Package 'IssueTrackeR'

May 7, 2026

Type Package

Title List Things to Do

Version 1.3.1

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 (>= 4.1)

Imports cli, crayon, gh, yaml, tools, stats, utils, grDevices

Suggests rlang, spelling, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

RoxygenNote 7.3.3

NeedsCompilation no

Author Tanguy Barthelemy [aut, cre, art]

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

Repository CRAN

Date/Publication 2025-10-27 13:10:19 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, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

Examples

```
append(1:5, 0:1, after = 3)
```

`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)

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

Arguments

`x`	An object of class `IssueTB` or `IssuesTB`.

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, ])
```

format_issues *Format the issue in a simpler format*

Description

Format the issue in a simpler format

Usage

```
format_issues(raw_issues, raw_comments, verbose = TRUE)
```

Arguments

raw_issues	a gh_response object output from the function gh which contains all the data and metadata for GitHub issues.
raw_comments	a gh_response object output from the function gh which contains all the data and metadata for GitHub comments.
verbose	A logical value indicating whether to print additional information. Default is TRUE.

Value

a list representing an issue with simpler structure (with number, title, body and labels) of all issues.

Examples

```
## Not run:
raw_issues <- gh::gh(
    repo = "rjdemetra",
    owner = "rjdverse",
    endpoint = "/repos/:owner/:repo/issues",
    .limit = Inf
)
raw_comments <- gh::gh(
    repo = "rjdemetra",
    owner = "rjdverse",
    endpoint = "/repos/:owner/:repo/issues/comments",
    .limit = Inf
)
all_issues <- format_issues(raw_issues = raw_issues,
                            raw_comments = raw_comments,
                            verbose = FALSE)

## End(Not run)
```

format_labels *Format the label in a simpler format*

Description

Format the label in a simpler format

Usage

```
format_labels(raw_labels, verbose = TRUE)
```

Arguments

raw_labels	a gh_response object output from the function gh which contains all the data and metadata for GitHub labels.
verbose	A logical value indicating whether to print additional information. Default is TRUE.

Value

a list representing labels with simpler structure (with name, description, colour)

Examples

```
## Not run:
# With labels
raw_labels <- gh::gh(
   repo = "rjdemetra",
   owner = "rjdverse",
   endpoint = "/repos/:owner/:repo/labels",
   .limit = Inf
)
format_labels(raw_labels)

## End(Not run)
```

format_milestones *Format the milestones in a simpler format*

Description

Format the milestones in a simpler format

Usage

```
format_milestones(raw_milestones, verbose = TRUE)
```

Arguments

raw_milestones	a gh_response object output from the function gh which contains all the data and metadata for GitHub milestones.
verbose	A logical value indicating whether to print additional information. Default is TRUE.

Value

a list representing milestones with simpler structure (with title, description and due_on)

Examples

```
## Not run:
# With milestones
milestones_jdplus_main <- gh::gh(
    repo = "jdplus-main",
    owner = "jdemetra",
    endpoint = "/repos/:owner/:repo/milestones",
    state = "all",
    .limit = Inf
 )
format_milestones(milestones_jdplus_main)

## End(Not run)
```

---

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

```
## Not run:
get_all_repos("rjdverse")

## End(Not run)
```

`get_issues` *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

```
## Not run:
# 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)

## End(Not run)

# 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_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 = Sys.Date(),
  closed_at = as.Date(NA_integer_),
  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.
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 "duplicated".

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 = 47,
    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 = Sys.Date(),
  closed_at = as.Date(NA_integer_),
  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.
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 "duplicated".

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 = 2,
    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 = 2,
    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 = Sys.Date()
)
```

---

print.IssueTB *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	a IssueTB or IssuesTB object.
...	Unused argument

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, ])
```

`sample` *Random Samples and Permutations*

Description

`sample` takes a sample of the specified size from the elements of `x` using either with or without replacement.

Usage

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

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

Arguments

`x`	either a vector of one or more elements from which to choose, or a positive integer. See 'Details.'
`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

If `x` has length 1, is numeric (in the sense of `is.numeric`) and `x >= 1`, sampling *via* `sample` takes place from `1:x`. *Note* that this convenience feature may lead to undesired behaviour when `x` is of varying length in calls such as `sample(x)`. See the examples.

Otherwise `x` can be any R object for which `length` and subsetting by integers make sense: S3 or S4 methods for these operations will be dispatched as appropriate.

For `sample` the default for `size` is the number of items inferred from the first argument, so that `sample(x)` generates a random permutation of the elements of `x` (or `1:x`).

It is allowed to ask for `size = 0` samples with `n = 0` or a length-zero `x`, but otherwise `n > 0` or positive `length(x)` is required.

Non-integer positive numerical values of `n` or `x` will be truncated to the next smallest integer, which has to be no larger than `.Machine$integer.max`.

The optional `prob` argument can be used to give a vector of weights for obtaining the elements of the vector being sampled. They need not sum to one, but they should be non-negative and not all zero. If `replace` is true, Walker's alias method (Ripley, 1987) is used when there are more than 200 reasonably probable values: this gives results incompatible with those from R < 2.2.0.

If `replace` is false, these probabilities are applied sequentially, that is the probability of choosing the next item is proportional to the weights amongst the remaining items. The number of nonzero weights must be at least `size` in this case.

`sample.int` is a bare interface in which both `n` and `size` must be supplied as integers.

Argument `n` can be larger than the largest integer of type `integer`, up to the largest representable integer in type `double`. Only uniform sampling is supported. Two random numbers are used to ensure uniform sampling of large integers.

Value

For `sample` a vector of length `size` with elements drawn from either `x` or from the integers `1:x`.

For `sample.int`, an integer vector of length `size` with elements from `1:n`, or a double vector if $n \geq 2^{31}$.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

Ripley, B. D. (1987) *Stochastic Simulation*. Wiley.

See Also

`RNGkind(sample.kind = ..)` about random number generation, notably the change of `sample()` results with R version 3.6.0.

CRAN package **sampling** for other methods of weighted sampling without replacement.

Examples

```
x <- 1:12
# a random permutation
sample(x)
# bootstrap resampling -- only if length(x) > 1 !
sample(x, replace = TRUE)

# 100 Bernoulli trials
sample(c(0,1), 100, replace = TRUE)

## More careful bootstrapping --  Consider this when using sample()
## programmatically (i.e., in your function or simulation)!

# sample()'s surprise -- example
x <- 1:10
    sample(x[x >  8]) # length 2
    sample(x[x >  9]) # oops -- length 10!
    sample(x[x > 10]) # length 0

## safer version:
resample <- function(x, ...) x[sample.int(length(x), ...)]
resample(x[x >  8]) # length 2
resample(x[x >  9]) # length 1
resample(x[x > 10]) # length 0

## R 3.0.0 and later
```

```
sample.int(1e10, 12, replace = TRUE)
sample.int(1e10, 12) # not that there is much chance of duplicates
```

`summary.IssueTB` *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, ...)

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

Arguments

`object`	a `IssueTB` or `IssuesTB` object.
`...`	Unused argument

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, ])
```

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

```
## Not run:
update_database(dataset_dir = tempdir())

## End(Not run)
```

with_comments *Check for comments in GitHub Issues*

Description

Function to filter issues with (or without) comments.

Usage

```
with_comments(x, negate = FALSE)

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

Arguments

x	An object of class IssuesTB.
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_issues_to_dataset` *Save datasets in a yaml file*

Description

Save datasets in a yaml file

Usage

```
write_issues_to_dataset(issues, ...)

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

## Default S3 method:
write_issues_to_dataset(issues, ...)

write_labels_to_dataset(
  labels,
  dataset_dir = getOption("IssueTrackeR.dataset.dir"),
  dataset_name = "list_labels.yaml",
  verbose = TRUE
)

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

Arguments

issues	a IssuesTB object.
...	Unused parameter.
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".
verbose	A logical value indicating whether to print additional information. Default is TRUE.
labels	a list representing all labels with simpler structure (with name, description, colour)
milestones	a list representing milestones with simpler structure (with title, description and due_on).

Details

Depending on the object, the defaults value of the argument dataset_name 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_issues_to_dataset(issues, dataset_dir = tempdir())
write_labels_to_dataset(labels, dataset_dir = tempdir())
```

```
write_milestones_to_dataset(milestones, dataset_dir = tempdir())
```

Index