

Package ‘linelist’

May 13, 2022

Title Tagging and Validating Epidemiological Data

Version 0.0.1

Description Provides tools to help storing and handling case line list data. The 'linelist' class adds a tagging system to classical 'data.frame' objects to identify key epidemiological data such as dates of symptom onset, epidemiological case definition, age, gender or disease outcome. Once tagged, these variables can be seamlessly used in downstream analyses, making data pipelines more robust and reliable.

License MIT + file LICENSE

URL <https://github.com/epiverse-trace/linelist>

BugReports <https://github.com/epiverse-trace/linelist/issues>

Encoding UTF-8

RoxygenNote 7.1.2

Config/testthat/edition 3

Imports checkmate

Suggests callr, covr, dplyr, knitr, magrittr, outbreaks, rmarkdown, testthat, tibble

VignetteBuilder knitr

NeedsCompilation no

Author Thibaut Jombart [aut, cre]

Maintainer Thibaut Jombart <thibaut@data.org>

Repository CRAN

Date/Publication 2022-05-13 16:30:02 UTC

R topics documented:

linelist	2
lost_tags_action	4
make_linelist	5
names<-linelist	7
print.linelist	8

rename.linelist	9
select.linelist	11
select_tags	12
set_tags	13
tags	15
tags_defaults	16
tags_df	16
tags_names	17
tags_types	18
validate_linelist	19
validate_tags	20
validate_types	21
[.linelist	22

Index	25
--------------	-----------

linelist	<i>Base Tools for Storing and Handling Case Line Lists</i>
----------	--

Description

The *linelist* package provides tools to help storing and handling case line list data. The `linelist` class adds a tagging system to classical `data.frame` or `tibble` objects which permits to identify key epidemiological data such as dates of symptom onset, epi case definition, age, gender or disease outcome. Once tagged, these variables can be seamlessly used in downstream analyses, making data pipelines more robust and reliable.

Main functions

- `make_linelist()`: to create `linelist` objects from a `data.frame` or a `tibble`, with the possibility to tag key epi variables
- `set_tags()`: to change or add tagged variables in a `linelist`
- `tags()`: to get the list of tags of a `linelist`
- `tags_df()`: to get a `data.frame` of all tagged variables
- `select_tags()`: like `dplyr::select()`, but for tagged variables
- `lost_tags_action()`: to change the behaviour of actions where tagged variables are lost (e.g. removing columns storing tagged variables) to issue warnings, errors, or do nothing
- `get_lost_tags_action()`: to check the current behaviour of actions where tagged variables are lost

Dedicated methods

Specific methods commonly used to handle `data.frame` are provided for `linelist` objects, typically to help flag or prevent actions which could alter or lose tagged variables (and may thus break downstream data pipelines).

- `names() <-` and `rename` (see `rename.linelist()`): will rename tags as needed

- `x[...] <-` and `x[[...]] <-` (see [sub_linelist](#)): will adopt the desired behaviour when tagged variables are lost
- `print()`: prints info about the linelist in addition to the `data.frame` or `tibble`

Author(s)

Thibaut Jombart <thibaut@data.org>

Examples

```
if (require(outbreaks)) {
  # using base R style

  ## dataset we'll create a linelist from, only using the first 50 entries
  measles_hagelloch_1861[1:50, ]

  ## create linelist
  x <- make_linelist(measles_hagelloch_1861[1:50, ],
                    id = "case_ID",
                    date_onset = "date_of_prodrome",
                    age = "age",
                    gender = "gender")

  x

  ## check tagged variables
  tags(x)

  ## extract tagged variables
  select_tags(x, "gender", "age")

  ## robust renaming
  names(x)[1] <- "identifier"
  x

  ## example of dropping tags by mistake - default: warning
  x[, 2:5]

  ## to silence warnings when tags are dropped
  lost_tags_action("none")
  x[, 2:5]

  ## to trigger errors when tags are dropped
  # lost_tags_action("error")
  # x[, 2:5]

  ## reset default behaviour
  lost_tags_action()

  # using tidyverse style
```

```

## example of creating a linelist, adding a new variable, and adding a tag
## for it

if (require(dplyr) && require(magrittr)) {
  x <- measles_hagelloch_1861 %>%
    tibble() %>%
    make_linelist(id = "case_ID",
                  date_onset = "date_of_prodrome",
                  age = "age",
                  gender = "gender") %>%
    mutate(result = if_else(is.na(date_of_death), "survived", "died")) %>%
    set_tags(outcome = "result") %>%
    rename(identifier = case_ID)

  x

  x %>%
    tags()

  x %>%
    select(starts_with("date"))

  ## disable warnings on the fly
  x %>%
    lost_tags_action("none") %>%
    select(starts_with("date"))

}
}

```

lost_tags_action

Check and set behaviour for lost tags

Description

This function determines the behaviour to adopt when tagged variables of a linelist are lost e.g. through subsetting. This is achieved using options defined for the `linelist` package. The function can be used in isolation, but it can also accept a dataset as first argument, so that it can be used in pipelines as well.

Usage

```

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

get_lost_tags_action()

```

Arguments

x	either a character, or an optional linelist object; if a character, it needs matching warning (default), error or none (see below)
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

Value

if a linelist is provided, it returns the object unchanged; otherwise, returns NULL; the option itself is set in options("linelist")

Author(s)

Thibaut Jombart <thibaut@data.org>

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_linelist

Create a linelist from a data.frame

Description

This function converts a `data.frame` or a `tibble` into a `linelist` object, where different types of epidemiologically relevant data are tagged. This includes dates of different events (e.g. onset of symptom, case reporting), information on the patient (e.g. age, gender, location) as well as other informations such as the type of case (e.g. confirmed, probable) or the outcome of the disease. The output will seem to be the same `data.frame`, but `linelist`-aware packages will then be able to automatically use tagged fields for further data cleaning and analysis.

Usage

```
make_linelist(x, ..., allow_extra = FALSE)
```

Arguments

<code>x</code>	a <code>data.frame</code> or a <code>tibble</code> containing case line list data, with cases in rows and variables in columns
<code>...</code>	a series of tags provided as <code>tag_name = "column_name"</code> , where <code>tag_name</code> indicates any of the known variables listed in 'Details'; alternatively, a named list of variables to be tagged, where names indicate the types of variable (to be selected from <code>tags_names()</code>), and values indicate their name in the input <code>data.frame</code> ; see details for a list of known variable types and their expected content
<code>allow_extra</code>	a logical indicating if additional data tags not currently recognized by <code>linelist</code> should be allowed; if <code>FALSE</code> , unknown tags will trigger an error

Details

Known variable types include:

- `id`: a unique case identifier as numeric or character
- `date_onset`: date of symptom onset (see below for date formats)
- `date_reporting`: date of case notification (see below for date formats)
- `date_admission`: date of hospital admission (see below for date formats)
- `date_discharge`: date of hospital discharge (see below for date formats)
- `date_outcome`: date of disease outcome (see below for date formats)
- `date_death`: date of death (see below for date formats)
- `gender`: a factor or character indicating the gender of the patient
- `age`: a numeric indicating the age of the patient, in years
- `location`: a factor or character indicating the location of the patient
- `occupation`: a factor or character indicating the professional activity of the patient
- `hcw`: a logical indicating if the patient is a health care worker
- `outcome`: a factor or character indicating the outcome of the disease (death or survival)

Dates can be provided in the following formats/types:

- Date objects (e.g. using `as.Date` on a character with a correct date format); this is the recommended format
- `POSIXct`/`POSIXlt` objects (when a finer scale than days is needed)
- numeric values, typically indicating the number of days since the first case

Value

The function returns a `linelist` object.

Author(s)

Thibaut Jombart <thibaut@data.org>

See Also

- An overview of the [linelist](#) package
- [tags_names\(\)](#): for a list of known tag names
- [tags_types\(\)](#): for the associated accepted types/classes
- [tags\(\)](#): for a list of tagged variables in a linelist
- [set_tags\(\)](#): for modifying tags
- [tags_df\(\)](#): for selecting variables by tags

Examples

```
if (require(outbreaks)) {  
  
  ## dataset we will convert to linelist  
  head(measles_hagelloch_1861)  
  
  ## create linelist  
  x <- make_linelist(measles_hagelloch_1861,  
                    id = "case_ID",  
                    date_onset = "date_of_prodrome",  
                    age = "age",  
                    gender = "gender")  
  
  ## print result - just first few entries  
  head(x)  
  
  ## check tags  
  tags(x)  
}
```

names<-.linelist	<i>Rename columns of a linelist</i>
------------------	-------------------------------------

Description

This function can be used to rename the columns a linelist, adjusting tags as needed.

Usage

```
## S3 replacement method for class 'linelist'  
names(x) <- value
```

Arguments

x a linelist object
value a character vector to set the new names of the columns of x

Value

a linelist with new column names

Author(s)

Thibaut Jombart <thibaut@data.org>

See Also

[rename.linelist\(\)](#) for renaming columns as with `dplyr::rename()`

Examples

```
if (require(outbreaks)) {  
  
  ## dataset to create a linelist from  
  measles_hagelloch_1861  
  
  ## create linelist  
  x <- make_linelist(measles_hagelloch_1861,  
                    id = "case_ID",  
                    date_onset = "date_of_prodrôme",  
                    age = "age",  
                    gender = "gender")  
  
  head(x)  
  
  ## change names  
  names(x)[1] <- "case_label"  
  
  ## see results: tags have been updated  
  head(x)  
  tags(x)  
}
```

print.linelist

Printing method for linelist objects

Description

This function prints linelist objects.

Usage

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


Arguments

x a linelist object
... further arguments to be passed to 'print'

Value

Invisibly returns the object.

Author(s)

Thibaut Jombart <thibaut@data.org>

Examples

```
if (require(outbreaks)) {  
  
  ## dataset we'll create a linelist from  
  measles_hagelloch_1861  
  
  ## create linelist  
  x <- make_linelist(measles_hagelloch_1861,  
                    id = "case_ID",  
                    date_onset = "date_of_prodrome",  
                    age = "age",  
                    gender = "gender")  
  
  ## print object - using only the first few entries  
  head(x)  
  
  # version with a tibble  
  if (require(tibble) && require(magrittr)) {  
    measles_hagelloch_1861 %>%  
      tibble() %>%  
      make_linelist(id = "case_ID",  
                   date_onset = "date_of_prodrome",  
                   age = "age",  
                   gender = "gender")  
  }  
}
```

rename.linelist

Rename columns of a linelist object

Description

This function works similarly to `dplyr::rename` and can be used to rename the columns of a linelist. Tagged variables are updated as needed to match new column names.

Usage

```
## S3 method for class 'linelist'  
rename(.data, ...)
```

Arguments

```
.data      a linelist object  
...        the variables to rename, using dplyr compatible syntax
```

Value

The function returns a linelist with renamed columns.

Author(s)

Thibaut Jombart <thibaut@data.org>

See Also

- [select.linelist\(\)](#) for selecting variables and tags
- [select_tags\(\)](#) for selecting tags
- [tags_df\(\)](#) to return a data.frame or a tibble of all agged variables

Examples

```
if (require(outbreaks) && require(dplyr) && require(magrittr)) {  
  
  ## dataset to create a linelist from  
  head(measles_hagelloch_1861)  
  
  ## create linelist  
  x <- measles_hagelloch_1861 %>%  
    tibble() %>%  
    make_linelist(id = "case_ID",  
                  date_onset = "date_of_prodrôme",  
                  age = "age",  
                  gender = "gender")  
  
  x  
  
  ## change names  
  x <- x %>%  
    rename(sex = gender, case = case_ID)  
  
  ## see results: tags have been updated  
  x  
  tags(x)  
}
```

select.linelist	<i>Subset columns of a linelist object</i>
-----------------	--

Description

This function works similarly to `dplyr::select()` but can in addition refer to tagged variables through the `tags` argument. When variables are selected using both procedures, tagged variables are output as the last columns.

Usage

```
## S3 method for class 'linelist'  
select(.data, ..., tags = NULL)
```

Arguments

<code>.data</code>	a linelist object
<code>...</code>	the variables to select, using <code>dplyr</code> compatible syntax
<code>tags</code>	a character indicating tagged variables to select using tag names (see <code>tags_names()</code>) for default values; values can be named, in which case the output columns will be renamed accordingly (e.g. <code>onset = "date_onset"</code> will output a column named 'onset').

Value

The function returns a linelist with selected columns.

Author(s)

Thibaut Jombart <thibaut@data.org>

See Also

- `select_tags()` to select tags only
- `tags_df()` to return a `data.frame` of all tagged variables

Examples

```
if (require(outbreaks) && require(dplyr) && require(magrittr)) {  
  
  ## dataset to create a linelist from  
  head(measles_hagelloch_1861)  
  
  ## create linelist  
  x <- measles_hagelloch_1861 %>%  
    tibble() %>%  
    make_linelist(id = "case_ID",  
                  date_onset = "date_of_prodrome",
```

```
        age = "age",
        gender = "gender")
x

## change select all dates and some tags
x %>%
  select(contains("date"), tags = c("id", "age", "gender"))

## showing warnings when tags are lost
x %>%
  select(1:3)

## getting rid of warnings on the fly
x %>%
  lost_tags_action("none") %>%
  select(1:3)

## reset default behaviour
lost_tags_action()
}
```

select_tags

Extract tagged variables of a linelist object

Description

This function is used to create a data.frame of tagged variables from a linelist object, and runs `dplyr::select` on the output. It is equivalent to running successively `tags_df()` and `dplyr::select()` on a linelist object. Note that the output no longer is a linelist object, but a regular data.frame (or tibble).

Usage

```
select_tags(x, ...)
```

Arguments

`x` a linelist object
`...` the tagged variables to select, using `dplyr::select()` compatible terminology; see `tags_names()` for default values

Value

A data.frame of tagged variables.

Author(s)

Thibaut Jombart <thibaut@data.org>

See Also

- [tags\(\)](#) for existing tags in a linelist
- [tags_df\(\)](#) to get a data.frame of all tags

Examples

```
if (require(outbreaks)) {

  ## dataset we'll create a linelist from
  measles_hagelloch_1861

  ## create linelist
  x <- make_linelist(measles_hagelloch_1861,
                    id = "case_ID",
                    date_onset = "date_of_prodrôme",
                    age = "age",
                    gender = "gender")

  head(x)

  ## check tagged variables
  tags(x)

  ## extract tagged variables
  select_tags(x, "gender", "age")
}
```

 set_tags

Changes tags of a linelist object

Description

This function changes the tags of a linelist object, using the same syntax as the constructor [make_linelist\(\)](#). If some of the default tags are missing, they will be added to the final object.

Usage

```
set_tags(x, ..., allow_extra = FALSE)
```

Arguments

x	a data.frame or a tibble containing case line list data, with cases in rows and variables in columns
...	a series of tags provided as tag_name = "column_name", where tag_name indicates any of the known variables listed in 'Details'; alternatively, a named list of variables to be tagged, where names indicate the types of variable (to be selected from tags_names()), and values indicate their name in the input data.frame; see details for a list of known variable types and their expected content

`allow_extra` a logical indicating if additional data tags not currently recognized by `linelist` should be allowed; if FALSE, unknown tags will trigger an error

Value

The function returns a `linelist` object.

Author(s)

Thibaut Jombart <thibaut@data.org>

See Also

[make_linelist\(\)](#) to create a `linelist` object

Examples

```
if (require(outbreaks)) {
  ## create a linelist
  x <- make_linelist(measles_hagelloch_1861, date_onset = "date_of_rash")
  tags(x)

  ## add new tags and fix an existing one
  x <- set_tags(x, age = "age",
               gender = "gender",
               date_onset = "date_of_prodrome")
  tags(x)

  ## add non-default tags using allow_extra
  x <- set_tags(x, severe = "complications", allow_extra = TRUE)
  tags(x)

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

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

tags	<i>Get the list of tags in a linelist</i>
------	---

Description

This function returns the list of tags identifying specific variable types in a linelist.

Usage

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

Arguments

x	a linelist object
show_null	a logical indicating if the complete list of tags, including NULL ones, should be returned; if FALSE, only tags with a non-NULL value are returned; defaults to FALSE

Details

Tags are stored as the tags attribute of the object.

Value

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

Author(s)

Thibaut Jombart <thibaut@data.org>

Examples

```
if (require(outbreaks)) {  
  ## make a linelist  
  x <- make_linelist(measles_hagelloch_1861, date_onset = "date_of_prodrome")  
  
  ## check non-null tags  
  tags(x)  
  
  ## get a list of all tags, including NULL ones  
  tags(x, TRUE)  
}
```

tags_defaults	<i>Generate default tags for a linelist</i>
---------------	---

Description

This function returns a named list providing the default tags for a linelist object (all default to NULL).

Usage

```
tags_defaults()
```

Value

A named list.

Author(s)

Thibaut Jombart <thibaut@data.org>

Examples

```
tags_defaults()
```

tags_df	<i>Extract a data.frame of all tagged variables</i>
---------	---

Description

This function returns a data.frame of all the tagged variables stored in a linelist. Note that the output is no longer a linelist, but a regular data.frame.

Usage

```
tags_df(x)
```

Arguments

x a linelist object

Value

A data.frame of tagged variables.

Author(s)

Thibaut Jombart <thibaut@data.org>

Examples

```
if (require(outbreaks) && require(dplyr) && require(magrittr)) {  
  
  ## create a tibble linelist  
  x <- measles_hagelloch_1861 %>%  
    tibble() %>%  
    make_linelist(id = "case_ID",  
                 date_onset = "date_of_prodrome",  
                 age = "age",  
                 gender = "gender")  
  
  x  
  
  ## get a data.frame of all tagged variables  
  tags_df(x)  
}
```

tags_names

Get the list of tag names used in linelist

Description

This function returns the a character of all tag names used to designate specific variable types in a linelist.

Usage

```
tags_names()
```

Value

The function returns a character vector.

Author(s)

Thibaut Jombart <thibaut@data.org>

See Also

[tags_defaults\(\)](#) for a list of default values of the tags

Examples

```
tags_names()
```

tags_types	<i>List acceptable variable types for tags</i>
------------	--

Description

This function returns a named list providing the acceptable data types for the default tags. If no argument is provided, it returns default values. Otherwise, provided values will be used to define the defaults.

Usage

```
tags_types(..., allow_extra = FALSE)
```

Arguments

...	a series of tags provided as tag_name = "column_name", where tag_name indicates any of the known variables listed in 'Details'; alternatively, a named list of variables to be tagged, where names indicate the types of variable (to be selected from tags_names()), and values indicate their name in the input data.frame; see details for a list of known variable types and their expected content
allow_extra	a logical indicating if additional data tags not currently recognized by linelist should be allowed; if FALSE, unknown tags will trigger an error

Value

A named list.

Author(s)

Thibaut Jombart <thibaut@data.org>

See Also

- [tags_defaults\(\)](#) for the default tags
- [validate_types\(\)](#) uses [tags_types\(\)](#) for validating tags
- [validate_linelist\(\)](#) uses [tags_types\(\)](#) for validating tags

Examples

```
# list default values
tags_types()

# change existing values
tags_types(date_onset = "Date") # impose a Date class

# add new types e.g. to allow genetic sequences using ape's format
tags_types(sequence = "DNABin", allow_extra = TRUE)
```

validate_linelist *Checks the content of a linelist object*

Description

This function evaluates the validity of a linelist object by checking the object class, its tags, and the types of the tagged variables. It combines validations checks made by [validate_types\(\)](#) and [validate_tags\(\)](#). See 'Details' section for more information on the checks performed.

Usage

```
validate_linelist(x, allow_extra = FALSE, ref_types = tags_types())
```

Arguments

x	a linelist object
allow_extra	a logical indicating if additional data tags not currently recognized by linelist should be allowed; if FALSE, unknown tags will trigger an error
ref_types	a list providing allowed types for all tags, as returned by tags_types()

Details

The following checks are performed:

- x is a linelist object
- x has a well-formed tags attribute
- all default tags are present (even if NULL)
- all tagged variables correspond to existing columns
- all tagged variables have an acceptable class
- (optional) x has no extra tag beyond the default tags

Value

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

Author(s)

Thibaut Jombart <thibaut@data.org>

See Also

- [tags_types\(\)](#) to change allowed types
- [validate_types\(\)](#) to check if tagged variables have the right classes
- [validate_tags\(\)](#) to perform a series of checks on the tags

Examples

```

if (require(outbreaks) && require(dplyr) && require(magrittr)) {

  ## create a valid linelist
  x <- measles_hagelloch_1861 %>%
    tibble() %>%
    make_linelist(id = "case_ID",
                  date_onset = "date_of_prodrôme",
                  age = "age",
                  gender = "gender")

  x

  ## validation
  validate_linelist(x)

  ## create an invalid linelist - onset date is a factor
  x <- measles_hagelloch_1861 %>%
    tibble() %>%
    make_linelist(id = "case_ID",
                  date_onset = "gender",
                  age = "age")

  x

  ## the below issues an error
  ## note: tryCatch is only used to avoid a genuine error in the example
  tryCatch(validate_linelist(x), error = paste)
}

```

validate_tags

Checks the tags of a linelist object

Description

This function evaluates the validity of the tags of a `linelist` object by checking that: i) tags are present ii) tags is a list of character iii) that all default tags are present iv) tagged variables exist v) that no extra tag exists (if `allow_extra` is `FALSE`).

Usage

```
validate_tags(x, allow_extra = FALSE)
```

Arguments

<code>x</code>	a <code>linelist</code> object
<code>allow_extra</code>	a logical indicating if additional data tags not currently recognized by <code>linelist</code> should be allowed; if <code>FALSE</code> , unknown tags will trigger an error

Value

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

Author(s)

Thibaut Jombart <thibaut@data.org>

See Also

[validate_types\(\)](#) to check if tagged variables have the right classes

Examples

```
if (require(outbreaks) && require(dplyr) && require(magrittr)) {  
  
  ## create a valid linelist  
  x <- measles_hagelloch_1861 %>%  
    tibble() %>%  
    make_linelist(id = "case_ID",  
                 date_onset = "date_of_prodrome",  
                 age = "age",  
                 gender = "gender")  
  
  x  
  
  ## validation  
  validate_tags(x)  
  
  ## hack to create an invalid tags (missing defaults)  
  attr(x, "tags") <- list(id = "case_ID")  
  
  ## the below issues an error  
  ## note: tryCatch is only used to avoid a genuine error in the example  
  tryCatch(validate_tags(x), error = paste)  
}
```

validate_types

Check tagged variables are the right class

Description

This function checks the class of each tagged variable in a `linelist` against pre-defined accepted classes in [tags_types\(\)](#).

Usage

```
validate_types(x, ref_types = tags_types())
```

Arguments

`x` a linelist object
`ref_types` a list providing allowed types for all tags, as returned by `tags_types()`

Value

A named list.

Author(s)

Thibaut Jombart <thibaut@data.org>

See Also

- `tags_types()` to change allowed types
- `validate_tags()` to perform a series of checks on the tags
- `validate_linelist()` to combine `validate_tags` and `validate_types`

Examples

```
if (require(outbreaks) && require(dplyr) && require(magrittr)) {

  ## create an invalid linelist - gender is a numeric
  x <- measles_hagelloch_1861 %>%
    tibble() %>%
    make_linelist(id = "case_ID",
                  gender = "infector")
  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, tags_types(gender = c("integer", "character", "factor")))
}
```

[.linelist

Subsetting of linelist objets

Description

The `[]` and `[[[]]` operators for linelist 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 'linelist'
x[i, j, drop = FALSE]

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

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

Arguments

x	a linelist object
i	a vector of integer or logical to subset the rows of the linelist
j	a vector of character, integer, or logical to subset the columns of the linelist
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 linelist class is lost as well; defaults to FALSE
value	the replacement to be used for the entries identified in x

Value

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

Author(s)

Thibaut Jombart <thibaut@data.org>

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(outbreaks) && require(dplyr) && require(magrittr)) {
  ## create a linelist
  x <- measles_hagelloch_1861 %>%
    tibble() %>%
    make_linelist(id = "case_ID",
                  date_onset = "date_of_prodrome",
                  age = "age",
                  gender = "gender") %>%
    mutate(result = if_else(is.na(date_of_death), "survived", "died")) %>%
    set_tags(outcome = "result") %>%
    rename(identifier = case_ID)
```

```
x  
  
## dangerous removal of a tagged column setting it to NULL issues a warning  
x[, 1] <- NULL  
x  
  
x[[2]] <- NULL  
x  
}
```


Index

`[.linelist`, 22
`[<-.linelist(.linelist)`, 22
`[[<-.linelist(.linelist)`, 22

`dplyr::rename()`, 8
`dplyr::select`, 12
`dplyr::select()`, 2, 11, 12

`get_lost_tags_action`
 (`lost_tags_action`), 4
`get_lost_tags_action()`, 2, 23

`linelist`, 2, 7
`lost_tags_action`, 4
`lost_tags_action()`, 2, 22, 23

`make_linelist`, 5
`make_linelist()`, 2, 13, 14

`names<- .linelist`, 7

`print.linelist`, 8

`rename.linelist`, 9
`rename.linelist()`, 2, 8

`select.linelist`, 11
`select.linelist()`, 10
`select_tags`, 12
`select_tags()`, 2, 10, 11
`set_tags`, 13
`set_tags()`, 2, 7
`sub_linelist`, 3
`sub_linelist(.linelist)`, 22

`tags`, 15
`tags()`, 2, 7, 13
`tags_defaults`, 16
`tags_defaults()`, 17, 18
`tags_df`, 16
`tags_df()`, 2, 7, 10–13

`tags_names`, 17
`tags_names()`, 6, 7, 11–13, 18
`tags_types`, 18
`tags_types()`, 7, 18, 19, 21, 22

`validate_linelist`, 19
`validate_linelist()`, 18, 22
`validate_tags`, 20
`validate_tags()`, 19, 22
`validate_types`, 21
`validate_types()`, 18, 19, 21