

Package: r4subcore (via r-universe)

October 11, 2026

Title Evidence Schema, Parsers, and Shared Primitives

Version 0.3.0

Description Foundational package in the R4SUB (Ready for Submission) ecosystem. Defines the core evidence table schema, parsers, indicator abstractions, and scoring primitives needed to quantify clinical submission readiness. Provides a standardized contract for ingesting heterogeneous sources (validation outputs, metadata, traceability) into a single evidence framework.

License MIT + file LICENSE

URL <https://r4sub.github.io/r4subcore/>,
<https://github.com/R4SUB/r4subcore>

BugReports <https://github.com/R4SUB/r4subcore/issues>

Depends R (>= 4.2)

Imports cli, jsonlite, rlang, stats, tibble, utils, xml2

Suggests knitr, r4subdata, readxl, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/pak/sysreqs libxml2-dev

Repository <https://r4sub.r-universe.dev>

Date/Publication 2026-09-11 16:27:06 UTC

RemoteUrl <https://github.com/r4sub/r4subcore>

RemoteRef HEAD

RemoteSha f7b003c0d12ddb94c635de4e3cbce89ec6d58ef0

Contents

adrg_sections	2
adrg_to_evidence	3
aggregate_indicator_score	4
as_evidence	5
bind_evidence	6
canon_result	6
canon_severity	7
check_define_consistency	8
define_variables	9
define_xml_to_evidence	9
evidence_diff	11
evidence_schema	12
evidence_schema_spec	13
evidence_schema_version	13
evidence_sources	14
evidence_summary	15
export_evidence	15
hash_id	16
import_evidence	17
json_safely	18
migrate_evidence	18
normalize_01	19
p21_to_evidence	20
r4sub_run_context	21
read_p21_report	22
register_indicator	23
result_to_score	24
severity_to_weight	24
validate_evidence	25
validate_indicator	26
Index	27

adrg_sections	<i>Standard ADRG Section Catalog</i>
---------------	--------------------------------------

Description

Returns the standard top-level and key subsections of the Analysis Data Reviewer's Guide (ADRG), following the PHUSE ADRG completion guideline. This is the checklist [adrg_to_evidence\(\)](#) scores a guide against.

Usage

```
adrg_sections()
```

Value

A tibble with columns `section_id`, `section_name`, and `required`.

Examples

```
adrg_sections()
```

adrg_to_evidence	<i>Convert ADRG Section Coverage to Evidence</i>
------------------	--

Description

Scores an Analysis Data Reviewer's Guide for completeness against the standard section catalog and emits R4SUB evidence rows, one per section.

Usage

```
adrg_to_evidence(
  present,
  ctx,
  source_name = "adrg",
  source_version = NULL,
  standard = adrg_sections()
)
```

Arguments

<code>present</code>	A character vector of <code>section_ids</code> the guide contains, or a <code>data.frame</code> with a <code>section_id</code> column and a logical <code>present</code> column.
<code>ctx</code>	An r4sub_run_context .
<code>source_name</code>	Source label. Default "adrg"; use "sdrg" for a Study Data Reviewer's Guide.
<code>source_version</code>	Optional source version string.
<code>standard</code>	The section catalog to score against. Defaults to adrg_sections() .

Details

ADRG and SDRG documents are Word or PDF, and reliable text extraction from them is out of scope for this package. `adrg_to_evidence()` therefore takes a structured summary of which sections a guide contains, which a document pipeline (or a human reviewer) can produce, and turns it into evidence. A present section passes. A missing required section fails at high severity; a missing recommended section warns at low severity. The rows land in the usability pillar, since the reviewer's guide is reviewer-facing documentation.

Value

An evidence `data.frame`, one row per standard section.

Examples

```
ctx <- suppressMessages(r4sub_run_context("STUDY1", "DEV"))
ev <- suppressMessages(adrg_to_evidence(c("1", "2", "3", "4", "5"), ctx))
ev[, c("indicator_id", "severity", "result")]
```

aggregate_indicator_score

Aggregate Indicator Scores

Description

Computes summary scores from an evidence table, grouped by one or more columns.

Usage

```
aggregate_indicator_score(
  ev,
  by = "indicator_id",
  method = c("mean", "min", "weighted")
)
```

Arguments

ev	A valid evidence data.frame.
by	Character vector of column names to group by. Default: c("indicator_id").
method	Aggregation method: "mean", "min", or "weighted". The "weighted" method uses severity_to_weight() and result_to_score() .

Value

A data.frame with grouping columns plus score (0–1) and n_evidence (count of rows).

Examples

```
ctx <- suppressMessages(r4sub_run_context("STUDY1", "DEV"))
ev <- suppressMessages(as_evidence(
  data.frame(
    asset_type = rep("validation", 3), asset_id = rep("ADSL", 3),
    source_name = rep("pinnacle21", 3),
    indicator_id = c("SD0001", "SD0001", "SD0002"),
    indicator_name = c("SD0001", "SD0001", "SD0002"),
    indicator_domain = rep("quality", 3),
    severity = c("high", "medium", "low"),
    result = c("fail", "warn", "pass"),
    stringsAsFactors = FALSE
  ),
  ctx = ctx
```

```

))
aggregate_indicator_score(ev, by = "indicator_id", method = "weighted")

```

as_evidence	<i>Coerce to Evidence Table</i>
-------------	---------------------------------

Description

Takes a data.frame and coerces it into a valid evidence table. Fills in missing nullable columns with NA of the correct type and validates controlled vocabulary columns.

Usage

```
as_evidence(x, ctx = NULL, ...)
```

Arguments

x	A data.frame (or tibble) with at least the required evidence columns.
ctx	An optional r4sub_run_context . If provided, run_id and study_id are filled from the context when missing.
...	Additional columns to set (e.g., asset_type = "validation").

Value

A data.frame conforming to the evidence schema.

Examples

```

ctx <- r4sub_run_context("STUDY1", "DEV")
df <- data.frame(
  asset_type = "validation",
  asset_id = "ADSL",
  source_name = "pinnacle21",
  indicator_id = "P21-001",
  indicator_name = "Missing variable",
  indicator_domain = "quality",
  severity = "high",
  result = "fail",
  message = "Variable AGEU missing",
  stringsAsFactors = FALSE
)
ev <- as_evidence(df, ctx = ctx)

```

bind_evidence	<i>Bind Evidence Tables</i>
---------------	-----------------------------

Description

Row-binds multiple evidence data.frames after validating each one.

Usage

```
bind_evidence(...)
```

Arguments

... Evidence data.frames to bind.

Value

A single combined evidence data.frame.

Examples

```
ctx <- suppressMessages(r4sub_run_context("STUDY1", "DEV"))
make_ev <- function(ind_id) {
  suppressMessages(as_evidence(
    data.frame(
      asset_type = "validation", asset_id = "ADSL",
      source_name = "pinnacle21", indicator_id = ind_id,
      indicator_name = ind_id, indicator_domain = "quality",
      severity = "low", result = "pass",
      stringsAsFactors = FALSE
    ),
    ctx = ctx
  ))
}
ev1 <- make_ev("IND-001")
ev2 <- make_ev("IND-002")
combined <- suppressMessages(bind_evidence(ev1, ev2))
nrow(combined)
```

canon_result	<i>Canonical Result Values</i>
--------------	--------------------------------

Description

Maps common result/status labels to the canonical set: pass, fail, warn, na.

Usage

```
canon_result(x)
```

Arguments

x Character vector of result values.

Value

Character vector with canonical result labels.

Examples

```
canon_result(c("PASS", "Failed", "Warning", "N/A"))
```

canon_severity	<i>Canonical Severity Values</i>
----------------	----------------------------------

Description

Maps common severity labels (case-insensitive) to the canonical set.

Usage

```
canon_severity(x)
```

Arguments

x Character vector of severity values.

Value

Character vector with canonical severity labels.

Examples

```
canon_severity(c("HIGH", "Low", "warning", "Error"))
```

 check_define_consistency

Check Define-XML Against Dataset Contents

Description

Compares the variables declared in Define-XML against the variables actually present in the datasets, and emits evidence for each: a match, a variable documented in Define but absent from the data, or a variable present in the data but not documented. Drift between Define-XML and the data is a common submission problem that reviewers notice quickly, and this makes it a scored indicator. The comparison is case-insensitive on dataset and variable names.

Usage

```
check_define_consistency(
  define,
  data,
  ctx,
  source_name = "define_vs_data",
  source_version = NULL
)
```

Arguments

define	Either a path to a Define-XML file, or a data.frame with dataset and variable columns (as returned by <code>define_variables()</code>).
data	A data.frame with dataset and variable columns listing the variables actually present in each dataset.
ctx	A <code>r4sub_run_context</code> .
source_name, source_version	Provenance recorded on the evidence.

Value

A data.frame conforming to the evidence schema, one row per variable checked, under indicator Q-DEFINE-CONSIST.

See Also

`define_variables()`, `define_xml_to_evidence()`.

Examples

```
ctx <- suppressMessages(r4sub_run_context("STUDY1", "DEV"))
define <- data.frame(
  dataset = c("ADSL", "ADSL", "ADSL"),
  variable = c("USUBJID", "AGE", "SEX")
)
```

```

)
data <- data.frame(
  dataset = c("ADSL", "ADSL", "ADSL"),
  variable = c("USUBJID", "AGE", "RACE") # SEX missing, RACE undocumented
)
ev <- suppressMessages(check_define_consistency(define, data, ctx))
ev[, c("asset_id", "result", "message")]

```

define_variables	<i>Extract the Variables Declared in a Define-XML File</i>
------------------	--

Description

Returns one row per variable declared in Define-XML, following the ItemGroupDef to ItemRef to ItemDef structure: the dataset name comes from the ItemGroupDef, and the variable name from the ItemDef its ItemRef points at. Elements are matched by local name, so the result does not depend on how the file mixes the ODM and def: namespaces.

Usage

```
define_variables(file)
```

Arguments

file	Path to a Define-XML file.
------	----------------------------

Value

A data.frame with columns dataset and variable.

See Also

[check_define_consistency\(\)](#).

define_xml_to_evidence	<i>Parse Define-XML to Evidence</i>
------------------------	-------------------------------------

Description

Reads a Define-XML 2.0/2.1 file and extracts dataset, variable, and derivation completeness checks into the standard evidence table format. Three indicators are evaluated for each asset:

Q-DEFINE-001 Dataset is present and has a non-empty label.

Q-DEFINE-002 Variable is documented (has label and dataType).

Q-DEFINE-003 Derivation text is present for derived variables.

Usage

```
define_xml_to_evidence(file, ctx, source_version = "2.1")
```

Arguments

file Character. Path to a Define-XML file (.xml).

ctx An `r4sub_run_context` providing run and study metadata.

source_version Character. Version label for the Define-XML standard. Default: "2.1".

Value

A data.frame conforming to the evidence schema, one row per dataset-level check (Q-DEFINE-001), per variable check (Q-DEFINE-002), and per derivation check (Q-DEFINE-003).

Examples

```
# Build a minimal Define-XML 2.1 document
xml_txt <- '<?xml version="1.0" encoding="UTF-8"?>
<ODM xmlns="http://www.cdisc.org/ns/odm/v1.3"
  xmlns:def="http://www.cdisc.org/ns/def/v2.1">
  <Study OID="STUDY001">
    <MetaDataVersion OID="MDV.001" Name="Define-XML 2.1">
      <def:ItemGroupDef OID="IG.ADSL" Name="ADSL" SASDatasetName="ADSL"
        Repeating="No" Purpose="Analysis"
        def:Label="Subject-Level Analysis Dataset">
        <ItemRef ItemOID="IT.ADSL.USUBJID" Mandatory="Yes"/>
        <ItemRef ItemOID="IT.ADSL.AGE" Mandatory="Yes"/>
      </def:ItemGroupDef>
      <ItemDef OID="IT.ADSL.USUBJID" Name="USUBJID"
        DataType="text" Length="20"
        SASFieldName="USUBJID">
        <Description><TranslatedText>Unique Subject Identifier</TranslatedText></Description>
      </ItemDef>
      <ItemDef OID="IT.ADSL.AGE" Name="AGE"
        DataType="integer" Length="8"
        SASFieldName="AGE">
        <Description><TranslatedText>Age</TranslatedText></Description>
        <def:Origin Type="Derived">
        <def:Description><TranslatedText>Derived from RFSTDTC</TranslatedText></def:Description>
        </def:Origin>
      </ItemDef>
    </MetaDataVersion>
  </Study>
</ODM>'
tmp <- tempfile(fileext = ".xml")
writeLines(xml_txt, tmp)
ctx <- r4sub_run_context("STUDY001", "DEV")
ev <- define_xml_to_evidence(tmp, ctx)
nrow(ev)
```

evidence_diff	<i>Compare Two Evidence Tables</i>
---------------	------------------------------------

Description

Reports what changed between two evidence tables, for example the same submission checked at two points in time. Rows are matched on a key (by default the indicator applied to a specific asset) and classified as added, removed, changed, or unchanged, with a direction that says whether each change made the submission better or worse.

Usage

```
evidence_diff(  
  old,  
  new,  
  key = c("indicator_id", "asset_id"),  
  include_unchanged = FALSE  
)
```

Arguments

- old, new Valid evidence data.frames, as produced by [as_evidence\(\)](#). old is the earlier state and new the later one.
- key Character vector of columns that identify a comparable row. Default c("indicator_id", "asset_id").
- include_unchanged Logical. If FALSE (default) unchanged matched rows are dropped from the result.

Details

When a key appears on more than one row within a table (an indicator that emits several messages for the same asset, say), the rows are collapsed to the worst one for that key before comparison: lowest result score first (fail beats warn beats pass), then highest severity. This keeps the diff at the indicator level, which is what a submission team acts on.

Value

A data.frame with the key columns plus indicator_name, indicator_domain, change (added, removed, changed, or unchanged), direction (improved, regressed, or neutral), result_from, result_to, severity_from, severity_to, metric_from, and metric_to. Rows are ordered with the most actionable first: regressions, then new findings, then resolved findings, then improvements. The returned object carries a "summary" attribute with the count in each change category.

Examples

```

ctx <- suppressMessages(r4sub_run_context("STUDY1", "DEV"))
mk <- function(id, result, severity) {
  suppressMessages(as_evidence(
    data.frame(
      asset_type = "validation", asset_id = "ADSL",
      source_name = "pinnacle21", indicator_id = id,
      indicator_name = id, indicator_domain = "quality",
      severity = severity, result = result,
      stringsAsFactors = FALSE
    ),
    ctx = ctx
  ))
}
old <- suppressMessages(bind_evidence(mk("A", "fail", "high"), mk("B", "pass", "low")))
new <- suppressMessages(bind_evidence(mk("A", "pass", "high"), mk("C", "fail", "medium")))
evidence_diff(old, new)

```

evidence_schema

*Evidence Table Schema Definition***Description**

Returns the column specification for the R4SUB evidence table. Each element describes a column's expected R type and, where applicable, the set of allowed values.

Usage

```
evidence_schema()
```

Value

A named list. Each element is a list with type (character) and optionally allowed (character vector) or nullable (logical).

Examples

```
str(evidence_schema())
```

evidence_schema_spec	<i>Evidence Schema Specification</i>
----------------------	--------------------------------------

Description

Returns the R4SUB evidence schema as a tidy table: one row per column with its type, whether it is required, the controlled vocabulary (when the column has one), and a short description. This is the published, machine-readable form of the schema that [evidence_schema\(\)](#) encodes, suitable for documentation, a data dictionary, or a Define-XML style reference.

Usage

```
evidence_schema_spec()
```

Value

A tibble with columns column, type, required, allowed_values, and description. allowed_values is a comma-separated string, or NA when the column is free text.

Examples

```
evidence_schema_spec()
```

evidence_schema_version	<i>Evidence Schema Version</i>
-------------------------	--------------------------------

Description

The evidence table carries a schema version so a table written by one release of the ecosystem can be recognized, validated, and migrated by another. `evidence_schema_version()` with no argument returns the version this release writes. Given an evidence table, it returns the version stamped on that table, or NA when the table predates versioning.

Usage

```
evidence_schema_version(x = NULL)
```

Arguments

x	Optional. An evidence data.frame. When supplied, the stamped version of that table is returned instead of the current package version.
---	--

Value

A single version string, or NA_character_ for an unstamped table.

See Also

[migrate_evidence\(\)](#) to upgrade an older table.

Examples

```
evidence_schema_version()

ctx <- suppressMessages(r4sub_run_context("STUDY1", "DEV"))
ev <- suppressMessages(as_evidence(
  data.frame(
    asset_type = "validation", asset_id = "ADSL",
    source_name = "pinnacle21", indicator_id = "SD0001",
    indicator_name = "SD0001", indicator_domain = "quality",
    severity = "high", result = "fail",
    stringsAsFactors = FALSE
  ),
  ctx = ctx
))
evidence_schema_version(ev)
```

evidence_sources	<i>Supported and Planned Evidence Sources</i>
------------------	---

Description

A reference table of the evidence sources R4SUB can ingest, their status, the format they come from, and the pillar they feed. Use it to see at a glance what is wired up today and what is on the roadmap.

Usage

```
evidence_sources()
```

Value

A tibble with columns source, status, format, pillar, and entry_point.

Examples

```
evidence_sources()
```

evidence_summary	<i>Summarize Evidence</i>
------------------	---------------------------

Description

Returns a summary data.frame with counts grouped by domain, severity, result, and source.

Usage

```
evidence_summary(ev)
```

Arguments

ev A valid evidence data.frame.

Value

A data.frame with columns: indicator_domain, severity, result, source_name, and n.

Examples

```
ctx <- suppressMessages(r4sub_run_context("STUDY1", "DEV"))
ev <- suppressMessages(as_evidence(
  data.frame(
    asset_type = "validation", asset_id = "ADSL",
    source_name = "pinnacle21", indicator_id = "SD0001",
    indicator_name = "SD0001", indicator_domain = "quality",
    severity = "high", result = "fail",
    stringsAsFactors = FALSE
  ),
  ctx = ctx
))
evidence_summary(ev)
```

export_evidence	<i>Export Evidence Table to File</i>
-----------------	--------------------------------------

Description

Validates the evidence table then writes it to disk in the requested format. Metadata attributes (exported_at, r4subcore_version, nrow) are attached to the returned path value for traceability.

Usage

```
export_evidence(evidence, file, format = c("csv", "rds", "json"))
```

Arguments

evidence	A valid evidence data.frame (as produced by as_evidence()).
file	Character. Destination file path (including extension).
format	Character. One of "csv", "rds", or "json". Default: "csv".

Value

Invisibly returns file with attributes:

exported_at POSIXct timestamp of export.

r4subcore_version Package version string.

nrow Number of rows written.

Examples

```
ctx <- suppressMessages(r4sub_run_context("STUDY1", "DEV"))
ev <- suppressMessages(as_evidence(
  data.frame(
    asset_type = "validation", asset_id = "ADSL",
    source_name = "pinnacle21", indicator_id = "SD0001",
    indicator_name = "SD0001", indicator_domain = "quality",
    severity = "high", result = "fail",
    stringsAsFactors = FALSE
  ),
  ctx = ctx
))
tmp_csv <- tempfile(fileext = ".csv")
out <- suppressMessages(export_evidence(ev, tmp_csv, format = "csv"))
file.exists(out)
```

hash_id

Generate a Stable Hash ID

Description

Creates a deterministic hash from one or more character inputs, hashed in memory with [rlang::hash\(\)](#).

Usage

```
hash_id(..., prefix = NULL)
```

Arguments

...	Character values to hash together. Concatenated with " ".
prefix	Optional prefix prepended to the hash (e.g., "RUN", "IND").

Value

A character string of the form `prefix-hexhash` or just `hexhash`.

Examples

```
hash_id("ADSL", "rule_001")
hash_id("my_study", "2024-01-01", prefix = "RUN")
```

import_evidence	<i>Import Evidence Table from File</i>
-----------------	--

Description

Reads an evidence table that was previously saved by `export_evidence()`, then validates it against the evidence schema.

Usage

```
import_evidence(file, format = c("csv", "rds", "json"))
```

Arguments

- `file` Character. Path to the file to read.
- `format` Character. One of "csv", "rds", or "json". Default: "csv".

Value

A validated evidence `data.frame` conforming to the evidence schema.

Examples

```
ctx <- suppressMessages(r4sub_run_context("STUDY1", "DEV"))
ev <- suppressMessages(as_evidence(
  data.frame(
    asset_type = "validation", asset_id = "ADSL",
    source_name = "pinnacle21", indicator_id = "SD0001",
    indicator_name = "SD0001", indicator_domain = "quality",
    severity = "high", result = "fail",
    stringsAsFactors = FALSE
  ),
  ctx = ctx
))
tmp_rds <- tempfile(fileext = ".rds")
suppressMessages(export_evidence(ev, tmp_rds, format = "rds"))
ev2 <- suppressMessages(import_evidence(tmp_rds, format = "rds"))
nrow(ev2)
```

json_safely	<i>Safely Serialize to JSON String</i>
-------------	--

Description

Converts an R object to a valid JSON string. Returns "{}" on failure or for NULL/empty inputs.

Usage

```
json_safely(x)
```

Arguments

x	An R object to serialize.
---	---------------------------

Value

A single character string containing valid JSON.

Examples

```
json_safely(list(a = 1, b = "hello"))  
json_safely(NULL)
```

migrate_evidence	<i>Migrate an Evidence Table to a Newer Schema Version</i>
------------------	--

Description

Upgrades an evidence table written by an older release to the requested schema version, applying each intervening migration step in order. A table with no stamp is treated as the earliest known version. A table already at the target is returned unchanged apart from the version stamp.

Usage

```
migrate_evidence(x, to = current_evidence_schema_version())
```

Arguments

x	An evidence data.frame.
to	Target schema version. Defaults to the current version.

Value

The migrated evidence data.frame, stamped with to.

See Also

[evidence_schema_version\(\)](#).

Examples

```
ctx <- suppressMessages(r4sub_run_context("STUDY1", "DEV"))
ev <- suppressMessages(as_evidence(
  data.frame(
    asset_type = "validation", asset_id = "ADSL",
    source_name = "pinnacle21", indicator_id = "SD0001",
    indicator_name = "SD0001", indicator_domain = "quality",
    severity = "high", result = "fail",
    stringsAsFactors = FALSE
  ),
  ctx = ctx
))
ev <- migrate_evidence(ev)
evidence_schema_version(ev)
```

normalize_01	<i>Normalize to 0–1 Range</i>
--------------	-------------------------------

Description

Applies min-max normalization to a numeric vector, optionally clamping values to [0, 1].

Usage

```
normalize_01(x, direction = c("higher_better", "lower_better"), clamp = TRUE)
```

Arguments

x	Numeric vector.
direction	Character. "higher_better" (default) maps max to 1; "lower_better" maps min to 1.
clamp	Logical. If TRUE, clamp output to [0, 1].

Value

Numeric vector normalized to 0–1.

Examples

```
normalize_01(c(10, 20, 30, 40, 50))
normalize_01(c(10, 20, 30), direction = "lower_better")
```

p21_to_evidence	<i>Parse Pinnacle21 Output to Evidence</i>
-----------------	--

Description

Converts a data.frame of Pinnacle21-style validation results into the standard evidence table format. Column names are detected case-insensitively.

Usage

```
p21_to_evidence(
  p21_df,
  ctx,
  asset_type = "validation",
  source_version = NULL,
  default_domain = "quality"
)
```

Arguments

p21_df	A data.frame containing Pinnacle21 validation output. Expected columns (case-insensitive): Rule (or Rule ID), Message, Severity, Dataset, Variable, Result (or Status).
ctx	A r4sub_run_context providing run and study metadata.
asset_type	Character. Asset type label. Default: "validation".
source_version	Character or NULL. Version of the P21 tool.
default_domain	Character. Indicator domain. Default: "quality".

Value

A data.frame conforming to the evidence schema.

Examples

```
p21_raw <- data.frame(
  Rule = c("SD0001", "SD0002"),
  Message = c("Missing variable label", "Invalid format"),
  Severity = c("Error", "Warning"),
  Dataset = c("ADSL", "ADAE"),
  Variable = c("AGE", "AESTDTC"),
  Status = c("Failed", "Warning"),
  stringsAsFactors = FALSE
)
ctx <- r4sub_run_context("STUDY1", "DEV")
ev <- p21_to_evidence(p21_raw, ctx)
```

r4sub_run_context	<i>Create a Run Context</i>
-------------------	-----------------------------

Description

A run context captures metadata for a particular evidence collection run. It provides a unique `run_id`, study identifier, environment label, and timestamps used throughout evidence ingestion.

Usage

```
r4sub_run_context(  
  study_id,  
  environment = c("DEV", "UAT", "PROD"),  
  user = NULL,  
  run_id = NULL,  
  timestamp = Sys.time()  
)
```

Arguments

<code>study_id</code>	Character. Study identifier (e.g., "ABC123").
<code>environment</code>	Character. One of "DEV", "UAT", "PROD".
<code>user</code>	Character or NULL. Username; defaults to system user.
<code>run_id</code>	Character or NULL. If NULL, a unique ID is generated.
<code>timestamp</code>	POSIXct. Defaults to current time.

Value

A list of class `r4sub_run_context` with elements: `run_id`, `study_id`, `environment`, `user`, `created_at`.

Examples

```
ctx <- r4sub_run_context(study_id = "STUDY001", environment = "DEV")  
ctx$run_id  
ctx$study_id
```

read_p21_report	<i>Read a Pinnacle 21 Report from Disk</i>
-----------------	--

Description

Loads a Pinnacle 21 validation report (the export most teams already produce for every submission) into a `data.frame`, ready for `p21_to_evidence()`. CSV exports are read with base R. Excel exports are read with the `readxl` package when it is installed; if it is not, export the report as CSV instead.

Usage

```
read_p21_report(path, format = c("auto", "csv", "xlsx"), sheet = NULL)
```

Arguments

<code>path</code>	Path to the report file.
<code>format</code>	One of "auto" (infer from the file extension), "csv", or "xlsx". Default "auto".
<code>sheet</code>	For Excel reports, the sheet to read. When NULL (the default), a sheet whose name looks like the findings list (matching "detail", "issue", or "finding") is used, otherwise the first sheet.

Details

This function only reads the file; it does not interpret the columns. Pass the result to `p21_to_evidence()`, which detects the rule, message, severity, dataset, variable, and result columns case-insensitively.

Value

A `data.frame` of the report rows with the original column names.

See Also

`p21_to_evidence()` to convert the result into evidence.

Examples

```
# Simulate a small CSV export and read it back.
p21 <- data.frame(
  "Rule ID" = c("SD0001", "SD0002"),
  Message   = c("Missing variable label", "Invalid format"),
  Severity  = c("Error", "Warning"),
  Dataset   = c("ADSL", "ADAE"),
  Variable  = c("AGEU", "AESEQ"),
  Status    = c("Failed", "Warning"),
  check.names = FALSE
)
csv <- tempfile(fileext = ".csv")
utils::write.csv(p21, csv, row.names = FALSE)
```

```
report <- read_p21_report(csv)
report
```

register_indicator	<i>Register an Indicator</i>
--------------------	------------------------------

Description

Adds an indicator definition to the local in-memory registry.

Usage

```
register_indicator(
  indicator_id,
  domain,
  description,
  expected_inputs = character(),
  default_thresholds = numeric(),
  tags = character()
)
```

Arguments

indicator_id	Character. Stable identifier for the indicator.
domain	Character. One of "quality", "trace", "risk", "usability".
description	Character. Human-readable description.
expected_inputs	Character vector. Evidence source types this indicator expects.
default_thresholds	Named numeric vector. Optional thresholds.
tags	Character vector. Optional tags (e.g., "define", "adam").

Value

The indicator definition list, invisibly.

Examples

```
register_indicator(
  indicator_id = "P21-001",
  domain = "quality",
  description = "Required variable is missing from dataset"
)
```

result_to_score	<i>Map Result to Numeric Score</i>
-----------------	------------------------------------

Description

Converts canonical result labels to numeric scores.

Usage

```
result_to_score(result)
```

Arguments

result Character vector of canonical result values (pass, fail, warn, na).

Value

Numeric vector: pass=1, warn=0.5, fail=0, na=NA.

Examples

```
result_to_score(c("pass", "fail", "warn", "na"))
```

severity_to_weight	<i>Map Severity to Numeric Weight</i>
--------------------	---------------------------------------

Description

Converts canonical severity labels to numeric penalty multipliers on a 0–1 scale.

Usage

```
severity_to_weight(severity)
```

Arguments

severity Character vector of canonical severity values (info, low, medium, high, critical).

Details

Default mapping:

- info = 0.00
- low = 0.25
- medium = 0.50
- high = 0.75
- critical = 1.00

Value

Numeric vector of weights.

Examples

```
severity_to_weight(c("low", "high", "critical"))
```

validate_evidence	<i>Validate Evidence Table</i>
-------------------	--------------------------------

Description

Checks that a data.frame conforms to the evidence schema. Verifies column presence, types, and controlled vocabulary values.

Usage

```
validate_evidence(ev)
```

Arguments

ev A data.frame to validate.

Value

TRUE invisibly if valid; throws an error otherwise.

Examples

```
ctx <- suppressMessages(r4sub_run_context("STUDY1", "DEV"))
ev <- suppressMessages(as_evidence(
  data.frame(
    asset_type = "validation", asset_id = "ADSL",
    source_name = "pinnacle21", indicator_id = "SD0001",
    indicator_name = "SD0001", indicator_domain = "quality",
    severity = "high", result = "fail",
    stringsAsFactors = FALSE
  ),
  ctx = ctx
))
validate_evidence(ev)
```

validate_indicator	<i>Validate Indicator Metadata</i>
--------------------	------------------------------------

Description

Checks that an indicator definition list is well-formed.

Usage

```
validate_indicator(indicator)
```

Arguments

indicator	A list with required fields: indicator_id, domain, description. Optional fields: expected_inputs, default_thresholds, tags.
-----------	---

Value

TRUE invisibly if valid; throws an error otherwise.

Examples

```
validate_indicator(list(  
  indicator_id = "P21-001",  
  domain = "quality",  
  description = "Missing required variable"  
))
```

Index

adrg_sections, [2](#)
adrg_sections(), [3](#)
adrg_to_evidence, [3](#)
adrg_to_evidence(), [2](#)
aggregate_indicator_score, [4](#)
as_evidence, [5](#)
as_evidence(), [11](#), [16](#)

bind_evidence, [6](#)

canon_result, [6](#)
canon_severity, [7](#)
check_define_consistency, [8](#)
check_define_consistency(), [9](#)

define_variables, [9](#)
define_variables(), [8](#)
define_xml_to_evidence, [9](#)
define_xml_to_evidence(), [8](#)

evidence_diff, [11](#)
evidence_schema, [12](#)
evidence_schema(), [13](#)
evidence_schema_spec, [13](#)
evidence_schema_version, [13](#)
evidence_schema_version(), [19](#)
evidence_sources, [14](#)
evidence_summary, [15](#)
export_evidence, [15](#)
export_evidence(), [17](#)

hash_id, [16](#)

import_evidence, [17](#)

json_safely, [18](#)

migrate_evidence, [18](#)
migrate_evidence(), [14](#)

normalize_01, [19](#)

p21_to_evidence, [20](#)
p21_to_evidence(), [22](#)

r4sub_run_context, [3](#), [5](#), [8](#), [10](#), [20](#), [21](#)
read_p21_report, [22](#)
register_indicator, [23](#)
result_to_score, [24](#)
result_to_score(), [4](#)
rlang::hash(), [16](#)

severity_to_weight, [24](#)
severity_to_weight(), [4](#)

validate_evidence, [25](#)
validate_indicator, [26](#)