

Package: prismaid (via r-universe)

October 4, 2026

Title Open Science AI Tools for Systematic, Protocol-Based Literature Reviews

Version 0.17.1

Description prismaAId leverages generative AI models to screen and extract data from scientific literature. It provides efficient, replicable, and user-friendly methods for conducting systematic reviews. Designed in line with Open Science practices, prismaAId requires no coding skills to use.

License AGPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Repository <https://open-and-sustainable.r-universe.dev>

Date/Publication 2026-09-09 06:26:07 UTC

RemoteUrl <https://github.com/open-and-sustainable/prismaid>

RemoteRef main

RemoteSha 31825216e4a97a06308822063a8ae3274bf49dcf

RemoteSubdir r-package

Contents

CheckConformance	2
Convert	3
DownloadURLList	4
DownloadZotero	4
GenerateRevAIsRecord	5
MergeRecordStage	6
ProtocolGuidance	6
RevAIsSchema	7
RunReview	8
Screening	10
ValidateConfig	13
ValidateRecord	14

CheckConformance	<i>Check Protocol Conformance</i>
------------------	-----------------------------------

Description

Validates a RevAIsE review-record JSON string against a protocol's SHACL shapes (for example "prisma-2020"). The verdict and the per-constraint messages come from the protocol shapes, so conformance is decided by the shapes rather than asserted by the tool.

Usage

```
CheckConformance(record_json, protocol)
```

Arguments

<code>record_json</code>	The RevAIsE review record as a JSON string.
<code>protocol</code>	The protocol identifier to check against (for example "prisma-2020").

Details

This function checks whether a RevAIsE review record conforms to a reporting protocol, using the SHACL shapes published by the RevAIsE model.

Value

A JSON string with the conformance report: an object with `protocol`, `conforms`, and `violations` (each carrying a message). On failure the JSON has an `error` field. Parse it with a JSON reader such as `jsonlite::fromJSON`.

Examples

```
## Not run:
record <- paste(readLines("review.revaise.json"), collapse = "\n")
report <- CheckConformance(record, "prisma-2020")

## End(Not run)
```

Convert	<i>Convert Files to Different Formats</i>
---------	---

Description

Converts files in the input directory to the requested formats.

Usage

```
Convert(input_dir, selected_formats, tika_address = "", single_file = "",
        ocr_only = FALSE)
```

Arguments

input_dir	Directory containing files to convert
selected_formats	Comma-separated list of target formats (e.g., "pdf,docx,html")
tika_address	Tika server address for OCR fallback (e.g., "localhost:9998"). Empty string disables OCR fallback. Defaults to "".
single_file	Convert only the specified PDF (PDF format only). Defaults to "".
ocr_only	Force OCR for PDFs via Tika (PDF format only). Defaults to FALSE.

Details

This function converts files in a directory to specified formats.

Value

A string indicating the result of the conversion process

Examples

```
## Not run:
Convert("/path/to/files", "pdf,docx")

Convert("/path/to/files", "pdf", "localhost:9998")

Convert("/path/to/files", "pdf", "localhost:9998", "", TRUE)

## End(Not run)
```

DownloadURLList	<i>Download Files from URL List</i>
-----------------	-------------------------------------

Description

Reads a file containing a list of URLs (one per line) and downloads the files.

Usage

```
DownloadURLList(path)
```

Arguments

path	Path to the file containing URLs
------	----------------------------------

Details

This function downloads files from URLs listed in a file.

Value

A string indicating the result of the download process

Examples

```
## Not run:
DownloadURLList("/path/to/url_list.txt")

## End(Not run)
```

DownloadZotero	<i>Download PDFs from Zotero</i>
----------------	----------------------------------

Description

Downloads PDF documents from a Zotero collection using the Zotero API. The TOML configuration must contain a '[zotero]' table and may contain an optional '[revaise]' block to document the download in a RevAIs record.

Usage

```
DownloadZotero(toml_content)
```

Arguments

toml_content	Zotero TOML configuration as a string
--------------	---------------------------------------

Details

This function downloads PDFs from Zotero using a TOML configuration.

Value

A string indicating the result of the download process

Examples

```
## Not run:
toml_content <- paste(readLines("zotero.toml"), collapse = "\n")
DownloadZotero(toml_content)

## End(Not run)
```

GenerateRevAIsRecord *Generate a Seed RevAIs Review Record*

Description

Produces a valid review header from the supplied parameters; when `include_manual_stage_stubs` is true it also adds empty stubs for the stages prismAId does not perform (registration, search, risk of bias, synthesis), ready to fill in.

Usage

```
GenerateRevAIsRecord(params_json)
```

Arguments

`params_json` A JSON string of record parameters, for example `'{"title":"...", "authors":["..."], "type":"SYS..."}`

Details

This function builds a seed RevAIs review record from a JSON parameters object.

Value

A JSON string: the seed review record, suitable for writing to the configuration's `record_file`. On failure the JSON has an error field.

Examples

```
## Not run:
record <- GenerateRevAIsRecord('{"title":"My review", "include_manual_stage_stubs":true}')
writeLines(record, "review.revaise.json")

## End(Not run)
```

MergeRecordStage	<i>Merge a Stage into a RevAIsE Review Record</i>
------------------	---

Description

Merges a stage into an existing RevAIsE review record. The stage (a JSON object with at least a stage_type) fills a matching stub (matched by stage_type and stage_label) or is appended when none matches.

Usage

```
MergeRecordStage(record_json, stage_json)
```

Arguments

record_json	The existing review record as a JSON string.
stage_json	The stage to merge, as a JSON object string.

Value

A JSON string: the updated review record. On failure the JSON has an error field.

Examples

```
## Not run:
updated <- MergeRecordStage(record, '{"stage_type":"search","stage_label":"Search"}')

## End(Not run)
```

ProtocolGuidance	<i>Protocol Guidance</i>
------------------	--------------------------

Description

Returns the requirements a protocol imposes (for example "prisma-2020"), grouped by the record class each applies to. It is advisory: it describes what a conforming review needs and does not constrain the order in which prismaAId's tools are used.

Usage

```
ProtocolGuidance(protocol)
```

Arguments

protocol	The protocol identifier (for example "prisma-2020").
----------	--

Details

This function returns a protocol's full requirement checklist, extracted from the SHACL shapes published by the RevAIschema model.

Value

A JSON string with the guidance: an object with protocol, name, version, status, and requirements (each carrying a target_class and a message). On failure the JSON has an error field. Parse it with a JSON reader such as `jsonlite::fromJSON`.

Examples

```
## Not run:
guidance <- ProtocolGuidance("prisma-2020")

## End(Not run)
```

RevAIschema

Serve the RevAIschema Data Model Schema

Description

By default it describes a type, or lists the available classes and enums when no type is given. The LinkML source is never used.

Usage

```
RevAIschema(params_json)
```

Arguments

params_json	A JSON string of parameters, for example <code>'{"type": "SearchStage"}'</code> to describe a type, <code>'{}'</code> to list classes and enums, <code>'{"raw": true}'</code> for the full JSON Schema, or <code>'{"context": true}'</code> for the JSON-LD context.
-------------	--

Details

This function serves the RevAIschema data model from the released, verified artifacts published by RevAIschema (JSON Schema and JSON-LD context), fetched live.

Value

A JSON string with the description (or the raw artifact). On failure the JSON has an error field. Parse it with `jsonlite::fromJSON`.

Examples

```
## Not run:
searchstage <- RevAIseSchema('{ "type": "SearchStage" }')

## End(Not run)
```

RunReview

Run Review

Description

The input data must be structured in a TOML format, consisting of several sections and parameters.

Usage

```
RunReview(input_string)
```

Arguments

`input_string` A string representing the input data.

Details

This function interfaces with a shared library to perform a review process on the input data.

[project]

- `name`: A string representing the project title. Example: "Use of LLM for systematic review".
- `author`: The name of the project author. Example: "John Doe".
- `version`: The version number for the project configuration. Example: "1.0".

[project.configuration]

- `input_directory`: The file path to the directory containing manuscripts to be reviewed. Example: "/path/to/txt/files".
- `results_file_name`: The path and base name for saving results (file extension will be added automatically). Example: "/path/to/save/results".
- `output_format`: The format for output results. Options: "csv" (default) or "json".
- `log_level`: Determines logging verbosity:
 - "low": Minimal logging (default).
 - "medium": Logs to standard output.
 - "high": Logs to a file (see user manual for details).
- `duplication`: Runs model queries twice for debugging purposes. Options: "yes" or "no" (default).
- `cot_justification`: Requests chain-of-thought justification from the model. Options: "yes" or "no" (default).

- **summary:** Generates and saves summaries of manuscripts. Options: "yes" or "no" (default).

[project.llm]

- Configuration for LLMs, supporting multiple instances (llm.1, llm.2, etc.) for ensemble reviews.
- Parameters for each LLM include:
 - **provider:** The LLM service provider. Options: "OpenAI", "GoogleAI", "Cohere", "Anthropic", "DeepSeek", or "Perplexity".
 - **api_key:** API key for the provider. If empty, environment variables will be checked.
 - **model:** Model name. Options vary by provider:
 - * OpenAI: "gpt-5-nano", "gpt-5-mini", "gpt-5.2", "gpt-5.1", "gpt-5", "o4-mini", "o3-mini", "o3", "o1-mini", "o1", "gpt-4.1-nano", "gpt-4.1-mini", "gpt-4.1", "gpt-4o-mini", "gpt-4o", "gpt-4-turbo", "gpt-3.5-turbo", or "" (default for cost optimization).
 - * GoogleAI: "gemini-3-flash-preview", "gemini-3-pro-preview", "gemini-2.5-flash-lite", "gemini-2.5-flash", "gemini-2.5-pro", "gemini-2.0-flash-lite", "gemini-2.0-flash", "gemini-1.5-flash", "gemini-1.5-pro", or "" (default for cost optimization).
 - * Cohere: "command-a-reasoning-08-2025", "command-a-03-2025", "command-r-08-2024", "command-r7b-12-2024", "command-r-plus", "command-r", "command-light", "command", or "" (default for cost optimization).
 - * Anthropic: "claude-4-5-haiku", "claude-4-5-sonnet", "claude-4-5-opus", "claude-4-0-opus", "claude-4-0-sonnet", "claude-3-7-sonnet", "claude-3-5-sonnet", "claude-3-5-haiku", "claude-3-opus", "claude-3-sonnet", "claude-3-haiku", or "" (default for cost optimization).
 - * DeepSeek: "deepseek-chat", "deepseek-reasoner", or "" (default for cost optimization).
 - * Perplexity: "sonar-deep-research", "sonar-reasoning-pro", "sonar-pro", "sonar", or "" (default for cost optimization).
 - **temperature:** Controls model randomness. Range: 0 to 1 (0 to 2 for GoogleAI). Lower values reduce randomness.
 - **tpm_limit:** Tokens per minute limit before delaying prompts. Default: 0 (no delay).
 - **rpm_limit:** Requests per minute limit before delaying prompts. Default: 0 (no delay).

[prompt]

- Defines the main components of the prompt for reviews.
- **persona:** Optional text specifying the model's role. Example: "You are an experienced scientist..."
- **task:** Required text framing the task for the model. Example: "Map the concepts discussed in a scientific paper..."
- **expected_result:** Required text describing the expected output structure in JSON.
- **definitions:** Optional text defining concepts to clarify instructions. Example: "'Interest rate' is defined as..."
- **example:** Optional example to illustrate concepts.
- **failsafe:** Specifies a fallback if the concepts cannot be identified. Example: "Respond with an empty '' value if concepts are unclear".

[review]

- Defines the keys and possible values in the JSON object for the review.
- Example entries:
 - [review.1]: key = "interest rate", values = [""]
 - [review.2]: key = "regression models", values = ["yes", "no"]
 - [review.3]: key = "geographical scale", values = ["world", "continent", "river basin"]

Optional chunking

- [project.configuration.chunking] is disabled by default. When enabled, it splits only prompts over the user-defined input_context_tokens limit.
- An enabled plan must define an explicit merge rule for every configured review field. See the Review Tool documentation for the TOML shape.
- Chunked runs write <results_file_name>.chunking-report.json with the chunk plan, value coercions, conflicts, and document failures. If a document fails, this function returns an error message only after the normal results file and sidecar have been preserved.

Value

A string indicating the result of the review process.

Examples

```
RunReview("example input")
```

Screening	<i>Screen Manuscripts for Systematic Review</i>
-----------	---

Description

Processes a list of manuscripts applying multiple filters to identify which should be excluded from a systematic review. Supports both rule-based and AI-assisted screening.

Usage

```
Screening(input_string)
```

Arguments

input_string A string containing the TOML configuration for screening.

Details

This function screens manuscripts to identify items for exclusion based on various criteria.

The input data must be structured in a TOML format with the following sections:

[project]

- name: Project title. Example: "Screening for climate change literature".
- author: Name of the project author. Example: "Jane Smith".
- version: Version number for the configuration. Example: "1.0".
- input_file: Path to CSV/JSON file containing manuscripts to screen. Example: "/path/to/manuscripts.csv".
- output_file: Path where screening results will be saved. Example: "/path/to/screening_results".
- text_column: Name of column containing text or path to text files. Example: "abstract" or "text_file_path".
- identifier_column: Column name for unique manuscript identifiers. Example: "doi" or "id".
- output_format: Format for results. Options: "csv" or "json".
- log_level: Logging verbosity. Options: "low", "medium", or "high".

[filters.deduplication]

- enabled: Whether to apply deduplication. Options: true or false.
- use_ai: Use AI for semantic similarity detection. Options: true or false.
- compare_fields: List of fields to compare. Example: "title", "abstract", "doi".

[filters.language]

- enabled: Whether to filter by language. Options: true or false.
- accepted_languages: List of accepted language codes. Example: "en", "es", "fr".
- use_ai: Use AI for language detection. Options: true or false.

[filters.article_type]

- enabled: Whether to filter by article type. Options: true or false.
- use_ai: Use AI for article classification. Options: true or false.
- exclude_reviews: Exclude review articles. Options: true or false.
- exclude_editorials: Exclude editorial articles. Options: true or false.
- exclude_letters: Exclude letters to editor. Options: true or false.
- exclude_theoretical: Exclude theoretical papers. Options: true or false.
- exclude_empirical: Exclude empirical studies. Options: true or false.
- exclude_methods: Exclude methodology papers. Options: true or false.
- exclude_single_case: Exclude single case studies. Options: true or false.
- exclude_sample: Exclude sample-based studies. Options: true or false.
- include_types: Specific article types to include. Example: "research", "case_study".

[filters.topic_relevance]

- **enabled:** Whether to filter by topic relevance. Options: true or false.
- **use_ai:** Use AI for relevance scoring. Options: true or false.
- **topics:** List of topic descriptions. Example: "climate change impacts", "adaptation strategies".
- **min_score:** Minimum relevance score (0-1) to include. Example: 0.7.
- **score_weights.keyword_match:** Weight for keyword matching (0-1). Example: 0.3.
- **score_weights.concept_match:** Weight for concept matching (0-1). Example: 0.4.
- **score_weights.field_relevance:** Weight for field relevance (0-1). Example: 0.3.

[filters.llm] (Optional, required if any filter has use_ai = true)

- Configuration for AI models, supporting multiple instances (llm.1, llm.2, etc.).
- Parameters for each LLM:
 - **provider:** LLM service provider. Options: "OpenAI", "GoogleAI", "Cohere", "Anthropic", "DeepSeek", or "Perplexity".
 - **api_key:** API key for the provider. If empty, environment variables will be checked.
 - **model:** Model name (see RunReview documentation for available models per provider).
 - **temperature:** Controls randomness (0-1, or 0-2 for GoogleAI).
 - **tpm_limit:** Tokens per minute limit. Default: 0 (no limit).
 - **rpm_limit:** Requests per minute limit. Default: 0 (no limit).

Value

A string indicating the result of the screening process.

Examples

```
## Not run:
config <- '
[project]
name = "Climate Literature Screening"
author = "Research Team"
version = "1.0"
input_file = "/data/manuscripts.csv"
output_file = "/results/screening"
text_column = "abstract"
identifier_column = "doi"
output_format = "csv"
log_level = "medium"

[filters.deduplication]
enabled = true
use_ai = false
compare_fields = ["title", "doi"]

[filters.language]
```

```

enabled = true
accepted_languages = ["en"]
use_ai = false

[filters.article_type]
enabled = true
use_ai = false
exclude_reviews = true
exclude_editorials = true
'
Screening(config)

## End(Not run)

```

ValidateConfig	<i>Validate a Configuration</i>
----------------	---------------------------------

Description

Parses a TOML configuration and checks required fields and value constraints, including any optional '[revaise]' block, without running the tool, accessing the network, or resolving API keys. Use it to check a configuration file before running a review, screening, or Zotero download.

Usage

```
ValidateConfig(config_type, input_string)
```

Arguments

config_type	Which configuration schema to validate against. One of "review", "screening", or "zotero".
input_string	The TOML configuration as a string.

Details

This function validates a prismAId configuration without executing it.

Value

A string indicating the validation result: a confirmation message when valid, or a description of the problem when invalid.

Examples

```

## Not run:
config <- paste(readLines("review.toml"), collapse = "\n")
ValidateConfig("review", config)

## End(Not run)

```

ValidateRecord*Validate a RevAlse Review Record*

Description

Validates a RevAlse review record against the released data-model JSON Schema, fetched live. This checks structural validity (field names, types, required slots), distinct from CheckConformance, which checks a reporting protocol.

Usage

```
ValidateRecord(record_json)
```

Arguments

`record_json` The review record as a JSON string.

Value

A JSON string with valid and errors. On an operational failure the JSON has an error field. Parse it with `jsonlite::fromJSON`.

Examples

```
## Not run:  
result <- ValidateRecord(record)  
  
## End(Not run)
```

Index

CheckConformance, [2](#)

Convert, [3](#)

DownloadURLList, [4](#)

DownloadZotero, [4](#)

GenerateRevAIsRecord, [5](#)

MergeRecordStage, [6](#)

ProtocolGuidance, [6](#)

RevAIsSchema, [7](#)

RunReview, [8](#)

Screening, [10](#)

ValidateConfig, [13](#)

ValidateRecord, [14](#)