Package: workspace (via r-universe)

May 27, 2026

Title Manage Collections of Datasets and Objects

Version 0.1.6

Description Create, store, read and manage structured collections of datasets and other objects using a 'workspace', then bundle it into a compressed archive. Using open and interoperable formats makes it possible to exchange bundled data from 'R' to other languages such as 'Python' or 'Julia'. Multiple formats are supported 'Parquet', 'JSON', 'yaml', spatial data and raster data are supported.

License MIT + file LICENSE

URL https://github.com/ardata-fr/workspace

BugReports https://github.com/ardata-fr/workspace/issues

Imports arrow, cli, dplyr, rlang, stringi, tibble, tools, utils, yaml, zip

Suggests sf, terra, testthat (>= 3.0.0), jsonlite

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/pak/sysreqs cmake libicu-dev libssl-dev

Repository https://ardata-fr.r-universe.dev

Date/Publication 2025-09-29 15:39:44 UTC

RemoteUrl https://github.com/ardata-fr/workspace

RemoteRef HEAD

RemoteSha 48026164a75cd7f05b45ecb9e6dd87dc1ac908b7

Contents

`delete_dataset` *Delete a dataset from a workspace*

Description

Delete a dataset stored in a workspace. This function removes the dataset file and updates the workspace's object descriptions.

Usage

```
delete_dataset(x, data_name)
```

Arguments

`x`	The workspace object.
`data_name`	The name of the dataset to delete from the workspace.

Value

Returns the workspace object passed to `x` parameter. Called primarily for side effects.

See Also

workspace for package documentation

Other functions to write in a workspace: `store_dataset()`, `store_json()`, `store_raster()`, `store_rds()`, `store_yaml()`

Examples

```
library(workspace)
dir_tmp <- tempfile(pattern = "ws")
z <- new_workspace(dir = dir_tmp)
z <- store_dataset(x = z, dataset = iris, name = "iris_dataset")
z <- store_dataset(x = z, dataset = mtcars, name = "mtcars")
z <- delete_dataset(x = z, data_name = "iris_dataset")
z
```

---

list_object_in_workspace

List Objects in a Workspace

Description

List all objects stored in a workspace object and returns results as a tibble.

Usage

```
list_object_in_workspace(x)
```

Arguments

x	the workspace

Value

Tibble object containing the following columns:

- `file`: file path to stored object relative to workspace directory
- `name`: name given to object upon storing
- `subdir`: subdirectory of file, such as 'datasets' or 'assets'.
- `type`: file type such as 'dataset', 'geospatial', 'yaml', etc...
- `timestamp`: timestamp of last modification

See Also

workspace for package documentation

Other functions to read in a workspace: `read_dataset_in_workspace()`, `read_json_str_in_workspace()`, `read_raster_in_workspace()`, `read_rds_in_workspace()`, `read_timestamp()`, `read_yaml_in_workspace()`

Examples

```
library(workspace)
dir_tmp <- tempfile(pattern = "ws")
z <- new_workspace(dir = dir_tmp)
z <- store_dataset(x = z, dataset = iris, name = "iris_dataset")
z <- store_dataset(x = z, dataset = mtcars, name = "mtcars")
list_object_in_workspace(z)
```

`new_workspace` *Create a new workspace*

Description

Create a new workspace, i.e. an object made of an R environment and a directory where datasets will be stored as parquet files.

Usage

```
new_workspace(dir = tempfile(pattern = "ws"))
```

Arguments

`dir`	directory used to store datasets, by default uses temporary directory.

Value

an empty workspace object.

See Also

workspace for package documentation

Other functions to manage workspaces: `pack_workspace()`, `unpack_workspace()`, `workspace_bind()`, `workspace_copy()`

Examples

```
z <- new_workspace()
z
```

pack_workspace *Pack a workspace*

Description

Pack a workspace into a compressed file.

Usage

```
pack_workspace(x, file)
```

Arguments

x	workspace object to pack.
file	file path to new compressed file.

Value

path to created compressed file.

See Also

workspace for package documentation

Other functions to manage workspaces: new_workspace(), unpack_workspace(), workspace_bind(), workspace_copy()

Examples

```
library(workspace)
dir_tmp <- tempfile(pattern = "ws")
z <- new_workspace(dir = dir_tmp)
z <- store_dataset(x = z, dataset = iris, name = "iris_dataset")
z <- store_dataset(x = z, dataset = mtcars, name = "mtcars")
json_str <- paste0("{\"first_name\": \"John\",\"last_name\": \"Smith\",\"is_alive\": true,",
"\"age\": 27, \"address\": { \"street_address\": \"21 2nd Street\",",
"\"city\": \"New York\",\"state\": \"NY\",\"postal_code\": \"10021-3100\"",
"}}")
z <- store_json(
  x = z,
  json_str = json_str,
  filename = "example.json",
  timestamp = "2023-11-12 11:37:41",
  subdir = "blah"
)
z <- store_rds(
  x = z,
  obj = mtcars,
  filename = "obj.rds",
  timestamp = "2023-11-12 11:37:41",
```

```
  subdir = "r-object"
)
file <- tempfile(fileext = ".zip")
pack_workspace(x = z, file = file)
```

`read_dataset_in_workspace` *Read a Table from a Workspace.*

Description

Read a table from a dataset stored as parquet file in a workspace.

Usage

```
read_dataset_in_workspace(x, name)
```

Arguments

`x`	the workspace
`name`	name of the dataset stored in the workspace

Value

A `tibble` if the dataset is stored as parquet, and a `sf` object if it is stored as a geospatial dataset (see `store_dataset()` for details).

See Also

workspace for package documentation

Other functions to read in a workspace: `list_object_in_workspace()`, `read_json_str_in_workspace()`, `read_raster_in_workspace()`, `read_rds_in_workspace()`, `read_timestamp()`, `read_yaml_in_workspace()`

Examples

```
library(workspace)
dir_tmp <- tempfile(pattern = "ws")
z <- new_workspace(dir = dir_tmp)
z <- store_dataset(x = z, dataset = iris, name = "iris_dataset")
z <- store_dataset(x = z, dataset = mtcars, name = "mtcars")
read_dataset_in_workspace(z, name = "mtcars")
```

read_json_str_in_workspace *Read JSON String from a Workspace.*

Description

Read a JSON file a dataset stored as a text file in a workspace.

Usage

```
read_json_str_in_workspace(x, name, subdir = NULL)
```

Arguments

x	the workspace
name	name associated with the json file stored in the workspace
subdir	Optional subdirectory used for the asset to retrieve

Value

A character string containing the JSON string.

See Also

workspace for package documentation

Other functions to read in a workspace: list_object_in_workspace(), read_dataset_in_workspace(), read_raster_in_workspace(), read_rds_in_workspace(), read_timestamp(), read_yaml_in_workspace()

Examples

```
library(workspace)
dir_tmp <- tempfile(pattern = "ws")
z <- new_workspace(dir = dir_tmp)
z <- store_dataset(x = z, dataset = iris, name = "iris_dataset")
z <- store_dataset(x = z, dataset = mtcars, name = "mtcars")
json_str <- paste0("{\"first_name\": \"John\",\"last_name\": \"Smith\",\"is_alive\": true,",
"\"age\": 27, \"address\": { \"street_address\": \"21 2nd Street\",",
"\"city\": \"New York\",\"state\": \"NY\",\"postal_code\": \"10021-3100\"",
"}}")
z <- store_json(
  x = z,
  json_str = json_str,
  filename = "example.json",
  name = "an_example",
  timestamp = "2023-11-12 11:37:41",
  subdir = "blah"
)
read_json_str_in_workspace(z, "an_example")
```

`read_raster_in_workspace` *Read a raster from a workspace*

Description

Read a raster dataset stored as a TIFF file in a workspace.

Usage

```
read_raster_in_workspace(x, name)
```

Arguments

`x`	the workspace object
`name`	name of the raster dataset stored in the workspace

Value

The `splatRaster` object that was stored in TIFF format.

See Also

workspace for package documentation

Other functions to read in a workspace: `list_object_in_workspace()`, `read_dataset_in_workspace()`, `read_json_str_in_workspace()`, `read_rds_in_workspace()`, `read_timestamp()`, `read_yaml_in_workspace()`

Examples

```
library(workspace)
dir_tmp <- tempfile(pattern = "ws")
z <- new_workspace(dir = dir_tmp)
# Create and store example raster (requires terra package)
if (requireNamespace("terra", quietly = TRUE)) {
  r <- terra::rast(ncols=10, nrows=10, vals=1:100)
  z <- store_raster(x = z, dataset = r, name = "example_raster")
  retrieved_raster <- read_raster_in_workspace(z, name = "example_raster")
  retrieved_raster
}
```

`read_rds_in_workspace` *Read an R Object from a Workspace.*

Description

Read an R Object from a dataset stored as a RDS file in a workspace.

Usage

```
read_rds_in_workspace(x, name, subdir = NULL)
```

Arguments

`x`	the workspace
`name`	name of the object stored in the workspace
`subdir`	Optional subdirectory used for the asset to retrieve

Value

the R object that was stored as an Rds file.

See Also

workspace for package documentation

Other functions to read in a workspace: `list_object_in_workspace()`, `read_dataset_in_workspace()`, `read_json_str_in_workspace()`, `read_raster_in_workspace()`, `read_timestamp()`, `read_yaml_in_workspace()`

Examples

```
library(workspace)
dir_tmp <- tempfile(pattern = "ws")
z <- new_workspace(dir = dir_tmp)
z <- store_dataset(x = z, dataset = mtcars, name = "mtcars")
z <- store_rds(
  x = z,
  obj = mtcars,
  filename = "obj.rds",
  timestamp = "2023-11-12 11:37:41",
  subdir = "r-object"
)
read_rds_in_workspace(z, name = "obj")
```

`read_timestamp` *Read Timestamp associated with an object in a workspace.*

Description

Read a timestamp associated with an object in a workspace.

Usage

```
read_timestamp(x, name, type, subdir = NULL)
```

Arguments

`x`	the workspace
`name`	name of the object stored in the workspace
`type`	content type
`subdir`	Optional subdirectory used for the asset to retrieve

Value

A character string corresponding to the timestamp (date last modified) of the stored object.

See Also

workspace for package documentation

Other functions to read in a workspace: `list_object_in_workspace()`, `read_dataset_in_workspace()`, `read_json_str_in_workspace()`, `read_raster_in_workspace()`, `read_rds_in_workspace()`, `read_yaml_in_workspace()`

Examples

```
library(workspace)
dir_tmp <- tempfile(pattern = "ws")
z <- new_workspace(dir = dir_tmp)
z <- store_dataset(x = z, dataset = mtcars, name = "mtcars")
z <- store_rds(
  x = z,
  obj = mtcars,
  filename = "obj.rds",
  timestamp = "2023-11-12 11:37:41",
  subdir = "r-object"
)
read_timestamp(z, name = "obj", type = "rds", subdir = "r-object")
```

read_yaml_in_workspace *Read YAML Object from a Workspace.*

Description

Read a YAML file stored as a list object in a workspace. Returns the YAML content as an R list object.

Usage

```
read_yaml_in_workspace(x, name, subdir = NULL)
```

Arguments

x	the workspace
name	name associated with the YAML file stored in the workspace
subdir	Optional subdirectory used for the asset to retrieve

Value

A list object as read from the stored YAML file.

See Also

workspace for package documentation

Other functions to read in a workspace: list_object_in_workspace(), read_dataset_in_workspace(), read_json_str_in_workspace(), read_raster_in_workspace(), read_rds_in_workspace(), read_timestamp()

Examples

```
library(workspace)
dir_tmp <- tempfile(pattern = "ws")
z <- new_workspace(dir = dir_tmp)

config_list <- list(
  database = list(
    host = "localhost",
    port = 5432
  ),
  settings = list(
    debug = TRUE,
    max_connections = 100
  )
)
z <- store_yaml(
  x = z,
```

```
  list = config_list,
  filename = "config.yaml",
  name = "app_config",
  timestamp = "2023-11-12 11:37:41",
  subdir = "configs"
)
read_yaml_in_workspace(z, "app_config", subdir = "configs")
```

`rm_object_in_workspace` *Remove an Object in a Workspace*

Description

Remove an object stored in a workspace.

Usage

```
rm_object_in_workspace(x, name, type, subdir = NULL)
```

Arguments

`x`	the workspace
`name`	name of the object stored in the workspace
`type`	content type
`subdir`	Optional subdirectory used for the asset to retrieve

Value

Returns the workspace object passed to `x` parameter. Called primarily for side effects.

See Also

workspace for package documentation

Examples

```
library(workspace)
dir_tmp <- tempfile(pattern = "ws")
z <- new_workspace(dir = dir_tmp)
z <- store_dataset(x = z, dataset = iris, name = "iris_dataset")
z <- store_dataset(x = z, dataset = mtcars, name = "mtcars")
json_str <- paste0("{\"first_name\": \"John\",\"last_name\": \"Smith\",\"is_alive\": true,",
"\"age\": 27, \"address\": { \"street_address\": \"21 2nd Street\",",
"\"city\": \"New York\",\"state\": \"NY\",\"postal_code\": \"10021-3100\"",
"}}")
z <- store_json(
  x = z,
```

```
  json_str = json_str,
  filename = "example.json",
  timestamp = "2023-11-12 11:37:41",
  subdir = "blah"
)
z <- store_rds(
  x = z,
  obj = mtcars,
  filename = "obj.rds",
  timestamp = "2023-11-12 11:37:41",
  subdir = "r-object"
)
rm_object_in_workspace(
  x = z,
  name = "obj",
  type = "rds",
  subdir = "r-object"
)
```

`store_dataset` *Store a dataset into a workspace*

Description

Store a dataset in an existing workspace. The file format used depends on the class of dataset.

- `data.frame` is written as a Parquet using the `arrow` package.
- `sf` is written as a geopackage file (.gpkg) using the `sf` package.
- `splatRaster` is written as a TIFF file (.tiff) format using `terra` package.

Usage

```
store_dataset(
  x,
  dataset,
  name,
  timestamp = format(Sys.time(), "%Y-%m-%d %H:%M:%S")
)
```

Arguments

`x`	the workspace object
`dataset`	the data.frame or sf to store in the workspace.
`name`	name associated with the data.frame, if a workspace file with this name exists already it will be replaced. The name is translated to ascii using `stringi::stri_trans_general()` to avoid file naming issues.
`timestamp`	A timestamp string to associate with the entry in the workspace.

Value

Returns the workspace object passed to x parameter. Called primarily for side effects.

Geospatial data

To store raster data, ensure that terra is installed. To store geospatial vector and polygon data, ensure that sf is installed.

Storing geospatial vector and polygon data will also create an accompanying metadata yaml file in the assets/sf_metadata/{name}.yaml location, containing metadata for when the file is read.

See Also

workspace for package documentation

Other functions to write in a workspace: delete_dataset(), store_json(), store_raster(), store_rds(), store_yaml()

Examples

```
library(workspace)
dir_tmp <- tempfile(pattern = "ws")
z <- new_workspace(dir = dir_tmp)
z <- store_dataset(x = z, dataset = iris, name = "iris_dataset")
z <- store_dataset(x = z, dataset = mtcars, name = "mtcars")
z
```

---

store_json *Store a JSON string in a workspace*

Description

Saves a JSON string as a file in an existing workspace.

This function allows users to save JSON strings into a specified workspace. The file is saved under the provided filename and can be organized within a specific subdirectory for better management.

Usage

```
store_json(
  x,
  json_str,
  filename,
  name = NULL,
  subdir,
  timestamp = format(Sys.time(), "%Y-%m-%d %H:%M:%S")
)
```

Arguments

x	The workspace object.
json_str	The JSON string to save in the workspace.
filename	The name of the file used to store the JSON string in the workspace.
name	name associated with the object, if a workspace file with this name exists already it will be replaced.
subdir	A subdirectory within the asset directory where the JSON file will be stored.
timestamp	A timestamp string to associate with the entry in the workspace.

Value

Returns the workspace object passed to x parameter. Called primarily for side effects.

See Also

workspace for package documentation

Other functions to write in a workspace: delete_dataset(), store_dataset(), store_raster(), store_rds(), store_yaml()

Examples

```
library(workspace)
dir_tmp <- tempfile(pattern = "ws")
z <- new_workspace(dir = dir_tmp)

json_str <- paste0("{\"first_name\": \"John\",\"last_name\": \"Smith\",\"is_alive\": true,",
"\"age\": 27, \"address\": { \"street_address\": \"21 2nd Street\",",
"\"city\": \"New York\",\"state\": \"NY\",\"postal_code\": \"10021-3100\"",
"}}")
z <- store_json(
  x = z,
  json_str = json_str,
  filename = "example.json",
  timestamp = "2023-11-12 11:37:41",
  subdir = "blah"
)
z
```

---

store_raster *Store a raster dataset in a workspace*

Description

Store a SpatRaster object as a TIFF file into an existing workspace.

Usage

```
store_raster(
  x,
  dataset,
  name,
  timestamp = format(Sys.time(), "%Y-%m-%d %H:%M:%S")
)
```

Arguments

x	the workspace object
dataset	the SpatRaster object to store in the workspace.
name	name associated with the SpatRaster, if a workspace file with this name exists already it will be replaced.
timestamp	A timestamp string to associate with the entry in the workspace.

Value

Returns the workspace object passed to x parameter. Called primarily for side effects.

See Also

workspace for package documentation

Other functions to write in a workspace: delete_dataset(), store_dataset(), store_json(), store_rds(), store_yaml()

Examples

```
library(workspace)
dir_tmp <- tempfile(pattern = "ws")
z <- new_workspace(dir = dir_tmp)
# Create example raster (requires terra package)
if (requireNamespace("terra", quietly = TRUE)) {
  r <- terra::rast(ncols=10, nrows=10, vals=1:100)
  z <- store_raster(x = z, dataset = r, name = "example_raster")
  z
}
```

---

store_rds *Store an RDS file in a workspace*

Description

Saves an R object as an RDS file in an existing workspace. This function allows users to save R objects in a workspace by serializing them as RDS files. The RDS file is saved under the specified filename and organized within a subdirectory for better structure. The timestamp parameter helps to keep track of when the file was stored.

Usage

```
store_rds(
  x,
  obj,
  filename,
  name = NULL,
  subdir,
  timestamp = format(Sys.time(), "%Y-%m-%d %H:%M:%S")
)
```

Arguments

x	The workspace object.
obj	The R object to save as an RDS file.
filename	The name of the file used to store the RDS file in the workspace.
name	name associated with the object, if a workspace file with this name exists already it will be replaced.
subdir	A subdirectory within the asset directory where the RDS file will be stored.
timestamp	A timestamp string to associate with the entry in the workspace.

Value

Returns the workspace object passed to x parameter. Called primarily for side effects.

See Also

workspace for package documentation

Other functions to write in a workspace: `delete_dataset()`, `store_dataset()`, `store_json()`, `store_raster()`, `store_yaml()`

Examples

```
library(workspace)
dir_tmp <- tempfile(pattern = "ws")
z <- new_workspace(dir = dir_tmp)
z <- store_rds(
  x = z,
  obj = mtcars,
  filename = "obj.rds",
  timestamp = "2023-11-12 11:37:41",
  subdir = "r-object"
)
z
```

store_yaml *Store a list as YAML in a workspace*

Description

Saves a list object as a YAML file in an existing workspace.

This function allows users to save R list objects as YAML files into a specified workspace. The file is saved under the provided filename and can be organized within a specific subdirectory for better management.

Usage

```
store_yaml(
  x,
  list,
  filename,
  name = NULL,
  subdir,
  timestamp = format(Sys.time(), "%Y-%m-%d %H:%M:%S")
)
```

Arguments

x	The workspace object.
list	The R list object to save as YAML in the workspace.
filename	The name of the file used to store the YAML file in the workspace.
name	name associated with the object, if a workspace file with this name exists already it will be replaced.
subdir	A subdirectory within the asset directory where the YAML file will be stored.
timestamp	A timestamp string to associate with the entry in the workspace.

Value

Returns the workspace object passed to x parameter. Called primarily for side effects.

See Also

workspace for package documentation

Other functions to write in a workspace: delete_dataset(), store_dataset(), store_json(), store_raster(), store_rds()

Examples

```
library(workspace)
dir_tmp <- tempfile(pattern = "ws")
z <- new_workspace(dir = dir_tmp)

config_list <- list(
  database = list(
    host = "localhost",
    port = 5432,
    name = "mydb"
  ),
  settings = list(
    debug = TRUE,
    max_connections = 100
  )
)
z <- store_yaml(
  x = z,
  list = config_list,
  filename = "config.yaml",
  timestamp = "2023-11-12 11:37:41",
  subdir = "configs"
)
z
```

---

unpack_workspace *Unpack a workspace*

Description

Unpack a compressed file into a workspace object.

Usage

```
unpack_workspace(file)
```

Arguments

file	Packed workspace

Value

Object of class workspace.

See Also

workspace for package documentation

Other functions to manage workspaces: new_workspace(), pack_workspace(), workspace_bind(), workspace_copy()

Examples

```
library(workspace)
dir_tmp <- tempfile(pattern = "ws")
z <- new_workspace(dir = dir_tmp)
z <- store_dataset(x = z, dataset = iris, name = "iris_dataset")
z <- store_dataset(x = z, dataset = mtcars, name = "mtcars")
json_str <- paste0("{\"first_name\": \"John\",\"last_name\": \"Smith\",\"is_alive\": true,",
"\"age\": 27, \"address\": { \"street_address\": \"21 2nd Street\",",
"\"city\": \"New York\",\"state\": \"NY\",\"postal_code\": \"10021-3100\"",
"}}")
z <- store_json(
  x = z,
  json_str = json_str,
  filename = "example.json",
  timestamp = "2023-11-12 11:37:41",
  subdir = "blah"
)
z <- store_rds(
  x = z,
  obj = mtcars,
  filename = "obj.rds",
  timestamp = "2023-11-12 11:37:41",
  subdir = "r-object"
)
file <- tempfile(fileext = ".zip")
pack_workspace(x = z, file = file)

z <- unpack_workspace(file = file)
z
```

`workspace` *Manage Collections of Datasets and Objects*

Description

Create, store, read and manage structured collections of datasets and other objects using a 'workspace', then bundle it into a compressed archive. Using open and interoperable formats makes it possible to exchange bundled data from 'R' to other languages such as 'Python' or 'Julia'. Multiple formats are supported 'Parquet', 'JSON', 'yaml', spatial data and raster data are supported.

Examples of usage are:

- Creating a collection of datasets and parameters to save in a zip file.
- Importing and exporting datasets and settings into from a shiny app.

To get started with workspace, you will need to create an empty one with `new_workspace()` or unpack an existing workspace zip file with `unpack_workspace()`. These functions return a list with class workspace that which you can interact with using the following functions.

Storing and reading workspace data

To store data:

- `store_dataset()` for data.frames, sf and splatRaster
- `store_json()` for json strings
- `store_rds()` for R objects
- store_yaml for storing a list in YAML format

Each object stored in a workspace will have an associated name which can then be used to retrieve the data with the following functions:

- For interactive use, `print()` the workspace to the console to show the directory of the workspace and its contents
- `list_object_in_workspace()` to list objects in a workspace
- `read_timestamp()` to read a timestamp associated with an object of a workspace. Each time an object is stored, the timestamp is recorded
- `read_dataset_in_workspace()` to read a dataset in a workspace
- `read_raster_in_workspace()` to read a raster file in a workspace
- `read_rds_in_workspace()` to read an Rds file in a workspace
- `read_json_str_in_workspace()` to read a JSON string in a workspace
- `read_yaml_in_workspace()` to read a YAML file in a workspace

Stored objects can be removed with:

- `rm_object_in_workspace()` for any object
- `delete_dataset()` for datasets

Workspace management

A workspace an also be compressed into a zip file using `pack_workspace()` and unzipped using `unpack_workspace()`, which allows for saving a workspace to reuse later or share a zip file with someone else.

Workspaces can be cloned using `workspace_copy()` or merged together with `workspace_bind()`.

Dataset formats

Tabular datasets, such as objects of class `data.frame` or `tibble` are stored as `Parquet` files. Geospatial formats are supported using `store_dataset()` for `splatRaster` objects from the `terra` package and `sf` objects from the `sf` package.

What is a workspace?

A `workspace` object is a list with class `workspace` with 2 elements; a directory folder and version number.

The former is where the files are stored and read from, and the latter, contains a version number for the `workspace` structure in case new versions are required in future. A `workspace` therefore represents a folder and its files.

Note, to ensure validity of workspaces, they should be created and modified with functions from this package, not manually.

Author(s)

Maintainer: Eli Daniels <eli.daniels@ardata.fr>

Authors:

- David Gohel <david.gohel@ardata.fr>

Other contributors:

- ArData [copyright holder, funder]

See Also

Useful links:

- https://github.com/ardata-fr/workspace
- Report bugs at https://github.com/ardata-fr/workspace/issues

---

`workspace_bind` *Bind a Workspace into Another Workspace*

Description

Bind a workspace into another workspace.

Usage

```
workspace_bind(x, y, replace = FALSE)
```

Arguments

`x`	the workspace where y elements will be copied
`y`	the workspace to add to `x` workspace
`replace`	scalar logical, set to TRUE if you want to overwrite existing elements.

Value

workspace object resulting from bind operation.

See Also

workspace for package documentation

Other functions to manage workspaces: `new_workspace()`, `pack_workspace()`, `unpack_workspace()`, `workspace_copy()`

Examples

```
library(workspace)
x <- new_workspace()
x <- store_dataset(x = x, dataset = iris, name = "iris_dataset")
z <- new_workspace()
z <- store_dataset(x = z, dataset = mtcars, name = "mtcars")
json_str <- paste0("{\"first_name\": \"John\",\"last_name\": \"Smith\",\"is_alive\": true,",
"\"age\": 27, \"address\": { \"street_address\": \"21 2nd Street\",",
"\"city\": \"New York\",\"state\": \"NY\",\"postal_code\": \"10021-3100\"",
"}}")
z <- store_json(
  x = z,
  json_str = json_str,
  filename = "example.json",
  timestamp = "2023-11-12 11:37:41",
  subdir = "blah"
)
z <- store_rds(
  x = z,
  obj = mtcars,
  filename = "obj.rds",
  timestamp = "2023-11-12 11:37:41",
  subdir = "r-object"
)

new_x <- workspace_bind(x, z)
new_x
```

---

workspace_copy *Clone a Workspace*

Description

Clone/Copy a workspace. This function is necessary as assignment would not change internal directory that is certainly defined in a temporary directory that will be removed when R session will be stopped.

Usage

```
workspace_copy(x)
```

Arguments

x	the workspace to copy

Value

New workspace copy

See Also

workspace for package documentation

Other functions to manage workspaces: new_workspace(), pack_workspace(), unpack_workspace(), workspace_bind()

Examples

```
library(workspace)
dir_tmp <- tempfile(pattern = "ws")
z <- new_workspace(dir = dir_tmp)
z <- store_dataset(x = z, dataset = iris, name = "iris_dataset")
z <- store_dataset(x = z, dataset = mtcars, name = "mtcars")
json_str <- paste0("{\"first_name\": \"John\",\"last_name\": \"Smith\",\"is_alive\": true,",
"\"age\": 27, \"address\": { \"street_address\": \"21 2nd Street\",",
"\"city\": \"New York\",\"state\": \"NY\",\"postal_code\": \"10021-3100\"",
"}}")
z <- store_json(
  x = z,
  json_str = json_str,
  filename = "example.json",
  timestamp = "2023-11-12 11:37:41",
  subdir = "blah"
)
z <- store_rds(
  x = z,
  obj = mtcars,
  filename = "obj.rds",
  timestamp = "2023-11-12 11:37:41",
  subdir = "r-object"
)

new_z <- workspace_copy(z)
```

Index