(x, ...)
```

Arguments

x	a data.frame or a tibble, equivalent to parameter .data in make_safeframe()
...	<dynamic-dots> A series of tags provided as tag_name = "column_name"

Value

The function returns a safeframe object.

See Also

make_safeframe() to create a safeframe object

Examples

```
## create a safeframe
x <- make_safeframe(cars, mph = "speed")
tags(x)

## add new tags and fix an existing one
x <- set_tags(x, distance = "dist")
tags(x)

## remove tags by setting them to NULL
old_tags <- tags(x)
x <- set_tags(x, mph = NULL, distance = NULL)
tags(x)

## setting tags providing a list (used to restore old tags here)
x <- set_tags(x, !!!old_tags)
tags(x)
```

tags *Get the list of tags in a safeframe*

Description

This function returns the list of tags identifying specific variable types in a `safeframe` object.

Usage

```
tags(x, show_null = FALSE)
```

Arguments

`x`	a `safeframe` object
`show_null`	DEPRECATED

Details

tags are stored as the `label` attribute of the column variable.

Value

The function returns a named `list` where names indicate generic types of data, and values indicate which column they correspond to.

Examples

```
## make a safeframe
x <- make_safeframe(cars, mph = "speed")

## check non-null tags
tags(x)

## get a list of all tags, including NULL ones
tags(x, TRUE)
```

tags_df *Extract a data.frame of all tagged variables*

Description

This function returns a `data.frame`, where tagged variables (as stored in the `safeframe` object) are renamed. Note that the output is no longer a `safeframe`, but a regular `data.frame`. untagged variables are unaffected.

Usage

```
tags_df(x)
```

Arguments

x	a safeframe object

Value

A data.frame of with variables renamed according to their tags

Examples

```
x <- make_safeframe(cars,
  mph = "speed",
  distance = "dist"
)

## get a data.frame with variables renamed based on tags
tags_df(x)
```

---

type *Type Selection Helper*

Description

Function to swiftly provide access to generic categories of types within R. These can be used to provide comprehensive typesetting when creating a safeframe object.

Usage

```
type(x)
```

Arguments

x	Character indicating the desired type. Options include date, category, numeric, binary at this time.

Value

A vector of classes

Examples

```
x <- make_safeframe(cars,
  mph = "speed",
  distance = "dist"
)

validate_types(
  x,
  mph = type("numeric"),
  distance = "numeric"
)
```

`validate_safeframe` *Checks the content of a safeframe object*

Description

This function evaluates the validity of a `safeframe` object by checking the object class, its tags, and the types of variables. It combines validation checks made by `validate_types()` and `validate_tags()`. See 'Details' section for more information on the checks performed.

Usage

```
validate_safeframe(x, ...)
```

Arguments

`x`	a `safeframe` object
`...`	<dynamic-dots> A named list with tags in `x` as list names and the related types as list values.

Details

The following checks are performed:

- `x` is a `safeframe` object
- variables in `x` have a well-formed `label` attribute
- variables correspond to the specified types

Value

If checks pass, a `safeframe` object; otherwise issues an error.

See Also

- `validate_types()` to check if variables have the right types
- `validate_tags()` to perform a series of checks on the tags

Examples

```
## create a valid safeframe
x <- cars |>
  make_safeframe(
    mph = "speed",
    distance = "dist"
  )
x

## validation
validate_safeframe(x,
  mph = c("numeric", "factor"),
  distance = "numeric"
)

## the below issues an error
## note: tryCatch is only used to avoid a genuine error in the example
tryCatch(validate_safeframe(x,
  mph = c("numeric", "factor"),
  distance = "factor"
), error = paste)
```

`validate_tags` *Checks the tags of a safeframe object*

Description

This function evaluates the validity of the tags of a `safeframe` object by checking that: i) tags are present ii) tags is a `list` of `character` or `NULL` values.

Usage

```
validate_tags(x)
```

Arguments

`x`	a `safeframe` object

Value

If checks pass, a `safeframe` object; otherwise issues an error.

See Also

`validate_types()` to check if tagged variables have the right classes

Examples

```
## create a valid safeframe
x <- cars |>
  make_safeframe(
    mph = "speed",
    distance = "dist"
  )
x

## the below issues an error as safeframe doesn't know any defaults
## note: tryCatch is only used to avoid a genuine error in the example
tryCatch(validate_safeframe(x), error = paste)

## validation requires you to specify the types directly
validate_safeframe(x,
  mph = c("integer", "numeric"),
  distance = "numeric"
)
```

`validate_types` *Type check variables*

Description

This function checks the type of variables in a `safeframe` against accepted classes. Only checks the type of provided variables and ignores those not provided.

Usage

```
validate_types(x, ...)
```

Arguments

`x`	a `safeframe` object
`...`	<dynamic-dots> A named list with tags in `x` as list names and the related types as list values.

Value

A named `list`.

See Also

- `validate_tags()` to perform a series of checks on variables
- `validate_safeframe()` to combine `validate_tags` and `validate_types`

Examples

```
x <- make_safeframe(cars,
  mph = "speed",
  distance = "dist"
)
x

## the below would issue an error
## note: tryCatch is only used to avoid a genuine error in the example
tryCatch(validate_types(x), error = paste)

## to allow other types, e.g. gender to be integer, character or factor
validate_types(x, mph = "numeric", distance = c(
  "integer",
  "character", "numeric"
))
```

`vars_tags` *Internal printing function for variables and tags*

Description

Internal printing function for variables and tags

Usage

```
vars_tags(vars, tags)
```

Arguments

`vars`	a `character` vector of variable names
`tags`	a `character` vector of tags

`[.safeframe` *Subsetting of safeframe objects*

Description

The `[]` and `[[]]` operators for `safeframe` objects behaves like for regular `data.frame` or `tibble`, but check that tagged variables are not lost, and takes the appropriate action if this is the case (warning, error, or ignore, depending on the general option set via `lost_tags_action()`) .

Usage

```
## S3 method for class 'safeframe'
x[i, j, drop = FALSE]

## S3 replacement method for class 'safeframe'
x[i, j] <- value

## S3 replacement method for class 'safeframe'
x[[i, j]] <- value

## S3 replacement method for class 'safeframe'
x$name <- value
```

Arguments

x	a safeframe object
i	a vector of integer or logical to subset the rows of the safeframe
j	a vector of character, integer, or logical to subset the columns of the safeframe
drop	a logical indicating if, when a single column is selected, the data.frame class should be dropped to return a simple vector, in which case the safeframe class is lost as well; defaults to FALSE
value	the replacement to be used for the entries identified in x
name	a literal character string or a name (possibly backtick quoted). For extraction, this is normally (see under 'Environments') partially matched to the names of the object.

Value

If no drop is happening, a safeframe. Otherwise an atomic vector.

See Also

- lost_tags_action() to set the behaviour to adopt when tags are lost through subsetting; default is to issue a warning
- get_lost_tags_action() to check the current the behaviour

Examples

```
if (require(dplyr) && require(magrittr)) {
  ## create a safeframe
  x <- cars %>%
    make_safeframe(
      mph = "speed",
      distance = "dist"
    ) %>%
    mutate(result = if_else(speed > 50, "fast", "slow")) %>%
    set_tags(ticket = "result")
```

```
  x

  ## dangerous removal of a tagged column setting it to NULL issues warning
  x[, 1] <- NULL
  x

  x[[2]] <- NULL
  x

  x$speed <- NULL
  x
}
```

Index