Package: rtransparency (via r-universe)

November 18, 2026

Type Package

Title Identifies Indicators of Transparency

Version 1.2.0

Description Use this package to identify indicators of transparency within the published literature. It can identify and extract text related to indicators of transparency from specifically formatted TXT files and from PMC XML files (i.e. XML files downloaded from the PubMed Central). It builds on the original 'rtransparent' tool of Serghiou et al. (2021) <doi:10.1371/journal.pbio.3001107>.

License GPL-3

Encoding UTF-8

LazyData true

URL https://github.com/choxos/rtransparency, https://choxos.github.io/rtransparency/

BugReports https://github.com/choxos/rtransparency/issues

Depends R (>= 3.5.0)

Imports dplyr (>= 1.0.0), magrittr, purrr (>= 1.0.0), readr, rlang (>= 1.0.0), stringi, stringr, tibble, xml2

Suggests furrr, future, ggplot2, jsonlite, knitr, pdftools, readxl, rmarkdown, testthat (>= 3.0.0)

Config/Needs/benchmark readxl

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libicu-dev libxml2-dev libx11-dev

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

Date/Publication 2026-09-23 13:37:44 UTC

RemoteUrl https://github.com/choxos/rtransparency

RemoteRef HEAD

RemoteSha 7197498f09d35f1c47200c4e902ca7bf2cc56a2f

Contents

rtransparency-package . . . 3
rt_accuracy . . . 3
rt_accuracy_2021 . . . 5
rt_ai . . . 5
rt_ai_pmc . . . 6
rt_all . . . 7
rt_all_pdf . . . 8
rt_all_pmc . . . 9
rt_all_pmc_dir . . . 10
rt_all_txt_dir . . . 12
rt_authors_pmc . . . 13
rt_check_links . . . 14
rt_coi . . . 15
rt_coi_pmc . . . 16
rt_coi_pubmed . . . 16
rt_convert_ids . . . 17
rt_data_code . . . 18
rt_data_code_pmc . . . 19
rt_data_code_pmc_list . . . 20
rt_demo . . . 21
rt_ethics . . . 22
rt_ethics_pmc . . . 22
rt_fetch_pmc . . . 23
rt_fill_coi_pubmed . . . 25
rt_fund . . . 26
rt_fund_pmc . . . 27
rt_funders_pmc . . . 28
rt_meta_pmc . . . 29
rt_novelty . . . 29
rt_novelty_pmc . . . 30
rt_oa . . . 31
rt_oa_pmc . . . 32
rt_plot . . . 33
rt_read_pdf . . . 34
rt_register . . . 35
rt_register_pmc . . . 36
rt_registration_timing . . . 37
rt_replication . . . 38
rt_replication_pmc . . . 39
rt_reporting . . . 39
rt_reporting_pmc . . . 40
rt_score . . . 41
rt_summary . . . 42
rt_trial_ids . . . 44

Index **45**

`rtransparency-package` *rtransparency: Identify indicators of transparency in the biomedical literature*

Description

Detects and extracts ten indicators of transparency (conflicts of interest, funding, protocol registration, novelty, replication, data sharing, code sharing, disclosure of generative-AI use, open-access licensing, and reporting-guideline use) from PubMed Central XML or plain-text articles. For each indicator it returns a prediction and the statement or value that triggered it. This package builds on the original **rtransparent** tool of Serghiou et al. (2021).

Author(s)

Maintainer: Ahmad Sofi-Mahmudi <a.sofimahmudi@gmail.com> (ORCID)

Authors:

- Ahmad Sofi-Mahmudi <a.sofimahmudi@gmail.com> (ORCID)
- Stylianos Serghiou <stelios.serghiou@gmail.com>

See Also

Useful links:

- https://github.com/choxos/rtransparency
- https://choxos.github.io/rtransparency/
- Report bugs at https://github.com/choxos/rtransparency/issues

`rt_accuracy` *Detector accuracy estimates*

Description

Sensitivity and specificity of each transparency detector, with the validation counts behind them, used by [rt_summary()] to correct an apparent prevalence for detector error (the Rogan-Gladen correction) and to carry the uncertainty of these estimates into the corrected interval.

Usage

```
rt_accuracy
```

Format

A tibble with 8 rows and 9 columns:

variable Indicator column name, as returned by [rt_all_pmc()].

label Human-readable indicator name.

sensitivity Detector sensitivity (true-positive rate), 0-1.

specificity Detector specificity (true-negative rate), 0-1.

tp, fn True positives and false negatives behind ‘sensitivity‘.

tn, fp True negatives and false positives behind ‘specificity‘.

source Where the estimate comes from.

Details

Every row describes the detectors shipped in this version, scored on hand-labeled articles (see ‘inst/benchmark/results_all_sets.csv‘ and the reports beside it):

* Conflicts of interest, funding and registration: the held-out, independently labeled test set of Serghiou et al. (2021). These are not the paper’s published values, which describe the 2021 detectors; those are kept in [rt_accuracy_2021]. * Data and code sharing: the same held-out set’s data and code labels. The native detector was developed against this set, so these are regression estimates rather than an untouched validation. * Novelty: the maintainer’s hand-labeled novelty/replication gold set. * Replication: sensitivity from a replication-enriched sample (111 positives) and specificity from the representative 2023 sample, so the correction mixes designs. * Reporting guideline: the 1000-article 2023 sample, hand-labeled.

Open-access licensing (structured metadata whose specificity cannot be estimated in the open-access subset) and AI-use disclosure (too few positives) are not included, so [rt_summary()] reports them uncorrected. Supply your own table to [rt_summary()] via its ‘accuracy‘ argument when you have study-specific or external estimates; the ‘data-raw/validation/‘ scripts produce one in this format.

Source

This package’s benchmarks (‘inst/benchmark/‘), including the held-out labels of Serghiou S, Contopoulos-Ioannidis DG, Boyack KW, Riedel N, Wallach JD, Ioannidis JPA (2021). Assessment of transparency indicators across the biomedical literature: How open is open? *PLOS Biology* 19(3): e3001107. doi:10.1371/journal.pbio.3001107.

See Also

[rt_summary()], [rt_accuracy_2021]

rt_accuracy_2021 *Published 2021 detector accuracy estimates*

Description

The importance-weighted sensitivity and specificity that Serghiou et al. (2021) published for their conflict-of-interest, funding and registration detectors. They describe the 2021 detectors, not the current ones (see [rt_accuracy]), and are kept for comparability with that paper: ‘rt_summary(data, accuracy = rt_accuracy_2021)‘. Without validation counts, [rt_summary()] uses the fixed interval for them.

Usage

```
rt_accuracy_2021
```

Format

A tibble with 3 rows and 5 columns: ‘variable‘, ‘label‘, ‘sensitivity‘, ‘specificity‘ and ‘source‘.

Source

Serghiou et al. (2021), PLOS Biology 19(3): e3001107. doi:10.1371/journal.pbio.3001107.

See Also

[rt_accuracy]

rt_ai *Identify disclosure of generative-AI use from a TXT file.*

Description

Detects whether an article discloses the use (or non-use) of generative AI or AI-assisted tools in preparing the manuscript, from a plain-text (typically PDF-derived) file. Unlike [rt_ai_pmc()] it applies **no publication-year gate**: a plain-text file carries no reliable publication date, so ‘is_ai_pred‘ is always ‘TRUE‘ or ‘FALSE‘ (never ‘NA‘). AI-use disclosure became an expected practice only in 2023, so the caller is responsible for restricting analysis to articles from 2023 onward. Plain text also lacks the section structure the PMC detector uses to confine the scan to back matter, acknowledgments and declaration sections, so an article that uses AI purely as a research method is more likely to be flagged than under [rt_ai_pmc()].

Usage

```
rt_ai(filename = NULL, text = NULL)
```

Arguments

filename	The path to a TXT file as a string.
text	Alternatively, the article text itself as a character vector (for example the output of [rt_read_pdf()]). Supply ‘filename‘ or ‘text‘.

Value

A tibble with the file name (‘article‘), the PMID (‘NA‘ if absent), whether an AI-use disclosure was found (‘is_ai_pred‘), the matched statement (‘ai_text‘) and what it says (‘ai_used‘, ‘ai_tools‘, ‘ai_purpose‘), as described in [rt_ai_pmc()].

See Also

[rt_ai_pmc()] for the PMC XML detector, which applies the 2023 publication-year gate.

Examples

```
# Write a short example article to a temporary text file.
filepath <- file.path(tempdir(), "PMID00000000-PMC0000000.txt")
writeLines(
  "The authors used ChatGPT to assist with drafting this manuscript.",
  filepath
)

# Identify and extract an AI-use disclosure.
rt_ai(filepath)
```

---

rt_ai_pmc *Identify disclosure of generative-AI use from a PMC XML file.*

Description

Detects whether an article discloses the use (or non-use) of generative AI or AI-assisted tools in preparing the manuscript, as required of articles since 2023. The indicator is only evaluated for articles published in 2023 or later; for earlier articles ‘is_ai_pred‘ is ‘NA‘.

Usage

```
rt_ai_pmc(filename, remove_ns = TRUE)
```

Arguments

filename	The filename of the PMC XML file to analyze.
remove_ns	Ignored since version 1.2.0 and kept for backward compatibility. Default XML namespaces are now always removed, so a namespaced PMC XML file gives the same result as a plain one.

Details

The year gate uses the earliest publication year the XML records (electronic, print or collection date), so an article first published online in December 2022 is gated out even if its issue is dated 2023.

A disclosure can state use or non-use, and ‘is_ai_pred‘ counts both. ‘ai_used‘ separates them: ‘TRUE‘ when a disclosure states that AI was used, ‘FALSE‘ when it states that no AI was used, and ‘NA‘ when there is no disclosure or the use cannot be read from it (for example when only a section title was found). ‘ai_tools‘ names the tools mentioned in statements of use (for example ‘"ChatGPT; DeepL"‘) and ‘ai_purpose‘ the stated purposes, from ‘"language editing"‘, ‘"translation"‘, ‘"drafting"‘, ‘"figures and images"‘, ‘"code and analysis"‘ and ‘"literature search"‘. These are read with the same rules as the disclosure itself and have not been separately validated.

Value

A tibble with the article IDs, the publication ‘year‘, whether an AI disclosure was found (‘is_ai_pred‘, ‘NA‘ before 2023), the matched statement (‘ai_text‘), what the disclosure says (‘ai_used‘, ‘ai_tools‘, ‘ai_purpose‘; see Details) and ‘is_success‘.

Examples

```
filepath <- system.file(
  "extdata", "PMID32171256-PMC7071725.xml", package = "rtransparency"
)
rt_ai_pmc(filepath)
```

`rt_all` *Identify and extract all transparency indicators from a TXT file.*

Description

Takes a plain-text article (a TXT file, or the text itself) and returns all ten indicators of transparency the package detects: conflicts of interest, funding, protocol registration, novelty, replication, data sharing, code sharing, generative-AI-use disclosure, open-access licensing and reporting-guideline use. The file is read once and every detector runs on the same text, with the same logic as the standalone plain-text functions ([rt_coi()], [rt_fund()], [rt_register()], [rt_novelty()], [rt_replication()], [rt_data_code()], [rt_ai()], [rt_oa()], [rt_reporting()]).

Usage

```
rt_all(filename = NULL, text = NULL)
```

Arguments

`filename`	The path to a TXT file as a string.
`text`	Alternatively, the article text itself as a character vector (for example the output of [rt_read_pdf()]). Supply ‘filename‘ or ‘text‘.

Value

A one-row tibble: the file name (‘article‘) and PMID (‘pmid‘, the digits after "PMID" in the file name, ‘NA‘ if absent), then each indicator with the text that triggered it. The indicator columns carry the same names as in [rt_all_pmc()] (‘is_coi_pred‘, ‘is_fund_pred‘, ‘is_register_pred‘, ‘is_novelty_pred‘, ‘is_replication_pred‘, ‘is_open_data‘, ‘is_open_code‘, ‘is_ai_pred‘, ‘is_open_access‘, ‘is_reporting_pred‘), so the result can be passed to [rt_summary()]. The pattern-function flags of the novelty and replication detectors are also returned; if one is ‘NA‘ it was not run. Unlike [rt_all_pmc()], ‘is_ai_pred‘ has no publication-year gate (see [rt_ai()]). ‘is_funded_pred‘ and ‘funding_text‘ are deprecated copies of the funding columns, kept for one release.

See Also

[rt_all_pdf()] for a PDF, [rt_all_txt_dir()] for many files, and [rt_all_pmc()] for PMC XML.

Examples

```
# Write a short example article to a temporary text file.
filepath <- file.path(tempdir(), "PMID00000000-PMC0000000.txt")
writeLines(c(
  "To our knowledge, this is the first study of its kind.",
  "Conflicts of interest: none declared.",
  "This work was supported by the National Institutes of Health (R01-000000).",
  "The protocol was registered at ClinicalTrials.gov (NCT00000000).",
  "All data and code are available at https://github.com/example/repo.",
  "We independently replicated the original analysis."
), filepath)

# Identify and extract indicators of transparency.
results_table <- rt_all(filepath)

# The same, from text already in memory.
results_table <- rt_all(text = readLines(filepath))
```

`rt_all_pdf` *Identify and extract all transparency indicators from a PDF file.*

Description

Converts a PDF to text with [rt_read_pdf()] and runs [rt_all()] on it, so a PDF can be scored in one call without writing an intermediate text file. Requires the poppler ‘pdftotext‘ utility.

Usage

```
rt_all_pdf(filepath)
```

Arguments

`filepath`	The path to the PDF file as a string.

Value

The same one-row tibble as [rt_all()], with ‘article‘ and ‘pmid‘ taken from the PDF file name.

See Also

[rt_all()], [rt_read_pdf()], [rt_all_txt_dir()]

Examples

```
## Not run:
pdf_path <- system.file(
  "extdata", "PMID32171256-PMC7071725.pdf", package = "rtransparency"
)
rt_all_pdf(pdf_path)

## End(Not run)
```

`rt_all_pmc` *Identify and extract all transparency indicators from a PMC XML.*

Description

Takes a PMC XML and returns relevant meta-data, as well as whether the article carries each of the ten transparency indicators: Conflicts of Interest (COI), Funding, Protocol Registration, Novelty, Replication, Data sharing, Code sharing, disclosure of generative-AI use, Open-access licensing and Reporting-guideline use. Where a statement is found, the relevant text is also extracted. This is the single-call entry point; it covers the same data and code detection as [rt_data_code_pmc()], the same AI detection as [rt_ai_pmc()], the same licensing detection as [rt_oa_pmc()] and the same reporting-guideline detection as [rt_reporting_pmc()].

Usage

```
rt_all_pmc(filename, remove_ns = TRUE, all_meta = FALSE)
```

Arguments

`filename`	The name of the PMC XML as a string.
`remove_ns`	Ignored since version 1.2.0 and kept for backward compatibility. Default XML namespaces are now always removed, so a namespaced PMC XML file gives the same result as a plain one.
`all_meta`	TRUE extracts all meta-data, FALSE extracts some (default).

Value

A dataframe of results. It returns the unique identifiers of the article, whether each indicator of transparency was identified (‘is_coi_pred‘, ‘is_fund_pred‘, ‘is_register_pred‘, ‘is_novelty_pred‘, ‘is_replication_pred‘, ‘is_open_data‘, ‘is_open_code‘, the year-gated ‘is_ai_pred‘, ‘is_open_access‘ with the ‘oa_license‘, and ‘is_reporting_pred‘ with the named ‘reporting_guideline‘), the relevant text identified, whether it was identified through a dedicated XML tag (such variables include "pmc" in their name, e.g. “fund_pmc_source”) and whether each labelling function identified relevant text or not. The labeling functions are returned to add flexibility in how this package is used; for example, future definitions of Registration may differ from the one we used. If a labelling function returns NA it means that it was not run. ‘is_ai_pred‘ is ‘NA‘ for articles published before 2023 (see [rt_ai_pmc()]).

Examples

```
# Path to a bundled example PMC XML file.
filepath <- system.file(
  "extdata", "PMID32171256-PMC7071725.xml", package = "rtransparency"
)

# Identify and extract meta-data and indicators of transparency.
results_table <- rt_all_pmc(filepath, all_meta = TRUE)
```

`rt_all_pmc_dir` *Identify transparency indicators across many PMC XML files.*

Description

A batch wrapper around [rt_all_pmc()] for corpus-scale runs over a directory (or an explicit vector) of PMC XML files. It isolates per-file failures so a single malformed file cannot abort the run, shows a progress bar, can resume an interrupted run, and can run in parallel when the **furrr** package is installed.

Usage

```
rt_all_pmc_dir(
  dir,
  pattern = "\\.xml$",
  recursive = FALSE,
  remove_ns = TRUE,
  all_meta = FALSE,
  output = NULL,
  parallel = FALSE,
  progress = TRUE,
  chunk_size = 200L
)
```

Arguments

`dir`	A directory containing PMC XML files, or a character vector of file paths.
`pattern`	A regular expression for file names, used only when ‘dir‘ is a single existing directory (default ‘"\.xml$"‘).
`recursive`	Whether to descend into subdirectories when ‘dir‘ is a directory (default ‘FALSE‘).
`remove_ns, all_meta`	Passed through to [rt_all_pmc()].
`output`	Optional path to a CSV file for incremental, resumable output (see Details). ‘NULL‘ (default) keeps results in memory only.
`parallel`	Whether to process files in parallel via **furrr** (default ‘FALSE‘).
`progress`	Whether to show a progress bar (default ‘TRUE‘).
`chunk_size`	Number of files per write/flush when ‘output‘ is set (default ‘200‘).

Details

When ‘output‘ is supplied, results are written to that CSV in chunks as the run proceeds. Re-running with the same ‘output‘ skips files already present in it and appends only the new results, so a long run can be resumed after an interruption. File names are recorded as absolute paths and resuming compares normalized paths, so it also works from another working directory or with relative instead of absolute paths. Each file is processed inside [tryCatch()]; a file that errors contributes a row with ‘is_success = FALSE‘ and the error message in ‘error‘ rather than stopping the run.

Parallelism uses **furrr**’s ‘future_map()‘ and honors whatever ‘future::plan()‘ is active (for example ‘future::plan("multisession")‘); with no plan it runs sequentially. Install **furrr** and **future** to use it.

Value

A [tibble][tibble::tibble] with one row per file (‘filename‘ is the file’s absolute path), carrying the same columns as [rt_all_pmc()] (plus any rows read back from a pre-existing ‘output‘). Files that could not be processed have ‘is_success = FALSE‘ and the reason in ‘error‘.

See Also

[rt_all_pmc()] for a single file.

Examples

```
# Process every PMC XML in a directory (here, the bundled example file).
dir <- system.file("extdata", package = "rtransparency")
out <- tempfile(fileext = ".csv")
res <- rt_all_pmc_dir(dir, output = out, parallel = FALSE)
```

`rt_all_txt_dir` *Identify transparency indicators across many TXT or PDF files.*

Description

The plain-text counterpart of [rt_all_pmc_dir()]: runs [rt_all()] on every text file, and [rt_all_pdf()] on every PDF, in a directory (or an explicit vector of paths), with the same per-file error isolation, progress bar, resumable CSV output and optional parallelism.

Usage

```
rt_all_txt_dir(
  dir,
  pattern = "\\.(txt|pdf)$",
  recursive = FALSE,
  output = NULL,
  parallel = FALSE,
  progress = TRUE,
  chunk_size = 200L
)
```

Arguments

`dir`	A directory containing TXT and/or PDF files, or a character vector of file paths.
`pattern`	A regular expression for file names, used only when ‘dir‘ is a single existing directory (default: ‘.txt‘ and ‘.pdf‘ files).
`recursive`, `parallel`, `progress`, `chunk_size`	As in [rt_all_pmc_dir()].
`output`	Optional path to a CSV file for incremental, resumable output, with the same behavior as in [rt_all_pmc_dir()].

Value

A [tibble][tibble::tibble] with one row per file: ‘filename‘ (the path), the columns of [rt_all()], ‘is_success‘ and ‘error‘.

See Also

[rt_all()], [rt_all_pdf()], [rt_all_pmc_dir()]

Examples

```
d <- file.path(tempdir(), "rt_txt_example")
dir.create(d, showWarnings = FALSE)
writeLines("Conflicts of interest: none declared.",
           file.path(d, "PMID00000001.txt"))
writeLines("This work was funded by the Wellcome Trust (grant 12345).",
```

```
            file.path(d, "PMID00000002.txt"))
res <- rt_all_txt_dir(d, progress = FALSE)
```

`rt_authors_pmc` *Author identifiers and contribution roles from a PMC XML file.*

Description

Reads the article's author list and reports how many authors carry an ORCID identifier and whether contributions are described with the CRediT (Contributor Roles Taxonomy) vocabulary. Both are structured JATS elements (‘<contrib-id contrib-id-type="orcid">‘ and ‘<role>‘), so they are read, not inferred from text.

Usage

```
rt_authors_pmc(filename, remove_ns = TRUE)
```

Arguments

`filename`	The name of the PMC XML as a string.
`remove_ns`	Ignored since version 1.2.0 and kept for backward compatibility. Default XML namespaces are now always removed, so a namespaced PMC XML file gives the same result as a plain one.

Details

A role counts as CRediT when it is tagged with the CRediT vocabulary (‘vocab="credit"‘) or its text is one of the 14 CRediT roles.

Value

A one-row tibble with the article IDs, ‘n_authors‘ (contributors of type author), ‘n_orcid‘ (authors with an ORCID), ‘orcid_coverage‘ (‘n_orcid / n_authors‘), ‘orcids‘ (the ORCID iDs, ‘"; "‘-separated), ‘has_credit‘ (whether any author has a CRediT role), ‘credit_roles‘ (the distinct CRediT roles used) and ‘is_success‘.

See Also

[rt_funders_pmc()], [rt_all_pmc()]

Examples

```
filepath <- system.file(
  "extdata", "PMID32171256-PMC7071725.xml", package = "rtransparency"
)
rt_authors_pmc(filepath)
```

`rt_check_links` *Check whether extracted data and code links resolve*

Description

Takes the links extracted by the data and code detectors (the ‘open_data_links‘ and ‘open_code_links‘ columns of [rt_data_code_pmc()], [rt_all_pmc()] or [rt_data_code()]) and checks whether each one resolves, following redirects. A shared-data statement whose link is dead is weaker evidence of sharing than one whose link works, which availability-statement indicators alone cannot tell apart.

Usage

```
rt_check_links(links, timeout = 10)
```

Arguments

`links`	A character vector of links; elements holding several links separated by ‘" ; "‘ (as the detectors return them) are split.
`timeout`	Seconds to wait for each server.

Details

DOIs are checked through ‘https://doi.org/‘, and database accessions in identifiers.org ‘prefix:accession‘ form through ‘https://identifiers.org/‘. Only the response headers are requested. Some servers refuse automated requests (status 403) or header-only requests (405) although the page exists, so a failing status is a prompt to look, not proof of a dead link.

Value

A tibble with one row per unique link: the ‘link‘, the ‘url‘ checked, the HTTP ‘status‘ (‘NA‘ when the server could not be reached), ‘is_ok‘ (a status below 400) and the ‘error‘ message when unreachable.

See Also

[rt_data_code_pmc()]

Examples

```
# Needs internet access; unreachable links are reported, not errors.
res <- rt_data_code_pmc(system.file(
  "extdata", "PMID32171256-PMC7071725.xml", package = "rtransparency"
))
if (capabilities("libcurl")) rt_check_links(res$open_data_links)
```

rt_coi *Identify and extract Conflicts of Interest (COI) statements in TXT files.*

Description

Takes a TXT file and returns data related to the presence of a COI statement, including whether a COI statement exists. If a COI statement exists, it extracts it. Detection runs through the same text helpers as [rt_coi_pmc()], so a plain-text article is scored with the same logic as a PMC XML one (only the XML-structural routes, which need tags a TXT file does not have, are unavailable).

Usage

```
rt_coi(filename = NULL, text = NULL)
```

Arguments

filename	The path to a TXT file as a string.
text	Alternatively, the article text itself as a character vector (for example the output of [rt_read_pdf()]). Supply ‘filename‘ or ‘text‘.

Value

A tibble with the file name (‘article‘), the PMID (the digits after "PMID" in the file name, ‘NA‘ if absent or for ‘text‘), whether a COI statement was found (‘is_coi_pred‘) and the text identified (‘coi_text‘).

Examples

```
# Write a short example article to a temporary text file.
filepath <- file.path(tempdir(), "PMID00000000-PMC0000000.txt")
writeLines(c(
  "To our knowledge, this is the first study of its kind.",
  "Conflicts of interest: none declared.",
  "This work was supported by the National Institutes of Health (R01-000000).",
  "The protocol was registered at ClinicalTrials.gov (NCT00000000).",
  "All data and code are available at https://github.com/example/repo.",
  "We independently replicated the original analysis."
), filepath)

# Identify and extract the COI statement.
results_table <- rt_coi(filepath)
```

`rt_coi_pmc` *Identify and extract Conflicts of Interest (COI) statements in PMC XML files.*

Description

Takes a PMC XML file and returns data related to the presence of a COI statement, including whether a COI statement exists. If a COI statement exists, it extracts it.

Usage

```
rt_coi_pmc(filename, remove_ns = TRUE)
```

Arguments

`filename`	The name of the PMC XML as a string.
`remove_ns`	Ignored since version 1.2.0 and kept for backward compatibility. Default XML namespaces are now always removed, so a namespaced PMC XML file gives the same result as a plain one.

Value

A dataframe of results. It returns unique article identifiers, whether this article was deemed relevant to COI, whether a COI was found, the text that suggested the presence of COI and the name of the function that identified this text. The functions are returned to add flexibility in how this package is used, such as future definitions of COI that may differ from the one we used.

Examples

```
# Path to a bundled example PMC XML file.
filepath <- system.file(
  "extdata", "PMID32171256-PMC7071725.xml", package = "rtransparency"
)

# Identify and extract meta-data and indicators of transparency.
results_table <- rt_coi_pmc(filepath)
```

`rt_coi_pubmed` *Fetch conflict-of-interest statements recorded in PubMed*

Description

Retrieves the ‘<CoiStatement>‘ that PubMed records for an article, which holds the conflict-of-interest disclosure also when it is absent from the full-text XML (the main source of missed disclosures in the XML detector).

Usage

```
rt_coi_pubmed(pmids, api_key = Sys.getenv("ENTREZ_KEY"))
```

Arguments

pmids	A character or numeric vector of PubMed IDs.
api_key	An NCBI API key; defaults to the 'ENTREZ_KEY' environment variable.

Value

A tibble with one row per unique PubMed ID: 'pmid', 'has_coi_statement' and 'coi_statement' ('""' when PubMed has none).

See Also

[rt_fill_coi_pubmed()], [rt_coi_pmc()]

Examples

```
# Needs internet access.
try(rt_coi_pubmed(c("32171256", "36696006")))
```

---

rt_convert_ids *Convert article identifiers with the PMC ID Converter*

Description

Maps PubMed IDs, PMCIDs and DOIs to one another with the NCBI PMC ID Converter API. Only articles in PubMed Central have a PMCID.

Usage

```
rt_convert_ids(ids, email = NULL)
```

Arguments

ids	A character vector of PubMed IDs, PMCIDs or DOIs (mixed freely). Bare numbers are treated as PubMed IDs.
email	An optional contact email passed to NCBI, as its usage guidelines request for heavy use.

Value

A tibble with one row per input: the input 'id', and its 'pmcid', 'pmid' and 'doi' ('NA' where not found).

See Also

[rt_fetch_pmc()]

Examples

```
# Needs internet access.
try(rt_convert_ids(c("32171256", "10.1186/s12874-020-0914-6", "PMC7071725")))
```

`rt_data_code` *Identify and extract Data and Code statements in TXT files.*

Description

Takes a TXT file and returns data related to the presence of Data and/or Code statements, including whether Data and/or Code statements exist. If such statements exist, it extracts them.

Usage

```
rt_data_code(filename = NULL, text = NULL)
```

Arguments

`filename`	The path to a TXT file as a string.
`text`	Alternatively, the article text itself as a character vector (for example the output of [rt_read_pdf()]). Supply ‘filename‘ or ‘text‘.

Value

A tibble with the file name (‘article‘), the PMID (‘NA‘ if absent), whether data or code sharing was found (‘is_open_data‘, ‘is_open_code‘), the statements that triggered each (‘open_data_statements‘, ‘open_code_statements‘) and the identifiers extracted from them (‘open_data_links‘, ‘open_code_links‘), with the same columns and meaning as [rt_data_code_pmc()].

Examples

```
# Write a short example article to a temporary text file.
filepath <- file.path(tempdir(), "PMID00000000-PMC0000000.txt")
writeLines(c(
  "To our knowledge, this is the first study of its kind.",
  "Conflicts of interest: none declared.",
  "This work was supported by the National Institutes of Health (R01-000000).",
  "The protocol was registered at ClinicalTrials.gov (NCT00000000).",
  "All data and code are available at https://github.com/example/repo.",
  "We independently replicated the original analysis."
), filepath)

# Identify and extract data and code availability.
results_table <- rt_data_code(filepath)
```

`rt_data_code_pmc` *Identify and extract Data and Code sharing from PMC XML files.*

Description

Takes a PMC XML file and returns data related to the presence of Data or Code, including whether Data or Code have been shared. If Data or Code exist, it will extract the relevant text for each. Detection is performed by the native detector (`.detect_data_code`); the package no longer depends on oddpub or `tokenizers`.

Usage

```
rt_data_code_pmc(filename, remove_ns = TRUE, specificity = NULL)
```

Arguments

`filename`	The filename of the XML file to be analyzed as a string.
`remove_ns`	Ignored since version 1.2.0 and kept for backward compatibility. Default XML namespaces are now always removed, so a namespaced PMC XML file gives the same result as a plain one.
`specificity`	Deprecated and ignored (it has not changed the result since the native detector replaced oddpub); supplying it gives a warning, and it will be removed in a future release. The native detector extracts a fixed, broad set of article text (body paragraphs and titles, back matter, footnotes and supplements) and applies repository, accession and availability-statement patterns.

Value

A dataframe of results: the unique IDs of the article, whether data or code sharing was found (`is_open_data`, `is_open_code`), the statement text that triggered each detection (`open_data_statements`, `open_code_statements`) and the persistent identifiers and URLs of what was shared (`open_data_links`, `open_code_links`). The links are the DOIs (as `doi.org` URLs), repository URLs and database accessions (as identifiers.org `prefix:accession`) extracted from the statements, separated by `" ; "`. `has_das` records whether the article has a data-availability section (tagged, or titled as such) and `das_text` holds its text: having the statement is itself a reporting indicator, separate from whether data were actually shared (a statement can say data are available only on request).

Examples

```
# Path to PMC XML
filepath <- system.file(
  "extdata", "PMID32171256-PMC7071725.xml", package = "rtransparency"
)

# Identify and extract indicators of data and code sharing
results_table <- rt_data_code_pmc(filepath)
```

`rt_data_code_pmc_list` *Identify and extract Data and Code sharing from a list of PMC XML files.*

Description

Takes a list of PMC XML files and returns data related to the presence of Data or Code, including whether Data or Code have been shared. If Data or Code exist, it will extract the relevant text for each.

Usage

```
rt_data_code_pmc_list(filenames, remove_ns = TRUE, specificity = NULL)
```

Arguments

`filenames`	A list of the PMC XML filenames as strings.
`remove_ns`	Ignored since version 1.2.0 and kept for backward compatibility. Default XML namespaces are now always removed, so a namespaced PMC XML file gives the same result as a plain one.
`specificity`	Retained for backward compatibility; see `rt_data_code_pmc`.

Value

A dataframe of results, one row per file.

Examples

```
# Paths to PMC XML files
filepath <- system.file(
  "extdata", "PMID32171256-PMC7071725.xml", package = "rtransparency"
)
filepaths <- list(filepath)

# Identify and extract indicators of data and code sharing
results_table <- rt_data_code_pmc_list(filepaths)
```

`rt_demo` *Simulated transparency indicators for a corpus of articles*

Description

A small, simulated set of detector output, with one row per article, used to illustrate [rt_summary()], [rt_score()] and [rt_plot()]. The values are **simulated**, not real detector output: prevalences and their trends over time are chosen to resemble published findings (frequent conflict-of- interest and funding disclosure, less frequent protocol registration, low but rising data sharing, rare code sharing, and a recent, fast-rising disclosure of generative-AI use) so the illustrations are realistic.

Usage

```
rt_demo
```

Format

A tibble with 1200 rows and 13 columns:

pmid A made-up PubMed identifier (character).

year Publication year, 2010-2026.

type Article type (research-article, review-article, systematic-review).

is_coi_pred Conflict-of-interest statement detected.

is_fund_pred Funding statement detected.

is_register_pred Protocol registration detected.

is_open_data Data sharing detected.

is_open_code Code sharing detected.

is_novelty_pred Novelty claim detected.

is_replication_pred Replication component detected.

is_ai_pred Disclosure of generative-AI use detected. 'NA' before 2023, when the practice did not yet exist (see [rt_ai_pmc()]).

is_open_access Open license detected.

is_reporting_pred Reporting-guideline use detected.

See Also

[rt_summary()], [rt_score()], [rt_plot()]

rt_ethics — *Identify ethics approval and informed consent statements in text (experimental)*

Description

The plain-text counterpart of [rt_ethics_pmc()], with the same rules and the same caveat: the detector is experimental and not yet validated.

Usage

```
rt_ethics(filename = NULL, text = NULL)
```

Arguments

filename	The path to a TXT file as a string.
text	Alternatively, the article text itself as a character vector (for example the output of [rt_read_pdf()]). Supply ‘filename‘ or ‘text‘.

Value

A tibble with the file name (‘article‘), the PMID (‘NA‘ if absent), ‘is_ethics_pred‘, ‘ethics_text‘, ‘ethics_approval_id‘, ‘is_consent_pred‘ and ‘consent_text‘.

See Also

[rt_ethics_pmc()]

Examples

```
rt_ethics(text = c(
  "The study was approved by the Ethics Committee of X (approval no. 2021-045).",
  "Written informed consent was obtained from all participants."
))
```

rt_ethics_pmc — *Identify ethics approval and informed consent statements (experimental)*

Description

Detects whether an article reports oversight by an ethics body (approval, waiver or exemption by an ethics committee or institutional review board, including a statement that approval was not required) and whether it reports how participant informed consent was handled (obtained, waived or not required). The committee’s approval number is extracted when stated. A bare "Not applicable" does not count.

Usage

```
rt_ethics_pmc(filename, remove_ns = TRUE)
```

Arguments

filename	The name of the PMC XML as a string.
remove_ns	Ignored since version 1.2.0 and kept for backward compatibility. Default XML namespaces are now always removed, so a namespaced PMC XML file gives the same result as a plain one.

Details

Experimental. Unlike the ten main indicators, this detector has not been validated against hand labels, so it is not part of [rt_all_pmc()], has no row in [rt_accuracy], and its output should be spot-checked before use. The scripts in 'data-raw/validation/' build blind labeling sheets for its validation.

Value

A one-row tibble with the article IDs, 'is_ethics_pred', 'ethics_text', 'ethics_approval_id', 'is_consent_pred', 'consent_text' and 'is_success'.

See Also

[rt_ethics()] for plain text.

Examples

```
filepath <- system.file(
  "extdata", "PMID32171256-PMC7071725.xml", package = "rtransparency"
)
rt_ethics_pmc(filepath)
```

---

rt_fetch_pmc *Download PubMed Central full-text XML*

Description

Downloads the full-text JATS XML of articles in PubMed Central, ready for [rt_all_pmc()] or [rt_all_pmc_dir()]. Identifiers can be PMCIDs, PubMed IDs or DOIs (mixed freely); PubMed IDs and DOIs are first converted to PMCIDs with [rt_convert_ids()]. Each article is saved as '<PMCID>.xml' in 'dir'.

Usage

```
rt_fetch_pmc(
  ids,
  dir = ".",
  overwrite = FALSE,
  source = c("ncbi", "europepmc"),
  api_key = Sys.getenv("ENTREZ_KEY"),
  progress = TRUE
)
```

Arguments

ids	A character vector of PMCIDs ("PMC7071725"), PubMed IDs ("32171256") or DOIs ("10.1186/s12874-020-0914-6").
dir	The directory to save the XML files in (created if needed).
overwrite	Whether to download again files that already exist.
source	Where to download from: '"ncbi"' (NCBI PMC, the default, with the OAI-PMH service as fallback) or '"europepmc"' (the Europe PMC REST API). The detectors give the same results on both in a comparison of 13 benchmark articles ('inst/benchmark/results_europepmc_parity.md'); Europe PMC omits some license URLs, so 'oa_license' is then read from the license text.
api_key	An NCBI API key; defaults to the 'ENTREZ_KEY' environment variable.
progress	Whether to show a progress bar.

Details

Full text comes from NCBI E-utilities (EFetch), with the PMC OAI-PMH service as a fallback. Only articles whose publisher allows XML download (the PMC open-access and author-manuscript collections) have a full text; for others NCBI returns the front matter only, which is saved but flagged with 'has_body = FALSE', because most indicators need the article body.

Existing non-empty files are reused, so an interrupted download can simply be re-run. Requests are paced to NCBI's limit of 3 per second, or 10 per second with an API key (set the 'ENTREZ_KEY' environment variable or pass 'api_key').

Value

A tibble with one row per identifier: the input 'id', its 'pmcid', the saved 'file' ('NA' on failure), 'is_success', 'has_body' (whether the XML contains the article body) and the 'error' message on failure.

See Also

[rt_convert_ids()], [rt_all_pmc_dir()]

Examples

```
# Needs internet access; failures are reported per identifier.
dir <- file.path(tempdir(), "pmc")
got <- rt_fetch_pmc("PMC7071725", dir, progress = FALSE)
got
```

`rt_fill_coi_pubmed` *Fill missed conflict-of-interest disclosures from PubMed*

Description

For rows of detector output where no conflict-of-interest statement was found (‘is_coi_pred‘ is ‘FALSE‘) but a PubMed ID is known, looks the article up with [rt_coi_pubmed()] and, when PubMed records a statement, sets ‘is_coi_pred‘ to ‘TRUE‘ and ‘coi_text‘ to that statement. A new ‘coi_source‘ column records where each disclosure came from.

Usage

```
rt_fill_coi_pubmed(data, api_key = Sys.getenv("ENTREZ_KEY"))
```

Arguments

`data`	A data frame with ‘pmid‘, ‘is_coi_pred‘ and ‘coi_text‘ columns, such as the output of [rt_all_pmc()] or [rt_coi_pmc()].
`api_key`	An NCBI API key; defaults to the ‘ENTREZ_KEY‘ environment variable.

Details

Note that the accuracy estimates in [rt_accuracy] describe the full-text detector alone; with this fallback the sensitivity is higher.

Value

‘data‘ as a tibble with ‘is_coi_pred‘ and ‘coi_text‘ filled where PubMed has a statement, and ‘coi_source‘: ‘"article"‘ (found in the full text), ‘"pubmed"‘ (filled from PubMed) or ‘NA‘ (none found).

See Also

[rt_coi_pubmed()]

Examples

```
# Needs internet access.
res <- rt_all_pmc(system.file(
  "extdata", "PMID32171256-PMC7071725.xml", package = "rtransparency"
))
res <- try(rt_fill_coi_pubmed(res))
```

`rt_fund` *Identify and extract Funding statements in TXT files.*

Description

Takes a TXT file and returns data related to the presence of a Funding statement, including whether a Funding statement exists. If a Funding statement exists, it extracts it.

Usage

```
rt_fund(filename = NULL, text = NULL)
```

Arguments

`filename`	The path to a TXT file as a string.
`text`	Alternatively, the article text itself as a character vector (for example the output of [rt_read_pdf()]). Supply 'filename' or 'text'.

Value

A tibble with the file name ('article'), the PMID ('NA' if absent), whether a statement that funding was received was found ('is_fund_pred') and the statement ('fund_text'). These are the same column names as [rt_fund_pmc()] and [rt_all_pmc()]. The former names 'is_funded_pred' and 'funding_text' are still returned, as deprecated copies, and will be removed in a future release.

Examples

```
# Write a short example article to a temporary text file.
filepath <- file.path(tempdir(), "PMID00000000-PMC0000000.txt")
writeLines(c(
  "To our knowledge, this is the first study of its kind.",
  "Conflicts of interest: none declared.",
  "This work was supported by the National Institutes of Health (R01-000000).",
  "The protocol was registered at ClinicalTrials.gov (NCT00000000).",
  "All data and code are available at https://github.com/example/repo.",
  "We independently replicated the original analysis."
), filepath)

# Identify and extract the funding statement.
results_table <- rt_fund(filepath)
```

rt_fund_pmc *Identify and extract Funding statements in PMC XML files.*

Description

Takes a PMC XML file and returns data related to the presence of a Funding statement, including whether a Funding statement exists. If a Funding statement exists, it extracts it.

Usage

```
rt_fund_pmc(filename, remove_ns = TRUE)
```

Arguments

filename	The name of the PMC XML as a string.
remove_ns	Ignored since version 1.2.0 and kept for backward compatibility. Default XML namespaces are now always removed, so a namespaced PMC XML file gives the same result as a plain one.

Value

A dataframe of results. It returns all unique article identifiers, whether this article was deemed relevant to funding (e.g. was the word "fund" found within the text), whether a funding statement was found, whether a statement within the PMC tags dedicated to funding was found, the text identified, whether this text is explicit (i.e. whether it clearly indicated that funding was received) and whether each of the labeling functions identified the text or not. The functions are returned to add flexibility in how this package is used; for example, future definitions of Funding may differ from the one we used.

Examples

```
# Path to a bundled example PMC XML file.
filepath <- system.file(
  "extdata", "PMID32171256-PMC7071725.xml", package = "rtransparency"
)

# Identify and extract meta-data and indicators of transparency.
results_table <- rt_fund_pmc(filepath)
```

`rt_funders_pmc` *Funders, funder identifiers and award numbers from a PMC XML file.*

Description

Reads the structured funding metadata of an article (the JATS '<funding-group>'): each funding source with its name, its Crossref Open Funder Registry DOI and ROR identifier when the publisher tagged them, and the award (grant) numbers of its award group. This complements [rt_fund_pmc()], which detects whether a funding statement exists, with identifiers that can be linked to funder databases.

Usage

```
rt_funders_pmc(filename, remove_ns = TRUE)
```

Arguments

`filename`	The name of the PMC XML as a string.
`remove_ns`	Ignored since version 1.2.0 and kept for backward compatibility. Default XML namespaces are now always removed, so a namespaced PMC XML file gives the same result as a plain one.

Value

A tibble with one row per funding source: the article IDs, 'funder' (the name as tagged), 'funder_doi' (a Crossref Funder Registry DOI such as '10.13039/100000002'), 'funder_ror' (a ROR URL), 'award_id' (the award numbers of the source's award group, '"; "'-separated) and 'is_success'. An article without a '<funding-group>' gives one row with 'NA' funder fields, so every file is represented.

See Also

[rt_fund_pmc()], [rt_authors_pmc()]

Examples

```
filepath <- system.file(
  "extdata", "PMID32171256-PMC7071725.xml", package = "rtransparency"
)
rt_funders_pmc(filepath)
```

rt_meta_pmc *Extract article metadata from a PMC XML file.*

Description

Reads a PMC XML file and returns its metadata as a one-row data frame: journal, publisher, article title, authors and affiliations, identifiers (PMID, PMCID, DOI), publication dates, and figure / table / reference counts.

Usage

```
rt_meta_pmc(filename, remove_ns = TRUE)
```

Arguments

filename	The path to the PMC XML file as a string.
remove_ns	Ignored since version 1.2.0 and kept for backward compatibility. Default XML namespaces are now always removed, so a namespaced PMC XML file gives the same result as a plain one.

Value

A one-row tibble of metadata. The column ‘is_success‘ indicates whether the file was parsed successfully.

Examples

```
filepath <- system.file(
  "extdata", "PMID32171256-PMC7071725.xml", package = "rtransparency"
)
rt_meta_pmc(filepath)
```

rt_novelty *Identify whether a study claims novelty in TXT files.*

Description

Takes a TXT file and returns data related to the presence of novelty claims, including whether a novelty claim exists. If a novelty claim exists, it extracts the relevant text. Novelty is defined as the study claiming to report something "for the first time."

Usage

```
rt_novelty(filename = NULL, text = NULL)
```

Arguments

filename	The path to a TXT file as a string.
text	Alternatively, the article text itself as a character vector (for example the output of [rt_read_pdf()]). Supply ‘filename‘ or ‘text‘.

Value

A tibble of results. It returns the file name (‘article‘), the PMID (‘NA‘ if absent), whether a novelty claim was found, the text identified, and whether each pattern-matching function identified relevant text or not.

Examples

```
# Write a short example article to a temporary text file.
filepath <- file.path(tempdir(), "PMID00000000-PMC0000000.txt")
writeLines(c(
  "To our knowledge, this is the first study of its kind.",
  "Conflicts of interest: none declared.",
  "This work was supported by the National Institutes of Health (R01-000000).",
  "The protocol was registered at ClinicalTrials.gov (NCT00000000).",
  "All data and code are available at https://github.com/example/repo.",
  "We independently replicated the original analysis."
), filepath)

# Identify and extract novelty claims.
results_table <- rt_novelty(filepath)
```

---

rt_novelty_pmc *Identify and extract novelty claims in PMC XML files.*

Description

Takes a PMC XML file and returns data related to the presence of novelty claims, including whether such claims exist and the relevant text. Novelty is defined as the study claiming to report something "for the first time."

Usage

```
rt_novelty_pmc(filename, remove_ns = TRUE)
```

Arguments

filename	The name of the PMC XML as a string.
remove_ns	Ignored since version 1.2.0 and kept for backward compatibility. Default XML namespaces are now always removed, so a namespaced PMC XML file gives the same result as a plain one.

Value

A tibble of results. It returns the unique identifiers of the article, whether a novelty claim was found, the relevant text and whether each pattern-matching function identified relevant text.

Examples

```
# Path to a bundled example PMC XML file.
filepath <- system.file(
  "extdata", "PMID32171256-PMC7071725.xml", package = "rtransparency"
)

# Identify and extract novelty claims.
results_table <- rt_novelty_pmc(filepath)
```

`rt_oa` *Identify the open-access status and reuse license from a TXT file.*

Description

The plain-text counterpart of [rt_oa_pmc()]. It detects an open-access declaration and a Creative Commons license from the article text (for example a "This is an open access article distributed under the terms of the Creative Commons Attribution License" statement). Plain text lacks the structured JATS ‘<license>‘ element, so detection relies on the prose and any license URL it contains.

Usage

```
rt_oa(filename = NULL, text = NULL)
```

Arguments

`filename`	The path to a TXT file as a string.
`text`	Alternatively, the article text itself as a character vector (for example the output of [rt_read_pdf()]). Supply ‘filename‘ or ‘text‘.

Value

A tibble with the file name (‘article‘), the PMID (‘NA‘ if absent), whether the article is openly licensed (‘is_open_access‘), the canonical license (‘oa_license‘) and the license statement (‘oa_text‘).

See Also

[rt_oa_pmc()] for the PMC XML detector.

Examples

```
# Write a short example article to a temporary text file.
filepath <- file.path(tempdir(), "PMID00000000-PMC0000000.txt")
writeLines(
  paste(
    "This is an open access article distributed under the terms of the",
    "Creative Commons Attribution License (CC BY 4.0)."
  ),
  filepath
)
rt_oa(filepath)
```

`rt_oa_pmc` *Identify the open-access status and reuse license of a PMC XML file.*

Description

Detects whether an article is openly licensed and, when it is, the canonical license identifier (for example 'CC-BY-4.0', 'CC-BY-NC-4.0', 'CC0-1.0'). The license is read from the JATS '<permissions>'/'<license>' element and its license reference URL. This is the article-level reuse signal (the "R" in FAIR): a permissive license (CC BY, CC0) allows redistribution and text and data mining, whereas a restrictive license (NC / ND) or retained copyright does not.

Usage

```
rt_oa_pmc(filename, remove_ns = TRUE)
```

Arguments

`filename`	The filename of the PMC XML file to analyze.
`remove_ns`	Ignored since version 1.2.0 and kept for backward compatibility. Default XML namespaces are now always removed, so a namespaced PMC XML file gives the same result as a plain one.

Value

A tibble with the article IDs, whether the article is openly licensed ('is_open_access'), the canonical license ('oa_license', '""' when none is found), the license statement ('oa_text') and 'is_success'.

Examples

```
filepath <- system.file(
  "extdata", "PMID32171256-PMC7071725.xml", package = "rtransparency"
)
rt_oa_pmc(filepath)
```

`rt_plot` *Plot transparency indicators*

Description

Produces a ‘ggplot‘ of either the prevalence of each indicator (a bar chart) or the prevalence over time (a line chart). Requires the ‘ggplot2‘ package.

Usage

```
rt_plot(
  x,
  type = c("prevalence", "trend"),
  indicators = NULL,
  by = NULL,
  year = NULL,
  adjusted = FALSE,
  accuracy = NULL,
  conf_level = 0.95
)
```

Arguments

`x`	Either a data frame with one row per article (it is summarized with [rt_summary()]) or an existing [rt_summary()] result.
`type`	‘"prevalence"‘ for a bar chart of each indicator’s prevalence (the default), or ‘"trend"‘ for prevalence over time (requires ‘year‘).
`indicators, by`	Passed to [rt_summary()] when ‘x‘ is article-level data. ‘by‘ adds facets to the ‘"prevalence"‘ plot.
`year`	For ‘type = "trend"‘, the name of the column in ‘x‘ holding the (numeric) publication year.
`adjusted`	If ‘TRUE‘, plot the sensitivity/specificity-corrected prevalence instead of the apparent prevalence. Defaults to ‘FALSE‘.
`accuracy, conf_level`	Passed to [rt_summary()].

Value

A ‘ggplot‘ object.

See Also

[rt_summary()]

Examples

```
data(rt_demo)

if (requireNamespace("ggplot2", quietly = TRUE)) {
  rt_plot(rt_demo)                          # prevalence bar chart
  rt_plot(rt_demo, type = "trend", year = "year")
}
```

`rt_read_pdf` *Convert a PDF file to text.*

Description

Takes a path to a PDF file and returns its text content as a single character string, extracted by default with the poppler ‘pdftotext‘ utility (the same extractor the original ‘oddpub‘ package relied on, called as a system command). Different extractors format text differently; the detectors were tuned to the reading-order layout ‘pdftotext‘ produces. The result can be passed straight to the plain-text detectors through their ‘text‘ argument, or scored in one call with [rt_all_pdf()].

Usage

```
rt_read_pdf(filepath, engine = c("pdftotext", "pdftools"))
```

Arguments

`filepath`	The path to the PDF file as a string (must end in ‘.pdf‘).
`engine`	‘"pdftotext"‘ (default) calls the poppler command-line utility, which must be on the PATH. ‘"pdftools"‘ uses the **pdftools** package instead, which bundles poppler (convenient on Windows) but keeps the physical page layout, so text in two-column articles can be interleaved and some statements missed; prefer ‘"pdftotext"‘ when it is available.

Value

A character string with the extracted text, transliterated to ASCII.

Examples

```
## Not run:
# Path to a PDF file.
pdf_path <- system.file(
  "extdata", "PMID32171256-PMC7071725.pdf", package = "rtransparency"
)

# Extract the text and run a detector on it, or score all indicators at once.
article_txt <- rt_read_pdf(pdf_path)
rt_coi(text = article_txt)
```

```
rt_all_pdf(pdf_path)

## End(Not run)
```

`rt_register` *Identify and extract Registration statements in TXT files.*

Description

Takes a TXT file and returns data related to the presence of a Registration statement, including whether a Registration statement exists. If a Registration statement exists, it extracts it.

Usage

```
rt_register(filename = NULL, text = NULL)
```

Arguments

`filename`	The path to a TXT file as a string.
`text`	Alternatively, the article text itself as a character vector (for example the output of [rt_read_pdf()]). Supply 'filename' or 'text'.

Value

A tibble. It returns the file name ('article'), the PMID ('NA' if absent), whether a registration statement was found, the identified statement, whether the text was deemed relevant (e.g. contained the word registration), whether a Methods section was identified, whether an NCT number was identified, whether a registration was explicitly identified (defunct) and whether each labeling function identified a relevant text or not. The labeling functions are returned to add flexibility in how this package is used; for example, future definitions of Registration may differ from the one we used.

Examples

```
# Write a short example article to a temporary text file.
filepath <- file.path(tempdir(), "PMID00000000-PMC0000000.txt")
writeLines(c(
  "To our knowledge, this is the first study of its kind.",
  "Conflicts of interest: none declared.",
  "This work was supported by the National Institutes of Health (R01-000000).",
  "The protocol was registered at ClinicalTrials.gov (NCT00000000).",
  "All data and code are available at https://github.com/example/repo.",
  "We independently replicated the original analysis."
), filepath)

# Identify and extract the registration statement.
results_table <- rt_register(filepath)
```

`rt_register_pmc` *Identify and extract Conflicts of Interest statements in PMC XML files.*

Description

Takes a PMC XML file and returns data related to the presence of a Funding statement, including whether a Funding statement exists. If a Funding statement exists, it extracts it.

Usage

```
rt_register_pmc(filename, remove_ns = TRUE)
```

Arguments

`filename`	The name of the PMC XML as a string.
`remove_ns`	Ignored since version 1.2.0 and kept for backward compatibility. Default XML namespaces are now always removed, so a namespaced PMC XML file gives the same result as a plain one.

Value

A dataframe of results. It returns the unique article identifiers, whether this article was deemed a research, review or systematic review, whether the text was deemed relevant to registration (e.g. contained the word registration), whether a Methods section was identified, whether an NCT number was identified, whether a registration was explicitly identified (defunct), whether a registration statement was found, what the registration statement was, whether it the registration was identified from the PMC XML (i.e. it was found within a dedicated registration tag) and whether each labeling function identified a relevant text or not. The labeling functions are returned to add flexibility in how this package is used; for example, future definitions of Registration may differ from the one we used.

Examples

```
# Path to a bundled example PMC XML file.
filepath <- system.file(
  "extdata", "PMID32171256-PMC7071725.xml", package = "rtransparency"
)

# Identify and extract meta-data and indicators of transparency.
results_table <- rt_register_pmc(filepath)
```

`rt_registration_timing` *Check whether ClinicalTrials.gov registrations were prospective*

Description

Looks up trials in the ClinicalTrials.gov registry (API version 2) and compares the date the registration was first submitted with the study start date. A registration is prospective when it was submitted no later than the start date (plus an optional grace period).

Usage

```
rt_registration_timing(nct_ids, grace_days = 0)
```

Arguments

`nct_ids`	A character vector of NCT numbers, for example from [rt_trial_ids()]. Other identifiers are returned with 'NA' dates.
`grace_days`	Days after the start date within which a registration still counts as prospective (default '0'). Some studies allow 30 days.

Details

When the registry gives the start date to the month only, a registration submitted within that month cannot be classified and 'is_prospective' is 'NA'; one submitted before the month is prospective and one after it is retrospective. The start date the registry holds may be an estimate for trials not yet started ('start_date_type').

Value

A tibble with one row per unique identifier: 'nct_id', 'first_submitted' and 'first_posted' (the registration dates), 'start_date', 'start_date_precision' ('"day"' or '"month"'), 'start_date_type' ('"ACTUAL"' or '"ESTIMATED"'), 'days_after_start' (submission date minus start date; negative when registered before the start) and 'is_prospective'. Trials the registry does not know have 'NA' dates.

See Also

[rt_trial_ids()], [rt_register_pmc()]

Examples

```
# Needs internet access and the jsonlite package.
if (requireNamespace("jsonlite", quietly = TRUE)) {
  try(rt_registration_timing(c("NCT04368728", "NCT00000102")))
}
```

`rt_replication` *Identify whether a study includes a replication component in TXT files.*

Description

Takes a TXT file and returns data related to the presence of a replication or validation component, including whether such a component exists. Replication is defined as the study independently confirming findings from a prior study in a new sample.

Usage

```
rt_replication(filename = NULL, text = NULL)
```

Arguments

`filename`	The path to a TXT file as a string.
`text`	Alternatively, the article text itself as a character vector (for example the output of [rt_read_pdf()]). Supply 'filename' or 'text'.

Value

A tibble of results. It returns the file name ('article'), the PMID ('NA' if absent), whether a replication component was found, the text identified, and whether each pattern-matching function identified relevant text or not.

Examples

```
# Write a short example article to a temporary text file.
filepath <- file.path(tempdir(), "PMID00000000-PMC0000000.txt")
writeLines(c(
  "To our knowledge, this is the first study of its kind.",
  "Conflicts of interest: none declared.",
  "This work was supported by the National Institutes of Health (R01-000000).",
  "The protocol was registered at ClinicalTrials.gov (NCT00000000).",
  "All data and code are available at https://github.com/example/repo.",
  "We independently replicated the original analysis."
), filepath)

# Identify and extract replication components.
results_table <- rt_replication(filepath)
```

rt_replication_pmc *Identify and extract replication components in PMC XML files.*

Description

Takes a PMC XML file and returns data related to the presence of a replication or validation component, including whether such a component exists and the relevant text. Replication is defined as the study independently confirming findings from a prior study in a new sample.

Usage

```
rt_replication_pmc(filename, remove_ns = TRUE)
```

Arguments

filename	The name of the PMC XML as a string.
remove_ns	Ignored since version 1.2.0 and kept for backward compatibility. Default XML namespaces are now always removed, so a namespaced PMC XML file gives the same result as a plain one.

Value

A tibble of results. It returns the unique identifiers of the article, whether a replication component was found, the relevant text and whether each pattern-matching function identified relevant text.

Examples

```
# Path to a bundled example PMC XML file.
filepath <- system.file(
  "extdata", "PMID32171256-PMC7071725.xml", package = "rtransparency"
)

# Identify and extract replication components.
results_table <- rt_replication_pmc(filepath)
```

rt_reporting *Identify use of a reporting guideline from a TXT file.*

Description

The plain-text counterpart of [rt_reporting_pmc()]. Detects whether an article states that it followed a reporting guideline and which one, using the same precision-first rules.

Usage

```
rt_reporting(filename = NULL, text = NULL)
```

Arguments

filename	The path to a TXT file as a string.
text	Alternatively, the article text itself as a character vector (for example the output of [rt_read_pdf()]). Supply ‘filename‘ or ‘text‘.

Value

A tibble with the file name (‘article‘), the PMID (‘NA‘ if absent), whether a reporting-guideline statement was found (‘is_reporting_pred‘), the guideline(s) named (‘reporting_guideline‘) and the matched statement (‘reporting_text‘).

See Also

[rt_reporting_pmc()] for the PMC XML detector.

Examples

```
# Write a short example article to a temporary text file.
filepath <- file.path(tempdir(), "PMID00000000-PMC0000000.txt")
writeLines(
  "This systematic review was reported following the PRISMA 2020 guideline.",
  filepath
)
rt_reporting(filepath)
```

---

rt_reporting_pmc *Identify use of a reporting guideline from a PMC XML file.*

Description

Detects whether an article states that it followed a reporting guideline (the EQUATOR-network checklists such as CONSORT, PRISMA, STROBE, ARRIVE, STARD, TRIPOD, COREQ, SQUIRE, CHEERS) and which one. Detection is precision-first: a guideline acronym is counted only when it appears in a reporting context (a reporting or adherence verb, or a guideline noun such as "statement", "checklist" or "guideline"), so a bare citation does not count.

Usage

```
rt_reporting_pmc(filename, remove_ns = TRUE)
```

Arguments

filename	The filename of the PMC XML file to analyze.
remove_ns	Ignored since version 1.2.0 and kept for backward compatibility. Default XML namespaces are now always removed, so a namespaced PMC XML file gives the same result as a plain one.

Value

A tibble with the article IDs, whether a reporting-guideline statement was found (‘is_reporting_pred‘), the guideline(s) named (‘reporting_guideline‘), the matched statement (‘reporting_text‘) and ‘is_success‘.

Examples

```
filepath <- system.file(
  "extdata", "PMID32171256-PMC7071725.xml", package = "rtransparency"
)
rt_reporting_pmc(filepath)
```

---

rt_score *Count the transparency indicators met by each article*

Description

Adds a column giving, for each article (row), how many of the transparency indicators were detected. This is the per-article transparency score used to describe how many practices an article adheres to.

Usage

```
rt_score(data, indicators = NULL, name = "n_indicators")
```

Arguments

data	A data frame with one row per article and indicator columns named as in [rt_all_pmc()].
indicators	Optional character vector of indicator columns to count. Defaults to the five openness practices present in ‘data‘ (conflicts of interest, funding, registration, data and code); novelty and replication are excluded unless requested explicitly, as they are not adherence practices.
name	Name of the count column to add (default ‘"n_indicators"‘).

Value

‘data‘ as a tibble with the integer count column added. Rows with no assessed indicators receive ‘NA‘ for the count. Tabulate it (for example with [table()] or ‘dplyr::count()‘) for the distribution of the number of practices met.

See Also

[rt_summary()]

Examples

```
data(rt_demo)
scored <- rt_score(rt_demo)
table(scored$n_indicators)
```

`rt_summary` *Summarize transparency indicators across a corpus of articles*

Description

Takes a data frame with one row per article (such as the output of [rt_all_pmc()] joined with [rt_data_code_pmc()], stacked over many articles) and returns the prevalence of each transparency indicator. For each indicator it reports the number of articles assessed, the number in which the indicator was detected, the apparent prevalence and its Wilson confidence interval and, optionally, a prevalence corrected for the detector's sensitivity and specificity (the Rogan-Gladen estimator).

Usage

```
rt_summary(
  data,
  indicators = NULL,
  by = NULL,
  adjust = TRUE,
  accuracy = NULL,
  conf_level = 0.95,
  adj_interval = c("simulation", "fixed"),
  n_sim = 10000,
  seed = 2021,
  register_assessed_only = TRUE
)
```

Arguments

`data`	A data frame with one row per article. Indicator columns must be logical or numeric 0/1 and named as in [rt_all_pmc()]: 'is_coi_pred', 'is_fund_pred', 'is_register_pred', 'is_open_data', 'is_open_code', 'is_novelty_pred', 'is_replication_pred', 'is_ai_pred', 'is_open_access' and 'is_reporting_pred'. 'NA' marks an article that was not assessed for that indicator (for example 'is_ai_pred' before 2023) and is excluded from its denominator. Other values are rejected rather than silently coerced.
`indicators`	Optional character vector of indicator columns to summarize. Defaults to every recognized indicator present in 'data'.
`by`	Optional name of a grouping column (for example a publication year, journal or article type); the summary is then computed within each group.
`adjust`	If 'TRUE' (default), add a prevalence corrected for detector sensitivity and specificity using 'accuracy'. Indicators absent from 'accuracy' receive 'NA' corrected values.
`accuracy`	A data frame of detector accuracy with columns 'variable', 'sensitivity' and 'specificity', and optionally the validation counts 'tp', 'fn', 'tn' and 'fp'. Defaults to [rt_accuracy].

`conf_level` Confidence level for the intervals (default ‘0.95‘).

`adj_interval` How the interval of the corrected prevalence is computed. ‘"simulation"‘ (the default) propagates the uncertainty of the detector’s sensitivity and specificity, estimated from the validation counts in ‘accuracy‘, together with that of the apparent prevalence (see Details). ‘"fixed"‘ treats sensitivity and specificity as known and corrects the bounds of the apparent-prevalence interval, which is too narrow when the validation sample is small. Indicators without validation counts always use ‘"fixed"‘.

`n_sim` Number of simulation draws for ‘adj_interval = "simulation"‘.

`seed` Random seed for the simulation, so results are reproducible. The caller’s random number stream is left untouched.

`register_assessed_only`
If ‘TRUE‘ (default) and ‘data‘ has an ‘is_research‘ column (as [rt_all_pmc()] output does), protocol registration is summarized over the articles the registration detector assesses: research articles and, when ‘is_review‘ is present, reviews (where PROSPERO registration applies). Editorials, letters, news and similar types are not assessed and return ‘FALSE‘, so counting them in the denominator understates registration. ‘FALSE‘ counts every article.

Details

Corrected prevalence. The Rogan-Gladen estimator corrects an apparent prevalence ‘p‘ for detector error: ‘(p + specificity - 1) / (sensitivity + specificity - 1)‘, truncated to ‘[0, 1]‘. With ‘adj_interval = "simulation"‘, the apparent prevalence, sensitivity and specificity are drawn from their Jeffreys posteriors (‘Beta(x + 0.5, n - x + 0.5)‘) using the corpus counts and the validation counts, each draw is corrected, and the percentile interval of the draws is reported. Because the correction divides by ‘sensitivity + specificity - 1‘ and subtracts ‘1 - specificity‘, uncertainty in specificity dominates for rare indicators (registration, code sharing, replication), which the fixed interval ignores.

Rows whose grouping value is ‘NA‘ form their own group, labelled ‘NA‘.

Value

A tibble with one row per indicator (per group, if ‘by‘ is given): the grouping column (when ‘by‘ is used), ‘indicator‘, ‘label‘, ‘n_articles‘, ‘n_detected‘, ‘percent‘, ‘conf_low‘, ‘conf_high‘ and, when ‘adjust = TRUE‘, ‘adj_percent‘, ‘adj_low‘ and ‘adj_high‘. Percentages and interval bounds are on the 0-100 scale.

See Also

[rt_score()], [rt_plot()], [rt_accuracy]

Examples

```
data(rt_demo)
rt_summary(rt_demo)

# Apparent prevalence only, no accuracy correction
rt_summary(rt_demo, adjust = FALSE)
```

```
# By article type
rt_summary(rt_demo, by = "type")
```

`rt_trial_ids` *Extract trial and review registration identifiers from text*

Description

Finds registry identifiers (ClinicalTrials.gov NCT numbers, ISRCTN, PROSPERO, ChiCTR, DRKS, ANZCTR, IRCT, UMIN, jRCT, CTRI, PACTR, KCT, EudraCT, CTIS and INPLASY) in text, typically the ‘register_text‘ returned by [rt_all_pmc()] or [rt_register()].

Usage

```
rt_trial_ids(text)
```

Arguments

`text`	A character vector.

Value

A tibble with one row per identifier found: the position in ‘text‘ (‘element‘), the ‘registry‘ and the ‘trial_id‘ (upper-cased, with spaces removed). Identifiers repeated within an element are listed once.

See Also

[rt_registration_timing()]

Examples

```
rt_trial_ids(c(
  "Registered at ClinicalTrials.gov (NCT04368728) and ISRCTN12345678.",
  "PROSPERO CRD42020123456",
  "No registration."
))
```

Index

∗ **datasets**
    rt_accuracy, 3
    rt_accuracy_2021, 5
    rt_demo, 21

rt_accuracy, 3
rt_accuracy_2021, 5
rt_ai, 5
rt_ai_pmc, 6
rt_all, 7
rt_all_pdf, 8
rt_all_pmc, 9
rt_all_pmc_dir, 10
rt_all_txt_dir, 12
rt_authors_pmc, 13
rt_check_links, 14
rt_coi, 15
rt_coi_pmc, 16
rt_coi_pubmed, 16
rt_convert_ids, 17
rt_data_code, 18
rt_data_code_pmc, 19, *20*
rt_data_code_pmc_list, 20
rt_demo, 21
rt_ethics, 22
rt_ethics_pmc, 22
rt_fetch_pmc, 23
rt_fill_coi_pubmed, 25
rt_fund, 26
rt_fund_pmc, 27
rt_funders_pmc, 28
rt_meta_pmc, 29
rt_novelty, 29
rt_novelty_pmc, 30
rt_oa, 31
rt_oa_pmc, 32
rt_plot, 33
rt_read_pdf, 34
rt_register, 35
rt_register_pmc, 36
rt_registration_timing, 37
rt_replication, 38
rt_replication_pmc, 39
rt_reporting, 39
rt_reporting_pmc, 40
rt_score, 41
rt_summary, 42
rt_trial_ids, 44
rtransparency *(*rtransparency-package*)*, 3
rtransparency-package, 3