Package: IssueTrackeR (via r-universe)

October 22, 2026

Type Package

Title List Things to Do

Version 2.0.0.9000

Description Manage a 'GitHub' problem using R: wrangle issues, labels and milestones. It includes functions for storing, prioritizing (sorting), displaying, adding, deleting, and selecting (filtering) issues based on qualitative and quantitative information. Issues (labels and milestones) are written in lists and categorized into the S3 class to be easily manipulated as datasets in R.

License MIT + file LICENSE

URL https://github.com/TanguyBarthelemy/IssueTrackeR, https://tanguybarthelemy.github.io/IssueTrackeR/

BugReports https://github.com/TanguyBarthelemy/IssueTrackeR/issues

Depends R (>= 4.2.0)

Imports cli, crayon, gh (>= 1.5.0), gitlabr, yaml, tools, stats, utils, graphics, grDevices, withr, checkmate, zoo

Suggests knitr, rlang, rmarkdown, spelling, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

LazyData TRUE

Roxygen list(markdown = TRUE, roclets = c(``collate", ``rd", ``namespace", ``devtag::dev_roclet"))

Config/roxygen2/version 8.1.1

Config/Needs/build moodymudskipper/devtag

VignetteBuilder knitr

Config/pak/sysreqs git libicu-dev libssl-dev

Repository https://tanguybarthelemy.r-universe.dev

Date/Publication 2026-10-22 13:44:52 UTC

RemoteUrl https://github.com/TanguyBarthelemy/IssueTrackeR

RemoteRef HEAD

RemoteSha 58ab5997b95085bf4953b9da30323a7cfca3f068

Contents

append . . . 2
author_last_comment . . . 3
check_github_connection . . . 4
compute_contribution . . . 5
count_issues . . . 6
extract_nth . . . 7
extraction-issues . . . 8
get . . . 9
get_all_repos . . . 10
get_nbr_comments . . . 11
init_selector . . . 12
IssueTrackeR-options . . . 13
merge_selector . . . 14
new_issue . . . 15
new_issues . . . 17
plot-issues . . . 19
print-issues . . . 21
rbind-issues . . . 22
remove_nth . . . 23
sample-issues . . . 24
subset.IssuesTB . . . 25
summary . . . 26
unique-issues . . . 27
update_database . . . 28
with_comments . . . 30
with_labels . . . 30
with_text . . . 32
write . . . 33

Index **36**

---

append *Vector Merging*

Description

Add elements to a vector.

Usage

```
append(x, values, after = length(x))

## S3 method for class 'IssuesTB'
append(x, values, after = nrow(x))
```

Arguments

x	the vector the values are to be appended to.
values	a IssueTB or a IssuesTB object.
after	a subscript, after which the values are to be appended.

Value

A vector containing the values in x with the elements of values appended after the specified element of x.

References

Becker RA, Chambers JM, Wilks AR (1988). *The New S Language*. Chapman and Hall/CRC, London. ISBN 053409192X.

Examples

```
issues_selector <- init_selector(
    source = "Local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = NULL
)
my_issues <- get_issues(selector = issues_selector)
new_issues <- append(my_issues, my_issues)
```

---

author_last_comment *Name of last commentator*

Description

Retrieve the name of the last commentator

Usage

```
author_last_comment(x, ...)

## S3 method for class 'IssueTB'
author_last_comment(x, verbose = TRUE, ...)

## S3 method for class 'IssuesTB'
author_last_comment(x, verbose = TRUE, ...)
```

Arguments

x	An object of class IssueTB or IssuesTB.
...	Currently not used.
verbose	Boolean indicating whether to print additional information.

Value

A string with the name of the last person which leaves a comment. If there is no comments, it returns an empty string.

Examples

```
issues_selector <- init_selector(
    source = "Local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = NULL
)
my_issues <- get_issues(selector = issues_selector)
author_last_comment(my_issues)
author_last_comment(my_issues[1L, ])
```

---

check_github_connection

Check GitHub connection

Description

Check GitHub connection

Usage

```
check_github_connection(verbose = TRUE)
```

Arguments

verbose	Boolean. Should messages be printed?

Value

Boolean. TRUE if the connection is valid and rate limit is positive, FALSE otherwise.

Examples

```
# Check if we can connect to GitHub
check_github_connection()
```

compute_contribution *Who contributes?*

Description

Calculates the contributions of users to a project by counting the number of issues they opened, commented on, and closed.

Usage

```
compute_contribution(x, n = Inf)

## S3 method for class 'IssuesTB'
compute_contribution(x, n = Inf)

## Default S3 method:
compute_contribution(x, n)
```

Arguments

x	An object of class `IssuesTB`.
n	Integer. Maximum number of top contributors to display. Contributions from other users are grouped under "other". Default: `Inf` (shows all contributors individually).

Value

A matrix of class `ContributionsTB` with users as columns and contribution types (opened, commented, closed) as rows. Each cell contains the count of contributions for that user and type.

Examples

```
issues_selector <- init_selector(
    source = "Local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = NULL
)
my_issues <- get_issues(selector = issues_selector)

# Compute contributions for all users
contributions <- compute_contribution(my_issues)

# Compute contributions for top 5 users
top_contributions <- compute_contribution(my_issues, n = 5)
```

`count_issues` *Count the number of Issues*

Description

Generic function to count the number of issues in a list of issues.

Usage

```
count_issues(x, ...)

## S3 method for class 'IssuesTB'
count_issues(x, verbose = TRUE, ...)

## Default S3 method:
count_issues(...)
```

Arguments

`x`	An object of class `IssuesTB`.
`...`	Currently not used.
`verbose`	Boolean indicating whether to print additional information.

Value

Integer. The number of issues.

Examples

```
issues_selector <- init_selector(
    source = "Local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = NULL
)
my_issues <- get_issues(selector = issues_selector)

count_issues(my_issues)
```

extract_nth — *Extract the nth Item from an List*

Description

Extract the nth issue from a `IssuesTB` object or the nth selector from a `SelectorTB` object.

Usage

```
extract_nth(x, ...)

## S3 method for class 'IssuesTB'
extract_nth(x, n, verbose = TRUE, ...)

## S3 method for class 'SelectorTB'
extract_nth(x, n, verbose = TRUE, ...)

## Default S3 method:
extract_nth(...)
```

Arguments

`x`	An object of class `IssuesTB` or `SelectorTB`.
`...`	Currently not used.
`n`	Integer. Position of the element to extract. 1 is for the first element.
`verbose`	Boolean indicating whether to print additional information.

Value

If `x` is a `IssuesTB`, it returns the nth issue as a `IssueTB` object. If `x` is a `SelectorTB`, it returns the nth selector as a `SelectorTB` object. If `n` exceeds the number of issues or selector, returns the last issue / selector with a warning.

Returns `NULL` if the list is empty.

Examples

```
issues_selector <- init_selector(
    source = "Local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = NULL
)

my_issues <- get_issues(selector = issues_selector)

first_issue <- extract_nth(my_issues, 1)
third_issue <- extract_nth(my_issues, 3)
```

extraction-issues *Extraction and replacement of information in issues*

Description

Extraction and replacement of information in issues

Usage

```
## S3 method for class 'IssuesTB'
x[i, j, drop = TRUE]

## S3 replacement method for class 'IssuesTB'
x[...] <- value

## S3 replacement method for class 'IssuesTB'
x[[...]] <- value
```

Arguments

x	An object of class IssuesTB.
i	Integer. Indices of the issues to extract.
j	Optional. Indices or names for column selection.
drop	Boolean. If TRUE, drop IssuesTB structure. Default: TRUE.
...	Other parameters passed to extraction method
value	Replacement value. According to the column selected, it can be a numeric, a character, an integer or a timestamp (POSIXct).

Value

Information inside the IssuesTB object

Examples

```
issues_selector <- init_selector(
    source = "Local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = NULL
)
open_issues <- get_issues(selector = issues_selector)

first_issue <- open_issues[1, ]
number <- open_issues[1, 1]
state <- open_issues[1, "state"]
open_issues[1, "state"] <- "closed"
open_issues[["number"]] <- seq_along(open_issues[["number"]])
```

get *Retrieve information from the issues of GitHub*

Description

use gh to ask the API of GitHub and et a list of issues with their labels and milestones.

Usage

```
get_issues(selector = getOption("IssueTrackeR.selector"), verbose = TRUE, ...)

get_labels(selector = getOption("IssueTrackeR.selector"), verbose = TRUE, ...)

get_milestones(
  selector = getOption("IssueTrackeR.selector"),
  verbose = TRUE,
  ...
)
```

Arguments

selector	A SelectorTB object to check. See the corresponding vignettes: https://tanguybarthelemy.github.io/IssueTrackeR/articles/selectors.html to have more information about selectors. By default getOption("IssueTrackeR.selector").
verbose	Boolean indicating whether to print additional information.
...	Additional arguments to connect to the GitLab as gitlab_url and private_token. (See the documentation of get to have more information on theses parameters)

Details

The functions of get type are useful to retrieve object related to issues from GitHub. So it's possible to retrieve issues, labels and milestones.

The defaults value for the argument dataset_name depends on the function:

- defaults is "list_issues.yaml" for get_issues()
- defaults is "list_milestones.yaml" for get_milestones()
- defaults is "list_labels.yaml" for get_labels()

Value

The function get_issues returns an object of class IssuesTB. It is a list composed by object of class IssueTB. An object of class IssueTB represents an issue with simpler structure (with number, title, body and labels).

The function get_labels returns a list representing labels with simpler structure (with name, description, colour).

The function get_milestones returns a list representing milestones with simpler structure (with title, description and due_on).

Examples

```
if (check_github_connection(verbose = FALSE)) {
    # From online

    online_selector <- init_selector(
        source = "GitHub",
        owner = "TanguyBarthelemy",
        repo = "IssueTrackeR"
    )
    issues <- get_issues(selector = online_selector)
    print(issues)

    labels <- get_labels(selector = online_selector)
    print(labels)

    milestones <- get_milestones(selector = online_selector)
    print(milestones)
}


# From local

local_selector <- init_selector(
    source = "Local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = NULL
)

issues <- get_issues(selector = local_selector)
labels <- get_labels(selector = local_selector)
milestones <- get_milestones(selector = local_selector)
```

`get_all_repos` *Retrieve all the visible repos from a user / an organisation*

Description

Returns a list of repos.

Usage

```
get_all_repos(owner, public = TRUE, private = TRUE, verbose = TRUE)
```

Arguments

`owner`	Character. GitHub repository owner . GitHub owner (user or organization).
`public`	Boolean. Should we include public repos? (Default `TRUE`)
`private`	Boolean. Should we include private repos? (Default `TRUE`)
`verbose`	Boolean indicating whether to print additional information.

Value

A string with the list of repo of a user or an organisation.

Examples

```
if (check_github_connection(verbose = FALSE)) {
    get_all_repos("rjdverse")
}
```

---

get_nbr_comments *Number of comments*

Description

Number of comments of an Issue or set of issues

Usage

```
get_nbr_comments(x)

## S3 method for class 'IssueTB'
get_nbr_comments(x)

## S3 method for class 'IssuesTB'
get_nbr_comments(x)
```

Arguments

x	An object of class IssueTB or IssuesTB.

Value

An integer or an integer vector with the number of comments of the different issues in x.

Examples

```
issues_selector <- init_selector(
    source = "Local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = NULL
)

my_issues <- get_issues(selector = issues_selector)

get_nbr_comments(my_issues)
get_nbr_comments(my_issues[1L, ])
```

init_selector — *Initialise a* `SelectorTB` *object*

Description

Creates a `SelectorTB` object for specifying data sources (GitHub, GitLab, or local). The selector can be used to fetch issues from different sources.

Usage

```
init_selector(source = NULL, ...)
```

Arguments

`source` a character string that is either:

- `"GitHub"` if you want to fetch information from GitHub
- `"GitLab"` if you want to fetch information from GitLab
- or `"local"` if you want to fetch information locally.

`...` Additional arguments specific to the source type:

- For GitHub: `owner`, `repo`
- For GitLab: `project_id`
- For local: `dataset_name`, `dataset_dir`

Details

`SelectorTB` objects define data sources for retrieving issues from GitHub, GitLab, or local YAML files. They allow combining multiple sources into a single query using `merge_selector()`.

See the corresponding vignettes: https://tanguybarthelemy.github.io/IssueTrackeR/articles/selectors.html to have more information about selectors. Also see `vignette("selectors")`

Value

A `SelectorTB` object configured for the specified source.

Examples

```
# GitHub selector for a specific repository
gh_sel <- init_selector(
    source = "GitHub",
    owner = "TanguyBarthelemy",
    repo = "IssueTrackeR"
)

# GitLab selector for multiple projects
gl_sel <- init_selector(source = "GitLab", project_id = c(4578, 2384))

# Local selector for a dataset
```

```
local_sel <- init_selector(
    source = "local",
    dataset_name = "my_dataset",
    dataset_dir = "/path/to/data"
)
```

IssueTrackeR-options *Set, get and reset options*

Description

Options to manage the default configuration of datasets and selectors.

Usage

```
reset_options(verbose = TRUE)

set_default_dataset_dir(path)

get_default_dataset_dir()

set_default_dataset_name(name)

get_default_dataset_name()

set_default_selector(selector)

get_default_selector()
```

Arguments

verbose	Boolean indicating whether to print additional information.
path	Character. Representing a directory path.
name	Character. Representing the name of the option.
selector	An object of class SelectorTB.

Value

- get_...: The value of the requested option.
- set_...: The value assigned, invisibly.
- reset_options: NULL invisibly.

Examples

```
# Resetting options
reset_options()

# Working with default dataset directory
get_default_dataset_dir()
set_default_dataset_dir("/tmp/my_data")

# Working with default dataset name
get_default_dataset_name()
set_default_dataset_name("my_dataset")

# Working with default selector
sel <- init_selector(source = "GitHub", owner = "test", repo = "test")
set_default_selector(sel)
get_default_selector()
```

---

merge_selector *Merge Multiple Selectors*

Description

Combines multiple selector objects into a single selector. Useful for fetching issues from multiple sources in one operation.

Usage

```
merge_selector(...)
```

Arguments

`...`	`SelectorTB` objects to merge.

Value

A merged `SelectorTB` object containing all elements from the input selectors.

Examples

```
s1 <- init_selector(
    source = "GitHub",
    owner = "TanguyBarthelemy",
    repo = "IssueTrackeR"
)

s2 <- init_selector(source = "GitHub", owner = "TractorTom", repo = NULL)
merged <- merge_selector(s1, s2)
```

new_issue *Create a new* IssueTB *object*

Description

Create a new IssueTB object

Usage

```
new_issue(x = NULL, ...)

## S3 method for class 'IssueTB'
new_issue(x, ...)

## S3 method for class 'data.frame'
new_issue(x, ...)

## S3 method for class 'list'
new_issue(x, ...)

## S3 method for class 'IssuesTB'
new_issue(x, ...)

## Default S3 method:
new_issue(
  x,
  title = NA_character_,
  body = NA_character_,
  number = NA_integer_,
  state = NA_character_,
  created_at = format_timestamp(NA_integer_),
  closed_at = format_timestamp(NA_integer_),
  closed_by = NA_character_,
  labels = NULL,
  comments = NULL,
  milestone = NA_character_,
  repo = NA_character_,
  owner = NA_character_,
  url = NA_character_,
  html_url = NA_character_,
  creator = NA_character_,
  assignee = NA_character_,
  state_reason = NA_character_,
  ...
)
```

Arguments

x	a object representing an issue (IssueTB object, a list or a data.frame)
...	Currently not used.
title	a string. The title of the issue.
body	a string. The body (text) of the issue.
number	a string. The number of the issue.
state	a string that is either "open" (by default) if the issue is still open or "closed" if the issue is now closed.
created_at	a date (or timestamp). The creation date of the issue.
closed_at	a date (or timestamp). The closing date of the issue.
closed_by	a string. The GitHub username of the person who closed the issue.
labels	a vector string (or NULL). The labels of the issue.
comments	vector of string (the comments of the issue)
milestone	a string (or NULL). The milestone of the issue.
repo	Character. Repository name.
owner	Character. Owner name.
url	a string. The URL of the API to the GitHub issue.
html_url	a string. The URL to the GitHub issue.
creator	a string. The GitHub username of the creator of the issue.
assignee	a string. The GitHub username of the assignee of the issue.
state_reason	a string. "open", "completed", "reopened", "not_planned" or "duplicate".

Value

a IssueTB object.

Examples

```
# Custom issue
issue <- new_issue(
    title = "Nouvelle issue",
    body = "Un nouveau bug pour la fonction...",
    number = 47L,
    created_at = Sys.Date()
)

dput(issue)

issue2 <- new_issue(x = issue)
```

new_issues *Create a new* IssuesTB *object*

Description

Create a new IssuesTB object

Usage

```
new_issues(x = NULL, ...)

## S3 method for class 'IssueTB'
new_issues(x, ...)

## S3 method for class 'IssuesTB'
new_issues(x, ...)

## S3 method for class 'data.frame'
new_issues(x, ...)

## S3 method for class 'list'
new_issues(x, ...)

## S3 method for class '`NULL`'
new_issues(x, ...)

## Default S3 method:
new_issues(
  x,
  title = NULL,
  body = NULL,
  number = NULL,
  state = NULL,
  created_at = format_timestamp(NA_integer_),
  closed_at = format_timestamp(NA_integer_),
  closed_by = NA_character_,
  labels = list(),
  comments = list(),
  milestone = NA_character_,
  repo = NA_character_,
  owner = NA_character_,
  url = NA_character_,
  html_url = NA_character_,
  creator = NA_character_,
  assignee = NA_character_,
  state_reason = NA_character_,
  ...
```

)

Arguments

x	a object representing a list of issues (IssuesTB object, a list or a data.frame) or a single issue (IssueTB class)
...	Currently not used.
title	a vector of string. The titles of the issues.
body	a vector of string. The bodies (text) of the issues.
number	a vector of string. The numbers of the issues.
state	a vector of string that is either "open" (by default) if the issues are still open or "closed" if the issues are now closed.
created_at	a vector of date (or timestamp). The creation date of the issues.
closed_at	a vector of date (or timestamp). The closing date of the issues.
closed_by	a vector of string. The usernames of the person who closed the issues.
labels	a list of vector string (or NULL). The labels of the issues.
comments	a list of vector string. The comments of the issues.
milestone	a vector of string (or NULL). The milestones of the issues.
repo	Character. Repositories name.
owner	Character string containing the owners name.
url	a vector of string. The URLs of the API to the issues.
html_url	a vector of string. The URLs to the issues.
creator	a vector of string. The usernames of the creator of the issues.
assignee	a vector of string. The usernames of the assignee of the issues.
state_reason	a vector of string. "open", "completed", "reopened", "not_planned" or "duplicate".

Value

a IssuesTB object.

Examples

```
# Empty list of issues
issues1 <- new_issues()

# List of issues from issue
issue1 <- new_issue(
    title = "Une autre issue",
    state = "open",
    body = "J'ai une question au sujet de...",
    number = 2L,
    created_at = Sys.Date()
)
issues2 <- new_issues(x = issue1)
```

```
# Custom issues
issues3 <- new_issues(
    title = "Une autre issue",
    state = "open",
    body = "J'ai une question au sujet de...",
    number = 2L,
    created_at = Sys.Date()
)

issues4 <- new_issues(
    title = c("Nouvelle issue", "Une autre issue"),
    body = c("Un nouveau bug pour la fonction...",
             "J'ai une question au sujet de..."),
    state = c("open", "closed"),
    number = 1:2,
    created_at = c(Sys.Date() - 30, Sys.Date())
)
```

`plot-issues` *Plot an* `IssuesTB` *object*

Description

Visualize the evolution of an issue tracker backlog.

Two types of plots are available:

- `"historic"`: displays the distribution of open issues by age.
- `"created-closed"`: displays backlog size together with the numbers of newly created and newly closed issues.

Usage

```
## S3 method for class 'IssuesTB'
plot(
  x,
  type = c("historic", "area-chart", "created-closed", "resolution-time"),
  by = NULL,
  n = 3L,
  ...
)

## S3 method for class 'ContributionsTB'
plot(x, ...)
```

Arguments

x	An object of class IssuesTB.
type	Character string indicating which plot to produce. Accepted values are "historic" and "created-closed". The default is "historic".
by	Character. The category. Default: "creator".
n	Integer specifying the number of age classes. Default: 3.
...	Currently not used.

Details

When type = "historic", a stacked area chart is produced showing the number of open issues by age over time. This visualization highlights the evolution and aging of the backlog.

The first classes correspond to one-year intervals (0-1y, 1-2y, ..., (n-1)-ny) and the last class groups all issues older than n years.

When type = "area-chart", the same graph as type = "historic" but this time with the number of issues by categories over time. The argument by is mandatory to specify which category to display.

When type = "created-closed", the total number of open issues is displayed together with the monthly numbers of newly created and newly closed issues. This visualization helps assess whether issue creation and resolution rates are balanced over time.

All statistics are aggregated monthly, from the month of the first issue creation to the current date.

When type = "resolution-time", the resolution times are computed and displayed in two forms:

- bar plot with categories from time
- ECDF to show the cumulative distribution of issues resolution times on a log scale

n is only used when type = "historic".

Value

Invisibly returns x.

Examples

```
issues_selector <- init_selector(
    source = "Local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = NULL
)
my_issues <- get_issues(selector = issues_selector)

plot(my_issues, type = "historic")
plot(my_issues, type = "created-closed")
plot(my_issues, type = "resolution-time")
plot(my_issues, type = "area-chart", by = "closed_by")
plot(my_issues, type = "area-chart", by = "milestone", n = 5)
plot(my_issues, type = "area-chart", by = "state_reason", n = 5)
```

```
plot(my_issues, type = "area-chart", by = "repo", n = 5)
```

---

print-issues *Display* IssueTB, IssuesTB, LabelsTB *or* SelectorTB *object*

Description

Display IssueTB, IssuesTB, LabelsTB or SelectorTB with formatted output in the console

Usage

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

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

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

## S3 method for class 'summary.IssueTB'
print(x, ...)

## S3 method for class 'summary.IssuesTB'
print(x, ...)

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

## S3 method for class 'summary.LabelsTB'
print(x, ...)

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

Arguments

x	An object of class IssueTB, IssuesTB, LabelsTB or SelectorTB.
...	Currently not used.

Details

This function displays an issue (IssueTB object), a list of issues (IssuesTB object) , a list of labels (LabelsTB object) or a list of selectors (SelectorTB objects) with a formatted output.

Value

The object x invisibly.

Examples

```
issues_selector <- init_selector(
    source = "Local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = NULL
)
my_issues <- get_issues(selector = issues_selector)

# Display one issue
print(my_issues[1, ])

# Display several issues
print(my_issues[1:10, ])

# Display a selector
print(issues_selector)

# Display the summary of one issue
summary(my_issues[2, ])

# Display the summary of
summary(my_issues[1:10, ])
```

---

rbind-issues *Combining Issues*

Description

S3 method for combining IssueTB objects by rows.

Usage

```
## S3 method for class 'IssueTB'
rbind(...)

## S3 method for class 'IssuesTB'
rbind(...)
```

Arguments

...	Objects of class IssueTB or IssuesTB.

Value

An IssueTB object containing the combined rows of all input objects.

See Also

rbind for the generic function

Examples

```
issues_selector <- init_selector(
    source = "Local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = NULL
)
open_issues <- get_issues(selector = issues_selector)
new_issues <- rbind(open_issues[1, ], open_issues[-1, ])
issues_selector <- init_selector(
    source = "Local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = NULL
)
open_issues <- get_issues(selector = issues_selector)
new_issues <- rbind(open_issues, open_issues)
```

---

remove_nth *Remove the nth Item from an List*

Description

Remove the nth issue from a IssuesTB object or the nth selector from a SelectorTB object.

Usage

```
remove_nth(x, ...)

## S3 method for class 'IssuesTB'
remove_nth(x, n, verbose = TRUE, ...)

## S3 method for class 'SelectorTB'
remove_nth(x, n, verbose = TRUE, ...)

## Default S3 method:
remove_nth(...)
```

Arguments

x	An object of class IssuesTB or SelectorTB.
...	Currently not used.
n	Integer. Position of the element to remove. 1 is for the first element.
verbose	Boolean indicating whether to print additional information.

Value

If x is a IssuesTB, it returns the list of issues (a IssuesTB object) without the nth issue. If x is a SelectorTB, it returns the list of selector (a SelectorTB object) without the nth selector If n exceeds the number of issues or selector, returns the initial list with a warning.

Examples

```
issues_selector <- init_selector(
    source = "Local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = NULL
)
my_issues <- get_issues(selector = issues_selector)
issues_wo_1 <- remove_nth(my_issues, 1)
issues_wo_3 <- remove_nth(my_issues, 3)
```

---

sample-issues *Random Sampling*

Description

Generic function for drawing a random sample from an object.

For objects of class IssuesTB, this method returns a random subset of the issues.

Usage

```
sample(x, size, replace = FALSE, prob = NULL)

## S3 method for class 'IssuesTB'
sample(x, size = nrow(x), replace = FALSE, prob = NULL)

## Default S3 method:
sample(x, size, replace = FALSE, prob = NULL)
```

Arguments

x	An object of class IssuesTB.
size	a non-negative integer giving the number of items to choose.
replace	should sampling be with replacement?
prob	a vector of probability weights for obtaining the elements of the vector being sampled.

Details

The arguments and overall behavior are consistent with base::sample(). For details about the sampling algorithm, probability weights, and special cases, refer to the original documentation: https://stat.ethz.ch/R-manual/R-devel/library/base/html/sample.html

Value

- For `IssuesTB` objects, an object of the same class containing the sampled issues.
- For all other objects, the result of `base::sample()`.

See Also

`base::sample()`

Examples

```
issues_selector <- init_selector(
    source = "Local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = NULL
)
open_issues <- get_issues(selector = issues_selector)
new_issues <- sample(open_issues, size = 5L)
```

---

subset.IssuesTB *Subsetting Vectors, Matrices and Data Frames*

Description

Return subsets of vectors, matrices or data frames which meet conditions.

Usage

```
## S3 method for class 'IssuesTB'
subset(x, ...)
```

Arguments

`x`	object to be subsetted.
`...`	further arguments to be passed to or from other methods.

Details

This is a generic function, with methods supplied for matrices, data frames and vectors (including lists). Packages and users can add further methods.

For ordinary vectors, the result is simply `x[subset & !is.na(subset)]`.

For data frames, the `subset` argument works on the rows. Note that `subset` will be evaluated in the data frame, so columns can be referred to (by name) as variables in the expression (see the examples).

The `select` argument exists only for the methods for data frames and matrices. It works by first replacing column names in the selection expression with the corresponding column numbers in the data frame and then using the resulting integer vector to index the columns. This allows the use of

the standard indexing conventions so that for example ranges of columns can be specified easily, or single columns can be dropped (see the examples).

The `drop` argument is passed on to the indexing method for matrices and data frames: note that the default for matrices is different from that for indexing.

Factors may have empty levels after subsetting; unused levels are not automatically removed. See `droplevels` for a way to drop all unused levels from a data frame.

Value

An object similar to `x` contain just the selected elements (for a vector), rows and columns (for a matrix or data frame), and so on.

Warning

This is a convenience function intended for use interactively. For programming it is better to use the standard subsetting functions like `[`, and in particular the non-standard evaluation of argument `subset` can have unanticipated consequences.

Author(s)

Peter Dalgaard and Brian Ripley

See Also

`[`, `transform` `droplevels`

Examples

```
issues_selector <- init_selector(
    source = "Local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = NULL
)
open_issues <- get_issues(selector = issues_selector)
new_issues <- subset(open_issues, number < 150)
```

`summary` *Compute a summary of an issue or a list of issues*

Description

Compute a summary of an issue or a list of issues

Usage

```
## S3 method for class 'IssueTB'
summary(object, ...)

## S3 method for class 'IssuesTB'
summary(object, with_labels = FALSE, ...)

## S3 method for class 'LabelsTB'
summary(object, ...)
```

Arguments

`object`	a `IssueTB` or `IssuesTB` object.
`...`	Currently not used.
`with_labels`	Boolean. Display the labels with the list of issues. Default is `FALSE`.

Details

This function compute the summary of an issue (`IssueTB` object) with adding some information (number of comments, ...). For a list of issues (`IssuesTB` object), it just summarise the information with statistics by modalities.

Value

invisibly (with `invisible()`) `NULL`.

Examples

```
issues_selector <- init_selector(
    source = "Local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = NULL
)
my_issues <- get_issues(selector = issues_selector)

# Summarise one issue
summary(my_issues[1, ])

# Summarise several issues
summary(my_issues[1:10, ])
```

`unique-issues` *Unique issues of an* `IssuesTB` *Object*

Description

Keep only different issues from a `IssuesTB` Object

Usage

```
## S3 method for class 'IssuesTB'
unique(x, incomparables = FALSE, ...)
```

Arguments

x	An object of class IssuesTB.
incomparables	a vector of values that cannot be compared. FALSE is a special value, meaning that all values can be compared, and may be the only value accepted for methods other than the default. It will be coerced internally to the same type as x.
...	arguments for particular methods.

Details

This method is consistent with base::unique(). For details about the generic function and its default methods, refer to the original documentation: https://stat.ethz.ch/R-manual/R-devel/library/base/html/unique.html

Value

An IssuesTB object containing only unique issues.

See Also

base::unique(), base::duplicated()

Examples

```
issues_selector <- init_selector(
    source = "Local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = NULL
)
open_issues <- get_issues(selector = issues_selector)
new_issues <- unique(open_issues)
```

---

update_database *Update database*

Description

Update the different local database (issues, labels and milestones) with the online reference.

Usage

```
update_database(
  selector = getOption("IssueTrackeR.selector"),
  dataset_dir = getOption("IssueTrackeR.dataset.dir"),
  dataset_name = getOption("IssueTrackeR.dataset.name"),
  verbose = TRUE,
  ...
)
```

Arguments

selector	A SelectorTB object to check. See the corresponding vignettes: https://tanguybarthelemy.github.io/IssueTrackeR/articles/selectors.html to have more information about selectors. By default getOption("IssueTrackeR.selector").
dataset_dir	Character. Path to the directory containing the YAML file. Default: value of option IssueTrackeR.dataset.dir.
dataset_name	Character. Name of the dataset. Default: value of option IssueTrackeR.dataset.name.
verbose	Boolean indicating whether to print additional information.
...	Currently not used.

Value

invisibly (with invisible()) TRUE.

Examples

```
if (check_github_connection(verbose = FALSE)) {
    jdemetra_selector <- init_selector(
        source = "GitHub",
        owner = "jdemetra",
        repo = "jdplus-incubator",
        state = "all"
    )

    update_database(
        selector = jdemetra_selector,
        dataset_dir = tempdir()
    )
}
```

with_comments — *Check for comments in GitHub Issues*

Description

Function to filter issues with (or without) comments.

Usage

```
with_comments(x, ...)

## S3 method for class 'IssuesTB'
with_comments(x, negate = FALSE, ...)
```

Arguments

x	An object of class IssuesTB.
...	Currently not used.
negate	boolean indicating if we are searching for issues WITHOUT comments. Default is FALSE.

Value

An object IssuesTB with issues that satisfy the condition.

Examples

```
issues_selector <- init_selector(
    source = "Local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = NULL
)
my_issues <- get_issues(selector = issues_selector)
with_comments(my_issues)
with_comments(my_issues, negate = TRUE)
```

with_labels — *Check for labels in GitHub Issues*

Description

Generic function to filter issues with labels

Usage

```
with_labels(x, ...)

## S3 method for class 'IssuesTB'
with_labels(x, ...)
```

Arguments

x — An object of class IssuesTB.

... — Arguments passed on to base::grepl

- pattern character string containing a regular expression (or character string for fixed = TRUE) to be matched in the given character vector. Coerced by as.character to a character string if possible. If a character vector of length 2 or more is supplied, the first element is used with a warning. Missing values are allowed except for regexpr, gregexpr and regexec.
- x,text a character vector where matches are sought, or an object which can be coerced by as.character to a character vector. Long vectors are supported.
- ignore.case logical. if FALSE, the pattern matching is *case sensitive* and if TRUE, case is ignored during matching.
- perl logical. Should Perl-compatible regexps be used?
- value logical. If FALSE, a vector containing the (integer) indices of the matches determined by grep is returned, and if TRUE, a vector containing the matching elements themselves is returned.
- fixed logical. If TRUE, pattern is a string to be matched as is. Overrides all conflicting arguments.
- useBytes logical. If TRUE the matching is done byte-by-byte rather than character-by-character. See 'Details'.
- invert logical. If TRUE return indices or values for elements that do *not* match.
- replacement a replacement for the matched pattern in sub and gsub. Coerced to character if possible. For fixed = FALSE this can include backreferences "\1" to "\9" to parenthesized subexpressions of pattern. For perl = TRUE only, it can also contain "\U" or "\L" to convert the rest of the replacement to upper or lower case and "\E" to end case conversion. If a character vector of length 2 or more is supplied, the first element is used with a warning. If NA, all elements in the result corresponding to matches will be set to NA.

Value

An object IssuesTB with issues that satisfy the condition.

Examples

```
issues_selector <- init_selector(
    source = "Local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = NULL
```

```
)
my_issues <- get_issues(selector = issues_selector)
with_labels(my_issues, pattern = "Bug")
```

`with_text` *Check for text in GitHub Issues*

Description

Generic function to filter issues with a given text pattern in the title, body, or comments of a GitHub Issue object or a collection of Issues.

Usage

```
with_text(x, ...)

## S3 method for class 'IssuesTB'
with_text(x, ..., in_title = TRUE, in_body = TRUE, in_comments = TRUE)
```

Arguments

`x` An object of class `IssuesTB`.

`...` Arguments passed on to `base::grepl`

`pattern` character string containing a regular expression (or character string for `fixed = TRUE`) to be matched in the given character vector. Coerced by `as.character` to a character string if possible. If a character vector of length 2 or more is supplied, the first element is used with a warning. Missing values are allowed except for `regexpr`, `gregexpr` and `regexec`.

`x,text` a character vector where matches are sought, or an object which can be coerced by `as.character` to a character vector. Long vectors are supported.

`ignore.case` logical. if `FALSE`, the pattern matching is *case sensitive* and if `TRUE`, case is ignored during matching.

`perl` logical. Should Perl-compatible regexps be used?

`value` logical. If `FALSE`, a vector containing the (`integer`) indices of the matches determined by `grep` is returned, and if `TRUE`, a vector containing the matching elements themselves is returned.

`fixed` logical. If `TRUE`, `pattern` is a string to be matched as is. Overrides all conflicting arguments.

`useBytes` logical. If `TRUE` the matching is done byte-by-byte rather than character-by-character. See 'Details'.

`invert` logical. If `TRUE` return indices or values for elements that do *not* match.

replacement a replacement for the matched pattern in sub and gsub. Coerced to character if possible. For fixed = FALSE this can include backreferences "\1" to "\9" to parenthesized subexpressions of pattern. For perl = TRUE only, it can also contain "\U" or "\L" to convert the rest of the replacement to upper or lower case and "\E" to end case conversion. If a character vector of length 2 or more is supplied, the first element is used with a warning. If NA, all elements in the result corresponding to matches will be set to NA.

in_title Boolean. Does the function search for text in the title? (Default TRUE)

in_body Boolean. Does the function search for text in the body? (Default TRUE)

in_comments Boolean. Does the function search for text in the comments? (Default TRUE)

Value

An object IssuesTB with issues that satisfy the condition.

Examples

```
issues_selector <- init_selector(
    source = "Local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = NULL
)
my_issues <- get_issues(selector = issues_selector)
with_text(my_issues, pattern = "Excel")
```

---

write *Save datasets in a yaml file*

Description

Save datasets in a yaml file

Usage

```
write(x, ...)

## S3 method for class 'IssuesTB'
write(
  x,
  dataset_dir = getOption("IssueTrackeR.dataset.dir"),
  dataset_name = getOption("IssueTrackeR.dataset.name"),
  overwrite = TRUE,
  verbose = TRUE,
  ...
)
```

```
## S3 method for class 'LabelsTB'
write(
  x,
  dataset_dir = getOption("IssueTrackeR.dataset.dir"),
  dataset_name = getOption("IssueTrackeR.dataset.name"),
  overwrite = TRUE,
  verbose = TRUE,
  ...
)

## S3 method for class 'MilestonesTB'
write(
  x,
  dataset_dir = getOption("IssueTrackeR.dataset.dir"),
  dataset_name = getOption("IssueTrackeR.dataset.name"),
  overwrite = TRUE,
  verbose = TRUE,
  ...
)

## Default S3 method:
write(...)
```

Arguments

`x`	an object of class `IssuesTB`, `LabelsTB` or `MilestonesTB`.
`...`	Currently not used.
`dataset_dir`	Character. Path to the directory containing the YAML file. Default: value of option `IssueTrackeR.dataset.dir`.
`dataset_name`	Character. Name of the dataset. Default: value of option `IssueTrackeR.dataset.name`.
`overwrite`	Boolean indicating whether to overwrite the file if it already exists. Defaults to `TRUE`.
`verbose`	Boolean indicating whether to print additional information.

Details

Depending on the object, `dataset_name = NULL` or `dataset_name = ""`, the defaults value of the file name:

- `"list_issues.yaml"` for issues;
- `"list_labels.yaml"` for labels;
- `"list_milestones.yaml"` for milestones.

The objects will be written in the `dataset_dir`.

Value

invisibly (with `invisible()`) `TRUE` if the export was successful and an error otherwise.

Examples

```
local_selector <- init_selector(
    source = "Local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = NULL
)

issues <- get_issues(selector = local_selector)
labels <- get_labels(selector = local_selector)
milestones <- get_milestones(selector = local_selector)

write(x = issues, dataset_dir = tempdir())
write(x = labels, dataset_dir = tempdir())
write(x = milestones, dataset_dir = tempdir())

write(x = issues, dataset_dir = tempdir(),
                  dataset_name = "my_issues")
write(x = labels, dataset_dir = tempdir(),
                  dataset_name = "my_labels")
write(x = milestones, dataset_dir = tempdir(),
                  dataset_name = "my_milestones")
```

Index

[, *26*
[.IssuesTB *(extraction-issues)*, 8
[<-.IssuesTB *(extraction-issues)*, 8
[[<-.IssuesTB *(extraction-issues)*, 8

append, 2
as.character, *31, 32*
author_last_comment, 3

base::duplicated(), *28*
base::grepl, *31, 32*
base::sample(), *24, 25*
base::unique(), *28*

check_github_connection, 4
compute_contribution, 5
count_issues, 6

droplevels, *26*

extract_nth, 7
extraction-issues, 8

get, 9, *9*
get_all_repos, 10
get_default_dataset_dir *(IssueTrackeR-options)*, 13
get_default_dataset_name *(IssueTrackeR-options)*, 13
get_default_selector *(IssueTrackeR-options)*, 13
get_issues *(get)*, 9
get_labels *(get)*, 9
get_milestones *(get)*, 9
get_nbr_comments, 11
gh, *9*

init_selector, 12
IssueTrackeR-options, 13

Long vectors, *31, 32*

merge_selector, 14

new_issue, 15
new_issues, 17

plot-issues, 19
plot.ContributionsTB *(plot-issues)*, 19
plot.IssuesTB *(plot-issues)*, 19
print-issues, 21
print.ContributionsTB *(print-issues)*, 21
print.IssuesTB *(print-issues)*, 21
print.IssueTB *(print-issues)*, 21
print.LabelsTB *(print-issues)*, 21
print.SelectorTB *(print-issues)*, 21
print.summary.IssuesTB *(print-issues)*, 21
print.summary.IssueTB *(print-issues)*, 21
print.summary.LabelsTB *(print-issues)*, 21

rbind, *23*
rbind-issues, 22
rbind.IssuesTB *(rbind-issues)*, 22
rbind.IssueTB *(rbind-issues)*, 22
regular expression, *31, 32*
remove_nth, 23
reset_options *(IssueTrackeR-options)*, 13

sample *(sample-issues)*, 24
sample-issues, 24
sample.default *(sample-issues)*, 24
sample.IssuesTB *(sample-issues)*, 24
set_default_dataset_dir *(IssueTrackeR-options)*, 13
set_default_dataset_name *(IssueTrackeR-options)*, 13
set_default_selector *(IssueTrackeR-options)*, 13
subset.IssuesTB, 25
summary, 26

transform, *26*

unique-issues, 27
unique.IssuesTB *(*unique-issues*)*, 27
update_database, 28

with_comments, 30
with_labels, 30
with_text, 32
write, 33