Package ‘airr’

August 29, 2023

Type Package

Version 1.5.0

Date 2023-08-29

Title AIRR Data Representation Reference Library

Description Schema definitions and read, write and validation tools for data formatted in accordance with the AIRR Data Representation schemas defined by the AIRR Community <`http://docs.airr-community.org`>.

License CC BY 4.0

URL `http://docs.airr-community.org`

BugReports `https://github.com/airr-community/airr-standards/issues`

BuildVignettes true

VignetteBuilder knitr

Encoding UTF-8

Depends R (>= 3.1.2)

Imports jsonlite, methods, readr, stats, stringi, tools, yaml

Suggests knitr, rmarkdown, tibble, testthat

RoxygenNote 7.2.3

NeedsCompilation no

Author Jason Vander Heiden [aut, cre],
Susanna Marquez [aut],
Scott Christley [aut],
Katharina Imkeller [aut],
Ulrik Stervbo [aut],
AIRR Community [cph]

Maintainer Jason Vander Heiden <`jason.vanderheiden@gmail.com`>

Repository CRAN

Date/Publication 2023-08-29 18:50:02 UTC

R topics documented:

ExampleData 2
load_schema 3
read_airr 4
read_tabular 5
Schema-class 6
validate_airr 8
validate_tabular 9
write_airr 10
write_tabular 11

Index **12**

`ExampleData` *Example AIRR data*

Description

Example data files compliant with the the AIRR Data Representation standards.

Format

- `extdata/rearrangement-example.tsv.gz`: Rearrangement TSV file.
- `extdata/repertoire-example.yaml`: Repertoire YAML file.
- `extdata/germline-example.json`: GermlineSet and GenotypeSet JSON file.

Examples

```
# Load Rearrangement example
file <- system.file("extdata", "rearrangement-example.tsv.gz", package="airr")
rearrangement <- read_rearrangement(file)

# Load Repertoire example
file <- system.file("extdata", "repertoire-example.yaml", package="airr")
repertoire <- read_airr(file)

# Load GermlineSet and GenotypeSet examples
file <- system.file("extdata", "germline-example.json", package="airr")
germline <- read_airr(file)
```

`load_schema` *Load a schema definition*

Description

`load_schema` loads an AIRR object definition from the internal definition set.

Usage

```
load_schema(definition)
```

Arguments

`definition`	name of the schema definition.

Details

Valid definitions include:

- `"Rearrangement"`
- `"Alignment"`
- `"Repertoire"`
- `"Study"`
- `"Subject"`
- `"Diagnosis"`
- `"Sample"`
- `"SampleProcessing"`
- `"DataProcessing"`
- `"GermlineSet"`
- `"GenotypeSet"`

Value

A Schema object for the `definition`.

See Also

See Schema for the return object.

Examples

```
# Load the Rearrangement definition
schema <- load_schema("Rearrangement")

# Load the Repertoire definition
schema <- load_schema("Repertoire")
```

`read_airr` *Read an AIRR Data Model file in YAML or JSON format*

Description

`read_airr` loads a YAML or JSON file containing AIRR Data Model records.

Usage

```
read_airr(
  file,
  format = c("auto", "yaml", "json"),
  validate = TRUE,
  model = TRUE
)
```

Arguments

`file`	path to the input file.
`format`	format of the input file. Must be one of `"auto"`, `"yaml"`, or `"json"`. If `"auto"` (default), the format will be detected from the `file` extension.
`validate`	run schema validation if `TRUE`.
`model`	if `TRUE` validate only AIRR DataFile defined objects. If `FALSE` attempt validation of all objects in `data`. Ignored if `validate=FALSE`

Value

A named nested `list` contained in the AIRR Data Model with the top-level names reflecting the individual AIRR objects.

See Also

See Schema for the AIRR schema definition objects. See write_airr for writing AIRR Data Model records in YAML or JSON format.

Examples

```
# Get path to the Reportoire and GermlineSet example files
f1 <- system.file("extdata", "repertoire-example.yaml", package="airr")
f2 <- system.file("extdata", "germline-example.json", package="airr")

# Load data files
repertoire <- read_airr(f1)
germline <- read_airr(f2)
```

`read_tabular` *Read AIRR tabular data*

Description

`read_tabular` reads a tab-delimited (TSV) file containing tabular AIRR records.

Usage

```
read_tabular(file, schema, base = c("1", "0"), aux_types = NULL, ...)

read_rearrangement(file, base = c("1", "0"), ...)

read_alignment(file, base = c("1", "0"), ...)
```

Arguments

`file`	input file path.
`schema`	`Schema` object defining the output format.
`base`	starting index for positional fields in the input file. If `"1"`, then these fields will not be modified. If `"0"`, then fields ending in `"_start"` and `"_end"` are 0-based half-open intervals (python style) in the input file and will be converted to 1-based closed-intervals (R style).
`aux_types`	named vector or list giving the type for fields that are not defined in `schema`. The field name is the name, the value the type, denoted by one of `"c"` (character), `"l"` (logical), `"i"` (integer), `"d"` (double), or `"n"` (numeric).
`...`	additional arguments to pass to read_delim.

Details

`read_rearrangement` reads an AIRR TSV containing Rearrangement data.

`read_alignment` reads an AIRR TSV containing Alignment data.

Value

A `data.frame` of the TSV file with appropriate type and position conversion for fields defined in the specification.

See Also

See Schema for the AIRR schema object definition. See write_tabular for writing AIRR data.

Examples

```
# Get path to the rearrangement-example file
file <- system.file("extdata", "rearrangement-example.tsv.gz", package="airr")

# Load data file
df <- read_rearrangement(file)
```

`Schema-class` *S4 class defining an AIRR standard schema*

Description

Schema defines a common data structure for AIRR Data Representation standards.

Usage

```
## S4 method for signature 'Schema'
names(x)

## S4 method for signature 'Schema,character'
x[i]

## S4 method for signature 'Schema'
x$name

InfoSchema

DataFileSchema

AlignmentSchema

RearrangementSchema

RepertoireSchema

GermlineSetSchema

GenotypeSetSchema

AIRRSchema
```

Arguments

`x`	Schema object.
`i`	field name.
`name`	field name.

Format

A `Schema` object.

An object of class `Schema` of length 1.

An object of class `Schema` of length 1.

An object of class `Schema` of length 1.

An object of class `Schema` of length 1.

An object of class `Schema` of length 1.

An object of class `Schema` of length 1.

An object of class `Schema` of length 1.

An object of class `list` of length 26.

Details

The following predefined Schema objects are defined:

`InfoSchema`: AIRR Info `Schema`.

`DataFileSchema`: AIRR DataFile `Schema`.

`AlignmentSchema`: AIRR Alignment `Schema`.

`RearrangementSchema`: AIRR Rearrangement `Schema`.

`RepertoireSchema`: AIRR Repertoire `Schema`.

`GermlineSetSchema`: AIRR GermlineSet `Schema`.

`GenotypeSetSchema`: AIRR GenotypeSet `Schema`.

`AIRRSchema`: named list containing all non-experimental AIRR `Schema` objects.

Slots

`definition` name of the schema definition.

`required` `character` vector of required fields.

`optional` `character` vector of non-required fields.

`properties` `list` of field definitions.

`info` `list` schema information.

See Also

See load_schema for loading a `Schema` from the definition set.

`validate_airr` *Validate an AIRR Data Model nested list representation*

Description

`validate_airr` validates the fields in a named nested list representation of the AIRR Data Model. Typically, generating by reading of JSON or YAML formatted AIRR files.

Usage

```
validate_airr(data, model = TRUE, each = FALSE)
```

Arguments

`data`	`list` containing records of an AIRR Data Model objected imported from a YAML or JSON representation.
`model`	if `TRUE` validate only AIRR DataFile defined objects. If `FALSE` attempt validation of all objects in `data`.
`each`	if `TRUE` return a logical vector with results for each object in `data` instead of a single `TRUE` or `FALSE` value.

Value

Returns `TRUE` if the input `data` is compliant with AIRR standards and `FALSE` if not. If `each=TRUE` is set, then a vector with results for each each object in `data` is returned instead.

See Also

See Schema for the AIRR schema definitions. See read_airr for loading AIRR Data Models from a file. See write_airr for writing AIRR Data Models to a file.

Examples

```
# Get path to the rearrangement-example file
f1 <- system.file("extdata", "repertoire-example.yaml", package="airr")
f2 <- system.file("extdata", "germline-example.json", package="airr")

# Load data file
repertoire <- read_airr(f1)
germline <- read_airr(f2)

# Validate a single record
validate_airr(repertoire)

# Return validation for individual objects
validate_airr(germline, each=TRUE)
```

`validate_tabular` *Validate tabular AIRR data*

Description

`validate_tabular` validates compliance of the contents of a `data.frame` to the AIRR standards.

Usage

```
validate_tabular(data, schema)

validate_rearrangement(data)
```

Arguments

`data`	`data.frame` of tabular data to validate.
`schema`	`Schema` object defining the data standard of the table.

Details

`validate_rearrangement` validates the standards compliance of AIRR Rearrangement data stored in a `data.frame`

Value

Returns `TRUE` if the input `data` is compliant and `FALSE` if not.

Examples

```
# Get path to the rearrangement-example file
file <- system.file("extdata", "rearrangement-example.tsv.gz", package="airr")

# Load data file
df <- read_rearrangement(file)

# Validate a data.frame against the Rearrangement schema
validate_rearrangement(df)
```

`write_airr` *Write AIRR Data Model records to YAML or JSON files*

Description

`write_airr` writes a YAML or JSON file containing AIRR Data Model records.

Usage

```
write_airr(
  data,
  file,
  format = c("auto", "yaml", "json"),
  validate = TRUE,
  model = TRUE
)
```

Arguments

`data`	`list` containing AIRR Model Records.
`file`	output file name.
`format`	format of the output file. Must be one of `"auto"`, `"yaml"`, or `"json"`. If `"auto"` (default), the format will be detected from the `file` extension.
`validate`	run schema validation prior to write if `TRUE`.
`model`	if `TRUE` validate and write only AIRR DataFile defined objects. If `FALSE` attempt validation and write of all objects in `data`.

See Also

See Schema for the AIRR schema definition objects. See read_airr for reading to AIRR Data Model files.

Examples

```
# Get path to the repertoire-example file
file <- system.file("extdata", "repertoire-example.yaml", package="airr")

# Load data file
repertoire <- read_airr(file)

# Write a Rearrangement data file
outfile <- file.path(tempdir(), "output.yaml")
write_airr(repertoire, outfile)
```

`write_tabular` *Write an AIRR tabular data*

Description

`write_tabular` writes a TSV containing AIRR tabular records.

Usage

```
write_tabular(data, file, schema, base = c("1", "0"), ...)

write_rearrangement(data, file, base = c("1", "0"), ...)

write_alignment(data, file, base = c("1", "0"), ...)
```

Arguments

`data`	`data.frame` of Rearrangement data.
`file`	output file name.
`schema`	Schema object defining the output format.
`base`	starting index for positional fields in the output file. Fields in the input `data` are assumed to be 1-based closed-intervals (R style). If `"1"`, then these fields will not be modified. If `"0"`, then fields ending in `_start` and `_end` will be converted to 0-based half-open intervals (python style) in the output file.
`...`	additional arguments to pass to write_delim.

Details

`write_rearrangement` writes a `data.frame` containing AIRR Rearrangement data to TSV.

`write_alignment` writes a `data.frame` containing AIRR Alignment data to TSV.

See Also

See Schema for the AIRR schema object definition. See read_tabular for reading to AIRR files.

Examples

```
# Get path to the rearrangement-example file
file <- system.file("extdata", "rearrangement-example.tsv.gz", package="airr")

# Load data file
df <- read_rearrangement(file)

# Write a Rearrangement data file
outfile <- file.path(tempdir(), "output.tsv")
write_tabular(df, outfile, schema=RearrangementSchema)
```

Index

∗ **datasets**
    Schema-class, 6
[,Schema,character-method
        (Schema-class), 6
$,Schema-method *(Schema-class)*, 6

AIRRSchema *(Schema-class)*, 6
AlignmentSchema *(Schema-class)*, 6

DataFileSchema *(Schema-class)*, 6

ExampleData, 2

GenotypeSetSchema *(Schema-class)*, 6
GermlineSetSchema *(Schema-class)*, 6

InfoSchema *(Schema-class)*, 6

load_schema, 3, *7*

names,Schema-method *(Schema-class)*, 6

read_airr, 4, *8*, *10*
read_alignment *(read_tabular)*, 5
read_delim, *5*
read_rearrangement *(read_tabular)*, 5
read_tabular, 5, *11*
RearrangementSchema *(Schema-class)*, 6
RepertoireSchema *(Schema-class)*, 6

Schema, *3–5*, *8*, *10*, *11*
Schema *(Schema-class)*, 6
Schema-class, 6
Schema-method *(Schema-class)*, 6

validate_airr, 8
validate_rearrangement
        (validate_tabular), 9
validate_tabular, 9

write_airr, *4*, *8*, 10
write_alignment *(write_tabular)*, 11
write_delim, *11*
write_rearrangement *(write_tabular)*, 11
write_tabular, *5*, 11