

Package: croissant (via r-universe)

October 27, 2026

Version 0.0.1

Title 'Croissant' Metadata Reader and Writer

Date 2026-10-25

Description Read and write 'croissant' metadata files for machine learning datasets.

License BSD_3_clause + file LICENSE

URL <https://www.rtemis.org>, <https://docs.rtemis.org/r/croissant-api/>,
<https://github.com/rtemis-org/croissant>

BugReports <https://github.com/rtemis-org/croissant/issues>

ByteCompile yes

Depends R (>= 4.1.0)

Imports data.table, jsonlite, methods, openssl, rtemis.core, S7,
stats, utils

Suggests arrow, jsonld, tibble, testthat (>= 3.0.0), withr

Additional_repositories <https://rtemis-org.r-universe.dev>

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

Config/testthat/parallel true

Config/roxygen2/version 8.1.1

Config/pak/sysreqs libssl-dev

Repository <https://rtemis-org.r-universe.dev>

Date/Publication 2026-10-27 18:10:24 UTC

RemoteUrl <https://github.com/rtemis-org/croissant>

RemoteRef HEAD

RemoteSha d9fe41d9d40400a87e7c31328b40a62e56a66183

Contents

croissant-package	2
CreativeWork	3
Croissant	4
croissant_context	6
croissant_from_data	7
CroissantIssues	8
Extract	9
Field	10
FileObject	11
FileSet	12
hf_download	14
is_valid	15
Organization	15
ParentField	16
Person	17
read_croissant	18
read_croissant_hf	19
read_record_set	19
RecordSet	21
Source	22
Transform	24
validate_croissant	25
write_croissant	26
Index	27

croissant-package	croissant: <i>Read and Write 'Croissant' Metadata</i>
-------------------	--

Description

Read, write, and inspect Croissant JSON-LD metadata for machine learning datasets. [read_croissant\(\)](#) reads a document into a [Croissant](#) object and [write_croissant\(\)](#) writes it back.

Author(s)

Maintainer: E.D. Gennatas <gennatas@gmail.com> ([ORCID](#)) [copyright holder]
Authors:

- E.D. Gennatas <gennatas@gmail.com> ([ORCID](#)) [copyright holder]

See Also

Useful links:

- <https://www.rtemis.org>
- <https://docs.rtemis.org/r/croissant-api/>
- <https://github.com/rtemis-org/croissant>
- Report bugs at <https://github.com/rtemis-org/croissant/issues>

CreativeWork

CreativeWork: a license or other work named by a dataset

Description

CreativeWork: a license or other work named by a dataset

Usage

```
CreativeWork(  
  id = NULL,  
  name = NULL,  
  description = NULL,  
  url = NULL,  
  other_properties = NULL  
)
```

Arguments

id	Optional Character: @id.
name	Optional Character: Name, or a character vector named by language tag.
description	Optional Character: Description, or a character vector named by language tag.
url	Optional Character: URL.
other_properties	Optional List: Properties without a typed counterpart, keyed by canonical JSON-LD key, with their parsed JSON values.

Value

CreativeWork object.

Author(s)

EDG

Examples

```
CreativeWork(name = "CC BY 4.0", url = "https://creativecommons.org/licenses/by/4.0/")
```

Description

The Responsible AI properties (`rai`: namespace) are character vectors. Dates are kept as the strings the document gives.

Usage

```
Croissant(  
  version = "1.1",  
  id = NULL,  
  name = NULL,  
  description = NULL,  
  conforms_to = NULL,  
  dataset_version = NULL,  
  sd_version = NULL,  
  license = NULL,  
  sd_license = NULL,  
  url = NULL,  
  same_as = NULL,  
  cite_as = NULL,  
  creator = NULL,  
  publisher = NULL,  
  date_created = NULL,  
  date_modified = NULL,  
  date_published = NULL,  
  keywords = NULL,  
  in_language = NULL,  
  is_live_dataset = NULL,  
  distribution = NULL,  
  record_set = NULL,  
  data_collection = NULL,  
  data_collection_type = NULL,  
  data_collection_missing_data = NULL,  
  data_collection_raw_data = NULL,  
  data_imputation_protocol = NULL,  
  data_preprocessing_protocol = NULL,  
  data_manipulation_protocol = NULL,  
  data_annotation_protocol = NULL,  
  data_annotation_platform = NULL,  
  data_annotation_analysis = NULL,  
  annotations_per_item = NULL,  
  annotator_demographics = NULL,  
  machine_annotation_tools = NULL,  
  data_biases = NULL,
```

```

    data_use_cases = NULL,
    data_limitations = NULL,
    data_social_impact = NULL,
    personal_sensitive_information = NULL,
    data_release_maintenance_plan = NULL,
    other_properties = NULL
)

```

Arguments

version	Character {"1.0", "1.1"}: Croissant version the metadata conforms to.
id	Optional Character: @id of the dataset.
name	Optional Character: Name, or a character vector named by language tag.
description	Optional Character: Description, or a character vector named by language tag.
conforms_to	Optional Character: IRIs of the specifications the metadata conforms to. write_croissant() writes the Croissant IRI of version when this omits it.
dataset_version	Optional Character: Version of the dataset.
sd_version	Optional Character: Version of the metadata document.
license	Optional Character or List: Licenses, as names or URLs, or a list of strings and CreativeWork objects.
sd_license	Optional Character or List: Licenses of the metadata document, in the form of license.
url	Optional Character: URL of the dataset.
same_as	Optional Character: URLs of the same dataset elsewhere.
cite_as	Optional Character: Citation.
creator	Optional List: Person and Organization objects.
publisher	Optional List: Person and Organization objects.
date_created	Optional Character: Creation date.
date_modified	Optional Character: Modification date.
date_published	Optional Character: Publication date.
keywords	Optional Character: Keywords.
in_language	Optional Character: Languages of the content, such as "en".
is_live_dataset	Optional Logical: Whether the dataset changes over time, in which case its files need no checksum.
distribution	Optional List: FileObject and FileSet objects.
record_set	Optional List: RecordSet objects.
data_collection, data_collection_type, data_collection_missing_data, data_collection_raw_data	Optional Character: How the data was collected (rai:dataCollection, rai:dataCollectionType, rai:dataCollectionMissingData, rai:dataCollectionRawData).

```

data_imputation_protocol,          data_preprocessing_protocol,
data_manipulation_protocol
    Optional Character: How the data was processed (rai:dataImputationProtocol,
    rai:dataPreprocessingProtocol, rai:dataManipulationProtocol).
data_annotation_protocol,          data_annotation_platform,
data_annotation_analysis,          annotations_per_item,
annotator_demographics, machine_annotation_tools
    Optional Character: How the data was annotated (rai:dataAnnotationProtocol,
    rai:dataAnnotationPlatform, rai:dataAnnotationAnalysis, rai:annotationsPerItem,
    rai:annotatorDemographics, rai:machineAnnotationTools).
data_biases, data_use_cases, data_limitations, data_social_impact,
personal_sensitive_information, data_release_maintenance_plan
    Optional Character: Uses, limits, and maintenance of the data (rai:dataBiases,
    rai:dataUseCases, rai:dataLimitations, rai:dataSocialImpact, rai:personalSensitiveInfor
    rai:dataReleaseMaintenancePlan).
other_properties
    Optional List: Properties without a typed counterpart, keyed by canonical JSON-
    LD key, with their parsed JSON values, such as the prov: properties.

```

Value

Croissant object.

Author(s)

EDG

Examples

```

Croissant(
  name = "Titanic",
  creator = list(Organization(name = "OpenML")),
  distribution = list(FileObject(id = "passengers.csv", content_url = "titanic.csv"))
)

```

croissant_context	<i>Croissant JSON-LD context</i>
-------------------	----------------------------------

Description

Returns the @context that `write_croissant()` writes for a Croissant version.

Usage

```
croissant_context(version = "1.1")
```

Arguments

version Character {"1.0", "1.1"}: Croissant version.

Value

Named list: The context, ready for `jsonlite::toJSON()`.

Author(s)

EDG

Examples

```
names(croissant_context("1.1"))
```

croissant_from_data	<i>Create Croissant metadata for a table</i>
---------------------	--

Description

Writes `x` to file as CSV or Parquet, and returns a Croissant object that describes it: one `FileObject` with the file's size and SHA-256 checksum, one `RecordSet`, and one `Field` per column, whose data type follows the column class: integer to `sc:Integer`, double to `sc:Float`, logical to `sc:Boolean`, Date to `sc:Date`, POSIXct to `sc:DateTime`, and character and factor to `sc:Text`. Write the metadata with `write_croissant()` into the directory of `file`, where `content_url` resolves.

Usage

```
croissant_from_data(
  x,
  file,
  name,
  format = c("csv", "parquet"),
  record_set = "default",
  content_url = basename(file),
  version = "1.1",
  ...
)
```

Arguments

<code>x</code>	data.frame, data.table, or tibble: The data.
<code>file</code>	Character: Path of the data file to write.
<code>name</code>	Character: Name of the dataset.
<code>format</code>	Character {"csv", "parquet"}: File format. "parquet" requires the 'arrow' package.
<code>record_set</code>	Character: @id of the record set.
<code>content_url</code>	Character: contentUrl of the file, relative to the metadata document.
<code>version</code>	Character {"1.0", "1.1"}: Croissant version.
<code>...</code>	Further properties of the Croissant object, such as description, license, cite_as, creator, or date_published.

Details

A factor is written as its labels and reads back as character. In CSV, a missing value is written as an empty field and an empty string as "", and they read back as NA and "". Parquet keeps every column type.

Value

Croissant object.

Author(s)

EDG

Examples

```
dir <- tempfile()
dir.create(dir)
x <- croissant_from_data(
  iris,
  file.path(dir, "iris.csv"),
  name = "iris",
  license = "CC0-1.0"
)
x
write_croissant(x, file.path(dir, "metadata.json"))
head(read_record_set(file.path(dir, "metadata.json"), "default"))
```

CroissantIssues

CroissantIssues: problems found in Croissant metadata

Description

Returned by `validate_croissant()`. An error makes the metadata invalid; a warning names a recommended property that is absent or a construct that is read in a way the author may not intend.

Usage

```
CroissantIssues(
  issues = data.frame(severity = character(), check = character(), node = character(),
    message = character())
)
```

Arguments

issues	data.frame: One row per issue, with character columns severity ("error" or "warning"), check (the check that found it), node (the @id of the node, or the node type when it has none), and message.
--------	---

Value

CroissantIssues object.

Author(s)

EDG

Examples

```
x <- validate_croissant(Croissant(name = "minimal"))
x
x@issues
```

Extract

Extract: how a field's values are read from a source

Description

Extract: how a field's values are read from a source

Usage

```
Extract(
  column = NULL,
  file_property = NULL,
  json_path = NULL,
  other_properties = NULL
)
```

Arguments

column	Optional Character: Column name in a tabular file.
file_property	Optional Character: File property, such as "content" or "filename".
json_path	Optional Character: JSONPath expression.
other_properties	Optional List: Properties without a typed counterpart, keyed by canonical JSON-LD key, with their parsed JSON values.

Value

Extract object.

Author(s)

EDG

Examples

```
Extract(column = "age")
```

Field

Field: one column or attribute of a record set

Description

Field: one column or attribute of a record set

Usage

```
Field(
  id = NULL,
  name = NULL,
  description = NULL,
  data_type = NULL,
  source = NULL,
  references = NULL,
  sub_field = NULL,
  parent_field = NULL,
  annotation = NULL,
  repeated = NULL,
  is_array = NULL,
  array_shape = NULL,
  equivalent_property = NULL,
  value = NULL,
  other_properties = NULL
)
```

Arguments

id	Optional Character: @id, unique within the document.
name	Optional Character: Name, or a character vector named by language tag.
description	Optional Character: Description, or a character vector named by language tag.
data_type	Optional Character: Data types, as compact IRIs such as "sc:Text" or "sc:Integer".
source	Optional Source object: Where the values come from.
references	Optional Source object: The field this field joins to.
sub_field	Optional List: Field objects nested in this field.
parent_field	Optional ParentField object.
annotation	Optional List: Field objects that annotate this field.
repeated	Optional Logical: Whether the field holds a list of values.
is_array	Optional Logical: Whether the field holds an array.
array_shape	Optional Character: Array shape, such as "28,28".
equivalent_property	Optional Character: IRIs of properties equivalent to this field, such as "prov:wasDerivedFrom".

- value Optional: Constant value of the field, any JSON value.
- other_properties Optional List: Properties without a typed counterpart, keyed by canonical JSON-LD key, with their parsed JSON values.

Value

Field object.

Author(s)

EDG

Examples

```
Field(  
  id = "passengers/name",  
  name = "name",  
  data_type = "sc:Text",  
  source = Source(file_object = "passengers.csv", extract = Extract(column = "name"))  
)
```

FileObject	<i>FileObject: a single file of the dataset</i>
------------	---

Description

FileObject: a single file of the dataset

Usage

```
FileObject(  
  id = NULL,  
  name = NULL,  
  description = NULL,  
  content_url = NULL,  
  content_size = NULL,  
  encoding_format = NULL,  
  sha256 = NULL,  
  md5 = NULL,  
  contained_in = NULL,  
  same_as = NULL,  
  source = NULL,  
  other_properties = NULL  
)
```

Arguments

id	Optional Character: @id, unique within the document.
name	Optional Character: Name, or a character vector named by language tag.
description	Optional Character: Description, or a character vector named by language tag.
content_url	Optional Character: URL or relative path of the file.
content_size	Optional Character: Size, such as "117743 B".
encoding_format	Optional Character: MIME types, such as "text/csv".
sha256	Optional Character: SHA-256 checksum.
md5	Optional Character: MD5 checksum.
contained_in	Optional Character: @id of each FileObject or FileSet the file is extracted from. A containedIn given as a source with a transform is kept in other_properties.
same_as	Optional Character: URLs of the same file elsewhere.
source	Optional Source object: The file this file is derived from.
other_properties	Optional List: Properties without a typed counterpart, keyed by canonical JSON-LD key, with their parsed JSON values.

Value

FileObject object.

Author(s)

EDG

Examples

```
FileObject(  
  id = "passengers.csv",  
  content_url = "data/titanic.csv",  
  encoding_format = "text/csv"  
)
```

FileSet	<i>FileSet: a set of files matched by glob patterns</i>
---------	---

Description

FileSet: a set of files matched by glob patterns

Usage

```
FileSet(  
  id = NULL,  
  name = NULL,  
  description = NULL,  
  encoding_format = NULL,  
  includes = NULL,  
  excludes = NULL,  
  contained_in = NULL,  
  other_properties = NULL  
)
```

Arguments

id	Optional Character: @id, unique within the document.
name	Optional Character: Name, or a character vector named by language tag.
description	Optional Character: Description, or a character vector named by language tag.
encoding_format	Optional Character: MIME types of the files.
includes	Optional Character: Glob patterns of the files included.
excludes	Optional Character: Glob patterns of the files excluded.
contained_in	Optional Character: @id of each FileObject or FileSet the files are extracted from. A containedIn given as a source with a transform is kept in other_properties.
other_properties	Optional List: Properties without a typed counterpart, keyed by canonical JSON-LD key, with their parsed JSON values.

Value

FileSet object.

Author(s)

EDG

Examples

```
FileSet(  
  id = "images",  
  contained_in = "archive.zip",  
  encoding_format = "image/jpeg",  
  includes = "*.jpg"  
)
```

hf_download

*Download a file from a Hugging Face dataset***Description**

Resolves an hf://datasets/ path to a commit, downloads the file from that commit into cache_dir, and verifies its size and its LFS SHA-256 or git blob SHA-1 against the Hub's listing. A cached file of the listed size is reused; a commit's files do not change. A path that names the repository only reads its one data file (.parquet, .csv, .tsv, .csv.gz, .tsv.gz, .rds, .xlsx, .dta, .arff, or .fasta).

Usage

```
hf_download(
  path,
  cache_dir = getOption("croissant.cache_dir", tools::R_user_dir("croissant", "cache")),
  token = NULL,
  verbosity = 1L
)
```

Arguments

path	Character: hf://datasets/<owner>/<name>[@<revision>][/<file>]. A revision that contains / is URL-encoded, as refs%2Fconvert%2Fparquet.
cache_dir	Character: Cache directory.
token	Optional Character: Hugging Face token. NULL reads HF_TOKEN.
verbosity	Integer: Verbosity level; 1L reports downloads.

Details

A gated or private dataset requires a token with access, read from HF_TOKEN by default. The Hub is at HF_ENDPOINT, or <https://huggingface.co>.

Value

Character: Local path of the file, with attribute hf, the path pinned to the commit (hf://datasets/<owner>/<name>@<commit>).

Author(s)

EDG

Examples

```
path <- hf_download("hf://datasets/ylecun/mnist@refs%2Fconvert%2Fparquet/mnist/test/0000.parquet")
attr(path, "hf")
```

is_valid	<i>Test whether validation found no errors</i>
----------	--

Description

Test whether validation found no errors

Usage

```
is_valid(x)
```

Arguments

x CroissantIssues object.

Value

Logical scalar: TRUE when x holds no error. Warnings do not count.

Author(s)

EDG

Examples

```
is_valid(validate_croissant(Croissant(name = "minimal")))
```

Organization	<i>Organization: an organization that created or published a dataset</i>
--------------	--

Description

Organization: an organization that created or published a dataset

Usage

```
Organization(  
  id = NULL,  
  name = NULL,  
  description = NULL,  
  email = NULL,  
  url = NULL,  
  other_properties = NULL  
)
```

Arguments

id	Optional Character: @id.
name	Optional Character: Name, or a character vector named by language tag.
description	Optional Character: Description, or a character vector named by language tag.
email	Optional Character: Email address.
url	Optional Character: URL.
other_properties	Optional List: Properties without a typed counterpart, keyed by canonical JSON-LD key, with their parsed JSON values.

Value

Organization object.

Author(s)

EDG

Examples

```
Organization(name = "MLCommons", url = "https://mlcommons.org")
```

ParentField	<i>ParentField: the field a nested field joins to</i>
-------------	---

Description

ParentField: the field a nested field joins to

Usage

```
ParentField(source = NULL, references = NULL, other_properties = NULL)
```

Arguments

source	Optional Source object.
references	Optional Source object.
other_properties	Optional List: Properties without a typed counterpart, keyed by canonical JSON-LD key, with their parsed JSON values.

Value

ParentField object.

Author(s)

EDG

Examples

```
ParentField(source = Source(file_object = "data.json", extract = Extract(json_path = "$.id")))
```

 Person

Person: a person who created or published a dataset

Description

Person: a person who created or published a dataset

Usage

```
Person(
  id = NULL,
  name = NULL,
  description = NULL,
  email = NULL,
  url = NULL,
  other_properties = NULL
)
```

Arguments

id	Optional Character: @id.
name	Optional Character: Name, or a character vector named by language tag.
description	Optional Character: Description, or a character vector named by language tag.
email	Optional Character: Email address.
url	Optional Character: URL.
other_properties	Optional List: Properties without a typed counterpart, keyed by canonical JSON-LD key, with their parsed JSON values.

Value

Person object.

Author(s)

EDG

Examples

```
Person(name = "Ada Lovelace", email = "ada@example.org")
```

read_croissant	<i>Read Croissant metadata</i>
----------------	--------------------------------

Description

Reads a Croissant JSON-LD document into a Croissant object. A document whose @context defines only standard Croissant terms is read with 'jsonlite'. A document with any other context is first compacted against the standard context by the 'jsonld' package, which must then be installed.

Usage

```
read_croissant(x)
```

Arguments

x	Character: Path to a Croissant JSON-LD file, an http(s) URL, a JSON string, or an hf://datasets/ path. A path naming a file of a Hugging Face dataset reads that file; one naming the dataset only, as hf://datasets/<owner>/<name>, reads the Croissant metadata the Hub generates for it, as read_croissant_hf() does.
---	--

Details

Croissant 1.0 and 1.1 documents are supported. A 0.8 document, or one that conforms to another version, raises an error of class `croissant_version_error`.

Value

Croissant object.

Author(s)

EDG

Examples

```
json <- '{
  "@context": {"@vocab": "https://schema.org/", "cr": "http://mlcommons.org/croissant/",
    "recordSet": "cr:recordSet", "field": "cr:field",
    "dataType": {"@id": "cr:dataType", "@type": "@vocab"}},
  "@type": "sc:Dataset",
  "name": "minimal",
  "conformsTo": "http://mlcommons.org/croissant/1.1",
  "recordSet": [{"@type": "cr:RecordSet", "@id": "rs", "field": [
    {"@type": "cr:Field", "@id": "rs/x", "dataType": "sc:Integer"}]}]
}'
x <- read_croissant(json)
x
```

read_croissant_hf	<i>Read the Croissant metadata of a Hugging Face dataset</i>
-------------------	--

Description

Downloads the metadata the Hub generates for a dataset (<endpoint>/api/datasets/<owner>/<name>/croissant) and reads it with [read_croissant\(\)](#). Its record sets read through [read_record_set\(\)](#), which downloads the Parquet files they name from the dataset's refs/convert/parquet branch.

Usage

```
read_croissant_hf(repo, token = NULL)
```

Arguments

repo	Character: Dataset, as <owner>/<name> or hf://datasets/<owner>/<name>.
token	Optional Character: Hugging Face token, needed for gated and private datasets. NULL reads HF_TOKEN.

Value

Croissant object.

Author(s)

EDG

Examples

```
x <- read_croissant_hf("ylecun/mnist")
x
head(read_record_set(x, "mnist", n_max = 5L))
```

read_record_set	<i>Read the records of a record set</i>
-----------------	---

Description

Reads the files a record set draws from, and returns its records as a table with one column per field, named by field @id. Subfields are columns of their own, named by their @id.

Usage

```
read_record_set(
  x,
  record_set,
  output = c("data.table", "tibble", "data.frame", "arrow"),
  base = NULL,
  cache_dir = getOption("croissant.cache_dir", tools::R_user_dir("croissant", "cache")),
  n_max = Inf,
  token = NULL,
  verbosity = 1L
)
```

Arguments

<code>x</code>	Croissant object, or Character: Path or URL of a Croissant JSON-LD file.
<code>record_set</code>	Character: @id or name of the record set.
<code>output</code>	Character {"data.table", "tibble", "data.frame", "arrow"}: Class of the result. "tibble" requires the 'tibble' package and "arrow" the 'arrow' package.
<code>base</code>	Optional Character: Directory or URL that relative contentUrls and FileSet globs resolve against. When x is a path or URL, this defaults to its directory.
<code>cache_dir</code>	Character: Directory for downloaded and extracted files.
<code>n_max</code>	Numeric [1, Inf]: Most records to return. Files of a FileSet are read in order until the first origin holds n_max records, so later files are not downloaded.
<code>token</code>	Optional Character: Hugging Face token, for files of gated or private Hub datasets. NULL reads HF_TOKEN.
<code>verbosity</code>	Integer: Verbosity level; 1L reports downloads.

Details

Files with an http(s) contentUrl are downloaded into cache_dir, and their sha256 or md5 is verified. Files of a Hugging Face dataset repository (a git+https FileObject with a Hub contentUrl, as in the metadata `read_croissant_hf()` reads, or an hf://datasets/ path) are downloaded from the resolved commit and verified against the Hub's listing, as `hf_download()` does. Zip and tar archives are extracted into cache_dir. Relative contentUrls and FileSet globs resolve against base. Columns are read from CSV, TSV, JSON, JSON Lines, and Parquet files; Parquet requires the 'arrow' package. A field extracting the content of an image, audio, or video file holds the file's local path; with a text data type it holds the file's text.

Values are cast by the field's first data type with an R counterpart: sc:Integer to integer, sc:Float and sc:Number to double, sc:Boolean to logical, sc:Date to Date, sc:DateTime to POSIXct (UTC), and sc:Text and sc:URL to character. A repeated field, or one split by a separator transform, is a list column.

Value

A data.table, tibble, data.frame, or arrow Table, by output.

Author(s)

EDG

Examples

```

dir <- tempfile()
dir.create(dir)
writeLines(c("name,age", "Alice,22", "Bob,23"), file.path(dir, "people.csv"))
x <- Croissant(
  name = "people",
  distribution = list(
    FileObject(id = "people.csv", content_url = "people.csv", encoding_format = "text/csv")
  ),
  record_set = list(RecordSet(
    id = "people",
    field = list(
      Field(
        id = "people/name",
        data_type = "sc:Text",
        source = Source(file_object = "people.csv", extract = Extract(column = "name"))
      ),
      Field(
        id = "people/age",
        data_type = "sc:Integer",
        source = Source(file_object = "people.csv", extract = Extract(column = "age"))
      )
    )
  ))
)
read_record_set(x, "people", base = dir)

```

RecordSet

*RecordSet: a set of records with a common structure***Description**

RecordSet: a set of records with a common structure

Usage

```

RecordSet(
  id = NULL,
  name = NULL,
  description = NULL,
  data_type = NULL,
  key = NULL,
  field = NULL,
  annotation = NULL,
  data = NULL,

```

```
    examples = NULL,  
    other_properties = NULL  
  )
```

Arguments

id	Optional Character: @id, unique within the document.
name	Optional Character: Name, or a character vector named by language tag.
description	Optional Character: Description, or a character vector named by language tag.
data_type	Optional Character: Data types, such as "sc:Enumeration".
key	Optional Character: @id of each field that forms the record key.
field	Optional List: Field objects.
annotation	Optional List: Field objects that annotate the record set.
data	Optional List: Records given inline, as parsed JSON.
examples	Optional List: Example records, as parsed JSON.
other_properties	Optional List: Properties without a typed counterpart, keyed by canonical JSON-LD key, with their parsed JSON values.

Value

RecordSet object.

Author(s)

EDG

Examples

```
RecordSet(id = "passengers", name = "passengers", field = list(Field(id = "passengers/name")))
```

Source	<i>Source: where a field's values come from</i>
--------	---

Description

Names exactly one of file_object, file_set, or field by @id, and optionally how values are extracted and transformed.

Usage

```
Source(  
  file_object = NULL,  
  file_set = NULL,  
  record_set = NULL,  
  field = NULL,  
  extract = NULL,  
  transform = NULL,  
  format = NULL,  
  sampling_rate = NULL,  
  other_properties = NULL  
)
```

Arguments

<code>file_object</code>	Optional Character: @id of a FileObject.
<code>file_set</code>	Optional Character: @id of a FileSet.
<code>record_set</code>	Optional Character: @id of a RecordSet.
<code>field</code>	Optional Character: @id of a Field.
<code>extract</code>	Optional Extract object.
<code>transform</code>	Optional List: Transform objects, applied in order.
<code>format</code>	Optional Character: Format of the source values.
<code>sampling_rate</code>	Optional Numeric: Sampling rate of audio content.
<code>other_properties</code>	Optional List: Properties without a typed counterpart, keyed by canonical JSON-LD key, with their parsed JSON values.

Value

Source object.

Author(s)

EDG

Examples

```
Source(file_object = "passengers.csv", extract = Extract(column = "name"))
```

Transform	<i>Transform: a transformation applied to extracted values</i>
-----------	--

Description

Transform: a transformation applied to extracted values

Usage

```
Transform(  
  format = NULL,  
  json_path = NULL,  
  read_lines = NULL,  
  regex = NULL,  
  replace = NULL,  
  sampling_rate = NULL,  
  separator = NULL,  
  un_archive = NULL,  
  other_properties = NULL  
)
```

Arguments

format	Optional Character: Format used to parse the value, such as a date format.
json_path	Optional Character: JSONPath expression.
read_lines	Optional Logical: Whether to split the content into lines.
regex	Optional Character: Regular expression whose first capture group is kept.
replace	Optional Character: Replacement pattern.
sampling_rate	Optional Numeric: Sampling rate of audio content.
separator	Optional Character: Separator used to split the value.
un_archive	Optional Logical: Whether to extract an archive.
other_properties	Optional List: Properties without a typed counterpart, keyed by canonical JSON-LD key, with their parsed JSON values.

Value

Transform object.

Author(s)

EDG

Examples

```
Transform(regex = "^((\\d+)_.*$")
```

validate_croissant	<i>Validate Croissant metadata</i>
--------------------	------------------------------------

Description

Reads `x` when it is a path, URL, or JSON string, then checks the metadata, and reports every problem found in one `CroissantIssues` object. Problems that `read_croissant()` raises as an error, such as a value of the wrong JSON type, are reported here as issues.

Usage

```
validate_croissant(x)
```

Arguments

<code>x</code>	Croissant object, or Character: Path to a Croissant JSON-LD file, an http(s) URL, or a JSON string.
----------------	---

Details

Errors cover: values of the wrong JSON type; node types; mandatory properties (`encodingFormat` and `contentType` of a `FileObject`, `encodingFormat` and `includes` of a `FileSet`); a checksum (`md5` or `sha256`) on each `FileObject` that is not contained in another file; `@ids` that contain whitespace or repeat; references to nodes that do not exist; sources that do not name exactly one field, `fileObject`, or `fileSet`; fields with neither source nor value; fields without a data type of their own or of their source field; record sets that combine several sources without a join (references); inline data whose records do not match the fields; `arrayShape` without `isArray` or with non-integer dimensions; and transforms that define no transformation, or more than one string transformation.

Warnings cover the recommended dataset properties `citeAs`, `datePublished`, `license`, and `version`; a missing `conformsTo`; nodes without `@id`, which no other node can reference; record set keys that name no field; and Croissant terms used as keys without a definition in the `@context`.

Value

`CroissantIssues` object.

Author(s)

EDG

Examples

```
x <- Croissant(  
  name = "example",  
  distribution = list(FileObject(id = "data.csv", encoding_format = "text/csv"))  
)  
issues <- validate_croissant(x)
```

```
issues
is_valid(issues)
```

write_croissant	<i>Write Croissant metadata</i>
-----------------	---------------------------------

Description

Writes a Croissant object as a JSON-LD document with the standard context of its version (see [croissant_context\(\)](#)). conformsTo always includes the Croissant IRI of x@version.

Usage

```
write_croissant(x, path = NULL, pretty = TRUE)
```

Arguments

x	Croissant object.
path	Optional Character: File to write. If NULL, the JSON text is returned.
pretty	Logical: Whether to indent the JSON.

Details

In version 1.1, a name or description named by language tag is written as a language map. In version 1.0, it is written as an array of @value/@language objects.

Value

If path is NULL, the JSON text as a character scalar of class json; otherwise path, invisibly.

Author(s)

EDG

Examples

```
x <- Croissant(
  name = "Titanic",
  distribution = list(
    FileObject(id = "passengers.csv", content_url = "titanic.csv", encoding_format = "text/csv")
  )
)
cat(write_croissant(x))
path <- tempfile(fileext = ".json")
write_croissant(x, path)
read_croissant(path)
```

Index

CreativeWork, [3](#)
Croissant, [2](#), [4](#)
croissant (croissant-package), [2](#)
croissant-package, [2](#)
croissant_context, [6](#)
croissant_context(), [26](#)
croissant_from_data, [7](#)
CroissantIssues, [8](#)

Extract, [9](#)

Field, [10](#)
FileObject, [11](#)
FileSet, [12](#)

hf_download, [14](#)
hf_download(), [20](#)

is_valid, [15](#)

Organization, [15](#)

ParentField, [16](#)
Person, [17](#)

read_croissant, [18](#)
read_croissant(), [2](#), [19](#), [25](#)
read_croissant_hf, [19](#)
read_croissant_hf(), [18](#), [20](#)
read_record_set, [19](#)
read_record_set(), [19](#)
RecordSet, [21](#)

Source, [22](#)

Transform, [24](#)

validate_croissant, [25](#)
validate_croissant(), [8](#)

write_croissant, [26](#)
write_croissant(), [2](#), [5–7](#)