Package: datapackage (via r-universe)

March 13, 2025

Type Package

Title Creating and Reading Data Packages

Version 0.1.1

Description Open, read data from and modify Data Packages. Data Packages are an open standard for bundling and describing data sets (<https://datapackage.org>). When data is read from a Data Package care is taken to convert the data as much a possible to R appropriate data types. The package can be extended with plugins for additional data types.

BugReports https://github.com/djvanderlaan/datapackage/issues

URL https://github.com/djvanderlaan/datapackage

Depends R (>= 4.1.0)

Imports yaml, jsonlite, iso8601, utils, tools, methods

Suggests simplermarkdown, data.table, codelist, LaF

VignetteBuilder simplermarkdown

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Repository https://djvanderlaan.r-universe.dev

RemoteUrl https://github.com/djvanderlaan/datapackage

RemoteRef HEAD

RemoteSha a424ce3034ef521fec0e1d4c5da0cf711db3a270

Contents

csv_reader *Read the CSV-data for a Data Resource*

Description

Read the CSV-data for a Data Resource

Usage

```
csv_reader(
  path,
  resource,
  use_fread = FALSE,
  convert_categories = c("no", "to_factor"),
  as_connection = FALSE,
  ...
)
```

Arguments

path	path to the data set.
resource	a Data Resource.
use_fread	use the fread function instead of read.csv and return a data.table.
convert_categories	how to handle columns for which the field descriptor has a categories property. Passed on to dp_apply_schema.
as_connection	This argument is ignored. The function will always return a data.frame.
...	additional arguments are passed on to read.csv or fread. Note that some arguments are already set by csv_reader, so not all arguments are available to use as additional arguments.

Value

Returns a data.frame with the data.

See Also

Generally used by calling dp_get_data.

csv_writer *Write data of data resource to CSV-file*

Description

Write data of data resource to CSV-file

Usage

```
csv_writer(x, resource_name, datapackage, use_fwrite = FALSE, ...)
```

Arguments

x	data.frame with the data to write
resource_name	name of the data resource in the data package.
datapackage	the Data Package to which the file should be written.
use_fwrite	write the file using fwrite from the data.table package.
...	ignored for now

Value

The function doesn't return anything. It is called for it's side effect of creating CSV-files in the directory of the data package.

dp_add_reader *Add a reader function for a specific format*

Description

Add a reader function for a specific format

Usage

```
dp_add_reader(
  format,
  reader,
  mediatypes = character(0),
  extensions = character(0)
)
```

Arguments

format	the data format read by the reader. Should be a length 1 character vector.
reader	the reader function. See details.
mediatypes	a character vector with the media-types that are used for the format.
extensions	a character vector with typical file extensions used by the format.

Details

Adds a reader for a given format. The reader is added to a list of reades references by the format. It is also possible to assign mediatypes and file extensions to the format. When the format for a given Data Resource is missing, dp_get_data will first check if a mediatype is associated with the resource and will try to look up which format belongs to the fiven mediatype. If that doesn't result in a valid format, dp_get_data will try the same with the extension of the file.

Note that adding a reader for an existing format will overwrite the existing reader.

Value

Does not return anything (invisible(NULL)).

Examples

```
# Add a very simple reader for json
json_reader <- function(path, resource, ...) {
  lapply(path, function(fn) {
    jsonlite::read_json(fn)
  })
}

dp_add_reader("json", json_reader, c("application/json"), "json")
```

---

dp_add_writer *Add a writer function for a specific format*

Description

Add a writer function for a specific format

Usage

```
dp_add_writer(format, writer)
```

Arguments

format	the data format read by the writer Should be a length 1 character vector.
writer	the writer function. See details.

Details

Adds a writer for a given format. The writer is added to a list of writers referenced by the format. The writer function should accept 'data' with the data as its first argument, 'resource_name' the name of the resource to which the data set belongs, 'datapackage' that datapackage to which the data should be written.

Note that adding a writer for an existing format will overwrite the existing writer

Value

Does not return anything (`invisible(NULL)`).

Examples

```
# Add a very simple writer for json
json_writer <- function(data, resource_name, datapackage, ...) {
  dataresource <- dp_resource(datapackage, resource_name)
  path <- dp_path(dataresource, full_path = TRUE)
  jsonlite::write_json(data, path)
}

dp_add_writer("json", json_writer)
```

`dp_apply_schema` *Convert columns of data.frame to their correct types using table schema*

Description

Convert columns of data.frame to their correct types using table schema

Usage

```
dp_apply_schema(
  dta,
  resource,
  convert_categories = c("no", "to_factor", "to_code"),
  ...
)
```

Arguments

`dta`	a `data.frame` or `data.table`.
`resource`	an object with the Data Resource of the data set.
`convert_categories`	how to handle columns for which the field descriptor has a `categories` property. This should either be the strings "no", "to_factor", "to_code", the name of a function or a function. When equal to "no" the field is returned as is; when equal to "to_factor" each column is transformed using `dp_to_factor`; when equal to "to_code" each column is transformed using `dp_to_code`. In other cased the function is called with the column as its first parameter and `warn = FALSE` as its second argument. The result of this function call is added to the resulting data set.
`...`	additional arguments are passed on to the `dp_to_<fieldtype>` functions (e.g. `dp_to_number`).

Details

Converts each column in `dta` to the correct R-type using the type information in the table schema. For example, if the original column type in `dta` is a character vector and the table schema specifies that the field is of type number, the column is converted to numeric using the decimal separator and thousands separator specified in the field descriptor (or default values for these if not).

Value

Returns a copy of the input data.frame with columns modified to match the types given in de table schema.

See Also

This function calls conversion functions for each of the columns, see `dp_to_number`, `dp_to_boolean`, `dp_to_integer`, `dp_to_date`. `dp_to_datetime`, `dp_to_yearmonth`, and `dp_to_string`.

---

dp_categorieslist *Get the a data.frame with the categories for a variable.*

Description

Get the a data.frame with the categories for a variable.

Usage

```
dp_categorieslist(x, ...)

## Default S3 method:
dp_categorieslist(
  x,
  fielddescriptor = attr(x, "fielddescriptor"),
  datapackage = dp_get_datapackage(fielddescriptor),
  ...
)

## S3 method for class 'fielddescriptor'
dp_categorieslist(
  x,
  datapackage = dp_get_datapackage(x),
  normalised = FALSE,
  ...
)
```

Arguments

x	the variable for which to get the Categories List
...	used to pass extra arguments on to other methods.
fielddescriptor	the Field Descriptor associated with the variable.
datapackage	the Data Package where the variable is from.
normalised	if TRUE the column with values will be named value and the the columnd with labels label.

Value

Returns a data.frame with the categories or NULL when none could be found.

---

dp_check_dataresource *Check if a data set is valid given a Data Resource*

Description

Check if a data set is valid given a Data Resource

Usage

```
dp_check_dataresource(
  x,
  dataresource = attr(x, "resource"),
  constraints = TRUE,
  throw = FALSE,
  tolerance = sqrt(.Machine$double.eps)
)
```

Arguments

x	data.frame to check
dataresource	dataresource object to check x against.
constraints	also check relevant constraints in the field descriptor.
throw	generate an error if the data set is not valid according to the dataresource.
tolerance	numerical tolerance used in some of the tests

Value

Returns TRUE when the field is valid. Returns a character vector with length >= 1 if the field is not valid. The text in the character values indicates why the field is not valid.

When throw = TRUE the function will generate an error instead of returning a character vector. When the dataset is valid the function returns TRUE invisibly.

See Also

Use isTRUE to check if the test was successful. See dp_check_field for a function that checks a column or vector.

`dp_check_field` *Check if a vector is valid given a field descriptor*

Description

Check if a vector is valid given a field descriptor

Usage

```
dp_check_field(
  x,
  fielddescriptor,
  constraints = TRUE,
  tolerance = sqrt(.Machine$double.eps)
)
```

Arguments

x	vector to test
fielddescriptor	field descriptor to test the vector against
constraints	also check relevant constraints in the field descriptor.
tolerance	numerical tolerance used in some of the tests

Value

Returns TRUE when the field is valid. Returns a character vector with length >= 1 if the field is not valid. The text in the character values indicates why the field is not valid.

See Also

Use isTRUE to check if the test was successful. See dp_check_dataresource for a function that checks a complete data set.

dp_field — *Get the Field Descriptor associated with a certain field in a Data Resource*

Description

Get the Field Descriptor associated with a certain field in a Data Resource

Usage

```
dp_field(x, field_name)
```

Arguments

x	a dataresource or tableschema object.
field_name	length one character vector with the name of the field.

Value

An object of type fielddescriptor.

dp_field_names — *List the fields in a Data Resource*

Description

List the fields in a Data Resource

Usage

```
dp_field_names(x)
```

Arguments

x	object for which to get the field names. This can either be a Data Resource or Table Schema.

Value

Returns a character vector with the fields in the Data Resource.

dp_generate_dataresource *Generate Data Resource for a dataset*

Description

Generate Data Resource for a dataset

Usage

```
dp_generate_dataresource(
  x,
  name,
  path = paste0(name, getextension(format)),
  format = "csv",
  mediatype = getmediatype(format),
  use_existing = FALSE,
  categories_type = c("regular", "resource"),
  categorieslist = iscategorieslist(x),
  ...
)
```

Arguments

x	data.frame for which to generate the Data Resources.
name	name of the Data Resource
path	name of the file in which to store the dataset. This should be a path relative to the location of the directory in which the Data Package in which the Data Resource will be stored.
format	the data format in which the data is stored.
mediatype	mediatype of the data
use_existing	use existing field descriptors if present.
categories_type	how should categories be stored. See dp_generate_fielddescriptor.
categorieslist	does the data resource function as a categories list for fields in another data resource
...	Currently ignored

Value

Returns a Data Resource object.

Note that this function does not create the file at path. The export of the Data Resource is automatically set to CSV.

Examples

```
# generate an example dataset
dta <- data.frame(a = 1:3, b = factor(letters[1:3]))
resource <- dp_generate_dataresource(dta, "example")
print(resource)
```

---

dp_generate_fielddescriptor *Generate a fielddescriptor for a given variable in a dataset*

Description

Generate a fielddescriptor for a given variable in a dataset

Usage

```
dp_generate_fielddescriptor(x, name, ...)

## Default S3 method:
dp_generate_fielddescriptor(x, name, ...)

## S3 method for class 'numeric'
dp_generate_fielddescriptor(
  x,
  name,
  use_existing = TRUE,
  use_categories = TRUE,
  categories_type = c("regular", "resource"),
  ...
)

## S3 method for class 'integer'
dp_generate_fielddescriptor(
  x,
  name,
  use_existing = TRUE,
  use_categories = TRUE,
  categories_type = c("regular", "resource"),
  ...
)

## S3 method for class 'logical'
dp_generate_fielddescriptor(
  x,
  name,
  use_existing = TRUE,
```

```
  use_categories = TRUE,
  categories_type = c("regular", "resource"),
  ...
)

## S3 method for class 'Date'
dp_generate_fielddescriptor(
  x,
  name,
  use_existing = TRUE,
  use_categories = TRUE,
  categories_type = c("regular", "resource"),
  ...
)

## S3 method for class 'character'
dp_generate_fielddescriptor(
  x,
  name,
  use_existing = TRUE,
  use_categories = TRUE,
  categories_type = c("regular", "resource"),
  ...
)

## S3 method for class 'factor'
dp_generate_fielddescriptor(
  x,
  name,
  use_existing = TRUE,
  use_categories = TRUE,
  categories_type = c("regular", "resource"),
  ...
)
```

Arguments

x	vector for which to generate the fielddescriptor
name	name of the field in the dataset.
...	used to pass extra arguments to methods.
use_existing	use existing field descriptor if present (assumes this is stored in the 'fielddescriptor' attribute).
use_categories	do not generate a categories field except when x is a factor.
categories_type	how should categories be stored. Note that type "resource" is not officially part of the standard.

Value

Returns a `fielddescriptor`.

`dp_get_connection` *Get a connection to the data belonging to a Data Resource*

Description

Get a connection to the data belonging to a Data Resource

Usage

```
dp_get_connection(x, ...)
```

Arguments

`x`	Can either be a Data Resource or Data Package.
`...`	Extra arguments are passed on to `dp_get_data`.

Details

When `x` is a Data Package a additional argument `resource_name` is needed to identify the correct Data Resource. See `dp_get_data`.

This function calls `dp_get_data` with an additional `as_connection = TRUE)` argument.

Value

Depending on the type of Data Resource a connection to the data is returned. Some readers will return the data set as a `data.frame`.

`dp_get_data` *Get the data belonging to a Data Resource*

Description

Get the data belonging to a Data Resource

Usage

```
dp_get_data(x, ..., as_connection = FALSE)

## S3 method for class 'dataresource'
dp_get_data(x, reader = "guess", ..., as_connection = FALSE)

## S3 method for class 'datapackage'
dp_get_data(x, resource_name, reader = "guess", ..., as_connection = FALSE)
```

Arguments

x	a dataresource or datapackage object.
...	passed on to the reader
as_connection	Try to return a connection to the data instead of the data itself. When a reader does not support returning connections it will return the data.
reader	the reader to use to read the data. This should be either a function accepting the path to the data set (a character vector with possibly multitple filenames) and the Data Resource as second argument, or the character string "guess".
resource_name	the name of the dataresource.

Details

When reader = "guess" the function will try to guess which reader to use based on the format and mediatype of the Data Resource. Currently only CSV is supported. For other data types a custom reader has to be provided unless the data is stored inside the Data Resource object.

It is also possible to assign default readers for data formats. For that see dp_add_reader.

Value

Will return the data. This will generally be a data.frame but depending on the file type can also be other types of R-objects.

When called with the as_connection = TRUE argument, it will try to return a connection to the data. This depends on the reader. When the reader does not support returning connections it will return the data.

See Also

dp_get_connection is a wrapper around dp_get_data(..., as_connection = TRUE).

---

dp_get_datapackage *Get the Data Package associated with the object*

Description

Get the Data Package associated with the object

Usage

```
dp_get_datapackage(x)
```

Arguments

x	the object for which to determine the associated Data Package

Details

This method can, of course, only determine the Data Package when this information is stored in one of the attributes of the object. This can be either be a `datapackage` attribute or an `dataresource` attribute.

Value

Returns a Data Resource object, or returns `NULL` when none could be found.

---

dp_load_from_datapackage

Quickly read a dataset from a Data Package

Description

Quickly read a dataset from a Data Package

Usage

```
dp_load_from_datapackage(path, resource_name, ...)
```

Arguments

`path`	the directory with the Data Package
`resource_name`	the name of the Data Resource. When omitted the Data Resource with the same name as the Data Package is read in and when no such resource exists the first Data Resource is read in.
`...`	passed on to `dp_get_data`.

Details

This function is a wrapper around `open_datapackage` and `dp_get_data`. It offers a quick way to read in a dataset from a Data Package.

Value

Returns a dataset. Usually a `data.frame`.

dp_nresources *Return the number of resources in a Data Package*

Description

Return the number of resources in a Data Package

Usage

```
dp_nresources(dp)
```

Arguments

dp	A Data Package object.

Value

Returns an integer with the number of resources in the Data Package.

dp_properties *Get a list of properties defined for the object*

Description

Get a list of properties defined for the object

Usage

```
dp_properties(x)

## S3 method for class 'readonlydatapackage'
dp_properties(x)

## S3 method for class 'editabledatapackage'
dp_properties(x)

## S3 method for class 'dataresource'
dp_properties(x)

## S3 method for class 'tableschema'
dp_properties(x)
```

Arguments

x	the object for which to obtain the properties

Value

Returns a character vector (possibly zero length) with the names of the properties.

See Also

The dp_property method can be used to get the values of the properties.

---

dp_property *Get and set properties of Data Packages and Data Resources*

Description

Get and set properties of Data Packages and Data Resources

Usage

```
dp_property(x, attribute)

## S3 method for class 'readonlydatapackage'
dp_property(x, attribute)

## S3 method for class 'editabledatapackage'
dp_property(x, attribute)

dp_property(x, attribute) <- value

## S3 replacement method for class 'readonlydatapackage'
dp_property(x, attribute) <- value

## S3 replacement method for class 'editabledatapackage'
dp_property(x, attribute) <- value

## S3 method for class 'dataresource'
dp_property(x, attribute)

## S3 replacement method for class 'dataresource'
dp_property(x, attribute) <- value

## S3 method for class 'tableschema'
dp_property(x, attribute)

## S3 replacement method for class 'tableschema'
dp_property(x, attribute) <- value

## S3 method for class 'fielddescriptor'
dp_property(x, attribute)
```

```
## S3 replacement method for class 'fielddescriptor'
dp_property(x, attribute) <- value
```

Arguments

x	a datapackage or dataresource object.
attribute	a length 1 character vector with the name of the property.
value	the new value of the property.

Value

Either returns the property or modifies the object.

See Also

See dp_name etc. for methods for specific properties for Data Packages and dp_encoding etc. for specific properties for Data Resources. These specific methods also check if the input is valid for the given property.

---

dp_resource *Modifying the resources of a Data Package*

Description

Modifying the resources of a Data Package

Usage

```
dp_resource(x, resource_name)

## S3 method for class 'datapackage'
dp_resource(x, resource_name)

dp_resource(x, resource_name) <- value

## S3 replacement method for class 'readonlydatapackage'
dp_resource(x, resource_name) <- value

## S3 replacement method for class 'editabledatapackage'
dp_resource(x, resource_name) <- value
```

Arguments

x	a datapackage object.
resource_name	the name of a resource.
value	a dataresource object.

Details

When a resource with the name already exists this resource is overwritten. Therefore, the assignment operator can also be used to modify existing resources.

Value

Either returns a Data Resource object or modifies the Data Package.

---

dp_resources<- *Modify a set of Data Resources in a Data Package*

Description

Modify a set of Data Resources in a Data Package

Usage

```
dp_resources(x) <- value
```

Arguments

x	a datapackage object
value	a dataresource object or a list of dataresource objects .

Value

Returns a modified x.

---

dp_resource_names *Get the names of the resources in a Data Package*

Description

Get the names of the resources in a Data Package

Usage

```
dp_resource_names(dp)
```

Arguments

dp	A datapackage object.

Value

Returns a character vector with the names of the data resources in the Data Package.

dp_save_as_datapackage *Save a dataset as a Data Package*

Description

Save a dataset as a Data Package

Usage

```
dp_save_as_datapackage(
  data,
  path,
  name,
  categories_type = c("regular", "resource")
)
```

Arguments

data	the data.frame with the data to save
path	directory in which to create the datapackage
name	name of the Data Resource. When omitted a name is generated.
categories_type	how should categories be stored. See dp_generate_fielddescriptor.

Details

This function is a wrapper function around new_datapackage, dp_generate_dataresource and dp_write_data. These functions are called with the default arguments. This allows for a quick way to save a data set with any necessary data needed to read the dataset.

Value

Does not return anything. Called for the side effect of creating a directory and creating a number of files in this directory. Together these form a complete Data Package.

dp_to_boolean *Convert a vector to 'boolean' using the specified field descriptor*

Description

Convert a vector to 'boolean' using the specified field descriptor

Usage

```
dp_to_boolean(x, fielddescriptor = list(), ...)
```

Arguments

x	the vector to convert.
fielddescriptor	the field descriptor for the field.
...	passed on to other methods.

Details

When fielddescriptor is missing a default field descriptor is generated.

Value

Will return an logical vector with fielddescriptor added as the 'fielddescriptor' attribute.

---

dp_to_code *Recode a variable to* code *using the associated categories*

Description

Recode a variable to code using the associated categories

Usage

```
dp_to_code(x, categorieslist = dp_categorieslist(x), ..., warn = FALSE)
```

Arguments

x	the variable to recode
categorieslist	a data.frame with the categories as a data.frame.
...	passed on to as.codelist.
warn	give a warning when there is no code list.

Details

Uses the code method from the 'codelist' package. This package therefore needs to be installed. See the documentation of that package for how to work with 'code' objects.

Value

Returns a 'code' object or x when no categories could be found (categorieslist = NULL).

See Also

An alternative is the dp_to_factor function to convert to regular R factor.

Examples

```
fn <- system.file("examples/iris", package = "datapackage")
dp <- open_datapackage(fn)
dta <- dp |> dp_get_data("complex", convert_categories = "no")
dp_to_code(dta$factor1)

dp |> dp_get_data("complex", convert_categories = "dp_to_code")
```

`dp_to_date` *Convert a vector to 'date' using the specified field descriptor*

Description

Convert a vector to 'date' using the specified field descriptor

Usage

```
dp_to_date(x, fielddescriptor = list(), ...)
```

Arguments

`x`	the vector to convert.
`fielddescriptor`	the field descriptor for the field.
`...`	passed on to other methods.

Details

When `fielddescriptor` is missing a default field descriptor is generated.

Value

Will return an `Date` vector with `fielddescriptor` added as the 'fielddescriptor' attribute.

dp_to_datetime *Convert a vector to 'datetime' using the specified field descriptor*

Description

Convert a vector to 'datetime' using the specified field descriptor

Usage

```
dp_to_datetime(x, fielddescriptor = list(), ...)
```

Arguments

x	the vector to convert.
fielddescriptor	the field descriptor for the field.
...	passed on to other methods.

Details

When `fielddescriptor` is missing a default field descriptor is generated.

For the default format 'iso8601::iso8601todatetime' is used to convert. This function allows more formats than the Data Package standard prescribes. When format equals "any" the default 'as.POSIXct' function is used.

When `x` is numeric or integer, it is assumed that these are seconds since the unix time epoch (1970-01-01T00:00:00).

Value

Will return an `POSIXct` vector with `fielddescriptor` added as the 'fielddescriptor' attribute.

dp_to_factor *Recode a variable to factor using the associated categories*

Description

Recode a variable to factor using the associated categories

Usage

```
dp_to_factor(x, categorieslist = dp_categorieslist(x), warn = TRUE)
```

Arguments

x	the variable to recode
categorieslist	a data.frame with the categories as a data.frame.
warn	give a warning when there is no code list.

Value

Returns a factor vector or x when no categories could be found (categorieslist = NULL).

See Also

An alternative is the dp_to_code function to convert to 'code' object from the 'codelist' package.

Examples

```
fn <- system.file("examples/iris", package = "datapackage")
dp <- open_datapackage(fn)
dta <- dp |> dp_get_data("complex", convert_categories = "no")
dp_to_factor(dta$factor1)

dp |> dp_get_data("complex", convert_categories = "to_factor")
```

---

dp_to_integer *Convert a vector to 'integer' using the specified field descriptor*

Description

Convert a vector to 'integer' using the specified field descriptor

Usage

```
dp_to_integer(x, fielddescriptor = list(), ...)
```

Arguments

x	the vector to convert.
fielddescriptor	the field descriptor for the field.
...	passed on to other methods.

Details

When fielddescriptor is missing a default field descriptor is generated.

Value

Will return an integer vector with fielddescriptor added as the 'fielddescriptor' attribute.

dp_to_number — *Convert a vector to 'number' using the specified field descriptor*

Description

Convert a vector to 'number' using the specified field descriptor

Usage

```
dp_to_number(x, fielddescriptor = list(), decimalChar = ".", ...)
```

Arguments

x	the vector to convert.
fielddescriptor	the field descriptor for the field.
decimalChar	decimal separator. Used when the field field descriptor does not specify a decimal separator.
...	passed on to other methods.

Details

When `fielddescriptor` is missing a default field descriptor is generated.

Value

Will return an `numeric` vector with `fielddescriptor` added as the 'fielddescriptor' attribute.

dp_to_string — *Convert a vector to 'string' using the specified fielddescriptor*

Description

Convert a vector to 'string' using the specified fielddescriptor

Usage

```
dp_to_string(x, fielddescriptor = list(), ...)
```

Arguments

x	the vector to convert.
fielddescriptor	the field descriptor for the field.
...	passed on to other methods.

Details

When `fielddescriptor` is missing a default field descriptor is generated.

Value

Will return an `character` vector with `fielddescriptor` added as the 'fielddescriptor' attribute.

`dp_to_time` *Convert a vector to 'time' using the specified field descriptor*

Description

Convert a vector to 'time' using the specified field descriptor

Usage

```
dp_to_time(x, fielddescriptor = list(), ...)
```

Arguments

`x`	the vector to convert.
`fielddescriptor`	the field descriptor for the field.
`...`	passed on to other methods.

Details

When `fielddescriptor` is missing a default field descriptor is generated.

For the default format 'iso8601::iso8601totime' is used to convert. This function allows more formats than the Data Package standard prescribes. When format equals "any" the default 'as.POSIXct' function is used.

When `x` is numeric or integer, it is assumed that these are seconds since the unix time epoch (1970-01-01T00:00:00Z).

Value

Will return an `Time` vector (see `iso8601totime` with `fielddescriptor` added as the 'fielddescriptor' attribute.

dp_to_year *Convert a vector to 'year' using the specified field descriptor*

Description

Convert a vector to 'year' using the specified field descriptor

Usage

```
dp_to_year(x, fielddescriptor = list(), ...)
```

Arguments

x	the vector to convert.
fielddescriptor	the field descriptor for the field.
...	passed on to other methods.

Details

When `fielddescriptor` is missing a default field descriptor is generated.

Value

Will return an integer vector with `fielddescriptor` added as the 'fielddescriptor' attribute.

dp_to_yearmonth *Convert a vector to 'yearmonth' using the specified field descriptor*

Description

Convert a vector to 'yearmonth' using the specified field descriptor

Usage

```
dp_to_yearmonth(x, fielddescriptor = list(), ...)
```

Arguments

x	the vector to convert.
fielddescriptor	the field descriptor for the field.
...	passed on to other methods.

Details

When `fielddescriptor` is missing a default field descriptor is generated.

Valid formats are "YYYY-MM" or "YYYYMM". When x is numeric or integer, it is assumed that it was a yearmonth in the format "YYYYMM" that was accidentally converted to numeric format.

Value

Will return an `Date` vector with `fielddescriptor` added as the 'fielddescriptor' attribute. The dates will be the first of the given month. Therefore, a 'yearmonth' "2024-05" is translated to a date "2024-05-01".

`dp_write_data` *Write data of resource to file*

Description

Write data of resource to file

Usage

```
dp_write_data(x, ..., write_categories = TRUE)

## S3 method for class 'datapackage'
dp_write_data(
  x,
  resource_name,
  data,
  writer = "guess",
  ...,
  write_categories = TRUE
)

## S3 method for class 'dataresource'
dp_write_data(
  x,
  data,
  datapackage = dp_get_datapackage(x),
  writer = "guess",
  ...,
  write_categories = TRUE
)
```

Arguments

`x`	the Data Package or Data Resource to which the data is to be written to.
`...`	additional arguments are passed on to the writer function.

write_categories
: write both the data set x itself and any categories lists of fields in the data set.

resource_name
: name of the Data Resource in the Data Package to which the data needs to be written.

data
: data.frame with the data to write.

writer
: the writer to use to write the data. This should be either a function accepting the Data Package, name of the Data Resource, the data and the write_categories argument or the character string "guess".

datapackage
: the Data Package to which the data needs to be written.

Details

When writer = "guess" the function will try to guess which writer to use based on the format and mediatype of the Data Resource.

Value

The function doesn't return anything. It is called for it's side effect of creating files in the directory of the Data Package.

---

fwf_reader *Read the FWF-data for a Data Resource*

Description

Read the FWF-data for a Data Resource

Usage

```
fwf_reader(path, resource, convert_categories = c("no", "to_factor"), ...)
```

Arguments

path
: path to the data set.

resource
: a Data Resource.

convert_categories
: how to handle columns for which the field descriptor has a categories property. Passed on to dp_apply_schema.

...
: additional arguments are passed on to dp_apply_schema.

Value

Returns a data.frame with the data.

See Also

Generally used by calling dp_get_data.

new_contributor *Creating and Adding Contributors to a Data Package*

Description

Creating and Adding Contributors to a Data Package

Usage

```
new_contributor(
  title,
  role = c("contributor", "author", "publisher", "maintainer", "wrangler"),
  path = NULL,
  email = NULL,
  organisation = NULL
)

dp_add_contributor(x, contributor)

dp_add_contributor(x) <- value
```

Arguments

title	A length 1 character vector with the full nam of the contributor.
role	The role of the contributor
path	A URL to e.g. a home page of the contributor
email	The email address of the contributor
organisation	The orgination the contributor belongs to.
x	The Data Package to which the contributor has to be added.
contributor	a contributor object
value	a contributor object

Value

new_contributor returns a list with the given properties. This function is meant to assist in creating valid contributors.

Examples

```
dp <- open_datapackage(system.file(package = "datapackage", "examples/iris"))
dp_contributors(dp)
dp_contributors(dp) <- list(
  new_contributor("John Doe", email = "j.doe@somewhere.org"),
  list(title = "Jane Doe", role = "maintainer")
)
dp_add_contributor(dp) <- new_contributor("Janet Doe")
```

`new_datapackage` *Create a new Data Package*

Description

Create a new Data Package

Usage

```
new_datapackage(path, name = NULL, title = NULL, description = NULL, ...)
```

Arguments

`path`	The directory which will contain the Data Package or the filename in which to write the Data Package.
`name`	The name of the Data Package.
`title`	The title of the Data Package.
`description`	The description of the Data Package.
`...`	Ignored for now.

Value

The directory of `path`, or the directory containing `path` if path is a file name, is created and the file with the Data Package information is created. When `path` is a directory a file `datapackage.json` is created. The function returns an editable `datapackage` object.

Examples

```
dir <- tempdir()
dp <- new_datapackage(dir, name = "test-package")

dp_title(dp) <- "A Test Data Package"
dp_add_contributor(dp) <- new_contributor(title = "John Doe")
```

new_dataresource *Create a new Data Resource*

Description

Create a new Data Resource

Usage

```
new_dataresource(
  name,
  title = NULL,
  description = NULL,
  path = NULL,
  format = NULL,
  mediatype = NULL,
  encoding = NULL,
  bytes = NULL,
  hash = NULL,
  ...
)
```

Arguments

name	The name of the Data Resource.
title	The title of the Data Resource.
description	The description of the Data Resource.
path	the path of the Data Resource
format	the format of the Data Resource
mediatype	the mediatype of the Data Resource
encoding	the encoding of the Data Resource
bytes	the number of bytes of the Data Resource
hash	the hash of the Data Resource
...	additional arguments are added as additional properties. It is checked if these are valid.

Value

Returns a dataresource object which is a list with the properties of the Data Resource.

Examples

```
dir <- tempdir()
dp <- new_datapackage(dir, name = "test-package")

res <- new_dataresource(name = "iris")
dp_title(res) <- "The Iris Data Set"
dp_encoding(res) <- "UTF-8"
dp_mediatype(res) <- "text/csv"

# resource adds a resource if it doesn't yet exist or updates
# an existing resource
dp_resource(dp, "iris") <- res
```

`open_datapackage` *Open a data package*

Description

Open a data package

Usage

```
open_datapackage(path, readonly = TRUE)
```

Arguments

`path`	The filename or the data package description or the directory in which the data package is located.
`readonly`	Open the data package as a read-only data package or not. See 'details'

Details

When `path` is a directory name, the function looks for the files 'datapackage.json' or 'datapackage.yaml' in the directory. Otherwise, the function assumes the file contains the description of the data package.

When the data package is read with `readonly = FALSE`, any operations reading properties from the data package read those properties directly from the file on disk. And setting the properties will change the file on disk. This ensures the file is always consistent.

Value

Returns a list with the contents of the data package when `readonly = TRUE`. Otherwise an empty list is returned. In both cases the filename of the data package description (typically 'datapackage.json') and the director in which the data package is located are stored in attributes of the result.

PropertiesDatapackage *Getting and setting properties of Data Packages*

Description

Getting and setting properties of Data Packages

Usage

```
dp_contributors(x, ...)

dp_contributors(x) <- value

## S3 method for class 'datapackage'
dp_contributors(x, ...)

## S3 replacement method for class 'datapackage'
dp_contributors(x) <- value

dp_name(x)

## S3 method for class 'datapackage'
dp_name(x)

dp_name(x) <- value

## S3 replacement method for class 'datapackage'
dp_name(x) <- value

dp_title(x)

## S3 method for class 'datapackage'
dp_title(x)

dp_title(x) <- value

## S3 replacement method for class 'datapackage'
dp_title(x) <- value

dp_description(x, ..., first_paragraph = FALSE, dots = FALSE)

## S3 method for class 'datapackage'
dp_description(x, ..., first_paragraph = FALSE, dots = FALSE)

dp_description(x) <- value

## S3 replacement method for class 'datapackage'
```

```
dp_description(x) <- value

dp_keywords(x, ...)

## S3 method for class 'datapackage'
dp_keywords(x, ...)

dp_keywords(x) <- value

## S3 replacement method for class 'datapackage'
dp_keywords(x) <- value

dp_created(x, ...)

## S3 method for class 'datapackage'
dp_created(x, ...)

dp_created(x) <- value

## S3 replacement method for class 'datapackage'
dp_created(x) <- value

dp_id(x, ...)

## S3 method for class 'datapackage'
dp_id(x, ...)

dp_id(x) <- value

## S3 replacement method for class 'datapackage'
dp_id(x) <- value
```

Arguments

x	a datapackage object.
...	used to pass additional arguments to other methods.
value	the new value of the property.
first_paragraph	Only return the first paragraph of the description.
dots	When returning only the first paragraph indicate missing paragraphs with ....

Value

Either returns the property or modifies the object.

See Also

See dp_resource for methods for getting and setting the resources of a Data Package.

See PropertiesDataresource and PropertiesFielddescriptor for methods for Data Resources and Field Descriptors respectively. Also see dp_property for a generic method for getting and setting properties. These functions can also be used to get and set 'unofficial' properties'

PropertiesDataresource

Getting and setting properties of Data Resources

Description

Getting and setting properties of Data Resources

Usage

```
## S3 method for class 'dataresource'
dp_name(x)

## S3 replacement method for class 'dataresource'
dp_name(x) <- value

## S3 method for class 'dataresource'
dp_title(x)

## S3 replacement method for class 'dataresource'
dp_title(x) <- value

## S3 method for class 'dataresource'
dp_description(x, ..., first_paragraph = FALSE, dots = FALSE)

## S3 replacement method for class 'dataresource'
dp_description(x) <- value

dp_path(x, ...)

dp_path(x) <- value

## S3 method for class 'dataresource'
dp_path(x, full_path = FALSE, ...)

## S3 replacement method for class 'dataresource'
dp_path(x) <- value

dp_format(x, ...)

dp_format(x) <- value

## S3 method for class 'dataresource'
```

```
dp_format(x, default = FALSE, ...)

## S3 replacement method for class 'dataresource'
dp_format(x) <- value

dp_mediatype(x, ...)

dp_mediatype(x) <- value

## S3 method for class 'dataresource'
dp_mediatype(x, ...)

## S3 replacement method for class 'dataresource'
dp_mediatype(x) <- value

dp_encoding(x, default = FALSE, ...)

dp_encoding(x) <- value

## S3 method for class 'dataresource'
dp_encoding(x, default = FALSE, ...)

## S3 replacement method for class 'dataresource'
dp_encoding(x) <- value

dp_bytes(x, ...)

dp_bytes(x) <- value

## S3 method for class 'dataresource'
dp_bytes(x, ...)

## S3 replacement method for class 'dataresource'
dp_bytes(x) <- value

dp_hash(x, ...)

dp_hash(x) <- value

## S3 method for class 'dataresource'
dp_hash(x, ...)

## S3 replacement method for class 'dataresource'
dp_hash(x) <- value

## S3 replacement method for class 'fielddescriptor'
dp_name(x) <- value
```

```
## S3 replacement method for class 'fielddescriptor'
dp_title(x) <- value

## S3 method for class 'fielddescriptor'
dp_description(x, ..., first_paragraph = FALSE, dots = FALSE)

## S3 replacement method for class 'fielddescriptor'
dp_format(x) <- value

dp_schema(x)

## S3 method for class 'dataresource'
dp_schema(x)
```

Arguments

x	a dataresource object.
value	the new value of the property.
...	used to pass additional arguments to other methods.
first_paragraph	Only return the first paragraph of the description.
dots	When returning only the first paragraph indicate missing paragraphs with ....
full_path	Return the full path including the path to the Data Package and not only the path relative to the Data Package. This is only relevant for relative paths.
default	return the default value if the property had a default value and the property is not set.

Value

Either returns the property or modifies the object. If the property of not set NULL is returned (unless default = TRUE).

See Also

See PropertiesDatapackage and PropertiesFielddescriptor for methods for Data Packages and Field Descriptors respectively. Also see dp_property for a generic method for getting and setting properties. These functions can also be used to get and set 'unofficial' properties'

PropertiesFielddescriptor

Getting and setting properties of Data Resources

Description

Getting and setting properties of Data Resources

Usage

```
## S3 method for class 'fielddescriptor'
dp_name(x)

## S3 method for class 'fielddescriptor'
dp_title(x)

## S3 replacement method for class 'fielddescriptor'
dp_description(x) <- value

## S3 method for class 'fielddescriptor'
dp_format(x, ...)
```

Arguments

x	a fielddescriptor object.
value	the new value of the property.
...	used to pass additional arguments to other methods.

Value

Either returns the property or modifies the object. If the property is not set NULL is returned (unless default = TRUE).

See Also

See PropertiesDatapackage and PropertiesDataresource for methods for Data Packages and Data Resources respectively. Also see dp_property for a generic method for getting and setting properties. These functions can also be used to get and set 'unofficial' properties'

Index