Package 'rj'

November 8, 2024

Type Package

Title Tools for managing the R journal

Version 0.2.4

Author Editor-in-Chief <R-journal@r-project.org>

Maintainer Editor-in-Chief <R-journal@r-project.org>

Description This package provides useful functions for the editors of the R journal.

License MIT + file LICENSE

Depends R (>= 4.2.0)

Imports cli, dplyr, fs, glue, gmailr, googledrive, googlesheets4 (>= 0.2.0), pdftools, purrr, rlang, stringr, tibble, tth, whisker, yaml, ctv, tidyr, scales, tinytex, xfun

Suggests DiagrammeR, knitr, testthat, rmarkdown, renv, rstudioapi, callr, ggplot2

Encoding UTF-8

RoxygenNote 7.2.3

VignetteBuilder knitr

NeedsCompilation no

Contents

`acknowledge_revision` *Send revision received acknowledgement*

Description

Send revision received acknowledgement

Usage

```
acknowledge_revision(article)
```

Arguments

article	this is the article id

`acknowledge_submission` *Send submission acknowledgements*

Description

Send submission acknowledgements

Usage

```
acknowledge_submission(article, editor)
```

Arguments

article	this is the article id
editor	optional string, if specified, also sets the `Editor:` field to that value and the handling editor will be CCd on the e-mail.

`acknowledge_submission_text` *Create an acknowledgement email in correspondence folder*

Description

Create an acknowledgement email in correspondence folder

Usage

```
acknowledge_submission_text(article)
```

Arguments

article	article id

actionable_articles

Show articles that require attention with the corresponding action

Description

Show articles that require attention with the corresponding action

Usage

```
actionable_articles(editor, invite = 7, review = 30, verbose = FALSE)
```

Arguments

editor	string, initial of an editor or an associate editor, defaults to RJ_EDITOR env var
invite	integer, number of days after which an invite is considered overdue
review	integer, number of days after which a review is considered overdue. Note that you can extend this by specifying a due date in the comment, e.g.: "2021-03-01 Agreed (until 2021-05-01)" would allow for two months.
verbose	logical, if TRUE it will always list the full reviewer report for each entry

active_articles *List articles.*

Description

List all articles in common directories.

Usage

```
active_articles()

accepted_articles()
```

Details

- active_articles: 'Submissions/', 'Accepted/'
- accepted_articles: 'Accepted/'

address *An S3 class to represent email addresses.*

Description

An S3 class to represent email addresses.

Usage

```
address(email = NULL, name = NULL, comment = NULL)
```

Arguments

email	Email address of the reviewer
name	Display name, optional
comment	comment, optional

Examples

```
address("h.wickham@gmail.com")
address("h.wickham@gmail.com", "Hadley Wickham")
```

add_ae *Add AE to the DESCRIPTION*

Description

Fuzzy match to find the initial of the AE to fill in the article DESCRIPTION. Checks that AE term has not ended. The status field is also updated with a new line of add AE.

Usage

```
add_ae(article, name, date = Sys.Date())
```

Arguments

article	article id
name	a name used to match AE, can be AE initials, name, github handle, or email
date	the date for updating status

`add_review` *Add the review file received from the reviewer*

Description

This function adds the review file received from the reviewer to the correspondence folder of the article.

Usage

```
add_review(
  article,
  reviewer_id,
  review,
  recommend = NULL,
  date = Sys.Date(),
  AE = is_AE()
)
```

Arguments

`article`	Article id, like `"2014-01"`
`reviewer_id`	Numeric, the index of the intended reviewer in the Reviewer field. 1 for the first reviewer, 2 for the second
`review`	Path to the review file, e.g. pdf, txt, or docx format. If not specified it is assumed that you added the new file into the correspondence directory and the last file for that reviewer will be used. If you specify `<i>-review-<j>.` filename (no path) and it already exists in the correspondence directory, it will be used.
`recommend`	Reviewer's recommendation, one of: Accept, Minor, Major, Reject. If not specified, an attempt is made to auto-detect it from the file by looking at the first occurrence of those keywords. If auto-detect fails, use "Received".
`date`	Date of the comment, defaults to today's date
`AE`	Logical, if `TRUE` then `"AE: "` prefix is added to the recommendation.

Examples

```
## Not run:
# add review file from the first reviewer and recommend accepting it
add_review("2020-114", reviewer_id = 1, review = file.choose(), recommend = "Accept")

## End(Not run)
```

add_reviewer *Invite an reviewer*

Description

This function adds the reviewer information(name and email) to the reviewers field in the DESCRIPTION as well as draft an email to invite teh reviewer.

Usage

```
add_reviewer(article, name, email, invite = TRUE)
```

Arguments

article	Article id, like "2014-01"
name	Full name of the reviewer
email	Email address of the reviewer
invite	Logical, whether to automatically construct an email to invite the reviewer

Examples

```
## Not run:
add_reviewer("2019-117", "Quiet Quokka", "qqplot@waspot.com")

## End(Not run)
```

AEs *Associate editor (AE) functions*

Description

Functions to determine if the user is an AE and retrieve relevant AE information

Usage

```
AEs()

detect_AE(path = ".", require = FALSE)

AE(path = ".")

is_AE(path = ".")
```

Arguments

path	string, path to the git repository to use for detection.
require	logical, if TRUE then failing to detect the AE is considered an error

Details

- `AEs()`: read the associate-editors.csv in the rj package as a data frame
- `detect_AE()`: determine AE from the remote of the repository in `path` or from the git config e-mail.this only work for AE and will fail for editors
- `AE()`: returns the corresponding row from `AEs()` if called by an associate editor or in an AE repository, otherwise `NULL`. It relies on either `RJ_EDITOR` environment variable to contain the name of the editor or detection from the git repository pointed to by `path` (see `detect_AE`).
- `is_AE()`: determine if the user is an AE or the repository is an AE repository

Value

- `AEs()`: a data frame with all associate editors
- `detect_AE()`: `NULL` if not found or the row from `AEs()` corresponding to the AE
- `AE`: `NULL` if not an AE or a row from `AEs()`
- `is_AE`: `TRUE` if the user if an AE or the repository is an AE repository, `FALSE` otherwise

---

`ae_workload` *Check the number of articles an AE is currently working on*

Description

This will examine the DESCRIPTION files for articles in the Submissions folder, check articles that have status "with AE".

Usage

```
ae_workload(articles = NULL, day_back = 365, active_only = FALSE)

get_AE(x)
```

Arguments

`articles`	a tibble summary of articles in the accepted and submissions folder. Output of `tabulate_articles()`
`day_back`	numeric; positive number of day to go back for calculating AE workload. Retains any article where any status entry for an article is newer than ‘day_back‘ days ago.
`active_only`	Toggle to show only active AEs (filtered by end year and comment field).
`x`	a single article, i.e. as.article("Submissions/2020-114")

Examples

```
## Not run:
ae_workload()

## End(Not run)
## Not run:
art <- as.article("Submissions/2020-114")
get_AE(art)

## End(Not run)
```

---

article *S3 class for article objects*

Description

Create or convert input into an s3 article object

Usage

```
article(..., quiet = FALSE)

as.article(id)
```

Arguments

`...`	Named arguments giving the components of an article: id, authors, title, editor, reviewers, status
`quiet`	if `TRUE` suppresses failure messages.
`id`	a path to a DESCRIPTION, a path to a directory containing DESCRIPTION, or a article name, found in a sub-directory rejected, accepted or submissions

Details

if the article is not parsable, `article()` will print an error message and return a unparsed blob. This ensures that information is not lost even if some articles have parsing errors.

Usually the best way to use `as_article()` is to have your working directory set to the admin repo, and refer to articles by their id. See the examples section.

Examples

```
## Not run:
as.article("2012-01")
as.article("2012-02")

## End(Not run)
```

article_status_plot *Generates a status plot for articles submitted in the last few years.*

Description

Generates a status plot for articles submitted in the last few years.

Usage

```
article_status_plot(years = NULL, save = TRUE)
```

Arguments

years	years considered. A vector of years, or defaults to last four years.
save	Defaults to TRUE. The plot is saved in the rjournal.github.io/resources folder.

Value

a ggplot, one bar per year (taken from article id)

Examples

```
## Not run:
article_status_plot()

## End(Not run)
```

author_emails *Extract email from leading author of an issue for emailing*

Description

If the email of leading author is not available, the first author with available email will be used.

Usage

```
author_emails(issue)
```

Arguments

issue	the issue number in the ‘Proofs‘ folder, i.e. "2021-2"

Examples

```
## Not run:
author_emails("2021-2")

## End(Not run)
```

build_latex *Build article from LaTeX*

Description

Build article from LaTeX

Usage

```
build_latex(
  article,
  share_path = normalizePath("../share", mustWork = TRUE),
  clean = TRUE
)
```

Arguments

article	The article to build
share_path	???
clean	Remove generated files

corr_author *Extract corresponding author from an article*

Description

Extract corresponding author from an article

Usage

```
corr_author(article)
```

Arguments

article	Article id, like "2014-01"

Examples

```
## Not run:
# extract from a single article
corr_author("Submissions/2020-114")

# extract corresponding authors from the active articles
all <- active_articles()
purrr::map_dfr(all, corr_author)

## End(Not run)
```

decline_reviewer *Update reviewer's response to invite*

Description

This function updates the reviewers field in the DESCRIPTION with reviewer's response: accept, decline, or abandon if no reply from the reviewer for a period of time.

Usage

```
decline_reviewer(article, reviewer_id)

agree_reviewer(article, reviewer_id)

abandon_reviewer(article, reviewer_id)
```

Arguments

article	Article id, like "2014-01"
reviewer_id	Numeric, the index of the intended reviewer in the Reviewer field. 1 for the first reviewer, 2 for the second

Examples

```
## Not run:
# first reviewer declined
decline_reviewer("2020-114", reviewer_id = 1)

# second reviewer agreed
agree_reviewer("2020-114", reviewer_id = 2)

# third reviewer doesn't reply and deemed abandon
abandon_reviewer("2020-114", reviwer_id = 3)

## End(Not run)
```

draft_acknowledge_submissions *Send submission acknowledgement drafts*

Description

Send submission acknowledgement drafts

Usage

```
draft_acknowledge_submissions(drafts)
```

Arguments

drafts	list of gmail_draft objects

email_template *Send an email template.*

Description

Interpolate article values into a template, and create a new (unsent) email in your default mail client. The email is created using `browseURL` and the mailto protocol, so it must be relatively brief.

Usage

```
email_template(article, template)
```

Arguments

`article`	An article id, e.g. `"2013-01"`
`template`	The name of a template (without extension) found in `inst/templates`.

Text format

The template should be divided into header and body with `---`. The header should contain fields and values separated by `:` - only a limited

Template parameters

The templates use whisker to insert template values. These have the form `{{field_name}}`. You can use any field from the description as well as the following special fields:

- name: the name of the first author
- email: email address of first author
- editor: name of editor
- me: your name, as determine by envvar `RJ_NAME`

email_text *Generate an email template.*

Description

Generate an email template.

Usage

```
email_text(text, means = getOption("RJ_EMAIL_TOOL", "mailto"))
```

Arguments

`text`	character vector, text of the e-mail (including headers)
`means`	string, one of "mailto", "browser" (boht uopen mailto: URL), "show" (file.show), "edit" (file.edit), "open" (shell open) or "clip" (system clipboard). Defaults to RJ_EMAIL_TOOL environment variable.

filter_status *Find articles with a given status.*

Description

Find articles with a given status.

Usage

```
filter_status(articles, status)
```

Arguments

articles	A vector of articles, as given by accepted_articles()
status	The status you are looking for

future_ids *Generate a new id value.*

Description

Inspects submissions/, accepted/ and rejected to figure out which id is next in sequence.

Usage

```
future_ids(ids, n = 1)
```

Arguments

ids	XXX
n	No. of ids to generate

get_accepted_articles *Functions for proofing articles*

Description

Functions for proofing articles

Usage

```
get_accepted_articles()

draft_proofing(article, update = TRUE)

proofing_article(drafts)

proofing_article_text(article)
```

Arguments

`article`	article id
`update`	logical, if `TRUE` then the status is updated to "out for proofing"
`drafts`	list of `gmail_draft` objects

Details

- `get_accepted_articles()`: get list of articles in the Accepted folder to be proofed. This can be used with `draft_proofing` to construct emails to authors on the final version.
- `draft_proofing()`: generate proofing email for one article
- `proofing_article()`: send proofing article emails
- `proofing_article_text()`: writes the email text into the correspondence folder

`get_accepted_but_not_online` *Get articles to go online*

Description

Find the articles that are accepted but have not yet been published to online

Usage

```
get_accepted_but_not_online()
```

`get_articles_path` *Get the directory of the articles repository, either as set by set_articles_path() or using git to determine repository root*

Description

This path is used to point to articles on your file system.

Usage

```
get_articles_path()
```

get_article_keywords *Extract keywords from a submitted article*

Description

Extract keywords from a submitted article

Usage

```
get_article_keywords(id)
```

Arguments

id	the article id

Value

a list with keywords and authors of the article

get_md_from_pdf *Get metadata from a PDF file*

Description

Get metadata from a PDF file

Usage

```
get_md_from_pdf(from, final = FALSE)
```

Arguments

from	pdf file
final	If in a final state, then also include page information.

get_reviewer_keywords *Extract keywords from reviewer list*

Description

Extract keywords from reviewer list

Usage

```
get_reviewer_keywords()
```

get_submissions *Download submissions.*

Description

Obtains submissions from the Google Sheets spreadsheet and downloads submission files from Google Drive.

Usage

```
get_submissions(dry_run = FALSE)
```

Arguments

dry_run	Use TRUE for testing, which will not change the sheet. Default is FALSE.

Process

The function does three things automatically:

1. Downloads and extracts submissions into appropriate directories.
2. Marks submissions as "read" in the spreadsheet.
3. Uploads acknowledgement emails to gmail account as drafts.

The user (editor-in-chief) then:

1. Ensures that the files have unzipped correctly (some authors incorrectly upload .rar or .tar.gz files) and that the latex compiles
2. Manually sends the draft emails

git_user *Detect user name and e-mail from GIT*

Description

Detect user name and e-mail from GIT

Usage

```
git_user()
```

Value

named string, e-mail of the user

invite_reviewers *Invite reviewer(s).*

Description

Once you have added reviewers to the DESCRIPTION file, you can use this function to draft invite emails to them all. As well as drafting the emails, it also caches them locally in the `correspodence/` directory of the corresponding article.

Usage

```
invite_reviewers(article, prefix = "1")

invite_reviewer(article, reviewer_id, prefix = "1")
```

Arguments

`article`	Article id, like `"2014-01"`
`prefix`	Prefix added to start file name - used to distinguish between multiple rounds of reviewers (if needed)
`reviewer_id`	Numeric, the index of the intended reviewer in the Reviewer field. 1 for the first reviewer, 2 for the second

list_reviewers *Summarise reviewers' progression of an article*

Description

This function summarises the status of reviewers who are willing to review for a particular article.

Usage

```
list_reviewers(article)

reviewer_status(article)
```

Arguments

`article`	Article id, like `"2014-01"`

Examples

```
## Not run: list_reviewers("Submissions/2020-114")
## Not run:
reviewer_status("Submissions/2020-114")

## End(Not run)
```

make_proof *Make a proof of an issue*

Description

The 'make_proof()' function is the first step to creating an issue. It moves the 'proofed' articles from the 'Accepted' folder and news articles from 'News_items/id' into 'Proofs/id'.

Usage

```
make_proof(id, exec = FALSE)
```

Arguments

id	The id of the issue to proof
exec	Set to TRUE to make the proof, the default (FALSE) allows a preview of which articles will be moved where.

Details

After the proof is made with this function, 'publish_issue()' can be used to publish these articles into the 'rjournal.github.io' repository.

match_keywords *Find reviewers through keywords matching*

Description

Find reviewers for an article through matching the article keywords to the keywords reviewers provided when registering. Notice that a googlesheet authenticate, with your email address printed, will first pop up to verify the access to the reviewer googlesheet.

Usage

```
match_keywords(id, n = 5)
```

Arguments

id	the article id in the description file
n	numeric; number of reviewer to display

Details

All the reviewers are ranked based on the number of matching keywords and when there is a tie, a random draw is used.

For example, an article A has 3 keywords. Two reviewers have all the 3 keywords matched, 5 reviewers have 2 matches, and another 10 have 1 match. To get 5 reviewers for article A, both reviewers with 3 matches are in and a random draw, among the five reviewers with 2 matches, is used to fill the remaining 3 places.

Examples

```
## Not run:
m1 <- match_keywords("2021-13")
m2 <- match_keywords("2021-13", n = 10)

## End(Not run)
```

`new_id` *Generate a new id value.*

Description

Inspects submissions/, accepted/ and rejected to figure out which id is next in sequence.

Usage

```
new_id()
```

`online_metadata` *Generate metadata needed for website.*

Description

Generate metadata needed for website.

Usage

```
online_metadata()
```

`online_metadata_for_article` *Generate metadata for one article.*

Description

Generate metadata for one article.

Usage

```
online_metadata_for_article(x, final = FALSE)
```

Arguments

`x`	An article
`final`	If in a final state, then also include page information.

parse_address_list *Parse a string into a rfc822 address list.*

Description

EBNF at http://tools.ietf.org/html/rfc2822#section-3.4

Usage

```
parse_address_list(x)
```

Arguments

x	string to parse

Value

a list of addresses

Examples

```
parse_address_list("<a@b.com> Alison, <c@d.com> Colin")
```

publish_article *Publish an individual article*

Description

This function will publish an individual article to the 'rjournal.github.io' website repo.

Usage

```
publish_article(
  article,
  volume,
  issue,
  home = get_articles_path(),
  web_path = file.path(home, "..", "rjournal.github.io"),
  legacy = FALSE,
  slug
)
```

Arguments

article	article id
volume	The volume of the article's issue (typically, year - 2008)
issue	The issue number of the article's issue
home	Location of the articles directory

web_path	Location of the web source root of the journal, i.e., where _article lives. The default assumes all repos are checked out by their name in the same top-level directory.
legacy	(Very) old way of referencing the R journal
slug	optional, explicitly set the slug name (for expert use only, useful to skip problematic articles by advancing the slug names manually)

Details

The function will complete the following tasks: 1. Assign an appropriate slug if one is not set in the article DESCRIPTION 2. Produce a zip containing supplementary files described in the DESCRIPTION 3. If legacy PDF article, the articles will be converted into HTML format suitable for the distill HTML website. If an Rmd file with the output format '"rjtools::rjournal_web_article"' is found, it will be directly copied across as-is. 4. Set the issue metadata for these articles in the produced/copied Rmd front matter. 5. Update the status of the article's DESCRIPTION to 'online' 6. Render the document to update the article's HTML and PDF output.

See Also

'publish_issue()', 'publish_news()'

---

publish_issue *Publish an issue*

Description

This function will publish an issue in the 'rjournal.github.io' repository from the Proofs folder. If any articles or news from this issue are not yet published, it will prompt you to also publish these articles and news.

Usage

```
publish_issue(issue, home = get_articles_path(), republish_all = FALSE)
```

Arguments

issue	The name of the issue, for example "2022-1".
home	Location of the articles directory
republish_all	If TRUE, then all articles and news will be published again in rjournal.github.io.

Details

This is the main function to be used for publishing an issue and its contents to the R Journal website. It completes these steps:

1. Publish any unpublished articles from this issue. 2. Publish any unpublished news from this issue. 3. Generate a templated R Markdown file for the issue, with some metadata completed in the document's front matter. 4. Open the generated issue R Markdown file for you to update the metadata, for example to update the editors of the issue.

publish_news *Publish a news article*

Description

This function will publish a news article to the ‘rjournal.github.io‘ repository.

Usage

```
publish_news(news, volume, issue, home = get_articles_path())
```

Arguments

news	File path to the directory containing the news article to publish
volume	The volume of the article’s issue (typically, year - 2008)
issue	The issue number of the article’s issue
home	Location of the articles directory

See Also

‘publish_article‘, ‘publish_issue()‘

reject *Accept, reject, or withdraw an article*

Description

This set of functions wraps around `update_status()` and `email_template` to first update the status field in the DESCRIPTION file and then draft an email from the template. Articles are verified to be under the Submission folder before carrying out the actions to avoid mistake input of article ID.

Usage

```
reject(article, comments = "", date = Sys.Date())

reject_format(article, comments = "", date = Sys.Date())

accept(article, comments = "", date = Sys.Date())

withdraw(article, comments = "", date = Sys.Date())

major_revision(article, comments = "", date = Sys.Date())

minor_revision(article, comments = "", date = Sys.Date())

check_in_submission_folder(article)
```

Arguments

article	Article id, like "2014-01"
comments	Any additional comments
date	Date of status update. If omitted defaults to today.

---

report *Generate a status report.*

Description

This should be run weekly.

Usage

```
report(articles = active_articles())
```

Arguments

articles	list of articles to generate report for. Defaults to all active reports in 'Submissions/'.

---

reviewer_summary *Summarise the reviewer's agree/decline ratio based on past invites*

Description

The function pulls out the agree/decline incidence of all the reviewers based on past invite and calculate the agree percentage of each reviewer. Use tabulate_articles first to tabulate all the articles in a particular directory and then apply this function.

Usage

```
reviewer_summary(articles, push = FALSE)
```

Arguments

articles	a tibble summary of articles in the accepted and submissions folder. Output of tabulate_articles()
push	whether the reviewer number of review completed by the reviewer should be pushed to the reviewer sheet

Examples

```
## Not run:
articles <- tabulate_articles()
reviewer_summary(articles)

## End(Not run)
```

rj *R Journal Package*

Description

A package to make it easier to work with the R-journal dcf files.

set_articles_path *Set the directory of the articles repository*

Description

This path is used to locate articles on your file system. If this path is not specified, the path will default to the root of the repository which contains the working directory

Usage

```
set_articles_path(path)
```

Arguments

path	Articles path

summarise_articles *Summarise editors current in-tray*

Description

This function summarises and prints the articles an editor currently have in hand. It also prints out the articles that has not been assigned to any editor on the top, if any. If assigned to an object, the unassigned articles will appear on the top of the data frame.

Usage

```
summarise_articles(editor = NULL, rejected = FALSE, other = FALSE)

get_assignments(editor, folder = "Submissions")

get_unassigned()

find_articles(editor, folder, role)

get_latest(assignments)
```

Arguments

editor	Editors initials. If NULL, looks for the environment variable RJ_EDITOR.
rejected	Default FALSE. If TRUE, show rejected articles.
other	Default FALSE. If TRUE, list all articles not covered by any of the other options (typically accepted and online)
folder	the folder to search for assignments, one of Submissions, Rejected, Accepted, or Proofs
role	string, take value of either "Editor" or "AE"
assignments	an output object from get_assignments() or get_unassigned()

---

tabulate_articles *Tabulate article descriptions*

Description

Produce a table describing articles in specific directories.

Usage

```
tabulate_articles(dirs = c("Accepted", "Submissions"))
```

Arguments

dirs	The directories containing articles to be searched and tabulated. If TRUE, then all directories will be searched.

---

time_to_accept_plot *Generates a plot of acceptance times for articles published in the last few years.*

Description

Generates a plot of acceptance times for articles published in the last few years.

Usage

```
time_to_accept_plot(years = NULL, save = TRUE)
```

Arguments

years	years considered. A vector of years, or defaults to last four years.
save	Defaults to TRUE. The plot is saved in the rjournal.github.io/resources folder.

Value

a ggplot, one boxplot per publication year

Examples

```
## Not run:
 time_to_accept_plot()

## End(Not run)
```

`update_status` *Update the article status*

Description

This is a general function for updating the status field in the DESCRIPTION.

Usage

```
update_status(
  article,
  status,
  comments = "",
  date = Sys.Date(),
  AE = is_AE(),
  replace = TRUE
)
```

Arguments

`article`	Article id, like `"2014-01"`
`status`	new status to add, see details section for more
`comments`	Any additional comments
`date`	Date of status update. If omitted defaults to today.
`AE`	Logical, if `TRUE`, `"AE: "` is prefixed to the status
`replace`	logical, if the last status already matches `status` then the status is only updated if this flag is set to `TRUE`.

Details

For AEs, status is prefixed with "AE: " and valid status includes "AE: major revision", "AE: minor revision", "AE: accept", and "AE: reject".

For Editors, use `accept()`, `reject()`, and `withdraw()` to update the status as well as draft an email to the correspondence author.

Check valid status with `valid_status`.

Examples

```
## Not run:
update_status("2020-114", status = "AE: major revision")

## End(Not run)
```

valid_reviewer_status *A list of all valid reviewer statuses.*

Description

A list of all valid reviewer statuses.

Usage

```
valid_reviewer_status
```

Format

An object of class `character` of length 9.

Examples

```
valid_reviewer_status
```

valid_status *A list of all valid statuses.*

Description

A list of all valid statuses.

Usage

```
valid_status
```

Format

An object of class `character` of length 25.

Examples

```
valid_status
```

Index