Package 'dataset'

December 23, 2024

Title Create Data Frames that are Easier to Exchange and Reuse

Version 0.3.4

Date 2024-12-20

Language en-US

Maintainer Daniel Antal <daniel.antal@dataobservatory.eu>

Description The aim of the 'dataset' package is to make tidy datasets easier to release, exchange and reuse. It organizes and formats data frame 'R' objects into well-referenced, well-described, interoperable datasets into release and reuse ready form.

License GPL (>= 3)

Encoding UTF-8

URL https://dataset.dataobservatory.eu/

BugReports https://github.com/dataobservatory-eu/dataset/issues/

LazyData true

Imports assertthat, cli, haven, ISOcodes, labelled, methods, pillar, rlang, tibble, utils, vctrs (>= 0.5.2)

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, spelling, testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 3.5)

VignetteBuilder knitr

NeedsCompilation no

Author Daniel Antal [aut, cre] (<https://orcid.org/0000-0001-7513-6760>), Marcelo Perlin [rev] (<https://orcid.org/0000-0002-9839-4268>)

Repository CRAN

Date/Publication 2024-12-23 11:10:02 UTC

Contents

`as_character` *Coerce to character vector*

Description

Coerce to character vector

Usage

```
as_character(x)

## S3 method for class 'haven_labelled_defined'
as_character(x)
```

Arguments

x	A vector created with defined.

Value

A character vector.

Examples

```
as_character(iris_dataset$Species)
```

`as_numeric` *Coerce a defined vector to numeric*

Description

Coerce a defined vector to numeric

Usage

```
as_numeric(x)

## S3 method for class 'haven_labelled_defined'
as_numeric(x)
```

Arguments

x	A vector created with defined.

Value

A numeric vector.

Examples

```
as_numeric(iris_dataset$Petal.Length)
```

`creator` *Get/set the Creator of the object.*

Description

Add the optional `Creator` property as an attribute to a dataset object.

Usage

```
creator(x)

creator(x, overwrite = TRUE) <- value
```

Arguments

`x`	A semantically rich data frame object created by `dataset::dataset_df` or `dataset::as_dataset_df`.
`overwrite`	If the attributes should be overwritten. In case it is set to `FALSE`, it gives a message with the current `Creator` property instead of overwriting it. Defaults to `TRUE` when the attribute is set to `value` regardless of previous setting.
`value`	The `Creator` as a `utils::person` object.

Details

The `Creator` corresponds to dct:creator in Dublin Core and Creator in DataCite. The name of the entity that holds, archives, publishes prints, distributes, releases, issues, or produces the dataset. This property will be used to formulate the citation, so consider the prominence of the role.

Value

The Creator attribute as a character of length one is added to `x`.

See Also

Other Bibliographic reference functions: `dataset_title()`

Examples

```
creator(iris_dataset)
# To change author:
creator(iris_dataset) <- person("Jane", "Doe")
# To add author:
creator(iris_dataset, overwrite=FALSE) <- person("John", "Doe")
```

`datacite` *Create a bibentry object with DataCite metadata fields*

Description

Add metadata conforming the DataCite Metadata Schema.

Usage

```
datacite(
  Title,
  Creator,
  Identifier = NULL,
  Publisher = NULL,
  PublicationYear = NULL,
  Subject = subject_create(term = "data sets", subjectScheme =
    "Library of Congress Subject Headings (LCSH)", schemeURI =
    "https://id.loc.gov/authorities/subjects.html", valueURI =
    "http://id.loc.gov/authorities/subjects/sh2018002256"),
  Type = "Dataset",
  Contributor = NULL,
  DateList = ":tba",
  Language = NULL,
  AlternateIdentifier = ":unas",
  RelatedIdentifier = ":unas",
  Format = ":tba",
  Version = "0.1.0",
  Rights = ":tba",
  Description = ":tba",
  Geolocation = ":unas",
  FundingReference = ":unas"
)

as_datacite(x, type = "bibentry", ...)

is.datacite(x)

## S3 method for class 'datacite'
is.datacite(x)
```

Arguments

`Title`	The name(s) or title(s) by which a resource is known. May be the title of a dataset or the name of a piece of software. Similar to dct:title.
`Creator`	The main researchers involved in producing the data, or the authors of the publication, in priority order. To supply multiple creators, repeat this property.

`Identifier`	The Identifier is a unique string that identifies a resource. For software, determine whether the identifier is for a specific version of a piece of software, (per the Force11 Software Citation Principles, or for all versions. Similar to `dct:title` in `dublincore()`.
`Publisher`	The name of the entity that holds, archives, publishes prints, distributes, releases, issues, or produces the resource. This property will be used to formulate the citation, so consider the prominence of the role. For software, use Publisher for the code repository. Mandatory in DataCite, and similar to `dct:publisher`. See `publisher()`.
`PublicationYear`	The year when the data was or will be made publicly available in `YYYY` format.See `publication_year()`.
`Subject`	Recommended for discovery. Subject, keyword, classification code, or key phrase describing the resource. Similar to dct:subject. Use `subject` to properly add a key phrase from a controlled vocabulary and create structured Subject objects with `subject_create`.
`Type`	Defaults to `Dataset`. The DataCite resourceType definition refers back to dcm:type. The `Type$resourceTypeGeneral` is set to `"Dataset"`, while the user can set a more specific `Type$resourceType` value.
`Contributor`	Recommended for discovery. The institution or person responsible for collecting, managing, distributing, or otherwise contributing to the development of the resource.
`DateList`	DataCite 4.4 allows to set multiple dates to a resource, they should be added as a list. See: datacite:Date.
`Language`	The primary language of the resource. Allowed values are taken from IETF BCP 47, ISO 639-1 language code. See `language()`.
`AlternateIdentifier`	An identifier or identifiers other than the primary Identifier applied to the resource being registered. This may be any alphanumeric string unique within its domain of issue. It may be used for local identifiers. `AlternateIdentifier` should be used for another identifier of the same instance (same location, same file). Defaults to `":unas"` for unassigned values.
`RelatedIdentifier`	Recommended for discovery. Defaults to `":unas"` for unassigned values. Similar to dct:relation.
`Format`	Technical format of the resource. Use file extension or MIME type where possible, e.g., PDF, XML, MPG or application/pdf, text/xml, video/mpeg. Similar to dct:format.
`Version`	Free text. Suggested practice: track major_version.minor_version. Defaults to `"0.1.0"`. See `version`.
`Rights`	Any rights information for this resource. The property may be repeated to record complex rights characteristics, but this is not yet supported. Free text. See `rights`. Defaults to `":tba"`.
`Description`	Recommended for discovery. All additional information that does not fit in any of the other categories. It may be used for technical information—a free text. Defaults to `":tba"`. Similar to dct:description.

`Geolocation`	Recommended for discovery. Spatial region or named place where the data was gathered or about which the data is focused. See `geolocation()`.
`FundingReference`	Information about financial support (funding) for the resource being registered. Defaults to `":unas"` for unassigned values. Complex types with subproperties are not yet implemented.
`x`	An object that is tested if it has a class "datacite".
`type`	A DataCite 4.4 metadata can be returned as a `type="list"`, a `type="dataset_df"`, or a `type="bibentry"` (default).
`...`	Optional parameters to add to a `datacite` object. `author=person("Jane", "Doe")` adds an author to the citation object if `type="dataset"`. as_datacite(iris_dataset, type="list")

Details

DataCite is a leading global non-profit organisation that provides persistent identifiers (DOIs) for research data and other research outputs. Organisations within the research community join DataCite as members to be able to assign DOIs to all their research outputs. This way, their outputs become discoverable, and associated metadata is made available to the community.

The `ResourceType` property will be by definition "Dataset". The `Size` attribute (e.g. bytes, pages, inches, etc.) will automatically added to the dataset.

Value

`datacite()` creates a `utils::bibentry` object extended with standard Dublin Core bibliographical metadata, `as_datacite()` retrieves the contents of this bibentry object of a dataset_df from its attributes, and returns the contents as list, dataset_df, or bibentry object.

`as_datacite(x, type)` returns the DataCite bibliographical metadata of x either as a list, a bibentry object, or a dataset_df object.

`is.datacite(x)` returns a logical values (if the object `x` is of class `datacite`).

Source

DataCite 4.3 Mandatory Properties and DataCite 4.3 Optional Properties

See Also

Other bibentry functions: `dublincore()`, `get_bibentry()`

Examples

```
datacite(
   Title = "Iris Dataset",
   Creator = person(family = "Anderson", given = "Edgar", role = "aut"),
   Publisher = "American Iris Society",
   PublicationYear = 1935,
   Geolocation = "US",
   Language = "en")
```

```
as_datacite(iris_dataset)
```

`dataset_df` *Create a new dataset_df object*

Description

The `dataset_df` constructor creates the objects of this class, which are semantically rich, modern data frames inherited from `tibble::tibble`.

Usage

```
dataset_df(
  ...,
  dataset_bibentry = NULL,
  var_labels = NULL,
  units = NULL,
  definitions = NULL,
  dataset_subject = NULL
)

as_dataset_df(
  df,
  var_labels = NULL,
  units = NULL,
  definitions = NULL,
  dataset_bibentry = NULL,
  dataset_subject = NULL,
  ...
)

is.dataset_df(x)

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

is_dataset_df(x)
```

Arguments

`...`	The vectors (variables) that should be included in the dataset.
`dataset_bibentry`	A list of bibliographic references and descriptive metadata about the dataset as a whole created with `datacite` or `dublincore`.
`var_labels`	The long, human readable labels of each variable.
`units`	The units of measurement for the measured variables.

`definitions`	The linked definitions of the variables, attributes, or constants.
`dataset_subject`	The subject of the dataset, see `subject`.
`df`	A `data.frame` to be converted to `dataset_df`.
`x`	A `dataset_df` object for S3 methods.

Details

To check if an object has the class dataset_df use `is.dataset_df`.

`print` is the method to print out the semantically rich data frames created with the constructor of `dataset_df`.

`summary` is the method to summarise these semantically rich data frames.

For more details, please check the `vignette("dataset_df", package = "dataset")` vignette.

Value

`dataset_df` is the constructor of this type, it returns an object inherited from a data frame with semantically rich metadata.

`is.dataset_df` returns a logical value (if the object is of class `dataset_df`.)

Examples

```
my_dataset <- dataset_df(
   country_name = defined(
     c("AD", "LI"),
     definition = "http://data.europa.eu/bna/c_6c2bb82d",
     namespace = "https://www.geonames.org/countries/$1/"),
   gdp = defined(
     c(3897, 7365),
     label = "Gross Domestic Product",
     unit = "million dollars",
     definition = "http://data.europa.eu/83i/aa/GDP")
)

print(my_dataset)

is.dataset_df(my_dataset)
```

`dataset_title` *Get/set the title of a dataset*

Description

Get or reset the dataset's main title.

Usage

```
dataset_title(x)

dataset_title(x, overwrite = FALSE) <- value
```

Arguments

`x`	A dataset object created with `dataset_df()` or `as_dataset_df()`.
`overwrite`	If the attributes should be overwritten. In case it is set to `FALSE`,it gives a warning with the current `title` property instead of overwriting it. Defaults to `FALSE`.
`value`	The name(s) or title(s) by which a resource is known. See: dct:title.

Details

In the DataCite definition, several titles can be used; it is not yet implemented.

Value

A string with the dataset's title; `set_dataset_title` returns a dataset object with the changed (main) title.

See Also

Other Bibliographic reference functions: `creator()`

Examples

```
dataset_title(iris_dataset)
dataset_title(iris_dataset, overwrite = TRUE) <-"The Famous Iris Dataset"
dataset_title(iris_dataset)
```

`dataset_to_triples` *Dataset to triples (three columns)*

Description

The dataset is converted into a three-column long format with columns `s` for subject, `p` for predicate and `o` for object.

Usage

```
dataset_to_triples(x, idcol = NULL)
```

Arguments

`x`	An R object that contains the data of the dataset (a data.frame or inherited from `data.frame`), for example, `dataset_df()`.
`idcol`	The identifier column. If `idcol` is `NULL` it attempts to use the `row.names(df)` as an `idcol`.

Value

The long form version of the original dataset, retaining the attributes and class.

Examples

```
dataset_to_triples(iris_dataset)
```

`defined` *Create a semantically well-defined, labelled vector*

Description

The `defined` constructor creates the objects of this class, which are semantically extended vectors inherited from `haven::labelled`.

Usage

```
defined(
  x,
  labels = NULL,
  label = NULL,
  unit = NULL,
  definition = NULL,
  namespace = NULL
)

is.defined(x)

## S3 method for class 'haven_labelled_defined'
as.character(x, ...)

## S3 method for class 'haven_labelled_defined'
summary(object, ...)
```

Arguments

`x`	A vector to label. Must be either numeric (integer or double) or character.
`labels`	A named vector or `NULL`. The vector should be the same type as `x`. Unlike factors, labels don't need to be exhaustive: only a fraction of the values might be labelled.
`label`	A short, human-readable description of the vector or `NULL`.
`unit`	A character string of length one containing the unit of measure or `NULL`.
`definition`	A character string of length one containing a linked definition or `NULL`.
`namespace`	A namespace for individual observations or categories or `NULL`.
`...`	Further parameters for inheritance, not in use.
`object`	An R object to be summarised.

Details

`as.character` coerces a defined vector into a character vector.
`summary` summarises the `defined` vector.
For more details, please check the `vignette("defined", package = "dataset")` vignette.

Value

The constructor `defined` returns a vector with defined value labels, a variable label, an optional unit of measurement and linked definition.
`is.defined` returns a logical value, stateing if the object is of class `defined`.

See Also

Other defined metadata methods and functions: `var_label()`, `var_namespace()`, `var_unit()`

Examples

```
gdp_vector <- defined(
  c(3897, 7365, 6753),
  label = "Gross Domestic Product",
  unit = "million dollars",
  definition = "http://data.europa.eu/83i/aa/GDP"
 )

 # To check the s3 class of the vector:
 is.defined(gdp_vector)

 # To print the defined vector:
 print(gdp_vector)

 # To summarise the defined vector:
 summary(gdp_vector)

 # Subsetting work as expected:
 gdp_vector[1:2]
```

`description` *Get/set the Description of the object.*

Description

Get/set the optional `Description` property as an attribute to an R object.

Usage

```
description(x)

description(x, overwrite = FALSE) <- value
```

Arguments

`x`	A dataset object created with `dataset::dataset_df` or `dataset::as_dataset_df`.
`overwrite`	If the `Description` attribute should be overwritten. In case it is set to `FALSE`, it gives a message with the current `Description` property instead of overwriting it. Defaults to `FALSE`, when it gives a warning at an accidental overwrite attempt.
`value`	The `Description` as a character set.

Details

The `Description` is recommended for discovery in DataCite. All additional information that does not fit in any of the other categories. May be used for technical information. A free text. Similar to dct:description.

Value

The `Description` attribute as a character of length 1 is added to `x`.

See Also

Other Reference metadata functions: `geolocation()`, `identifier()`, `language`, `publication_year()`, `publisher()`, `rights()`

Examples

```
description(iris_dataset, overwrite = TRUE) <- "The famous iris dataset used
in R language examples."
description(iris_dataset)
```

`dublincore` *Add or get Dublin Core metadata*

Description

Add metadata conforming the DCMI Metadata Terms. to datasets, i.e. structured R data.frame or list objects, for an accurate and consistent identification of a resource for citation and retrieval purposes.

Usage

```
dublincore(
  title,
  creator,
  identifier = NULL,
  publisher = NULL,
  subject = NULL,
  type = "DCMITYPE:Dataset",
  contributor = NULL,
  date = NULL,
  language = NULL,
  relation = NULL,
  format = "application/r-rds",
  rights = NULL,
  datasource = NULL,
  description = NULL,
  coverage = NULL
)

as_dublincore(x, type = "bibentry", ...)

is.dublincore(x)

## S3 method for class 'dublincore'
is.dublincore(x)
```

Arguments

`title`	dct:title, a name given to the resource. `datacite` allows the use of alternate titles, too. See `dataset_title`.
`creator`	An entity primarily responsible for making the resource. dct:creator Corresponds to `Creator` in `datacite`. See `creator`.
`identifier`	An unambiguous reference to the resource within a given context. Recommended practice is to identify the resource by means of a string conforming to an identification system. Examples include International Standard Book Number (ISBN), Digital Object Identifier (DOI), and Uniform Resource Name (URN). Select and identifier scheme from registered URI schemes maintained by IANA. More details: Guidelines for using resource identifiers in Dublin Core metadata and IEEE LOM. Similar to `Identifier` in `datacite`. See `identifier`.
`publisher`	Corresponds to dct:publisher and Publisher in DataCite. The name of the entity that holds, archives, publishes prints, distributes, releases, issues, or produces the resource. This property will be used to formulate the citation, so consider the prominence of the role. For software, use `Publisher` for the code repository. If there is an entity other than a code repository, that "holds, archives, publishes, prints, distributes, releases, issues, or produces" the code, use the property Contributor/contributorType/hostingInstitution for the code repository. See `publisher`.
`subject`	Defaults to `NULL`. See `subject` to add subject descriptions to your dataset.
`type`	The nature or genre of the resource. Recommended best practice is to use a controlled vocabulary such as the DCMI Type Vocabulary DCMITYPE. For a dataset, the correct term is `Dataset`. To describe the file format, physical medium, or dimensions of the resource, use the Format element.
`contributor`	An entity responsible for making contributions to the dataset. See DCMI: Contributor.
`date`	Corresponds to a point or period of time associated with an event in the lifecycle of the resource. dct:date. `Date` is also recommended for discovery in `datacite`,

	but it requires a different formatting.
`language`	A language of the dataset. See DCMI: Language.
`relation`	A related resource. Recommended best practice is to identify the related resource by means of a string conforming to a formal identification system. See: dct:relation. Similar to `RelatedItem` in `datacite`, which is recommended for discovery.
`format`	The file format, physical medium, or dimensions of the dataset. See DCMI: Format.
`rights`	Corresponds to dct:rights and `datacite` Rights. Information about rights held in and over the resource. Typically, rights information includes a statement about various property rights associated with the resource, including intellectual property rights. See `rights`.
`datasource`	The source of the dataset, DCMI: Source, which corresponds to a `relatedItem` in the DataCite vocabulary. We use `datasource` instead of `source` to avoid naming conflicts with the
`description`	An account of the resource. It may include but is not limited to: an abstract, a table of contents, a graphical representation, or a free-text account of the resource. dct:description. In `datacite` it is recommended for discovery. See `description`.
`coverage`	The spatial or temporal topic of the resource, spatial applicability of the dataset, or jurisdiction under which the dataset is relevant. See DCMI: Coverage.
`x`	An object that is tested if it has a class "dublincore".
`...`	Optional parameters to add to a `dublincore` object. `author=person("Jane", "Doe")` adds an author to the citation object if `type="dataset"`.

Details

The Dublin Core, also known as the Dublin Core Metadata Element Set (DCMES), is a set of fifteen main metadata items for describing digital or physical resources, such as datasets or their printed versions. Dublin Core has been formally standardized internationally as ISO 15836, as IETF RFC 5013 by the Internet Engineering Task Force (IETF), as well as in the U.S. as ANSI/NISO Z39.85.

The `ResourceType` property will be by definition "Dataset". The `Size` attribute (e.g. bytes, pages, inches, etc.) will automatically added to the dataset.

Value

`dublincore()` creates a `utils::bibentry` object extended with standard Dublin Core bibliographical metadata, `as_dublincore()` retrieves the contents of this bibentry object of a dataset_df from its attributes, and returns the contents as list, dataset_df, or bibentry object.

A logical value, if the bibliographic entries are listed according to the Dublin Core specification.

Source

DCMI Metadata Terms.

See Also

Other bibentry functions: `datacite()`, `get_bibentry()`

Examples

```
my_bibentry <- dublincore(
   title = "Iris Dataset",
   creator = person("Edgar", "Anderson", role = "aut"),
   publisher = "American Iris Society",
   datasource = "https://doi.org/10.1111/j.1469-1809.1936.tb02137.x",
   date = 1935,
   language = "en",
   description = "This famous (Fisher's or Anderson's) iris data set gives the
   measurements in centimeters of the variables sepal length and width and petal length
   and width, respectively, for 50 flowers from each of 3 species of iris.
   The species are Iris setosa, versicolor, and virginica."
  )

as_dublincore(iris_dataset, type="list")
```

`geolocation` *Get/set the Geolocation of the object.*

Description

Get/set the optional `Geolocation` property as an attribute to an R object.

Usage

```
geolocation(x)

geolocation(x, overwrite = TRUE) <- value
```

Arguments

`x`	A semantically rich data frame object created by `dataset::dataset_df` or `dataset::as_dataset_df`.
`overwrite`	If the attributes should be overwritten. In case it is set to `FALSE`, it gives a message with the current `Geolocation` property instead of overwriting it. Defaults to `TRUE` when the attribute is set to `value` regardless of previous setting.
`value`	The `Geolocation` as a character string.

Details

The `Geolocation` is recommended for discovery in DataCite 4.4. Spatial region or named place where the data was gathered or about which the data is focused. See: datacite:Geolocation.

Value

The `Geolocation` attribute as a character of length 1 is added to `x`.

See Also

Other Reference metadata functions: `description()`, `identifier()`, `language`, `publication_year()`, `publisher()`, `rights()`

Examples

```
iris_dataset <- iris
geolocation(iris_dataset) <- "US"
geolocation(iris_dataset)

geolocation(iris_dataset, overwrite = FALSE) <- "GB"
```

`get_bibentry` *Get/set the Bibentry of the object.*

Description

The `dataset_df` objects contain among their attributes bibliographic entries which are stored in a `utils::bibentry` object. Upon creation, these entries are filled with default values when applicable.

To retrieve the bibentry of a dataset_df object, use `get_bibentry`.

To create a new bibentry, use the `datacite` function for an interface and default values according to the DataCite standard, or the `dublincore` function for the more general Dublin Core standard.

To change or an entire new bibliographic entry to a dataset_df object (or any data.frame-like object), use the `` `set_bibentry<-` `` function (see examples.) For more details, please check the `vignette("bibentry", package="dataset")` vignette.

Usage

```
get_bibentry(dataset)

set_bibentry(dataset) <- value
```

Arguments

`dataset`	A dataset created with `dataset_df`.
`value`	A `utils::bibentry` object, or a newly initialised bibentry object with DataCite default values for unassigned entries.

Value

The `get_bibentry` returns from the `bibentry` object of `x` from its attributes; the `` `set_bibentry<-` `` assignment function sets this attribute to `value` and invisibly returns `x` with the changed attributes. To set well-formatted input `value`, refer to `datacite` or `dublincore` (see Details.)

See Also

Other bibentry functions: `datacite()`, `dublincore()`

Examples

```
# Get the bibentry of a dataset_df object:
iris_bibentry <- get_bibentry(iris_dataset)

# Create a well-formatted bibentry object:
alternative_bibentry <- datacite(
     Creator=person("Jane Doe"),
     Title ="The Famous Iris Dataset",
     Publisher = "MyOrg")

# Assign the new bibentry object:
set_bibentry(iris_dataset) <- alternative_bibentry

# Print the bibentry object according to the DataCite notation:
as_datacite(iris_dataset, "list")

# Print the bibentry object according to the Dublin Core notation:
as_dublincore(iris_dataset, "list")
```

`identifier` *Get/set the Identifier of the object.*

Description

Add the optional Identifier property as an attribute to an R object.

Usage

```
identifier(x)

identifier(x, overwrite = TRUE) <- value
```

Arguments

`x`	An R object, such as a data.frame, a tibble, or a data.table.
`overwrite`	If the attributes should be overwritten. In case it is set to `FALSE`, it gives a message with the current `Identifier` property instead of overwriting it. Defaults to `TRUE` when the attribute is set to `value` regardless of previous setting.
`value`	The `Identifier` as a character string.

Details

The Identifier is an unambiguous reference to the resource within a given context. Recommended practice is to identify the resource by means of a string conforming to an identification system. Examples include International Standard Book Number (ISBN), Digital Object Identifier (DOI), and Uniform Resource Name (URN). Select and identifier scheme from registered URI schemes maintained by IANA. More details: Guidelines for using resource identifiers in Dublin Core metadata and IEEE LOM. Similar to `Identifier` in `datacite`. DataCite 4.4.
It is not part of the "core" Dublin Core terms, but we always add it to the metadata attributes of a dataset (in case you use a strict Dublin Core property sheet you can omit it.) Dublin Core metadata terms.

Value

The `Identifier` attribute as a character of length 1 is added to `x`.

See Also

Other Reference metadata functions: `description()`, `geolocation()`, `language`, `publication_year()`, `publisher()`, `rights()`

Examples

```
identifier(iris_dataset) <- "https://doi.org/10.1111/j.1469-1809.1936.tb02137.x"
identifier(iris_dataset)
```

`id_to_column` *Add identifier to columns*

Description

Add a prefixed identifier to the first column of the dataset.

Usage

```
id_to_column(x, prefix = "eg:", ids = NULL)
```

Arguments

`x`	A dataset created with `dataset_df`.
`prefix`	Defaults to `eg:` (example.com).
`ids`	Defaults to `NULL`.

Value

A dataset conforming the original sub-class of `x`.

Examples

```
# Example with a dataset_df object:
id_to_column(iris_dataset)

# Example with a data.frame object:#'
id_to_column(iris, prefix="eg:iris-o")
```

---

iris_dataset *Edgar Anderson's Iris Data*

Description

This famous (Fisher's or Anderson's) iris data set gives the measurements in centimetres of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are *Iris setosa*, *versicolor*, and *virginica*. This is a replication of `datasets::iris` as *dataset* s3 class.

Usage

```
iris_dataset
```

Format

iris is a data frame with 150 cases (rows) and 6 variables (columns) named rowid, Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, and Species.

Details

See `datasets::iris` for details.

Source

Fisher, R. A. (1936) The use of multiple measurements in taxonomic problems. Annals of Eugenics, **7**, Part II, p179–188.

The data were collected by Anderson, Edgar (1935). The irises of the Gaspe Peninsula, Bulletin of the American Iris Society, **59**, 2–5.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) The New S Language. Wadsworth & Brooks/Cole.

`language` *Get/Set the primary language of the dataset*

Description

Add the optional Language property as an attribute to an R object.

Usage

```
language(x)

language(x, iso_639_code = "639-3") <- value
```

Arguments

`x`	A semantically rich data frame object created by `dataset_df` or `as_dataset_df`.
`iso_639_code`	Defaults to `ISO 639-3`, alternative is `ISO 639-1`.
`value`	The language to be added to the object attributes, added by name, or as a 2- or 3-character code for the language. You can add a language code or language name, and the parameter is normalized to `tolower(language)`. (The ISO 639 standard capitalizes language names and uses lower case for the codes.)

Details

Language is an optional property in DataCite 4.4; see: datacite:Language
It is a part of the "core" of the Dublin Core metadata terms. The language parameter is validated against the `[ISOcodes]{ISO_639_2}` table.
The attribute `language` is added to the object. It will be exported into DataCite applications in a capitalized `Lanugage` format.

Value

The Language is added to the `x` as `ISO 639-1`, the Datacite recommendation, or `ISO 639-3` used by the Zenodo data repository.

See Also

Other Reference metadata functions: `description()`, `geolocation()`, `identifier()`, `publication_year()`, `publisher()`, `rights()`

Examples

```
myiris <- iris_dataset
language(myiris) <- "English"
language(myiris)
language(myiris) <- "fr"
language(myiris)
```

n_triple *Create an N-Triple*

Description

Create a single N-Triple triple.

Usage

```
n_triple(s, p, o)
```

Arguments

s	The subject of a triplet.
p	The predicate of a triplet.
o	The object of a triplet.

Details

N-Triples is an easy to parse line-based subset of Turtle to serialize RDF. An N-Triple triple is a sequence of RDF terms representing the subject, predicate and object of an RDF Triple. Use `n_triples` to serialize multiple statements.

Value

A character vector containing one N-Triple string.

Source

RDF 1.1 N-Triples

Examples

```
s <- "http://example.org/show/218"
p <- "http://www.w3.org/2000/01/rdf-schema#label"
o <- "That Seventies Show"
n_triple(s, p, o)
```

n_triples — *Create N-Triples*

Description

Create triple statements to annotate your dataset with standard, interoperable metadata.

Usage

```
n_triples(triples)
```

Arguments

triples	Concatenated N-Triples created with n_triple.

Details

N-Triples is an easy to parse line-based subset of Turtle to serialize RDF. See RDF 1.2 N-Triples. A line-based syntax for an RDF graph.

Value

A character vector containing unique N-Triple strings.

Examples

```
triple_1 <- n_triple("http://example.org/show/218",
                      "http://www.w3.org/2000/01/rdf-schema#label",
                      "That Seventies Show")
triple_2 <- n_triple("http://example.org/show/218",
                     "http://example.org/show/localName",
                     '"Cette Série des Années Septante"@fr-be')
n_triples(c(triple_1, triple_2, triple_1))
```

provenance — *Get or update provenance information*

Description

Add or update information about the history (provenance) of the dataset.

Usage

```
provenance(x)

provenance(x) <- value
```

Arguments

`x`	A dataset created with `dataset_df`.
`value`	Use `n_triples` to add further statement values.

Value

`provenance(x)` returns the provenance attributes created by `n_triples` as a text; `provenance(x)<-value` adds the new provenance attributes and returns `x` invisibly.

Examples

```
provenance(iris_dataset)

## add a statement:

 provenance(iris_dataset) <- n_triple(
  "https://doi.org/10.5281/zenodo.10396807",
  "http://www.w3.org/ns/prov#wasInformedBy",
  "http://example.com/source#1")
```

`publication_year` *Get/set the publication_year of the object.*

Description

Get/set the optional `publication_year` property as an attribute to an R object.

Usage

```
publication_year(x)

publication_year(x, overwrite = TRUE) <- value
```

Arguments

`x`	A semantically rich data frame object created by `dataset::dataset_df` or `dataset::as_dataset_df`.
`overwrite`	If the attributes should be overwritten. In case it is set to `FALSE`, it gives a message with the current `PublicationYear` property instead of overwriting it. Defaults to `TRUE` when the attribute is set to `value` regardless of previous setting.
`value`	The publication_year as a character set.

Details

The `PublicationYear` is the year when the data was or will be made publicly available in `YYYY` format. See Publication Year: DataCite Additional Guidance.

Value

Returns the `year` metadata field of the `DataBibentry` of the dataset

See Also

Other Reference metadata functions: `description()`, `geolocation()`, `identifier()`, `language`, `publisher()`, `rights()`

Examples

```
publication_year(iris_dataset)
publication_year(iris_dataset) <- 1936
```

`publisher` *Get/set the Publisher of the object.*

Description

Add the optional `Publisher` property as an attribute to an R object.

Usage

```
publisher(x)

publisher(x, overwrite = TRUE) <- value
```

Arguments

`x`	A dataset object created with `dataset::dataset_df` or `dataset::as_dataset_df`.
`overwrite`	If the attributes should be overwritten. In case it is set to `FALSE`,it gives a warning with the current `publisher` property instead of overwriting it. Defaults to `FALSE`.
`value`	The `Publisher` as a character set.

Details

The `Publisher` corresponds to dct:publisher and Publisher in DataCite. The name of the entity that holds, archives, publishes prints, distributes, releases, issues, or produces the resource. This property will be used to formulate the citation, so consider the prominence of the role. For software, use Publisher for the code repository. If there is an entity other than a code repository, that "holds, archives, publishes, prints, distributes, releases, issues, or produces" the code, use the property Contributor/contributorType/ hostingInstitution for the code repository.

Value

The Publisher attribute as a character of length 1 is added to `x`.

See Also

Other Reference metadata functions: `description()`, `geolocation()`, `identifier()`, `language`, `publication_year()`, `rights()`

Examples

```
publisher(iris_dataset) <- "American Iris Society"
publisher(iris_dataset)
```

`rights` *Get/set the Rights of the object.*

Description

Get/set the optional `Rights` property as an attribute to an R object.

Usage

```
rights(x)

rights(x, overwrite = FALSE) <- value
```

Arguments

`x`	A semantically rich data frame object created by `dataset::dataset_df` or `dataset::as_dataset_df`.
`overwrite`	If the `Rights` attribute should be overwritten. In case it is set to `FALSE`, it gives a message with the current `Rights` property instead of overwriting it. Defaults to `FALSE`.
`value`	The `Rights` as a character set.

Details

`Rights` corresponds to dct:rights and `datacite` Rights. Information about rights held in and over the resource. Typically, rights information includes a statement about various property rights associated with the resource, including intellectual property rights.

Value

The `Rights` attribute as a character of length 1 is added to `x`.

See Also

Other Reference metadata functions: `description()`, `geolocation()`, `identifier()`, `language`, `publication_year()`, `publisher()`

Examples

```
rights(iris_dataset) <- "CC-BY-SA"
rights(iris_dataset)
```

`subject` *Create/add/retrieve a subject*

Description

Create/add/retrieve a subject

Usage

```
subject(x)

subject_create(
  term,
  schemeURI = NULL,
  valueURI = NULL,
  prefix = NULL,
  subjectScheme = NULL,
  classificationCode = NULL
)

subject(x) <- value

is.subject(x)
```

Arguments

`x`	A dataset object created with `dataset_df` or `dataset::as_dataset_df`.
`term`	A subject term, for example, `"Data sets"`.
`schemeURI`	The URI of the subject identifier scheme, for example `"http://id.loc.gov/authorities/subjects"`
`valueURI`	The URI of the subject term. `"https://id.loc.gov/authorities/subjects/sh2018002256"`
`prefix`	An abbreviated prefix of a scheme URI, for example, `"lcch:"` representing `"http://id.loc.gov/authorities/subjects"`. Widely used namespaces (schemes) have conventional abbreviations.
`subjectScheme`	The name of the subject scheme or classification code or authority if one is used. It is a namespace.
`classificationCode`	The classificationCode subproperty may be used for subject schemes, like ANZSRC, which do not have valueURIs for each subject term.
`value`	A subject field created by `subject`. The subject field is overwritten with this value.

Details

The subject class and its function record the subject property of the dataset. The DataCite definition allows the use of multiple subproperties, however, these cannot be added to the standard `utils::bibentry` object. Therefore, if the user sets the value of the subject field to a character string, it is added to the bibentry of the dataset, and also to a separate `subject` attribute. If the user wants to use the more detailed subproperties (see examples with `subject_create`), then the subject$term value is added to the bibentry as a text, and the more complex subject object is added as a separate attribute to the dataset_df object.#'

Value

`subject(x)` returns the subject attribute of the `dataset_df` object `x`, `subject(x)<-value` sets the same attribute to `value` and invisibly returns the `x` object with the changed attributes.

A `subject_create` returns a named list with the subject term, the subject scheme, URIs and prefix.

`is.subject` returns a logical value, `TRUE` if the subject as a list is well-formatted by `subject_create` with its necessary key-value pairs.

Examples

```
# To set the subject of a dataset_df object:
subject(iris_dataset) <- subject_create(
         term  = "Irises (plants)",
         schemeURI = "http://id.loc.gov/authorities/subjects",
         valueURI = "https://id.loc.gov/authorities/subjects/sh85068079",
         subjectScheme = "LCCH",
         prefix = "lcch:")

# To retrieve the subject with its subproperties:
subject(iris_dataset)
```

`var_definition` *Get / set a definition for a vector or a dataset*

Description

Get / set a definition for a vector or a dataset

Usage

```
var_definition(x, ...)

var_definition(x) <- value

definition_attribute(x)

get_definition_attribute(x)
```

```
set_definition_attribute(x, value)

definition_attribute(x) <- value
```

Arguments

x	a vector
...	Further parameters for inheritance, not in use.
value	a character string or NULL to remove the definition of measure.

Details

`get_variable_definitions()` is identical to `var_definition()`.

Value

The (linked) definition of the meaning of the data contained by a vector constructed with `defined`.

Examples

```
small_country_dataset <- dataset_df(
  country_name = defined(c("Andorra", "Lichtenstein"), label  = "Country"),
  gdp = defined(c(3897, 7365),
                       label = "Gross Domestic Product",
                       unit = "million dollars")
)
var_definition(small_country_dataset$country_name) <- "http://data.europa.eu/bna/c_6c2bb82d"
var_definition(small_country_dataset$country_name)
# To remove a definition of measure
var_definition(small_country_dataset$country_name) <- NULL
```

---

var_label *Get / Set a variable label*

Description

Add a human readable, easier to understand label as a metadata attribute to a variable or vector than the programmatic vector object name, or column name in the data frame.

Usage

```
## S3 method for class 'defined'
var_label(x, ...)

## S3 method for class 'dataset_df'
var_label(
  x,
  unlist = FALSE,
  null_action = c("keep", "fill", "skip", "na", "empty"),
  recurse = FALSE,
  ...
)

label_attribute(x)

## S3 replacement method for class 'defined'
var_label(x) <- value

## S3 replacement method for class 'dataset_df'
var_label(x) <- value
```

Arguments

`x`	a vector or a data.frame
`...`	Further potential parameters reserved for inherited classes.
`unlist`	for data frames, return a named vector instead of a list
`null_action`	for data frames, by default `NULL` will be returned for columns with no variable label. Use `"fill"` to populate with the column name instead, `"skip"` to remove such values from the returned list, `"na"` to populate with `NA` or `"empty"` to populate with an empty string (`""`).
`recurse`	if `TRUE`, will apply `var_label()` on packed columns (see `tidyr::pack()`) to return the variable labels of each sub-column; otherwise, the label of the group of columns will be returned.
`value`	a character string or `NULL` to remove the label For data frames, with `var_labels()`, it could also be a named list or a character vector of same length as the number of columns in `x`.

Details

See `labelled::var_label` for details about variable labels.
See `vignette("defined", package = "dataset")` to use comprehensively with variable labels, namespaces, units of measures, and machine-independent permanent variable identifiers.

Value

`var_label()` returns returns the `label` attribute as a character string. The `var_label<-` assignment method allows to add, remove, or overwrite this attribute on a vector `x`. The assignment function returns the `x` vector invisibly.

See Also

Other defined metadata methods and functions: `defined()`, `var_namespace()`, `var_unit()`

Examples

```
iris_dataset_2 <- iris_dataset

# Retrieve the label attribute:
var_label(iris_dataset_2$Sepal.Length)
```

`var_namespace` *Get / Set a namespace of measure*

Description

Retain the namespace part of a permanent, global variable identifier which is independent of the R instance in use.

Usage

```
var_namespace(x, ...)

var_namespace(x) <- value

get_variable_namespaces(x, ...)

namespace_attribute(x)

get_namespace_attribute(x)

set_namespace_attribute(x, value)

namespace_attribute(x) <- value
```

Arguments

`x`	a vector
`...`	Further potential parameters reserved for inherited classes.
`value`	a character string or `NULL` to remove the namespace of measure.

Details

The namespace attribute is useful when users join or concatenate data from remote, linked, and open data sources. In such cases, variable identifiers (labels or names) are often resolved with a common namespace prefix, which, together with the namespace, forms a URI or IRI permanent identifier for the variable. Retaining the namespace in such cases allows cross-validation or success later updates of the vector (as a column of a dataset.)

`get_variable_namespaces()` is identical to `var_namespace()`. See `vignette("defined", package = "dataset")` to use comprehensively with variable labels, namespaces, units of measures, and machine-independent permanent variable identifiers.

Value

The namespace attribute of a vector constructed with `defined`.

See Also

Other defined metadata methods and functions: `defined()`, `var_label()`, `var_unit()`

Examples

```
qid = defined(c("Q275912", "Q116196078"), namespace = "https://www.wikidata.org/wiki/")
var_namespace(qid)

# To remove a namespace
var_namespace(qid) <- NULL
```

`var_unit` *Get / Set a unit of measure*

Description

Get / Set a unit of measure

Usage

```
var_unit(x, ...)

var_unit(x) <- value

get_variable_units(x, ...)

unit_attribute(x)

get_unit_attribute(x)

set_unit_attribute(x, value)

unit_attribute(x) <- value
```

Arguments

`x`	A vector.
`...`	Further potential parameters reserved for inherited classes.
`value`	A character string or `NULL` to remove the unit of measure.

Details

The aim of the `unit` attribute is to add to the R vector object its unit of measure (for example, physical units like gram and kilogram or currency units like dollars or euros), so that they are not concatenated or joined in a syntactically correct but semantically incorrect way (i.e., accidentally concatenating values quoted in dollars and euros from different subvectors.) This is particularly useful when working with linked open data, i.e., when joins or concatenations are performed on data arriving from a remote source.

`get_variable_units()` is identical to `var_unit()`.

See `vignette("defined", package = "dataset")` to use comprehensively with variable labels, namespaces, units of measures, and machine-independent permanent variable identifiers.

Value

The unit attribute of a vector constructed with `defined`, or any vector that is enriched with a unit attribute.

The `var_unit<-` assignment method allows to add, remove, or overwrite this attribute on a vector `x`. The assignment function returns the `x` vector invisibly.

See Also

Other defined metadata methods and functions: `defined()`, `var_label()`, `var_namespace()`

Examples

```
# The defined vector class and dataset_df support units of measure attributes:
var_unit(iris_dataset$Sepal.Length)

# Normally columns of a data.frame do not have a unit attribute:
var_unit(iris$Sepal.Length)

# You can add them with the assignment function:
var_unit(iris$Sepal.Length) <- "centimeter"

# To remove a unit of measure assign the NULL value:
var_unit(iris$Sepal.Length) <- NULL
```

`xsd_convert` *Convert to XML Schema Definition (XSD) types*

Description

Convert the numeric, boolean and Date/time columns of a dataset `xs:decimal`, `xsLboolean`, `xs:date` and `xs:dateTime`.

Usage

```
xsd_convert(x, idcol, ...)

## S3 method for class 'data.frame'
xsd_convert(x, idcol = NULL, ...)

## S3 method for class 'dataset'
xsd_convert(x, idcol = NULL, ...)

## S3 method for class 'tibble'
xsd_convert(x, idcol = NULL, ...)

## S3 method for class 'character'
xsd_convert(x, idcol = NULL, ...)

## S3 method for class 'numeric'
xsd_convert(x, idcol = NULL, ...)

## S3 method for class 'haven_labelled_defined'
xsd_convert(x, idcol = NULL, ...)

## S3 method for class 'integer'
xsd_convert(x, idcol = NULL, ...)

## S3 method for class 'logical'
xsd_convert(x, idcol = NULL, ...)

## S3 method for class 'factor'
xsd_convert(x, idcol = NULL, ...)

## S3 method for class 'POSIXct'
xsd_convert(x, idcol = NULL, ...)

## S3 method for class 'Date'
xsd_convert(x, idcol = NULL, ...)
```

Arguments

`x`	An object to be coerced to an XLM Schema defined string format.
`idcol`	The name or position of the column that contains the row (observation) identifiers. If `NULL`, it will make a new `idcol` from `row.names()`.
`...`	Further optional parameters for generic method.

Value

A serialisation of an R vector or data frame (dataset) in XML.

Examples

```
# Convert data.frame to XML Schema Definition
xsd_convert(head(iris))

# Convert dataset to XML Schema Definition
xsd_convert(head(iris_dataset))
# Convert integers or doubles, numbers:
xsd_convert(1:3)
# Convert logical values:
xsd_convert(TRUE)
```

Index