Package 'IssueTrackeR'

August 21, 2026

Type Package

Title List Things to Do

Version 1.5.0

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), yaml, tools, stats, utils, graphics, grDevices, withr, checkmate, zoo

Suggests rlang, spelling, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

Config/roxygen2/version 8.1.0

Config/Needs/build moodymudskipper/devtag

NeedsCompilation no

Author Tanguy Barthelemy [aut, cre, art, cph]

Maintainer Tanguy Barthelemy <tanguy.barthelemy@insee.fr>

Repository CRAN

Date/Publication 2026-08-21 05:46:20 UTC

Contents

---

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

```
path <- system.file("data_issues", package = "IssueTrackeR")
open_issues <- get_issues(
    source = "local",
    dataset_dir = path,
    dataset_name = "open_issues.yaml"
)
closed_issues <- get_issues(
    source = "local",
    dataset_dir = path,
    dataset_name = "closed_issues.yaml"
)
new_issues <- append(open_issues, closed_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	A logical value indicating whether to print additional information. Default is TRUE.

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

```
all_issues <- get_issues(
    source = "local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = "open_issues.yaml"
)
author_last_comment(all_issues)
author_last_comment(all_issues[1L, ])
```

---

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	A logical value indicating whether to print additional information. Default is TRUE.

Value

Integer. The number of issues.

Examples

```
all_issues <- get_issues(
    source = "local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = "open_issues.yaml"
)

count_issues(all_issues)
```

`extract_nth` *Extract the nth Issue from an List of Issues*

Description

Extract the nth issue from a `IssuesTB` object.

Usage

```
extract_nth(x, ...)

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

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

Arguments

`x`	An object of class `IssuesTB`.
`...`	Currently not used.
`n`	Integer. Position of the element to extract. 1 is for the first element.
`verbose`	A logical value indicating whether to print additional information. Default is `TRUE`.

Value

The nth issue as a `IssueTB` object. If `n` exceeds the number of issues, returns the last issue with a warning. Returns `NULL` if the issues list is empty.

Examples

```
all_issues <- get_issues(
    source = "local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = "open_issues.yaml"
)
```

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

---

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(
  source = c("local", "online"),
  dataset_dir = getOption("IssueTrackeR.dataset.dir"),
  dataset_name = "open_issues.yaml",
  repo = getOption("IssueTrackeR.repo"),
  owner = getOption("IssueTrackeR.owner"),
  state = c("open", "closed", "all"),
  verbose = TRUE
)

get_labels(
  source = c("local", "online"),
  dataset_dir = getOption("IssueTrackeR.dataset.dir"),
  dataset_name = "list_labels.yaml",
  repo = getOption("IssueTrackeR.repo"),
  owner = getOption("IssueTrackeR.owner"),
  verbose = TRUE
)

get_milestones(
  source = c("local", "online"),
  dataset_dir = getOption("IssueTrackeR.dataset.dir"),
  dataset_name = "list_milestones.yaml",
  repo = getOption("IssueTrackeR.repo"),
  owner = getOption("IssueTrackeR.owner"),
  state = c("open", "closed", "all"),
  verbose = TRUE
)
```

Arguments

source	a character string that is either "online" if you want to fetch information from GitHub or "local" (by default) if you want to fetch information locally.

dataset_dir	A character string specifying the path which contains the datasets (only taken into account if source is set to "local"). Defaults to the package option IssueTrackeR.dataset.dir.
dataset_name	A character string specifying the name of the datasets which will be written (only taken into account if source is set to "local"). Defaults to "open_issues.yaml".
repo	A character string specifying the GitHub repository name (only taken into account if source is set to "online"). Defaults to the package option IssueTrackeR.repo.
owner	A character string specifying the GitHub owner (only taken into account if source is set to "online"). Defaults to the package option IssueTrackeR.owner.
state	a character string that is either "open" (by default) if you want to fetch only open issues from GitHub, "closed" if you want to fetch only closed issues from GitHub or "all" if you want to fetch all issues from GitHub (closed and open). Only taken into account if source is set to "online".
verbose	A logical value indicating whether to print additional information. Default is TRUE.

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 (gh::gh_token_exists() && gh::gh_rate_limit()$remaining > 0) {
    # From online

    issues <- get_issues(source = "online", owner = "rjdverse", repo = NULL)
    issues <- get_issues(source = "online")
    print(issues)

    labels <- get_labels(source = "online")
    print(labels)
```

```
    milestones <- get_milestones(source = "online")
    print(milestones)
}

# From local

path <- system.file("data_issues", package = "IssueTrackeR")
issues <- get_issues(
    source = "local",
    dataset_dir = path,
    dataset_name = "open_issues.yaml"
)
milestones <- get_milestones(
    source = "local",
    dataset_dir = path,
    dataset_name = "list_milestones.yaml"
)
labels <- get_labels(
    source = "local",
    dataset_dir = path,
    dataset_name = "list_labels.yaml"
)
```

`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`	A character string specifying the GitHub owner (only taken into account if `source` is set to `"online"`). Defaults to the package option `IssueTrackeR.owner`.
`public`	Boolean. Should we include public repos? (Default `TRUE`)
`private`	Boolean. Should we include private repos? (Default `TRUE`)
`verbose`	A logical value indicating whether to print additional information. Default is `TRUE`.

Value

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

Examples

```
if (gh::gh_token_exists() && gh::gh_rate_limit()$remaining > 0) {
    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

```
all_issues <- get_issues(
    source = "local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = "open_issues.yaml"
)
get_nbr_comments(all_issues)
get_nbr_comments(all_issues[1L, ])
```

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 = as.Date(NA_integer_),
  closed_at = as.Date(NA_integer_),
  closed_by = NA_character_,
  labels = NULL,
  milestone = NA_character_,
  repo = NA_character_,
  owner = NA_character_,
  url = NA_character_,
  html_url = NA_character_,
  comments = NULL,
  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)
...	Other information we would like to add to the issue.
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 missing). The labels of the issue.
milestone	a string (or missing). The milestone of the issue.
repo	A character string specifying the GitHub repository name (only taken into account if source is set to "online"). Defaults to the package option IssueTrackeR.repo.
owner	A character string specifying the GitHub owner (only taken into account if source is set to "online"). Defaults to the package option IssueTrackeR.owner.
url	a string. The URL of the API to the GitHub issue.
html_url	a string. The URL to the GitHub issue.
comments	vector of string (the comments of the 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

```
# Empty issue
issue1 <- new_issue()

# Custom issue
issue1 <- new_issue(
    title = "Nouvelle issue",
    body = "Un nouveau bug pour la fonction...",
    number = 47L,
    created_at = Sys.Date()
)

issue2 <- new_issue(x = issue1)
```

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, ...)

## Default S3 method:
new_issues(
  x,
  title,
  body,
  number,
  state,
  created_at = as.Date(NA_integer_),
  closed_at = as.Date(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)
...	Other information we would like to add to the issue.
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 GitHub usernames of the person who closed the issues.
labels	a list of vector string (or missing). The labels of the issues.
comments	a list of vector string. The comments of the issues.
milestone	a vector of string (or missing). The milestones of the issues.
repo	A character string specifying the GitHub repository name (only taken into account if source is set to "online"). Defaults to the package option IssueTrackeR.repo.
owner	A character string specifying the GitHub owner (only taken into account if source is set to "online"). Defaults to the package option IssueTrackeR.owner.
url	a vector of string. The URLs of the API to the GitHub issues.
html_url	a vector of string. The URLs to the GitHub issues.
creator	a vector of string. The GitHub usernames of the creator of the issues.
assignee	a vector of string. The GitHub 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", "author", "created-closed", "resolution-time"),
  n = 3L,
  ...
)
```

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"`.
`n`	Integer specifying the number of age classes to display when `type = "historic"`.
`...`	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 = "author"`, the same graph as `type = "historic"` but this time with the number of open issues by author over time.

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

Value

Invisibly returns `x`.

Examples

```
all_issues <- rbind(
    get_issues(
        source = "local",
        dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
        dataset_name = "open_issues.yaml"
    ),
    get_issues(
        source = "local",
        dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
        dataset_name = "closed_issues.yaml"
    )
)

plot(all_issues, type = "historic")
plot(all_issues, type = "author")
plot(all_issues, type = "created-closed")
plot(all_issues, type = "resolution-time")
```

---

print-issues *Display IssueTB and IssuesTB object*

Description

Display IssueTB and IssuesTB 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 '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, ...)
```

Arguments

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

Details

This function displays an issue (IssueTB object) or a list of issues (IssuesTB object) with a formatted output.

Value

invisibly (with invisible()) NULL.

Examples

```
all_issues <- get_issues(
    source = "local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = "open_issues.yaml"
)

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

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

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

```
# Display the summary of
summary(all_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

```
path <- system.file("data_issues", package = "IssueTrackeR")
open_issues <- get_issues(
    source = "local",
    dataset_dir = path,
    dataset_name = "open_issues.yaml"
)
new_issues <- rbind(open_issues[1, ], open_issues[-1, ])
path <- system.file("data_issues", package = "IssueTrackeR")
open_issues <- get_issues(
    source = "local",
    dataset_dir = path,
    dataset_name = "open_issues.yaml"
)
closed_issues <- get_issues(
    source = "local",
    dataset_dir = path,
    dataset_name = "closed_issues.yaml"
)
new_issues <- rbind(open_issues, closed_issues)
```

reset_options *Reset options*

Description

Reset options

Usage

```
reset_options(verbose = TRUE)
```

Arguments

verbose	A logical value indicating whether to print additional information. Default is TRUE.

Value

NULL invisibly

Examples

```
getOption("IssueTrackeR.owner")
reset_options()
getOption("IssueTrackeR.owner")
```

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

```
path <- system.file("data_issues", package = "IssueTrackeR")
open_issues <- get_issues(
    source = "local",
    dataset_dir = path,
    dataset_name = "open_issues.yaml"
)
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	An object of class IssuesTB.
...	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

```
path <- system.file("data_issues", package = "IssueTrackeR")
open_issues <- get_issues(
    source = "local",
    dataset_dir = path,
    dataset_name = "open_issues.yaml"
)
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

```
all_issues <- get_issues(
    source = "local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = "open_issues.yaml"
)

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

# Summarise several issues
summary(all_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

```
path <- system.file("data_issues", package = "IssueTrackeR")
open_issues <- get_issues(
    source = "local",
    dataset_dir = path,
    dataset_name = "open_issues.yaml"
)
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(
  dataset_dir = getOption("IssueTrackeR.dataset.dir"),
 datasets_name = c(open = "open_issues.yaml", closed = "closed_issues.yaml", labels =
    "list_labels.yaml", milestones = "list_milestones.yaml"),
  verbose = TRUE,
  ...
)
```

Arguments

dataset_dir	A character string specifying the path which contains the datasets (only taken into account if source is set to "local"). Defaults to the package option IssueTrackeR.dataset.dir.
datasets_name	A named character string of length 4, specifying the names of the different datasets which will be written. The names datasets_name have to be "open", "closed", "labels" and "milestones". Defaults to c(open = "open_issues.yaml", closed = "closed_issues.yaml", labels = "list_labels.yaml", milestones = "list_milestones.yaml") .
verbose	A logical value indicating whether to print additional information. Default is TRUE.
...	Additional arguments for connecting to the GitHub repository: • repo A character string specifying the GitHub repository name. Defaults to the package option IssueTrackeR.repo. • owner A character string specifying the GitHub owner. Defaults to the package option IssueTrackeR.owner. (See the documentation of get to have more information on theses parameters):

Value

invisibly (with invisible()) TRUE.

Examples

```
if (gh::gh_token_exists() && gh::gh_rate_limit()$remaining > 0) {
    update_database(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

```
all_issues <- get_issues(
    source = "local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = "open_issues.yaml"
)
with_comments(all_issues)
with_comments(all_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.
...	Additional arguments passed to grepl(), such as pattern and ignore.case.

Value

An object IssuesTB with issues that satisfy the condition.

Examples

```
all_issues <- get_issues(
    source = "local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = "open_issues.yaml"
)
with_labels(all_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.
...	Additional arguments passed to grepl(), such as pattern and ignore.case.
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

```
all_issues <- get_issues(
    source = "local",
    dataset_dir = system.file("data_issues", package = "IssueTrackeR"),
    dataset_name = "open_issues.yaml"
)
with_text(all_issues, pattern = "Excel")
```

---

write_to_dataset *Save datasets in a yaml file*

Description

Save datasets in a yaml file

Usage

```
write_to_dataset(x, ...)

## S3 method for class 'IssuesTB'
write_to_dataset(
  x,
  dataset_dir = getOption("IssueTrackeR.dataset.dir"),
  dataset_name = "list_issues.yaml",
  overwrite = TRUE,
  verbose = TRUE,
  ...
)

## S3 method for class 'LabelsTB'
write_to_dataset(
  x,
  dataset_dir = getOption("IssueTrackeR.dataset.dir"),
  dataset_name = "list_labels.yaml",
  overwrite = TRUE,
  verbose = TRUE,
  ...
)

## S3 method for class 'MilestonesTB'
write_to_dataset(
  x,
```

```
  dataset_dir = getOption("IssueTrackeR.dataset.dir"),
  dataset_name = "list_milestones.yaml",
  overwrite = TRUE,
  verbose = TRUE,
  ...
)

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

Arguments

x	an object of class IssuesTB, LabelsTB or MilestonesTB.
...	Currently not used.
dataset_dir	A character string specifying the path which contains the datasets (only taken into account if source is set to "local"). Defaults to the package option IssueTrackeR.dataset.dir.
dataset_name	A character string specifying the name of the datasets which will be written (only taken into account if source is set to "local"). Defaults to "open_issues.yaml".
overwrite	Boolean. If the dataset file already exists, should it be overwrite? Default is TRUE.
verbose	A logical value indicating whether to print additional information. Default is TRUE.

Details

Depending on the object, the defaults value of the argument dataset_name (by default) is:

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

Value

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

Examples

```
path <- system.file("data_issues", package = "IssueTrackeR")
issues <- get_issues(
    source = "local",
    dataset_dir = path,
    dataset_name = "open_issues.yaml"
)
milestones <- get_milestones(
    source = "local",
    dataset_dir = path,
    dataset_name = "list_milestones.yaml"
```

```
)
labels <- get_labels(
    source = "local",
    dataset_dir = path,
    dataset_name = "list_labels.yaml"
)

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

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

Index