Package 'h5'

August 30, 2017

Type Package

Title Interface to the 'HDF5' Library

Version 0.9.9

Date 2017-08-27

Copyright See COPYRIGHTS file.

Description S4 Interface to the 'HDF5' library supporting fast storage and retrieval of R-objects like vectors, matrices and arrays to binary files in a language independent format. The 'HDF5' format can therefore be used as an alternative to R's save/load mechanism. Since h5 is able to access only subsets of stored data it can also handle data sets which do not fit into memory.

License BSD_2_clause + file LICENSE

BugReports https://github.com/mannau/h5/issues

URL https://github.com/mannau/h5

SystemRequirements libhdf5 (>= 1.8.12) with C++ interface (--enable-cxx=yes) and v18 API enabled

Depends R (>= 3.2)

Imports Rcpp (>= 0.11.5), methods

LinkingTo Rcpp

Suggests testthat, roxygen2, zoo, microbenchmark, knitr, rmarkdown

VignetteBuilder knitr

Collate 'H5Location-Attribute.R' 'Attribute.R' 'CommonFG.R' 'CommonFG-DataSet.R' 'CommonFG-Group.R' 'DataSet.R' 'DataSpace.R' 'Dataset-Extend.R' 'Dataset-Subset.R' 'H5File.R' 'H5Group.R' 'Helpers.R' 'RcppExports.R' 'h5-package.R'

RoxygenNote 6.0.1.9000

NeedsCompilation yes

Author Mario Annau [aut, cre]

Maintainer Mario Annau <mario.annau@gmail.com>

Repository CRAN

Date/Publication 2017-08-30 12:33:09 UTC

R topics documented:

h5-package . . . 2
Attribute . . . 4
CommonFG . . . 5
CommonFG-DataSet . . . 6
CommonFG-Group . . . 8
DataSet . . . 9
DataSet-Extend . . . 10
DataSet-Subset . . . 11
DataSpace . . . 12
H5File . . . 13
H5Group . . . 15
H5Location-Attribute . . . 16

Index **18**

`h5-package` *H5 - Interface to the HDF5 API*

Description

h5 provides an interface to the HDF5 API through S4-classes. HDF5 is a binary data format designed for flexible and efficient I/O, high–volume and complex data. An HDF5 file can be structured in a hierarchical way to store data sets in groups—quite similar to the folder structure in a file system. It supports fast storage and retrieval of R-objects like vectors, matrices and arrays to binary files in a language independent format (currently no data.frames). The package can therefore be used as an alternative to R's save/load mechanism. Since h5 is able to access only subsets of stored data it can also handle data sets which do not fit into memory.

Details

h5 can currently only handle homogeneous data sets consisting of one single data type like `numeric`, `integer`, `character` or `logical`. The creation of metadata through attributes is also supported.

The following objects are supported by **h5** and represented through S4 classes:

H5File holds the pointer to the binary HDF5 file which can include various `DataSets` in a hierarchical structure defined by `H5Groups`.

H5Group can hold various HDF5 objects like `DataSets` and other `H5Groups`.

DataSet stores homogeneous data like vectors, matrices and arrays.

Attribute stores metadata about other HDF5 objects like `H5Group`, `H5File` and `DataSet`.

DataSpace Objects defining selections on specified `DataSets`.

These classes share common functionality through the following base classes:

CommonFG implements common functionality for `H5File` and `H5Group` to create/access sub–`H5Groups` and `DataSets`.

H5Location is the base class of H5File, H5Group and DataSet and implements functions for Attribute creation and retrieval.

The example below shows some typical use cases handling data with HDF5:

1. Create/Open HDF5 File using H5File, specifying file access mode.
2. Create/Open Groups and DataSets either implicitly using subsetting operators or explicitly using the S4–methods like createGroup/openGroup or createDataSet/openDataSet, see also CommonFG, CommonFG-Group and CommonFG-DataSet.
3. Create/Open meta data for HDF5 objects using e.g. h5attr, see also H5Location-Attribute and Attribute.
4. Retrieve data from DataSets either implicitly using subsetting operators or explicitly with readDataSet which requires a DataSpace object to specify the selection area, see also DataSet, DataSet-Subset and DataSpace.
5. Extend DataSet using predefined functions like c for 1-dim. vectors or rbind/cbind 2-dimensional DataSets, see also DataSet-Extend.
6. Close H5File–, H5Group– DataSet– or DataSpace objects using h5close.

Examples

```
# 1. Create/Open file 'test.h5' (mode set to 'a'ppend)
file <- h5file("test.h5", 'a')

# 2. Store character vector in group '/test' and dataset 'testvec'
file["test/testvec"] <- LETTERS[1:9]
# Store integer matrix in group '/test/testmat' and dataset 'testmat'
mat <- matrix(1:9, nrow = 3)
rownames(mat) <- LETTERS[1:3]
colnames(mat) <- c("A", "BE", "BUU")
file["test/testmat/testmat"] <- mat
# Store numeric array in group '/test' and dataset 'testarray'
file["test/testarray"] <- array(as.numeric(1:45), dim = c(3, 3, 5))

# 3. Store rownames and column names of matrix as attributes
# Get created data set as object
dset <- file["test/testmat/testmat"]
# Store rownames in attribute 'dimnames_1'
h5attr(dset, "dimnames_1") <- rownames(mat)
# Store columnnames in attribute 'dimnames_2'
h5attr(dset, "dimnames_2") <- colnames(mat)

# 4. Read first 3 elements of testvec
testvec <- file["test/testvec"]
testvec[1:3]
# Read first 2 rows of testmat
testmat <- file["test/testmat/testmat"]
res <- testmat[1:2, ]
# attach rownames and columnnames
rownames(res) <- attr(testmat, "rownames")[1:2]
colnames(res) <- attr(testmat, "colnames")
```

```
# 5. Extend testvec
testvec <- c(testvec, LETTERS[10:26])
# Retrieve entire testvec
testvec[]

# 6. Close open handles
h5close(testvec)
h5close(testmat)
h5close(file)
```

`Attribute` *The Attribute Class*

Description

An HDF5 `Attribute` object is used to store meta data about a primary data object like `DataSet`, `H5Group` or `H5File`. The functions described in this section show a low–level interface to read and write attributes. See `H5Location` for the easier-to-use `h5attr` convenience functions.

Usage

```
writeAttribute(.Object, data = GetDimensions(data), transpose = TRUE)

## S4 method for signature 'Attribute'
writeAttribute(.Object, data = GetDimensions(data),
  transpose = TRUE)

readAttribute(.Object)

## S4 method for signature 'Attribute'
readAttribute(.Object)
```

Arguments

`.Object`	Attribute; S4 object of class `Attribute`.
`data`	object; Object to be stored in HDF5 file, can be either of type vector, matrix or array.
`transpose`	logical; Determine if data object should be transposed.

Details

`Attributes` are assumed to be like small `DataSet`s since they can store the same data types in homogeneous objects like vectors, matrices and arrays. However, there are two important differences compared to `DataSet` objects:

1. Attributes do not support compression or chunking.
2. Subsetting is not possible for attribute data.

References

http://www.hdfgroup.org/HDF5/doc/H5.intro.html#Intro-OAttributes

See Also

H5Location-Attribute

CommonFG *The CommonFG Class*

Description

CommonFG is the base class of H5File and H5Group and represents common functionality of these two classes. The CommonFG base class supports various subsetting operators to easily access and manipulate H5Group and DataSet objects (see also CommonFG-Group and CommonFG-DataSet).

Usage

```
h5close(.Object)

## S4 method for signature 'CommonFG,character,ANY'
x[i, j, ..., drop = TRUE]

## S4 replacement method for signature 'CommonFG,character,ANY'
x[i, j, ...] <- value

h5unlink(.Object, path)

## S4 method for signature 'CommonFG,character'
h5unlink(.Object, path)
```

Arguments

.Object	CommonFG; S4 object of class CommonFG;
x	CommonFG; object to be subsetted
i	character; Name of H5Group
j	character; Name of DataSet
...	Additional arguments passed to createDataSet; only relevant for assignment operator.
drop	logical; specify if class of result set should be dropped (not implemented yet).
value	vector/matrix/array; Value to be assigend to dataset
path	character; Path to be deleted, either specifying group or dataset.

Details

Subsetting operators on CommonFG objects represent a convenient way to create/access H5Group and DataSet objects. Currently, only character arguments are supported whereas the first argument specifies the group to be created/accesses and the second the dataset name.

Groups can be created/accessed by simply using one character parameter, e.g. `group <- obj["groupname"]`.

DataSets can be either accessed by using `dset <- obj["groupname", "datasetname"]` if existing or initialized by using `obj["groupname", "datasetname"] <- value`.

All created objects e.g. `group` or `dset` should be closed in the end using `h5close`.

See Also

CommonFG-Group CommonFG-DataSet H5Location-Attribute

Examples

```
file <- h5file("test.h5")
# Create new DataSet 'testset' in H5Group 'testgroup'
file["testgroup/testset"] <- matrix(1:9, nrow = 3)
# Create new DataSet 'testset2' in file root
file["testset2"] <- 1:10
# Retrieve H5Group 'testgroup'
group <- file["testgroup"]
# Retrieve DataSet 'testset'
dset <- group["testset"]
h5close(dset)
h5close(group)
h5close(file)
file.remove("test.h5")
```

CommonFG-DataSet *Functions to Create/Open DataSets in CommonFG objects Although DataSet objects can implicitly be created using subsetting operators (see CommonFG)* **h5** *implements more explicit functions (used by subsetting operators under the hood) to create and open DataSets.*

Description

Functions to Create/Open DataSets in CommonFG objects

Although DataSet objects can implicitly be created using subsetting operators (see CommonFG) **h5** implements more explicit functions (used by subsetting operators under the hood) to create and open DataSets.

Usage

```
createDataSet(.Object, datasetname, data, type, dimensions, chunksize = -1,
  maxdimensions = NA_integer_, compression = 4L, size = -1)


  ## S4 method for signature
## 'CommonFG,character,missing,character,ANY,ANY,ANY,ANY,ANY'
createDataSet(.Object,
  datasetname, type, dimensions, chunksize, maxdimensions, compression, size)


  ## S4 method for signature
## 'CommonFG,character,ANY,missing,missing,ANY,ANY,ANY,numeric'
createDataSet(.Object,
  datasetname, data, chunksize, maxdimensions, compression, size)


  ## S4 method for signature
## 'CommonFG,character,ANY,missing,missing,ANY,ANY,ANY,missing'
createDataSet(.Object,
  datasetname, data, chunksize, maxdimensions, compression)

openDataSet(.Object, datasetname, type)

## S4 method for signature 'CommonFG,character'
openDataSet(.Object, datasetname, type)

list.datasets(.Object, path = "/", full.names = TRUE, recursive = TRUE,
  follow.links = FALSE)

## S4 method for signature 'CommonFG'
list.datasets(.Object, path = "/", full.names = TRUE,
  recursive = TRUE, follow.links = FALSE)

existsDataSet(.Object, datasetname)

## S4 method for signature 'CommonFG,character'
existsDataSet(.Object, datasetname)
```

Arguments

.Object	CommonFG; S4 object of class CommonFG;
datasetname	character; HDF5 DataSet name to be used.
data	object; Object to be stored in HDF5 file, can be either of type vector, matrix or array.
type	character; Character specifying data type, can be either one of:
	double Double precision floating–point number.

	integer 32–Bit integer.
	logical Boolean, which is mapped to 1/0 integer values.
	character Variable–length character strings.
`dimensions`	integer; Dimensions of dataset to be created.
`chunksize`	integer; Chunksize to be used for dataset. If set to `NA`, chunking is disabled for dataset; `maxdimensionions` and `compression` have no effect and extensions of DataSet (e.g. through `cbind`, `rbind`) are not possible.
`maxdimensions`	integer; Maximum dimensions used for dataset, `NA` sets maxdimensions to 'unlimited'.
`compression`	integer; Default GZIP compression level to be used, from 0 (no compression) to 9 (maximum compression), defaults to 4.
`size`	numeric; Character length for fixed-length string data types. Default value of -1 creates variable-length strings.
`path`	character; Relative path to .Object.
`full.names`	character; Specify if absolute DataSet path names should be returned.
`recursive`	logical; Specify DatSets should be retrieved recursively from .Object.
`follow.links`	logical; Specify if symbolic links should be followed (only applies if recursive = TRUE).

`CommonFG-Group` *Functions to Create/Open Groups in* `CommonFG` *objects*

Description

Although `H5Group` objects can implicitly be created using subsetting operators (see `CommonFG`) **h5** implements more explicit functions (used by subsetting operators under the hood) to create and open `H5Group`s.

Usage

```
createGroup(.Object, groupname)

## S4 method for signature 'CommonFG,character'
createGroup(.Object, groupname)

openGroup(.Object, groupname)

## S4 method for signature 'CommonFG,character'
openGroup(.Object, groupname)

existsGroup(.Object, groupname)

## S4 method for signature 'CommonFG,character'
existsGroup(.Object, groupname)
```

```
getH5Group(.Object, groupname)

## S4 method for signature 'CommonFG,character'
getH5Group(.Object, groupname)

list.groups(.Object, path = "/", full.names = TRUE, recursive = TRUE)

## S4 method for signature 'CommonFG'
list.groups(.Object, path = "/", full.names = TRUE,
  recursive = TRUE)
```

Arguments

.Object	CommonFG; S4 object of class CommonFG;
groupname	character; HDF5 Group name to be used.
path	character; Relative path to .Object.
full.names	character; Specify if absolute DataSet path names should be returned.
recursive	logical; Specify DatSets should be retrieved recursively from .Object.

DataSet *The DataSet Class*

Description

DataSets are used to store data objects in the HDF5 tree. Data objects contain homogeneous data of one type like numeric, integer or character and can be subsetted, extended and enriched with Attributes (see DataSet-Subset, DataSet-Extend and H5Location-Attribute). Although subsetting operators provide a convenient way to handle DataSet objects the S4 methods described in this section are used under the hood and give more control. Especially for big DataSets it can be advantageous to use these methods with DataSpace objects including hyperslab selections.

Usage

```
writeDataSet(.Object, data, dspace = selectDataSpace(.Object, rep(NA_integer_,
  length(.Object@dim)), GetDimensions(data)), transpose = TRUE)

## S4 method for signature 'DataSet'
writeDataSet(.Object, data,
  dspace = selectDataSpace(.Object, rep(NA_integer_, length(.Object@dim)),
  GetDimensions(data)), transpose = TRUE)

readDataSet(.Object, dspace = selectDataSpace(.Object))

## S4 method for signature 'DataSet'
readDataSet(.Object, dspace = selectDataSpace(.Object))
```

```
## S4 method for signature 'DataSet'
h5close(.Object)
```

Arguments

.Object	DataSet; S4 object of class DataSet;
data	object; Object to be stored in HDF5 file, can be either of type vector, matrix or array.
dspace	DataSpace; Data space object used for data selection.
transpose	logical; Determine if data object (if is array) should be transposed.
...	additional arguments passed to c.

References

https://www.hdfgroup.org/HDF5/doc/H5.intro.html#Intro-ODatasets

---

DataSet-Extend *Functions to Extend a DataSet*

Description

DataSets can be extended with R–objects (e.g. vectors, matrices, arrays) if the following conditions are met:

1. Datatype of DataSet and R-object are compatible.
2. Dimensions of DataSet and R-object match (no recycling).
3. DataSet does not exceed maximum dimensions as specified at creation.

Usage

```
extendDataSet(.Object, dims)

## S4 method for signature 'DataSet,numeric'
extendDataSet(.Object, dims)

## S4 method for signature 'DataSet'
c(x, ..., recursive = FALSE)
```

Arguments

.Object, x	DataSet; S4 object of class DataSet;
dims	numeric; Dimensions of DataSet.
...	additional arguments passed to c.
recursive	logical; Argument passed to c.

Details

Known base functions have been overloaded to extend vectors (`c`) and matrices (`rbind`, `cbind`). Also the lower–level S4–method `extendDataSet` can be used to extend existing `DataSet` objects.

`DataSet-Subset` *Operators to Subset DataSet Objects*

Description

`DataSet` objects can be subsetted to retrieve or write sub–regions as defined by a `DataSpace`. The subsetting command should be the same to base R subsetting of vectors, matrices or arrays. Also missing subsetting parameters are supported.

Usage

```
## S4 method for signature 'DataSet,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'DataSet,missing,missing'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'DataSet,numeric,missing'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'DataSet,missing,numeric'
x[i, j, ..., drop = TRUE]

## S4 replacement method for signature 'DataSet,missing,missing'
x[i, j, ...] <- value

## S4 replacement method for signature 'DataSet,numeric,missing'
x[i, j, ...] <- value

## S4 replacement method for signature 'DataSet,missing,numeric'
x[i, j, ...] <- value

## S4 replacement method for signature 'DataSet,ANY,ANY'
x[i, j, ...] <- value
```

Arguments

`x`	DataSet; S4 object of class `DataSet`;
`i`	integer; row index
`j`	integer; column index
`...`	additional arguments for subsetting
`drop`	logical; specify if
`value`	object; Value to be assigned to dataset

Details

For subset–write operations it has to be noted that no recycling is done by the function and therefore the written object needs to have the same dimensions as the subsetted region.

Note

Subsetting DataSets can become slow for many selection points since element–wise selection is used, see also DataSpace.

Examples

```
# Write submatrix to sub-region of DataSet
testmat_n <- matrix(as.integer(1:90), ncol = 9)
file <- h5file("test.h5", "a")
file["testgroup/testmat_n2"] <- testmat_n
submat <- matrix(-1L:-9L, nrow = 3)
dset2 <- file["testgroup/testmat_n2"]
dset2[c(1, 3, 5), c(1, 3, 5)] <- submat
h5close(dset2)
h5close(file)
file.remove("test.h5")
```

DataSpace *The DataSpace Class*

Description

The DataSpace class used to select data elements from DataSet objects. DataSpace objects can be generated using the S4–method selectDataSpace from DataSets objects in 3 ways:

1. Without any parameters: Select all elements in DataSet.
2. With offset and count: Select contiguous sub–region (hyperslab) from DataSet.
3. With parameter elem: Select sub-elements specified by matrix holding selected indices. This selection type can become slow for many data points.

Usage

```
## S4 method for signature 'DataSpace'
h5close(.Object)

selectDataSpace(.Object, offset = rep(NA_integer_, length(.Object@dim)),
  count = rep(NA_integer_, length(.Object@dim)), elem)

## S4 method for signature 'DataSet,missing,missing,missing'
selectDataSpace(.Object)

## S4 method for signature 'DataSet,ANY,ANY,missing'
```

```
selectDataSpace(.Object, offset, count)

## S4 method for signature 'DataSet,missing,missing,matrix'
selectDataSpace(.Object, elem)
```

Arguments

.Object	DataSet; S4 object of class DataSet (DataSpace for h5close).
offset	numeric; Offset to be selected from Hyperslab.
count	numeric; Count to be selected from Hyperslab.
elem	matrix; Matrix specifying element selection coordinates. Columns specify rank, rows specify different points.
...	additional arguments passed to c.

H5File *HDF5 File Objects*

Description

H5File objects are are the main entry point to access HDF5 data from binary files. The *H5File* S4 class directly maps https://www.hdfgroup.org/HDF5/doc/cpplus_RM/class_h5_1_1_h5_file.html objects from the C++ API to R. Through the implemented class hierarchy it shares common functionality with *H5Group*.

Usage

```
h5file(name, mode = "a")

H5File(name, mode = "a")

h5flush(.Object)

## S4 method for signature 'H5File'
h5flush(.Object)

## S4 method for signature 'H5File'
h5close(.Object)

is.h5file(name)
```

Arguments

name	character; File path pointing to H5File.
mode	mode used for file The following modes are supported by h5file:
	r Read only, file must exist.

r+ Read/write, file must exist.

w Create file, truncate if exists.

w- Create file, fail if exists.

a Read/write if exists, create otherwise (default).

`.Object` H5File; S4 object of class `H5File`;

Details

HDF5 files can be opened or generated using the `h5file()` function and a specified file access mode. `h5file()` returns a `H5File` object which can be used to access `H5Group`s and `DataSet`s using subsetting parameters or according class methods.

HDF5 files which have been created or opened through `h5file()` need to be closed afterwards using `h5close()`.

`h5flush()` can be used to flush unwritten data to an HDF5 file.

HDF5 Files can contain the following objects:

Groups Similar to a file system folder, used to organize HDF5 objects in a hierarchical way.

Datasets Objects to store actual data.

Attributes Meta data objects to store extra informatino about Files, Groups and Datasets.

Reading and Writing Files

HDF5 files can be created and accessed using `h5file()`: `file <- h5file(name = "test.h5", mode = "a")`

The following access-modes are defined:

Mode	Description
a	Read/write if exists, create otherwise (default).
r	Read only, file must exist.
r+	Read/write, file must exist.
w	Create file, truncate if exists.
w-	Create file, fail if exists.

Show File Contents

HDF5 objects stored in a file are shown with the following symbols:

Mode	Description
+	HDF5 Group.
D	HDF5 Dataset.
A	HDF5 Attribute.

Extract/List File Contents

The following functions are defined to extract HDF5 file contents:

list.groups List HDF5 groups in file.

list.datasets List HDF5 datasets in file.

list.attributes List Attributes of HDF5 object (file, group or dataset).

See Also

CommonFG CommonFG-Group CommonFG-DataSet H5Location-Attribute

Examples

```
# The following examples generates a HDF5 file with the different HDF5
# Objects and shows its contents:
file <- h5file(name = "test1.h5", mode = "a")
file["testdataset"] <- 1:10
h5attr(file, "testattrib") <- LETTERS[1:10]
file["testgroup/testdataset2"] <- 1:10
file
# Close file and delete
h5close(file)
if(file.exists("test.h5")) file.remove("test.h5")

# The following example shows hdf5 file contents and how to use them to iterate over HDF5 elements:
file <- h5file(name = "test2.h5", mode = "a")
file["testgroup1/testset1"] <- 1:10
file["testgroup2/testset2"] <- 11:20
file["testgroup3/testset3"] <- 21:30

# Extract first 3 elements from each dataset and combine result to matrix
sapply(list.datasets(file, recursive = TRUE), function(x) file[x][1:3])
# Add new dataset to each group in HDF5 file
for(g in list.groups(file)) {
  file[paste(g, "testsetx", collapse = "/")] <- 1:10
}
list.datasets(file, recursive = TRUE)
# Close file
h5close(file)
```

H5Group *The H5Group Class*

Description

HDF5 Groups are represented by the H5Group class and are the building blocks of the hierarchical organization of an H5File. They form containers for HDF5 objects and are therefore similar to file system folders.

Usage

```
## S4 method for signature 'H5Group'
h5close(.Object)
```

Arguments

.Object	H5Group; S4 object of class H5Group;

Details

In addition to Group–specific capabilities listed below H5Group shares common functionality with H5File through the CommonFG base class.

References

https://www.hdfgroup.org/HDF5/doc/H5.intro.html#Intro-OGroups

---

H5Location-Attribute *Read and Create Attributes for H5Location Objects*

Description

H5Location is the base class of H5File, H5Group and DataSet and implements common methods to create and access attributes for inherited classes.

Usage

```
createAttribute(.Object, attributename, data, size = -1)

## S4 method for signature 'H5Location,character'
createAttribute(.Object, attributename, data,
  size = -1)

openAttribute(.Object, attributename)

## S4 method for signature 'H5Location,character'
openAttribute(.Object, attributename)

h5attr(.Object, attributename)

## S4 method for signature 'H5Location,character'
h5attr(.Object, attributename)

h5attr(.Object, attributename, ...) <- value

## S4 replacement method for signature 'H5Location,character'
h5attr(.Object, attributename, ...) <- value

list.attributes(.Object)

## S4 method for signature 'H5Location'
list.attributes(.Object)
```

Arguments

.Object	H5Location; S4 object of class H5Location;
attributename	character; Name of attribute to be read/created.
data	object; Data object to be used for attribute creation, can be either of type vector, matrix or array.
size	numeric; Character length for fixed-length string data types. Default value of -1 creates variable-length strings.
...	Additional parameters passed to createAttribute.
value	object; Object to be stored in HDF5 Attribute, can be either of type vector, matrix or array.

See Also

Attribute H5File H5Group DataSet

Examples

```
# Write Attributes for H5File, H5Group and DataSet
file <- h5file("test.h5")
h5attr(file, "fileattrib") <- 1:10
group <- file["testgroup"]
h5attr(group, "groupattrib") <- matrix(1:9, nrow = 3)
h5attr(group, "groupattrib")
group["testdataset"] <- 1:10
dset <- group["testdataset"]
h5attr(dset, "dsetattrib") <- LETTERS[1:10]
h5close(dset)
h5close(group)
h5close(file)
file.remove("test.h5")
```

Index

[,CommonFG,character,ANY-method (CommonFG), 5
[,DataSet,ANY,ANY-method (DataSet-Subset), 11
[,DataSet,missing,missing-method (DataSet-Subset), 11
[,DataSet,missing,numeric-method (DataSet-Subset), 11
[,DataSet,numeric,missing-method (DataSet-Subset), 11
[<-,CommonFG,character,ANY-method (CommonFG), 5
[<-,DataSet,ANY,ANY-method (DataSet-Subset), 11
[<-,DataSet,missing,missing-method (DataSet-Subset), 11
[<-,DataSet,missing,numeric-method (DataSet-Subset), 11
[<-,DataSet,numeric,missing-method (DataSet-Subset), 11

Attribute, *2, 3*, 4, *17*
Attribute-class (Attribute), 4

c, *10*, *13*
c,DataSet-method (DataSet-Extend), 10
CommonFG, *2, 3*, 5, *6*, *8*, *15*
CommonFG-class (CommonFG), 5
CommonFG-DataSet, *3*, 6
CommonFG-Group, *3*, 8
createAttribute, *17*
createAttribute (H5Location-Attribute), 16
createAttribute,H5Location,character-method (H5Location-Attribute), 16
createDataSet, *3*, *5*
createDataSet (CommonFG-DataSet), 6
createDataSet,CommonFG,character,ANY,missing,missing,ANY,ANY-method (CommonFG-DataSet), 6
createDataSet,CommonFG,character,ANY,missing,missing,ANY,A (CommonFG-DataSet), 6
createDataSet,CommonFG,character,missing,character,ANY,ANY (CommonFG-DataSet), 6
createGroup, *3*
createGroup (CommonFG-Group), 8
createGroup,CommonFG,character-method (CommonFG-Group), 8

DataSet, *2–6*, 9, *10–12*, *14*, *16, 17*
DataSet-class (DataSet), 9
DataSet-Extend, *3*, 10
DataSet-Subset, *3*, 11
DataSpace, *2, 3*, *9*, *11, 12*, 12
DataSpace-class (DataSpace), 12

existsDataSet (CommonFG-DataSet), 6
existsDataSet,CommonFG,character-method (CommonFG-DataSet), 6
existsGroup (CommonFG-Group), 8
existsGroup,CommonFG,character-method (CommonFG-Group), 8
extendDataSet (DataSet-Extend), 10
extendDataSet,DataSet,numeric-method (DataSet-Extend), 10

getH5Group (CommonFG-Group), 8
getH5Group,CommonFG,character-method (CommonFG-Group), 8

h5-package, 2
h5attr, *3, 4*
h5attr (H5Location-Attribute), 16
h5attr,H5Location,character-method (H5Location-Attribute), 16
h5attr<- (H5Location-Attribute), 16
h5attr<-,H5Location,character-method (H5Location-Attribute), 16
h5close, *3*
h5close,ANY,ANY,ANY,missing-method (CommonFG), 5

h5close,DataSet-method (DataSet), 9
h5close,DataSpace-method (DataSpace), 12
h5close,H5File-method (H5File), 13
h5close,H5Group-method (H5Group), 15
H5File, *2–5*, 13, *16, 17*
h5file (H5File), 13
H5File-class (H5File), 13
h5flush (H5File), 13
h5flush,H5File-method (H5File), 13
H5Group, *2*, *4–6*, *8*, *14*, 15, *16, 17*
H5Group-class (H5Group), 15
H5Location, *3, 4*
H5Location (H5Location-Attribute), 16
H5Location-Attribute, *3*, 16
H5Location-class (H5Location-Attribute), 16
h5unlink (CommonFG), 5
h5unlink,CommonFG,character-method (CommonFG), 5

is.h5file (H5File), 13

list.attributes (H5Location-Attribute), 16
list.attributes,H5Location-method (H5Location-Attribute), 16
list.datasets (CommonFG-DataSet), 6
list.datasets,CommonFG-method (CommonFG-DataSet), 6
list.groups (CommonFG-Group), 8
list.groups,CommonFG-method (CommonFG-Group), 8

openAttribute (H5Location-Attribute), 16
openAttribute,H5Location,character-method (H5Location-Attribute), 16
openDataSet, *3*
openDataSet (CommonFG-DataSet), 6
openDataSet,CommonFG,character-method (CommonFG-DataSet), 6
openGroup, *3*
openGroup (CommonFG-Group), 8
openGroup,CommonFG,character-method (CommonFG-Group), 8

readAttribute (Attribute), 4
readAttribute,Attribute-method (Attribute), 4
readDataSet, *3*
readDataSet (DataSet), 9
readDataSet,DataSet-method (DataSet), 9

selectDataSpace (DataSpace), 12
selectDataSpace,DataSet,ANY,ANY,missing-method (DataSpace), 12
selectDataSpace,DataSet,missing,missing,matrix-method (DataSpace), 12
selectDataSpace,DataSet,missing,missing,missing-method (DataSpace), 12

writeAttribute (Attribute), 4
writeAttribute,Attribute-method (Attribute), 4
writeDataSet (DataSet), 9
writeDataSet,DataSet-method (DataSet), 9