Package: metacheck (via r-universe)

June 25, 2026

Title Check Research Outputs for Best Practices

Version 0.1.0

Date 2026-06-16

Description A modular, extendable system for checking research outputs for best practices using text search, R code, and/or (optional) LLM queries.

License AGPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

Suggests ggplot2, devtools, httptest2, rmarkdown, testthat (>= 3.0.0), rstudioapi, withr, shiny, shinyjs, shinydashboard

Config/testthat/edition 3

Config/testthat/parallel false

Imports dplyr, tidyr, readr, knitr, quarto, DT, bibtex, ellmer, jsonlite, xml2, curl, httr2, rvest, rappdirs, tidytext, statcheck, base64enc, gitcreds, roxygen2, jpeg, progress

VignetteBuilder knitr

URL https://github.com/scienceverse/metacheck, https://scienceverse.github.io/metacheck/

BugReports https://github.com/scienceverse/metacheck/issues

Depends R (>= 4.3.0)

LazyData true

LazyDataCompression xz

Config/Needs/website rmarkdown

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake git make libicu-dev libjpeg-dev libuv1-dev libxml2-dev libssl-dev libx11-dev

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

Date/Publication 2026-06-25 04:55:45 UTC

RemoteUrl https://github.com/scienceverse/metacheck

RemoteRef HEAD

RemoteSha 32659a9b9e1659b9d9cda3440bb3417492694b3a

Contents

`accuracy` *Accuracy*

Description

Signal detection values for modules that classify papers as having a feature or not

Usage

```
accuracy(expected, observed)
```

Arguments

`expected`	a vector of logical values for the expected values
`observed`	a vector of logical values for the observed values

Value

a list of accuracy parameters

add_bib_match *Match table from bib table*

Description

Match table from bib table

Usage

```
add_bib_match(paper, min_score = 50)
```

Arguments

paper	a paper or paperlist object
min_score	minimal score that is taken to be a reliable match

Value

the paper or paperlist with bib_match table added

Examples

```
## Not run:
paper <- demopaper()
paper$bib_match <- NULL # remove existing
paper2 <- add_bib_match(paper)
paper2$bib_match

## End(Not run)
```

aspredicted_info *Retrieve info from AsPredicted by URL*

Description

Retrieve info from AsPredicted by URL

Usage

```
aspredicted_info(ap_url, id_col = 1, wait = 1)
```

Arguments

ap_url	an AsPredicted URL, or a table containing them (e.g., as created by `aspredicted_links()`)
id_col	the index or name of the column that contains AsPredicted URLs, if id is a table
wait	wait time in seconds

Value

a data frame of information

---

aspredicted_links *Find AsPredicted Links in Papers*

Description

Find AsPredicted Links in Papers

Usage

```
aspredicted_links(paper)
```

Arguments

paper	a paper object or paperlist object

Value

a table with the AsPredicted url in the first (href) column

Examples

```
aspredicted_links(psychsci)
```

---

causal_relations *Extract causal relations from sentence(s) via a Hugging Face Space*

Description

Sends one or more input sentences to a public Gradio app hosted on Hugging Face (the *lakens-causal-sentences* Space), created based on code by Rasoul Norouzi, retrieves the result via Server-Sent Events (SSE), and returns a tidy data frame with one row per detected cause–effect relation.

Usage

```
causal_relations(
  sentence,
  rel_mode = "auto",
  rel_threshold = 0.5,
  cause_decision = "cls+span",
  timeout = 10,
  verbose = FALSE
)
```

Arguments

sentence	A character vector of one or more sentences to analyze for causal relations.
rel_mode	Relation extraction mode. Options are "auto" (default) or "neural_only".
rel_threshold	Numeric threshold (default 0.5) for deciding whether a relation is considered causal.
cause_decision	Strategy for cause/effect detection. Options: "cls_only", "span_only", or "cls+span" (default).
timeout	Maximum time (in seconds) to wait for the Hugging Face Space to return a result via SSE before aborting. Default is 10.
verbose	Logical; if TRUE, prints diagnostic information (URLs, status codes, event progression). Default FALSE.

Details

The function uses Gradio's two-step queue API: (1) a POST request enqueues the job and returns an event_id; (2) a GET request streams text/event-stream frames until event: complete. Many Gradio apps emit a **double-encoded** completion payload of the form ["<final JSON string>"]. This function unwraps that to obtain the final JSON structure (an array of items containing causal and relations) before parsing.

If a sentence has **no relations**, the output includes one row with cause = NA, effect = NA, and the sentence's causal flag (as returned by the model). For sentences with **multiple relations**, the function returns one row per relation.

Value

A base data.frame with columns:

- sentence (character): the original input sentence,
- causal (logical): whether the sentence is causal per the model,
- cause (character): extracted cause span (or NA),
- effect (character): extracted effect span (or NA).

References

Norouzi, R., Kleinberg, B., Vermunt, J. K., & van Lissa, C. J. (2025). Capturing causal claims: A fine-tuned text mining model for extracting causal sentences from social science papers. *Research Synthesis Methods*, 16(1), 139–156. https://doi.org/10.1017/rsm.2024.13

Hugging Face Model Card: rasoultilburg/SocioCausaNet https://huggingface.co/rasoultilburg/SocioCausaNet

Examples

```
## Not run:
# Single sentence
df1 <- causal_relations("Smoking causes cancer")
print(df1)
```

```
# Multiple sentences (batch)
df2 <- causal_relations(c("Insomnia causes depression.", "Rain leads to flooding."))
print(df2)

# Custom parameters and verbose diagnostics
df3 <- causal_relations(
  sentence = "Stress increases blood pressure.",
  rel_mode = "auto",
  rel_threshold = 0.4,
  cause_decision = "cls+span",
  timeout = 10,
  verbose = TRUE
)
print(df3)

## End(Not run)
```

---

code_abs_path *Return Absolute Paths*

Description

Check code for the presence of absolute paths

Usage

```
code_abs_path(code_text)
```

Arguments

code_text	the text of the code, excluding comments

Value

a vector of absolute paths

Examples

```
code_text <- c(
  "file <- 'C:/User/lakens/file.R'",
  "tmp <- '/User/lakens/file.html'",
  "convert(file, tmp)"
)
code_abs_path(code_text)
```

code_extract_r *Convert Rmd/qmd files to R code only*

Description

Convert Rmd/qmd files to R code only

Usage

```
code_extract_r(
  file_path = NULL,
  save_path = NULL,
  documentation = 0,
  text = NULL
)
```

Arguments

file_path	a vector of file paths to check
save_path	if NULL, returns a text vector, else a path to save to
documentation	0:2 value to pass to knitr::purl
text	alternative to file_path, pass text directly

Value

a character vector

Examples

```
file_path <- demofile("qmd")
code_text <- code_extract_r(file_path)
```

code_file_refs *Get files referenced in code*

Description

Get files referenced in code

Usage

```
code_file_refs(code_text, lang = c("R", "SPSS", "SAS", "Stata"))
```

Arguments

code_text	the code text for a single file
lang	the language (we only currently handle R, SPSS, SAS, Stata)

Value

a vector of files that are referenced in the code

Examples

```
code_text <- c(
  'source("functions.R")',
  'a <- "bread"',
  'b <- read.csv("file.csv")'
)
code_file_refs(code_text, "R")
```

---

code_lang *Detect Code Language*

Description

Detects code language used in files, only for languages metacheck currently processes (R, SAS, SPSS, Stata).

Usage

```
code_lang(file_name)
```

Arguments

file_name	a vector of file names

Value

a vector of languages

Examples

```
file_name <- "file.R"
code_lang(file_name)

file_name <- c("file.Rmd", "file.SAS", "file.r", "file.qmd", "file.txt")
code_lang(file_name)
```

code_library_lines *Get Code Library Lines*

Description

Returns the lines on which library/require calls exist. This is a helper function for the code_check module.

Usage

```
code_library_lines(code_text, lang = c("R", "SPSS", "SAS", "Stata"))
```

Arguments

code_text	the code text for a single file
lang	the language (we only currently handle R, SPSS, SAS, Stata)

Value

a data frame with columns code and line (the line numbers on which library calls exist, after removing blank lines and comments)

Examples

```
code_text <- c(
  "library(dplyr)",
  "",
  "# this line won't count",
  "library(tidyr)",
  "renv::install('metacheck')"
)
code_library_lines(code_text, "R")
```

code_line_stats *Get Code Composition Stats*

Description

Get Code Composition Stats

Usage

```
code_line_stats(code_text, lang = c("R", "SPSS", "SAS", "Stata"))
```

Arguments

code_text	the code text for a single file
lang	the language (we only currently handle R, SPSS, SAS, Stata)

Value

list with items total_lines, comment_lines, code_lines, and percent_comment

Examples

```
code_text <- c(
  "library(dplyr)",
  "",
  "# this line is a comment",
  "a <- 1"
)
code_line_stats(code_text, "R")
```

---

code_parse_r *Parse code to check for errors*

Description

Parse code to check for errors

Usage

```
code_parse_r(file_path = "", text = NULL)
```

Arguments

file_path	a vector of file paths to check
text	alternative to file_path, pass text directly

Value

a data frame with columns file_path and line

Examples

```
file_path <- demofile("qmd")
code_parse_r(file_path)
```

code_read *Read code from files*

Description

Read code from files

Usage

```
code_read(file_path)
```

Arguments

file_path	a file path or url to read in

Value

a character vector of the file contents

Examples

```
file_path <- demofile("json")
text <- code_read(file_path)
```

code_remove_comments *Remove comments from code text*

Description

Remove comments from code text

Usage

```
code_remove_comments(code_text, lang = c("R", "SPSS", "SAS", "Stata"))
```

Arguments

code_text	the code text for a single file
lang	the language (we only currently handle R, SPSS, SAS, Stata)

Value

the code_text minus comment lines

Examples

```
code_text <- c(
  "# this is a comment",
  "",
  "x <- 'And this is code'"
)
code_text_nc <- code_remove_comments(code_text, "R")
```

`collapse_section` *Make Collapsible Section*

Description

A helper function for making module reports.

Usage

```
collapse_section(
  text,
  title = "Learn More",
  callout = c("tip", "note", "warning", "important", "caution"),
  collapse = TRUE
)
```

Arguments

`text`	The text to put in the collapsible section; vectors will be collapse with line breaks between (e.g., into paragraphs)
`title`	The title of the collapse header
`callout`	the type of quarto callout block
`collapse`	whether to collapse the block at the start

Value

text

Examples

```
text <- c("Paragraph 1...", "Paragraph 2...")
collapse_section(text) |> cat()
```

convert *Convert documents*

Description

Uses grobid or bibr to convert a file to paper format.

Usage

```
convert(
  file_path,
  save_path = ".",
  method = c("auto", "bibr", "grobid", "xml"),
  crossref_lookup = FALSE,
  keep_xml = TRUE,
  ...
)
```

Arguments

file_path	Path to the document file, or a directory of documents
save_path	Path to a directory in which to save the JSON file
method	whether to use bibr, grobid, or xml (grobid_to_bibr) to convert a file (see Details)
crossref_lookup	whether to add the bib_match table from crossref
keep_xml	if the method is grobid, whether to keep intermediate XML files
...	further arguments to pass to convert_bibr, convert_grobid, or grobid_to_bibr

Details

Both bibr and grobid can handle PDF files. Only bibr can convert doc or docx files. Already-converted grobid XML files can be converted to bibr format (set crossref_lookup=TRUE to add a bib_match table). If the file_path is a directory, the method will be xml if any XML files are present, and bibr if only doc or docx files are present.

Value

the path to the JSON file

convert_bibr *Convert documents using bibr*

Description

Converts document files (PDF, DOC, DOCX) to structured JSON using the bibr extraction service. Supports two backends: the Scienceverse platform (`"scivrs"`) which uses a job queue with load balancing, and a self-hosted bibr instance (`"selfhosted"`) for direct API access.

Usage

```
convert_bibr(
  file_path,
  save_path = ".",
  backend = c("auto", "scivrs", "selfhosted"),
  api_key = NULL,
  api_url = NULL,
  include_figures = FALSE,
  start_page = 1,
  end_page = Inf,
  poll_interval = 2,
  timeout = 600
)
```

Arguments

`file_path`	Path to the document file, or a directory of documents
`save_path`	Path to a directory in which to save the JSON file
`backend`	Which backend to use: `"auto"` (default) detects from the available API key, `"scivrs"` uses the Scienceverse platform, `"selfhosted"` uses a direct bibr API instance.
`api_key`	API key (scivrs backend only). A Bearer token starting with `sv_`, defaults to the `SCIVRS_API_KEY` env var. Ignored for the `"selfhosted"` backend, which requires no authentication.
`api_url`	Base URL of the API. Defaults to the appropriate URL for the selected backend.
`include_figures`	Whether to include base64-encoded figure images in the output (default FALSE)
`start_page`	First page of the file to extract (default 1)
`end_page`	Last page of the file to extract (default Inf for all pages)
`poll_interval`	Seconds between status polls, scivrs backend only (default 2)
`timeout`	Maximum seconds to wait for processing, scivrs backend only (default 600)

Details

When backend = "auto" (the default), the "scivrs" backend is used if api_key is provided or the SCIVRS_API_KEY environment variable is set. Otherwise, "selfhosted" is used (no authentication required).

Value

Path(s) to the saved JSON file(s)

Examples

```
## Not run:
# Auto-detect backend from environment variables
pdf <- demofile("pdf")
convert_bibr(pdf)

# Explicitly use Scienceverse platform
convert_bibr(pdf, backend = "scivrs")

# Use self-hosted bibr instance
convert_bibr(pdf, backend = "selfhosted")

# Extract specific pages
convert_bibr(pdf, start_page = 1, end_page = 10)

# Directory of papers
dir <- system.file("demo", package = "metacheck")
convert_bibr(dir, save_path = "results/")

## End(Not run)
```

---

convert_grobid *Convert a PDF to Grobid XML*

Description

This function uses a GDPR-compliant public grobid server maintained by Eindhoven Technical University. You can set up your own local grobid server following instructions from https://grobid.readthedocs.io/ and set the argument api_url to its path (probably http://localhost:8070). See https://github.com/grobidOrg/grobid#demo for other publicly available servers (we cannot guarantee their privay).

Usage

```
convert_grobid(
  file_path,
  save_path = ".",
  api_url = "https://grobid.hti.ieis.tue.nl",
```

```
  start_page = -1,
  end_page = -1,
  consolidate_citations = 0,
  consolidate_header = 0,
  consolidate_funders = 0
)
```

Arguments

file_path	path to the PDF, a vector of paths, or a directory name that contains PDFs
save_path	directory or file path to save to; set to NULL to return the XML directly
api_url	the URL to the grobid server
start_page	the first page of the PDF to read (defaults to -1 to read all pages)
end_page	the last page of the PDF to read (defaults to -1 to read all pages)
consolidate_citations	whether to fix/enhance citations
consolidate_header	whether to fix/enhance paper info
consolidate_funders	whether to fix/enhance funder info

Details

Consolidation of citations, headers, and funders looks up these items in CrossRef or another database to fix or enhance information (see https://grobid.readthedocs.io/en/latest/Consolidation/). This can slow down conversion. Consolidating headers is only useful for published papers, and can be set to 0 for work in prep. We recommend you leave these defaults at 0 and use crossref_loookup = TRUE when converting from grobid XML to bibr JSON format with the convert() function.

Value

XML object

---

crossref_doi *CrossRef Info from DOI*

Description

Valid selects for crossref API are:

Usage

```
crossref_doi(
  doi,
  select = c("DOI", "type", "title", "author", "container-title", "volume", "issue",
    "page", "URL", "abstract", "year", "error")
)
```

Arguments

doi	the DOI of the paper to get info for
select	what fields to select from the crossref API

Details

abstract, URL, resource, member, posted, score, created, degree, update-policy, short-title, license, ISSN, container-title, issued, update-to, issue, prefix, approved, indexed, article-number, clinical-trial-number, accepted, author, group-title, DOI, is-referenced-by-count, updated-by, event, chair, standards-body, original-title, funder, translator, published, archive, published-print, alternative-id, subject, subtitle, published-online, publisher-location, content-domain, reference, title, link, type, publisher, volume, references-count, ISBN, issn-type, assertion, deposited, page, content-created, short-container-title, relation, editor

Value

data frame with DOIs and info

Examples

```
doi <- "10.7717/peerj.4375"
## Not run:
# cr_info <- crossref_doi(doi)

## End(Not run)
```

---

crossref_query *Look up Reference in CrossRef*

Description

Look up Reference in CrossRef

Usage

```
crossref_query(
  ref,
  min_score = 50,
  rows = 1,
  select = c("DOI", "score", "type", "title", "author", "editor", "publisher",
    "container-title", "year", "volume", "issue", "page", "URL")
)
```

Arguments

ref	the full text reference of the paper to get info for, see Details
min_score	minimal score that is taken to be a reliable match (default 50)
rows	the maximum number of rows to return per reference (default 1)
select	what fields to select from the crossref API

Details

The argument `ref` can take many formats. Crossref queries only look for authors, title, and container-title (e.g., journal or book), but extra information doesn't seem to hurt.

- a text reference or fragment
- a bibentry object (authors, title and container will be extracted)
- a vector of text or bibentry objects
- a paper object (the bib table will be extracted)

Valid selects for this route are: abstract, URL, resource, member, posted, score, created, degree, update-policy, short-title, license, ISSN, container-title, issued, update-to, issue, prefix, approved, indexed, article-number, clinical-trial-number, accepted, author, group-title, DOI, is-referenced-by-count, updated-by, event, chair, standards-body, original-title, funder, translator, published, archive, published-print, alternative-id, subject, subtitle, published-online, publisher-location, content-domain, reference, title, link, type, publisher, volume, references-count, ISBN, issn-type, assertion, deposited, page, content-created, short-container-title, relation, editor

Value

doi

Examples

```
ref <- paste(
  "Lakens, D., Mesquida, C., Rasti, S., & Ditroilo, M. (2024).",
  "The benefits of preregistration and Registered Reports.",
  "Evidence-Based Toxicology, 2(1)."
)
## Not run:
cr <- crossref_query(ref)

## End(Not run)
```

`datacite_doi` *Doi.org Info from DataCite*

Description

Doi.org Info from DataCite

Usage

```
datacite_doi(doi)
```

Arguments

`doi`	the DOI(s) to get info for

Value

bib_match data frame

Examples

```
doi <- "10.5281/zenodo.2669586"
## Not run:
doi_info <- datacite_doi(doi)

## End(Not run)
```

`demofile` *Get a demo file*

Description

Return the file path for various versions of the demo paper. Use `demopaper()` to directly read it as a paper object from the json file.

Usage

```
demofile(ext = c("json", "pdf", "docx", "doc", "xml", "qmd"))
```

Arguments

`ext`	the extension of the file

Value

file path

Examples

```
json <- demofile()
pdf <- demofile("pdf")
```

demopaper *Get demo paper*

Description

Get demo paper

Usage

```
demopaper()
```

Value

paper object

Examples

```
paper <- demopaper()
```

doi_clean *Clean DOIs*

Description

Clean DOIs

Usage

```
doi_clean(doi)
```

Arguments

doi	a character vector of one or more DOIs

Value

a character vector of cleaned DOIs (no https://doi.org or DOI:)

Examples

```
doi_clean("https://doi.org/10.1038/nphys1170")
doi_clean("doi:10.1038/nphys1170")
doi_clean("DOI: 10.1038/nphys1170")
```

doi_lookup *Doi.org Info from DOI*

Description

Doi.org Info from DOI

Usage

```
doi_lookup(doi)
```

Arguments

doi	the DOI(s) to get info for

Value

data frame with DOIs and info

Examples

```
doi <- "10.7717/peerj.4375"
## Not run:
doi_info <- doi_lookup(doi)

## End(Not run)
```

doi_resolves *Check whether a DOI resolves*

Description

Checks the doi.org API to see if a DOI is registered and has an associated URL (using `https://doi.org/api/handles`). Returns TRUE if it does, FALSE if the DOI does not exist or does not have an associated URL, and NA if the test failed. Clearly invalid DOIs (i.e. not starting with "10.") will return FALSE without server requests.

Usage

```
doi_resolves(doi, timeout = 10)
```

Arguments

doi	Character vector. One or more DOIs to check.
timeout	Numeric. Request timeout in seconds. Default is `10`.

Value

Logical vector. For each input DOI, returns TRUE if the DOI resolves, FALSE if it does not resolve (or does not start with 10.), and NA if the check failed.

Examples

```
## Not run:
doi_resolves("10.1038/nphys1170") # Expected: TRUE
doi_resolves("10.1234/invalid.doi") # Expected: FALSE

## End(Not run)
```

`doi_valid_format` *Validate DOI format*

Description

Validate DOI format

Usage

```
doi_valid_format(doi)
```

Arguments

`doi`	a character vector of one or more DOIs

Value

a logical vector

Examples

```
doi_valid_format("10.1038/nphys1170")
doi_valid_format("no.no.10.1038")
```

email *Set or get email*

Description

Set or get email

Usage

```
email(email = NULL)
```

Arguments

email	if a string, sets the email

Value

the current option value (character)

Examples

```
email()
```

emojis *Emojis*

Description

Useful emojis

Usage

```
emojis
```

Format

An object of class `list` of length 32.

Details

- General: "check", "star", warning", "stop", "x", "no", "thumbs_up", "thumbs_down", "info", "question"
- Traffic Lights: "tl_green", "tl_yellow", "tl_red", "tl_info", "tl_na", "tl_fail"
- Hearts: "red", "orange", "yellow", "green", "blue", "purple", "brown", "black", "white", "pink"

extract_eq *Extract Equations*

Description

List all equations in the text, returning the matched text (e.g., 't(28) = 2.4', 'p = 0.04') and document location in a table. This is the canonical extractor for reported statistics and effect sizes; modules that need statistics should read from this table rather than re-scanning the text.

Usage

```
extract_eq(paper)
```

Arguments

paper	a paper object or paperlist object

Details

This will catch most comparators like =<>~and most versions of scientific notation like 5.0 x 10^-2 or 5.0e-2. If you find any formats that are not correctly handled by this function, please contact the author.

Value

a data frame with one row per equation and the columns `lhs` (the statistic name, e.g. "t", "F", "p"), `df` (parenthetical degrees of freedom such as "(28)" or "(2, 57)", otherwise NA), `comp` (the comparator, e.g. "="), `rhs` (the reported value as text), `grp_id` (groups equations in the same sentence), `text_id`, and `paper_id`.

Examples

```
paper <- demopaper()
equations <- extract_eq(paper)
```

extract_p_values *Extract P-Values*

Description

List all p-values in the text, returning the matched text (e.g., 'p = 0.04') and document location in a table.

Usage

```
extract_p_values(paper)
```

Arguments

paper	a paper object or paperlist object

Details

Note that this will not catch p-values reported like "the p-value is 0.03" because that results in a ton of false positives when papers discuss p-value thresholds. If you need to detect text like that, use the `text_search()` function and a custom pattern.

This will catch most comparators like =<>~ and most versions of scientific notation like 5.0 x 10^-2 or 5.0e-2. If you find any formats that are not correctly handled by this function, please contact the author.

Value

a table

Examples

```
paper <- demopaper()
p_values <- extract_p_values(paper)
```

---

extract_urls *Extract URLs*

Description

Get a table of URLs from a paper or paperlist. Matches urls that start with http or doi:

Usage

```
extract_urls(paper)
```

Arguments

paper	a paper object or paperlist object

Value

a table

Examples

```
paper <- demopaper()
urls <- extract_urls(paper)
```

fig_image_view *View a figure image*

Description

View a figure image

Usage

```
fig_image_view(paper, figure_id = 1)
```

Arguments

paper	a paper object
figure_id	the id for the figure to show

Value

plots the figure

Examples

```
paper <- demopaper()
fig_image_view(paper, 1)
fig_image_view(paper, 2)
```

file_category *Categorise files*

Description

Categorise files

Usage

```
file_category(contents)
```

Arguments

contents	a table with columns name, path such as from osf_contents()

Value

the table with new column file_category

Examples

```
contents <- c("script.R", "data.csv", "README", "codebook.csv")
file_category(contents)
```

filetype *Get file Type from Extension*

Description

Get file Type from Extension

Usage

```
filetype(filename)
```

Arguments

filename	the file name

Value

a named vector of file types

Examples

```
filetype("script.R")
```

FLoRA *FORRT Replication Database (FLoRA)*

Description

FLoRA database containing DOIs of original studies and replications. Use `FLoRA_date()` to find the date it was downloaded, and `FLoRA_update()` to update it.

Usage

```
FLoRA()
```

Format

A data frame with 8 columns:

doi_o DOI of original study

apa_ref_o APA reference of original study

doi_r DOI of replication study (may be NA if url_r is provided)

apa_ref_r APA reference of replication study

url_r URL of replication study (used when DOI is not available)

outcome replication outcome

outcome_quote quote describing replication outcome

type replication or reproduction

Value

a data frame

Source

https://osf.io/9r62x/files/t4j8f

Examples

```
FLoRA()
```

FLoRA_date *Get date FLoRA was updated*

Description

Get date FLoRA was updated

Usage

```
FLoRA_date()
```

Value

the date

Examples

```
FLoRA_date()
```

FLoRA_update *Update FLoRA*

Description

metacheck comes with a built-in data frame called FLoRA. We update it regularly, but you can use this function to download the newest version. The download is >5MB, but this function will summarise the information into a smaller version and delete the original file.

Usage

```
FLoRA_update()
```

Value

the path to the data frame (invisibly)

format_bib_authors *Format Bib Authors*

Description

Formats a structured author list (data frame with given/family columns) as a display string.

Usage

```
format_bib_authors(authors)
```

Arguments

authors	a data frame with given and family columns, or a list of such data frames

Value

a character string (or vector) of formatted author names

Examples

```
authors <- data.frame(given = c("Alice H.", "Wendy"),
                      family = c("Eagly", "Wood"))
format_bib_authors(authors)
```

format_ref *Format Reference*

Description

Format a reference for display in a report.

Usage

```
format_ref(bib)
```

Arguments

bib	a bibentry object or list of bibentry objects

Details

The argument bib should be a bibentry object (e.g., like those made by citation(), but it can also handle a bibtex object or a bibtex formatted character vector. If these do not read in as valid bibtex, the original text of bib will be returned unformatted.

Value

formatted text

Examples

```
mc <- citation("metacheck")
format_ref(mc)

# handles bibtext
bib_mc <- utils::toBibtex(mc)
format_ref(bib_mc)

paper <- demopaper()
format_ref(paper$bib$ref[1:2])
```

---

get_prev_outputs *Get Previous Outputs*

Description

A helper for creating modules. Checks for previous module outputs in a chain and returns the named list item if it exists in any parent environment.

Usage

```
get_prev_outputs(module, item, parent_n = 2)
```

Arguments

module	the name of a previously run module
item	the name of the list item to extract
parent_n	the number of parents to traverse up the chain. Noramlly 2 if you are calling this from a module function, but maybe more if you are calling it from a helper function.

Value

the extracted list item, or NULL if not found

Examples

```
# .__mc__prev_outputs is usually created by `module_run()`
.__mc__prev_outputs <- list(mod_1 = list(a = 1, b = 2))
f <- function(item) {
  get_prev_outputs("mod_1", item)
}
f("a")
f("d")
```

github_files *Get File List from GitHub*

Description

Get File List from GitHub

Usage

```
github_files(repo, dir = "", recursive = FALSE)
```

Arguments

repo	The URL of the repository (in the format "username/repo" or "https://github.com/username/repo")
dir	an optional directory name to search
recursive	whether to search the files recursively

Value

a data frame of files

Examples

```
## Not run:
github_files("scienceverse/metacheck")

## End(Not run)
```

github_info *Get GitHub Repo Info*

Description

Get GitHub Repo Info

Usage

```
github_info(repo, recursive = FALSE)
```

Arguments

repo	The URL of the repository (in the format "username/repo" or "https://github.com/username/repo")
recursive	whether to search the files recursively

Value

a list of information about the repo

Examples

```
## Not run:
github_info("scienceverse/metacheck")

## End(Not run)
```

`github_languages` *Get Languages from GitHub Repo*

Description

Get Languages from GitHub Repo

Usage

```
github_languages(repo)
```

Arguments

`repo`	The URL of the repository (in the format "username/repo" or "https://github.com/username/repo")

Value

vector of languages

Examples

```
## Not run:
github_languages("scienceverse/metacheck")

## End(Not run)
```

github_links *Find GitHub Links in Papers*

Description

GitHub links can be in PDFs in several ways.

Usage

```
github_links(paper)
```

Arguments

paper	a paper object or paperlist object

Value

a table with the GitHub url in the first (text) column

Examples

```
github_links(psychsci)
```

github_readme *Get README from GitHub*

Description

Get README from GitHub

Usage

```
github_readme(repo)
```

Arguments

repo	The URL of the repository (in the format "username/repo" or "https://github.com/username/repo")

Value

a character string of the README contents

Examples

```
## Not run:
github_readme("scienceverse/metacheck")

## End(Not run)
```

github_repo *Get Short GitHub Repo Name*

Description

Get Short GitHub Repo Name

Usage

```
github_repo(repo)
```

Arguments

repo	The URL of the repository (in the format "username/repo" or "https://github.com/username/repo")

Value

character string of short repo name

Examples

```
github_repo("scienceverse/metacheck")
github_repo("https://github.com/scienceverse/metacheck/")
github_repo("https://github.com/scienceverse/metacheck.git")
```

grobid_to_bibr *Convert Grobid TEI XML file to bibr format*

Description

Convert Grobid TEI XML file to bibr format

Usage

```
grobid_to_bibr(xml_path, save_path = ".", crossref_lookup = FALSE)
```

Arguments

xml_path	the path to the XML file
save_path	directory or file path to save to; set to NULL to return a paper object
crossref_lookup	whether to look up references in crossref

Value

a paper object

json_expand *Expand a JSON column*

Description

It is useful to ask an LLM to return data in JSON structured format, but can be frustrating to extract the data, especially where the LLM makes syntax mistakes. This function tries to expand a column with a JSON-formatted response into columns and deals with it gracefully (sets an 'error' column to "parsing error") if there are errors. It also fixes column data types, if possible.

Usage

```
json_expand(table, col = "answer", suffix = c("", ".json"))
```

Arguments

table	the table with a column to expand
col	the name or index of the column to expand (defaults to "answer" or the first column)
suffix	the suffix for the extracted columns if they conflict with names in the table

Value

the table plus the expanded columns

Examples

```
table <- data.frame(
  paper_id = 1:5,
  answer = c(
    '{"number": "1", "letter": "A", "bool": true}',
    '{"number": "2", "letter": "B", "bool": "FALSE"}',
    '{"number": "3", "letter": "", "bool": null}',
    "oh no, the LLM misunderstood",
    '{"number": "5", "letter": ["E", "F"], "bool": false}'
  )
)

expanded <- json_expand(table, "answer")
expanded
```

lastlog *Get the last log*

Description

Get the last log

Usage

```
lastlog(i = 1, logpath = NULL)
```

Arguments

i	the indices to return
logpath	an optional file path to read the log from

Value

a list of the last log item, or a data frame of multiple items

Examples

```
# set up 2 log items
logger("test", list(msg = "hi"))
logger("test", list(msg = "hi again"))

lastlog()
lastlog(2)
lastlog(1:2)
```

link *Make an html link*

Description

Make an html link

Usage

```
link(url, text = url, new_window = TRUE, type = "")
```

Arguments

url	the URL to link to
text	the text to link
new_window	whether to open in a new window
type	handle common links, like "doi" ()

Value

string

Examples

```
link("https://scienceverse.org")
```

`llm` *Query an LLM*

Description

Ask a large language model (LLM) any question you want about a vector of text or the text from a text_search(). When `type` is provided, uses ellmer's structured output API to guarantee output conforming to the type spec; otherwise returns free-text responses in an `answer` column.

Usage

```
llm(
  text,
  system_prompt,
  type = NULL,
  text_col = "text",
  model = llm_model(),
  params = list()
)
```

Arguments

`text`	The text to send to the LLM (vector of strings, or data frame with the text in a column)
`system_prompt`	A system prompt to set the behavior of the assistant
`type`	An optional ellmer type specification for structured extraction (e.g., from `type_object()`, `type_from_schema()`). When provided, the provider enforces the schema and returns structured columns instead of free text.
`text_col`	The name of the text column if text is a data frame
`model`	the LLM model name (see `llm_model_list()`) in the format "provider" or "provider/model"
`params`	a named list to pass to `ellmer::params()`

Details

You will need to get your own API key from https://console.groq.com/keys. To avoid having to type it out, add it to the .Renviron file in the following format (you can use `usethis::edit_r_environ()` to access the .Renviron file)

GROQ_API_KEY="key_value_asdf"

See https://console.groq.com/docs for more information

Value

a data frame of results

Examples

```
## Not run:
# Free-text query
text <- c("hello", "number", "ten", 12)
system_prompt <- "Is this a number? Answer only 'TRUE' or 'FALSE'"
is_number <- llm(text, system_prompt)

# Structured extraction
type_spec <- ellmer::type_object(
  is_number = ellmer::type_boolean("Whether the input is a number")
)
result <- llm(c("hello", "42"), "Classify the input.", type = type_spec)

## End(Not run)
```

---

llm_max_calls *Set the maximum number of calls to the LLM*

Description

Set the maximum number of calls to the LLM

Usage

```
llm_max_calls(n = NULL)
```

Arguments

n	The maximum number of calls that the llm() function can make

---

llm_model *Set the default LLM model*

Description

Use `llm_model_list()` to get a list of available models

Usage

```
llm_model(model = NULL)
```

Arguments

model	the name of the model

llm_model_list *List LLM Models*

Description

List available LLM models for the specified platform.

Usage

```
llm_model_list(platform = NULL)
```

Arguments

platform	The platform. If NULL, checks all platforms for which you have a valid API_KEY.

Details

For platforms other than groq, returns the value from the corresponding ellmer::models_platform function.

Value

a data frame of models and info

Examples

```
## Not run:
llm_model_list()

## End(Not run)
```

llm_use *Set or get metacheck LLM use*

Description

Mainly for use in optional LLM workflows in modules

Usage

```
llm_use(llm_use = NULL)
```

Arguments

llm_use	if logical, sets whether to use LLMs

Value

the current option value (logical)

Examples

```
if (llm_use()) {
  print("We can use LLMs")
} else {
  print("We will not use LLMs")
}
```

`local_files` *List Local Files*

Description

Lists all files in a local directory recursively and returns a data frame compatible with the `repo_check` output table, for use with `code_check`.

Usage

```
local_files(path, recursive = FALSE)
```

Arguments

`path`	path to a local directory or file, or a vector of paths
`recursive`	whether to search the files recursively

Value

a data frame with columns `repo_url`, `file_name`, `file_url`, `file_location`, `file_size`, `file_type`

Examples

```
## Not run:
  local_files("my_project")

## End(Not run)
```

logger *Log messages*

Description

Adds a logging message to the log. Keeps the log as a maximum of 1000 rows.

Usage

```
logger(label = "", contents = list(), logpath = NULL)
```

Arguments

label	a string with the context (e.g.,module name)
contents	a named list of the log contents
logpath	an optional file path to save the log in

Value

called for side effects of writing to log, returns logpath

Examples

```
logpath <- tempfile(fileext = ".log")
logger("test", list(x = 1), logpath)
lastlog()
```

module_help *Get Module Help*

Description

See the help files for a module by name (get a list of names from `module_list()`)

Usage

```
module_help(module = NULL)
```

Arguments

module	the name of a module or path to a module

Value

the help text

Examples

```
module_help("marginal")
```

`module_info` *Get module information*

Description

Get module information

Usage

```
module_info(module)
```

Arguments

`module`	the name of a module or path to a module

Value

a list of module info

Examples

```
module_info("all_p_values")
```

`module_list` *List modules*

Description

List modules

Usage

```
module_list(module_dir = system.file("modules", package = "metacheck"))
```

Arguments

`module_dir`	the directory to search for modules (defaults to the built-in modules)

Value

a data frame of modules

Examples

```
mods <- module_list()
```

module_report *Report from module output*

Description

Report from module output

Usage

```
module_report(module_output, header = 3)
```

Arguments

module_output	the output of a module_run()
header	header level (default 2)

Value

text

Examples

```
paper <- demopaper()
op <- module_run(paper, "stat_p_exact")
module_report(op) |> cat()
```

module_run *Run a module*

Description

Run a module

Usage

```
module_run(paper, module, ...)
```

Arguments

paper	a paper object or a list of paper objects
module	the name of a module or path to a module to run on this object
...	further arguments to the module (e.g., arguments for the llm() function like params); these will override any arguments in the module

Value

a list of the returned table and report text

Examples

```
module_run(psychsci[[1]], "all_p_values")
```

---

module_template *Create a Module from a Template*

Description

Create a Module from a Template

Usage

```
module_template(module_name, path = "./modules")
```

Arguments

module_name	The short name of the module (should contain only letters, numbers, and _)
path	The path of the directory to save the module in (defaults to a directory called "modules" in the working directory)

Value

the file path (invisibly)

---

openalex_doi *OpenAlex info from DOI*

Description

See details for a list of root-level fields that can be selected.

Usage

```
openalex_doi(doi, select = NULL)
```

Arguments

doi	the DOI of the paper to get info for
select	a vector of fields to return, NULL returns all

Details

See https://docs.openalex.org/api-entities/works/work-object for explanations of the information you can retrieve about works.

Root-level fields for the select argument:

- id
- doi
- title
- display_name
- publication_year
- publication_date
- ids
- language
- primary_location
- type
- type_crossref
- indexed_in
- open_access
- authorships
- institution_assertions
- countries_distinct_count
- institutions_distinct_count
- corresponding_author_ids
- corresponding_institution_ids
- apc_list
- apc_paid
- fwci
- has_fulltext
- fulltext_origin
- cited_by_count
- citation_normalized_percentile
- cited_by_percentile_year
- biblio
- is_retracted
- is_paratext
- primary_topic
- topics
- keywords

- concepts
- mesh
- locations_count
- locations
- best_oa_location
- sustainable_development_goals
- grants
- datasets
- versions
- referenced_works_count
- referenced_works
- related_works
- abstract_inverted_index
- abstract_inverted_index_v3
- cited_by_api_url
- counts_by_year
- updated_date
- created_date

Value

list with DOIs and info

Examples

```
doi <- "10.7717/peerj.4375"
## Not run:
oa_info <- openalex_doi(doi)
oa_info <- openalex_doi(doi, "title")

## End(Not run)
```

---

openalex_query *Look up a reference in OpenAlex*

Description

Look up a reference in OpenAlex

Usage

```
openalex_query(title, source = NA, authors = NA, strict = TRUE)
```

Arguments

title	The title of the work
source	The source (journal or book)
authors	The authors
strict	Whether to return NULL or the best match if there isn't a single match

Value

A data frame with citation info

Examples

```
## Not run:
openalex_query("Sample Size Justification", "Collabra Psychology")

## End(Not run)
```

---

osf_api_check *Check OSF API Server Status*

Description

Check the status of the OSF API server.

Usage

```
osf_api_check(
  osf_api = getOption("metacheck.osf.api"),
  on_error = c("stop", "warn", "ignore")
)
```

Arguments

osf_api	the OSF API to use (e.g., "https://api.osf.io/v2")
on_error	whether to stop, warn, or ignore errors

Details

The OSF API server is down a lot, so it's often good to check it before you run a bunch of OSF functions. When the server is down, it can take several seconds to return an error, so scripts where you are checking many URLs can take a long time before you realise they aren't working.

You can only make 100 API requests per hour, unless you authorise your requests, when you can make 10K requests per day. The osf functions in metacheck often make several requests per URL to get all of the info. You can authorise them by creating an OSF token at https://osf.io/settings/tokens and including the following line in your .Renviron file:

OSF_PAT="replace-with-your-token-string"

Value

the OSF status

Examples

```
osf_api_check()
```

`osf_check_id` *Check OSF IDs*

Description

Check if strings are valid OSF IDs, URLs, or waterbutler IDs. Basically an improved wrapper for `osfr::as_id()` that returns NA for invalid IDs in a vector.

Usage

```
osf_check_id(osf_id)
```

Arguments

`osf_id`	a vector of OSF IDs or URLs

Value

a vector of valid IDs, with NA in place of invalid IDs

Examples

```
osf_check_id("pngda")
osf_check_id("osf.io/pngda")
osf_check_id("https://osf.io/pngda")
osf_check_id("https://osf .io/png da") # rogue whitespace
osf_check_id("pnda") # invalid
```

osf_delay *Set the OSF delay*

Description

Sometimes the OSF gets fussy if you make too many calls, so you can set a delay of a few seconds before each call. Use `osf_delay()` to get or set the OSF delay.

Usage

```
osf_delay(delay = NULL)
```

Arguments

`delay`	the number of seconds to wait between OSF calls

Examples

```
osf_delay()
```

osf_file_download *Download all OSF Project Files*

Description

Creates a directory for the OSF ID and downloads all of the files using a folder structure from the OSF project nodes and file storage structure. Returns (invisibly) a data frame with file info.

Usage

```
osf_file_download(
  osf_id,
  download_to = ".",
  max_file_size = 10,
  max_download_size = 100,
  max_folder_length = Inf,
  ignore_folder_structure = FALSE,
  pb = NULL
)
```

Arguments

osf_id	an OSF ID or URL
download_to	path to download to
max_file_size	maximum file size to download (in MB) - set to NULL for no restrictions
max_download_size	maximum total size to download
max_folder_length	maximum folder name length (set to make sure paths are <260 character on some Windows OS)
ignore_folder_structure	if TRUE, download all files into a single folder
pb	a progress bar passed from another function

Details

Some differences may exist because the OSF allows longer file names with characters that may not be allowed on a file system, so these are cleaned up when downloading.

You can limit downloads to only files under a specific size (defaults to 10MB) and only a maximum download size (largest files will be omitted until total size is under the limit). Omitted files will be listed as messages in verbose mode, and included in the returned data frame with the downloaded column value set to FALSE.

Value

data frame of file info

Examples

```
## Not run:
osf_file_download("6nt4v")

## End(Not run)
```

---

osf_get_all_pages *Get All OSF API Query Pages*

Description

OSF API queries only return up to 10 items per page, so this helper functions checks for extra pages and returns all of them

Usage

```
osf_get_all_pages(url, page_end = Inf)
```

Arguments

url	the OSF API URL
page_end	The last page to get

Value

a table of the returned data

Examples

```
# get the 20 newest preprints
## Not run:
osf_api <- getOption("metacheck.osf.api")
url <- sprintf("%s/preprints/?search=date_created-desc", osf_api)
preprints <- osf_get_all_pages(url, 2)

## End(Not run)
```

---

osf_info *Retrieve info from the OSF by ID*

Description

Retrieve info from the OSF by ID

Usage

```
osf_info(osf_url, id_col = 1, recursive = FALSE, pb = NULL)
```

Arguments

osf_url	an OSF ID or URL, or a table containing them
id_col	the index or name of the column that contains OSF IDs or URLs, if id is a table
recursive	whether to retrieve all children
pb	a progress bar passed from another function

Value

a data frame of information

Examples

```
## Not run:
# get info on one OSF node
osf_info("pngda")

# also get child nodes and files
osf_info("https://osf.io/6nt4v", recursive = TRUE)

## End(Not run)
```

`osf_links` *Find OSF Links in Papers*

Description

Get all OSF links.

Usage

```
osf_links(paper)
```

Arguments

`paper`	a paper object or paperlist object

Value

a table with the OSF url in the first (href) column

Examples

```
osf_links(psychsci)
```

`osf_preprint_list` *Get A list of preprints from the OSF*

Description

Get A list of preprints from the OSF

Usage

```
osf_preprint_list(
  provider = NULL,
  date_created = NULL,
  date_modified = NULL,
  page_start = 1,
  page_end = page_start
)
```

Arguments

provider	a vector of the preprint providers, e.g. psyarxiv, socarxiv, edarxiv (see https://osf.io/preprints/discover)
date_created	a single date or a vector of two date (min and max)
date_modified	a single date or a vector of two date (min and max)
page_start	the first page of 10 entries
page_end	the last page of 10 entires to read

Value

a table of preprint info

Examples

```
## Not run:
dc <- c("2025-09-01", "2025-10-01")
pp <- osf_preprint_list("psyarxiv", date_created = dc)
files <- pp$primary_file

## End(Not run)
```

---

osf_type *Get OSF GUID Type*

Description

Get OSF GUID Type

Usage

```
osf_type(guid)
```

Arguments

guid	the 5-letter GUID

Value

the type

Examples

```
# osf_type("pngda")
```

paper_id *Get Paper IDs*

Description

Get Paper IDs

Usage

```
paper_id(paper)
```

Arguments

paper	a paper or paperlist

Value

a vector of paper_ids

Examples

```
paper_id(psychsci)
```

paper_table *Paper tables*

Description

Return a table from a paper object or concatenate tables across a list of paper objects.

Usage

```
paper_table(paper, table, cols = NULL)
```

Arguments

paper	a paper or paperlist
table	a table name
cols	the columns to return from the table (default all columns)

Value

a merged table

Examples

```
biblio <- paper_table(psychsci[1:10], "bib")
xrefs <- paper_table(psychsci[1:10], "xref")
```

paper_validate *Validate a Paper Object*

Description

Checks if a paper object conforms to the JSON schema.

Usage

```
paper_validate(paper)
```

Arguments

paper	a paper object

Value

TRUE or error

Examples

```
paper <- list(paper_id = "Not a paper object")
tryCatch(
  paper_validate(paper),
  error = \(e) print(e$message)
)

paper <- demopaper()
paper_validate(paper)
```

paper_write *Write paper*

Description

Save a paper as a JSON file.

Usage

```
paper_write(paper, file_name = NULL, save_path = ".")
```

Arguments

paper	a paper object
file_name	the name of the file (if NULL, defaults to the paper_id)
save_path	the directory to save the JSON file in

Value

the path to the JSON file

Examples

```
## Not run:
paper <- demopaper()
paper$info$title <- "New title"
paper_write(paper, "new_paper")

## End(Not run)
```

`path_sanitize` *Sanitize File Path*

Description

Make sure user-input file names are not problematic.

Usage

```
path_sanitize(
  path,
  replacement = "_",
  remove_whitespace = TRUE,
  keep_sep = TRUE
)
```

Arguments

`path`	the path to sanitize (can be a vector of paths)
`replacement`	the character to replace invalid characters with
`remove_whitespace`	whether to include whitespace as a problem
`keep_sep`	whether to keep the path separator /

Value

the sanitized vector

Examples

```
path <- "/My Files/x><y.pdf"
path_sanitize(path)
path_sanitize(path, replacement = "~")
path_sanitize(path, remove_whitespace = FALSE)
path_sanitize(path, keep_sep = FALSE)
```

plural *Pluralise*

Description

Helper function for conditional plurals. For example, if you want to return "1 error" or "2 errors", you can use this in a sprintf().

Usage

```
plural(n, singular = "", plural = "s")
```

Arguments

n	the number
singular	the word or ending when n = 1
plural	the word or ending n != 1

Value

a string

Examples

```
n <- 0:3
sprintf("I have %d friend%s", n, plural(n))
sprintf("I have %d %s", n, plural(n, "octopus", "octopi"))
```

psychsci *Psychological Science Open Access Paper Set*

Description

250 open access papers from Psychological Science.

Usage

```
psychsci
```

Format

A list of 250 paper objects

Source

https://journals.sagepub.com/home/pss

pubpeer_comments *Get Pubpeer Comments*

Description

Takes a DOI, and retrieves information from pubpeer related to post-publication peer review comments.

Usage

```
pubpeer_comments(doi)
```

Arguments

doi	a vector of paper DOIs

Value

a dataframe with information from pubpeer

Examples

```
doi <- c(
  "10.1038/s41598-025-24662-9",
  "10.1177/0146167211398138"
)
pubpeer_comments(doi)
```

rbox_file_download *Retrieve files from ResearchBox by URL*

Description

Retrieve files from ResearchBox by URL

Usage

```
rbox_file_download(rb_url, pb = NULL)
```

Arguments

rb_url	a vector of ResearchBox URLs
pb	a progress bar passed from another function

Value

a data frame of information

rbox_info *Retrieve info from ResearchBox by URL*

Description

Retrieve info from ResearchBox by URL

Usage

```
rbox_info(rb_url, id_col = 1, pb = NULL)
```

Arguments

rb_url	an ResearchBox URL, or a table containing them (e.g., as created by `rbox_links()`)
id_col	the index or name of the column that contains ResearchBox URLs, if id is a table
pb	a progress bar passed from another function

Value

a data frame of information

Examples

```
## Not run:
# get info on one OSF node
rbox_info("https://researchbox.org/801")

## End(Not run)
```

rbox_links *Find ResearchBox Links in Papers*

Description

Find ResearchBox Links in Papers

Usage

```
rbox_links(paper)
```

Arguments

paper	a paper object or paperlist object

Value

a table with the ResearchBox url in the first (href) column

Examples

```
rbox_links(psychsci)
```

---

read *Read in grobid XML or bibr JSON*

Description

Read in grobid XML or bibr JSON

Usage

```
read(file_path, include_images = FALSE, recursive = FALSE)
```

Arguments

file_path	path to a single directory containing XML and/or JSON files, or a vector of XML/JSON paths
include_images	whether to include images in the figures table of the paper object (they make object size larger, only relevant to bibr imports)
recursive	whether to read files in subfolders (files should have unique paper_ids, or errors can occur)

Value

a paper or paperlist

---

ref_table *Reference and DOI table*

Description

Return a table with fixed DOIs and reference text from a paper object or concatenate tables across a list of paper objects.

Usage

```
ref_table(paper)
```

Arguments

paper	a paper or paperlist

Value

a merged table

Examples

```
biblio <- ref_table(psychsci[[1]])
```

`report` *Create a Report*

Description

Run specified modules on a paper and generate a report in quarto (qmd), html, or pdf format.

Usage

```
report(
  paper,
 modules = c("prereg_check", "funding_check", "coi_check", "power", "repo_check",
  "code_check", "stat_check", "stat_p_exact", "stat_p_nonsig", "stat_effect_size",
  "marginal", "ref_accuracy", "ref_replication", "ref_retraction", "ref_pubpeer",
   "ref_summary"),
  output_file = paste0(paper$paper_id, "_report.", output_format),
  output_format = c("html", "qmd"),
  args = list()
)
```

Arguments

`paper`	a paper object or a paperlist object
`modules`	a vector of modules to run (names for built-in modules or paths for custom modules)
`output_file`	the name of the output file
`output_format`	the format to create the report in
`args`	a list of arguments to pass to modules (see Details)

Details

Pass arguments to modules in a named list of lists, using the same names as the `modules` argument. You only need to specify modules with arguments.

```
args <- list(power = list(seed = 8675309))
```

Value

the file path the report is saved to

Examples

```
## Not run:
paper <- demopaper()
report(paper)

## End(Not run)
```

`report_app` *Launch Report App*

Description

Launch the Report app: upload a PDF and generate a report with one click, with privacy options for what is sent to external servers.

Usage

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

Arguments

`quiet`	whether to show debugging messages in the console
`...`	arguments to pass to shiny::runApp

Value

NULL (invisibly)

Examples

```
## Not run:
report_app()

## End(Not run)
```

report_module_run *Run modules for a report*

Description

Runs modules in order on the paper and orders by section and traffic light.

Usage

```
report_module_run(paper, modules, args = list())
```

Arguments

paper	a paper object
modules	a vector of modules to run
args	optional list of arguments to pass to modules

Details

Pass arguments to modules in a named list of lists, using the same names as the modules argument. You only need to specify modules with arguments.

```
args <- list(power = list(seed = 8675309))
```

Value

a list of module outputs

Examples

```
paper <- demopaper()
modules <- c("stat_p_exact", "stat_p_nonsig")
module_output <- report_module_run(paper, modules)
```

report_qmd *Create Report from Module Output*

Description

Create Report from Module Output

Usage

```
report_qmd(module_output, paper = list())
```

Arguments

module_output	a list of module output (usually from report_module_run())
paper	a paper object

Value

report text

---

report_table *Display a Table in a Report*

Description

A function to display tables in reports.

Usage

```
report_table(table, colwidths = "auto", maxrows = 2, escape = FALSE)
```

Arguments

table	the data frame to show in a table, or a vector for a list
colwidths	set column widths as a vector of px (number > 1) or percent (numbers <= 1)
maxrows	if the table has more rows than this, paginate
escape	whether or not to escape the DT (necessary if using raw html)

Value

the datatable

Examples

```
report_table(iris)
```

retractionwatch *RetractionWatch data*

Description

DOIs and nature of statements from the RetractionWatch database. Use `rw_date()` to find the date it was downloaded, and `rw_update()` to update it.

Usage

```
retractionwatch()

rw()
```

Format

A data frame with 44784+ rows and 2 columns:

doi Document ID

retractionwatch Nature of note(s)

Value

a data frame

Source

https://api.labs.crossref.org/data/retractionwatch

Examples

```
retractionwatch()
```

rw_date *Get date retractionwatch was updated*

Description

Get date retractionwatch was updated

Usage

```
rw_date()
```

Value

the date

Examples

```
rw_date()
```

`rw_update` *Update retractionwatch*

Description

metacheck comes with a built-in data frame called `retractionwatch`. We update it regularly, but you can use this function to download the newest version. The download is >50MB, but this function will summarise the information into a smaller version (~0.5 MB) and delete the original file.

Usage

```
rw_update()
```

Value

the path to the data frame (invisibly)

`scroll_table` *Make Scroll Table*

Description

A helper function for making module reports.

Usage

```
scroll_table(
  table,
  colwidths = "auto",
  maxrows = 2,
  escape = FALSE,
  column = "body"
)
```

Arguments

`table`	the data frame to show in a table, or a vector for a list
`colwidths`	set column widths as a vector of px (number > 1) or percent (numbers <= 1)
`maxrows`	if the table has more rows than this, paginate
`escape`	whether or not to escape the DT (necessary if using raw html)
`column`	which quarto column to show tables in

Details

See quarto article layout for column options. The most common are "body" (centre column), "page" (span all columns"), and "margin" (only in right margin).

To set colwidths, use a numeric or character vector. For a numeric vector, numbers greater than 1 will be interpreted as pixels, less than 1 as percents. Character vectors will be passed as is (e.g., "3em"). If you only want to specify some columns, set the others to NA, like c(200, NA, 200, NA).

Value

the markdown R chunk to create this table

Examples

```
scroll_table(LETTERS)
```

---

stats *Check Stats*

Description

Check Stats

Usage

```
stats(text, ...)
```

Arguments

text	the search table (or list of paper objects)
...	arguments to pass to statcheck()

Value

a table of statistics

Examples

```
paper <- demopaper()
stats(paper)
```

test_paper *Test paper*

Description

Create a paper object with the specified text (mainly for testing/demos).

Usage

```
test_paper(text = LETTERS, url = character(0))
```

Arguments

text	a vector of text to add
url	a vector of URLs to add

Value

a paper object

Examples

```
# to test a paper with a specific URL
p <- test_paper("https://osf.io/abcde")
```

text_expand *Expand text*

Description

If you have a table resulting from `text_search()` or a module return object, you can expand the text column to the full sentence, paragraph, or section. You can also set `plus` and `minus` to append and prepend sentences to the result (only when `expand_to` is "sentence").

Usage

```
text_expand(
  results_table,
  paper,
  expand_to = c("sentence", "paragraph", "div", "section"),
  plus = 0,
  minus = 0
)

expand_text(
  results_table,
```

```
  paper,
  expand_to = c("sentence", "paragraph", "div", "section"),
  plus = 0,
  minus = 0
)
```

Arguments

results_table	the table to expand
paper	a metacheck paper object or a list of paper objects to look up the expanded text from
expand_to	whether to expand to the sentence, paragraph, div, or section level
plus	append additional sentences after the target expansion
minus	prepend additional sentences before the target expansion

Value

a results table with the expanded text

Examples

```
# single paper search
paper <- demopaper()
res_tbl <- text_search(paper, "p =", return = "match")
expanded <- text_expand(res_tbl, paper)

# multiple paper search
papers <- psychsci
res_tbl <- text_search(papers, "replicate")
expanded <- text_expand(res_tbl, papers, plus = 1, minus = 1)
```

---

text_search *Search text*

Description

Search the text of a paper or list of paper objects. Also works on the table results of a `text_search()` call.

Usage

```
text_search(
  paper,
  pattern = ".*",
  return = c("sentence", "paragraph", "section", "header", "match", "paper_id"),
  ignore.case = TRUE,
  fixed = FALSE,
```

```
  perl = FALSE,
  exclude = FALSE,
  search_header = FALSE,
  include_refs = FALSE
)

search_text(
  paper,
  pattern = ".*",
  return = c("sentence", "paragraph", "section", "header", "match", "paper_id"),
  ignore.case = TRUE,
  fixed = FALSE,
  perl = FALSE,
  exclude = FALSE,
  search_header = FALSE,
  include_refs = FALSE
)
```

Arguments

paper	a paper object or a list of paper objects
pattern	the regex pattern to search for, if a vector with length > 1, the patterns will be searched separately and combined
return	the kind of text to return, the full sentence, paragraph, header, or section that the text is in, or just the (regex) match, or all body text for a paper (paper_id)
ignore.case	whether to ignore case when text searching
fixed	logical. If TRUE, pattern is a string to be matched as is. Overrides all conflicting arguments.
perl	logical. Should Perl-compatible regexps be used?
exclude	should matches be included or excluded
search_header	also search the header
include_refs	whether to include the reference section in the search

Details

The section argument can take a vector of section names, or a PERL regular expression (use ".*" to match all sections). Possible section types are abstract, intro, method, results, discussion, references, acknowledgment, funding, endnote, footnote, table, figure, and unknown. The default includes all sections except references, tables and figures.

Value

a data frame of matches

Examples

```
paper <- demopaper()
all_text <- text_search(paper)
study <- text_search(paper, "study")
equations <- text_search(paper, "\\b\\S+\\s*(=|<)\\s*[0-9\\.]+", return = "match")
no_numbers <- text_search(paper, "\\d", exclude = TRUE)
```

`validate` *Validate*

Description

Validate

Usage

```
validate(gt, module, compare = "table")
```

Arguments

`gt`	a data frame or vector of text
`module`	the module
`compare`	name of the module output table for comparison

Value

something

Examples

```
validate("p < .05", "stat_p_exact")
```

`zenodo_file_download` *Download all Zenodo Project Files*

Description

Creates a directory for the Zenodo ID and downloads all of the files using a folder structure from the Zenodo project nodes and file storage structure. Returns (invisibly) a data frame with file info.

Usage

```
zenodo_file_download(
  zenodo_id,
  download_to = ".",
  max_file_size = 10,
  max_download_size = 100,
  pb = NULL
)
```

Arguments

zenodo_id	an Zenodo ID or URL
download_to	path to download to
max_file_size	maximum file size to download (in MB) - set to NULL or Inf for no restrictions
max_download_size	maximum total size to download - set to NULL of Inf for no restrictions
pb	a progress bar passed from another function

Details

You can limit downloads to only files under a specific size (defaults to 10MB) and only a maximum download size (largest files will be omitted until total size is under the limit). Omitted files will be listed as messages in verbose mode, and included in the returned data frame with the downloaded column value set to FALSE.

Value

data frame of file info

Examples

```
## Not run:
  zenodo_file_download("2591593")

## End(Not run)
```

---

zenodo_info *Retrieve info from Zenodo by URL*

Description

Retrieve info from Zenodo by URL

Usage

```
zenodo_info(zenodo_url, id_col = 1, pb = NULL)
```

Arguments

zenodo_url	an Zenodo URL, or a table containing them (e.g., as created by zenodo_links())
id_col	the index or name of the column that contains Zenodo URLs, if id is a table
pb	a progress bar passed from another function

Value

a data frame of information

Examples

```
## Not run:
  # get info on one zenodo link
  zenodo_info("https://doi.org/10.5281/zenodo.18648142")

## End(Not run)
```

---

zenodo_links *Find Zenodo Links in Papers*

Description

Find Zenodo Links in Papers

Usage

```
zenodo_links(paper)
```

Arguments

paper	a paper object or paperlist object

Value

a table with the Zenodo url in the first (text) column

Examples

```
zenodo_links(psychsci)
```

Index