Package 'metascience'

November 23, 2026

Title Catalogue and Shared Infrastructure for Open Metascience Tools

Version 0.1.0

Description Catalogue and shared conventions for a set of independent, interoperable tools that study research: research indicators, narrative CVs, academic careers, artificial intelligence use in research, science communication and the integrity and reproducibility of research outputs. Lists the tools with their domain, maturity and release status; reports which of them are installed; validates and normalises persistent identifiers (DOI, 'ORCID' iD, 'ROR' ID) with their check digits; harmonises the result statuses the tools use, so that an unverifiable result is never read as a negative one; writes a machine-readable provenance manifest for any analysis; and offers a 'Shiny' interface with a section for each installed tool. Every tool remains usable on its own.

License MIT + file LICENSE

URL https://github.com/pablosancheznunez/metascience

BugReports https://github.com/pablosancheznunez/metascience/issues

Depends R (>= 4.1.0)

Imports jsonlite, tools, utils

Suggests shiny, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Pablo Sánchez-Núñez [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-7845-9506>)

Maintainer Pablo Sánchez-Núñez <psancheznunez@outlook.com>

Repository CRAN

Date/Publication 2026-11-23 15:00:09 UTC

Contents

ms_app . . . 2
ms_catalogue . . . 3
ms_doi . . . 4
ms_harmonise_status . . . 5
ms_identify . . . 5
ms_manifest . . . 6
ms_status . . . 7
ms_vocabulary . . . 8

Index **9**

`ms_app` *Interactive interfaces for the ecosystem*

Description

Launches one Shiny application with a section per tool: research indicators (openresearchmetrics), forensic scientometrics (FoSci), AI use in research (aiinresearch), reproducibility (researchreproducibility) and science communication (scicomm), plus the catalogue. Each section uses its package if it is installed and otherwise says how to install it, so the packages stay independent of the interface.

Usage

```
ms_app(...)
```

Arguments

`...`	Passed to `shiny::shinyApp()` (for example `options = list(port = 8080)`).

Details

Nothing is sent anywhere except the open APIs the tools themselves query (for example OpenAlex for an ORCID iD). Uploaded files stay in a temporary directory of the R session.

Value

A Shiny app object; printing it launches the app.

Examples

```
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  ms_app()
}
```

`ms_catalogue` *Catalogue of the metascience ecosystem*

Description

Lists the tools of the ecosystem with their research domain, maturity and release status. Maturity uses five levels, so that a prototype is never presented as a finished product: `implemented`, `implemented_not_validated`, `prototype`, `planned` and `blocked`.

Usage

```
ms_catalogue(domain = NULL, maturity = NULL)

ms_domains()

ms_outside_scope()
```

Arguments

`domain`	Optional domain id(s) to keep; see `ms_domains()`.
`maturity`	Optional maturity level(s) to keep.

Details

A component with `frozen = TRUE` has been submitted to CRAN and is not changed while CRAN reviews it; the ecosystem integrates it by reference.

Value

A data frame with one row per component.

For `ms_outside_scope()`, the projects deliberately kept outside the ecosystem and the reason.

Examples

```
ms_catalogue()
ms_catalogue(domain = "science_of_science")
ms_catalogue(maturity = "planned")
```

`ms_doi` *Validate and normalise persistent identifiers*

Description

Each function accepts identifiers as typed by people or returned by APIs (bare, with a prefix such as `doi:`, or as a resolver URL) and returns the canonical bare form, or `NA` when the input is not a valid identifier. They are vectorised and never call the network: a valid identifier is well-formed, which does not prove that it is registered.

Usage

```
ms_doi(x, url = FALSE)

ms_orcid(x, url = FALSE)

ms_ror(x, url = FALSE)
```

Arguments

`x`	A character vector.
`url`	If `TRUE`, return the resolver URL instead of the bare form.

Details

- `ms_doi()`: lower-cased `10.xxxx/suffix`. DOIs are case-insensitive.
- `ms_orcid()`: `0000-0000-0000-000X`, with the ISO 7064 MOD 11-2 check digit verified.
- `ms_ror()`: the nine-character ROR ID, with its ISO 7064 MOD 97-10 check digits verified.

Value

A character vector the length of `x`.

Examples

```
ms_doi(c("https://doi.org/10.5281/ZENODO.23881633", "doi:10.1000/xyz", "nope"))
ms_orcid(c("https://orcid.org/0000-0002-1825-0097", "0000-0002-1825-0098"))
ms_ror("https://ror.org/03ths8210")
ms_ror("03ths8210", url = TRUE)
```

ms_harmonise_status *Harmonise result statuses across tools*

Description

Each tool reports results in its own terms (for example `computed` in openresearchmetrics or `not_found_in_sources` in narrativecv). This function translates them to the common `result_status` vocabulary so that results from several tools can be combined in one table without reading an unverifiable result as a negative one.

Usage

```
ms_harmonise_status(status, tool)

ms_status_mappings()
```

Arguments

`status`	A character vector of statuses as reported by `tool`.
`tool`	The tool that produced them; one of the names of `ms_status_mappings()`.

Value

A character vector of common statuses. An unknown status is an error, never silently mapped.

For `ms_status_mappings()`, a data frame with columns `tool`, `status` and `common_status`.

Examples

```
ms_harmonise_status(c("found", "not_found_in_sources", "source_unavailable"),
                    tool = "narrativecv")
ms_status_mappings()
```

ms_identify *Recognise the type of persistent identifiers*

Description

Recognise the type of persistent identifiers

Usage

```
ms_identify(x)
```

Arguments

`x`	A character vector of identifiers in any common form.

Value

A data frame with columns `input`, `type` (`"doi"`, `"orcid"`, `"ror"` or `NA`) and `id` (the normalised identifier).

Examples

```
ms_identify(c("10.5281/zenodo.23881631", "0000-0002-1825-0097",
              "https://ror.org/03ths8210", "not an identifier"))
```

`ms_manifest` *Provenance manifest for an analysis*

Description

Records what is needed to understand and repeat an analysis: the software environment, the versions of the ecosystem packages, the data sources consulted (and whether they answered), the methodology versions applied, checksums of input files and the known limitations. The manifest is plain JSON (schema `metascience-manifest/1`), so any tool can read it.

Usage

```
ms_manifest(
  sources = list(),
  methods = character(),
  inputs = character(),
  packages = character(),
  limitations = character(),
  generated_at = utc_now()
)

ms_write_manifest(manifest, path)
```

Arguments

`sources`	Data sources consulted: a data frame or a list of records with at least `source`, and optionally `ok`, `count`, `error` and `retrieved_at`. The `consulted` element returned by the collectors of openresearchmetrics and narrativecv can be passed as is.
`methods`	Named character vector of methodology versions, for example `c("oa-publications-share" = "1.0.0")`.
`inputs`	Paths of input files; each is recorded with its size and MD5 checksum.
`packages`	Further packages whose versions should be recorded, in addition to the ecosystem packages that are installed.
`limitations`	Character vector of limitations to report with the results.
`generated_at`	Timestamp (ISO 8601, UTC).
`manifest`	An object from `ms_manifest()`.
`path`	File to write.

Value

An object of class `ms_manifest` (a list).

For `ms_write_manifest()`, `path`, invisibly.

Examples

```
m <- ms_manifest(
  sources = list(list(source = "openalex", ok = TRUE, count = 25,
                      retrieved_at = "2026-11-02T00:00:00Z")),
  methods = c("oa-publications-share" = "1.0.0"),
  limitations = "OpenAlex coverage is weaker for books."
)
m
ms_write_manifest(m, tempfile(fileext = ".json"))
```

`ms_status` *Which ecosystem packages are installed*

Description

Reports, for every component that is an R package, whether it is installed and which version. Nothing is installed or loaded.

Usage

```
ms_status(lib.loc = NULL)
```

Arguments

`lib.loc`	Library paths to search, as in `utils::packageVersion()`.

Value

A data frame with columns `id`, `r_package`, `maturity`, `installed` and `version`.

Examples

```
ms_status()
```

ms_vocabulary *Shared vocabularies of the ecosystem*

Description

Controlled terms that every tool uses in the same sense:

Usage

```
ms_vocabulary(name)

ms_vocabularies()
```

Arguments

name	The vocabulary; one of `ms_vocabularies()`.

Details

- `result_status`: what the sources allowed to be said about a result (`reported`, `absent_in_sources`, `unverifiable`, `not_automatable`);
- `evidence_level`: how strong the evidence behind an integrity-related finding is (`signal`, `pattern`, `documentary`, `established`);
- `ai_use_basis`: how a statement about AI use is known (`declared`, `inferred`, `verified`).

Each term comes with its definition and with how it may be read, so that reports can state the limits of a result next to the result itself.

Value

For `ms_vocabulary()`, a data frame with columns `id`, `definition` and `may_be_read_as`. For `ms_vocabularies()`, the names of the vocabularies.

Examples

```
ms_vocabularies()
ms_vocabulary("evidence_level")
```

Index

ms_app, 2
ms_catalogue, 3
ms_doi, 4
ms_domains *(*ms_catalogue*)*, 3
ms_harmonise_status, 5
ms_identify, 5
ms_manifest, 6
ms_orcid *(*ms_doi*)*, 4
ms_outside_scope *(*ms_catalogue*)*, 3
ms_ror *(*ms_doi*)*, 4
ms_status, 7
ms_status_mappings
        *(*ms_harmonise_status*)*, 5
ms_vocabularies *(*ms_vocabulary*)*, 8
ms_vocabulary, 8
ms_write_manifest *(*ms_manifest*)*, 6

shiny::shinyApp(), *2*

utils::packageVersion(), *7*