Package 'rDataPipeline'

May 9, 2026

Title Functions to Interact with the 'FAIR Data Pipeline'

Version 0.60.0

Description R implementation of the 'FAIR Data Pipeline API'. The 'FAIR Data Pipeline' is intended to enable tracking of provenance of FAIR (findable, accessible and interoperable) data used in epidemiological modelling.

License GPL (>= 3)

Imports assertthat, cli, configr, dplyr, git2r, httr, jsonlite, openssl, R6, rhdf5, semver, stats, usethis, utils, yaml

Suggests units, testthat

biocViews rhdf5

Encoding UTF-8

RoxygenNote 7.3.2

URL https://www.fairdatapipeline.org/rDataPipeline/, https://github.com/FAIRDataPipeline/rDataPipeline

BugReports https://github.com/FAIRDataPipeline/rDataPipeline/issues

NeedsCompilation no

Author Sonia Mitchell [aut] (ORCID: <https://orcid.org/0000-0003-1536-2066>), Ryan Field [cre, aut] (ORCID: <https://orcid.org/0000-0002-4424-9890>)

Maintainer Ryan Field <ryan.field@glasgow.ac.uk>

Repository CRAN

Date/Publication 2024-10-08 09:20:02 UTC

Contents

`rDataPipeline-package` *rDataPipeline*

Description

FAIR Data Pipeline API

Details

For more information see https://www.fairdatapipeline.org/

Author(s)

Maintainer: Ryan Field <ryan.field@glasgow.ac.uk> (ORCID)

Authors:

- Sonia Mitchell (ORCID)

See Also

Useful links:

- https://www.fairdatapipeline.org/rDataPipeline/
- https://github.com/FAIRDataPipeline/rDataPipeline
- Report bugs at https://github.com/FAIRDataPipeline/rDataPipeline/issues

---

`add_read` *add_read*

Description

Add data product to `read` block of user-written config file. Used in combination with `create_config()` for unit testing.

Usage

```
add_read(
  path,
  data_product,
  component,
  version,
  use_data_product,
  use_component,
  use_version,
  use_namespace
)
```

Arguments

`path`	config file path
`data_product`	data_product field
`component`	component field
`version`	(optional) version field
`use_data_product`	(optional) use_data_product field
`use_component`	(optional) use_component field
`use_version`	(optional) use_version field
`use_namespace`	(optional) use_namespace field

Examples

```
## Not run:
path <- "test_config/config.yaml"

# Write run_metadata block
create_config(path = path,
              description = "test",
              input_namespace = "test_user",
              output_namespace = "test_user")

# Write read block
add_read(path = path,
         data_product = "test/array",
         component = "level/a/s/d/f/s",
         version = "0.2.0")

## End(Not run)
```

`add_write` *add_write*

Description

Add data product to `read` block of user-written config file. Used in combination with `create_config()` for unit testing.

Usage

```
add_write(
  path,
  data_product,
  description,
  version,
  file_type,
  use_data_product,
  use_component,
  use_version,
  use_namespace
)
```

Arguments

`path`	config file path
`data_product`	data_product field
`description`	component field
`version`	(optional) version field

file_type	(optional) file type field
use_data_product	(optional) use_data_product field
use_component	(optional) use_component field
use_version	(optional) use_version field
use_namespace	(optional) use_namespace field

Examples

```
## Not run:
path <- "test_config/config.yaml"

# Write run_metadata block
create_config(path = path,
              description = "test",
              input_namespace = "test_user",
              output_namespace = "test_user")

# Write read block
add_write(path = path,
          data_product = "test/array",
          description = "data product description",
          version = "0.2.0")

## End(Not run)
```

---

create_config *create_config*

Description

Generates (user generated) config.yaml files for unit tests. Use `add_read()` and `add_write()` functions to add read and write blocks.

Usage

```
create_config(
  path,
  description,
  input_namespace,
  output_namespace,
  write_data_store = file.path(tempdir(), "datastore", ""),
  force = TRUE,
  local_repo = "local_repo"
)
```

Arguments

path	config file path
description	description field
input_namespace	input_namespace field
output_namespace	output_namespace field
write_data_store	write_data_store field
force	force
local_repo	local_repo

---

fair_init *fair_init*

Description

fair_init

Usage

```
fair_init(name, identifier, endpoint = "http://127.0.0.1:8000/api/")
```

Arguments

name	a string specifying the full name or organisation name of the author; note that at least one of name or identifier must be specified
identifier	(optional) a string specifying the full URL identifier (*e.g.* ORCiD or ROR ID) of the author
endpoint	a string specifying the registry endpoint

---

fair_run *fair_run*

Description

fair_run

Usage

```
fair_run(
  path = "config.yaml",
  endpoint = "http://127.0.0.1:8000/api/",
  skip = FALSE
)
```

Arguments

path	string
endpoint	a string specifying the registry endpoint
skip	don't bother checking whether the repo is clean

---

fdp-class *fdp-class*

Description

fdp-class

fdp-class

Details

Container for class fdp

Public fields

yaml a list containing the contents of the working config.yaml

fdp_config_dir a string specifying the directory passed from fair run

model_config a string specifying the URL of an entry in the object table associated with the storage_location of the working config.yaml

submission_script a string specifying the URL of an entry in the object table associated with the storage_location of the submission script

code_repo a string specifying the URL of an entry in the object table associated with the GitHub repository

code_run a string specifying the URL of an entry in the code_run table

inputs a data.frame containing metadata associated with code_run inputs

outputs a data.frame containing metadata associated with code_run outputs

issues a data.frame containing metadata associated with code_run issues

Methods

Public methods:

- fdp$new()
- fdp$print()
- fdp$input()
- fdp$output()
- fdp$output_index()
- fdp$raise_issue()
- fdp$finalise_output_hash()

- fdp$finalise_output_url()
- fdp$clone()

Method `new()`**:** Create a new `fdp` object

Usage:

```
fdp$new(
  yaml,
  fdp_config_dir,
  model_config,
  submission_script,
  code_repo,
  code_run
)
```

Arguments:

`yaml` a `list` containing the contents of the working config.yaml

`fdp_config_dir` a `string` specifying the directory passed from `fair run`

`model_config` a `string` specifying the URL of an entry in the `object` table associated with the `storage_location` of the working config.yaml

`submission_script` a `string` specifying the URL of an entry in the `object` table associated with the `storage_location` of the submission script

`code_repo` a `string` specifying the URL of an entry in the `object` table associated with the GitHub repository

`code_run` a `string` specifying the URL of an entry in the `code_run` table

Returns: Returns a new `fdp` object

Method `print()`**:** Print method

Usage:

```
fdp$print(...)
```

Arguments:

`...` additional parameters, currently none are used

Method `input()`**:** Record `code_run` inputs in `fdp` object

Usage:

```
fdp$input(
  data_product,
  use_data_product,
  use_component,
  use_version,
  use_namespace,
  path,
  component_url
)
```

Arguments:

`data_product` a `string` specifying the name of the data product, used as a reference

use_data_product a string specifying the name of the data product, used as input in the code_run

use_component a string specifying the name of the data product component, used as input in the code_run

use_version a string specifying the data product version, used as input in the code_run

use_namespace a string specifying the namespace in which the data product resides, used as input in the code_run

path a string specifying the location of the data product in the local data store

component_url a string specifying the URL of an entry in the object_component table

Returns: Returns an updated fdp object

Method `output()`**:** Record code_run outputs in fdp object

Usage:

```
fdp$output(
  data_product,
  use_data_product,
  use_component,
  use_version,
  use_namespace,
  path,
  data_product_description,
  component_description,
  public
)
```

Arguments:

data_product a string specifying the name of the data product, used as a reference

use_data_product a string specifying the name of the data product, used as output in the code_run

use_component a string specifying the name of the data product component, used as output in the code_run

use_version a string specifying the version of the data product, used as output in the code_run

use_namespace a string specifying the namespace in which the data product resides, used as output in the code_run

path a string specifying the location of the data product in the local data store

data_product_description a string containing a description of the data product

component_description a string containing a description of the data product component

public

Returns: Returns an updated fdp object

Method `output_index()`**:** Return index of data product recorded in fdp object so that an issue may be attached

Usage:

```
fdp$output_index(data_product, component, version, namespace)
```

Arguments:

`data_product` a `string` specifying the name of the data product, used as output in the `code_run`

`component` a `string` specifying the name of the data product component, used as output in the `code_run`

`version` a `string` specifying the name of the data product version, used as output in the `code_run`

`namespace` a `string` specifying the namespace in which the data product resides, used as input in the `code_run`

Returns: Returns an index used to identify the data product

Method `raise_issue()`**:** Record `issue` in `fdp` object

Usage:

```
fdp$raise_issue(
  index,
  type,
  use_data_product,
  use_component,
  use_version,
  use_namespace,
  issue,
  severity
)
```

Arguments:

`index` a `numeric` index, used to identify each input and output in the `fdp` object

`type` a `string` specifying the type of issue (one of "data", "config", "script", "repo")

`use_data_product` a `string` specifying the name of the data product, used as output in the `code_run`

`use_component` a `string` specifying the name of the data product component, used as output in the `code_run`

`use_version` a `string` specifying the name of the data product version, used as output in the `code_run`

`use_namespace` a `string` specifying the namespace in which the data product resides, used as input in the `code_run`

`issue` a `string` containing a free text description of the `issue`

`severity` an `integer` specifying the severity of the `issue`

Returns: Returns an updated `fdp` object

Method `finalise_output_hash()`**:** Record file hash and update path name in `fdp` object

Usage:

```
fdp$finalise_output_hash(
  use_data_product,
  use_data_product_runid,
  use_version,
  use_namespace,
  hash,
  new_path,
```

```
  data_product_url,
  delete_if_duplicate = FALSE
)
```

Arguments:

`use_data_product` a `string` specifying the name of the data product, used as output in the `code_run`

`use_data_product_runid` a `string` specifying the name of the data product, the same as `use_data_product` excluding the RUN_ID variable

`use_version` a `string` specifying the name of the data product version, used as output in the `code_run`

`use_namespace` a `string` specifying the namespace in which the data product resides, used as input in the `code_run`

`hash` a `string` specifying the hash of the file

`new_path` a `string` specifying the updated location (filename is now the hash of the file) of the data product in the local data store

`data_product_url` a `string` specifying the URL of an `object` associated with the `data_product`

`delete_if_duplicate` (optional) default is `FALSE`

Returns: Returns an updated `fdp` object

Method `finalise_output_url()`**:** Record `data_product` and component URLs in `fdp` object

Usage:

```
fdp$finalise_output_url(
  use_data_product,
  use_component,
  use_version,
  use_namespace,
  component_url
)
```

Arguments:

`use_data_product` a `string` specifying the name of the data product, used as output in the `code_run`

`use_component` a `string` specifying the name of the data product component, used as output in the `code_run`

`use_version` a `string` specifying the name of the data product version, used as output in the `code_run`

`use_namespace` a `string` specifying the namespace in which the data product resides, used as input in the `code_run`

`component_url` a `string` specifying the URL of an entry in the `object_component` table

Returns: Returns an updated `fdp` object

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
fdp$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

finalise *Finalise code run*

Description

Finalise Code Run and push associated metadata to the local registry.

Usage

```
finalise(handle, delete_if_empty = FALSE, delete_if_duplicate = FALSE)
```

Arguments

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
delete_if_empty	(optional) default is FALSE; see Details
delete_if_duplicate	(optional) default is FALSE; see Details

Details

If a Code Run does not read an input, write an output, or attach an issue, then delete the Code Run entry when delete_if_empty is set to TRUE.

If a data product has the same hash as a previous version, remove it from the registry when delete_if_duplicate is set to TRUE.

findme *findme*

Description

Returns metadata associated with the calculated hash of a target file. When multiple entries exist in the data registry all are returned.

Usage

```
findme(file, endpoint)
```

Arguments

file	file path
endpoint	endpoint

find_read_match *Find matching read aliases in config file*

Description

Find read aliases in working config that match wildcard string

Usage

```
find_read_match(handle, data_product)
```

Arguments

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string specifying the data product name

find_write_match *Find matching write aliases in config file*

Description

Find write aliases in working config that match wildcard string

Usage

```
find_write_match(handle, data_product)
```

Arguments

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string specifying the data product name

get_components *Get H5 file components*

Description

Returns the names of the items at the root of the file

Usage

```
get_components(filename)
```

Arguments

filename	a string specifying a filename

Value

Returns the names of the items at the root of the file

See Also

Other get functions: get_entry(), get_existing(), get_file_hash(), get_github_hash()

get_dataproduct *get_dataproduct*

Description

get_dataproduct

Usage

```
get_dataproduct(
  data_product,
  version,
  namespace,
  endpoint = "http://127.0.0.1:8000/api/"
)
```

Arguments

data_product	data_product
version	version
namespace	namespace
endpoint	endpoint

get_entry *Return all fields associated with a table entry in the data registry*

Description

Return all fields associated with a table entry in the data registry

Usage

```
get_entry(table, query, endpoint = "http://127.0.0.1:8000/api/")
```

Arguments

table	a string specifying the name of the table
query	a list containing a valid query for the table, *e.g.* list(field = value)
endpoint	a string specifying the registry endpoint

Value

Returns a list of fields present in the specified entry

See Also

Other get functions: get_components(), get_existing(), get_file_hash(), get_github_hash()

initialise *Initialise code run*

Description

Reads in a working config file, generates new Code Run entry, and returns a handle containing various metadata.

Usage

```
initialise(config, script)
```

Arguments

config	a string specifying the location of the working config file in the data store
script	a string specifying the location of the submission script in the data store

Value

Returns an object of class fdp, R6 containing metadata required by the Data Pipeline API

link_read *Link path to external format data*

Description

Link path to external format data

Usage

```
link_read(handle, data_product)
```

Arguments

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string representing an external object in the config.yaml file

Value

Returns a string specifying the location of the data product to be read

link_write *Link path for external format data*

Description

Link path for external format data

Usage

```
link_write(handle, data_product)
```

Arguments

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string representing an external object in the config.yaml file

Value

Returns a string specifying the location in which the data product should be written

raise_issue *raise_issue*

Description

raise_issue

Usage

```
raise_issue(
  index,
  handle,
  component = NA,
  data_product,
  issue,
  severity,
  whole_object = FALSE
)
```

Arguments

index	index returned from `link_*()`, `read_()`, or `write()`
handle	an object of class `fdp`, `R6` containing metadata required by the Data Pipeline API
component	a `string` specifying the component name
data_product	a `string` specifying the data product name
issue	a `string` specifying the issue
severity	a `numeric` value specifying the severity
whole_object	a `boolean` flag specifying whether or not to reference the whole_object

raise_issue_config *Raise issue with config file*

Description

Raise issue with config file

Usage

```
raise_issue_config(handle, issue, severity)
```

Arguments

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
issue	a string specifying the issue
severity	a numeric value specifying the severity

---

raise_issue_repo *Raise issue with remote repository*

Description

Raise issue with remote repository

Usage

```
raise_issue_repo(handle, issue, severity)
```

Arguments

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
issue	a string specifying the issue
severity	a numeric value specifying the severity

---

raise_issue_script *Raise issue with submission script*

Description

Raise issue with submission script

Usage

```
raise_issue_script(handle, issue, severity)
```

Arguments

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
issue	a string specifying the issue
severity	a numeric value specifying the severity

random_hash *random_hash*

Description

Generates a random hash

Usage

```
random_hash()
```

read_array *Read array component from HDF5 file*

Description

Function to read array type data from hdf5 file.

Usage

```
read_array(handle, data_product, component)
```

Arguments

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string specifying a data product
component	a string specifying a data product component

Value

Returns an array with attached Dimension_i_title, Dimension_i_units, Dimension_i_values, and units attributes, if available

read_distribution *Read distribution component from TOML file*

Description

Function to read distribution type data from toml file.

Usage

```
read_distribution(handle, data_product, component)
```

Arguments

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string specifying a data product
component	a string specifying a data product component

read_estimate *Read estimate component from TOML file*

Description

Function to read point-estimate type data from toml file.

Usage

```
read_estimate(handle, data_product, component)
```

Arguments

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string specifying a data product
component	a string specifying a data product component

read_table *Read table component from HDF5 file*

Description

Function to read table type data from hdf5 file.

Usage

```
read_table(handle, data_product, component)
```

Arguments

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string specifying a data product
component	a string specifying a data product component

Value

Returns a data.frame with attached column_units attributes, if available

write_array *Write array component to HDF5 file*

Description

Function to populate hdf5 file with array type data.

Usage

```
write_array(
  array,
  handle,
  data_product,
  component,
  description,
  dimension_names,
  dimension_values,
  dimension_units,
  units
)
```

Arguments

array	an array containing the data
handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string specifying the name of the data product
component	a string specifying a location within the hdf5 file
description	a string describing the data product component
dimension_names	a list where each element is a vector containing the labels associated with a particular dimension (e.g. element 1 corresponds to dimension 1, which corresponds to row names) and the name of each element describes the contents of each dimension (e.g. age classes).
dimension_values	(optional) a list of values corresponding to each dimension (e.g. list element 2 corresponds to columns)
dimension_units	(optional) a list of units corresponding to each dimension (e.g. list element 2 corresponds to columns)
units	(optional) a string specifying the units of the data as a whole

Value

Returns a handle index associated with the just written component, which can be used to raise an issue if necessary

See Also

Other write functions: write_distribution(), write_estimate(), write_table()

---

write_distribution *Write distribution component to TOML file*

Description

Write distribution component to TOML file

Usage

```
write_distribution(
  distribution,
  parameters,
  handle,
  data_product,
  component,
  description
)
```

Arguments

distribution	a string specifying the name of the distribution
parameters	a list specifying the distribution parameters
handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string specifying the name of the data product
component	a string specifying a location within the toml file
description	a string describing the data product component

Value

Returns a handle index associated with the just written component, which can be used to raise an issue if necessary

See Also

Other write functions: write_array(), write_estimate(), write_table()

---

write_estimate *Write estimate component to TOML file*

Description

Function to populate toml file with point-estimate type data. If a file already exists at the specified location, an additional component will be added.

Usage

```
write_estimate(value, handle, data_product, component, description)
```

Arguments

value	an object of class numeric
handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string specifying the name of the data product
component	a string specifying a location within the toml file
description	a string describing the data product component

Value

Returns a handle index associated with the just written component, which can be used to raise an issue if necessary

See Also

Other write functions: write_array(), write_distribution(), write_table()

`write_table` *Write table component to HDF5 file*

Description

Function to populate hdf5 file with array type data.

Usage

```
write_table(
  df,
  handle,
  data_product,
  component,
  description,
  row_names,
  column_units
)
```

Arguments

`df`	an `dataframe` containing the data
`handle`	an object of class `fdp, R6` containing metadata required by the Data Pipeline API
`data_product`	a `string` specifying the name of the data product
`component`	a `string` specifying a location within the hdf5 file,
`description`	a `string` describing the data product component
`row_names`	(optional) a `vector` of rownames
`column_units`	(optional) a `vector` comprising column units

Value

Returns a handle index associated with the just written component, which can be used to raise an issue if necessary

See Also

Other write functions: `write_array()`, `write_distribution()`, `write_estimate()`

Index