Package 'frictionless'

July 22, 2025

Title Read and Write Frictionless Data Packages

Version 1.2.1

Date 2025-05-23

Description Read and write Frictionless Data Packages. A 'Data Package' (<https://specs.frictionlessdata.io/data-package/>) is a simple container format and standard to describe and package a collection of (tabular) data. It is typically used to publish FAIR (<https://www.go-fair.org/fair-principles/>) and open datasets.

License MIT + file LICENSE

URL https://github.com/frictionlessdata/frictionless-r, https://docs.ropensci.org/frictionless/

BugReports https://github.com/frictionlessdata/frictionless-r/issues

Depends R (>= 3.6.0)

Imports cli, dplyr, httr, jsonlite, purrr, readr (>= 2.1.0), rlang, utils, yaml

Suggests hms, knitr, lubridate, rmarkdown, stringi, testthat (>= 3.0.0), tibble

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Peter Desmet [aut, cre] (ORCID: <https://orcid.org/0000-0002-8442-8025>, affiliation: Research Institute for Nature and Forest (INBO)), Damiano Oldoni [aut] (ORCID: <https://orcid.org/0000-0003-3445-7562>, affiliation: Research Institute for Nature and Forest (INBO)), Pieter Huybrechts [aut] (ORCID: <https://orcid.org/0000-0002-6658-6062>, affiliation: Research Institute for Nature and Forest (INBO)),

Sanne Govaert [aut] (ORCID: <https://orcid.org/0000-0002-8939-1305>,
affiliation: Research Institute for Nature and Forest (INBO)),
Kyle Husmann [ctb] (ORCID: <https://orcid.org/0000-0001-9875-8976>,
affiliation: Pennsylvania State University),
Research Institute for Nature and Forest (INBO) [cph] (ROR:
<https://ror.org/00j54wy13>),
Research Foundation - Flanders [fnd] (https://lifewatch.be),
Beatriz Milz [rev] (ORCID: <https://orcid.org/0000-0002-3064-4486>),
João Martins [rev] (ORCID: <https://orcid.org/0000-0001-7961-4280>)

Maintainer Peter Desmet <peter.desmet@inbo.be>

Repository CRAN

Date/Publication 2025-05-23 15:32:01 UTC

Contents

`add_resource` *Add a Data Resource*

Description

Adds a Data Resource to a Data Package. The resource will be a Tabular Data Resource. The resource name can only contain lowercase alphanumeric characters plus `.`, `-` and `_`.

Usage

```
add_resource(
  package,
  resource_name,
  data,
  schema = NULL,
  replace = FALSE,
```

```
  delim = ",",
  ...
)
```

Arguments

package	Data Package object, as returned by read_package() or create_package().
resource_name	Name of the Data Resource.
data	Data to attach, either a data frame or path(s) to CSV file(s): • Data frame: attached to the resource as data and written to a CSV file when using write_package(). • One or more paths to CSV file(s) as a character (vector): added to the resource as path. The last file will be read with readr::read_delim() to create or compare with schema and to set format, mediatype and encoding. The other files are ignored, but are expected to have the same structure and properties.
schema	Either a list, or path or URL to a JSON file describing a Table Schema for the data. If not provided, one will be created using create_schema().
replace	If TRUE, the added resource will replace an existing resource with the same name.
delim	Single character used to separate the fields in the CSV file(s), e.g. \t for tab delimited file. Will be set as delimiter in the resource Table Dialect, so read functions . know how to read the file(s).
...	Additional metadata properties to add to the resource, e.g. title = "My title", validated = FALSE. These are not verified against specifications and are ignored by read_resource(). The following properties are automatically set and can't be provided with ...: name, data, path, schema, profile, format, mediatype, encoding and dialect.

Details

See vignette("data-resource") (and to a lesser extend vignette("table-dialect")) to learn how this function implements the Data Package standard.

Value

package with one additional resource.

See Also

Other edit functions: remove_resource()

Examples

```
# Load the example Data Package
package <- example_package()

# List the resources
resources(package)
```

```
# Create a data frame
df <- data.frame(
  multimedia_id = c(
    "aed5fa71-3ed4-4284-a6ba-3550d1a4de8d",
    "da81a501-8236-4cbd-aa95-4bc4b10a05df"
  ),
  x = c(718, 748),
  y = c(860, 900)
)

# Add the resource "positions" from the data frame
package <- add_resource(package, "positions", data = df)

# Add the resource "positions_with_schema", with a user-defined schema and title
my_schema <- create_schema(df)
package <- add_resource(
  package,
  resource_name = "positions_with_schema",
  data = df,
  schema = my_schema,
  title = "Positions with schema"
)

# Replace the resource "observations" with a file-based resource (2 TSV files)
path_1 <-
system.file("extdata", "v1", "observations_1.tsv", package = "frictionless")
path_2 <-
system.file("extdata", "v1", "observations_2.tsv", package = "frictionless")
package <- add_resource(
  package,
  resource_name = "observations",
  data = c(path_1, path_2),
  replace = TRUE,
  delim = "\t"
)

# List the resources ("positions" and "positions_with_schema" added)
resources(package)
```

---

check_package *Check a Data Package object*

Description

Check if an object is a Data Package object with the required properties.

Usage

```
check_package(package)
```

Arguments

package	Data Package object, as returned by read_package() or create_package().

Value

package invisibly or an error.

Examples

```
# Load the example Data Package
package <- example_package()

# Check if the Data Package is valid (invisible return)
check_package(package)
```

---

create_package *Create a Data Package*

Description

Initiates a Data Package object, either from scratch or from an existing list. This Data Package object is a list with the following characteristics:

- A datapackage subclass.
- All properties of the original descriptor.
- A resources property, set to an empty list if undefined.
- A directory property, set to "." for the current directory if undefined. It is used as the base path to access resources with read_resource().

Usage

```
create_package(descriptor = NULL)
```

Arguments

descriptor	List to be made into a Data Package object. If undefined, an empty Data Package will be created from scratch.

Details

See vignette("data-package") to learn how this function implements the Data Package standard. check_package() is automatically called on the created package to make sure it is valid.

Value

A Data Package object.

See Also

Other create functions: `create_schema()`

Examples

```
# Create a Data Package
package <- create_package()

package

# See the structure of the (empty) Data Package
str(package)
```

`create_schema` *Create a Table Schema for a data frame*

Description

Creates a Table Schema for a data frame, listing all column names and types as field names and (converted) types.

Usage

```
create_schema(data)
```

Arguments

`data`	A data frame.

Details

See `vignette("table-schema")` to learn how this function implements the Data Package standard.

Value

List describing a Table Schema.

See Also

Other create functions: `create_package()`

Examples

```
# Create a data frame
df <- data.frame(
  id = c(as.integer(1), as.integer(2)),
  timestamp = c(
    as.POSIXct("2020-03-01 12:00:00", tz = "EET"),
    as.POSIXct("2020-03-01 18:45:00", tz = "EET")
  ),
  life_stage = factor(c("adult", "adult"), levels = c("adult", "juvenile"))
)

# Create a Table Schema from the data frame
schema <- create_schema(df)
str(schema)
```

`example_package` *Read the example Data Package*

Description

Reads the example Data Package included in `frictionless`. This dataset is used in examples, vignettes, and tests and contains dummy camera trap data organized in 3 Data Resources:

1. `deployments`: one local data file referenced in `"path": "deployments.csv"`.
2. `observations`: two local data files referenced in `"path": ["observations_1.tsv", "observations_2.tsv"]`.
3. `media`: inline data stored in `data`.

Usage

```
example_package(version = "1.0")
```

Arguments

`version`	Data Package standard version.

Details

The example Data Package is available in two versions:

- `1.0`: specified as a Data Package v1.
- `2.0`: specified as a Data Package v2.

Value

A Data Package object, see `create_package()`.

Examples

```
# Version 1
example_package()

# Version 2
example_package(version = "2.0")
```

`get_schema` *Get the Table Schema of a Data Resource*

Description

Returns the Table Schema of a Data Resource (in a Data Package), i.e. the content of its `schema` property, describing the resource's fields, data types, relationships, and missing values. The resource must be a Tabular Data Resource.

Usage

```
get_schema(package, resource_name)
```

Arguments

`package`	Data Package object, as returned by `read_package()` or `create_package()`.
`resource_name`	Name of the Data Resource.

Details

See `vignette("table-schema")` to learn more about Table Schema.

Value

List describing a Table Schema.

Examples

```
# Load the example Data Package
package <- example_package()

# Get the Table Schema for the resource "observations"
schema <- get_schema(package, "observations")
str(schema)
```

print.datapackage *Print a Data Package*

Description

Prints a human-readable summary of a Data Package, including its resources and a link to more information (if provided in `package$id`).

Usage

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

Arguments

`x`	Data Package object, as returned by `read_package()` or `create_package()`.
`...`	Further arguments, they are ignored by this function.

Value

`print()` with a summary of the Data Package object.

Examples

```
# Load the example Data Package
package <- example_package()

# Print a summary of the Data Package
package # Or print(package)
```

read_package *Read a Data Package descriptor file (`datapackage.json`)*

Description

Reads information from a `datapackage.json` file, i.e. the descriptor file that describes the Data Package metadata and its Data Resources.

Usage

```
read_package(file = "datapackage.json")
```

Arguments

`file`	Path or URL to a `datapackage.json` file.

Details

See `vignette("data-package")` to learn how this function implements the Data Package standard.

Value

A Data Package object, see `create_package()`.

See Also

Other read functions: `read_resource()`, `resources()`

Examples

```
# Read a datapackage.json file
package <- read_package(
  system.file("extdata", "v1", "datapackage.json", package = "frictionless")
)

package

# Access the Data Package properties
package$name
package$created
```

---

read_resource *Read data from a Data Resource into a tibble data frame*

Description

Reads data from a Data Resource (in a Data Package) into a tibble (a Tidyverse data frame). The resource must be a Tabular Data Resource. The function uses `readr::read_delim()` to read CSV files, passing the resource properties `path`, CSV dialect, column names, data types, etc. Column names are taken from the provided Table Schema (`schema`), not from the header in the CSV file(s).

Usage

```
read_resource(package, resource_name, col_select = NULL)
```

Arguments

`package`	Data Package object, as returned by `read_package()` or `create_package()`.
`resource_name`	Name of the Data Resource.
`col_select`	Character vector of the columns to include in the result, in the order provided. Selecting columns can improve read speed.

Details

See `vignette("data-resource")`, `vignette("table-dialect")` and `vignette("table-schema")` to learn how this function implements the Data Package standard.

Value

A `tibble::tibble()` with the Data Resource's tabular data. If there are parsing problems, a warning will alert you. You can retrieve the full details by calling `problems()` on your data frame.

See Also

Other read functions: `read_package()`, `resources()`

Examples

```
# Read a datapackage.json file
package <- read_package(
  system.file("extdata", "v1", "datapackage.json", package = "frictionless")
)

package

# Read data from the resource "observations"
read_resource(package, "observations")

# The above tibble is merged from 2 files listed in the resource path
package$resources[[2]]$path

# The column names and types are derived from the resource schema
purrr::map_chr(package$resources[[2]]$schema$fields, "name")
purrr::map_chr(package$resources[[2]]$schema$fields, "type")

# Read data from the resource "deployments" with column selection
read_resource(package, "deployments", col_select = c("latitude", "longitude"))
```

---

remove_resource *Remove a Data Resource*

Description

Removes a Data Resource from a Data Package, i.e. it removes one of the described `resources`.

Usage

```
remove_resource(package, resource_name)
```

Arguments

`package`	Data Package object, as returned by `read_package()` or `create_package()`.
`resource_name`	Name of the Data Resource.

Value

`package` with one fewer resource.

See Also

Other edit functions: add_resource()

Examples

```
# Load the example Data Package
package <- example_package()

# List the resources
resources(package)

# Remove the resource "observations"
package <- remove_resource(package, "observations")

# List the resources ("observations" removed)
resources(package)
```

---

resources *List Data Resources*

Description

Lists the names of the Data Resources included in a Data Package.

Usage

```
resources(package)
```

Arguments

package	Data Package object, as returned by read_package() or create_package().

Value

Character vector with the Data Resource names.

See Also

Other read functions: read_package(), read_resource()

Examples

```
# Load the example Data Package
package <- example_package()

# List the resources
resources(package)
```

write_package *Write a Data Package to disk*

Description

Writes a Data Package and its related Data Resources to disk as a `datapackage.json` and CSV files. Already existing CSV files of the same name will not be overwritten. The function can also be used to download a Data Package in its entirety. The Data Resources are handled as follows:

- Resource `path` has at least one local path (e.g. `deployments.csv`): CSV files are copied or downloaded to `directory` and `path` points to new location of file(s).
- Resource `path` has only URL(s): resource stays as is.
- Resource has inline `data` originally: resource stays as is.
- Resource has inline `data` as result of adding data with `add_resource()`: data are written to a CSV file using `readr::write_csv()`, `path` points to location of file, `data` property is removed. Use `compress = TRUE` to gzip those CSV files.

Usage

```
write_package(package, directory, compress = FALSE)
```

Arguments

`package`	Data Package object, as returned by `read_package()` or `create_package()`.
`directory`	Path to local directory to write files to.
`compress`	If `TRUE`, data of added resources will be gzip compressed before being written to disk (e.g. `deployments.csv.gz`).

Value

`package` invisibly, as written to file.

Examples

```
# Load the example Data Package from disk
package <- read_package(
  system.file("extdata", "v1", "datapackage.json", package = "frictionless")
)

package

# Write the (unchanged) Data Package to disk
write_package(package, directory = "my_directory")

# Check files
list.files("my_directory")
```

```
# No files written for the "observations" resource, since those are all URLs.
# No files written for the "media" resource, since it has inline data.

# Clean up (don't do this if you want to keep your files)
unlink("my_directory", recursive = TRUE)
```

Index