

Package: siera (via r-universe)

October 15, 2026

Type Package

Title Generate Analysis Results Programmes Using ARS Metadata

Version 0.5.6

Description Analysis Results Standard (ARS), a foundational standard by CDISC (Clinical Data Interchange Standards Consortium), provides a logical data model for metadata describing all components to calculate Analysis Results.
<<https://www.cdisc.org/standards/foundational/analysis-results-standard>>
Using 'siera', ARS metadata is ingested (JSON format), producing R programmes to generate Analysis Results Datasets (ARDs). Supports arbitrary-depth table structures, multi-value conditions, data-driven groupings, and CDISC-compliant ARD column stamping.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1)

Imports magrittr, dplyr, tibble, tidyr, jsonlite, stringr, readxl, cli

Suggests knitr, rmarkdown, testthat (>= 3.0.0), withr, readr, haven, cards, cardx, datasetjson, covr, parameters, broom, broom.helpers, openxlsx

VignetteBuilder knitr

URL <https://clymbclinical.github.io/siera/>,
<https://github.com/clymbclinical/siera>

BugReports <https://github.com/clymbclinical/siera/issues>

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libicu-dev

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

Date/Publication 2026-10-15 08:39:41 UTC

RemoteUrl <https://github.com/clymbclinical/siera>

RemoteRef HEAD
RemoteSha 9208a9b6a00876f13d2b17f01aa318290c066674

Contents

ARD_script_example	2
ARS_example	3
ars_grouping	3
ars_stamp	4
ars_xlsx_to_json	6
method_library	7
readARS	8
Index	10

ARD_script_example	<i>Get path to example ARD scripts, auto-generated by siera</i>
--------------------	---

Description

siera comes bundled with some example ARD-producing scripts in its ‘inst/script’ directory. This function make them easy to access.

Usage

```
ARD_script_example(path = NULL)
```

Arguments

path Name of R file. If ‘NULL’, the example files will be listed.

Value

A list of example R files (if path is NULL), or a file itself if path is used.

Examples

```
ARD_script_example()  
ARD_script_example("ARD_Out14-1-1.R")
```

ARS_example	<i>Get path to ARS example files</i>
-------------	--------------------------------------

Description

siera comes bundled with some example files in its ‘inst/extdata’ directory. This function make them easy to access.

Usage

```
ARS_example(path = NULL)
```

Arguments

path	Name of file. If ‘NULL’, the example files will be listed.
------	--

Value

A list of example files (if path is NULL), or a file itself if path is used.

Examples

```
ARS_example()
ARS_example("Common_Safety_Displays_cards.json")
```

ars_grouping	<i>Describe one ARS analysis grouping for [ars_stamp()]</i>
--------------	---

Description

Builds a small, validated description of one entry of an analysis’ ‘orderedGroupings’, so that [ars_stamp()] can add the CDISC ARD columns ‘group<n>_groupingId’, ‘group<n>_groupId’ and (for data-driven groupings) ‘group<n>_groupValue’. siera calls this function in the scripts it generates, with the values copied from the ARS metadata - each argument maps to one ARS element:

Usage

```
ars_grouping(id, groups = NULL, data_driven = FALSE)
```

Arguments

id	Grouping id: a single, non-empty string (‘analyses[].orderedGroupings[].groupingId’).
groups	Optional named character vector describing the pre-defined groups of a non-data-driven grouping. Names are group ids (‘analysisGroupings[].groups[].id’), values the matching condition values (‘analysisGroupings[].groups[].condition.value’). ‘NULL’ (default) means the grouping defines no groups.
data_driven	Single logical, ‘analysisGroupings[].dataDriven’. Defaults to ‘FALSE’.

Details

Argument	ARS JSON path	Notes
'id'	'analyses[].orderedGroupings[].groupingId' (= 'analysisGroupings[].id')	Written to 'group<n>_groupingId'
'groups'	'analysisGroupings[].groups[]'	Names are the group ids ('groups')
'data_driven'	'analysisGroupings[].dataDriven'	When 'TRUE' the groups are discovered

Value

An object of class 'siera_ars_grouping', to be supplied in the 'groupings' list of [ars_stamp()].

See Also

[ars_stamp()]

Examples

```
# A pre-defined grouping (e.g. treatment arms)
ars_grouping(
  "AnlsGrouping_01_Trt",
  groups = c(
    "AnlsGrouping_01_Trt_1" = "Placebo",
    "AnlsGrouping_01_Trt_2" = "Xanomeline Low Dose"
  )
)

# A data-driven grouping (categories are discovered from the data)
ars_grouping("AnlsGrouping_04_AEBODSYS", data_driven = TRUE)
```

ars_stamp	<i>Link ARS identifiers to an analysis' ARD</i>
-----------	---

Description

The last step of every analysis in a siera-generated ARD script: it takes the ARD that the analysis method computed (typically a **cards** object) and links it back to the ARS metadata by adding the CDISC ARD identifier columns:

Usage

```
ars_stamp(ard, analysis_id, method_id, output_id, groupings = list())
```

Arguments

ard	The ARD computed by the analysis method: a data frame (typically a cards object), or 'NULL' when the analysis had no data.
analysis_id	Analysis id: a single string ('analyses[].id').
method_id	Method id: a single string ('analyses[].methodId').
output_id	Output id: a single string (the 'outputId' of the 'mainListOfContents' list item the analysis belongs to).
groupings	A list of [ars_grouping()] objects ('analyses[].orderedGroupings[]'). The position in the list is the ARD group number: the first grouping is stamped onto 'group1_*', the second onto 'group2_*', and so on. Each needs the matching 'group<n>_level' column in 'ard'. Default: no groupings.

Details

* 'AnalysisId', 'MethodId' and 'OutputId'; * for each grouping, 'group<n>_groupingId' plus either 'group<n>_groupId' (pre-defined groups, looked up from the 'group<n>_level' column) or 'group<n>_groupValue' (data-driven groupings); * all '*_level' columns are coerced to character, so ARDs of different analyses can be row-bound safely.

When 'ard' is 'NULL' - the analysis had no data, so the method was skipped - a statistics-free stub row carrying only 'AnalysisId', 'MethodId' and 'OutputId' is returned.

siera calls this function in the scripts it generates, with the values copied from the ARS metadata - each argument maps to one ARS element:

Argument	ARS JSON path	ARD column(s)
'ard'	result of the analysis method ('methods[].codeTemplate')	(input)
'analysis_id'	'analyses[].id'	'AnalysisId'
'method_id'	'analyses[].methodId'	'MethodId'
'output_id'	'mainListOfContents' list item 'outputId'	'OutputId'
'groupings'	'analyses[].orderedGroupings[]'	'group<n>_groupingId', 'group<n>_groupId', 'gr

Value

'ard' with the identifier columns added (a 'data.frame' stub when 'ard' is 'NULL').

See Also

[ars_grouping()], [readARS()]

Examples

```
# A tiny ARD, as an analysis method might return it
ard <- tibble::tibble(
  group1_level = c("Placebo", "Xanomeline Low Dose"),
  stat_name = "n",
  stat = list(86L, 84L)
)
```

```

ars_stamp(
  ard,
  analysis_id = "An_01",
  method_id = "Mth_01",
  output_id = "Out_01",
  groupings = list(
    ars_grouping(
      "AnlsGrouping_01_Trt",
      groups = c(
        "AnlsGrouping_01_Trt_1" = "Placebo",
        "AnlsGrouping_01_Trt_2" = "Xanomeline Low Dose"
      )
    )
  )
)

# No data: a stub row identifying the analysis is returned
ars_stamp(NULL, "An_01", "Mth_01", "Out_01")

```

ars_xlsx_to_json

Convert an ARS Excel workbook to ARS JSON

Description

****Deprecated.**** siera is moving to JSON-only ARS metadata, and both ‘.xlsx’ input to [readARS()] and this converter will be removed in a future release. Use it now to convert each workbook once, then keep the ‘.json’ file as the source of truth for your ARS metadata. Every call warns with class ‘siera_deprecated_xlsx’.

Converts a CDISC Analysis Results Standard (ARS) metadata workbook (‘.xlsx’) into an ARS ReportingEvent JSON file. This is an R-native reimplementation of CDISC’s Python ‘excel2ars.py’ utility and performs a faithful whole-workbook conversion: every ARS worksheet is mapped, not only the sheets that [readARS()] itself consumes.

Usage

```
ars_xlsx_to_json(xlsx_path, json_path = NULL)
```

Arguments

xlsx_path	Path to the ARS Excel workbook (‘.xlsx’).
json_path	Optional path for the output JSON file. When ‘NULL’ (the default), the JSON is written beside ‘xlsx_path’ with the same base name and a ‘.json’ extension.

Details

The converter maps the full ARS object model: ‘about’, ‘studyInfo’, the main and other lists of contents, reference documents, terminology extensions, analysis output categorizations, analysis sets, analysis groupings, data subsets, methods (with operations and code templates), analyses, global display sections and outputs (with displays).

Multi-value cells (for example the values of an ‘IN’ condition, or ‘categoryIds’) are split on ‘|’, with or without surrounding spaces, and each value is trimmed. Commas are never treated as separators. Multi-value ARS fields are always written as JSON arrays, even when they hold one value.

Note that the ARS spec evolves; this converter targets the worksheet layout emitted by TFL Designer and the CDISC ARS template and may need updating for future ARS versions.

Value

The path to the written JSON file, invisibly.

Examples

```
json <- suppressWarnings(
  ars_xlsx_to_json(ARS_example("exampleARS_6.xlsx"),
    json_path = tempfile(fileext = ".json"))
)
readARS(json, output_path = tempdir(), adam_path = tempdir())
```

method_library

Browse the bundled analysis-method template library

Description

siera ships a plain-text library of analysis-method code templates (the ‘templateCode’ + parameters + valueSource recipes that drive ARD generation) under ‘inst/method-library’. This accessor lists the available methods and resolves the on-disk location of a method’s files, so you do not have to call [system.file()] directly. Each method’s stable ‘id’ is the key intended for an ARS file to reference.

Usage

```
method_library(id = NULL)
```

Arguments

id	Method id (e.g. “risk_difference”). If ‘NULL’ (default), the available method ids are returned.
----	---

Details

The human-readable catalog (‘METHODS.md’) and the valueSource registry (‘constructs.json’) live at the library root, i.e. ‘system.file(“method-library”, package = “siera”)’.

Value

When 'id' is 'NULL', a character vector of available method ids. Otherwise, the path to that method's directory (containing 'method.json' and 'template.R').

Examples

```
method_library()
method_library("risk_difference")
```

readARS	<i>Ingest ARS (Analysis Results Standard) metadata, produce ARD (Analysis Results Dataset) code for each output</i>
---------	---

Description

Ingest ARS (Analysis Results Standard) metadata, and meta-programme R scripts that could be run as-is to produce Analysis Results Datasets when ingesting ADaM datasets

Usage

```
readARS(
  ARS_path,
  output_path = tempdir(),
  adam_path = tempdir(),
  spec_output = "",
  output_format = "none",
  code_style = "wrapped",
  column_labels = NULL
)
```

Arguments

ARS_path	A file containing ARS metadata for a reporting event
output_path	Path to store .R ARD scripts
adam_path	Path to folder containing ADaM datasets, to be run in ARD program. Datasets may be supplied as CSV ('.csv'), SAS transport ('.xpt') or CDISC Dataset-JSON ('.json') files. <i>siera</i> chooses the reader for each dataset from the file extension found in this folder — '.csv' files are read with <code>readr::read_csv()</code> , '.xpt' files with <code>haven::read_xpt()</code> and '.json' files with <code>datasetjson::read_dataset_json()</code> — so no extra argument is required. Reading '.xpt' datasets in the generated script requires the haven package, and '.json' datasets the datasetjson package. When several formats exist for the same dataset, precedence is '.csv', then '.xpt', then '.json'.
spec_output	The output ID for a specific output to be run from the metadata

output_format	Format for emitting the generated ARD. Must be exactly one of "none" or "datasetjson" (no partial matching). "none" (default) generates the ARD scripts only. "datasetjson" additionally appends code that writes each ARD as a CDISC Dataset-JSON file ('ARD_<OutputId>.json') when the generated script is run. The Dataset-JSON export requires the optional datasetjson package to be installed in the environment that runs the generated script.
code_style	Style of the generated analysis code. Must be exactly one of "wrapped" or "expanded" (no partial matching). "wrapped" (default) ends each analysis with a call to <code>siera::ars_stamp()</code> , which links the analysis' ARD to the ARS metadata ('AnalysisId', 'MethodId', 'OutputId' and the 'group[n]_*' identifiers), so every analysis reads as analysis set, data subset, method, identifier linking. "expanded" writes all of that linking out as plain dplyr code in the script, so the generated script does not call siera functions for the linking step. Both styles produce an identical ARD.
column_labels	Optional named character vector of Dataset-JSON variable labels, e.g. 'c(AGEGR1 = "Pooled Age Group 1", TRT01A = "Actual Treatment")'. Names are ARD column names and values are the labels to write. Overrides take precedence over siera's built-in label dictionary, so they can also reword built-in labels. Because the ARD's columns are only known once the generated script runs, a name that matches no ARD column triggers a warning at script runtime (likely a typo) rather than an error. Only used when 'output_format = "datasetjson"'; otherwise it is ignored with a warning.

Value

R programmes generating ARDs - one for each output (or analysis from an output) specified in the ARS metadata

Examples

```
# path to file containing ARS metadata
ARS_path <- ARS_example("Common_Safety_Displays_cards.json")

# output path for R programs
output_dir <- tempdir()

# folder containing ADaM datasets
adam_folder <- tempdir()

# run function, write to temp directory
readARS(ARS_path, output_dir, adam_folder)
```

Index

ARD_script_example, [2](#)

ARS_example, [3](#)

ars_grouping, [3](#)

ars_stamp, [4](#)

ars_xlsx_to_json, [6](#)

method_library, [7](#)

readARS, [8](#)