

Package: r4subtrace (via r-universe)

October 11, 2026

Title Traceability Engine for Clinical Submission Readiness

Version 0.3.0

Description Quantifies and explains end-to-end traceability between clinical submission artifacts (ADaM (Analysis Data Model) outputs, derivations, SDTM (Study Data Tabulation Model) sources, specs, code). Builds trace models from metadata and mapping sheets, computes trace levels, and emits standardized R4SUB (Ready for Submission) evidence table rows via 'r4subcore'.

License MIT + file LICENSE

URL <https://r4sub.github.io/r4subtrace/>,
<https://github.com/R4SUB/r4subtrace>

BugReports <https://github.com/R4SUB/r4subtrace/issues>

Depends R (>= 4.2)

Imports cli, dplyr, r4subcore, rlang, stringr, tibble, utils

Suggests igraph, knitr, r4subdata, rmarkdown, testthat (>= 3.0.0),
writexl

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/pak/sysreqs libicu-dev libxml2-dev

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

Date/Publication 2026-09-11 16:27:10 UTC

RemoteUrl <https://github.com/r4sub/r4subtrace>

RemoteRef HEAD

RemoteSha 6c48977eb07c0016cb8331f426c806ff23d57fcf

Contents

build_trace_model	2
compute_trace_levels	3
export_traceability_report	4
plot_trace_graph	5
print.trace_model	6
trace_chains	7
trace_config_default	8
trace_impact	9
trace_indicator_scores	10
trace_model_to_evidence	11
trace_problems	12
traceability_coverage	13
validate_mapping	14
validate_metadata	15
Index	16

build_trace_model	<i>Build a Trace Model</i>
-------------------	----------------------------

Description

Constructs a directed trace model (nodes + edges + diagnostics) from ADaM metadata, SDTM metadata, and an optional mapping sheet.

Usage

```
build_trace_model(  
  adam_meta,  
  sdtm_meta,  
  mapping = NULL,  
  spec = NULL,  
  config = trace_config_default()  
)
```

Arguments

adam_meta	A data.frame of ADaM variable metadata. Must contain dataset and variable columns.
sdtm_meta	A data.frame of SDTM variable metadata. Must contain dataset and variable columns.
mapping	An optional data.frame describing ADaM-to-SDTM mappings. Must contain adam_dataset, adam_var, sdtm_domain, sdtm_var.
spec	Reserved for future use (ADaM spec ingestion).
config	A trace_config object from trace_config_default() .

Value

A list of class "trace_model" with elements:

- nodes: tibble of asset nodes (datasets and variables)
- edges: tibble of relationships between nodes
- diagnostics: list of tibbles (orphans, ambiguities, conflicts)
- config: the configuration used

Examples

```
adam_meta <- data.frame(
  dataset = "ADSL", variable = c("STUDYID", "USUBJID", "AGE"),
  label = c("Study ID", "Unique Subject ID", "Age")
)
sdm_meta <- data.frame(
  dataset = "DM", variable = c("STUDYID", "USUBJID", "AGE"),
  label = c("Study ID", "Unique Subject ID", "Age")
)
map <- data.frame(
  adam_dataset = "ADSL", adam_var = c("STUDYID", "USUBJID", "AGE"),
  sdm_domain = "DM", sdm_var = c("STUDYID", "USUBJID", "AGE")
)
tm <- build_trace_model(adam_meta, sdm_meta, mapping = map)
tm$nodes
tm$edges
```

compute_trace_levels *Compute Trace Levels for ADaM Variables*

Description

Assigns a traceability level (L0–L3) to each ADaM variable in the trace model based on available mapping, derivation text, and confidence scores.

Usage

```
compute_trace_levels(trace_model)
```

Arguments

trace_model A trace_model object from `build_trace_model()`.

Details

Trace levels:

- **L0:** No mapping and no derivation text.
- **L1:** Derivation text present but no SDTM mapping.
- **L2:** Mapping to SDTM variable/domain exists.
- **L3:** Mapping exists AND (confidence $\geq$ threshold OR derivation text present alongside mapping).

Value

A tibble with columns: adam_dataset, adam_var, trace_level, has_mapping, has_derivation_text, n_candidates, max_confidence.

Examples

```
adam_meta <- data.frame(
  dataset = "ADSL", variable = c("STUDYID", "USUBJID", "AGE", "AGEGR1"),
  label = c("Study ID", "Unique Subject ID", "Age", "Age Group")
)
sdm_meta <- data.frame(
  dataset = "DM", variable = c("STUDYID", "USUBJID", "AGE"),
  label = c("Study ID", "Unique Subject ID", "Age")
)
map <- data.frame(
  adam_dataset = "ADSL", adam_var = c("STUDYID", "USUBJID", "AGE"),
  sdm_domain = "DM", sdm_var = c("STUDYID", "USUBJID", "AGE"),
  confidence = c(1.0, 1.0, 0.9)
)
tm <- build_trace_model(adam_meta, sdm_meta, mapping = map)
compute_trace_levels(tm)
```

export_traceability_report

Export a Traceability Coverage Report

Description

Produces the coverage table an ADRG appendix needs: per-dataset traceability percentages, the untraced variables, and an overall summary row. The report can be returned as a data.frame or written to CSV or Excel for inclusion in a submission package.

Usage

```
export_traceability_report(
  trace_model,
  format = c("data.frame", "csv", "xlsx"),
  path = NULL,
  traced_min_level = 2L
)
```

Arguments

`trace_model` A `trace_model` from `build_trace_model()`.

`format` One of "data.frame", "csv", or "xlsx".

`path` Output file path, required for "csv" and "xlsx".

`traced_min_level` The minimum trace level that counts as traced. Passed to `traceability_coverage()`.

Value

For "data.frame", the report tibble. For "csv" and "xlsx", the report is written to path and the path is returned invisibly.

Examples

```
adam_meta <- data.frame(
  dataset = "ADSL", variable = c("STUDYID", "AGEGR1"),
  label = c("Study ID", NA)
)
sdm_meta <- data.frame(dataset = "DM", variable = "STUDYID")
map <- data.frame(
  adam_dataset = "ADSL", adam_var = "STUDYID",
  sdm_domain = "DM", sdm_var = "STUDYID", confidence = 1
)
tm <- build_trace_model(adam_meta, sdm_meta, mapping = map)
export_traceability_report(tm)
```

plot_trace_graph	<i>Plot a Derivation Graph</i>
------------------	--------------------------------

Description

Draws the derivation graph behind `trace_chains()` using `igraph`: ADaM and SDTM variables as nodes, derivation links as directed edges.

Usage

```
plot_trace_graph(trace_model, adam_derivations = NULL, ...)
```

Arguments

`trace_model` A `trace_model` from `build_trace_model()`.
`adam_derivations` Optional ADaM-to-ADaM derivations, as in `trace_chains()`.
`...` Passed to `igraph::plot.igraph()`.

Value

The `igraph` object, invisibly.

Examples

```
## Not run:
plot_trace_graph(tm, adam_derivations = deriv)

## End(Not run)
```

<code>print.trace_model</code>	<i>Print Trace Model</i>
--------------------------------	--------------------------

Description

Print Trace Model

Usage

```
## S3 method for class 'trace_model'
print(x, ...)
```

Arguments

`x` A `trace_model` object.
`...` Ignored.

trace_chains

*Follow Multi-Step Derivation Chains***Description**

Traces each ADaM variable back to its ultimate source through chains of any depth, not just a single ADaM-to-SDTM hop. Real submissions often derive an ADaM variable from another ADaM variable that is itself derived from SDTM (for example an ADAE variable built from ADSL, which in turn comes from DM). Shallow tracing misses those links and understates coverage.

Usage

```
trace_chains(trace_model, adam_derivations = NULL, trace_depth = 3L)
```

Arguments

`trace_model` A trace_model from `build_trace_model()`.

`adam_derivations` Optional data.frame of ADaM-to-ADaM derivations with columns `from_dataset`, `from_var`, `to_dataset`, `to_var`.

`trace_depth` Maximum number of hops to follow, a positive integer. Default 3.

Details

The ADaM-to-SDTM links come from the trace model's edges. ADaM-to-ADaM links are supplied through `adam_derivations`, a data.frame with columns `from_dataset`, `from_var`, `to_dataset`, `to_var`, where `from` is the derived variable and `to` is its source. Each ADaM variable is walked toward its sources until the path reaches an SDTM variable (grounded), dead-ends at an ADaM variable with no further source (ungrounded), revisits a node (a cycle), or reaches `trace_depth` hops.

Value

A tibble with one row per ADaM variable: `adam_dataset`, `adam_var`, `max_depth` (longest chain in hops), `grounded` (a path reaches SDTM), `terminal_source`, `has_cycle`, `truncated` (a path hit `trace_depth`), `n_paths`, and `longest_chain` (a readable path string).

Examples

```
adam <- data.frame(
  dataset = c("ADSL", "ADAE"), variable = c("AGE", "AAGE"),
  label = c("Age", "Analysis Age")
)
sdm <- data.frame(dataset = "DM", variable = "AGE", label = "Age")
map <- data.frame(
  adam_dataset = "ADSL", adam_var = "AGE",
  sdm_domain = "DM", sdm_var = "AGE", confidence = 1
)
```

```
tm <- build_trace_model(adam, sdtm, mapping = map)
deriv <- data.frame(
  from_dataset = "ADAE", from_var = "AAGE",
  to_dataset = "ADSL", to_var = "AGE"
)
trace_chains(tm, adam_derivations = deriv)
```

trace_config_default *Default Trace Configuration*

Description

Returns a list of default configuration values for trace model building and evidence emission.

Usage

```
trace_config_default(
  severity_by_level = c(L0 = "high", L1 = "medium", L2 = "low", L3 = "info"),
  result_by_level = c(L0 = "fail", L1 = "warn", L2 = "warn", L3 = "pass"),
  confidence_threshold_L3 = 0.8,
  uppercase_datasets = TRUE
)
```

Arguments

severity_by_level
Named character vector mapping trace levels to severity.

result_by_level
Named character vector mapping trace levels to result.

confidence_threshold_L3
Numeric threshold for L3 classification. A mapping must have confidence $\geq$ this value to qualify for L3.

uppercase_datasets
Logical; if TRUE, dataset and domain names are uppercased during canonicalization.

Value

A list of class "trace_config" with elements: severity_by_level, result_by_level, confidence_threshold_L3, uppercase_datasets.

Examples

```
cfg <- trace_config_default()
cfg$severity_by_level

# Override a single setting
cfg2 <- trace_config_default(confidence_threshold_L3 = 0.9)
```

trace_impact

Impact Analysis for a Changed Source Variable

Description

Answers the question a data team asks before touching a source variable: if this changes, what downstream ADaM variables are affected? Given a trace model and a variable that is about to change, `trace_impact()` follows the derivation graph in reverse and returns every variable that derives from it, directly or through a chain.

Usage

```
trace_impact(trace_model, changed, adam_derivations = NULL, max_depth = Inf)
```

Arguments

<code>trace_model</code>	A <code>trace_model</code> from <code>build_trace_model()</code> .
<code>changed</code>	A single variable identifier as "DATASET.VARIABLE", for example "DM.AGE" or "ADSL.AGE".
<code>adam_derivations</code>	Optional data.frame of ADaM-to-ADaM derivations with columns <code>from_dataset</code> , <code>from_var</code> , <code>to_dataset</code> , <code>to_var</code> , as in <code>trace_chains()</code> .
<code>max_depth</code>	Optional maximum number of hops to report. <code>Inf</code> (the default) reports the full transitive set.

Details

The trace model records derivations as edges from a derived variable to its source. Impact analysis walks those edges backwards from the changed variable to find its dependents. ADaM-to-ADaM derivations are included when supplied through `adam_derivations`, exactly as in `trace_chains()`, so multi-step impact (an ADTTE variable built from an ADSL variable built from DM) is captured. Each dependent is reported once, at its shortest distance from the change, with a readable path back to it. Circular references are handled and do not loop.

Value

A tibble with one row per impacted variable: `dataset`, `variable`, `role`, `depth` (hops from the change), and `path` (a readable derivation path from the impacted variable back to the changed one). Rows are ordered nearest first. Zero rows means nothing derives from the variable. The result carries a "changed" attribute with the resolved identifier.

Examples

```
adam <- data.frame(
  dataset = c("ADSL", "ADAE"), variable = c("AGE", "AAGE"),
  label = c("Age", "Analysis Age")
)
```

```

sdm <- data.frame(dataset = "DM", variable = "AGE", label = "Age")
map <- data.frame(
  adam_dataset = "ADSL", adam_var = "AGE",
  sdm_domain = "DM", sdm_var = "AGE", confidence = 1
)
tm <- build_trace_model(adam, sdm, mapping = map)
deriv <- data.frame(
  from_dataset = "ADAE", from_var = "AAGE",
  to_dataset = "ADSL", to_var = "AGE"
)
# If DM.AGE changes, what is affected?
trace_impact(tm, "DM.AGE", adam_derivations = deriv)

```

trace_indicator_scores

Compute Trace Indicator Scores

Description

Computes summary metrics from evidence rows generated by [trace_model_to_evidence\(\)](#). Returns key traceability indicators.

Usage

```
trace_indicator_scores(evidence)
```

Arguments

evidence A data.frame of evidence rows (must contain `indicator_id` and `metric_value` columns).

Value

A tibble with columns: `indicator`, `value`, `description`.

Examples

```

library(r4subcore)
ctx <- r4sub_run_context(study_id = "TEST001", environment = "DEV")
adam_meta <- data.frame(
  dataset = "ADSL", variable = c("STUDYID", "AGE", "AGEGR1"),
  label = c("Study ID", "Age", "Age Group")
)
sdm_meta <- data.frame(
  dataset = "DM", variable = c("STUDYID", "AGE"),
  label = c("Study ID", "Age")
)
map <- data.frame(

```

```

adam_dataset = "ADSL", adam_var = c("STUDYID", "AGE"),
sdtm_domain = "DM", sdtm_var = c("STUDYID", "AGE")
)
tm <- build_trace_model(adam_meta, sdtm_meta, mapping = map)
ev <- trace_model_to_evidence(tm, ctx = ctx)
trace_indicator_scores(ev)

```

trace_model_to_evidence

Convert Trace Model to R4SUB Evidence

Description

Emits evidence rows compatible with `r4subcore::validate_evidence()` for each ADaM variable's trace level, plus diagnostic rows for orphans, ambiguities, and conflicts.

Usage

```

trace_model_to_evidence(
  trace_model,
  ctx,
  source_name = "r4subtrace",
  source_version = NULL
)

```

Arguments

<code>trace_model</code>	A <code>trace_model</code> object from <code>build_trace_model()</code> .
<code>ctx</code>	An <code>r4sub_run_context</code> from <code>r4subcore::r4sub_run_context()</code> .
<code>source_name</code>	Character; the name of the evidence source.
<code>source_version</code>	Character or NULL; version of the source.

Value

A data.frame of evidence rows passing `r4subcore::validate_evidence()`.

Examples

```

library(r4subcore)
ctx <- r4sub_run_context(study_id = "TEST001", environment = "DEV")
adam_meta <- data.frame(
  dataset = "ADSL", variable = c("STUDYID", "AGE"),
  label = c("Study ID", "Age")
)
sdtm_meta <- data.frame(
  dataset = "DM", variable = c("STUDYID", "AGE"),
  label = c("Study ID", "Age")
)

```

```

)
map <- data.frame(
  adam_dataset = "ADSL", adam_var = c("STUDYID", "AGE"),
  sdtm_domain = "DM",   sdtm_var = c("STUDYID", "AGE")
)
tm <- build_trace_model(adam_meta, sdtm_meta, mapping = map)
ev <- trace_model_to_evidence(tm, ctx = ctx)
r4subcore::validate_evidence(ev)

```

trace_problems

The Traceability Problems Worth Acting On

Description

A full coverage table lists every variable; on a real submission with tens of thousands of variables that is not what a team consumes. This returns only the problems: variables that are not traced to the required level, and, when `adam_derivations` is supplied, derivation chains that never reach SDTM, that are circular, or that were cut off at the depth limit. One variable can appear more than once when it has more than one kind of problem.

Usage

```

trace_problems(
  trace_model,
  traced_min_level = 2L,
  adam_derivations = NULL,
  trace_depth = 3L
)

```

Arguments

<code>trace_model</code>	A <code>trace_model</code> from <code>build_trace_model()</code> .
<code>traced_min_level</code>	The minimum trace level that counts as traced, an integer in 0:3. Default 2.
<code>adam_derivations</code>	Optional ADaM-to-ADaM derivation table (see <code>trace_chains()</code>). When supplied, chain problems are included.
<code>trace_depth</code>	Maximum chain depth to follow, passed to <code>trace_chains()</code> .

Value

A tibble with one row per problem: `adam_dataset`, `adam_var`, `problem` ("untraced", "ungrounded", "cycle", or "truncated"), and a detail string. Empty when nothing is wrong.

See Also

`traceability_coverage()` for the full per-dataset rollout, `trace_chains()` for the chains themselves.

Examples

```
adam_meta <- data.frame(
  dataset = "ADSL", variable = c("USUBJID", "AGE", "DERIVED1"),
  label = c("Unique Subject ID", "Age", "Derived")
)
sdm_meta <- data.frame(dataset = "DM", variable = c("USUBJID", "AGE"))
map <- data.frame(
  adam_dataset = "ADSL", adam_var = c("USUBJID", "AGE"),
  sdm_domain = "DM", sdm_var = c("USUBJID", "AGE"), confidence = c(1, 1)
)
tm <- build_trace_model(adam_meta, sdm_meta, mapping = map)
trace_problems(tm)
```

traceability_coverage *Traceability Coverage by ADaM Dataset*

Description

Summarizes how much of each ADaM dataset is traced back to SDTM. A variable counts as traced when its trace level reaches `traced_min_level`. The default of 2 treats an existing SDTM mapping (level L2) as traced.

Usage

```
traceability_coverage(trace_model, traced_min_level = 2L)
```

Arguments

`trace_model` A `trace_model` from `build_trace_model()`.
`traced_min_level` The minimum trace level that counts as traced, an integer in 0:3. Default 2.

Value

A tibble with one row per ADaM dataset: `adam_dataset`, `n_variables`, `n_traced`, `n_untraced`, `pct_traced`, and `untraced_vars` (a comma-separated list, empty when the dataset is fully traced).

Examples

```
adam_meta <- data.frame(
  dataset = "ADSL", variable = c("STUDYID", "USUBJID", "AGEGR1"),
  label = c("Study ID", "Unique Subject ID", NA)
)
sdm_meta <- data.frame(dataset = "DM", variable = c("STUDYID", "USUBJID"))
map <- data.frame(
  adam_dataset = "ADSL", adam_var = c("STUDYID", "USUBJID"),
  sdm_domain = "DM", sdm_var = c("STUDYID", "USUBJID"),
  confidence = c(1, 1)
)
```

```
)  
tm <- build_trace_model(adam_meta, sdtm_meta, mapping = map)  
traceability_coverage(tm)
```

validate_mapping	<i>Validate Trace Mapping</i>
------------------	-------------------------------

Description

Checks that a mapping data.frame contains the required columns (adam_dataset, adam_var, sdtm_domain, sdtm_var) and canonicalizes names, trims whitespace, and optionally uppercases dataset/domain names.

Usage

```
validate_mapping(df, uppercase_datasets = TRUE)
```

Arguments

df	A data.frame describing ADaM-to-SDTM variable mappings.
uppercase_datasets	Logical; if TRUE, uppercases adam_dataset and sdtm_domain. Default TRUE.

Value

A tibble with canonicalized column names and values.

Examples

```
map <- data.frame(  
  ADAM_DATASET = "adsl", ADAM_VAR = "AGE",  
  SDTM_DOMAIN = "dm", SDTM_VAR = "AGE"  
)  
validate_mapping(map)
```

validate_metadata	<i>Validate Dataset Metadata</i>
-------------------	----------------------------------

Description

Checks that an ADaM or SDTM metadata data.frame contains the required columns (dataset, variable) and canonicalizes column names to lowercase.

Usage

```
validate_metadata(df, kind = c("adam", "sdm"))
```

Arguments

df	A data.frame of dataset metadata.
kind	Character; "adam" or "sdm". Used in error messages only.

Value

A tibble with canonicalized column names.

Examples

```
meta <- data.frame(DATASET = "ADSL", VARIABLE = "SUBJID", LABEL = "Subject ID")
validate_metadata(meta, kind = "adam")
```

Index

`build_trace_model`, [2](#)
`build_trace_model()`, [3](#), [5–7](#), [9](#), [11–13](#)

`compute_trace_levels`, [3](#)

`export_traceability_report`, [4](#)

`igraph::plot.igraph()`, [6](#)

`plot_trace_graph`, [5](#)
`print.trace_model`, [6](#)

`trace_chains`, [7](#)
`trace_chains()`, [5](#), [6](#), [9](#), [12](#)
`trace_config_default`, [8](#)
`trace_config_default()`, [2](#)
`trace_impact`, [9](#)
`trace_indicator_scores`, [10](#)
`trace_model_to_evidence`, [11](#)
`trace_model_to_evidence()`, [10](#)
`trace_problems`, [12](#)
`traceability_coverage`, [13](#)
`traceability_coverage()`, [5](#), [12](#)

`validate_mapping`, [14](#)
`validate_metadata`, [15](#)