

Package: scholid (via r-universe)

October 8, 2026

Type Package

Title Scholarly and Academic Identifier Utilities

Version 0.2.1

Language en-US

Description Detects, normalizes, classifies, and extracts scholarly identifier strings, locates them in free text, gives keys for comparing and deduplicating them, and writes them as resolver URLs, CURIEs, or arXiv DOIs. Provides lightweight, dependency-free helpers for identifier types including DOIs, ORCID iDs, ISBNs, ISSNs, arXiv and PubMed identifiers, ROR and ISNI, OpenAlex and ADS bibcodes, RRID, ARK, SWHID, and selected life-science accessions (UniProt, RefSeq, SRA, GEO, BioProject, and genome assemblies). Functions are vectorized, predictable, and suitable as low-level building blocks for other R packages and data workflows. Use 'scholid_types()' for the authoritative type list. For online lookup, conversion, metadata retrieval, and linked identifier discovery, see 'scholidonline'.

License MIT + file LICENSE

URL <https://thomas-rauter.github.io/scholid/>,
<https://thomas-rauter.github.io/scholidonline/>

BugReports <https://github.com/Thomas-Rauter/scholid/issues>

Depends R (>= 3.5.0)

Suggests testthat (>= 3.0.0), knitr (>= 1.30), rmarkdown

Encoding UTF-8

LazyData false

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

VignetteBuilder knitr

Repository <https://thomas-rauter.r-universe.dev>

Date/Publication 2026-09-29 09:37:08 UTC
RemoteUrl https://github.com/thomas-rauter/scholid
RemoteRef HEAD
RemoteSha 76c42f0a424c55484d20b4fb47541e9f26a9f3bc

Contents

scholid-package	2
classify_scholid	3
detect_scholid_type	3
extract_scholid	4
format_scholid	5
is_scholid	6
locate_scholid	8
normalize_scholid	9
scholid_key	10
scholid_types	11
Index	12

scholid-package	<i>Scholarly and Academic Identifier Utilities</i>
-----------------	--

Description

scholid provides lightweight, dependency-free utilities for detecting, normalizing, comparing, formatting, classifying, extracting, and locating scholarly identifier strings. See [scholid_types\(\)](#) for the supported identifier types and their classification order.

Vignettes

- [Getting started](#) introduces the exported functions and typical workflows for mixed identifier data.
- [scholid_definitions](#) (*About identifiers*) documents per-type formats, validation rules, and classification precedence.

Author(s)

Maintainer: Thomas Rauter <rauterthomas0@gmail.com> ([ORCID](#)) [funder]

See Also

[is_scholid\(\)](#), [normalize_scholid\(\)](#), [scholid_key\(\)](#), [format_scholid\(\)](#), [extract_scholid\(\)](#), [locate_scholid\(\)](#), [classify_scholid\(\)](#), [detect_scholid_type\(\)](#), [scholid_types\(\)](#)

classify_scholid	<i>Classify scholarly identifiers</i>
------------------	---------------------------------------

Description

Performs best-guess classification of scholarly identifier strings. For each element of the input, the function returns the first matching identifier type, or `NA_character_` if no supported type matches.

Classification is based on canonical identifier syntax. Types are checked in the order returned by [scholid_types\(\)](#) (most specific first); the first match wins. Wrapped forms (e.g., URLs or labels) should be normalized first with [normalize_scholid\(\)](#).

Usage

```
classify_scholid(x)
```

Arguments

`x` A vector of candidate identifier values.

Value

A character vector of the same length as `x`, giving the detected identifier type for each element, or `NA_character_` if no match is found.

See Also

[detect_scholid_type\(\)](#), [scholid_types\(\)](#), [scholid_definitions](#)

Examples

```
classify_scholid(c("10.1000/182", "0000-0002-1825-0097", "not an id"))
classify_scholid(normalize_scholid("https://doi.org/10.1000/182", "doi"))
```

detect_scholid_type	<i>Detect scholarly identifier types</i>
---------------------	--

Description

Performs best-effort detection of scholarly identifier types from possibly wrapped identifier strings (e.g., URLs or labels).

For each element of the input, the function returns the first matching identifier type, or `NA_character_` if no supported type matches.

Detection first attempts classification based on canonical identifier syntax (see [classify_scholid\(\)](#)). If no match is found, the function attempts per-type normalization (see [normalize_scholid\(\)](#)) and

returns the first type for which normalization yields a non-missing result. PMID is checked last as a fallback when no more specific type matches.

Use [normalize_scholid\(\)](#) to convert detected values to canonical form once the identifier type is known.

Usage

```
detect_scholid_type(x)
```

Arguments

x A vector of candidate identifier values.

Value

A character vector of the same length as **x**, giving the detected identifier type for each element, or `NA_character_` if no match is found.

See Also

[classify_scholid\(\)](#), [normalize_scholid\(\)](#), [scholid_types\(\)](#)

Examples

```
detect_scholid_type(c(
  "https://doi.org/10.1000/182",
  "doi:10.1000/182",
  "https://orcid.org/0000-0002-1825-0097",
  "arXiv:2101.12345v2",
  "PMID: 12345678",
  "PMCID: PMC1234567",
  "not an id"
))
```

extract_scholid

Extract scholarly identifiers from text

Description

Extract identifiers of a single supported type from free text.

The result is a list with one element per input element. Each element is a character vector of matches (possibly length 0). NA inputs yield an empty character vector.

Matches are returned as extracted identifier tokens from the text. Surrounding prose punctuation or markup fragments may be removed where necessary to isolate the identifier. Use [normalize_scholid\(\)](#) to convert identifiers to canonical form.

Before matching, invisible characters are removed from the text and Unicode spaces count as whitespace. For some types, Unicode dashes and full-width digits are read as their ASCII forms, and

the returned tokens use the ASCII forms. See "Input characters" in `vignette("scholid_definitions", package = "scholid")`.

Usage

```
extract_scholid(text, type)
```

Arguments

<code>text</code>	A character vector of text.
<code>type</code>	A single string giving the identifier type. See <code>scholid_types()</code> for supported values.

Value

A list with the same length as `text`, whose elements are character vectors of extracted identifiers.

See Also

`locate_scholid()` to find identifiers of several types at once, with their positions in the text, `normalize_scholid()`, `scholid_types()`

Examples

```
extract_scholid("See https://doi.org/10.1000/182.", "doi")
extract_scholid("ORCID 0000-0002-1825-0097", "orcid")
```

format_scholid

Write scholarly identifiers as resolver URLs, CURIEs, or DOIs

Description

Vectorized writer that gives identifiers in a form for links, reports, and linked data:

- "url": the resolver URL, such as `https://doi.org/10.1000/182`.
- "curie": the CURIE, the type's Bioregistry prefix, a colon, and the identifier, such as `pubmed:12345678`. For types whose canonical form already is a CURIE, it is that form.
- "doi": for arXiv identifiers, the DOI that arXiv registers, such as `10.48550/arXiv.2101.00001`. The DOI names the preprint, not a version, so the version is left out, and so is the subject class of old-style identifiers.

`x` is first normalized with the rules of `normalize_scholid()`, so wrapped input such as URLs and labels works. The output is plain text: identifiers are written into URLs as they are, except for characters that the resolver requires to be percent-encoded, such as `#` in a DOI.

Which types have which forms, and the sources of the forms, are in "Resolver URLs and CURIEs" and each type's section of the *How Scholarly Identifiers Are Defined* vignette (`vignette("scholid_definitions", package = "scholid")`).

What `format_scholid()` writes, `normalize_scholid()` reads back: `normalize_scholid(format_scholid(x, type, as), type)` equals `normalize_scholid(x, type)` for "url" and "curie". For "doi", `normalize_scholid(format_scholid(x, "arxiv", "doi"), "arxiv")` is the arXiv identifier without its version.

Usage

```
format_scholid(x, type, as = c("url", "curie", "doi"))
```

Arguments

<code>x</code>	A vector of values to write.
<code>type</code>	A single string giving the identifier type. See <code>scholid_types()</code> for supported values.
<code>as</code>	A single string giving the form: "url" (the default), "curie", or "doi". Matched with <code>match.arg()</code> , so it can be abbreviated.

Value

A character vector with the same length as `x`. Values that are NA or don't normalize yield `NA_character_`. If `type` has no form `as`, such as a URL for an ISBN, `format_scholid()` stops with an error that names the forms `type` has.

See Also

`normalize_scholid()`, `scholid_key()`, `scholid_types()`

Examples

```
format_scholid("https://doi.org/10.1000/182", "doi")
format_scholid(c("PMID: 12345678", "not a PMID", NA), "pmid")
format_scholid("0000-0002-1825-0097", "orcid", as = "curie")
format_scholid("arXiv:2101.00001v1", "arxiv", as = "doi")

# normalize_scholid() reads the output back
url <- format_scholid("P12345", "uniprot")
normalize_scholid(url, "uniprot")
```

Description

Vectorized predicate that tests whether values are valid scholarly identifiers of a given supported type.

For identifier types that define a checksum, checksum correctness is verified. The same checksum rules apply to [normalize_scholid\(\)](#). Which types define one is in the overview table of the *How Scholarly Identifiers Are Defined* vignette (`vignette("scholid_definitions", package = "scholid")`).

The main difference from normalization is input form: `is_scholid()` expects values in canonical (or near-canonical) form. Wrapped values such as URLs or prefixed labels should be normalized first with [normalize_scholid\(\)](#). Values that contain an invisible character or a Unicode space are not canonical and return FALSE. Unicode dashes and full-width digits are not read as their ASCII forms. See "Input characters" in the same vignette.

Inputs that are NA yield NA. Non-matching values return FALSE.

Usage

```
is_scholid(x, type)
```

Arguments

x	A vector of values to test.
type	A single string giving the identifier type. See scholid_types() for supported values.

Value

A logical vector of the same length as x, indicating whether each element is a valid identifier of the specified type.

See Also

[normalize_scholid\(\)](#), [scholid_types\(\)](#)

Examples

```
is_scholid("10.1000/182", "doi")
is_scholid("0000-0002-1825-0097", "orcid")
```

locate_scholid	<i>Locate scholarly identifiers in text</i>
----------------	---

Description

Find identifiers of several types in free text at once, and report where each one is. For each type, the identifiers found are those that [extract_scholid\(\)](#) returns.

The result is a data frame with one row per identifier found and these columns:

- `element`: the index of the element of text it was found in.
- `type`: its identifier type.
- `id`: the token [extract_scholid\(\)](#) returns for it.
- `match`: the identifier as written in the text. It leaves out a URL or label in front and the punctuation after it, but keeps a prefix of the canonical form, such as `RRID:` or `ark:`. It can differ from `id` in case, in spaces between digit groups, and in look-alike and invisible characters.
- `start`, `end`: the positions of `match` in `text[element]`, counted as `substr()` counts them, so that `substr(text[element], start, end) == match`. They include the invisible characters that matching ignores.

Rows are sorted by `element`, then `start`. NA elements give no rows. When nothing is found, the result has zero rows and the same columns.

The same stretch of text can be found by several types, such as an ISBN and the PMID-like digits inside it. Within one element, when the spans of two hits overlap, only one is kept: the longer span wins, and for equal lengths the type that comes first in [scholid_types\(\)](#), the order [classify_scholid\(\)](#) uses. Hits are taken in that order, and each is kept unless it overlaps one already kept. Only the types in `types` take part, so a type left out can't hide another.

Usage

```
locate_scholid(text, types = scholid_types())
```

Arguments

<code>text</code>	A character vector of text.
<code>types</code>	A character vector of identifier types to look for. See scholid_types() for supported values. Duplicates and order don't matter.

Value

A data frame with one row per identifier found and the columns `element` (integer), `type`, `id`, `match` (character), `start`, and `end` (integer).

See Also

[extract_scholid\(\)](#) to extract identifiers of one type, [scholid_types\(\)](#)

Examples

```
locate_scholid(c(
  "See doi:10.1000/182 and ORCID 0000-0002-1825-0097.",
  NA,
  "PMID: 12345678"
))

# The ISBN hides the PMID-like digits inside it ...
locate_scholid("ISBN 978 0 306 40615 7")
# ... unless ISBN is not looked for.
locate_scholid("ISBN 978 0 306 40615 7", types = "pmid")
```

normalize_scholid	<i>Normalize scholarly identifiers</i>
-------------------	--

Description

Vectorized normalizer that converts supported scholarly identifier values to a canonical form (e.g., removing URL prefixes, labels, or separators).

Normalization requires that inputs match the expected identifier structure. For identifier types that define a checksum, normalization also requires checksum-valid values. Inputs that do not meet these requirements yield `NA_character_`.

Normalized outputs are canonical, type-specific representations of valid identifiers.

Use `is_scholid()` to test whether already-canonical values are valid identifiers of a given type. Both functions apply checksum verification where applicable; normalization additionally accepts wrapped input forms and returns canonical strings.

Before normalizing, invisible characters are removed and Unicode spaces count as whitespace. For some types, Unicode dashes and full-width digits are read as their ASCII forms. See "Input characters" in the *How Scholarly Identifiers Are Defined* vignette (`vignette("scholid_definitions", package = "scholid")`). Its DOI section covers DOI case and the percent-decoding of `doi.org` URLs.

Wrapped forms include each type's resolver URL and CURIE, and arXiv DOIs for arXiv identifiers. "Resolver URLs and CURIEs" in the same vignette says how they are read, and each type's section names its forms. `format_scholid()` writes these forms.

Usage

```
normalize_scholid(x, type)
```

Arguments

<code>x</code>	A vector of values to normalize.
<code>type</code>	A single string giving the identifier type. See <code>scholid_types()</code> for supported values.

Value

A character vector with the same length as `x`. Values that are NA, invalid, checksum-failing, or structurally non-matching yield `NA_character_`.

See Also

[is_scholid\(\)](#), [format_scholid\(\)](#), [scholid_key\(\)](#), [scholid_types\(\)](#)

Examples

```
normalize_scholid("https://doi.org/10.1000/182", "doi")
normalize_scholid("https://orcid.org/0000-0002-1825-0097", "orcid")
normalize_scholid("pubmed:12345678", "pmid")
normalize_scholid("https://doi.org/10.48550/arXiv.2101.00001", "arxiv")
```

scholid_key	<i>Comparison keys for scholarly identifiers</i>
-------------	--

Description

Vectorized helper that gives each value a key for comparing identifiers. Values that name the same thing get the same key, even when they are written differently: an ISBN-10 and its ISBN-13, versions of one arXiv preprint or RefSeq sequence, or DOIs that differ only in the case of ASCII letters. Use keys with `duplicated()`, `unique()`, `match()`, and `merge()`.

`x` is first normalized with the rules of [normalize_scholid\(\)](#), so wrapped input such as URLs and labels works. Then the type's key rule is applied. For a type without a key rule, the key is the normalized value. The rules are in each type's "Validation in scholid" section of the *How Scholarly Identifiers Are Defined* vignette (`vignette("scholid_definitions", package = "scholid")`).

A key is for comparing, not an identifier. Some keys don't pass [is_scholid\(\)](#), such as `NM_000546` for the RefSeq accession `NM_000546.5`, and a key can leave out what tells two records apart, such as a version. Don't store or show keys as identifiers; keep the normalized values for that.

Usage

```
scholid_key(x, type)
```

Arguments

<code>x</code>	A vector of values to compare.
<code>type</code>	A single string giving the identifier type. See scholid_types() for supported values.

Value

A character vector with the same length as `x`. Values that are NA or don't normalize yield `NA_character_`.

See Also

`normalize_scholid()`, `format_scholid()`, `is_scholid()`, `scholid_types()`

Examples

```
isbns <- c("0306406152", "ISBN 978-0-306-40615-7", "0-8044-2957-X")
scholid_key(isbns, "isbn")
duplicated(scholid_key(isbns, "isbn"))

dois <- c("10.1000/abc", "https://doi.org/10.1000/ABC")
duplicated(scholid_key(dois, "doi"))

# Versions of one preprint share a key
scholid_key(c("2101.00001v1", "arXiv:2101.00001v2"), "arxiv")
```

scholid_types

Supported scholid identifier types

Description

Returns the set of identifier types supported by the scholid package in classification priority order (most specific first). For per-type formats, validation rules, and classification precedence, see the *How Scholarly Identifiers Are Defined* vignette (`vignette("scholid_definitions", package = "scholid")`).

Usage

```
scholid_types()
```

Value

A character vector of supported identifier type strings.

Examples

```
scholid_types()
"orcid" %in% scholid_types()
```

Index

`classify_scholid`, [3](#)
`classify_scholid()`, [2–4](#), [8](#)

`detect_scholid_type`, [3](#)
`detect_scholid_type()`, [2](#), [3](#)

`extract_scholid`, [4](#)
`extract_scholid()`, [2](#), [8](#)

`format_scholid`, [5](#)
`format_scholid()`, [2](#), [9–11](#)

`is_scholid`, [6](#)
`is_scholid()`, [2](#), [9–11](#)

`locate_scholid`, [8](#)
`locate_scholid()`, [2](#), [5](#)

`match.arg()`, [6](#)

`normalize_scholid`, [9](#)
`normalize_scholid()`, [2–7](#), [10](#), [11](#)

`scholid` (`scholid`-package), [2](#)
`scholid`-package, [2](#)
`scholid_key`, [10](#)
`scholid_key()`, [2](#), [6](#), [10](#)
`scholid_types`, [11](#)
`scholid_types()`, [2–11](#)