

Package ‘hdf5r.Extra’

January 29, 2024

Type Package

Title Extensions for 'HDF5' R Interfaces

Version 0.0.5

Date 2024-01-28

Description Some methods to manipulate 'HDF5' files, extending the 'hdf5r' package. Reading and writing R objects to 'HDF5' formats follow the specification of 'Ann-Data' <<https://anndata.readthedocs.io/en/latest/fileformat-prose.html>>.

Depends R (>= 4.1.0), hdf5r (>= 1.3.8), methods, utils

Imports checkmate, dplyr, easy.utils, Matrix, MatrixExtra, rlang

License MIT + file LICENSE

RoxygenNote 7.2.3

Encoding UTF-8

URL <https://github.com/ycli1995/hdf5r.Extra>

BugReports <https://github.com/ycli1995/hdf5r.Extra/issues>

NeedsCompilation no

Author Yuchen Li [aut, cre]

Maintainer Yuchen Li <ycli1995@outlook.com>

Repository CRAN

Date/Publication 2024-01-29 08:30:07 UTC

R topics documented:

H5-attributes	2
H5-dataset-info	6
h5AbsLinkName	7
h5Backup	8
h5Class	9
h5Copy	10
h5CreateDataset	11
h5CreateFile	13

h5CreateGroup	14
h5Delete	15
h5Exists	16
h5GuessDtype	17
h5List	18
h5Move	19
h5Open	20
h5Overwrite	21
h5Prep	22
h5Read	23
h5ReadDataset	24
h5TryOpen	25
h5Write	26
h5WriteDataset	29
h5WriteScalar	33

Index	35
--------------	-----------

H5-attributes	<i>Manipulate HDF5 attributes</i>
---------------	-----------------------------------

Description

Functions to get, set or delete HDF5 attributes for an existing link.

Usage

```
h5Attr(x, which, ...)
```

```
h5Attributes(x, ...)
```

```
h5AttrNames(x, ...)
```

```
h5DeleteAttr(x, which, ...)
```

```
h5WriteAttr(x, which, robj, ...)
```

```
## S3 method for class 'H5D'  
h5Attr(x, which, ...)
```

```
## S3 method for class 'H5Group'  
h5Attr(x, which, name = NULL, ...)
```

```
## S3 method for class 'H5File'  
h5Attr(x, which, name = NULL, ...)
```

```
## S3 method for class 'character'  
h5Attr(x, which, name = NULL, ...)
```

```
## S3 method for class 'H5D'
h5AttrNames(x, ...)

## S3 method for class 'H5Group'
h5AttrNames(x, name = NULL, ...)

## S3 method for class 'H5File'
h5AttrNames(x, name = NULL, ...)

## S3 method for class 'character'
h5AttrNames(x, name = NULL, ...)

## S3 method for class 'H5D'
h5Attributes(x, ...)

## S3 method for class 'H5Group'
h5Attributes(x, name = NULL, ...)

## S3 method for class 'H5File'
h5Attributes(x, name = NULL, ...)

## S3 method for class 'character'
h5Attributes(x, name = NULL, ...)

## S3 method for class 'H5D'
h5WriteAttr(
  x,
  which,
  robj,
  overwrite = TRUE,
  check.scalar = TRUE,
  stype = c("utf8", "ascii7"),
  ...
)

## S3 method for class 'H5Group'
h5WriteAttr(
  x,
  which,
  robj,
  name = NULL,
  overwrite = TRUE,
  check.scalar = TRUE,
  stype = c("utf8", "ascii7"),
  ...
)
```

```

## S3 method for class 'H5File'
h5WriteAttr(
  x,
  which,
  robj,
  name = NULL,
  overwrite = TRUE,
  check.scalar = TRUE,
  stype = c("utf8", "ascii7"),
  ...
)

## S3 method for class 'character'
h5WriteAttr(
  x,
  which,
  robj,
  name = NULL,
  overwrite = TRUE,
  check.scalar = TRUE,
  stype = c("utf8", "ascii7"),
  ...
)

## S3 method for class 'H5D'
h5DeleteAttr(x, which, ...)

## S3 method for class 'H5Group'
h5DeleteAttr(x, which, name = NULL, ...)

## S3 method for class 'H5File'
h5DeleteAttr(x, which, name = NULL, ...)

## S3 method for class 'character'
h5DeleteAttr(x, which, name = NULL, ...)

```

Arguments

<code>x</code>	An H5File , H5Group , H5D or a path name of HDF5 file.
<code>which</code>	Name of the HDF5 attribute
<code>...</code>	Arguments passed to other methods.
<code>robj</code>	An R object to be written as HDF5 attribute
<code>name</code>	Name of an existing HDF5 sub-link. Default is NULL, which will use current link.
<code>overwrite</code>	Whether or not to overwrite the existing HDF5 attribute.
<code>check.scalar</code>	Whether or not to use scalar space when <code>robj</code> is a scalar. If FALSE, the attribute written will be treated as an array.

stype Passed to [h5GuessDtype](#)

Value

H5Attr:

- If which exists in link name, will return an R object representing the attribute. If which doesn't exist or contains empty data, will return NULL.
- If name doesn't exist, will raise an error from `H5File$attr_exists_by_name()`.

`h5AttrNames` will return a character vector containing all attribute names for the given link.

`h5Attributes` will return a list containing all attributes for the given link.

Examples

```
file <- system.file("extdata", "pbmc_small.h5ad", package = "hdf5r.Extra")

# Read H5 attribute
x <- h5Attr(file, "encoding-version")
x <- h5Attr(file, "column-order", "raw/var") ## An empty attribute
stopifnot(length(x) == 0)

h5obj <- h5Open(file, "raw/var", mode = "r")
x <- h5Attr(h5obj, "column-order")

# Read H5 attribute names
h5AttrNames(file)
h5AttrNames(file, "X")
h5AttrNames(h5obj)

# Read all H5 attributes
a1 <- h5Attributes(file, "raw/var")
a2 <- h5Attributes(h5obj)
stopifnot(identical(a1, a2))

# Write H5 attribute
tmp.file <- tempfile(fileext = ".h5")
file.copy(file, tmp.file)

new_a <- character() # Can write an empty attribute
h5WriteAttr(tmp.file, "new_a", robj = new_a, name = "X")
new_a <- c("a", "b")
h5WriteAttr(tmp.file, "new_a", robj = new_a, name = "X", overwrite = TRUE)
h5Attr(tmp.file, "new_a", name = "X")

# Delete H5 attribute
h5DeleteAttr(tmp.file, "new_a", name = "X")
stopifnot(length(h5Attr(tmp.file, "new_a", name = "X")) == 0)
```

Description

Functions to get the information from an HDF5 dataset.

Usage

```
h5Dims(x, ...)  
  
h5MaxDims(x, ...)  
  
## S3 method for class 'H5D'  
h5Dims(x, ...)  
  
## S3 method for class 'H5Group'  
h5Dims(x, name, ...)  
  
## S3 method for class 'H5File'  
h5Dims(x, name, ...)  
  
## S3 method for class 'character'  
h5Dims(x, name, ...)  
  
## S3 method for class 'H5D'  
h5MaxDims(x, ...)  
  
## S3 method for class 'H5D'  
h5MaxDims(x, ...)  
  
## S3 method for class 'H5Group'  
h5MaxDims(x, name, ...)  
  
## S3 method for class 'H5File'  
h5MaxDims(x, name, ...)  
  
## S3 method for class 'character'  
h5MaxDims(x, name, ...)
```

Arguments

x	An H5File , H5Group , H5D or a path name of HDF5 file.
...	Arguments passed to other methods.
name	A link in file. Must represent an H5D. Used when x is an H5Group, H5File or an HDF5 file.

Value

h5Dims returns the dimension of the HDF5 dataset.

h5MaxDims returns the maximal dimension of the HDF5 dataset.

See Also

[H5D-class](#)

Examples

```
file <- system.file("extdata", "pbmc_small.h5ad", package = "hdf5r.Extra")
h5obj <- h5open(file, "X", mode = "r")

h5Dims(file, "X")
h5Dims(h5obj)

h5MaxDims(file, "X")
h5MaxDims(h5obj)
```

h5AbsLinkName	<i>Format an absolute path name for HDF5 link</i>
---------------	---

Description

Format an absolute path name for HDF5 link

Usage

```
h5AbsLinkName(name)
```

Arguments

name String representing an expected name of HDF5 link.

Details

If name contains any of "", NA or NULL, will simply return "/".

Value

An update name starting with '/'.

Examples

```

h5AbsLinkName("ggg")
h5AbsLinkName("ggg/ddd")
h5AbsLinkName(NA)
h5AbsLinkName("")
h5AbsLinkName(NULL)

```

h5Backup

Back up contents from one HDF5 file to another

Description

Function to back up HDF5 file, with optionally excluding specific links.

Usage

```

h5Backup(
  from.file,
  to.file = NULL,
  exclude = NULL,
  overwrite = FALSE,
  verbose = TRUE,
  ...
)

```

Arguments

from.file	The source HDF5 file.
to.file	The target HDF5 file. Cannot be the same file as from.file. If NULL, will generate an R temp file.
exclude	Names of HDF5 links not to be backed up.
overwrite	When the to.file already exists, whether or not to overwrite it.
verbose	Print progress.
...	Arguments passed to <code>H5File\$obj_copy_from()</code>

Details

When any HDF5 link is to be excluded, it will copy the rest of links from from.file using [h5Copy](#). Otherwise, it will simply copy the from.file to the to.file via [file.copy](#)

Value

Path of the to.file

Examples

```
file <- system.file("extdata", "pbmc_small.h5ad", package = "hdf5r.Extra")
to.file <- tempfile(fileext = ".h5")

h5Backup(file, to.file, exclude = "X")

x <- h5Read(file)
x2 <- h5Read(to.file)
x$X <- NULL # Remove 'X'
stopifnot(identical(x, x2)) # Now these two should be identical
```

h5Class	<i>Get the class of an HDF5 link</i>
---------	--------------------------------------

Description

Functions to get or check the class of an HDF5 link.

Usage

```
h5Class(file, name)

is.H5D(file, name)

is.H5Group(file, name)
```

Arguments

file	An existing HDF5 file
name	Name of a link in file

Value

h5Class returns a character specifying the class of the query HDF5 link (typically H5D, H5Group or H5File).

is.H5D and is.H5Group return a logical value.

Examples

```
file <- system.file("extdata", "pbmc_small.h5ad", package = "hdf5r.Extra")
h5Class(file, "X")
h5Class(file, "obs")
is.H5D(file, "X")
is.H5Group(file, "obs")
```

h5Copy*Copy an HDF5 link*

Description

Copy an HDF5 link from one file to another file.

Usage

```
h5Copy(  
  from.file,  
  from.name,  
  to.file,  
  to.name,  
  overwrite = FALSE,  
  verbose = TRUE,  
  ...  
)
```

Arguments

<code>from.file</code>	The source HDF5 file.
<code>from.name</code>	The source link name.
<code>to.file</code>	The target HDF5 file.
<code>to.name</code>	The destination HDF5 link name.
<code>overwrite</code>	Whether or not to overwrite the existing link.
<code>verbose</code>	Print progress.
<code>...</code>	Arguments passed to <code>H5File\$obj_copy_from()</code>

Value

This is an operation function and no return. Any failure should raise an error.

Note

- Copying can still work even if the `to.file` is actually identical to the `from.file`.
- Attributes of `from.name` will be kept, while those of its parent `H5Groups` will not.

See Also

[H5File](#)

Examples

```

file <- system.file("extdata", "pbmc_small.h5ad", package = "hdf5r.Extra")
to.file <- tempfile(fileext = ".h5")

# Copy a link to a new file
h5Copy(file, "obs", to.file, "obs")
obs <- h5Read(file, "obs")
obs2 <- h5Read(to.file, "obs")
stopifnot(identical(obs, obs2))

# The parent link (H5Group) will be created automatically
h5Copy(file, "obsm/tsne", to.file, "obsm/tsne")
obsm <- h5Read(to.file, "obsm")

# Copy the whole file
x <- h5Read(file)
h5Copy(file, "/", to.file, "/", overwrite = TRUE)
x2 <- h5Read(to.file)
stopifnot(identical(x, x2))

```

h5CreateDataset

Create a new empty HDF5 dataset

Description

Create a new empty HDF5 dataset

Usage

```

h5CreateDataset(x, name, ...)

## S3 method for class 'H5Group'
h5CreateDataset(
  x,
  name,
  dims,
  dtype = NULL,
  storage.mode = numeric(),
  stype = c("utf8", "ascii7"),
  maxdims = NULL,
  chunk_size = "auto",
  gzip_level = 6,
  ...
)

## S3 method for class 'H5File'
h5CreateDataset(

```

```

    x,
    name,
    dims,
    dtype = NULL,
    storage.mode = numeric(),
    stype = c("utf8", "ascii7"),
    maxdims = NULL,
    chunk_size = "auto",
    gzip_level = 6,
    ...
)

## S3 method for class 'character'
h5CreateDataset(
  x,
  name,
  dims,
  dtype = NULL,
  storage.mode = numeric(),
  stype = c("utf8", "ascii7"),
  maxdims = NULL,
  overwrite = FALSE,
  chunk_size = "auto",
  gzip_level = 6,
  ...
)

```

Arguments

<code>x</code>	An H5File , H5Group or a path name of HDF5 file.
<code>name</code>	Name of the new HDF5 dataset.
<code>...</code>	Arguments passed to <code>H5File\$create_dataset()</code> . Also see <code>[hdf5r:H5File-class]{H5File}</code> .
<code>dims</code>	Dimensions of the new dataset.
<code>dtype</code>	The H5 datatype to use for the creation of the object. Must be an H5T . If set to <code>NULL</code> , it will be guessed through h5GuessDtype , according to <code>storage.mode</code> .
<code>storage.mode</code>	Object used to guess the HDF5 datatype. Default is <code>numeric()</code> .
<code>stype</code>	'utf8' or 'ascii7'. Passed to h5GuessDtype .
<code>maxdims</code>	The maximal dimensions of the space. Default is <code>dims</code> .
<code>chunk_size</code>	Size of the chunk. Must have the same length as the dataset dimension. If <code>NULL</code> , no chunking is used. If set to "auto", the size of each chunk will be estimated according to <code>maxdims</code> and the byte size of <code>dtype</code> , using guess_chunks .
<code>gzip_level</code>	Enable zipping at the level given here. Only if <code>chunk_dims</code> is not <code>NULL</code> .
<code>overwrite</code>	Whether or not to overwrite the existing HDF5 dataset.

Value

This is an operation function and no return. Any failure should raise an error.

See Also

[H5File](#) and [H5Group](#) for the `$create_dataset()` methods.

Examples

```
tmp.file <- tempfile(fileext = ".h5")
h5CreateFile(tmp.file)

m <- matrix(0, 10, 5)
h5CreateDataset(tmp.file, "g1/m", dim(m))

m2 <- c("a", "b", "c")
h5CreateDataset(tmp.file, "g2/m2", length(m2), storage.mode = m2)
```

h5CreateFile	<i>Create a new HDF5 file</i>
--------------	-------------------------------

Description

A wrapper for [H5File](#)`$new()`. If file exists, will only raise a warning.

Usage

```
h5CreateFile(x, ...)

## S3 method for class 'character'
h5CreateFile(x, ...)
```

Arguments

`x` Name of the new HDF5 file.
`...` Arguments passed to [H5File](#)`$new()`

Value

This is an operation function and no return. Any failure should raise an error.

Examples

```
tmp.file <- tempfile(fileext = ".h5")
h5CreateFile(tmp.file)
```

h5CreateGroup	<i>Create new HDF5 group</i>
---------------	------------------------------

Description

Create new HDF5 group

Usage

```
h5CreateGroup(x, name, ...)  
  
## S3 method for class 'H5Group'  
h5CreateGroup(x, name, show.warnings = TRUE, ...)  
  
## S3 method for class 'H5File'  
h5CreateGroup(x, name, show.warnings = TRUE, ...)  
  
## S3 method for class 'character'  
h5CreateGroup(x, name, show.warnings = TRUE, ...)
```

Arguments

x	An H5File , H5Group or a path name of HDF5 file.
name	Name of the new HDF5 group. Can be recursive, such as "g/sub_g".
...	Arguments passed to <code>H5Group\$create_group()</code> .
show.warnings	When the group name already exists, whether or not to show warning messages.

Value

This is an operation function and no return. Any failure should raise an error.

See Also

[H5Group](#)

Examples

```
tmp.file <- tempfile(fileext = ".h5")  
h5CreateFile(tmp.file)  
  
h5CreateGroup(tmp.file, "g1")  
h5CreateGroup(tmp.file, "g2/g3")
```

h5Delete	<i>Delete an HDF5 link</i>
----------	----------------------------

Description

Delete an HDF5 link

Usage

```
h5Delete(x, name, ...)  
  
## S3 method for class 'H5Group'  
h5Delete(x, name, verbose = TRUE, ...)  
  
## S3 method for class 'H5File'  
h5Delete(x, name, verbose = TRUE, ...)  
  
## S3 method for class 'character'  
h5Delete(x, name, verbose = TRUE, ...)
```

Arguments

x	An existing HDF5 file
name	Name of HDF5 link to be deleted. If name doesn't exist, nothing will be done.
...	Arguments passed to <code>H5Group\$link_delete()</code>
verbose	Print progress.

Value

This is an operation function and no return. Any failure should raise an error.

Examples

```
file <- system.file("extdata", "pbmc_small.h5ad", package = "hdf5r.Extra")  
to.file <- tempfile(fileext = ".h5")  
file.copy(file, to.file)  
  
h5Delete(to.file, "obs")  
h5Delete(to.file, "xxxx") # Delete something not existing.
```

h5Exists	<i>Check existence of an HDF5 link</i>
----------	--

Description

Check existence of an HDF5 link

Usage

```
h5Exists(x, name, ...)  
  
## S3 method for class 'H5Group'  
h5Exists(x, name, ...)  
  
## S3 method for class 'H5File'  
h5Exists(x, name, ...)  
  
## S3 method for class 'character'  
h5Exists(x, name, ...)
```

Arguments

x	An H5File , H5Group or a path name of HDF5 file
name	Name of HDF5 link to be checked.
...	Arguments passed to <code>H5File\$exists()</code>

Value

If any parent directory of name doesn't exist, will simply return FALSE

Examples

```
file <- system.file("extdata", "pbmc_small.h5ad", package = "hdf5r.Extra")  
h5Exists(file, "/")  
h5Exists(file, "obs")  
h5Exists(file, "X")  
  
h5fh <- h5TryOpen(file, mode = "r")  
h5Exists(h5fh, "obs")  
  
h5obj <- h5open(h5fh, "obs")  
h5Exists(h5obj, "groups")
```

h5GuessDtype	<i>Guess an HDF5 Datatype</i>
--------------	-------------------------------

Description

Wrapper around `hdf5r::guess_dtype`, allowing for the customization of string types such as utf-8 rather than defaulting to variable-length ASCII-encoded strings.

Usage

```
h5GuessDtype(x, stype = c("utf8", "ascii7"), ...)
```

Arguments

<code>x</code>	The object for which to guess the HDF5 datatype
<code>stype</code>	'utf8' or 'ascii7'
<code>...</code>	Arguments passed to <code>hdf5r::guess_dtype</code>

Value

An object of class `H5T`

References

<https://github.com/mojaveazure/seurat-disk/blob/163f1aade5bac38ed1e9e9c912283a7e74781610/R/zzz.R>

See Also

[guess_dtype](#)

Examples

```
h5GuessDtype(0)
h5GuessDtype("abc")
```

h5List	<i>List the contents of an HDF5 group</i>
--------	---

Description

Function to list the contents of an HDF5 group.

Usage

```
h5List(x, ...)

## S3 method for class 'H5Group'
h5List(
  x,
  recursive = FALSE,
  full.names = FALSE,
  simplify = TRUE,
  detailed = FALSE,
  ...
)

## S3 method for class 'H5File'
h5List(
  x,
  name = "/",
  recursive = FALSE,
  full.names = FALSE,
  simplify = TRUE,
  detailed = FALSE,
  ...
)

## S3 method for class 'character'
h5List(
  x,
  name = "/",
  recursive = FALSE,
  full.names = FALSE,
  simplify = TRUE,
  detailed = FALSE,
  ...
)
```

Arguments

x	An H5File , H5Group or a path name of HDF5 file.
...	Additional parameters passed to <code>\$ls()</code>

recursive	If TRUE, the contents of the whole group hierarchy will be listed.
full.names	Whether or not to return the absolute object path names.
simplify	Whether or not to only return the object names.
detailed	Whether or not to show the detailed information.
name	A link in file. Must refer to an H5Group. Default is "/".

Value

If simplify, will return a character vector specifying names of H5 links, otherwise will return a data.frame to show details.

See Also

[H5Group\\$ls\(\)](#)

Examples

```
file <- system.file("extdata", "pbmc_small.h5ad", package = "hdf5r.Extra")

h5List(file)
h5List(file, "obs")
h5List(file, recursive = TRUE)
h5List(file, "obs", simplify = FALSE, recursive = TRUE)

h5g <- h5open(file, "obs", mode = "r")
h5List(h5g)
```

h5Move

Move link in an HDF5 file

Description

Move one HDF5 link to another position within the same file.

Usage

```
h5Move(file, from.name, to.name, overwrite = FALSE, verbose = TRUE, ...)
```

Arguments

file	An HDF5 file.
from.name	Name of the source link.
to.name	Name of the destination link.
overwrite	When to.name already exists, whether or not to overwrite it.
verbose	Print progress.
...	Arguments passed to H5File\$link_move_from()

Value

This is an operation function and no return. Any failure should raise an error.

See Also

[H5File](#)

Examples

```
file <- system.file("extdata", "pbmc_small.h5ad", package = "hdf5r.Extra")
to.file <- tempfile(fileext = ".h5")
file.copy(file, to.file)

obs <- h5Read(to.file, "obs")
h5Move(to.file, "obs", "obs2")
obs2 <- h5Read(to.file, "obs2")
stopifnot(identical(obs, obs2))

# Move an object to an existing link
h5Move(to.file, "obs2", "var") # Warning
h5Move(to.file, "obs2", "var", overwrite = TRUE)

# Move a non-existing object will raise an error
try(h5Move(to.file, "obs", "obs3"))
```

h5open

Open an HDF5 file, file-handler or group object

Description

Open an HDF5 file, file-handler or group object

Usage

```
h5open(x, name, ...)

## S3 method for class 'H5Group'
h5open(x, name, ...)

## S3 method for class 'H5File'
h5open(x, name, ...)

## S3 method for class 'character'
h5open(x, name, mode = c("a", "r", "r+", "w", "w-", "x"), ...)
```

Arguments

x	An H5File , H5Group or a path name of HDF5 file.
name	Name of the opened HDF5 link.
...	Arguments passed to <code>H5Group\$open()</code> .
mode	Passed to h5TryOpen

Value

An opened [H5File](#), [H5Group](#) or [H5D](#).

Examples

```
file <- system.file("extdata", "pbmc_small.h5ad", package = "hdf5r.Extra")
obs <- h5open(file, "obs", mode = "r")
stopifnot(inherits(obs, "H5Group"))
tsne <- h5open(file, "obsm/tsne", mode = "r")
stopifnot(inherits(tsne, "H5D"))
```

h5overwrite	<i>Overwrite an existing HDF5 link</i>
-------------	--

Description

Overwrite an existing HDF5 link

Usage

```
h5overwrite(file, name, overwrite)
```

Arguments

file	An existing HDF5 file
name	Name of HDF5 link to be overwritten.
overwrite	Whether or not to overwrite name.

Details

- When file doesn't exist, will create it.
- When the old link name doesn't exist, will simply return file.
- When name exists and overwrite is TRUE, will copy the rest of HDF5 links to an updated file with [h5Backup](#). If name is "/", will create a new file and overwrite the old one.
- When name exists and overwrite is FALSE, will raise an error.

Value

Path to file which is ready to be written.

Examples

```
file <- system.file("extdata", "pbmc_small.h5ad", package = "hdf5r.Extra")
tmp.file <- tempfile(fileext = ".h5")
file.copy(file, tmp.file)

obs <- h5Read(tmp.file, "obs")

h5Overwrite(tmp.file, "layers", TRUE)
stopifnot(!h5Exists(tmp.file, "layers"))

# You can still read other links.
obs2 <- h5Read(tmp.file, "obs")
stopifnot(identical(obs, obs2))
```

h5Prep

Prepare an R object to be written into HDF5 file

Description

Methods to transform a complex R object (for example, S4 object) into combination of base R objects, such as vector, array, data.frame or list, so that it can be written into HDF5 file.

Usage

```
h5Prep(x, ...)
```

Default S3 method:
h5Prep(x, ...)

Arguments

x	The R object to be transformed
...	Arguments to be passed to other methods

Details

In this package, h5Prep will return x itself by default. Extended methods can be easily added for specific S4 class.

Value

An R object of converted x.

h5Read	<i>Read data from an existing HDF5 link</i>
--------	---

Description

Function to read data from an existing HDF5 group.

Usage

```
h5Read(x, name = NULL, ...)  
  
## S3 method for class 'H5Group'  
h5Read(x, name = NULL, transpose = FALSE, toS4.func = NULL, ...)  
  
## S3 method for class 'H5File'  
h5Read(x, name = NULL, transpose = FALSE, toS4.func = NULL, ...)  
  
## S3 method for class 'character'  
h5Read(x, name = NULL, transpose = FALSE, toS4.func = NULL, ...)
```

Arguments

x	An H5File , H5Group or a path name of HDF5 file.
name	Name of the HDF5 link to be read.
...	Arguments passed to h5ReadDataset .
transpose	Whether or not to transpose the read matrix. Only works for a 2-dimension array-like data.
toS4.func	A function to convert the read R list into an S4 object.

Value

The load R object. Currently support vector, matrix, data.frame, list and sparse matrix (dgCMatrix and dgRMatrix).

Examples

```
file <- system.file("extdata", "pbmc_small.h5ad", package = "hdf5r.Extra")  
  
# Read a matrix  
x <- h5Read(file, name = "X")  
x <- h5Read(file, name = "X", transpose = TRUE)  
x <- h5Read(file, name = "X", idx_list = list(1:10, 1:20))  
x <- h5Read(  
  file,  
  name = "X",  
  idx_list = list(1:10, 1:20),  
  transpose = TRUE
```

```

)

# Read a dgCMatix
x <- h5Read(file, name = "raw/X")
x <- h5Read(file, name = "raw/X", transpose = TRUE)

# Read a data.frame
x <- h5Read(file, name = "obs")
x <- h5Read(file, name = "raw/var") # Read a data.frame with empty column

# Read a list
x <- h5Read(file)
x <- h5Read(file, "raw")
x <- h5Read(file, "obsm")

```

h5ReadDataset

Read data from an existing H5 dataset

Description

Low-level helper function to read atomic R data from an existing H5 dataset.

Usage

```

h5ReadDataset(x, ...)

## S3 method for class 'H5D'
h5ReadDataset(x, idx_list = NULL, transpose = FALSE, ...)

## S3 method for class 'H5Group'
h5ReadDataset(x, name, idx_list = NULL, transpose = FALSE, ...)

## S3 method for class 'H5File'
h5ReadDataset(x, name, idx_list = NULL, transpose = FALSE, ...)

## S3 method for class 'character'
h5ReadDataset(x, name, transpose = FALSE, idx_list = NULL, ...)

```

Arguments

x	An H5File , H5Group , H5D or a path name of HDF5 file.
...	Arguments passed to <code>H5D\$read()</code> .
idx_list	The indices for each dimension of name to subset given as a list. If NULL, the entire dataset will be read. Passed to <code>H5D\$read(args)</code> .
transpose	Whether or not to transpose the read matrix. Only works for a 2-dimension array-like data.
name	Name of the HDF5 link to be read. Must be an H5 dataset.

Value

An array-like object with the data read.

Examples

```
file <- system.file("extdata", "pbmc_small.h5ad", package = "hdf5r.Extra")

x <- h5ReadDataset(file, name = "X")
x <- h5ReadDataset(file, name = "X", transpose = TRUE)
x <- h5ReadDataset(file, name = "X", idx_list = list(1:10, 1:20))
x <- h5ReadDataset(
  file,
  name = "X",
  idx_list = list(1:10, 1:20),
  transpose = TRUE
)
```

h5TryOpen

Automatically retry opening HDF5 file

Description

Helper function to open an HDF5 file. When the opening fails, will retry it until reach a timeout.

Usage

```
h5TryOpen(
  filename,
  mode = c("a", "r", "r+", "w", "w-", "x"),
  timeout = getOption(x = "h5TryOpen.timeout", default = 0),
  interval = getOption(x = "h5TryOpen.interval", default = 0),
  ...
)
```

Arguments

filename	An HDF5 file to open
mode	How to open it: <ul style="list-style-type: none"> • a creates a new file or opens an existing one for read/write. • r opens an existing file for reading • r+ opens an existing file for read/write. • w creates a file, truncating any existing ones. • w- and x are synonyms, creating a file and failing if it already exists.
timeout	Positive integer. The timeout for retrying.
interval	Positive integer. The interval seconds of retrying.
...	Arguments passed to H5File\$new()

Details

timeout and interval must be positive. Otherwise no retrying, which is default setting.

Value

When file is opened successfully, an [H5File](#) will be returned. Otherwise, will keep retrying. When a timeout is reached, will raise an error and terminate the current R session.

See Also

[H5File](#) for mode

Examples

```
file <- system.file("extdata", "pbmc_small.h5ad", package = "hdf5r.Extra")
h5fh <- h5TryOpen(file, mode = "r")
h5fh
h5fh$close_all()
```

h5Write

Write an R object to HDF5 file

Description

Methods to write an R object to an HDF5 file.

Usage

```
h5Write(x, file, name, ...)

## Default S3 method:
h5Write(x, file, name, overwrite = FALSE, gzip_level = 6, ...)

## S3 method for class 'array'
h5Write(
  x,
  file,
  name,
  overwrite = FALSE,
  transpose = FALSE,
  block_size = 5000L,
  gzip_level = 6,
  ...
)

## S3 method for class 'factor'
```

```

h5Write(x, file, name, overwrite = FALSE, ordered = TRUE, gzip_level = 6, ...)

## S3 method for class 'data.frame'
h5Write(x, file, name, overwrite = FALSE, gzip_level = 6, ...)

## S3 method for class 'dgCMatrix'
h5Write(
  x,
  file,
  name,
  overwrite = FALSE,
  transpose = FALSE,
  add.shape = FALSE,
  dimnames = list(),
  gzip_level = 6,
  ...
)

## S3 method for class 'dgRMatrix'
h5Write(
  x,
  file,
  name,
  overwrite = FALSE,
  transpose = FALSE,
  add.shape = FALSE,
  dimnames = list(),
  gzip_level = 6,
  ...
)

## S3 method for class 'list'
h5Write(x, file, name, overwrite = FALSE, gzip_level = 6, ...)

```

Arguments

x	An R object to be written
file	An existing HDF5 file
name	Name of the HDF5 link to be written into
...	Arguments passed to other methods.
overwrite	Whether or not to overwrite the existing HDF5 link.
gzip_level	Enable zipping at the level given here.
transpose	Whether or not to transpose the input matrix. Only works for a 2-dimension array-like object.
block_size	Default size for number of columns when transpose is TRUE.
ordered	When writing a factor, whether or not the categories are ordered.

add.shape	When writing a CSC- or CSR-matrix, whether or not to also write the number of dimensions into an HDF5 dataset.
dimnames	When writing a CSC- or CSR-matrix, whether or not to also write the dimension names.

Details

By default, `h5Write` will try to transform any S4 object `x` into combination of base R objects using `h5Prep` before writing it.

Value

This is an operation function and no return. Any failure should raise an error.

References

<https://anndata.readthedocs.io/en/latest/fileformat-prose.html>

Examples

```
file <- system.file("extdata", "pbmc_small.h5ad", package = "hdf5r.Extra")
tmp.file <- tempfile(fileext = ".h5")
h5CreateFile(tmp.file)

# vector -----
x <- h5Read(file, "/raw/X/data")
h5Write(x, tmp.file, "raw/X/data")
x2 <- h5Read(tmp.file, "raw/X/data")
stopifnot(identical(x, x2))

# matrix -----
x <- h5Read(file, "X")
h5Write(x, tmp.file, "X")
x2 <- h5Read(tmp.file, "X")
stopifnot(identical(x, x2))

h5Write(x, tmp.file, "X2", transpose = TRUE)
x2 <- h5Read(tmp.file, "X2")
stopifnot(identical(t(x), x2))

# data.frame -----
x <- h5Read(file, "obs")
h5Write(x, tmp.file, "obs")
x2 <- h5Read(tmp.file, "obs")
stopifnot(identical(x, x2))

x <- h5Read(file, "raw/var") # data.frame with empty column
```

```
h5Write(x, tmp.file, "raw/var")
x2 <- h5Read(tmp.file, "raw/var")
stopifnot(identical(x, x2))

# dgMatrix -----
x <- h5Read(file, "raw/X")
h5Write(x, tmp.file, "raw/X", overwrite = TRUE)
x2 <- h5Read(tmp.file, "raw/X")
stopifnot(identical(x, x2))

# list -----
x <- h5Read(file)
h5Write(x, tmp.file, name = NULL, overwrite = TRUE)
x2 <- h5Read(tmp.file)
stopifnot(identical(x, x2))
```

h5WriteDataset

Write array-like data into an existing H5 dataset

Description

Low-level helper function to write atomic R data into an existing H5 dataset. All data written will be treated as array for HDF5.

Usage

```
h5WriteDataset(x, robj, ...)

## S3 method for class 'H5D'
h5WriteDataset(
  x,
  robj,
  idx_list = NULL,
  transpose = FALSE,
  block_size = 5000L,
  verbose = TRUE,
  ...
)

## S3 method for class 'H5Group'
h5WriteDataset(
  x,
  robj,
```

```

    name,
    idx_list = NULL,
    transpose = FALSE,
    block_size = 5000L,
    verbose = TRUE,
    ...
)

## S3 method for class 'H5File'
h5WriteDataset(
  x,
  robj,
  name,
  idx_list = NULL,
  transpose = FALSE,
  block_size = 5000L,
  verbose = TRUE,
  ...
)

## S3 method for class 'character'
h5WriteDataset(
  x,
  robj,
  name,
  idx_list = NULL,
  transpose = FALSE,
  block_size = 5000L,
  verbose = TRUE,
  ...
)

```

Arguments

x	An H5File , H5Group , H5D or a path name of HDF5 file.
robj	An R array.
...	Arguments passed to <code>H5D\$write()</code> .
idx_list	The indices for each dimension of name to subset given as a list. If NULL, the entire dataset will be use. Passed to <code>H5D\$write(args)</code>
transpose	Whether or not to transpose the input matrix. Only works for a 2-dimension array-like object.
block_size	Default size for number of columns to transpose in a single writing. Increasing <code>block_size</code> may speed up but at an additional memory cost.
verbose	Print progress.
name	Name of the HDF5 dataset to be written.

Value

This is an operation function and no return. Any failure should raise an error.

Note

If you want to write `robj` into scalar space, you should use [h5WriteScalar](#).

Examples

```
tmp.file <- tempfile(fileext = ".h5")
h5CreateFile(tmp.file)

# Scalar (will be written into array space for HDF5) #####
h5CreateDataset(
  tmp.file,
  name = "test/bool",
  dims = 1,
  storage.mode = logical()
) # Must create the dataset first
h5WriteDataset(tmp.file, FALSE, name = "test/bool")
x <- h5Read(tmp.file, name = "test/bool")
stopifnot(!x)

h5CreateDataset(tmp.file, name = "test/num", dims = 1)
h5WriteDataset(tmp.file, 100.0, name = "test/num")
x <- h5Read(tmp.file, name = "test/num")
stopifnot(identical(x, 100.0))

h5CreateDataset(
  tmp.file,
  name = "test/string",
  dims = 1,
  storage.mode = character()
)
h5WriteDataset(tmp.file, "ABC", name = "test/string")
x <- h5Read(tmp.file, name = "test/string")
stopifnot(identical(x, "ABC"))

# Vector (1d array) #####
x1 <- rep(FALSE, 10)
h5CreateDataset(
  tmp.file,
  name = "vec/bool",
  dims = 10,
  storage.mode = logical()
)
h5WriteDataset(tmp.file, x1, name = "vec/bool")
x <- h5Read(tmp.file, name = "vec/bool")
stopifnot(identical(x, x1))

x1 <- rep(1.1, 10)
h5CreateDataset(
```

```

    tmp.file,
    name = "vec/num",
    dims = 10
  )
h5WriteDataset(tmp.file, x1, name = "vec/num")
x <- h5Read(tmp.file, name = "vec/num")
stopifnot(identical(x, x1))

x1 <- rep(2.0, 5)
h5WriteDataset(
  tmp.file,
  x1,
  name = "vec/num",
  idx_list = list(c(1, 3, 5, 7, 9)) # Set each indices to be written
)
x <- h5Read(tmp.file, name = "vec/num")
stopifnot(identical(x, rep(c(2.0, 1.1), 5)))

# matrix #####
x1 <- matrix(1.0, 7, 5)
h5CreateDataset(
  tmp.file,
  name = "mat/num",
  dims = dim(x1)
)
h5WriteDataset(
  tmp.file,
  x1,
  name = "mat/num"
)
x <- h5Read(tmp.file, name = "mat/num")
stopifnot(identical(x, x1))

x1 <- matrix(2.0, 3, 4)
h5WriteDataset(
  tmp.file,
  x1,
  name = "mat/num",
  idx_list = list(2:4, 1:4)
)
x <- h5Read(tmp.file, name = "mat/num")
print(x)

h5WriteDataset(
  tmp.file,
  x1,
  name = "mat/num",
  idx_list = list(1:4, 2:4), # idx_list must match the transposed matrix
  transpose = TRUE
)
x <- h5Read(tmp.file, name = "mat/num")
print(x)

```

h5WriteScalar	<i>Write a scalar into HDF5 file</i>
---------------	--------------------------------------

Description

Low-level helper function to write scalar R data into HDF5 dataset. Data will be written into scalar space instead of array space.

Usage

```
h5WriteScalar(x, name, robj, ...)

## S3 method for class 'H5Group'
h5WriteScalar(x, name, robj, stype = c("utf8", "ascii7"), ...)

## S3 method for class 'H5File'
h5WriteScalar(x, name, robj, ...)

## S3 method for class 'character'
h5WriteScalar(x, name, robj, overwrite = FALSE, ...)
```

Arguments

x	An H5File , H5Group or a path name of HDF5 file.
name	Name of an HDF5 link.
robj	A scalar object.
...	Arguments passed to <code>H5File\$create_dataset()</code> . See link[hdf5r:H5File-class]{H5File} .
stype	'utf8' or 'ascii7'. Passed to h5GuessDtype .
overwrite	Whether or not to overwrite the existing name.

Value

This is an operation function and no return. Any failure should raise an error.

Note

If you want to write robj into array space, you should use [h5WriteDataset](#).

Examples

```
tmp.file <- tempfile(fileext = ".h5")
h5CreateFile(tmp.file)

h5WriteScalar(tmp.file, name = "test/scalar", TRUE)
x <- h5ReadDataset(tmp.file, name = "test/scalar")
stopifnot(x)
```

```
h5WriteScalar(tmp.file, name = "test/scalar", 100.0, overwrite = TRUE)
x <- h5ReadDataset(tmp.file, name = "test/scalar")
stopifnot(identical(x, 100.0))
```

```
h5WriteScalar(tmp.file, name = "test/scalar", "ABC", overwrite = TRUE)
x <- h5Read(tmp.file, name = "test/scalar")
stopifnot(identical(x, "ABC"))
```

```
h5WriteScalar(tmp.file, name = "test/factor", factor("ABC"))
x <- h5ReadDataset(tmp.file, name = "test/factor")
stopifnot(identical(x, factor("ABC")))
```

Index

file.copy, 8

guess_chunks, 12
guess_dtype, 17

H5-attributes, 2
H5-dataset-info, 6
h5AbsLinkName, 7
h5Attr (H5-attributes), 2
h5Attributes (H5-attributes), 2
h5AttrNames (H5-attributes), 2
h5Backup, 8, 21
h5Class, 9
h5Copy, 8, 10
h5CreateDataset, 11
h5CreateFile, 13
h5CreateGroup, 14
H5D, 4, 6, 21, 24, 30
h5Delete, 15
h5DeleteAttr (H5-attributes), 2
h5Dims (H5-dataset-info), 6
h5Exists, 16
H5File, 4, 6, 10, 12–14, 16, 18, 20, 21, 23, 24, 26, 30, 33
H5Group, 4, 6, 12–14, 16, 18, 19, 21, 23, 24, 30, 33
h5GuessDtype, 5, 12, 17, 33
h5List, 18
h5MaxDims (H5-dataset-info), 6
h5Move, 19
h5Open, 20
h5Overwrite, 21
h5Prep, 22, 28
h5Read, 23
h5ReadDataset, 23, 24
H5T, 12, 17
h5TryOpen, 21, 25
h5Write, 26
h5WriteAttr (H5-attributes), 2
h5WriteDataset, 29, 33
h5WriteScalar, 31, 33
hdf5r::guess_dtype, 17
is.H5D (h5Class), 9
is.H5Group (h5Class), 9
numeric, 12