Package: linelist (via r-universe)

October 9, 2024

Title Tagging and Validating Epidemiological Data

Version 1.1.4.9000

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://epiverse-trace.github.io/linelist/, https://github.com/epiverse-trace/linelist

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

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Config/testthat/edition 3

Config/testthat/parallel true

Config/Needs/README incidence2 (>= 2.1.1), ggplot2

Depends R (>= 3.1.0)

Imports checkmate, dplyr, lifecycle, rlang, tidyselect

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

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

VignetteBuilder knitr

Language en-GB

Repository https://epiverse-connect.r-universe.dev

RemoteUrl https://github.com/epiverse-trace/linelist

RemoteRef HEAD

RemoteSha f0b1cc25fd4848ca494877d20b4fa87511e6fb48

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 listing the variables you want to operate on

Value

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

Examples

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

  ## 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"
  )
  head(x)

  x %>%
    select(has_tag(c("id", "age"))) %>%
    head()
}
```

`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.

Usage

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

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
`x`	deprecated

Details

The errors or warnings generated by linelist in case of tagged variable loss has a custom class of `linelist_error` and `linelist_warning` respectively.

Value

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

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 symptoms, case reporting), information on the patient (e.g. age, gender, location) as well as other information 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

`x`	a `data.frame` or a `tibble` containing case line list data, with cases in rows and variables in columns
`...`	<dynamic-dots> A series of tags provided as `tag_name = "column_name"`, where `tag_name` indicates any of the known variables listed in 'Details' and values indicate their name in `x`; 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

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.

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)

  ## Tags can also be passed as a list with the splice operator (!!!)
  my_tags <- list(
    id = "case_ID",
    date_onset = "date_of_prodrome",
    age = "age",
    gender = "gender"
  )
  new_x <- make_linelist(measles_hagelloch_1861, !!!my_tags)

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

---

names<-.linelist *Rename columns of a linelist*

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

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_prodrome",
    age = "age",
    gender = "gender"
  )
  head(x)

  ## change names
  names(x)[1] <- "case_label"

  ## see results: tags have been updated
  head(x)
  tags(x)

 # This also works with using `dplyr::rename()` because it uses names<-()
 # under hood
 if (require(dplyr)) {
   x <- x %>%
     rename(case_id= case_label)
   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.

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"
      )
  }
}
```

---

select.linelist *Subset columns of a linelist object*

Description

[Superseded]

This function was deprecated to ensure full compatibility with the default dplyr::select() methods. The tag selection feature is now possible via the has_tag() selection helper.

Usage

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

Arguments

.data	a linelist object
...	the variables to select, using dplyr compatible syntax
tags	**[Deprecated]** It is now recommended to leverage the has_tag() selection helper rather than this argument.

Value

The function returns a linelist with selected columns.

See Also

- tags_df() to return a data.frame of all tagged variables

---

select_tags *Extract tagged variables of a linelist object*

Description

[Deprecated] This function was equivalent to running successively tags_df() and dplyr::select() on a linelist object. To encourage users to understand what is going on and in order to follow the software engineering good practice of providing just one way to do a given task, this function is now deprecated.

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.

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_prodrome",
    age = "age",
    gender = "gender"
  )
  head(x)

  ## check tagged variables
  tags(x)

  # DEPRECATED!
  select_tags(x, "gender", "age")

  # Instead, use:
  library(dplyr)
  x %>%
    tags_df() %>%
    select(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
`...`	<dynamic-dots> A series of tags provided as `tag_name = "column_name"`, where `tag_name` indicates any of the known variables listed in 'Details' and values indicate their name in `x`; 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.

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 *Get the list of tags in a linelist*

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.

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 *Generate default tags for a linelist*

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.

Examples

```
tags_defaults()
```

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

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.

Examples

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

  ## create a tibble linelist
  x <- measles_hagelloch_1861 %>%
    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.

See Also

`tags_defaults()` for a `list` of default values of the tags

Examples

```
tags_names()
```

---

tags_types *List acceptable variable types for tags*

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

`...`	<dynamic-dots> A series of tags provided as `tag_name = "column_name"`, where `tag_name` indicates any of the known variables listed in 'Details' and values indicate their name in `x`; 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`.

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 (invisibly); otherwise issues an error.

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(magrittr)) {

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

  ## validation
  validate_linelist(x)

  ## create an invalid linelist - onset date is a factor
  x <- measles_hagelloch_1861 %>%
    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

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

Value

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

See Also

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

Examples

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

  ## create a valid linelist
  x <- measles_hagelloch_1861 %>%
    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.

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(magrittr)) {

  ## create an invalid linelist - gender is a numeric
  x <- measles_hagelloch_1861 %>%
    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 objects*

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

## S3 replacement method for class 'linelist'
x$name <- 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
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 linelist. 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(outbreaks) && require(dplyr) && require(magrittr)) {
  ## create a linelist
  x <- measles_hagelloch_1861 %>%
    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

x$age <- NULL
x
}
```

Index