

# "h2o"

September 24, 2015

## R topics documented:

h2o-package . . . . .	5
aaa . . . . .	6
apply,H2OFrame-method . . . . .	6
as.character,H2OFrame-method . . . . .	7
as.data.frame,H2OFrame . . . . .	7
as.environment,H2OFrame-method . . . . .	8
as.factor,H2OFrame-method . . . . .	8
as.h2o . . . . .	9
as.numeric,H2OFrame-method . . . . .	9
ASTNode-class . . . . .	10
colnames<-,H2OFrame,H2OFrame-method . . . . .	10
h2o.aic . . . . .	11
h2o.anomaly . . . . .	12
h2o.anyFactor . . . . .	13
h2o.assign . . . . .	14
h2o.auc . . . . .	14
h2o.betweeness . . . . .	15
h2o.biases . . . . .	16
h2o.cbind . . . . .	16
h2o.centers . . . . .	17
h2o.centersSTD . . . . .	17
h2o.centroid_stats . . . . .	18
h2o.clearLog . . . . .	18
h2o.clusterInfo . . . . .	19
h2o.clusterIsUp . . . . .	19
h2o.clusterStatus . . . . .	20
h2o.cluster_sizes . . . . .	20
h2o.coef . . . . .	21
h2o.coef_norm . . . . .	21
h2o.confusionMatrix . . . . .	22
h2o.createFrame . . . . .	23
h2o.cummax . . . . .	24
h2o.cummin . . . . .	25

h2o.cumprod	25
h2o.cumsum	26
h2o.cut	26
h2o.day	27
h2o.dayOfWeek	28
h2o.dct	29
h2o.ddply	30
h2o.deepfeatures	31
h2o.deeplearning	32
h2o.dim	36
h2o.downloadAllLogs	37
h2o.downloadCSV	37
h2o.download_pojo	38
h2o.exportFile	39
h2o.exportHDFS	40
h2o.filterNACols	40
h2o.gbm	41
h2o.getConnection	43
h2o.getFrame	43
h2o.getGrid	44
h2o.getModel	44
h2o.getTimezone	45
h2o.giniCoef	46
h2o.glm	47
h2o.glm	50
h2o.grid	52
h2o.group_by	53
h2o.gsub	54
h2o.head	54
h2o.hist	55
h2o.hit_ratio_table	56
h2o.hour	56
h2o.ifelse	57
h2o.importFile	58
h2o.impute	59
h2o.init	61
h2o.insertMissingValues	63
h2o.interaction	64
h2o.is_client	65
h2o.killMinus3	66
h2o.kmeans	66
h2o.length	67
h2o.levels	68
h2o.listTimezones	69
h2o.loadModel	69
h2o.logAndEcho	70
h2o.logloss	71
h2o.ls	71

h2o.makeGLMModel . . . . .	72
h2o.match . . . . .	72
h2o.mean . . . . .	73
h2o.mean_residual_deviance . . . . .	74
h2o.median . . . . .	75
h2o.merge . . . . .	75
h2o.metric . . . . .	76
h2o.month . . . . .	78
h2o.mse . . . . .	78
h2o.naiveBayes . . . . .	79
h2o.networkTest . . . . .	81
h2o.nlevels . . . . .	81
h2o.nrow . . . . .	82
h2o.null_deviance . . . . .	82
h2o.null_dof . . . . .	83
h2o.num_iterations . . . . .	84
h2o.openLog . . . . .	84
h2o.parseRaw . . . . .	85
h2o.parseSetup . . . . .	86
h2o.performance . . . . .	86
h2o.prcomp . . . . .	87
h2o.quantile . . . . .	88
h2o.r2 . . . . .	89
h2o.randomForest . . . . .	90
h2o.rbind . . . . .	92
h2o.removeAll . . . . .	93
h2o.removeVecs . . . . .	93
h2o.rep_len . . . . .	94
h2o.residual_deviance . . . . .	94
h2o.residual_dof . . . . .	95
h2o.rm . . . . .	96
h2o.runif . . . . .	96
h2o.saveModel . . . . .	97
h2o.scale . . . . .	98
h2o.scoreHistory . . . . .	99
h2o.sd . . . . .	99
h2o.sdev . . . . .	100
h2o.setLevel . . . . .	100
h2o.setLevels . . . . .	101
h2o.setTimezone . . . . .	101
h2o.shim . . . . .	102
h2o.shutdown . . . . .	102
h2o.splitFrame . . . . .	103
h2o.startGLMJob . . . . .	104
h2o.startLogging . . . . .	106
h2o.stopLogging . . . . .	106
h2o.strsplit . . . . .	107
h2o.sub . . . . .	107

h2o.subset . . . . .	108
h2o.summary . . . . .	108
h2o.svd . . . . .	109
h2o.table . . . . .	110
h2o.tabulate . . . . .	111
h2o.tolower . . . . .	112
h2o.totss . . . . .	112
h2o.tot_withinss . . . . .	113
h2o.toupper . . . . .	113
h2o.transform . . . . .	114
h2o.trim . . . . .	115
h2o.var . . . . .	115
h2o.varimp . . . . .	116
h2o.week . . . . .	116
h2o.weights . . . . .	117
h2o.which . . . . .	117
h2o.withinss . . . . .	118
h2o.year . . . . .	118
H2OClusteringModel-class . . . . .	119
H2OConnection-class . . . . .	120
H2OFrame-class . . . . .	121
H2OFrame-Extract . . . . .	121
H2OGrid-class . . . . .	122
H2OModel-class . . . . .	123
H2OModelFuture-class . . . . .	124
H2OModelMetrics-class . . . . .	124
H2OObject-class . . . . .	125
H2ORawData-class . . . . .	125
H2OS4groupGeneric . . . . .	126
H2OW2V-class . . . . .	127
is.factor,H2OFrame-method . . . . .	128
is.numeric,H2OFrame-method . . . . .	128
ModelAccessors . . . . .	129
na.omit,H2OFrame-method . . . . .	130
Node-class . . . . .	130
predict.H2OModel . . . . .	131
print.H2OTable . . . . .	131
range,H2OFrame-method . . . . .	132
sapply,H2OFrame-method . . . . .	132
str.H2OFrame . . . . .	133
summary,H2OGrid-method . . . . .	133
summary,H2OModel-method . . . . .	134
zzz . . . . .	134

---

h2o-package*H2O R Interface*

---

## Description

This is a package for running H2O via its REST API from within R. To communicate with a H2O instance, the version of the R package must match the version of H2O. When connecting to a new H2O cluster, it is necessary to re-run the initializer.

## Details

Package: h2o  
Type: Package  
Version: 3.2.0.5  
Branch: rel-slater  
Date: Thu Sep 24 17:27:46 PDT 2015  
License: Apache License (== 2.0)  
Depends: R (>= 2.13.0), RCurl, jsonlite, statmod, tools, methods, utils

This package allows the user to run basic H2O commands using R commands. In order to use it, you must first have H2O running. To run H2O on your local machine, call `h2o.init` without any arguments, and H2O will be automatically launched at `localhost:54321`, where the IP is "127.0.0.1" and the port is 54321. If H2O is running on a cluster, you must provide the IP and port of the remote machine as arguments to the `h2o.init()` call.

H2O supports a number of standard statistical models, such as GLM, K-means, and Random Forest. For example, to run GLM, call `h2o.glm` with the H2O parsed data and parameters (response variable, error distribution, etc...) as arguments. (The operation will be done on the server associated with the data object where H2O is running, not within the R environment).

Note that no actual data is stored in the R workspace; and no actual work is carried out by R. R only saves the named objects, which uniquely identify the data set, model, etc on the server. When the user makes a request, R queries the server via the REST API, which returns a JSON file with the relevant information that R then displays in the console.

If you are using an older version of H2O, use the following porting guide to update your scripts:

[Porting Scripts](#)

## Author(s)

Anqi Fu, Tom Kraljevic and Petr Maj, with contributions from the 0xdata team

Maintainer: Tom Kraljevic <tomk@0xdata.com>

## References

- [0xdata Homepage](#)
- [H2O Documentation](#)

- [H2O on Github](#)

---

aaa	<i>Starting H2O For examples</i>
-----	----------------------------------

---

**Description**

Starting H2O For examples

**Examples**

```
h2o.init()
```

---

apply,H2OFrame-method	<i>Apply on H2O Datasets</i>
-----------------------	------------------------------

---

**Description**

Method for apply on [H2OFrame](#) objects. Closures are not supported: an error message stating this fact will stop execution.

**Usage**

```
## S4 method for signature H2OFrame
apply(X, MARGIN, FUN, ...)
```

**Arguments**

X	an <a href="#">H2OFrame</a> object on which apply will operate.
MARGIN	the vector on which the function will be applied over, either 1 for rows or 2 for columns.
FUN	the function to be applied.
...	optional arguments to FUN.

**Value**

Produces a new [H2OFrame](#) of the output of the applied function. The output is stored in H2O so that it can be used in subsequent H2O processes.

**See Also**

[apply](#) for the base generic

**Examples**

```

localH2O = h2o.init()
irisPath = system.file("extdata", "iris.csv", package="h2o")
iris.hex = h2o.importFile(localH2O, path = irisPath, destination_frame = "iris.hex")
summary(apply(iris.hex, 1, sum))

```

---

as.character,H2OFrame-method

*Convert H2O Data to Characters*


---

**Description**

Converts an H2O column into character columns.

**Usage**

```

## S4 method for signature H2OFrame
as.character(x)

```

**Arguments**

x	a column from an <a href="#">H2OFrame</a> data set. localH2O <- h2o.init() iris.hex <- as.h2o(iris) iris.hex[,5] <- as.character(iris.hex[,5])
---	---

---

as.data.frame.H2OFrame

*Converts a Parsed H2O data into a Data Frame*


---

**Description**

Downloads the H2O data and then scans it in to an R data frame.

**Usage**

```

## S3 method for class H2OFrame
as.data.frame(x, ...)

```

**Arguments**

x	An <a href="#">H2OFrame</a> object.
...	Further arguments to be passed down from other methods.

**Examples**

```

localH2O <- h2o.init()
prosPath <- system.file("extdata", "prostate.csv", package="h2o")
prostate.hex <- h2o.uploadFile(localH2O, path = prosPath)
as.data.frame(prostate.hex)

```

---

as.environment,H2OFrame-method

*Convert H2O Data to an R Environment*

---

**Description**

Converts an [H2OFrame](#) to an environment.

**Usage**

```

## S4 method for signature H2OFrame
as.environment(x)

```

**Arguments**

x                      an [H2OFrame](#) class object.

**Value**

Returns an R environment object based on the [H2OFrame](#). localH2O <- h2o.init() prosPath <- system.file("extdata", "prostate.csv", package="h2o") prostate.hex <- h2o.uploadFile(localH2O, path = prosPath) names(as.environment) aa <- as.environment(prostate.hex) ls(aa)

---

as.factor,H2OFrame-method

*Convert H2O Data to Factors*

---

**Description**

Convert a column into a factor column.

**Usage**

```

## S4 method for signature H2OFrame
as.factor(x)

```

**Arguments**

x                      a column from an [H2OFrame](#) data set.



**See Also**

[is.factor.](#)

**Examples**

```
localH2O <- h2o.init()
prosPath <- system.file("extdata", "prostate.csv", package="h2o")
prostate.hex <- h2o.uploadFile(localH2O, path = prosPath)
prostate.hex[,2] <- as.factor(prostate.hex[,2])
summary(prostate.hex)
```

---

as.h2o	<i>R data.frame -&gt; H2OFrame</i>
--------	------------------------------------

---

**Description**

Import a local R data frame to the H2O cloud.

**Usage**

```
as.h2o(object, conn = h2o.getConnection(), destination_frame = "")
```

**Arguments**

object	An R data frame.
conn	An <a href="#">H2OConnection</a> object containing the IP address and port number of the H2O server.
destination_frame	A string with the desired name for the H2O Frame.

---

as.numeric,H2OFrame-method	<i>Convert H2O Data to Numeric</i>
----------------------------	------------------------------------

---

**Description**

Converts an H2O column into a numeric value column.

**Usage**

```
## S4 method for signature H2OFrame
as.numeric(x)
```

Arguments

x                    a column from an [H2OFrame](#) data set. localH2O <- h2o.init() prosPath <- system.file("extdata", "prostate.csv", package="h2o") prostate.hex <- h2o.uploadFile(localH2O, path = prosPath) prostate.hex[,2] <- as.factor(prostate.hex[,2]) prostate.hex[,2] <- as.numeric(prostat.hex[,2])

---

ASTNode-class	<i>The ASTNode class.</i>
---------------	---------------------------

---

Description

This class represents a node in the abstract syntax tree. An ASTNode has a root. The root has children that either point to another ASTNode, or to a leaf node, which may be of type ASTNumeric or ASTFrame.

Usage

```
## S4 method for signature ASTNode
show(object)
```

Arguments

object            An ASTNode class object.

Slots

root    Object of type Node  
children    Object of type list

---

colnames<- ,H2OFrame,H2OFrame-method	<i>Returns Column Names for a Parsed H2O Data Object.</i>
--------------------------------------	---

---

Description

Returns column names for an [H2OFrame](#) object.

**Usage**

```
## S4 replacement method for signature H2OFrame,H2OFrame
colnames(x) <- value

## S4 replacement method for signature H2OFrame,character
colnames(x) <- value

## S4 method for signature H2OFrame
names(x)

## S4 replacement method for signature H2OFrame
names(x) <- value

h2o.colnames(x)

h2o.names(x)

## S4 method for signature H2OFrame
colnames(x)
```

**Arguments**

x                      An [H2OFrame](#) object.

value                  a character string to rename columns.

**See Also**

[colnames](#) for the base R method.

**Examples**

```
library(h2o)
localH2O <- h2o.init()
irisPath <- system.file("extdata", "iris.csv", package="h2o")
iris.hex <- h2o.uploadFile(localH2O, path = irisPath)
summary(iris.hex)
colnames(iris.hex)
```

---

h2o.aic

*Retrieve the AIC. If "train", "valid", and "xval" parameters are FALSE (default), then the training AIC value is returned. If more than one parameter is set to TRUE, then a named vector of AICs are returned, where the names are "train", "valid" or "xval".*

---

**Description**

Retrieve the AIC. If "train", "valid", and "xval" parameters are FALSE (default), then the training AIC value is returned. If more than one parameter is set to TRUE, then a named vector of AICs are returned, where the names are "train", "valid" or "xval".

**Usage**

```
h2o.aic(object, train = FALSE, valid = FALSE, xval = FALSE, ...)
```

**Arguments**

object	An <a href="#">H2OModel</a> or <a href="#">H2OModelMetrics</a> .
train	Retrieve the training AIC
valid	Retrieve the validation AIC
xval	Retrieve the cross-validation AIC
...	extra arguments to be passed if 'object' is of type <a href="#">H2OModel</a> (e.g. train=TRUE)

---

h2o.anomaly

*Anomaly Detection via H2O Deep Learning Model*


---

**Description**

Detect anomalies in an H2O dataset using an H2O deep learning model with auto-encoding.

**Usage**

```
h2o.anomaly(object, data, per_feature = FALSE)
```

**Arguments**

object	An <a href="#">H2OAutoEncoderModel</a> object that represents the model to be used for anomaly detection.
data	An <a href="#">H2OFrame</a> object.
per_feature	Whether to return the per-feature squared reconstruction error

**Value**

Returns an [H2OFrame](#) object containing the reconstruction MSE or the per-feature squared error.

**See Also**

[h2o.deeplearning](#) for making an [H2OAutoEncoderModel](#).

**Examples**

```
library(h2o)
localH2O = h2o.init()
prosPath = system.file("extdata", "prostate.csv", package = "h2o")
prostate.hex = h2o.importFile(localH2O, path = prosPath)
prostate.dl = h2o.deeplearning(x = 3:9, training_frame = prostate.hex, autoencoder = TRUE,
                             hidden = c(10, 10), epochs = 5)
prostate.anon = h2o.anomaly(prostate.dl, prostate.hex)
head(prostate.anon)
prostate.anon.per.feature = h2o.anomaly(prostate.dl, prostate.hex, per_feature=T)
head(prostate.anon.per.feature)
```

---

h2o.anyFactor	<i>Check H2OFrame columns for factors</i>
---------------	---

---

**Description**

Determines if any column of an H2OFrame object contains categorical data.

**Usage**

```
h2o.anyFactor(x)
```

**Arguments**

x                      An [H2OFrame](#) object.

**Value**

Returns a logical value indicating whether any of the columns in x are factors.

**Examples**

```
library(h2o)
localH2O <- h2o.init()
irisPath <- system.file("extdata", "iris_wheader.csv", package="h2o")
iris.hex <- h2o.importFile(localH2O, path = irisPath)
h2o.anyFactor(iris.hex)
```

---

h2o.assign	<i>Rename an H2O object.</i>
------------	------------------------------

---

### Description

Makes a copy of the data frame and gives it the desired the key.

### Usage

```
h2o.assign(data, key, deepCopy = FALSE)
```

### Arguments

data	An <a href="#">H2OFrame</a> object
key	The hex key to be associated with the H2O parsed data object
deepCopy	Should it do a deepCopy of the frame. Default is FALSE.

---

h2o.auc	<i>Retrieve the AUC</i>
---------	-------------------------

---

### Description

Retrieves the AUC value from an [H2OBinomialMetrics](#). If "train", "valid", and "xval" parameters are FALSE (default), then the training AUC value is returned. If more than one parameter is set to TRUE, then a named vector of AUCs are returned, where the names are "train", "valid" or "xval".

### Usage

```
h2o.auc(object, train = FALSE, valid = FALSE, xval = FALSE, ...)
```

### Arguments

object	An <a href="#">H2OBinomialMetrics</a> object.
train	Retrieve the training AUC
valid	Retrieve the validation AUC
xval	Retrieve the cross-validation AUC
...	extra arguments to be passed if 'object' is of type <a href="#">H2OModel</a> (e.g. train=TRUE)

### See Also

[h2o.giniCoef](#) for the Gini coefficient, [h2o.mse](#) for MSE, and [h2o.metric](#) for the various threshold metrics. See [h2o.performance](#) for creating H2OModelMetrics objects.

## Examples

```
library(h2o)
h2o.init()

prosPath <- system.file("extdata", "prostate.csv", package="h2o")
hex <- h2o.uploadFile(prosPath)

hex[,2] <- as.factor(hex[,2])
model <- h2o.gbm(x = 3:9, y = 2, training_frame = hex, distribution = "bernoulli")
perf <- h2o.performance(model, hex)
h2o.auc(perf)
```

---

h2o.betweenss	<i>Get the between cluster sum of squares. If "train", "valid", and "xval" parameters are FALSE (default), then the training betweenss value is returned. If more than one parameter is set to TRUE, then a named vector of betweenss' are returned, where the names are "train", "valid" or "xval".</i>
---------------	--

---

## Description

Get the between cluster sum of squares. If "train", "valid", and "xval" parameters are FALSE (default), then the training betweenss value is returned. If more than one parameter is set to TRUE, then a named vector of betweenss' are returned, where the names are "train", "valid" or "xval".

## Usage

```
h2o.betweenss(object, train = FALSE, valid = FALSE, xval = FALSE, ...)
```

## Arguments

object	An <a href="#">H2OClusteringModel</a> object.
train	Retrieve the training between cluster sum of squares
valid	Retrieve the validation between cluster sum of squares
xval	Retrieve the cross-validation between cluster sum of squares
...	further arguments to be passed on (currently unimplemented)

---

h2o.biases	<i>Return the respective bias vector</i>
------------	--

---

### Description

Return the respective bias vector

### Usage

```
h2o.biases(object, vector_id = 1, ...)
```

### Arguments

object	An <a href="#">H2OModel</a> or <a href="#">H2OModelMetrics</a>
vector_id	An integer, ranging from 1 to number of layers + 1, that specifies the bias vector to return.
...	further arguments to be passed to/from this method.

---

h2o.cbind	<i>Combine H2O Datasets by Columns</i>
-----------	--

---

### Description

Takes a sequence of H2O data sets and combines them by column

### Usage

```
h2o.cbind(...)
```

### Arguments

...	A sequence of <a href="#">H2OFrame</a> arguments. All datasets must exist on the same H2O instance (IP and port) and contain the same number of rows.
-----	---

### Value

An [H2OFrame](#) object containing the combined ... arguments column-wise.

### See Also

[cbind](#) for the base R method.



**Examples**

```
library(h2o)
localH2O <- h2o.init()
prosPath <- system.file("extdata", "prostate.csv", package="h2o")
prostate.hex <- h2o.uploadFile(localH2O, path = prosPath)
prostate.cbind <- h2o.cbind(prostate.hex, prostate.hex)
head(prostate.cbind)
```

---

h2o.centers	<i>Retrieve the Model Centers</i>
-------------	-----------------------------------

---

**Description**

Retrieve the Model Centers

**Usage**

```
h2o.centers(object, ...)
```

**Arguments**

object	An <a href="#">H2OClusteringModel</a> object.
...	further arguments to be passed on (currently unimplemented)

---

h2o.centersSTD	<i>Retrieve the Model Centers STD</i>
----------------	---------------------------------------

---

**Description**

Retrieve the Model Centers STD

**Usage**

```
h2o.centersSTD(object, ...)
```

**Arguments**

object	An <a href="#">H2OClusteringModel</a> object.
...	further arguments to be passed on (currently unimplemented)

---

h2o.centroid_stats	<i>Retrieve the centroid statistics If "train", "valid", and "xval" parameters are FALSE (default), then the training centroid stats value is returned. If more than one parameter is set to TRUE, then a named list of centroid stats data frames are returned, where the names are "train", "valid" or "xval".</i>
--------------------	--

---

### Description

Retrieve the centroid statistics If "train", "valid", and "xval" parameters are FALSE (default), then the training centroid stats value is returned. If more than one parameter is set to TRUE, then a named list of centroid stats data frames are returned, where the names are "train", "valid" or "xval".

### Usage

```
h2o.centroid_stats(object, train = FALSE, valid = FALSE, xval = FALSE,
  ...)
```

### Arguments

object	An <a href="#">H2OClusteringModel</a> object.
train	Retrieve the training centroid statistics
valid	Retrieve the validation centroid statistics
xval	Retrieve the cross-validation centroid statistics
...	further arguments to be passed on (currently unimplemented)

---

h2o.clearLog	<i>Delete All H2O R Logs</i>
--------------	------------------------------

---

### Description

Clear all H2O R command and error response logs from the local disk. Used primarily for debugging purposes.

### Usage

```
h2o.clearLog()
```

### See Also

[h2o.startLogging](#), [h2o.stopLogging](#), [h2o.openLog](#)

## Examples

```
library(h2o)
localH2O = h2o.init()
h2o.startLogging()
ausPath = system.file("extdata", "australia.csv", package="h2o")
australia.hex = h2o.importFile(localH2O, path = ausPath)
h2o.stopLogging()
h2o.clearLog()
```

---

h2o.clusterInfo	<i>Print H2O cluster info</i>
-----------------	-------------------------------

---

## Description

Print H2O cluster info

## Usage

```
h2o.clusterInfo(conn = h2o.getConnection())
```

## Arguments

conn	H2O connection object
------	-----------------------

---

h2o.clusterIsUp	<i>Determine if an H2O cluster is up or not</i>
-----------------	---

---

## Description

Determine if an H2O cluster is up or not

## Usage

```
h2o.clusterIsUp(conn = h2o.getConnection())
```

## Arguments

conn	H2O connection object
------	-----------------------

## Value

TRUE if the cluster is up; FALSE otherwise

---

h2o.clusterStatus	<i>Return the status of the cluster</i>
-------------------	---

---

### Description

Retrieve information on the status of the cluster running H2O.

### Usage

```
h2o.clusterStatus(conn = h2o.getConnection())
```

### Arguments

conn	the <a href="#">H2OConnection</a> object containing the IP address and port of the server running H2O.
------	--

### See Also

[H2OConnection](#), [h2o.init](#)

### Examples

```
localH2O <- h2o.init()
h2o.clusterStatus(localH2O)
```

---

h2o.cluster_sizes	<i>Retrieve the cluster sizes If "train", "valid", and "xval" parameters are FALSE (default), then the training cluster sizes value is returned. If more than one parameter is set to TRUE, then a named list of cluster size vectors are returned, where the names are "train", "valid" or "xval".</i>
-------------------	---

---

### Description

Retrieve the cluster sizes If "train", "valid", and "xval" parameters are FALSE (default), then the training cluster sizes value is returned. If more than one parameter is set to TRUE, then a named list of cluster size vectors are returned, where the names are "train", "valid" or "xval".

### Usage

```
h2o.cluster_sizes(object, train = FALSE, valid = FALSE, xval = FALSE, ...)
```

**Arguments**

object	An <a href="#">H2OClusteringModel</a> object.
train	Retrieve the training cluster sizes
valid	Retrieve the validation cluster sizes
xval	Retrieve the cross-validation cluster sizes
...	further arguments to be passed on (currently unimplemented)

---

h2o.coef	<i>Retrieve the model coefficeints</i>
----------	--

---

**Description**

Retrieve the model coefficeints

**Usage**

```
h2o.coef(object)
```

**Arguments**

object	an <a href="#">H2OModel</a> object.
--------	-------------------------------------

---

h2o.coef_norm	<i>Retrieve the normalized coefficients</i>
---------------	---

---

**Description**

Retrieve the normalized coefficients

**Usage**

```
h2o.coef_norm(object)
```

**Arguments**

object	an <a href="#">H2OModel</a> object.
--------	-------------------------------------

---

`h2o.confusionMatrix`     *Access H2O Confusion Matrices*

---

## Description

Retrieve either a single or many confusion matrices from H2O objects.

## Usage

```
h2o.confusionMatrix(object, ...)

## S4 method for signature 'H2OModel'
h2o.confusionMatrix(object, newdata, valid = FALSE, ...)

## S4 method for signature 'H2OModelMetrics'
h2o.confusionMatrix(object, thresholds = NULL,
  metrics = NULL)
```

## Arguments

<code>object</code>	Either an <a href="#">H2OModel</a> object or an <a href="#">H2OModelMetrics</a> object.
<code>...</code>	Extra arguments for extracting train or valid confusion matrices.
<code>newdata</code>	An <a href="#">H2OFrame</a> object that can be scored on. Requires a valid response column.
<code>valid</code>	Retrieve the validation metric.
<code>thresholds</code>	(Optional) A value or a list of valid values between 0.0 and 1.0. This value is only used in the case of <a href="#">H2OBinomialMetrics</a> objects.
<code>metrics</code>	(Optional) A metric or a list of valid metrics ("min_per_class_accuracy", "absolute_MCC", "tnr", "fmr", "fpr", "tpr", "precision", "accuracy", "f0point5", "f2", "f1"). This value is only used in the case of <a href="#">H2OBinomialMetrics</a> objects.

## Details

The [H2OModelMetrics](#) version of this function will only take [H2OBinomialMetrics](#) or [H2OMultinomialMetrics](#) objects. If no threshold is specified, all possible thresholds are selected.

## Value

Calling this function on [H2OModel](#) objects returns a confusion matrix corresponding to the [predict](#) function. If used on an [H2OBinomialMetrics](#) object, returns a list of matrices corresponding to the number of thresholds specified.

## See Also

[predict](#) for generating prediction frames, [h2o.performance](#) for creating [H2OModelMetrics](#).

## Examples

```
library(h2o)
h2o.init()
prosPath <- system.file("extdata", "prostate.csv", package="h2o")
hex <- h2o.uploadFile(prosPath)
hex[,2] <- as.factor(hex[,2])
model <- h2o.gbm(x = 3:9, y = 2, training_frame = hex, distribution = "bernoulli")
h2o.confusionMatrix(model, hex)
# Generating a ModelMetrics object
perf <- h2o.performance(model, hex)
h2o.confusionMatrix(perf)
```

---

h2o.createFrame	<i>Data Frame Creation in H2O</i>
-----------------	-----------------------------------

---

## Description

Creates a data frame in H2O with real-valued, categorical, integer, and binary columns specified by the user.

## Usage

```
h2o.createFrame(conn = h2o.getConnection(), key = "", rows = 10000,
  cols = 10, randomize = TRUE, value = 0, real_range = 100,
  categorical_fraction = 0.2, factors = 100, integer_fraction = 0.2,
  integer_range = 100, binary_fraction = 0.1, binary_ones_fraction = 0.02,
  missing_fraction = 0.01, response_factors = 2, has_response = FALSE,
  seed)
```

## Arguments

conn	A <a href="#">H2OConnection</a> object.
key	A string indicating the destination key. If empty, this will be auto-generated by H2O.
rows	The number of rows of data to generate.
cols	The number of columns of data to generate. Excludes the response column if has_response = TRUE.
randomize	A logical value indicating whether data values should be randomly generated. This must be TRUE if either categorical_fraction or integer_fraction is non-zero.
value	If randomize = FALSE, then all real-valued entries will be set to this value.
real_range	The range of randomly generated real values.
categorical_fraction	The fraction of total columns that are categorical.

factors	The number of (unique) factor levels in each categorical column.
integer_fraction	The fraction of total columns that are integer-valued.
integer_range	The range of randomly generated integer values.
binary_fraction	The fraction of total columns that are binary-valued.
binary_ones_fraction	The fraction of values in a binary column that are set to 1.
missing_fraction	The fraction of total entries in the data frame that are set to NA.
response_factors	If has_response = TRUE, then this is the number of factor levels in the response column.
has_response	A logical value indicating whether an additional response column should be pre-pended to the final H2O data frame. If set to TRUE, the total number of columns will be cols+1.
seed	A seed used to generate random values when randomize = TRUE.

**Value**

Returns a [H2OFrame](#) object.

**Examples**

```
library(h2o)
localH2O <- h2o.init()
hex <- h2o.createFrame(localH2O, rows = 1000, cols = 100, categorical_fraction = 0.1,
                        factors = 5, integer_fraction = 0.5, integer_range = 1,
                        has_response = TRUE)

head(hex)
summary(hex)

hex2 <- h2o.createFrame(localH2O, rows = 100, cols = 10, randomize = FALSE, value = 5,
                        categorical_fraction = 0, integer_fraction = 0)

summary(hex2)
```

---

h2o.cummax	<i>Cumulative Max</i>
------------	-----------------------

---

**Description**

Obtain the cumulative max of a column.

**Usage**

```
h2o.cummax(x)
```



Arguments

x                    An [H2OFrame](#) object.

Examples

```
localH2O <- h2o.init()
fr <- as.h2o(iris)
h2o.cummax(fr[,1])
```

---

h2o.cummin	<i>Cumulative Min</i>
------------	-----------------------

---

Description

Obtain the cumulative min of a column.

Usage

```
h2o.cummin(x)
```

Arguments

x                    An [H2OFrame](#) object.

Examples

```
localH2O <- h2o.init()
fr <- as.h2o(iris)
h2o.cummin(fr[,1])
```

---

h2o.cumprod	<i>Cumulative Product</i>
-------------	---------------------------

---

Description

Obtain the cumulative product of a column.

Usage

```
h2o.cumprod(x)
```

Arguments

x                    An [H2OFrame](#) object.

**Examples**

```
localH2O <- h2o.init()
fr <- as.h2o(iris)
h2o.cumprod(fr[,1])
```

---

h2o.cumsum

*Cumulative Sum*


---

**Description**

Obtain the cumulative sum of a column.

**Usage**

```
h2o.cumsum(x)
```

**Arguments**

x                      An [H2OFrame](#) object.

**Examples**

```
localH2O <- h2o.init()
fr <- as.h2o(iris)
h2o.cumsum(fr[,1])
```

---

h2o.cut

*Cut H2O Numeric Data to Factor*


---

**Description**

Divides the range of the H2O data into intervals and codes the values according to which interval they fall in. The leftmost interval corresponds to the level one, the next is level two, etc.

**Usage**

```
h2o.cut(x, breaks, labels = NULL, include.lowest = FALSE, right = TRUE,
        dig.lab = 3, ...)
```

**Arguments**

x	An <a href="#">H2OFrame</a> object with numeric columns.
breaks	A numeric vector of two or more unique cut points.
labels	Labels for the levels of the resulting category. By default, labels are constructed using "(a,b]" interval notation.
include.lowest	Logical, indicating if an 'x[i]' equal to the lowest (or highest, for right = FALSE 'breaks' value should be included
right	Logical, indicating if the intervals should be closed on the right (opened on the left) or vice versa.
dig.lab	Integer which is used when labels are not given, determines the number of digits used in formatting the break numbers.
...	Further arguments passed to or from other methods.

**Value**

Returns an [H2OFrame](#) object containing the factored data with intervals as levels.

**Examples**

```
library(h2o)
localH2O <- h2o.init()
irisPath <- system.file("extdata", "iris_wheader.csv", package="h2o")
iris.hex <- h2o.uploadFile(localH2O, path = irisPath, destination_frame = "iris.hex")
summary(iris.hex)

# Cut sepal length column into intervals determined by min/max/quantiles
sepal_len.cut = cut(iris.hex$sepal_len, c(4.2, 4.8, 5.8, 6, 8))
head(sepal_len.cut)
summary(sepal_len.cut)
```

---

h2o.day

---

*Convert Milliseconds to Day of Month in H2O Datasets*


---

**Description**

Converts the entries of a [H2OFrame](#) object from milliseconds to days of the month (on a 1 to 31 scale).

**Usage**

```
h2o.day(x)
```

```
day(x)
```

```
## S3 method for class H2OFrame
```

```
day(x)
```

**Arguments**

x                      An [H2OFrame](#) object.

**Value**

A [H2OFrame](#) object containing the entries of x converted to days of the month.

**See Also**

[h2o.month](#)

---

h2o.dayOfWeek

*Convert Milliseconds to Day of Week in H2O Datasets*

---

**Description**

Converts the entries of a [H2OFrame](#) object from milliseconds to days of the week (on a 0 to 6 scale).

**Usage**

```
h2o.dayOfWeek(x)
```

```
dayOfWeek(x)
```

```
## S3 method for class H2OFrame  
dayOfWeek(x)
```

**Arguments**

x                      An [H2OFrame](#) object.

**Value**

A [H2OFrame](#) object containing the entries of x converted to days of the week.

**See Also**

[h2o.day](#), [h2o.month](#)

---

h2o.dct	<i>Compute DCT of an H2OFrame</i>
---------	-----------------------------------

---

## Description

Compute the Discrete Cosine Transform of every row in the H2OFrame

## Usage

```
h2o.dct(data, destination_frame, dimensions, inverse = F)
```

## Arguments

data	An <a href="#">H2OFrame</a> object representing the dataset to transform
destination_frame	A frame ID for the result
dimensions	An array containing the 3 integer values for height, width, depth of each sample. The product of HxWxD must total up to less than the number of columns. For 1D, use c(L,1,1), for 2D, use C(N,M,1).
inverse	Whether to perform the inverse transform

## Examples

```
library(h2o)
localH2O = h2o.init()
df <- h2o.createFrame(localH2O, rows = 1000, cols = 8*16*24,
                      categorical_fraction = 0, integer_fraction = 0, missing_fraction = 0)
df1 <- h2o.dct(data=df, dimensions=c(8*16*24,1,1))
df2 <- h2o.dct(data=df1,dimensions=c(8*16*24,1,1),inverse=TRUE)
max(abs(df1-df2))

df1 <- h2o.dct(data=df, dimensions=c(8*16,24,1))
df2 <- h2o.dct(data=df1,dimensions=c(8*16,24,1),inverse=TRUE)
max(abs(df1-df2))

df1 <- h2o.dct(data=df, dimensions=c(8,16,24))
df2 <- h2o.dct(data=df1,dimensions=c(8,16,24),inverse=TRUE)
max(abs(df1-df2))
```

h2o.ddply

*Split H2O Dataset, Apply Function, and Return Results***Description**

For each subset of an H2O data set, apply a user-specified function, then combine the results. This is an experimental feature.

**Usage**

```
h2o.ddply(.data, .variables, .fun = NULL, ..., .progress = "none")
```

**Arguments**

<code>.data</code>	An <a href="#">H2OFrame</a> object to be processed.
<code>.variables</code>	Variables to split <code>.data</code> by, either the indices or names of a set of columns.
<code>.fun</code>	Function to apply to each subset grouping.
<code>.progress</code>	Name of the progress bar to use. <i>#TODO: (Currently unimplemented)</i>
<code>...</code>	Additional arguments passed on to <code>.fun</code> . <i>#TODO: (Currently unimplemented)</i>

**Value**

Returns a [H2OFrame](#) object containing the results from the split/apply operation, arranged

**See Also**

[ddply](#) for the plyr library implementation.

**Examples**

```
library(h2o)
localH2O <- h2o.init()

# Import iris dataset to H2O
irisPath <- system.file("extdata", "iris_wheader.csv", package = "h2o")
iris.hex <- h2o.uploadFile(localH2O, path = irisPath, destination_frame = "iris.hex")
# Add function taking mean of sepal_len column
fun = function(df) { sum(df[,1], na.rm = T)/nrow(df) }
# Apply function to groups by class of flower
# uses h2os ddply, since iris.hex is an H2OFrame object
res = h2o.ddply(iris.hex, "class", fun)
head(res)
```

---

h2o.deepfeatures*Feature Generation via H2O Deep Learning Model*

---

## Description

Extract the non-linear feature from an H2O data set using an H2O deep learning model.

## Usage

```
h2o.deepfeatures(object, data, layer = 1)
```

## Arguments

object	An <a href="#">H2OModel</a> object that represents the deep learning model to be used for feature extraction.
data	An <a href="#">H2OFrame</a> object.
layer	Index of the hidden layer to extract.

## Value

Returns an [H2OFrame](#) object with as many features as the number of units in the hidden layer of the specified index.

## See Also

`link{h2o.deeplearning}` for making deep learning models.

## Examples

```
library(h2o)
localH2O = h2o.init()
prosPath = system.file("extdata", "prostate.csv", package = "h2o")
prostate.hex = h2o.importFile(localH2O, path = prosPath)
prostate.dl = h2o.deeplearning(x = 3:9, y = 2, training_frame = prostate.hex,
                              hidden = c(100, 200), epochs = 5)
prostate.deepfeatures_layer1 = h2o.deepfeatures(prostate.dl, prostate.hex, layer = 1)
prostate.deepfeatures_layer2 = h2o.deepfeatures(prostate.dl, prostate.hex, layer = 2)
head(prostate.deepfeatures_layer1)
head(prostate.deepfeatures_layer2)
```

## Description

Performs Deep Learning neural networks on an [H2OFrame](#)

## Usage

```
h2o.deeplearning(x, y, training_frame, model_id = "",
  overwrite_with_best_model, validation_frame, checkpoint,
  autoencoder = FALSE, use_all_factor_levels = TRUE,
  activation = c("Rectifier", "Tanh", "TanhWithDropout",
    "RectifierWithDropout", "Maxout", "MaxoutWithDropout"), hidden = c(200,
    200), epochs = 10, train_samples_per_iteration = -2, seed,
  adaptive_rate = TRUE, rho = 0.99, epsilon = 1e-08, rate = 0.005,
  rate_annealing = 1e-06, rate_decay = 1, momentum_start = 0,
  momentum_ramp = 1e+06, momentum_stable = 0,
  nesterov_accelerated_gradient = TRUE, input_dropout_ratio = 0,
  hidden_dropout_ratios, l1 = 0, l2 = 0, max_w2 = Inf,
  initial_weight_distribution = c("UniformAdaptive", "Uniform", "Normal"),
  initial_weight_scale = 1, loss = c("Automatic", "CrossEntropy",
    "MeanSquare", "Absolute", "Huber"), distribution = c("AUTO", "gaussian",
    "bernoulli", "multinomial", "poisson", "gamma", "tweedie", "laplace",
    "huber"), tweedie_power = 1.5, score_interval = 5, score_training_samples,
  score_validation_samples, score_duty_cycle, classification_stop,
  regression_stop, quiet_mode, max_confusion_matrix_size, max_hit_ratio_k,
  balance_classes = FALSE, class_sampling_factors, max_after_balance_size,
  score_validation_sampling, diagnostics, variable_importances, fast_mode,
  ignore_const_cols, force_load_balance, replicate_training_data,
  single_node_mode, shuffle_training_data, sparse, col_major,
  average_activation, sparsity_beta, max_categorical_features,
  reproducible = FALSE, export_weights_and_biases = FALSE,
  offset_column = NULL, weights_column = NULL, nfolds = 0,
  fold_column = NULL, fold_assignment = c("AUTO", "Random", "Modulo"),
  keep_cross_validation_predictions = FALSE, ...)
```

## Arguments

x	A vector containing the character names of the predictors in the model.
y	The name of the response variable in the model.
training_frame	An <a href="#">H2OFrame</a> object containing the variables in the model.
model_id	(Optional) The unique id assigned to the resulting model. If none is given, an id will automatically be generated.
overwrite_with_best_model	Logical. If TRUE, overwrite the final model with the best model found during training. Defaults to TRUE.



validation_frame	(Optional) An <a href="#">H2OFrame</a> object indicating the validation dataset used to construct the confusion matrix. If left blank, this defaults to the training data when <code>nfolds = 0</code>
checkpoint	"Model checkpoint (either key or <code>H2ODeepLearningModel</code> ) to resume training with."
autoencoder	Enable auto-encoder for model building.
use_all_factor_levels	Logical. Use all factor levels of categorical variance. Otherwise the first factor level is omitted (without loss of accuracy). Useful for variable importances and auto-enabled for autoencoder.
activation	A string indicating the activation function to use. Must be either "Tanh", "TanhWithDropout", "Rectifier", "RectifierWithDropout", "Maxout", or "MaxoutWithDropout"
hidden	Hidden layer sizes (e.g. <code>c(100,100)</code> )
epochs	How many times the dataset should be iterated (streamed), can be fractional
train_samples_per_iteration	Number of training samples (globally) per MapReduce iteration. Special values are: <b>0</b> one epoch; <b>-1</b> all available data (e.g., replicated training data); or <b>-2</b> auto-tuning (default)
seed	Seed for random numbers (affects sampling) - Note: only reproducible when running single threaded
adaptive_rate	Logical. Adaptive learning rate (ADAELTA)
rho	Adaptive learning rate time decay factor (similarity to prior updates)
epsilon	Adaptive learning rate parameter, similar to learn rate annealing during initial training phase. Typical values are between $1.0e-10$ and $1.0e-4$
rate	Learning rate (higher => less stable, lower => slower convergence)
rate_annealing	Learning rate annealing: $(rate)/(1 + rate_{annealing} * samples)$
rate_decay	Learning rate decay factor between layers (N-th layer: $rate * \alpha^{(N - 1)}$ )
momentum_start	Initial momentum at the beginning of training (try 0.5)
momentum_ramp	Number of training samples for which momentum increases
momentum_stable	Final momentum after the amp is over (try 0.99)
nesterov_accelerated_gradient	Logical. Use Nesterov accelerated gradient (recommended)
input_dropout_ratio	A fraction of the features for each training row to be omitted from training in order to improve generalization (dimension sampling).
hidden_dropout_ratios	Input layer dropout ratio (can improve generalization) specify one value per hidden layer, defaults to 0.5
l1	L1 regularization (can add stability and improve generalization, causes many weights to become 0)

l2	L2 regularization (can add stability and improve generalization, causes many weights to be small)
max_w2	Constraint for squared sum of incoming weights per unit (e.g. Rectifier)
initial_weight_distribution	Can be "Uniform", "UniformAdaptive", or "Normal"
initial_weight_scale	Uniform: -value ... value, Normal: stddev
loss	Loss function: "Automatic", "CrossEntropy" (for classification only), "Mean-Square", "Absolute" (experimental) or "Huber" (experimental)
distribution	A character string. The distribution function of the response. Must be "AUTO", "bernoulli", "multinomial", "poisson", "gamma", "tweedie", "laplace", "huber" or "gaussian"
tweedie_power	Tweedie power (only for Tweedie distribution, must be between 1 and 2)
score_interval	Shortest time interval (in secs) between model scoring
score_training_samples	Number of training set samples for scoring (0 for all)
score_validation_samples	Number of validation set samples for scoring (0 for all)
score_duty_cycle	Maximum duty cycle fraction for scoring (lower: more training, higher: more scoring)
classification_stop	Stopping criterion for classification error fraction on training data (-1 to disable)
regression_stop	Stopping criterion for regression error (MSE) on training data (-1 to disable)
quiet_mode	Enable quiet mode for less output to standard output
max_confusion_matrix_size	Max. size (number of classes) for confusion matrices to be shown
max_hit_ratio_k	Max number (top K) of predictions to use for hit ratio computation (for multi-class only, 0 to disable)
balance_classes	Balance training data class counts via over/under-sampling (for imbalanced data)
class_sampling_factors	Desired over/under-sampling ratios per class (in lexicographic order). If not specified, sampling factors will be automatically computed to obtain class balance during training. Requires balance_classes.
max_after_balance_size	Maximum relative size of the training data after balancing class counts (can be less than 1.0)
score_validation_sampling	Method used to sample validation dataset for scoring
diagnostics	Enable diagnostics for hidden layers

variable_importances	Compute variable importances for input features (Gedeon method) - can be slow for large networks)
fast_mode	Enable fast mode (minor approximations in back-propagation)
ignore_const_cols	Ignore constant columns (no information can be gained anyway)
force_load_balance	Force extra load balancing to increase training speed for small datasets (to keep all cores busy)
replicate_training_data	Replicate the entire training dataset onto every node for faster training
single_node_mode	Run on a single node for fine-tuning of model parameters
shuffle_training_data	Enable shuffling of training data (recommended if training data is replicated and <code>train_samples_per_iteration</code> is close to <code>numRows * numNodes</code> )
sparse	Sparse data handling (Experimental)
col_major	Use a column major weight matrix for input layer. Can speed up forward propagation, but might slow down backpropagation (Experimental)
average_activation	Average activation for sparse auto-encoder (Experimental)
sparsity_beta	Sparsity regularization (Experimental)
max_categorical_features	Max. number of categorical features, enforced via hashing (Experimental)
reproducible	Force reproducibility on small data (requires setting the seed argument and this will be slow - only uses 1 thread)
export_weights_and_biases	Whether to export Neural Network weights and biases to H2O Frames"
offset_column	Specify the offset column.
weights_column	Specify the weights column.
nfolds	(Optional) Number of folds for cross-validation. If <code>nfolds &gt;= 2</code> , then validation must remain empty.
fold_column	(Optional) Column with cross-validation fold index assignment per observation
fold_assignment	Cross-validation fold assignment scheme, if <code>fold_column</code> is not specified Must be "AUTO", "Random" or "Modulo"
keep_cross_validation_predictions	Whether to keep the predictions of the cross-validation models
...	extra parameters to pass onto functions (not implemented)

**See Also**

[predict.H2OModel](#) for prediction.

### Examples

```
library(h2o)
localH2O <- h2o.init()
iris.hex <- as.h2o(iris)
iris.dl <- h2o.deeplearning(x = 1:4, y = 5, training_frame = iris.hex)

# now make a prediction
predictions <- h2o.predict(iris.dl, iris.hex)
```

---

h2o.dim	<i>Returns the Dimensions of a Parsed H2O Data Object.</i>
---------	--

---

### Description

Returns the number of rows and columns for an [H2OFrame](#) object.

### Usage

```
h2o.dim(x)

## S4 method for signature 'H2OFrame'
dim(x)
```

### Arguments

x                      An [H2OFrame](#) object.

### See Also

[dim](#) for the base R method.

### Examples

```
localH2O <- h2o.init()
irisPath <- system.file("extdata", "iris.csv", package="h2o")
iris.hex <- h2o.uploadFile(localH2O, path = irisPath)
dim(iris.hex)
```

---

h2o.downloadAllLogs	<i>Download H2O Log Files to Disk</i>
---------------------	---------------------------------------

---

**Description**

h2o.downloadAllLogs downloads all H2O log files to local disk. Generally used for debugging purposes.

**Usage**

```
h2o.downloadAllLogs(conn = h2o.getConnection(), dirname = ".",  
  filename = NULL)
```

**Arguments**

conn	An H2OConnection object pointing to a running H2O cluster.
dirname	(Optional) A character string indicating the directory that the log file should be saved in.
filename	(Optional) A character string indicating the name that the log file should be saved to.

**See Also**

[H2OConnection](#)

---

h2o.downloadCSV	<i>Download H2O Data to Disk</i>
-----------------	----------------------------------

---

**Description**

Download an H2O data set to a CSV file on the local disk

**Usage**

```
h2o.downloadCSV(data, filename)
```

**Arguments**

data	an <a href="#">H2OFrame</a> object to be downloaded.
filename	A string indicating the name that the CSV file should be saved to.

**Warning**

Files located on the H2O server may be very large! Make sure you have enough hard drive space to accomodate the entire file.



```
# h2o.download_pojo(my_model, getwd(), getjar = FALSE ) # save only the POJO to the current
#                                                    working directory, NOT RUN
h2o.download_pojo(my_model, getwd()) # save to the current working directory
```

---

h2o.exportFile	<i>Export an H2O Data Frame to a File</i>
----------------	---

---

## Description

Exports an [H2OFrame](#) (which can be either VA or FV) to a file. This file may be on the H2O instace's local filesystem, or to HDFS (preface the path with hdfs://) or to S3N (preface the path with s3n://).

## Usage

```
h2o.exportFile(data, path, force = FALSE)
```

## Arguments

data	An <a href="#">H2OFrame</a> data frame.
path	The path to write the file to. Must include the directory and filename. May be prefaced with hdfs:// or s3n://. Each row of data appears as line of the file.
force	logical, indicates how to deal with files that already exist.

## Details

In the case of existing files `force = TRUE` will overwrite the file. Otherwise, the operation will fail.

## Examples

```
library(h2o)
localH2O <- h2o.init()
irisPath <- system.file("extdata", "iris.csv", package = "h2o")
iris.hex <- h2o.uploadFile(localH2O, path = irisPath)

# These arent real paths
# h2o.exportFile(iris.hex, path = "/path/on/h2o/server/filesystem/iris.csv")
# h2o.exportFile(iris.hex, path = "hdfs://path/in/hdfs/iris.csv")
# h2o.exportFile(iris.hex, path = "s3n://path/in/s3/iris.csv")
```

---

h2o.exportHDFS	<i>Export a H2OFrame to HDFS</i>
----------------	----------------------------------

---

**Description**

Exports an [H2OFrame](#) to HDFS.

**Usage**

```
h2o.exportHDFS(data, path, force = FALSE)
```

**Arguments**

data	an <a href="#">H2OFrame</a> class object.
path	The path to write the data to. Must include the driectory and filename.
force	logical, indicates how to deal with files that already exist.

---

h2o.filterNACols	<i>Filter NA Columns</i>
------------------	--------------------------

---

**Description**

Filter NA Columns

**Usage**

```
h2o.filterNACols(data, frac = 0.2)
```

**Arguments**

data	A dataset to filter on.
frac	The threshold of NAs to allow per column (columns >= this threshold are filtered)



h2o.gbm

*Gradient Boosted Machines***Description**

Builds gradient boosted classification trees, and gradient boosted regression trees on a parsed data set.

**Usage**

```
h2o.gbm(x, y, training_frame, model_id, checkpoint, distribution = c("AUTO",
  "gaussian", "bernoulli", "multinomial", "poisson", "gamma", "tweedie"),
  tweedie_power = 1.5, ntrees = 50, max_depth = 5, min_rows = 10,
  learn_rate = 0.1, nbins = 20, nbins_cats = 1024,
  validation_frame = NULL, balance_classes = FALSE,
  max_after_balance_size = 1, seed, build_tree_one_node = FALSE,
  nfolds = 0, fold_column = NULL, fold_assignment = c("AUTO", "Random",
  "Modulo"), keep_cross_validation_predictions = FALSE,
  score_each_iteration = FALSE, offset_column = NULL,
  weights_column = NULL, ...)
```

**Arguments**

x	A vector containing the names or indices of the predictor variables to use in building the GBM model.
y	The name or index of the response variable. If the data does not contain a header, this is the column index number starting at 0, and increasing from left to right. (The response must be either an integer or a categorical variable).
training_frame	An <a href="#">H2OFrame</a> object containing the variables in the model.
model_id	(Optional) The unique id assigned to the resulting model. If none is given, an id will automatically be generated.
checkpoint	"Model checkpoint (either key or H2ODeepLearningModel) to resume training with."
distribution	A character string. The distribution function of the response. Must be "AUTO", "bernoulli", "multinomial", "poisson", "gamma", "tweedie" or "gaussian"
tweedie_power	Tweedie power (only for Tweedie distribution, must be between 1 and 2)
ntrees	A nonnegative integer that determines the number of trees to grow.
max_depth	Maximum depth to grow the tree.
min_rows	Minimum number of rows to assign to terminal nodes.
learn_rate	An integer from 0.0 to 1.0
nbins	For numerical columns (real/int), build a histogram of this many bins, then split at the best point
nbins_cats	For categorical columns (enum), build a histogram of this many bins, then split at the best point. Higher values can lead to more overfitting.

validation_frame	An <a href="#">H2OFrame</a> object indicating the validation dataset used to construct the confusion matrix. If left blank, this defaults to the training data when <code>nfolds = 0</code>
balance_classes	logical, indicates whether or not to balance training data class counts via over/under-sampling (for imbalanced data)
max_after_balance_size	Maximum relative size of the training data after balancing class counts (can be less than 1.0)
seed	Seed for random numbers (affects sampling when <code>balance_classes=T</code> )
build_tree_one_node	Run on one node only; no network overhead but fewer cpus used. Suitable for small datasets.
nfolds	(Optional) Number of folds for cross-validation. If <code>nfolds &gt;= 2</code> , then validation must remain empty.
fold_column	(Optional) Column with cross-validation fold index assignment per observation
fold_assignment	Cross-validation fold assignment scheme, if <code>fold_column</code> is not specified Must be "AUTO", "Random" or "Modulo"
keep_cross_validation_predictions	Whether to keep the predictions of the cross-validation models
score_each_iteration	Attempts to score each tree.
offset_column	Specify the offset column.
weights_column	Specify the weights column.
...	extra arguments to pass on (currently no implemented)

## Details

The default distribution function will guess the model type based on the response column type. In order to run properly, the response column must be an numeric for "gaussian" or an enum for "bernoulli" or "multinomial".

## See Also

[predict.H2OModel](#) for prediction.

## Examples

```
library(h2o)
localH2O = h2o.init()

# Run regression GBM on australia.hex data
ausPath <- system.file("extdata", "australia.csv", package="h2o")
australia.hex <- h2o.uploadFile(localH2O, path = ausPath)
independent <- c("premax", "salmax", "minairtemp", "maxairtemp", "maxsst",
                 "maxsoilmoist", "Max_czcs")
```

```
dependent <- "runoffnew"
h2o.gbm(y = dependent, x = independent, training_frame = australia.hex,
        ntrees = 3, max_depth = 3, min_rows = 2)
```

---

h2o.getConnection	<i>Retrieve an H2O Connection</i>
-------------------	-----------------------------------

---

**Description**

Attempt to recover an h2o connection.

**Usage**

```
h2o.getConnection()
```

**Value**

Returns an [H2OConnection](#) object.

---

h2o.getFrame	<i>Get an R Reference to an H2O Dataset</i>
--------------	---

---

**Description**

Get the reference to a frame with the given frame\_id in the H2O instance.

**Usage**

```
h2o.getFrame(frame_id, conn = h2o.getConnection(), linkToGC = FALSE)
```

**Arguments**

frame_id	A string indicating the unique frame of the dataset to retrieve.
conn	<a href="#">H2OConnection</a> object containing the IP address and port of the server running H2O.
linkToGC	a logical value indicating whether to remove the underlying frame from the H2O cluster when the R proxy object is garbage collected.

---

h2o.getGrid	<i>Get a grid object from H2O distributed K/V store.</i>
-------------	--

---

**Description**

Get a grid object from H2O distributed K/V store.

**Usage**

```
h2o.getGrid(grid_id, conn = h2o.getConnection())
```

**Arguments**

grid_id	ID of existing grid object to fetch
conn	H2O connection

**Examples**

```
library(h2o)
library(jsonlite)
localH2O <- h2o.init()
iris.hex <- as.h2o(iris)
h2o.grid("gbm", grid_id = "gbm_grid", x = c(1:4), y = 5,
        training_frame = iris.hex, hyper_params = list(ntrees = c(1,2,3)))
grid <- h2o.getGrid("gbm_grid")
# Get grid summary
summary(grid)
# Fetch grid models
model_ids <- grid@model_ids
models <- lapply(model_ids, function(id) { h2o.getModel(id)})
```

---

h2o.getModel	<i>Get an R reference to an H2O model</i>
--------------	---

---

**Description**

Returns a reference to an existing model in the H2O instance.

**Usage**

```
h2o.getModel(model_id, conn = h2o.getConnection(), linkToGC = FALSE)
```

**Arguments**

model_id	A string indicating the unique model_id of the model to retrieve.
conn	<a href="#">H2OConnection</a> object containing the IP address and port of the server running H2O.
linkToGC	A logical value indicating whether to remove the underlying model from the H2O cluster when the R proxy object is garbage collected.

**Value**

Returns an object that is a subclass of [H2OModel](#).

**Examples**

```
library(h2o)
localH2O <- h2o.init()

iris.hex <- as.h2o(iris, localH2O, "iris.hex")
model_id <- h2o.gbm(x = 1:4, y = 5, training_frame = iris.hex)@model_id
model.retrieved <- h2o.getModel(model_id, localH2O)
```

---

h2o.getTimezone	<i>Get the Time Zone on the H2O Cloud</i>
-----------------	---

---

**Description**

Get the Time Zone on the H2O Cloud

**Usage**

```
h2o.getTimezone(conn = h2o.getConnection())
```

**Arguments**

conn	An <a href="#">H2OConnection</a> object.
------	--

h2o.giniCoef

*Retrieve the GINI Coefficient***Description**

Retrieves the GINI coefficient from an [H2OBinomialMetrics](#). If "train", "valid", and "xval" parameters are FALSE (default), then the training GINI value is returned. If more than one parameter is set to TRUE, then a named vector of GINIs are returned, where the names are "train", "valid" or "xval".

**Usage**

```
h2o.giniCoef(object, train = FALSE, valid = FALSE, xval = FALSE, ...)
```

**Arguments**

object	an <a href="#">H2OBinomialMetrics</a> object.
train	Retrieve the training GINI Coefficient
valid	Retrieve the validation GINI Coefficient
xval	Retrieve the cross-validation GINI Coefficient
...	extra arguments to be passed if 'object' is of type <a href="#">H2OModel</a> (e.g. train=TRUE)

**See Also**

[h2o.auc](#) for AUC, [h2o.giniCoef](#) for the GINI coefficient, and [h2o.metric](#) for the various. See [h2o.performance](#) for creating H2OModelMetrics objects. threshold metrics.

**Examples**

```
library(h2o)
h2o.init()

prosPath <- system.file("extdata", "prostate.csv", package="h2o")
hex <- h2o.uploadFile(prosPath)

hex[,2] <- as.factor(hex[,2])
model <- h2o.gbm(x = 3:9, y = 2, training_frame = hex, distribution = "bernoulli")
perf <- h2o.performance(model, hex)
h2o.giniCoef(perf)
```

## Description

Fit a generalized linear model, specified by a response variable, a set of predictors, and a description of the error distribution.

## Usage

```
h2o.glm(x, y, training_frame, model_id, validation_frame, max_iterations = 50,
        beta_epsilon = 0, solver = c("IRLSM", "L_BFGS"), standardize = TRUE,
        family = c("gaussian", "binomial", "poisson", "gamma", "tweedie"),
        link = c("family_default", "identity", "logit", "log", "inverse",
                  "tweedie"), tweedie_variance_power = NaN, tweedie_link_power = NaN,
        alpha = 0.5, prior = 0, lambda = 1e-05, lambda_search = FALSE,
        nlambdas = -1, lambda_min_ratio = -1, nfolds = 0, fold_column = NULL,
        fold_assignment = c("AUTO", "Random", "Modulo"),
        keep_cross_validation_predictions = FALSE, beta_constraints = NULL,
        offset_column = NULL, weights_column = NULL, intercept = TRUE,
        max_active_predictors = -1, ...)
```

## Arguments

x	A vector containing the names or indices of the predictor variables to use in building the GLM model.
y	A character string or index that represent the response variable in the model.
training_frame	An <a href="#">H2OFrame</a> object containing the variables in the model.
model_id	(Optional) The unique id assigned to the resulting model. If none is given, an id will automatically be generated.
validation_frame	An <a href="#">H2OFrame</a> object containing the variables in the model.
max_iterations	A non-negative integer specifying the maximum number of iterations.
beta_epsilon	A non-negative number specifying the magnitude of the maximum difference between the coefficient estimates from successive iterations. Defines the convergence criterion for h2o.glm.
solver	A character string specifying the solver used: IRLSM (supports more features), L_BFGS (scales better for datasets with many columns)
standardize	A logical value indicating whether the numeric predictors should be standardized to have a mean of 0 and a variance of 1 prior to training the models.
family	A character string specifying the distribution of the model: gaussian, binomial, poisson, gamma, tweedie.

link	<p>A character string specifying the link function. The default is the canonical link for the family. The supported links for each of the family specifications are:</p> <p>"gaussian": "identity", "log", "inverse"</p> <p>"binomial": "logit", "log"</p> <p>"poisson": "log", "identity"</p> <p>"gamma": "inverse", "log", "identity"</p> <p>"tweedie": "tweedie"</p>
tweedie_variance_power	A numeric specifying the power for the variance function when family = "tweedie".
tweedie_link_power	A numeric specifying the power for the link function when family = "tweedie".
alpha	<p>A numeric in [0, 1] specifying the elastic-net mixing parameter. The elastic-net penalty is defined to be:</p> $P(\alpha, \beta) = (1 - \alpha)/2 \ \beta\ _2^2 + \alpha \ \beta\ _1 = \sum_j [(1 - \alpha)/2 \beta_j^2 + \alpha  \beta_j ]$ <p>, making alpha = 1 the lasso penalty and alpha = 0 the ridge penalty.</p>
prior	(Optional) A numeric specifying the prior probability of class 1 in the response when family = "binomial". The default prior is the observational frequency of class 1.
lambda	A non-negative shrinkage parameter for the elastic-net, which multiplies $P(\alpha, \beta)$ in the objective function. When lambda = 0, no elastic-net penalty is applied and ordinary generalized linear models are fit.
lambda_search	A logical value indicating whether to conduct a search over the space of lambda values starting from the lambda max, given lambda is interpreted as lambda min.
nlambdas	The number of lambda values to use when lambda_search = TRUE.
lambda_min_ratio	Smallest value for lambda as a fraction of lambda.max. By default if the number of observations is greater than the the number of variables then lambda_min_ratio = 0.0001; if the number of observations is less than the number of variables then lambda_min_ratio = 0.01.
nfolds	(Optional) Number of folds for cross-validation. If nfolds >= 2, then validation must remain empty.
fold_column	(Optional) Column with cross-validation fold index assignment per observation
fold_assignment	Cross-validation fold assignment scheme, if fold_column is not specified Must be "AUTO", "Random" or "Modulo"
keep_cross_validation_predictions	Whether to keep the predictions of the cross-validation models
beta_constraints	<p>A data.frame or H2OParsedData object with the columns ["names", "lower_bounds", "upper_bounds", "beta_given"], where each row corresponds to a predictor in the GLM. "names" contains the predictor names, "lower_bounds" and "upper_bounds" are the lower and upper bounds of beta, and "beta_given" is some supplied starting values for beta.</p>



`offset_column` Specify the offset column.  
`weights_column` Specify the weights column.  
`intercept` Logical, include constant term (intercept) in the model  
`max_active_predictors`  
 (Optional) Convergence criteria for number of predictors when using L1 penalty.  
`...` (Currently Unimplemented) coefficients.

## Value

A subclass of `H2OModel` is returned. The specific subclass depends on the machine learning task at hand (if it's binomial classification, then an `H2OBinomialModel` is returned, if it's regression then a `H2ORegressionModel` is returned). The default print-out of the models is shown, but further GLM-specific information can be queried out of the object. To access these various items, please refer to the `seealso` section below.

Upon completion of the GLM, the resulting object has coefficients, normalized coefficients, residual/null deviance, aic, and a host of model metrics including MSE, AUC (for logistic regression), degrees of freedom, and confusion matrices. Please refer to the more in-depth GLM documentation available here: <http://h2o-release.s3.amazonaws.com/h2o-dev/rel-shannon/2/docs-website/h2o-docs/index.html#Data+Science+Algorithms-GLM>,

## See Also

`predict.H2OModel` for prediction, `h2o.mse`, `h2o.auc`, `h2o.confusionMatrix`, `h2o.performance`, `h2o.giniCoef`, `h2o.logloss`, `h2o.varimp`, `h2o.scoreHistory`

## Examples

```

localH2O = h2o.init()

# Run GLM of CAPSULE ~ AGE + RACE + PSA + DCAPS
prostatePath = system.file("extdata", "prostate.csv", package = "h2o")
prostate.hex = h2o.importFile(localH2O, path = prostatePath, destination_frame = "prostate.hex")
h2o.glm(y = "CAPSULE", x = c("AGE", "RACE", "PSA", "DCAPS"), training_frame = prostate.hex,
        family = "binomial", nfolds = 0, alpha = 0.5, lambda_search = FALSE)

# Run GLM of VOL ~ CAPSULE + AGE + RACE + PSA + GLEASON
myX = setdiff(colnames(prostate.hex), c("ID", "DPROS", "DCAPS", "VOL"))
h2o.glm(y = "VOL", x = myX, training_frame = prostate.hex, family = "gaussian",
        nfolds = 0, alpha = 0.1, lambda_search = FALSE)

# GLM variable importance
# Also see:
# https://github.com/h2oai/h2o/blob/master/R/tests/testdir_demos/runit_demo_VI_all_algos.R
data.hex = h2o.importFile(
  localH2O,
  path = "https://raw.githubusercontent.com/h2oai/h2o/master/smldata/bank-additional-full.csv",
  destination_frame = "data.hex")
myX = 1:20
myY="y"

```

```
my.glm = h2o.glm(x=myX, y=myY, training_frame=data.hex, family="binomial", standardize=TRUE,
  lambda_search=TRUE)
```

h2o.glm

*Generalized Low Rank Model*

## Description

Generalized low rank decomposition of a H2O dataset.

## Usage

```
h2o.glm(training_frame, x, k, model_id, validation_frame, loading_name,
  ignore_const_cols, transform = c("NONE", "DEMEAN", "DESCALE", "STANDARDIZE",
  "NORMALIZE"), loss = c("Quadratic", "L1", "Huber", "Poisson", "Hinge",
  "Logistic"), multi_loss = c("Categorical", "Ordinal"), loss_by_col = NULL,
  loss_by_col_idx = NULL, regularization_x = c("None", "Quadratic", "L2",
  "L1", "NonNegative", "OneSparse", "UnitOneSparse", "Simplex"),
  regularization_y = c("None", "Quadratic", "L2", "L1", "NonNegative",
  "OneSparse", "UnitOneSparse", "Simplex"), gamma_x = 0, gamma_y = 0,
  max_iterations = 1000, init_step_size = 1, min_step_size = 0.001,
  init = c("Random", "PlusPlus", "SVD"), recover_svd = FALSE, seed)
```

## Arguments

training_frame	An <a href="#">H2OFrame</a> object containing the variables in the model.
x	(Optional) A vector containing the data columns on which k-means operates.
k	The rank of the resulting decomposition. This must be between 1 and the number of columns in the training frame, inclusive.
model_id	(Optional) The unique id assigned to the resulting model. If none is given, an id will automatically be generated.
validation_frame	An <a href="#">H2OFrame</a> object containing the variables in the model.
loading_name	(Optional) The unique name assigned to the loading matrix X in the XY decomposition. Automatically generated if none is provided.
ignore_const_cols	(Optional) A logical value indicating whether to ignore constant columns in the training frame. A column is constant if all of its non-missing values are the same value.
transform	A character string that indicates how the training data should be transformed before running PCA. Possible values are "NONE": for no transformation, "DEMEAN": for subtracting the mean of each column, "DESCALE": for dividing by the standard deviation of each column, "STANDARDIZE": for demeaning and descaling, and "NORMALIZE": for demeaning and dividing each column by its range (max - min).

loss	A character string indicating the default loss function for numeric columns. Possible values are "Quadratic" (default), "L1", "Huber", "Poisson", "Hinge" and "Logistic".
multi_loss	A character string indicating the default loss function for enum columns. Possible values are "Categorical" and "Ordinal".
loss_by_col	A vector of strings indicating the loss function for specific columns by corresponding index in loss_by_col_idx. Will override loss for numeric columns and multi_loss for enum columns.
loss_by_col_idx	A vector of column indices to which the corresponding loss functions in loss_by_col are assigned. Must be zero indexed.
regularization_x	A character string indicating the regularization function for the X matrix. Possible values are "None" (default), "Quadratic", "L2", "L1", "NonNegative", "OneSparse", "UnitOneSparse", and "Simplex".
regularization_y	A character string indicating the regularization function for the Y matrix. Possible values are "None" (default), "Quadratic", "L2", "L1", "NonNegative", "OneSparse", "UnitOneSparse", and "Simplex".
gamma_x	The weight on the X matrix regularization term.
gamma_y	The weight on the Y matrix regularization term.
max_iterations	The maximum number of iterations to run the optimization loop. Each iteration consists of an update of the X matrix, followed by an update of the Y matrix.
init_step_size	Initial step size. Divided by number of columns in the training frame when calculating the proximal gradient update. The algorithm begins at init_step_size and decreases the step size at each iteration until a termination condition is reached.
min_step_size	Minimum step size upon which the algorithm is terminated.
init	A character string indicating how to select the initial Y matrix. Possible values are "Random": for initialization to a random array from the standard normal distribution, "PlusPlus": for initialization using the clusters from k-means++ initialization, or "SVD": for initialization using the first k right singular vectors. Additionally, the user may specify the initial Y as a matrix, data.frame, H2OFrame, or list of vectors.
recover_svd	A logical value indicating whether the singular values and eigenvectors should be recovered during post-processing of the generalized low rank decomposition.
seed	(Optional) Random seed used to initialize the X and Y matrices.

### Value

Returns an object of class [H2ODimReductionModel](#).

### References

M. Udell, C. Horn, R. Zadeh, S. Boyd (2014). Generalized Low Rank Models[<http://arxiv.org/abs/1410.0342>]. Unpublished manuscript, Stanford Electrical Engineering Department.

## Examples

```
library(h2o)
localH2O <- h2o.init()
ausPath <- system.file("extdata", "australia.csv", package="h2o")
australia.hex <- h2o.uploadFile(localH2O, path = ausPath)
h2o.glm(training_frame = australia.hex, k = 5, loss = "Quadratic", regularization_x = "L1",
        gamma_x = 0.5, gamma_y = 0, max_iterations = 1000)
```

---

h2o.grid

*H2O Grid Support*


---

## Description

Provides a set of functions to launch a grid search and get its results.

## Usage

```
h2o.grid(algorithm, grid_id, ..., hyper_params = list(),
        is_supervised = NULL, do_hyper_params_check = FALSE,
        conn = h2o.getConnection())
```

## Arguments

algorithm	name of algorithm to use in grid search (gbm, randomForest, kmeans, glm, deeplearning, naivebayes, pca)
grid_id	optional id for resulting grid search, if it is not specified then it is autogenerated
...	arguments describing parameters to use with algorithm (i.e., x, y, training_frame). Look at the specific algorithm - h2o.gbm, h2o.glm, h2o.kmeans, h2o.deepLearning
hyper_params	list of hyper parameters (i.e., list(ntrees=c(1,2), max_depth=c(5,7)))
is_supervised	if specified then override default heuristics which decide if given algorithm name and parameters specify super/unsupervised algorithm.
do_hyper_params_check	perform client check for specified hyper parameters. It can be time expensive for large hyper space
conn	connection to H2O cluster

## Details

Launch grid search with given algorithm and parameters.

**Examples**

```
library(h2o)
library(jsonlite)
localH2O <- h2o.init()
iris.hex <- as.h2o(iris)
grid <- h2o.grid("gbm", x = c(1:4), y = 5, training_frame = iris.hex,
               hyper_params = list(ntrees = c(1,2,3)))
# Get grid summary
summary(grid)
# Fetch grid models
model_ids <- grid$model_ids
models <- lapply(model_ids, function(id) { h2o.getModel(id)})
```

---

h2o.group_by	<i>Group and Apply by Column</i>
--------------	----------------------------------

---

**Description**

Performs a group by and apply similar to ddply.

**Usage**

```
h2o.group_by(data, by, ..., order.by = NULL, gb.control = list(na.methods =
  NULL, col.names = NULL))
```

**Arguments**

data	an <a href="#">H2OFrame</a> object.
by	a list of column names
order.by	Takes a vector column names or indices specifying how to order the group by result.
gb.control	a list of how to handle NA values in the dataset as well as how to name output columns. See <a href="#">Details:</a> for more help.
...	any supported aggregate function.

**Details**

In the case of `na.methods` within `gb.control`, there are three possible settings. "all" will include NAs in computation of functions. "rm" will completely remove all NA fields. "ignore" will remove NAs from the numerator but keep the rows for computational purposes. If a list smaller than the number of columns groups is supplied, the list will be padded by "ignore".

Similar to `na.methods`, `col.names` will pad the list with the default column names if the length is less than the number of columns groups supplied.

**Value**

Returns a new [H2OFrame](#) object with columns equivalent to the number of groups created

---

h2o.gsub	<i>String Global Substitute</i>
----------	---------------------------------

---

**Description**

Mutates the input. Changes the all occurrences of pattern with replacement.

**Usage**

```
h2o.gsub(pattern, replacement, x, ignore.case = FALSE)
```

**Arguments**

pattern	The pattern to replace.
replacement	The replacement pattern.
x	The column on which to operate.
ignore.case	Case sensitive or not

---

h2o.head	<i>Return the Head or Tail of an H2O Dataset.</i>
----------	---

---

**Description**

Returns the first or last rows of an H2O parsed data object.

**Usage**

```
h2o.head(x, n = 6L, ...)
h2o.tail(x, n = 6L, ...)

## S4 method for signature H2OFrame
head(x, n = 6L, ...)

## S4 method for signature H2OFrame
tail(x, n = 6L, ...)
```

**Arguments**

x	An <a href="#">H2OFrame</a> object.
n	(Optional) A single integer. If positive, number of rows in x to return. If negative, all but the n first/last number of rows in x.
...	Further arguments passed to or from other methods.

**Value**

A data frame containing the first or last n rows of an [H2OFrame](#) object.

**Examples**

```
library(h2o)
localH2O <- h2o.init(ip = "localhost", port = 54321, startH2O = TRUE)
ausPath <- system.file("extdata", "australia.csv", package="h2o")
australia.hex <- h2o.uploadFile(localH2O, path = ausPath)
head(australia.hex, 10)
tail(australia.hex, 10)
```

---

h2o.hist	<i>Compute A Histogram</i>
----------	----------------------------

---

**Description**

Compute a histogram over a numeric column. If `breaks=="FD"`, the MAD is used over the IQR in computing bin width. Note that we do not beautify the breakpoints as R does.

**Usage**

```
h2o.hist(x, breaks = "Sturges", plot = TRUE)
```

**Arguments**

<code>x</code>	A single numeric column from an <a href="#">H2OFrame</a> .
<code>breaks</code>	Can be one of the following: A string: "Sturges", "Rice", "sqrt", "Doane", "FD", "Scott" A single number for the number of breaks splitting the range of the vec into number of breaks bins of equal width A vector of numbers giving the split points, e.g., <code>c(-50,213.2123,9324834)</code>
<code>plot</code>	A logical value indicating whether or not a plot should be generated (default is <code>TRUE</code> ).

---

h2o.hit_ratio_table	<i>Retrieve the Hit Ratios If "train", "valid", and "xval" parameters are FALSE (default), then the training Hit Ratios value is returned. If more than one parameter is set to TRUE, then a named list of Hit Ratio tables are returned, where the names are "train", "valid" or "xval".</i>
---------------------	---

---

### Description

Retrieve the Hit Ratios If "train", "valid", and "xval" parameters are FALSE (default), then the training Hit Ratios value is returned. If more than one parameter is set to TRUE, then a named list of Hit Ratio tables are returned, where the names are "train", "valid" or "xval".

### Usage

```
h2o.hit_ratio_table(object, train = FALSE, valid = FALSE, xval = FALSE,
  ...)
```

### Arguments

object	An <a href="#">H2OModel</a> object.
train	Retrieve the training Hit Ratio
valid	Retrieve the validation Hit Ratio
xval	Retrieve the cross-validation Hit Ratio
...	further arguments to be passed on (currently unimplemented)

---

h2o.hour	<i>Convert Milliseconds to Hour of Day in H2O Datasets</i>
----------	--

---

### Description

Converts the entries of a [H2OFrame](#) object from milliseconds to hours of the day (on a 0 to 23 scale).

### Usage

```
h2o.hour(x)

hour(x)

## S3 method for class H2OFrame
hour(x)
```

### Arguments

x	An <a href="#">H2OFrame</a> object.
---	-------------------------------------



**Value**

A [H2OFrame](#) object containing the entries of x converted to hours of the day.

**See Also**

[h2o.day](#)

---

h2o.ifelse

*H2O Apply Conditional Statement*


---

**Description**

Applies conditional statements to numeric vectors in H2O parsed data objects when the data are numeric.

**Usage**

```
h2o.ifelse(test, yes, no)

## S4 method for signature H2OFrame,ANY,ANY
ifelse(test, yes, no)

## S4 method for signature ANY,H2OFrame,H2OFrame
ifelse(test, yes, no)
```

**Arguments**

test	A logical description of the condition to be met (>, <, =, etc...)
yes	The value to return if the condition is TRUE.
no	The value to return if the condition is FALSE.

**Details**

Only numeric values can be tested, and only numeric results can be returned for either condition. Categorical data is not currently supported for this function and returned values cannot be categorical in nature.

**Value**

Returns a vector of new values matching the conditions stated in the ifelse call.

## Examples

```
localH2O = h2o.init(ip = "localhost", port = 54321, startH2O = TRUE)
ausPath = system.file("extdata", "australia.csv", package="h2o")
australia.hex = h2o.importFile(localH2O, path = ausPath)
australia.hex[,9] <- ifelse(australia.hex[,3] < 279.9, 1, 0)
summary(australia.hex)
```

---

h2o.importFile	<i>Import Files into H2O</i>
----------------	------------------------------

---

## Description

Imports files into an H2O cloud. The default behavior is to pass-through to the parse phase automatically.

## Usage

```
h2o.importFolder(path, conn = h2o.getConnection(), pattern = "",
  destination_frame = "", parse = TRUE, header = NA, sep = "",
  col.names = NULL, na.strings = NULL, parse_type = NULL)
```

```
h2o.importURL(path, conn = h2o.getConnection(), destination_frame = "",
  parse = TRUE, header = NA, sep = "", col.names = NULL,
  na.strings = NULL, parse_type = NULL)
```

```
h2o.importHDFS(path, conn = h2o.getConnection(), pattern = "",
  destination_frame = "", parse = TRUE, header = NA, sep = "",
  col.names = NULL, na.strings = NULL, parse_type = NULL)
```

```
h2o.uploadFile(path, conn = h2o.getConnection(), destination_frame = "",
  parse = TRUE, header = NA, sep = "", col.names = NULL,
  col.types = NULL, na.strings = NULL, progressBar = FALSE,
  parse_type = NULL)
```

## Arguments

path	The complete URL or normalized file path of the file to be imported. Each row of data appears as one line of the file.
conn	an <a href="#">H2OConnection</a> class object.
pattern	(Optional) Character string containing a regular expression to match file(s) in the folder.
destination_frame	(Optional) The unique hex key assigned to the imported file. If none is given, a key will automatically be generated based on the URL path.

parse	(Optional) A logical value indicating whether the file should be parsed after import.
header	(Optional) A logical value indicating whether the first line of the file contains column headers. If left empty, the parser will try to automatically detect this.
sep	(Optional) The field separator character. Values on each line of the file are separated by this character. If sep = "", the parser will automatically detect the separator.
col.names	(Optional) A <a href="#">H2ORawData</a> or <a href="#">H2OFrame</a> (version = 2) object containing a single delimited line with the column names for the file.
na.strings	(Optional) H2O will interpret these strings as missing.
parse_type	(Optional) Specify which parser type H2O will use. Valid types are "ARFF", "XLS", "CSV", "SVMLight"
col.types	(Optional) A vector to specify whether columns should be forced to a certain type upon import parsing.
progressBar	(Optional) When FALSE, tell H2O parse call to block synchronously instead of polling. This can be faster for small datasets but loses the progress bar.

## Details

Other than `h2o.uploadFile`, if the given path is relative, then it will be relative to the start location of the H2O instance. Additionally, the file must be on the same machine as the H2O cloud. In the case of `h2o.uploadFile`, a relative path will resolve relative to the working directory of the current R session.

Import an entire directory of files. If the given path is relative, then it will be relative to the start location of the H2O instance. The default behavior is to pass-through to the parse phase automatically.

`h2o.importURL` and `h2o.importHDFS` are both deprecated functions. Instead, use `h2o.importFile`

## Examples

```
localH2O = h2o.init(ip = "localhost", port = 54321, startH2O = TRUE)
prosPath = system.file("extdata", "prostate.csv", package = "h2o")
prostate.hex = h2o.uploadFile(localH2O, path = prosPath, destination_frame = "prostate.hex")
class(prostate.hex)
summary(prostate.hex)
```

---

```
h2o.impute      # children <- list(c(paste0('nAggs'), unlist(lapply(aggs, function(l)
                      .args.to.ast(.args=l))))))
```

---

## Description

Basic Imputation of H2O Vectors

**Usage**

```
h2o.impute(data, column, method = c("mean", "median", "mode"),
  combine_method = c("interpolate", "average", "lo", "hi"), by = NULL,
  inplace = TRUE)
```

**Arguments**

<code>data</code>	The dataset containing the column to impute.
<code>column</code>	The column to impute.
<code>method</code>	"mean" replaces NAs with the column mean; "median" replaces NAs with the column median; "mode" replaces with the most common factor (for factor columns only);
<code>combine_method</code>	If method is "median", then choose how to combine quantiles on even sample sizes. This parameter is ignored in all other cases.
<code>by</code>	group by columns
<code>inplace</code>	Perform the imputation inplace or make a copy. Default is to perform the imputation in place.

**Details**

Perform simple imputation on a single vector by filling missing values with aggregates computed on the "na.rm'd" vector. Additionally, it's possible to perform imputation based on groupings of columns from within data; these columns can be passed by index or name to the `by` parameter. If a factor column is supplied, then the method must be one "mode". Anything else results in a full stop.

The default method is selected based on the type of the column to impute. If the column is numeric then "mean" is selected; if it is categorical, then "mode" is selected. Otherwise column types (e.g. String, Time, UUID) are not supported.

**Value**

a H2OFrame with imputed values

**Examples**

```
h2o.init()
fr <- as.h2o(iris, destination_frame="iris")
fr[sample(nrow(fr),40),5] <- NA # randomly replace 50 values with NA
# impute with a group by
h2o.impute(fr, "Species", "mode", by=c("Sepal.Length", "Sepal.Width"))
```

h2o.init

*Initialize and Connect to H2O***Description**

Attempts to start and/or connect to and H2O instance.

**Usage**

```
h2o.init(ip = "127.0.0.1", port = 54321, startH2O = TRUE,
  forceDL = FALSE, Xmx, beta = FALSE, assertion = TRUE, license = NULL,
  nthreads = -2, max_mem_size = NULL, min_mem_size = NULL,
  ice_root = tempdir(), strict_version_check = TRUE, https = FALSE,
  insecure = FALSE, username = NA_character_, password = NA_character_)
```

**Arguments**

ip	Object of class character representing the hostname or IP address of the server where H2O is running.
port	Object of class numeric representing the port number of the H2O server.
startH2O	(Optional) A logical value indicating whether to try to start H2O from R if no connection with H2O is detected. This is only possible if ip = "localhost" or ip = "127.0.0.1". If an existing connection is detected, R does not start H2O.
forceDL	(Optional) A logical value indicating whether to force download of the H2O executable. Defaults to FALSE, so the executable will only be downloaded if it does not already exist in the h2o R library resources directory h2o/java/h2o.jar. This value is only used when R starts H2O.
Xmx	(Optional) (DEPRECATED) A character string specifying the maximum size, in bytes, of the memory allocation pool to H2O. This value must be a multiple of 1024 greater than 2MB. Append the letter m or M to indicate megabytes, or g or G to indicate gigabytes. This value is only used when R starts H2O.
beta	(Optional) A logical value indicating whether H2O should launch in beta mode. This value is only used when R starts H2O.
assertion	(Optional) A logical value indicating whether H2O should be launched with assertions enabled. Used mainly for error checking and debugging purposes. This value is only used when R starts H2O.
license	(Optional) A character string value specifying the full path of the license file. This value is only used when R starts H2O.
nthreads	(Optional) Number of threads in the thread pool. This relates very closely to the number of CPUs used. -2 means use the CRAN default of 2 CPUs. -1 means use all CPUs on the host. A positive integer specifies the number of CPUs directly. This value is only used when R starts H2O.

max_mem_size	(Optional) A character string specifying the maximum size, in bytes, of the memory allocation pool to H2O. This value must a multiple of 1024 greater than 2MB. Append the letter m or M to indicate megabytes, or g or G to indicate gigabytes. This value is only used when R starts H2O.
min_mem_size	(Optional) A character string specifying the minimum size, in bytes, of the memory allocation pool to H2O. This value must a multiple of 1024 greater than 2MB. Append the letter m or M to indicate megabytes, or g or G to indicate gigabytes. This value is only used when R starts H2O.
ice_root	(Optional) A directory to handle object spillage. The default varies by OS.
strict_version_check	(Optional) Setting this to FALSE is unsupported and should only be done when advised by technical support.
https	(Optional) Set this to TRUE to use https instead of http.
insecure	(Optional) Set this to TRUE to disable SSL certificate checking.
username	(Optional) Username to login with.
password	(Optional) Password to login with.

### Details

By default, this method first checks if an H2O instance is connectible. If it cannot connect and `start = TRUE` with `ip = "localhost"`, it will attempt to start an instance of H2O at `localhost:54321`. Otherwise it stops with an error.

When initializing H2O locally, this method searches for `h2o.jar` in the R library resources (`system.file("java", "h2o.jar", package="h2o")`) and if the file does not exist, it will automatically attempt to download the correct version from Amazon S3. The user must have Internet access for this process to be successful.

Once connected, the method checks to see if the local H2O R package version matches the version of H2O running on the server. If there is a mismatch and the user indicates she wishes to upgrade, it will remove the local H2O R package and download/install the H2O R package from the server.

### Value

this method will load it and return a `H2OConnection` object containing the IP address and port number of the H2O server.

### Note

Users may wish to manually upgrade their package (rather than waiting until being prompted), which requires that they fully uninstall and reinstall the H2O package, and the H2O client package. You must unload packages running in the environment before upgrading. It's recommended that users restart R or R studio after upgrading.

### See Also

[H2O R package documentation](#) for more details. [h2o.shutdown](#) for shutting down from R.

**Examples**

```
# Try to connect to a local H2O instance that is already running.
# If not found, start a local H2O instance from R with the default settings.
localH2O = h2o.init()

# Try to connect to a local H2O instance.
# If not found, raise an error.
localH2O = h2o.init(startH2O = FALSE)

# Try to connect to a local H2O instance that is already running.
# If not found, start a local H2O instance from R with 5 gigabytes of memory.
localH2O = h2o.init(max_mem_size = "5g")
```

---

h2o.insertMissingValues

*Inserting Missing Values to an H2O DataFrame*


---

**Description**

\*This is primarily used for testing\*. Randomly replaces a user-specified fraction of entries in a H2O dataset with missing values.

**Usage**

```
h2o.insertMissingValues(data, fraction = 0.1, seed = -1)
```

**Arguments**

data	An <a href="#">H2OFrame</a> object representing the dataset.
fraction	A number between 0 and 1 indicating the fraction of entries to replace with missing.
seed	A random number used to select which entries to replace with missing values. Default of seed = -1 will automatically generate a seed in H2O.

**WARNING**

This will modify the original dataset. Unless this is intended, this function should only be called on a subset of the original.

**Examples**

```
library(h2o)
localH2O <- h2o.init()
irisPath <- system.file("extdata", "iris.csv", package = "h2o")
iris.hex <- h2o.importFile(localH2O, path = irisPath)
```

```
summary(iris.hex)
irismiss.hex <- h2o.insertMissingValues(iris.hex, fraction = 0.25)
head(irismiss.hex)
summary(irismiss.hex)
```

h2o.interaction

*Categorical Interaction Feature Creation in H2O***Description**

Creates a data frame in H2O with n-th order interaction features between categorical columns, as specified by the user.

**Usage**

```
h2o.interaction(data, destination_frame, factors, pairwise, max_factors,
  min_occurrence)
```

**Arguments**

data	An <a href="#">H2OFrame</a> object containing the categorical columns.
destination_frame	A string indicating the destination key. If empty, this will be auto-generated by H2O.
factors	Factor columns (either indices or column names).
pairwise	Whether to create pairwise interactions between factors (otherwise create one higher-order interaction). Only applicable if there are 3 or more factors.
max_factors	Max. number of factor levels in pair-wise interaction terms (if enforced, one extra catch-all factor will be made)
min_occurrence	Min. occurrence threshold for factor levels in pair-wise interaction terms

**Value**

Returns a [H2OFrame](#) object.

**Examples**

```
library(h2o)
localH2O <- h2o.init()

# Create some random data
myframe = h2o.createFrame(localH2O, framekey, rows = 20, cols = 5,
  seed = -12301283, randomize = TRUE, value = 0,
  categorical_fraction = 0.8, factors = 10, real_range = 1,
  integer_fraction = 0.2, integer_range = 10,
```



```

        binary_fraction = 0, binary_ones_fraction = 0.5,
        missing_fraction = 0.2,
        response_factors = 1)
# Turn integer column into a categorical
myframe[,5] <- as.factor(myframe[,5])
head(myframe, 20)

# Create pairwise interactions
pairwise <- h2o.interaction(myframe, destination_frame = pairwise,
                           factors = list(c(1,2),c("C2","C3","C4")),
                           pairwise=TRUE, max_factors = 10, min_occurrence = 1)

head(pairwise, 20)
h2o.levels(pairwise,2)

# Create 5-th order interaction
higherorder <- h2o.interaction(myframe, destination_frame = higherorder, factors = c(1,2,3,4,5),
                              pairwise=FALSE, max_factors = 10000, min_occurrence = 1)

head(higherorder, 20)

# Limit the number of factors of the "categoricalized" integer column
# to at most 3 factors, and only if they occur at least twice
head(myframe[,5], 20)
trim_integer_levels <- h2o.interaction(myframe, destination_frame = trim_integers, factors = "C5",
                                       pairwise = FALSE, max_factors = 3, min_occurrence = 2)

head(trim_integer_levels, 20)

# Put all together
myframe <- h2o.cbind(myframe, pairwise, higherorder, trim_integer_levels)
myframe
head(myframe,20)
summary(myframe)

```

---

h2o.is\_client

---

*Check Client Mode Connection*


---

## Description

Check Client Mode Connection

## Usage

h2o.is\_client()

---

h2o.killMinus3	<i>Dump the stack into the JVM's stdout.</i>
----------------	--

---

**Description**

A poor man's profiler, but effective.

**Usage**

```
h2o.killMinus3(conn = h2o.getConnection())
```

**Arguments**

conn	an <a href="#">H2OConnection</a> class object.
------	--

---

h2o.kmeans	<i>KMeans Model in H2O</i>
------------	----------------------------

---

**Description**

Performs k-means clustering on an H2O dataset.

**Usage**

```
h2o.kmeans(training_frame, x, k, model_id, max_iterations = 1000,
  standardize = TRUE, init = c("Furthest", "Random", "PlusPlus"), seed,
  nfolds = 0, fold_column = NULL, fold_assignment = c("AUTO", "Random",
    "Modulo"), keep_cross_validation_predictions = FALSE)
```

**Arguments**

training_frame	An <a href="#">H2OFrame</a> object containing the variables in the model.
x	(Optional) A vector containing the data columns on which k-means operates.
k	The number of clusters. Must be between 1 and 1e7 inclusive. k may be omitted if the user specifies the initial centers in the init parameter. If k is not omitted, in this case, then it should be equal to the number of user-specified centers.
model_id	(Optional) The unique id assigned to the resulting model. If none is given, an id will automatically be generated.
max_iterations	The maximum number of iterations allowed. Must be between 0
standardize	Logical, indicates whether the data should be standardized before running k-means.

init	A character string that selects the initial set of k cluster centers. Possible values are "Random": for random initialization, "PlusPlus": for k-means plus initialization, or "Furthest": for initialization at the furthest point from each successive center. Additionally, the user may specify a the initial centers as a matrix, data.frame, H2OFrame, or list of vectors. For matrices, data.frames, and H2OFrames, each row of the respective structure is an initial center. For lists of vectors, each vector is an initial center.
seed	(Optional) Random seed used to initialize the cluster centroids.
nfolds	(Optional) Number of folds for cross-validation. If nfolds >= 2, then validation must remain empty.
fold_column	(Optional) Column with cross-validation fold index assignment per observation
fold_assignment	Cross-validation fold assignment scheme, if fold_column is not specified Must be "AUTO", "Random" or "Modulo"
keep_cross_validation_predictions	Whether to keep the predictions of the cross-validation models

**Value**

Returns an object of class [H2OClusteringModel](#).

**See Also**

[h2o.cluster\\_sizes](#), [h2o.totss](#), [h2o.num\\_iterations](#), [h2o.betweenss](#), [h2o.tot\\_withinss](#), [h2o.withinss](#), [h2o.centersSTD](#), [h2o.centers](#)

**Examples**

```
library(h2o)
localH2O <- h2o.init()
prosPath <- system.file("extdata", "prostate.csv", package="h2o")
prostate.hex <- h2o.uploadFile(localH2O, path = prosPath)
h2o.kmeans(training_frame = prostate.hex, k = 10, x = c("AGE", "RACE", "VOL", "GLEASON"))
```

---

h2o.length	<i>Returns the Length of a Parsed H2O Data Object.</i>
------------	--

---

**Description**

Returns the length of an [H2OFrame](#)

**Usage**

```
h2o.length(x)

## S4 method for signature 'H2OFrame'
length(x)
```

**Arguments**

`x` An [H2OFrame](#) object.

**See Also**

[length](#) for the base R method.

**Examples**

```
localH2O <- h2o.init()
irisPath <- system.file("extdata", "iris.csv", package = "h2o")
iris.hex <- h2o.uploadFile(localH2O, path = irisPath)
length(iris.hex)
```

---

h2o.levels

*Return the levels from the column requested column.*

---

**Description**

Return the levels from the column requested column.

**Usage**

```
h2o.levels(x, i)
```

**Arguments**

`x` An [H2OFrame](#) object.

`i` The index of the column whose domain is to be returned.

**See Also**

[levels](#) for the base R method.

**Examples**

```
localH2O <- h2o.init()
iris.hex <- as.h2o(localH2O, iris)
h2o.levels(iris.hex, 5) # returns "setosa" "versicolor" "virginica"
```

---

h2o.listTimezones	<i>List all of the Time Zones Acceptable by the H2O Cloud.</i>
-------------------	--

---

**Description**

List all of the Time Zones Acceptable by the H2O Cloud.

**Usage**

```
h2o.listTimezones(conn = h2o.getConnection())
```

**Arguments**

conn	An H2OConnection object.
------	--------------------------

---

h2o.loadModel	<i>Load H2O Model from HDFS or Local Disk</i>
---------------	---

---

**Description**

Load a saved H2O model from disk.

**Usage**

```
h2o.loadModel(path, conn = h2o.getConnection())
```

**Arguments**

path	The path of the H2O Model to be imported. For example, if the ‘dir’ argument in h2o.saveModel was set to "/Users/UserName/Desktop" then the ‘path’ argument in h2o.loadModel should be set to something like "/Users/UserName/Desktop/K-meansModel__a7cebf318ca5827185e209edf47c4052"
conn	an <a href="#">H2OConnection</a> object containing the IP address and port of the server running H2O.

**Value**

Returns a [H2OModel](#) object of the class corresponding to the type of model built.

**See Also**

[h2o.saveModel](#), [H2OModel](#)

**Examples**

```
# library(h2o)
# localH2O = h2o.init()
# prosPath = system.file("extdata", "prostate.csv", package = "h2o")
# prostate.hex = h2o.importFile(localH2O, path = prosPath, destination_frame = "prostate.hex")
# prostate.glm = h2o.glm(y = "CAPSULE", x = c("AGE", "RACE", "PSA", "DCAPS"),
#   training_frame = prostate.hex, family = "binomial", alpha = 0.5)
# glmmodel.path = h2o.saveModel(prostate.glm, dir = "/Users/UserName/Desktop")
# glmmodel.load = h2o.loadModel(localH2O, glmmodel.path)
```

---

h2o.logAndEcho

*Log a message on the server-side logs*


---

**Description**

This is helpful when running several pieces of work one after the other on a single H2O cluster and you want to make a notation in the H2O server side log where one piece of work ends and the next piece of work begins.

**Usage**

```
h2o.logAndEcho(message, conn = h2o.getConnection())
```

**Arguments**

message	A character string with the message to write to the log.
conn	An H2OConnection object pointing to a running H2O cluster.

**Details**

h2o.logAndEcho sends a message to H2O for logging. Generally used for debugging purposes.

**See Also**

[H2OConnection](#)

---

h2o.logloss	<i>Retrieve the Log Loss Value</i>
-------------	------------------------------------

---

**Description**

Retrieves the log loss output for a [H2OBinomialMetrics](#) or [H2OMultinomialMetrics](#) object. If "train", "valid", and "xval" parameters are FALSE (default), then the training Log Loss value is returned. If more than one parameter is set to TRUE, then a named vector of Log Losses are returned, where the names are "train", "valid" or "xval".

**Usage**

```
h2o.logloss(object, train = FALSE, valid = FALSE, xval = FALSE, ...)
```

**Arguments**

object	a <a href="#">H2OModelMetrics</a> object of the correct type.
train	Retrieve the training Log Loss
valid	Retrieve the validation Log Loss
xval	Retrieve the cross-validation Log Loss
...	Extra arguments to be passed if 'object' is of type <a href="#">H2OModel</a> (e.g. train=TRUE)

---

h2o.ls	<i>List Keys on an H2O Cluster</i>
--------	------------------------------------

---

**Description**

Accesses a list of object keys in the running instance of H2O.

**Usage**

```
h2o.ls(conn = h2o.getConnection())
```

**Arguments**

conn	An <a href="#">H2OConnection</a> object containing the IP address and port number of the H2O server.
------	--

**Value**

Returns a list of hex keys in the current H2O instance.

## Examples

```
library(h2o)
localH2O <- h2o.init()
prosPath <- system.file("extdata", "prostate.csv", package="h2o")
prostate.hex <- h2o.uploadFile(localH2O, path = prosPath)
h2o.ls(localH2O)
```

---

h2o.makeGLMModel	<i>Set betas of an existing H2O GLM Model</i>
------------------	---

---

## Description

This function allows setting betas of an existing glm model.

## Usage

```
h2o.makeGLMModel(model, beta)
```

## Arguments

model	an <a href="#">H2OModel</a> corresponding from a h2o.glm call.
beta	a new set of betas (a named vector)

---

h2o.match	<i>Value Matching in H2O</i>
-----------	------------------------------

---

## Description

match and %in% return values similar to the base R generic functions.

## Usage

```
h2o.match(x, table, nomatch = 0, incomparables = NULL)
```

```
## S4 method for signature H2OFrame
match(x, table, nomatch = 0, incomparables = NULL)
```

```
## S4 method for signature H2OFrame,character
x %in% table
```

```
## S4 method for signature H2OFrame,numeric
x %in% table
```



**Arguments**

x	a categorical vector from an <a href="#">H2OFrame</a> object with values to be matched.
table	an R object to match x against.
nomatch	the value to be returned in the case when no match is found.
incomparables	a vector of values that cannot be matched. Any value in x matching a value in this vector is assigned the nomatch value.

**See Also**

[match](#) for base R implementation.

**Examples**

```
h2o.init()
hex <- as.h2o(iris)
match(hex[,5], c("setosa", "versicola")) # versipepsi
```

---

h2o.mean	<i>Mean of a column</i>
----------	-------------------------

---

**Description**

Obtain the mean of a column of a parsed H2O data object.

**Usage**

```
h2o.mean(x, trim = 0, na.rm = FALSE, ...)

## S4 method for signature H2OFrame
mean(x, trim = 0, na.rm = FALSE, ...)
```

**Arguments**

x	An <a href="#">H2OFrame</a> object.
trim	The fraction (0 to 0.5) of observations to trim from each end of x before the mean is computed.
na.rm	A logical value indicating whether NA or missing values should be stripped before the computation.
...	Further arguments to be passed from or to other methods.

**See Also**

[mean](#) for the base R implementation.

**Examples**

```

localH2O <- h2o.init()
prosPath <- system.file("extdata", "prostate.csv", package="h2o")
prostate.hex <- h2o.uploadFile(localH2O, path = prosPath)
mean(prostate.hex$AGE)

```

---

h2o.mean\_residual\_deviance

*Retrieve the Mean Residual Deviance value*

---

**Description**

Retrieves the Mean Residual Deviance value from an H2O model. If "train", "valid", and "xval" parameters are FALSE (default), then the training Mean Residual Deviance value is returned. If more than one parameter is set to TRUE, then a named vector of Mean Residual Deviances are returned, where the names are "train", "valid" or "xval".

**Usage**

```

h2o.mean_residual_deviance(object, train = FALSE, valid = FALSE,
  xval = FALSE, ...)

```

**Arguments**

object	An <a href="#">H2OModel</a> object.
train	Retrieve the training Mean Residual Deviance
valid	Retrieve the validation Mean Residual Deviance
xval	Retrieve the cross-validation Mean Residual Deviance
...	extra arguments to be passed if 'object' is of type <a href="#">H2OModel</a> (e.g. train=TRUE)

**Examples**

```

library(h2o)

h <- h2o.init()
fr <- as.h2o(iris)

m <- h2o.deeplearning(x=2:5,y=1,training_frame=fr)

h2o.mean_residual_deviance(m)

```

---

h2o.median	<i>H2O Median</i>
------------	-------------------

---

**Description**

Compute the arithmetic mean of a [H2OFrame](#).

**Usage**

```
h2o.median(x, na.rm = TRUE)

## S4 method for signature 'H2OFrame'
median(x, na.rm = TRUE)
```

**Arguments**

x	An <a href="#">H2OFrame</a> object.
na.rm	a logical, indicating whether na's are omitted.

**Examples**

```
localH2O <- h2o.init()
prosPath <- system.file("extdata", "prostate.csv", package="h2o")
prostate.hex <- h2o.uploadFile(localH2O, path = prosPath, destination_frame = "prostate.hex")
```

---

h2o.merge	<i>Merge Two H2O Data Frames</i>
-----------	----------------------------------

---

**Description**

Merges two [H2OFrame](#) objects by shared column names. Unlike the base R implementation, h2o.merge only supports merging through shared column names.

**Usage**

```
h2o.merge(x, y, all.x = FALSE, all.y = FALSE)
```

**Arguments**

x,y	<a href="#">H2OFrame</a> objects
all.x	a logical value indicating whether or not shared values are preserved or ignored in x.
all.y	a logical value indicating whether or not shared values are preserved or ignored in y.

**Details**

In order for `h2o.merge` to work in multinode clusters, one of the datasets must be small enough to exist in every node. Currently, this function only supports `all.x = TRUE`. All other permutations will fail.

**Examples**

```
h2o.init()
left <- data.frame(fruit = c(apple, orange, banana, lemon, strawberry, blueberry),
  color = c(red, orange, yellow, yellow, red, blue))
right <- data.frame(fruit = c(apple, orange, banana, lemon, strawberry, watermelon),
  citrus = c(FALSE, TRUE, FALSE, TRUE, FALSE, FALSE))
l.hex <- as.h2o(left)
r.hex <- as.h2o(right)
left.hex <- h2o.merge(l.hex, r.hex, all.x = TRUE)
```

---

h2o.metric

*H2O Model Metric Accessor Functions*


---

**Description**

A series of functions that retrieve model metric details.

**Usage**

```
h2o.metric(object, thresholds, metric)

h2o.F0point5(object, thresholds)

h2o.F1(object, thresholds)

h2o.F2(object, thresholds)

h2o.accuracy(object, thresholds)

h2o.error(object, thresholds)

h2o.maxPerClassError(object, thresholds)

h2o.mcc(object, thresholds)

h2o.precision(object, thresholds)

h2o.tpr(object, thresholds)

h2o.fpr(object, thresholds)
```

```
h2o.fnr(object, thresholds)
h2o.tnr(object, thresholds)
h2o.recall(object, thresholds)
h2o.sensitivity(object, thresholds)
h2o.fallout(object, thresholds)
h2o.missrate(object, thresholds)
h2o.specificity(object, thresholds)
```

### Arguments

object	An <a href="#">H2OModelMetrics</a> object of the correct type.
thresholds	A value or a list of values between 0.0 and 1.0.
metric	A specified paramter to retrieve.

### Details

Many of these functions have an optional thresholds parameter. Currently only increments of 0.1 are allowed. If not specified, the functions will return all possible values. Otherwise, the function will return the value for the indicated threshold.

Currently, the these functions are only supported by [H2OBinomialMetrics](#) objects.

### Value

Returns either a single value, or a list of values.

### See Also

[h2o.auc](#) for AUC, [h2o.giniCoef](#) for the GINI coefficient, and [h2o.mse](#) for MSE. See [h2o.performance](#) for creating H2OModelMetrics objects.

### Examples

```
library(h2o)
h2o.init()

prosPath <- system.file("extdata", "prostate.csv", package="h2o")
hex <- h2o.uploadFile(prosPath)

hex[,2] <- as.factor(hex[,2])
model <- h2o.gbm(x = 3:9, y = 2, training_frame = hex, distribution = "bernoulli")
perf <- h2o.performance(model, hex)
h2o.F1(perf)
```

---

h2o.month

*Convert Milliseconds to Months in H2O Datasets*


---

### Description

Converts the entries of a [H2OFrame](#) object from milliseconds to months (on a 1 to 12 scale).

### Usage

```
h2o.month(x)
```

```
month(x)
```

```
## S3 method for class H2OFrame
month(x)
```

### Arguments

x                      An [H2OFrame](#) object.

### Value

A [H2OFrame](#) object containing the entries of x converted to months of the year.

### See Also

[h2o.year](#)

---

h2o.mse

*Retrieves Mean Squared Error Value*


---

### Description

Retrieves the mean squared error value from an [H2OModelMetrics](#) object. If "train", "valid", and "xval" parameters are FALSE (default), then the training MSEvalue is returned. If more than one parameter is set to TRUE, then a named vector of MSEs are returned, where the names are "train", "valid" or "xval".

### Usage

```
h2o.mse(object, train = FALSE, valid = FALSE, xval = FALSE, ...)
```

**Arguments**

object	An <a href="#">H2OModelMetrics</a> object of the correct type.
train	Retrieve the training MSE
valid	Retrieve the validation MSE
xval	Retrieve the cross-validation MSE
...	Extra arguments to be passed if ‘object’ is of type <a href="#">H2OModel</a> (e.g. train=TRUE)

**Details**

This function only supports [H2OBinomialMetrics](#), [H2OMultinomialMetrics](#), and [H2ORegressionMetrics](#) objects.

**See Also**

[h2o.auc](#) for AUC, [h2o.mse](#) for MSE, and [h2o.metric](#) for the various threshold metrics. See [h2o.performance](#) for creating H2OModelMetrics objects.

**Examples**

```
library(h2o)
h2o.init()

prosPath <- system.file("extdata", "prostate.csv", package="h2o")
hex <- h2o.uploadFile(prosPath)

hex[,2] <- as.factor(hex[,2])
model <- h2o.gbm(x = 3:9, y = 2, training_frame = hex, distribution = "bernoulli")
perf <- h2o.performance(model, hex)
h2o.mse(perf)
```

---

h2o.naiveBayes	<i>Naive Bayes Model in H2O</i>
----------------	---------------------------------

---

**Description**

Compute naive Bayes probabilities on an H2O dataset.

**Usage**

```
h2o.naiveBayes(x, y, training_frame, model_id, laplace = 0,
  threshold = 0.001, eps = 0, compute_metrics = TRUE)
```

**Arguments**

<code>x</code>	A vector containing the names or indices of the predictor variables to use in building the model.
<code>y</code>	The name or index of the response variable. If the data does not contain a header, this is the column index number starting at 0, and increasing from left to right. The response must be a categorical variable with at least two levels.
<code>training_frame</code>	An <a href="#">H2OFrame</a> object containing the variables in the model.
<code>model_id</code>	(Optional) The unique id assigned to the resulting model. If none is given, an id will automatically be generated.
<code>laplace</code>	A positive number controlling Laplace smoothing. The default zero disables smoothing.
<code>threshold</code>	The minimum standard deviation to use for observations without enough data. Must be at least 1e-10.
<code>eps</code>	A threshold cutoff to deal with numeric instability, must be positive.
<code>compute_metrics</code>	A logical value indicating whether model metrics should be computed. Set to FALSE to reduce the runtime of the algorithm.

**Details**

The naive Bayes classifier assumes independence between predictor variables conditional on the response, and a Gaussian distribution of numeric predictors with mean and standard deviation computed from the training dataset. When building a naive Bayes classifier, every row in the training dataset that contains at least one NA will be skipped completely. If the test dataset has missing values, then those predictors are omitted in the probability calculation during prediction.

The naive Bayes classifier assumes independence between predictor variables conditional on the response, and a Gaussian distribution of numeric predictors with mean and standard deviation computed from the training dataset. When building a naive Bayes classifier, every row in the training dataset that contains at least one NA will be skipped completely. If the test dataset has missing values, then those predictors are omitted in the probability calculation during prediction.

**Value**

Returns an object of class [H2OBinomialModel](#) if the response has two categorical levels, and [H2OMultinomialModel](#) otherwise.

**Examples**

```
localH2O <- h2o.init()
votesPath <- system.file("extdata", "housevotes.csv", package="h2o")
votes.hex <- h2o.uploadFile(localH2O, path = votesPath, header = TRUE)
h2o.naiveBayes(x = 2:17, y = 1, training_frame = votes.hex, laplace = 3)
```



---

h2o.networkTest	<i>View Network Traffic Speed</i>
-----------------	-----------------------------------

---

**Description**

View speed with various file sizes.

**Usage**

```
h2o.networkTest(conn = h2o.getConnection())
```

**Arguments**

conn                    an [H2OConnection](#) object.

**Value**

Returns a table listing the network speed for 1B, 10KB, and 10MB.

---

h2o.nlevels	<i>Return the number of levels in the column.</i>
-------------	---

---

**Description**

If a frame or non-categorical column is passed, returns 0.

**Usage**

```
h2o.nlevels(object)
```

**Arguments**

object                An H2OFrame object.

---

h2o.nrow	<i>The Number of Rows/Columns of an H2O Dataset</i>
----------	---

---

**Description**

Returns a count of the number of rows or columns in an [H2OFrame](#) object.

**Usage**

```
h2o.nrow(x)

h2o.ncol(x)

## S4 method for signature H2OFrame
nrow(x)

## S4 method for signature H2OFrame
ncol(x)
```

**Arguments**

x                      An [H2OFrame](#) object.

**See Also**

[dim](#) for all the dimensions. [nrow](#) for the default R method.

**Examples**

```
library(h2o)
localH2O <- h2o.init()
irisPath <- system.file("extdata", "iris.csv", package="h2o")
iris.hex <- h2o.uploadFile(localH2O, path = irisPath)
nrow(iris.hex)
ncol(iris.hex)
```

---

h2o.null_deviance	<i>Retrieve the null deviance If "train", "valid", and "xval" parameters are FALSE (default), then the training null deviance value is returned. If more than one parameter is set to TRUE, then a named vector of null deviances are returned, where the names are "train", "valid" or "xval".</i>
-------------------	---

---

**Description**

Retrieve the null deviance If "train", "valid", and "xval" parameters are FALSE (default), then the training null deviance value is returned. If more than one parameter is set to TRUE, then a named vector of null deviances are returned, where the names are "train", "valid" or "xval".

**Usage**

```
h2o.null_deviance(object, train = FALSE, valid = FALSE, xval = FALSE, ...)
```

**Arguments**

object	An <a href="#">H2OModel</a> or <a href="#">H2OModelMetrics</a>
train	Retrieve the training null deviance
valid	Retrieve the validation null deviance
xval	Retrieve the cross-validation null deviance
...	further arguments to be passed to/from this method.

---

h2o.null_dof	<i>Retrieve the null degrees of freedom If "train", "valid", and "xval" parameters are FALSE (default), then the training null degrees of freedom value is returned. If more than one parameter is set to TRUE, then a named vector of null degrees of freedom are returned, where the names are "train", "valid" or "xval".</i>
--------------	--

---

**Description**

Retrieve the null degrees of freedom If "train", "valid", and "xval" parameters are FALSE (default), then the training null degrees of freedom value is returned. If more than one parameter is set to TRUE, then a named vector of null degrees of freedom are returned, where the names are "train", "valid" or "xval".

**Usage**

```
h2o.null_dof(object, train = FALSE, valid = FALSE, xval = FALSE, ...)
```

**Arguments**

object	An <a href="#">H2OModel</a> or <a href="#">H2OModelMetrics</a>
train	Retrieve the training null degrees of freedom
valid	Retrieve the validation null degrees of freedom
xval	Retrieve the cross-validation null degrees of freedom
...	further arguments to be passed to/from this method.

---

h2o.num_iterations	<i>Retrieve the number of iterations.</i>
--------------------	---

---

**Description**

Retrieve the number of iterations.

**Usage**

```
h2o.num_iterations(object)
```

**Arguments**

object	An <a href="#">H2OClusteringModel</a> object.
...	further arguments to be passed on (currently unimplemented)

---

h2o.openLog	<i>View H2O R Logs</i>
-------------	------------------------

---

**Description**

Open existing logs of H2O R POST commands and error responses on local disk. Used primarily for debugging purposes.

**Usage**

```
h2o.openLog(type)
```

**Arguments**

type	Currently unimplemented.
------	--------------------------

**See Also**

[h2o.startLogging](#), [h2o.stopLogging](#), [h2o.clearLog](#)

**Examples**

```
localH2O = h2o.init()

h2o.startLogging()
ausPath = system.file("extdata", "australia.csv", package="h2o")
australia.hex = h2o.importFile(localH2O, path = ausPath)
h2o.stopLogging()
```

```
# Not run to avoid windows being opened during R CMD check
# h2o.openLog("Command")
# h2o.openLog("Error")
```

h2o.parseRaw

*H2O Data Parsing*

## Description

The second phase in the data ingestion step.

## Usage

```
h2o.parseRaw(data, destination_frame = "", header = NA, sep = "",
  col.names = NULL, col.types = NULL, na.strings = NULL,
  blocking = FALSE, parse_type = NULL)
```

## Arguments

data	An <a href="#">H2ORawData</a> object to be parsed.
destination_frame	(Optional) The hex key assigned to the parsed file.
header	(Optional) A logical value indicating whether the first row is the column header. If missing, H2O will automatically try to detect the presence of a header.
sep	(Optional) The field separator character. Values on each line of the file are separated by this character. If sep = "", the parser will automatically detect the separator.
col.names	(Optional) A <a href="#">H2OFrame</a> object containing a single delimited line with the column names for the file.
col.types	(Optional) A vector specifying the types to attempt to force over columns.
na.strings	(Optional) H2O will interpret these strings as missing.
blocking	(Optional) Tell H2O parse call to block synchronously instead of polling. This can be faster for small datasets but loses the progress bar.
parse_type	(Optional) Specify which parser type H2O will use. Valid types are "ARFF", "XLS", "CSV", "SVMLight"

## Details

Parse the Raw Data produced by the import phase.

---

h2o.parseSetup	<i>Get a parse setup back for the staged data.</i>
----------------	--

---

### Description

Get a parse setup back for the staged data.

### Usage

```
h2o.parseSetup(data, destination_frame = "", header = NA, sep = "",
  col.names = NULL, col.types = NULL, na.strings = NULL,
  parse_type = NULL)
```

### Arguments

data	An <a href="#">H2ORawData</a> object to be parsed.
destination_frame	(Optional) The hex key assigned to the parsed file.
header	(Optional) A logical value indicating whether the first row is the column header. If missing, H2O will automatically try to detect the presence of a header.
sep	(Optional) The field separator character. Values on each line of the file are separated by this character. If sep = "", the parser will automatically detect the separator.
col.names	(Optional) A <a href="#">H2OFrame</a> object containing a single delimited line with the column names for the file.
col.types	(Optional) A vector specifying the types to attempt to force over columns.
na.strings	(Optional) H2O will interpret these strings as missing.
parse_type	(Optional) Specify which parser type H2O will use. Valid types are "ARFF", "XLS", "CSV", "SVMLight"

---

h2o.performance	<i>Model Performance Metrics in H2O</i>
-----------------	---

---

### Description

Given a trained h2o model, compute its performance on the given dataset

### Usage

```
h2o.performance(model, data = NULL, valid = FALSE, ...)
```

**Arguments**

model	An <a href="#">H2OModel</a> object
data	An <a href="#">H2OFrame</a> . The model will make predictions on this dataset, and subsequently score them. The dataset should match the dataset that was used to train the model, in terms of column names, types, and dimensions. If data is passed in, then train and valid are ignored.
valid	A logical value indicating whether to return the validation metrics (constructed during training).
...	Extra args passed in for use by other functions.

**Value**

Returns an object of the [H2OModelMetrics](#) subclass.

**Examples**

```
library(h2o)
localH2O <- h2o.init()
prosPath <- system.file("extdata", "prostate.csv", package="h2o")
prostate.hex <- h2o.uploadFile(localH2O, path = prosPath)
prostate.hex$CAPSULE <- as.factor(prostate.hex$CAPSULE)
prostate.gbm <- h2o.gbm(3:9, "CAPSULE", prostate.hex)
h2o.performance(model = prostate.gbm, data=prostate.hex)
```

h2o.prcomp

*Principal Components Analysis***Description**

Principal components analysis of a H2O dataset using the power method to calculate the singular value decomposition of the Gram matrix.

**Usage**

```
h2o.prcomp(training_frame, x, k, model_id, max_iterations = 1000,
  transform = c("NONE", "DEMEAN", "DESCALE", "STANDARDIZE"),
  pca_method = c("GramSVD", "Power", "GLRM"), seed, use_all_factor_levels)
```

**Arguments**

training_frame	An <a href="#">H2OFrame</a> object containing the variables in the model.
x	(Optional) A vector containing the data columns on which SVD operates.
k	The number of principal components to be computed. This must be between 1 and min(ncol(training_frame), nrow(training_frame)) inclusive.

model_id	(Optional) The unique hex key assigned to the resulting model. Automatically generated if none is provided.
max_iterations	The maximum number of iterations to run each power iteration loop. Must be between 1 and 1e6 inclusive.
transform	A character string that indicates how the training data should be transformed before running PCA. Possible values are "NONE": for no transformation, "DE-MEAN": for subtracting the mean of each column, "DESCALE": for dividing by the standard deviation of each column, "STANDARDIZE": for demeaning and dscaling, and "NORMALIZE": for demeaning and dividing each column by its range (max - min).
pca_method	A character string that indicates how PCA should be calculated. Possible values are "GramSVD": distributed computation of the Gram matrix followed by a local SVD using the JAMA package, "Power": computation of the SVD using the power iteration method, "GLRM": fit a generalized low rank model with an l2 loss function (no regularization) and solve for the SVD using local matrix algebra.
seed	(Optional) Random seed used to initialize the right singular vectors at the beginning of each power method iteration.
use_all_factor_levels	(Optional) A logical value indicating whether all factor levels should be included in each categorical column expansion. If FALSE, the indicator column corresponding to the first factor level of every categorical variable will be dropped. Defaults to FALSE.

### Value

Returns an object of class [H2ODimReductionModel](#).

### See Also

[h2o.svd](#), [h2o.glrm](#)

### Examples

```
library(h2o)
localH2O <- h2o.init()
ausPath <- system.file("extdata", "australia.csv", package="h2o")
australia.hex <- h2o.uploadFile(localH2O, path = ausPath)
h2o.prcomp(training_frame = australia.hex, k = 8, transform = "STANDARDIZE")
```

---

h2o.quantile

*Quantiles of H2O Data Frame.*


---

### Description

Obtain and display quantiles for H2O parsed data.



**Usage**

```
h2o.quantile(x, probs = c(0.001, 0.01, 0.1, 0.25, 0.333, 0.5, 0.667, 0.75,
  0.9, 0.99, 0.999), combine_method = c("interpolate", "average", "avg",
  "low", "high"), ...)
```

**Arguments**

x	An <a href="#">H2OFrame</a> object with a single numeric column.
probs	Numeric vector of probabilities with values in [0,1].
combine_method	How to combine quantiles for even sample sizes. Default is to do linear interpolation. E.g., If method is "lo", then it will take the lo value of the quantile. Abbreviations for average, low, and high are acceptable (avg, lo, hi).
...	Further arguments passed to or from other methods.

**Details**

quantile.H2OFrame, a method for the [quantile](#) generic. Obtain and return quantiles for an [H2OFrame](#) object.

**Value**

A vector describing the percentiles at the given cutoffs for the [H2OFrame](#) object.

**Examples**

```
# Request quantiles for an H2O parsed data set:
library(h2o)
localH2O <- h2o.init()
prosPath <- system.file("extdata", "prostate.csv", package="h2o")
prostate.hex <- h2o.uploadFile(localH2O, path = prosPath)
# Request quantiles for a subset of columns in an H2O parsed data set
quantile(prostate.hex[,3])
for(i in 1:ncol(prostate.hex))
  quantile(prostate.hex[,i])
```

---

h2o.r2

---

*Retrieve the R2 value*


---

**Description**

Retrieves the R2 value from an H2O model. If "train", "valid", and "xval" parameters are FALSE (default), then the training R2 value is returned. If more than one parameter is set to TRUE, then a named vector of R2s are returned, where the names are "train", "valid" or "xval".

**Usage**

```
h2o.r2(object, train = FALSE, valid = FALSE, xval = FALSE, ...)
```

**Arguments**

object	An <a href="#">H2OModel</a> object.
train	Retrieve the training R2
valid	Retrieve the validation R2
xval	Retrieve the cross-validation R2
...	extra arguments to be passed if 'object' is of type <a href="#">H2OModel</a> (e.g. train=TRUE)

**Examples**

```
library(h2o)

h <- h2o.init()
fr <- as.h2o(iris)

m <- h2o.deeplearning(x=2:5,y=1,training_frame=fr)

h2o.r2(m)
```

---

h2o.randomForest	<i>Build a Big Data Random Forest Model</i>
------------------	---

---

**Description**

Builds a Random Forest Model on an [H2OFrame](#)

**Usage**

```
h2o.randomForest(x, y, training_frame, model_id, validation_frame, checkpoint,
  mtries = -1, sample_rate = 0.632, build_tree_one_node = FALSE,
  ntrees = 50, max_depth = 20, min_rows = 1, nbins = 20,
  nbins_cats = 1024, binomial_double_trees = FALSE,
  balance_classes = FALSE, max_after_balance_size = 5, seed,
  offset_column = NULL, weights_column = NULL, nfolds = 0,
  fold_column = NULL, fold_assignment = c("AUTO", "Random", "Modulo"),
  keep_cross_validation_predictions = FALSE, ...)
```

**Arguments**

x	A vector containing the names or indices of the predictor variables to use in building the GBM model.
y	The name or index of the response variable. If the data does not contain a header, this is the column index number starting at 1, and increasing from left to right. (The response must be either an integer or a categorical variable).
training_frame	An <a href="#">H2OFrame</a> object containing the variables in the model.
model_id	(Optional) The unique id assigned to the resulting model. If none is given, an id will automatically be generated.
validation_frame	An <a href="#">H2OFrame</a> object containing the variables in the model.
checkpoint	"Model checkpoint (either key or H2ODeepLearningModel) to resume training with."
mtries	Number of variables randomly sampled as candidates at each split. If set to -1, defaults to sqrt(p) for classification, and p/3 for regression, where p is the number of predictors.
sample_rate	Sample rate, from 0 to 1.0.
build_tree_one_node	Run on one node only; no network overhead but fewer cpus used. Suitable for small datasets.
ntrees	A nonnegative integer that determines the number of trees to grow.
max_depth	Maximum depth to grow the tree.
min_rows	Minimum number of rows to assign to terminal nodes.
nbins	For numerical columns (real/int), build a histogram of this many bins, then split at the best point.
nbins_cats	For categorical columns (enum), build a histogram of this many bins, then split at the best point. Higher values can lead to more overfitting.
binomial_double_trees	For binary classification: Build 2x as many trees (one per class) - can lead to higher accuracy.
balance_classes	logical, indicates whether or not to balance training data class counts via over/under-sampling (for imbalanced data)
max_after_balance_size	Maximum relative size of the training data after balancing class counts (can be less than 1.0)
seed	Seed for random numbers (affects sampling) - Note: only reproducible when running single threaded
offset_column	Specify the offset column.
weights_column	Specify the weights column.
nfolds	(Optional) Number of folds for cross-validation. If nfolds >= 2, then validation must remain empty.

fold\_column (Optional) Column with cross-validation fold index assignment per observation  
 fold\_assignment Cross-validation fold assignment scheme, if fold\_column is not specified Must be "AUTO", "Random" or "Modulo"  
 keep\_cross\_validation\_predictions Whether to keep the predictions of the cross-validation models  
 ... (Currently Unimplemented)

### Value

Creates a [H2OModel](#) object of the right type.

### See Also

[predict.H2OModel](#) for prediction.

---

h2o.rbind	<i>Combine H2O Datasets by Rows</i>
-----------	-------------------------------------

---

### Description

Takes a sequence of H2O data sets and combines them by rows

### Usage

```
h2o.rbind(...)
```

### Arguments

... A sequence of [H2OFrame](#) arguments. All datasets must exist on the same H2O instance (IP and port) and contain the same number of rows.

### Value

An [H2OFrame](#) object containing the combined ... arguments column-wise.

### See Also

[rbind](#) for the base R method.

### Examples

```

library(h2o)
localH2O <- h2o.init()
prosPath <- system.file("extdata", "prostate.csv", package="h2o")
prostate.hex <- h2o.uploadFile(localH2O, path = prosPath)
prostate.cbind <- h2o.rbind(prostate.hex, prostate.hex)
head(prostate.cbind)

```

---

h2o.removeAll	<i>Remove All Objects on the H2O Cluster</i>
---------------	--

---

**Description**

Removes the data from the h2o cluster, but does not remove the local references.

**Usage**

```
h2o.removeAll(conn = h2o.getConnection(), timeout_secs = 0)
```

**Arguments**

conn	An <a href="#">H2OConnection</a> object containing the IP address and port number of the H2O server.
timeout_secs	Timeout in seconds. Default is no timeout.

**See Also**

[h2o.rm](#)

**Examples**

```
library(h2o)
localH2O <- h2o.init()
prosPath <- system.file("extdata", "prostate.csv", package = "h2o")
prostate.hex <- h2o.uploadFile(localH2O, path = prosPath)
h2o.ls(localH2O)
h2o.removeAll(localH2O)
h2o.ls(localH2O)
```

---

h2o.removeVecs	<i>Delete Columns from a H2OFrame</i>
----------------	---------------------------------------

---

**Description**

Delete the specified columns from the H2OFrame. Returns a H2OFrame without the specified columns. This will trigger any lazy computation of the frame, and has side-effects.

**Usage**

```
h2o.removeVecs(data, cols)
```

Arguments

- data                   The H2OFrame.
- cols                   The columns to remove.

---

h2o.rep_len	<i>Replicate Elements of Vectors or Lists into H2O</i>
-------------	--

---

Description

h2o.rep performs just as rep does. It replicates the values in x in the H2O backend.

Usage

```
h2o.rep_len(x, length.out)
```

Arguments

- x                   a vector (of any mode including a list) or a factor
- length.out          non negative integer. The desired length of the output vector.

Value

Creates a [H2OFrame](#) vector of the same type as x

---

h2o.residual_deviance	<i>Retrieve the residual deviance If "train", "valid", and "xval" parameters are FALSE (default), then the training residual deviance value is returned. If more than one parameter is set to TRUE, then a named vector of residual deviances are returned, where the names are "train", "valid" or "xval".</i>
-----------------------	---

---

Description

Retrieve the residual deviance If "train", "valid", and "xval" parameters are FALSE (default), then the training residual deviance value is returned. If more than one parameter is set to TRUE, then a named vector of residual deviances are returned, where the names are "train", "valid" or "xval".

Usage

```
h2o.residual_deviance(object, train = FALSE, valid = FALSE, xval = FALSE, ...)
```

**Arguments**

object	An <a href="#">H2OModel</a> or <a href="#">H2OModelMetrics</a>
train	Retrieve the training residual deviance
valid	Retrieve the validation residual deviance
xval	Retrieve the cross-validation residual deviance
...	further arguments to be passed to/from this method.

---

h2o.residual_dof	<i>Retrieve the residual degrees of freedom If "train", "valid", and "xval" parameters are FALSE (default), then the training residual degrees of freedom value is returned. If more than one parameter is set to TRUE, then a named vector of residual degrees of freedom are returned, where the names are "train", "valid" or "xval".</i>
------------------	--

---

**Description**

Retrieve the residual degrees of freedom If "train", "valid", and "xval" parameters are FALSE (default), then the training residual degrees of freedom value is returned. If more than one parameter is set to TRUE, then a named vector of residual degrees of freedom are returned, where the names are "train", "valid" or "xval".

**Usage**

```
h2o.residual_dof(object, train = FALSE, valid = FALSE, xval = FALSE, ...)
```

**Arguments**

object	An <a href="#">H2OModel</a> or <a href="#">H2OModelMetrics</a>
train	Retrieve the training residual degrees of freedom
valid	Retrieve the validation residual degrees of freedom
xval	Retrieve the cross-validation residual degrees of freedom
...	further arguments to be passed to/from this method.

---

h2o.rm

Delete Objects In H2O

---

### Description

Remove the h2o Big Data object(s) having the key name(s) from ids.

### Usage

```
h2o.rm(ids, conn = h2o.getConnection())
```

### Arguments

ids	The hex key associated with the object to be removed.
conn	An <a href="#">H2OConnection</a> object containing the IP address and port number of the H2O server.

### See Also

[h2o.assign](#), [h2o.ls](#)

---

h2o.runif

Produce a Vector of Random Uniform Numbers

---

### Description

Creates a vector of random uniform numbers equal in length to the length of the specified H2O dataset.

### Usage

```
h2o.runif(x, seed = -1)
```

### Arguments

x	An <a href="#">H2OFrame</a> object.
seed	A random seed used to generate draws from the uniform distribution.

### Value

A vector of random, uniformly distributed numbers. The elements are between 0 and 1.



## Examples

```
library(h2o)
localH2O = h2o.init()
prosPath = system.file("extdata", "prostate.csv", package="h2o")
prostate.hex = h2o.importFile(localH2O, path = prosPath, destination_frame = "prostate.hex")
s = h2o.runif(prostate.hex)
summary(s)

prostate.train = prostate.hex[s <= 0.8,]
prostate.train = h2o.assign(prostate.train, "prostate.train")
prostate.test = prostate.hex[s > 0.8,]
prostate.test = h2o.assign(prostate.test, "prostate.test")
nrow(prostate.train) + nrow(prostate.test)
```

---

**h2o.saveModel***Save an H2O Model Object to Disk*

---

## Description

Save an [H2OModel](#) to disk.

## Usage

```
h2o.saveModel(object, path = "", force = FALSE)
```

## Arguments

object	an <a href="#">H2OModel</a> object.
path	string indicating the directory the model will be written to.
force	logical, indicates how to deal with files that already exist.

## Details

In the case of existing files `force = TRUE` will overwrite the file. Otherwise, the operation will fail.

## See Also

[h2o.loadModel](#) for loading a model to H2O from disk

## Examples

```
# library(h2o)
# localH2O <- h2o.init()
# prostate.hex <- h2o.importFile(localH2O, path = paste("https://raw.githubusercontent.com",
#   "h2oai/h2o-2/master/smalldata/logreg/prostate.csv", sep = "/"),
#   destination_frame = "prostate.hex")
# prostate.glm <- h2o.glm(y = "CAPSULE", x = c("AGE", "RACE", "PSA", "DCAPS"),
#   training_frame = prostate.hex, family = "binomial", alpha = 0.5)
# h2o.saveModel(object = prostate.glm, path = "/Users/UserName/Desktop", force=TRUE)
```

---

h2o.scale

*Scaling and Centering of an H2O Frame*


---

## Description

Centers and/or scales the columns of an H2O dataset.

## Usage

```
h2o.scale(x, center = TRUE, scale = TRUE)
```

## Arguments

x	An <a href="#">H2OFrame</a> object.
center	either a logical value or numeric vector of length equal to the number of columns of x.
scale	either a logical value or numeric vector of length equal to the number of columns of x.

## Examples

```
library(h2o)
localH2O <- h2o.init()
irisPath <- system.file("extdata", "iris_wheader.csv", package="h2o")
iris.hex <- h2o.uploadFile(localH2O, path = irisPath, destination_frame = "iris.hex")
summary(iris.hex)

# Scale and center all the numeric columns in iris data set
scale(iris.hex[, 1:4])
```

---

h2o.scoreHistory	<i>Retrieve Model Score History</i>
------------------	-------------------------------------

---

**Description**

Retrieve Model Score History

**Usage**

```
h2o.scoreHistory(object, ...)
```

**Arguments**

object	An <a href="#">H2OModel</a> object.
...	further arguments to be passed on (currently unimplemented)

---

h2o.sd	<i>Standard Deviation of a column of data.</i>
--------	--

---

**Description**

Obtain the standard deviation of a column of data.

**Usage**

```
h2o.sd(x, na.rm = FALSE)

## S4 method for signature 'H2OFrame'
sd(x, na.rm = FALSE)
```

**Arguments**

x	An <a href="#">H2OFrame</a> object.
na.rm	logical. Should missing values be removed?

**See Also**

[h2o.var](#) for variance, and [sd](#) for the base R implementation.

**Examples**

```
localH2O <- h2o.init()
prosPath <- system.file("extdata", "prostate.csv", package="h2o")
prostate.hex <- h2o.uploadFile(localH2O, path = prosPath)
sd(prostate.hex$AGE)
```

---

h2o.sdev	<i>Retrieve the standard deviations of principal components</i>
----------	---

---

**Description**

Retrieve the standard deviations of principal components

**Usage**

h2o.sdev(object)

**Arguments**

object            An [H2ODimReductionModel](#) object.

---

h2o.setLevel	<i>Set a Factor Column to a Level</i>
--------------	---------------------------------------

---

**Description**

A method to set a factor column to one of the levels.

**Usage**

h2o.setLevel(x, level)

**Arguments**

x                    a column from an [H2OFrame](#) object.  
level                The level at which the column will be set.

**Details**

Replace all other occurrences with ‘level’ in a factor column.

**Value**

An object of class [H2OFrame](#).

**Examples**

```
localH2O <- h2o.init()
hex <- as.h2o(localH2O , iris)
hex$Species <- h2o.setLevel(hex$Species, "versicolor")
```

---

h2o.setLevels	<i>Set Levels of H2O Factor Column</i>
---------------	--

---

**Description**

Works on a single categorical vector. New domains must be aligned with the old domains. This call has SIDE EFFECTS and mutates the column in place (does not make a copy).

**Usage**

```
h2o.setLevels(x, levels)
```

**Arguments**

x	A single categorical column.
levels	A character vector specifying the new levels. The number of new levels must match the number of old levels.

---

h2o.setTimezone	<i>Set the Time Zone on the H2O Cloud</i>
-----------------	---

---

**Description**

Set the Time Zone on the H2O Cloud

**Usage**

```
h2o.setTimezone(tz, conn = h2o.getConnection())
```

**Arguments**

tz	The desired timezone.
conn	An H2OConnection object.

h2o.shim

*Deprecated Script Shim***Description**

Due to the many improvements implemented in H2O-Dev and the differences in architecture between H2O and H2O-Dev, some parameters, options, and objects are no longer supported. To assist our legacy H2O users in upgrading their workflows for compatibility with H2O-Dev, we have developed the "Deprecated Script Shim" tool to detect deprecated parameters, options, and objects in H2O scripts being imported into H2O-Dev and suggest updated alternatives.

**Usage**

```
h2o.shim(enable = TRUE)
```

**Arguments**

enable                    a logical value indicating whether the shim should be enabled or disabled.

**See Also**

<https://github.com/h2oai/h2o-dev/blob/master/h2o-docs/src/product/upgrade/H2ODevPortingRScripts.md>, For more information on converting legacy H2O scripts so that they will run in H2O-Dev

h2o.shutdown

*Shut Down H2O Instance***Description**

Shut down the specified instance. All data will be lost.

**Usage**

```
h2o.shutdown(conn = h2o.getConnection(), prompt = TRUE)
```

**Arguments**

conn                    An [H2OConnection](#) object containing the IP address and port of the server running H2O.

prompt                  A logical value indicating whether to prompt the user before shutting down the H2O server.

**Details**

This method checks if H2O is running at the specified IP address and port, and if it is, shuts down that H2O instance.

**WARNING**

All data, models, and other values stored on the server will be lost! Only call this function if you and all other clients connected to the H2O server are finished and have saved your work.

**Note**

Users must call `h2o.shutdown` explicitly in order to shut down the local H2O instance started by R. If R is closed before H2O, then an attempt will be made to automatically shut down H2O. This only applies to local instances started with `h2o.init`, not remote H2O servers.

**See Also**

[h2o.init](#)

**Examples**

```
# Dont run automatically to prevent accidentally shutting down a cloud
## Not run:
library(h2o)
localH2O = h2o.init()
h2o.shutdown(localH2O)

## End(Not run)
```

---

h2o.splitFrame

---

*Split an H2O Data Set*


---

**Description**

Split an existing H2O data set according to user-specified ratios.

**Usage**

```
h2o.splitFrame(data, ratios = 0.75, destination_frames)
```

**Arguments**

data	An <a href="#">H2OFrame</a> object representing the dataste to split.
ratios	A numeric value or array indicating the ratio of total rows contained in each split. Must total up to less than 1.
destination_frames	An array of frame IDs equal to the number of ratios specified plus one.

## Examples

```
library(h2o)
localH2O = h2o.init()
irisPath = system.file("extdata", "iris.csv", package = "h2o")
iris.hex = h2o.importFile(localH2O, path = irisPath)
iris.split = h2o.splitFrame(iris.hex, ratios = c(0.2, 0.5))
head(iris.split[[1]])
summary(iris.split[[1]])
```

---

h2o.startGLMJob

*Start an H2O Generalized Linear Model Job*


---

## Description

Creates a background H2O GLM job.

## Usage

```
h2o.startGLMJob(x, y, training_frame, model_id, validation_frame,
  max_iterations = 50, beta_epsilon = 0, solver = c("IRLSM", "L_BFGS"),
  standardize = TRUE, family = c("gaussian", "binomial", "poisson", "gamma",
  "tweedie"), link = c("family_default", "identity", "logit", "log",
  "inverse", "tweedie"), tweedie_variance_power = NaN,
  tweedie_link_power = NaN, alpha = 0.5, prior = 0, lambda = 1e-05,
  lambda_search = FALSE, nlambda = -1, lambda_min_ratio = 1,
  nfolds = 0, beta_constraints = NULL, ...)
```

## Arguments

- |                  |   |
|------------------|---|
| x                | A vector containing the names or indices of the predictor variables to use in building the GLM model.   |
| y                | A character string or index that represent the response variable in the model.  |
| training_frame   | An <a href="#">H2OFrame</a> object containing the variables in the model.   |
| model_id         | (Optional) The unique id assigned to the resulting model. If none is given, an id will automatically be generated.  |
| validation_frame | An <a href="#">H2OFrame</a> object containing the variables in the model.   |
| max_iterations   | A non-negative integer specifying the maximum number of iterations.   |
| beta_epsilon     | A non-negative number specifying the magnitude of the maximum difference between the coefficient estimates from successive iterations. Defines the convergence criterion for h2o.glm. |
| solver           | A character string specifying the solver used: IRLSM (supports more features), L_BFGS (scales better for datasets with many columns)  |



standardize	A logical value indicating whether the numeric predictors should be standardized to have a mean of 0 and a variance of 1 prior to training the models.
family	A character string specifying the distribution of the model: gaussian, binomial, poisson, gamma, tweedie.
link	A character string specifying the link function. The default is the canonical link for the family. The supported links for each of the family specifications are: "gaussian": "identity", "log", "inverse" "binomial": "logit", "log" "poisson": "log", "identity" "gamma": "inverse", "log", "identity" "tweedie": "tweedie"
tweedie_variance_power	A numeric specifying the power for the variance function when family = "tweedie".
tweedie_link_power	A numeric specifying the power for the link function when family = "tweedie".
alpha	A numeric in [0, 1] specifying the elastic-net mixing parameter. The elastic-net penalty is defined to be: $P(\alpha, \beta) = (1 - \alpha)/2   \beta  _2^2 + \alpha   \beta  _1 = \sum_j [(1 - \alpha)/2 \beta_j^2 + \alpha  \beta_j ]$ , making alpha = 1 the lasso penalty and alpha = 0 the ridge penalty.
prior	(Optional) A numeric specifying the prior probability of class 1 in the response when family = "binomial". The default prior is the observational frequency of class 1.
lambda	A non-negative shrinkage parameter for the elastic-net, which multiplies $P(\alpha, \beta)$ in the objective function. When lambda = 0, no elastic-net penalty is applied and ordinary generalized linear models are fit.
lambda_search	A logical value indicating whether to conduct a search over the space of lambda values starting from the lambda max, given lambda is interpreted as lambda min.
nlambdas	The number of lambda values to use when lambda_search = TRUE.
lambda_min_ratio	Smallest value for lambda as a fraction of lambda.max. By default if the number of observations is greater than the the number of variables then lambda_min_ratio = 0.0001; if the number of observations is less than the number of variables then lambda_min_ratio = 0.01.
nfolds	(Optional) Number of folds for cross-validation. If nfolds >= 2, then validation must remain empty.
beta_constraints	A data.frame or H2OParsedData object with the columns ["names", "lower_bounds", "upper_bounds", "beta_given"], where each row corresponds to a predictor in the GLM. "names" contains the predictor names, "lower_bounds" and "upper_bounds" are the lower and upper bounds of beta, and "beta_given" is some supplied starting values for beta.
...	(Currently Unimplemented) coefficients.

**Value**

Returns a [H2OModelFuture](#) class object.

---

h2o.startLogging	<i>Start Writing H2O R Logs</i>
------------------	---------------------------------

---

**Description**

Begin logging H2o R POST commands and error responses to local disk. Used primarily for debugging purposes.

**Usage**

```
h2o.startLogging(file)
```

**Arguments**

file	a character string name for the file, automatically generated
------	---

**See Also**

[h2o.stopLogging](#), [h2o.clearLog](#), [h2o.openLog](#)

**Examples**

```
library(h2o)
localH2O = h2o.init()
h2o.startLogging()
ausPath = system.file("extdata", "australia.csv", package="h2o")
australia.hex = h2o.importFile(localH2O, path = ausPath)
h2o.stopLogging()
```

---

h2o.stopLogging	<i>Stop Writing H2O R Logs</i>
-----------------	--------------------------------

---

**Description**

Halt logging of H2O R POST commands and error responses to local disk. Used primarily for debugging purposes.

**Usage**

```
h2o.stopLogging()
```

See Also

[h2o.startLogging](#), [h2o.clearLog](#), [h2o.openLog](#)

Examples

```
library(h2o)
localH2O = h2o.init()
h2o.startLogging()
ausPath = system.file("extdata", "australia.csv", package="h2o")
australia.hex = h2o.importFile(localH2O, path = ausPath)
h2o.stopLogging()
```

---

h2o.strsplit	<i>String Split</i>
--------------	---------------------

---

Description

String Split

Usage

```
h2o.strsplit(x, split)
```

Arguments

- |       |   |
|-------|---|
| x     | The column whose strings must be split. |
| split | The pattern to split on.                |

---

h2o.sub	<i>String Substitute</i>
---------	--------------------------

---

Description

Mutates the input. Changes the first occurence of pattern with replacement.

Usage

```
h2o.sub(pattern, replacement, x, ignore.case = FALSE)
```

Arguments

- |             |                                 |
|-------------|---------------------------------|
| pattern     | The pattern to replace.         |
| replacement | The replacement pattern.        |
| x           | The column on which to operate. |
| ignore.case | Case sensitive or not           |

h2o.subset

*Subsetting an H2O Frame*

---

**Description**

Returns a subset of an [H2OFrame](#) which meets conditions.

**Usage**

```
h2o.subset(x, subset, select, drop = FALSE, ...)
```

**Arguments**

x	a <a href="#">H2OFrame</a> to be subsetting
subset	logical expression indicating elements or rows to keep
select	expression, indicating columns to select from a data frame
drop	passed on the the <code>[]</code> indexing operator
...	further arguments to be passed to or from other methods

**See Also**

For the base R implementation see [subset](#)

---

h2o.summary*Summarizes the columns of a H2O data frame.*

---

**Description**

A method for the [summary](#) generic. Summarizes the columns of an H2O data frame or subset of columns and rows using vector notation (e.g. `dataset[row, col]`)

**Usage**

```
## S4 method for signature H2OFrame  
summary(object, factors = 6L, ...)
```

**Arguments**

object	An <a href="#">H2OFrame</a> object.
factors	The number of factors to return in the summary. Default is the top 6.
...	Further arguments passed to or from other methods.

**Value**

A table displaying the minimum, 1st quartile, median, mean, 3rd quartile and maximum for each numeric column, and the levels and category counts of the levels in each categorical column.

**Examples**

```
library(h2o)
localH2O = h2o.init()
prosPath = system.file("extdata", "prostate.csv", package="h2o")
prostate.hex = h2o.importFile(localH2O, path = prosPath)
summary(prostate.hex)
summary(prostate.hex$GLEASON)
summary(prostate.hex[,4:6])
```

---

h2o.svd	<i>Singular Value Decomposition</i>
---------	-------------------------------------

---

**Description**

Singular value decomposition of a H2O dataset using the power method.

**Usage**

```
h2o.svd(training_frame, x, nv, destination_key, max_iterations = 1000,
  transform = "NONE", svd_method = c("GramSVD", "Power", "Randomized"),
  seed, use_all_factor_levels)
```

**Arguments**

training_frame	An <a href="#">H2OFrame</a> object containing the variables in the model.
x	(Optional) A vector containing the data columns on which SVD operates.
nv	The number of right singular vectors to be computed. This must be between 1 and min(ncol(training_frame), nrow(training_frame)) inclusive.
destination_key	(Optional) The unique hex key assigned to the resulting model. Automatically generated if none is provided.
max_iterations	The maximum number of iterations to run each power iteration loop. Must be between 1 and 1e6 inclusive.
transform	A character string that indicates how the training data should be transformed before running PCA. Possible values are: "NONE" for no transformation; "DE-MEAN" for subtracting the mean of each column; "DESCALE" for dividing by the standard deviation of each column; "STANDARDIZE" for demeaning and dscaling; and "NORMALIZE" for demeaning and dividing each column by its range (max - min).

svd_method	A character string that indicates how SVD should be calculated. Possible values are "GramSVD": distributed computation of the Gram matrix followed by a local SVD using the JAMA package, "Power": computation of the SVD using the power iteration method, "Randomized": approximate SVD by projecting onto a random subspace (see references).
seed	(Optional) Random seed used to initialize the right singular vectors at the beginning of each power method iteration.
use_all_factor_levels	(Optional) A logical value indicating whether all factor levels should be included in each categorical column expansion. If FALSE, the indicator column corresponding to the first factor level of every categorical variable will be dropped. Defaults to TRUE.

**Value**

Returns an object of class [H2ODimReductionModel](#).

**References**

N. Halko, P.G. Martinsson, J.A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions[<http://arxiv.org/abs/0909.4061>]. SIAM Rev., Survey and Review section, Vol. 53, num. 2, pp. 217-288, June 2011.

**Examples**

```
library(h2o)
localH2O <- h2o.init()
ausPath <- system.file("extdata", "australia.csv", package="h2o")
australia.hex <- h2o.uploadFile(localH2O, path = ausPath)
h2o.svd(training_frame = australia.hex, nv = 8)
```

---

h2o.table	<i>Cross Tabulation and Table Creation in H2O</i>
-----------	---

---

**Description**

Uses the cross-classifying factors to build a table of counts at each combination of factor levels.

**Usage**

```
h2o.table(x, y = NULL)
```

**Arguments**

- x An [H2OFrame](#) object with at most two columns.
- y An [H2OFrame](#) similar to x, or NULL.

**Value**

Returns a tabulated [H2OFrame](#) object.

**Examples**

```
library(h2o)
localH2O <- h2o.init()
prosPath <- system.file("extdata", "prostate.csv", package="h2o")
prostate.hex <- h2o.uploadFile(localH2O, path = prosPath, destination_frame = "prostate.hex")
summary(prostate.hex)

# Counts of the ages of all patients
head(h2o.table(prostate.hex[,3]))
h2o.table(prostate.hex[,3])

# Two-way table of ages (rows) and race (cols) of all patients
head(h2o.table(prostate.hex[,c(3,4)]))
h2o.table(prostate.hex[,c(3,4)])
```

---

h2o.tabulate

---

*Tabulation between Two Columns of a H2O Frame*


---

**Description**

Simple Co-Occurrence based tabulation of X vs Y, where X and Y are two Vectors in a given dataset. Uses histogram of given resolution in X and Y. Handles numerical/categorical data and missing values. Supports observation weights.

**Usage**

```
h2o.tabulate(data, x, y, weights_column = NULL, nbins_x = 50,
  nbins_y = 50)
```

**Arguments**

data	An <a href="#">H2OFrame</a> object.
x	predictor column
y	response column
weights_column	(optional) observation weights column
nbins_x	number of bins for predictor column
nbins_y	number of bins for response column

**Value**

Returns two TwoDimTables of 3 columns each count\_table: X Y counts response\_table: X mean Y counts

Examples

```
library(h2o)
localH2O <- h2o.init()
df <- as.h2o(iris)
h2o.tabulate(data = df, x = "Sepal.Length", y = "Petal.Width",
             weights_column = NULL, nbins_x = 10, nbins_y = 10)
```

---

h2o.tolower	To Lower
-------------	----------

---

Description

Mutates the input!

Usage

```
h2o.tolower(x)
```

Arguments

x                    An H2OFrame object whose strings should be lower'd

---

h2o.totss	<i>Get the total sum of squares. If "train", "valid", and "xval" parameters are FALSE (default), then the training totss value is returned. If more than one parameter is set to TRUE, then a named vector of totss' are returned, where the names are "train", "valid" or "xval".</i>
-----------	--

---

Description

Get the total sum of squares. If "train", "valid", and "xval" parameters are FALSE (default), then the training totss value is returned. If more than one parameter is set to TRUE, then a named vector of totss' are returned, where the names are "train", "valid" or "xval".

Usage

```
h2o.totss(object, train = FALSE, valid = FALSE, xval = FALSE, ...)
```

Arguments

object	An <a href="#">H2OClusteringModel</a> object.
train	Retrieve the training total sum of squares
valid	Retrieve the validation total sum of squares
xval	Retrieve the cross-validation total sum of squares
...	further arguments to be passed on (currently unimplemented)



---

h2o.tot_withinss	<i>Get the total within cluster sum of squares. If "train", "valid", and "xval" parameters are FALSE (default), then the training tot_withinss value is returned. If more than one parameter is set to TRUE, then a named vector of tot_withinss' are returned, where the names are "train", "valid" or "xval".</i>
------------------	---

---

### Description

Get the total within cluster sum of squares. If "train", "valid", and "xval" parameters are FALSE (default), then the training tot\_withinss value is returned. If more than one parameter is set to TRUE, then a named vector of tot\_withinss' are returned, where the names are "train", "valid" or "xval".

### Usage

```
h2o.tot_withinss(object, train = FALSE, valid = FALSE, xval = FALSE, ...)
```

### Arguments

object	An <a href="#">H2OClusteringModel</a> object.
train	Retrieve the training total within cluster sum of squares
valid	Retrieve the validation total within cluster sum of squares
xval	Retrieve the cross-validation total within cluster sum of squares
...	further arguments to be passed on (currently unimplemented)

---

h2o.toupper	<i>To Upper</i>
-------------	-----------------

---

### Description

Mutates the input!

### Usage

```
h2o.toupper(x)
```

### Arguments

x	An H2OFrame object whose strings should be upper'd
---	--

---

h2o.transform

*Transform Columns in an H2OFrame Object.*


---

## Description

Functions that facilitate column transformations of an [H2OFrame](#) object.

## Usage

```
h2o.transform(_data, ...)
```

```
h2o.within(data, expr, ...)
```

## Arguments

<code>_data, data</code>	An <a href="#">H2OFrame</a> object.
<code>...</code>	For transform method, column transformations in the form tag=value.
<code>expr</code>	For within method, column transformations specified as an expression.

## See Also

[transform](#), [within](#) for the base R methods.

## Examples

```
library(h2o)
localH2O <- h2o.init()
iris.hex <- as.h2o(iris, localH2O)
transformed1 <- transform(iris.hex,
                          Sepal.Ratio = Sepal.Length / Sepal.Width,
                          Petal.Ratio = Petal.Length / Petal.Width )
transformed1
transformed2 <- within(iris.hex,
                       {Sepal.Product <- Sepal.Length * Sepal.Width
                        Petal.Product <- Petal.Length * Petal.Width
                        Sepal.Petal.Ratio <- Sepal.Product / Petal.Product
                        Sepal.Length <- Sepal.Width <- NULL
                        Petal.Length <- Petal.Width <- NULL
                       })
transformed2
```

---

h2o.trim	<i>Trim Space</i>
----------	-------------------

---

**Description**

Trim Space

**Usage**

```
h2o.trim(x)
```

**Arguments**

x	The column whose strings should be trimmed.
---	---

---

h2o.var	<i>Variance of a column.</i>
---------	------------------------------

---

**Description**

Obtain the variance of a column of a parsed H2O data object.

**Usage**

```
h2o.var(x, y = NULL, na.rm = FALSE, use)
```

```
## S4 method for signature 'H2OFrame'  
var(x, y = NULL, na.rm = FALSE, use)
```

**Arguments**

x	An <a href="#">H2OFrame</a> object.
y	NULL (default) or a column of an <a href="#">H2OFrame</a> object. The default is equivalent to y = x (but more efficient).
na.rm	logical. Should missing values be removed?
use	An optional character string to be used in the presence of missing values. This must be one of the following strings. "everything", "all.obs", or "complete.obs".

**See Also**

[var](#) for the base R implementation. [h2o.sd](#) for standard deviation.

Examples

```
localH2O <- h2o.init()
prosPath <- system.file("extdata", "prostate.csv", package="h2o")
prostate.hex <- h2o.uploadFile(localH2O, path = prosPath)
var(prostate.hex$AGE)
```

---

h2o.varimp	<i>Retrieve the variable importance.</i>
------------	--

---

Description

Retrieve the variable importance.

Usage

```
h2o.varimp(object, ...)
```

Arguments

- |        |   |
|--------|---|
| object | An <a href="#">H2OModel</a> object.                         |
| ...    | further arguments to be passed on (currently unimplemented) |

---

h2o.week	<i>Convert Milliseconds to Week of Week Year in H2O Datasets</i>
----------	--

---

Description

Converts the entries of a [H2OFrame](#) object from milliseconds to weeks of the week year (starting from 1).

Usage

```
h2o.week(x)

week(x)

## S3 method for class H2OFrame
week(x)
```

Arguments

- |   |                                     |
|---|-------------------------------------|
| x | An <a href="#">H2OFrame</a> object. |
|---|-------------------------------------|

**Value**

A [H2OFrame](#) object containing the entries of x converted to weeks of the week year.

**See Also**

[h2o.month](#)

---

h2o.weights	<i>Retrieve the respective weight matrix</i>
-------------	--

---

**Description**

Retrieve the respective weight matrix

**Usage**

```
h2o.weights(object, matrix_id = 1, ...)
```

**Arguments**

object	An <a href="#">H2OModel</a> or <a href="#">H2OModelMetrics</a>
matrix_id	An integer, ranging from 1 to number of layers + 1, that specifies the weight matrix to return.
...	further arguments to be passed to/from this method.

---

h2o.which	<i>H2O Which</i>
-----------	------------------

---

**Description**

1-based indices similar to R's which

**Usage**

```
h2o.which(x)
```

**Arguments**

x	An <a href="#">H2OFrame</a> object.
---	-------------------------------------

**Examples**

```
localH2O <- h2o.init()
fr <- as.h2o(iris)
h2o.which(fr[,5] == "setosa")
```

---

h2o.withinss	<i>Get the Within SS</i>
--------------	--------------------------

---

**Description**

Get the Within SS

**Usage**

```
h2o.withinss(object, ...)
```

**Arguments**

object	An <a href="#">H2OClusteringModel</a> object.
...	further arguments to be passed on (currently unimplemented)

---



---

h2o.year	<i>Convert Milliseconds to Years in H2O Datasets</i>
----------	--

---

**Description**

Convert the entries of a [H2OFrame](#) object from milliseconds to years, indexed starting from 1900.

**Usage**

```
h2o.year(x)

year(x)

## S3 method for class H2OFrame
year(x)
```

**Arguments**

x	An <a href="#">H2OFrame</a> object.
---	-------------------------------------

**Details**

This method calls the function of the `MutableDateTime` class in Java.

**Value**

A [H2OFrame](#) object containig the entries of x converted to years starting from 1900, e.g. 69 corresponds to the year 1969.

**See Also**

[h2o.month](#)

---

H2OClusteringModel-class

*The H2OClusteringModel object.*


---

## Description

This virtual class represents a clustering model built by H2O.

## Details

This object has slots for the key, which is a character string that points to the model key existing in the H2O cloud, the data used to build the model (an object of class H2OFrame).

## Slots

**conn** Object of class H2OConnection, which is the client object that was passed into the function call.

**model\_id** A character string specifying the key for the model fit in the H2O cloud's key-value store.

**finalizers** A list object containing environments with finalizers that remove keys from the H2O key-value store.

**algorithm** A character string specifying the algorithm that was used to fit the model.

**parameters** A list containing the parameter settings that were used to fit the model that differ from the defaults.

**allparameters** A list containing all parameters used to fit the model.

**model** A list containing the characteristics of the model returned by the algorithm.

**size** The number of points in each cluster.

**totss** Total sum of squared error to grand mean.

**withinss** A vector of within-cluster sum of squared error.

**tot\_withinss** Total within-cluster sum of squared error.

**betweenss** Between-cluster sum of squared error.

**finalizers** A list object containing environments with finalizers that remove keys from the H2O key-value store.

---

H2OConnection-class	<i>The H2OConnection class.</i>
---------------------	---------------------------------

---

## Description

This class represents a connection to an H2O cloud.

## Usage

```
## S4 method for signature H2OConnection  
show(object)
```

## Arguments

`object`            an H2OConnection object.

## Details

Because H2O is not a master-slave architecture, there is no restriction on which H2O node is used to establish the connection between R (the client) and H2O (the server).

A new H2O connection is established via the `h2o.init()` function, which takes as parameters the ‘ip’ and ‘port’ of the machine running an instance to connect with. The default behavior is to connect with a local instance of H2O at port 54321, or to boot a new local instance if one is not found at port 54321.

## Slots

`ip` A character string specifying the IP address of the H2O cloud.

`port` A numeric value specifying the port number of the H2O cloud.

`https` Set this to TRUE to use https instead of http.

`insecure` Set this to TRUE to disable SSL certificate checking.

`username` Username to login with.

`password` Password to login with.

`mutable` An H2OConnectionMutableState object to hold the mutable state for the H2O connection.



---

H2OFrame-class	<i>The H2OFrame class</i>
----------------	---------------------------

---

**Description**

The H2OFrame class

**Usage**

```
## S4 method for signature H2OFrame
show(object)
```

**Arguments**

object            An H2OConnection object.

**Slots**

conn   An H2OConnection object specifying the connection to an H2O cloud.  
 frame\_id   A character string specifying the identifier for the frame in the H2O cloud.  
 finalizers   A list object containing environments with finalizers that remove objects from the H2O cloud.  
 mutable   An H2OFrameMutableState object to hold the mutable state for the H2O frame.

---

H2OFrame-Extract	<i>Extract or Replace Parts of an H2OFrame Object</i>
------------------	---

---

**Description**

Operators to extract or replace parts of H2OFrame objects.

**Usage**

```
## S4 method for signature H2OFrame
x[i, j, ..., drop = TRUE]

## S4 method for signature H2OFrame
x$name

## S4 method for signature H2OFrame
x[[i, exact = TRUE]]

## S4 replacement method for signature H2OFrame
x[i, j, ...] <- value
```

```
## S4 replacement method for signature H2OFrame
x$name <- value
```

```
## S4 replacement method for signature H2OFrame
x[[i]] <- value
```

### Arguments

x	object from which to extract element(s) or in which to replace element(s).
i, j, ...	indices specifying elements to extract or replace. Indices are numeric or character vectors or empty (missing) or will be matched to the names.
drop	a logical, whether or not to attempt to reduce dimensions to the lowest possible.
name	a literal character string or a name (possibly backtick quoted).
exact	controls possible partial matching of [] when extracting a character
value	an array-like H2O object similar to x.

---

H2OGrid-class

*H2O Grid*


---

### Description

A class to contain the information about grid results

Format grid object in user-friendly way

### Usage

```
## S4 method for signature H2OGrid
show(object)
```

### Arguments

object	an H2OGrid object.
--------	--------------------

### Slots

conn	an <a href="#">H2OConnection</a>
grid_id	the final identifier of grid
model_ids	list of model IDs which are included in the grid object
hyper_names	list of parameter names used for grid search
failed_params	list of model parameters which caused a failure during model building, it can contain a null value
failure_details	list of detailed messages which correspond to failed parameters field
failure_stack_traces	list of stack traces corresponding to model failures reported by failed_params and failure_details fields
failed_raw_params	list of failed raw parameters

**See Also**

[H2OModel](#) for the final model types.

---

H2OModel-class	<i>The H2OModel object.</i>
----------------	-----------------------------

---

**Description**

This virtual class represents a model built by H2O.

**Usage**

```
## S4 method for signature H2OModel
show(object)
```

**Arguments**

object            an H2OModel object.

**Details**

This object has slots for the key, which is a character string that points to the model key existing in the H2O cloud, the data used to build the model (an object of class H2OFrame).

**Slots**

conn    Object of class H2OConnection, which is the client object that was passed into the function call.

model\_id    A character string specifying the key for the model fit in the H2O cloud's key-value store.

finalizers    A list object containing environments with finalizers that remove keys from the H2O key-value store.

algorithm    A character string specifying the algorithm that were used to fit the model.

parameters    A list containing the parameter settings that were used to fit the model that differ from the defaults.

allparameters    A list containing all parameters used to fit the model.

model    A list containing the characteristics of the model returned by the algorithm.

---

H2OModelFuture-class    *H2O Future Model*


---

### Description

A class to contain the information for background model jobs.

### Slots

conn an [H2OConnection](#)  
 job\_key a character key representing the identification of the job process.  
 model\_id the final identifier for the model

### See Also

[H2OModel](#) for the final model types.

---

H2OModelMetrics-class    *The H2OModelMetrics Object.*


---

### Description

A class for constructing performance measures of H2O models.

### Usage

```
## S4 method for signature H2OModelMetrics
show(object)

## S4 method for signature H2OBinomialMetrics
show(object)

## S4 method for signature H2OMultinomialMetrics
show(object)

## S4 method for signature H2ORegressionMetrics
show(object)

## S4 method for signature H2OClusteringMetrics
show(object)

## S4 method for signature H2OAutoEncoderMetrics
show(object)

## S4 method for signature H2ODimReductionMetrics
show(object)
```

**Arguments**

object            An H2OModelMetrics object

---

H2OObject-class            *The H2OObject class*

---

**Description**

The H2OObject class

**Usage**

```
## S4 method for signature H2OObject
initialize(.Object, ...)
```

**Arguments**

.Object            an H2OObject  
...                additional parameters to pass on to functions

**Slots**

conn An H2OConnection object specifying the connection to an H2O cloud.  
id A character string specifying the key in the H2O cloud's key-value store.  
finalizers A list object containing environments with finalizers that remove keys from the H2O key-value store.

---

H2ORawData-class            *The H2ORawData class.*

---

**Description**

This class represents data in a post-import format.

**Usage**

```
## S4 method for signature H2ORawData
show(object)
```

**Arguments**

object            a H2ORawData object.

**Details**

Data ingestion is a two-step process in H2O. First, a given path to a data source is `_imported_` for validation by the user. The user may continue onto `_parsing_` all of the data into memory, or the user may choose to back out and make corrections. Imported data is in a staging area such that H2O is aware of the data, but the data is not yet in memory.

The `H2ORawData` is a representation of the imported, not yet parsed, data.

**Slots**

`conn` An `H2OConnection` object containing the IP address and port number of the H2O server.

`frame_id` An object of class "character", which is the name of the key assigned to the imported data.

`finalizers` A list object containing environments with finalizers that remove objects from the H2O cloud.

---

H2OS4groupGeneric

*S4 Group Generic Functions for H2O*


---

**Description**

Methods for group generic functions and H2O objects.

**Usage**

```
## S4 method for signature missing,H2OFrame
Ops(e1, e2)
```

```
## S4 method for signature H2OFrame,missing
Ops(e1, e2)
```

```
## S4 method for signature H2OFrame,H2OFrame
Ops(e1, e2)
```

```
## S4 method for signature numeric,H2OFrame
Ops(e1, e2)
```

```
## S4 method for signature H2OFrame,numeric
Ops(e1, e2)
```

```
## S4 method for signature H2OFrame,character
Ops(e1, e2)
```

```
## S4 method for signature character,H2OFrame
Ops(e1, e2)
```

```
## S4 method for signature H2OFrame
```

```

Math(x)

## S4 method for signature H2OFrame
Math2(x, digits)

## S4 method for signature H2OFrame
Summary(x, ..., na.rm = FALSE)

## S4 method for signature H2OFrame
!x

## S4 method for signature H2OFrame
is.na(x)

## S4 method for signature H2OFrame
t(x)

## S4 method for signature H2OFrame
log(x, ...)

## S4 method for signature H2OFrame
trunc(x, ...)

## S4 method for signature H2OFrame,H2OFrame
x %*% y

```

### Arguments

<code>x,y,e1,e2</code>	objects.
<code>digits</code>	number of digits to be used in round or signif
<code>na.rm</code>	logical: should missing values be removed?
<code>...</code>	further arguments passed to or from methods

---

H2OW2V-class

*The H2OW2V object.*


---

### Description

This class represents a h2o-word2vec object.

is.factor,H2OFrame-method

*Is H2O Data Frame column a enum*

---

### Description

Is H2O Data Frame column a enum

### Usage

```
## S4 method for signature H2OFrame  
is.factor(x)
```

### Arguments

x                    an [H2OFrame](#) object column.

### Value

Returns logical value.

---

is.numeric,H2OFrame-method

*Is H2O Data Frame column numeric*

---

### Description

Is H2O Data Frame column numeric

### Usage

```
## S4 method for signature H2OFrame  
is.numeric(x)
```

### Arguments

x                    an [H2OFrame](#) object column.

### Value

Returns logical value.



**Description**

Function accessor methods for various H2O output fields.

**Usage**

```
getParms(object)

## S4 method for signature H2OModel
getParms(object)

getCenters(object)

getCentersStd(object)

getWithinSS(object)

getTotWithinSS(object)

getBetweenSS(object)

getTotSS(object)

getIterations(object)

getClusterSizes(object)

## S4 method for signature H2OClusteringModel
getCenters(object)

## S4 method for signature H2OClusteringModel
getCentersStd(object)

## S4 method for signature H2OClusteringModel
getWithinSS(object)

## S4 method for signature H2OClusteringModel
getTotWithinSS(object)

## S4 method for signature H2OClusteringModel
getBetweenSS(object)

## S4 method for signature H2OClusteringModel
getTotSS(object)
```

```
## S4 method for signature H2OClusteringModel
getIterations(object)

## S4 method for signature H2OClusteringModel
getClusterSizes(object)
```

Arguments

object                    an [H2OModel](#) class object.

---

na.omit,H2OFrame-method
<i>Remove Rows With NAs</i>

---

Description

Remove Rows With NAs

Usage

```
## S4 method for signature H2OFrame
na.omit(object, ...)
```

Arguments

object                    H2OFrame object  
...                       Ignored

---

Node-class	<i>The Node class.</i>
------------	------------------------

---

Description

An object of type Node inherits from an H2OFrame, but holds no H2O-aware data. Every node in the abstract syntax tree has as its ancestor this class.

This class represents an operator between one or more H2O objects. ASTApply nodes are always root nodes in a tree and are never leaf nodes. Operators are discussed more in depth in ops.R.

Details

Every node in the abstract syntax tree will have a symbol table, which is a dictionary of types and names for all the relevant variables and functions defined in the current scope. A missing symbol is therefore discovered by looking up the tree to the nearest symbol table defining that symbol.

---

predict.H2OModel	<i>Predict on an H2O Model</i>
------------------	--------------------------------

---

**Description**

Obtains predictions from various fitted H2O model objects.

**Usage**

```
## S3 method for class H2OModel
predict(object, newdata, ...)

h2o.predict(object, newdata, ...)
```

**Arguments**

object	a fitted <a href="#">H2OModel</a> object for which prediction is desired
newdata	A <a href="#">H2OFrame</a> object in which to look for variables with which to predict.
...	additional arguments to pass on.

**Details**

This method dispatches on the type of H2O model to select the correct prediction/scoring algorithm.

**Value**

Returns an [H2OFrame](#) object with probabilities and default predictions.

**See Also**

`link{h2o.deeplearning}`, `link{h2o.gbm}`, `link{h2o.glm}`, `link{h2o.randomForest}` for model generation in h2o.

---

print.H2OTable	<i>Print method for H2OTable objects</i>
----------------	--

---

**Description**

This will print a truncated view of the table if there are more than 20 rows.

**Usage**

```
## S3 method for class H2OTable
print(x, header = TRUE, ...)
```

**Arguments**

x	An H2OTable object
header	A logical value dictating whether or not the table name should be printed.
...	Further arguments passed to or from other methods.

**Value**

The original x object

---

range,H2OFrame-method	<i>Range of an H2O Column</i>
-----------------------	-------------------------------

---

**Description**

Range of an H2O Column

**Usage**

```
## S4 method for signature H2OFrame
range(x, na.rm = TRUE)
```

**Arguments**

x	An H2OFrame object.
na.rm	ignore missing values

---

apply,H2OFrame-method	<i>Apply Over a List in H2O</i>
-----------------------	---------------------------------

---

**Description**

Functions equivalent to the default R apply

**Usage**

```
## S4 method for signature H2OFrame
apply(X, FUN, ..., simplify = TRUE, USE.NAMES = TRUE)
```

**Arguments**

X	an <a href="#">H2OFrame</a> object on which apply will operate.
FUN	the function to be applied.
simplify,USE.NAMES	ignored parameters from base function
...	optional arguments to FUN.

See Also

link[base]{sapply} for the base implementation. export

---

str.H2OFrame	<i>Describe an H2OFrame object</i>
--------------	------------------------------------

---

Description

Describe an H2OFrame object

Usage

```
## S3 method for class H2OFrame
str(object, cols = FALSE, ...)
```

Arguments

object	An H2OFrame object.
cols	Logical indicating whether or not to do the str for all columns.
...	Extra args

---

summary,H2OGrid-method	<i>Format grid object in user-friendly way</i>
------------------------	--

---

Description

Format grid object in user-friendly way

Usage

```
## S4 method for signature H2OGrid
summary(object, show_stack_traces = F)
```

Arguments

object	an H2OGrid object.
show_stack_traces	a flag to show stack traces for model failures

---

summary,H2OModel-method  
*Print the Model Summary*

---

**Description**

Print the Model Summary

**Usage**

```
## S4 method for signature H2OModel
summary(object, ...)
```

**Arguments**

object	An <a href="#">H2OModel</a> object.
...	further arguments to be passed on (currently unimplemented)

---

zzz *Shutting down H2O for examples*

---

**Description**

Shutting down H2O for examples

**Examples**

```
h2o.shutdown(prompt=FALSE)
Sys.sleep(3)
```

# Index

!, H2OFrame-method (H2OS4groupGeneric),  
126  
\*Topic **package**  
h2o-package, 5  
[, H2OFrame-method (H2OFrame-Extract),  
121  
[<-, H2OFrame-method (H2OFrame-Extract),  
121  
[[, H2OFrame-method (H2OFrame-Extract),  
121  
[[<-, H2OFrame-method  
(H2OFrame-Extract), 121  
\$, H2OFrame-method (H2OFrame-Extract),  
121  
\$<-, H2OFrame-method (H2OFrame-Extract),  
121  
%\*%, H2OFrame, H2OFrame-method  
(H2OS4groupGeneric), 126  
%in%, H2OFrame, character-method  
(h2o.match), 72  
%in%, H2OFrame, numeric-method  
(h2o.match), 72  
  
aaa, 6  
apply, 6  
apply, H2OFrame-method, 6  
as.character, H2OFrame-method, 7  
as.data.frame.H2OFrame, 7  
as.environment, H2OFrame-method, 8  
as.factor, H2OFrame-method, 8  
as.h2o, 9  
as.numeric, H2OFrame-method, 9  
ASTApply-class (Node-class), 130  
ASTBody-class (Node-class), 130  
ASTElse-class (Node-class), 130  
ASTEmpty-class (Node-class), 130  
ASTFor-class (Node-class), 130  
ASTFun-class (Node-class), 130  
ASTIf-class (Node-class), 130  
ASTNode (ASTNode-class), 10

ASTNode-class, 10  
ASTReturn-class (Node-class), 130  
ASTSeries-class (Node-class), 130  
ASTSpan-class (Node-class), 130  
  
cbind, 16  
colnames, 11  
colnames, H2OFrame-method  
(colnames<-, H2OFrame, H2OFrame-method),  
10  
colnames<-, H2OFrame, character-method  
(colnames<-, H2OFrame, H2OFrame-method),  
10  
colnames<-, H2OFrame, H2OFrame-method,  
10  
  
day (h2o.day), 27  
dayOfWeek (h2o.dayOfWeek), 28  
ddply, 30  
dim, 36, 82  
dim, H2OFrame-method (h2o.dim), 36  
  
getBetweenSS (ModelAccessors), 129  
getBetweenSS, H2OClusteringModel-method  
(ModelAccessors), 129  
getCenters (ModelAccessors), 129  
getCenters, H2OClusteringModel-method  
(ModelAccessors), 129  
getCentersStd (ModelAccessors), 129  
getCentersStd, H2OClusteringModel-method  
(ModelAccessors), 129  
getClusterSizes (ModelAccessors), 129  
getClusterSizes, H2OClusteringModel-method  
(ModelAccessors), 129  
getIterations (ModelAccessors), 129  
getIterations, H2OClusteringModel-method  
(ModelAccessors), 129  
getParms (ModelAccessors), 129  
getParms, H2OModel-method  
(ModelAccessors), 129

- getTotSS (ModelAccessors), 129
- getTotSS, H2OClusteringModel-method (ModelAccessors), 129
- getTotWithinSS (ModelAccessors), 129
- getTotWithinSS, H2OClusteringModel-method (ModelAccessors), 129
- getWithinSS (ModelAccessors), 129
- getWithinSS, H2OClusteringModel-method (ModelAccessors), 129
- h2o (h2o-package), 5
- h2o-package, 5
- h2o.accuracy (h2o.metric), 76
- h2o.aic, 11
- h2o.anomaly, 12
- h2o.anyFactor, 13
- h2o.assign, 14, 96
- h2o.auc, 14, 46, 49, 77, 79
- h2o.betweenss, 15, 67
- h2o.biases, 16
- h2o.cbind, 16
- h2o.centers, 17, 67
- h2o.centersSTD, 17, 67
- h2o.centroid\_stats, 18
- h2o.clearLog, 18, 84, 106, 107
- h2o.cluster\_sizes, 20, 67
- h2o.clusterInfo, 19
- h2o.clusterIsUp, 19
- h2o.clusterStatus, 20
- h2o.coef, 21
- h2o.coef\_norm, 21
- h2o.colnames (colnames<-, H2OFrame, H2OFrame-method), 10
- h2o.confusionMatrix, 22, 49
- h2o.confusionMatrix, H2OModel-method (h2o.confusionMatrix), 22
- h2o.confusionMatrix, H2OModelMetrics-method (h2o.confusionMatrix), 22
- h2o.createFrame, 23
- h2o.cummax, 24
- h2o.cummin, 25
- h2o.cumprod, 25
- h2o.cumsum, 26
- h2o.cut, 26
- h2o.day, 27, 28, 57
- h2o.dayOfWeek, 28
- h2o.dct, 29
- h2o.ddply, 30
- h2o.deepfeatures, 31
- h2o.deeplearning, 12, 32
- h2o.dim, 36
- h2o.download\_pojo, 38
- h2o.downloadAllLogs, 37
- h2o.downloadCSV, 37
- h2o.error (h2o.metric), 76
- h2o.exportFile, 39
- h2o.exportHDFS, 40
- h2o.F0point5 (h2o.metric), 76
- h2o.F1 (h2o.metric), 76
- h2o.F2 (h2o.metric), 76
- h2o.fallout (h2o.metric), 76
- h2o.filterNACols, 40
- h2o.fnr (h2o.metric), 76
- h2o.fpr (h2o.metric), 76
- h2o.gbm, 41
- h2o.getConnection, 43
- h2o.getFrame, 43
- h2o.getGrid, 44
- h2o.getModel, 44
- h2o.getTimezone, 45
- h2o.giniCoef, 14, 46, 46, 49, 77
- h2o.glm, 5, 47
- h2o.glm, 50, 88
- h2o.grid, 52
- h2o.group\_by, 53
- h2o.gsub, 54
- h2o.head, 54
- h2o.hist, 55
- h2o.hit\_ratio\_table, 56
- h2o.hour, 56
- h2o.ifelse, 57
- h2o.importFile, 58
- h2o.importFolder (h2o.importFile), 58
- h2o.importHDFS (h2o.importFile), 58
- h2o.importURL (h2o.importFile), 58
- h2o.impute, 59
- h2o.init, 20, 61, 103
- h2o.insertMissingValues, 63
- h2o.interaction, 64
- h2o.is\_client, 65
- h2o.killMinus3, 66
- h2o.kmeans, 66
- h2o.length, 67
- h2o.levels, 68
- h2o.listTimezones, 69
- h2o.loadModel, 69, 97



h2o.logAndEcho, 70  
 h2o.logloss, 49, 71  
 h2o.ls, 71, 96  
 h2o.makeGLMModel, 72  
 h2o.match, 72  
 h2o.maxPerClassError (h2o.metric), 76  
 h2o.mcc (h2o.metric), 76  
 h2o.mean, 73  
 h2o.mean\_residual\_deviance, 74  
 h2o.median, 75  
 h2o.merge, 75  
 h2o.metric, 14, 46, 76, 79  
 h2o.missrate (h2o.metric), 76  
 h2o.month, 28, 78, 117, 118  
 h2o.mse, 14, 49, 77, 78, 79  
 h2o.naiveBayes, 79  
 h2o.names  
     (colnames<-, H2OFrame, H2OFrame-method), 10  
 h2o.ncol (h2o.nrow), 82  
 h2o.networkTest, 81  
 h2o.nlevels, 81  
 h2o.nrow, 82  
 h2o.null\_deviance, 82  
 h2o.null\_dof, 83  
 h2o.num\_iterations, 67, 84  
 h2o.openLog, 18, 84, 106, 107  
 h2o.parseRaw, 85  
 h2o.parseSetup, 86  
 h2o.performance, 14, 22, 46, 49, 77, 79, 86  
 h2o.prcomp, 87  
 h2o.precision (h2o.metric), 76  
 h2o.predict (predict.H2OModel), 131  
 h2o.quantile, 88  
 h2o.r2, 89  
 h2o.randomForest, 90  
 h2o.rbind, 92  
 h2o.recall (h2o.metric), 76  
 h2o.removeAll, 93  
 h2o.removeVecs, 93  
 h2o.rep\_len, 94  
 h2o.residual\_deviance, 94  
 h2o.residual\_dof, 95  
 h2o.rm, 93, 96  
 h2o.runif, 96  
 h2o.saveModel, 69, 97  
 h2o.scale, 98  
 h2o.scoreHistory, 49, 99  
 h2o.sd, 99, 115  
 h2o.sdev, 100  
 h2o.sensitivity (h2o.metric), 76  
 h2o.setLevel, 100  
 h2o.setLevels, 101  
 h2o.setTimezone, 101  
 h2o.shim, 102  
 h2o.shutdown, 62, 102  
 h2o.specificity (h2o.metric), 76  
 h2o.splitFrame, 103  
 h2o.startGLMJob, 104  
 h2o.startLogging, 18, 84, 106, 107  
 h2o.stopLogging, 18, 84, 106, 106  
 h2o.strsplit, 107  
 h2o.sub, 107  
 h2o.subset, 108  
 h2o.summary, 108  
 h2o.svd, 88, 109  
 h2o.table, 110  
 h2o.tabulate, 111  
 h2o.tail (h2o.head), 54  
 h2o.tnr (h2o.metric), 76  
 h2o.tolower, 112  
 h2o.tot\_withinss, 67, 113  
 h2o.totss, 67, 112  
 h2o.toupper, 113  
 h2o.tpr (h2o.metric), 76  
 h2o.transform, 114  
 h2o.trim, 115  
 h2o.uploadFile (h2o.importFile), 58  
 h2o.var, 99, 115  
 h2o.varimp, 49, 116  
 h2o.week, 116  
 h2o.weights, 117  
 h2o.which, 117  
 h2o.within (h2o.transform), 114  
 h2o.withinss, 67, 118  
 h2o.year, 78, 118  
 H2OAutoEncoderMetrics-class  
     (H2OModelMetrics-class), 124  
 H2OAutoEncoderModel, 12  
 H2OAutoEncoderModel-class  
     (H2OModel-class), 123  
 H2OBinomialMetrics, 14, 22, 46, 71, 77, 79  
 H2OBinomialMetrics-class  
     (H2OModelMetrics-class), 124  
 H2OBinomialModel, 49, 80  
 H2OBinomialModel-class

- (H2OModel-class), 123
- H2OClusteringMetrics-class
  - (H2OModelMetrics-class), 124
- H2OClusteringModel, 15, 17, 18, 21, 67, 84, 112, 113, 118
- H2OClusteringModel-class, 119
- H2OConnection, 9, 20, 23, 37, 43, 45, 58, 66, 69–71, 81, 93, 96, 102, 122, 124
- H2OConnection (H2OConnection-class), 120
- H2OConnection-class, 120
- H2ODimReductionMetrics-class
  - (H2OModelMetrics-class), 124
- H2ODimReductionModel, 51, 88, 100, 110
- H2ODimReductionModel-class
  - (H2OModel-class), 123
- H2OFrame, 6–8, 10–14, 16, 22, 24–33, 36, 37, 39–42, 47, 50, 53–57, 59, 63, 64, 66–68, 73, 75, 78, 80, 82, 85–87, 89–92, 94, 96, 98–100, 103, 104, 108–111, 114–118, 128, 131, 132
- H2OFrame (H2OFrame-class), 121
- H2OFrame-class, 121
- H2OFrame-Extract, 121
- H2OGrid (H2OGrid-class), 122
- H2OGrid-class, 122
- H2OModel, 12, 14, 16, 21, 22, 31, 45, 46, 49, 56, 69, 71, 72, 74, 79, 83, 87, 90, 92, 95, 97, 99, 116, 117, 123, 124, 130, 131, 134
- H2OModel (H2OModel-class), 123
- H2OModel-class, 123
- H2OModelFuture, 106
- H2OModelFuture-class, 124
- H2OModelMetrics, 12, 16, 22, 71, 77–79, 83, 87, 95, 117
- H2OModelMetrics
  - (H2OModelMetrics-class), 124
- H2OModelMetrics-class, 124
- H2OMultinomialMetrics, 22, 71, 79
- H2OMultinomialMetrics-class
  - (H2OModelMetrics-class), 124
- H2OMultinomialModel, 80
- H2OMultinomialModel-class
  - (H2OModel-class), 123
- H2OObject (H2OObject-class), 125
- H2OObject-class, 125
- H2ORawData, 59, 85, 86
- H2ORawData (H2ORawData-class), 125
- H2ORawData-class, 125
- H2ORegressionMetrics, 79
- H2ORegressionMetrics-class
  - (H2OModelMetrics-class), 124
- H2ORegressionModel, 49
- H2ORegressionModel-class
  - (H2OModel-class), 123
- H2OS4groupGeneric, 126
- H2OUnknownMetrics-class
  - (H2OModelMetrics-class), 124
- H2OUnknownModel-class (H2OModel-class), 123
- H2OW2V (H2OW2V-class), 127
- H2OW2V-class, 127
- head, H2OFrame-method (h2o.head), 54
- hour (h2o.hour), 56
- ifelse, ANY, H2OFrame, H2OFrame-method (h2o.ifelse), 57
- ifelse, H2OFrame, ANY, ANY-method (h2o.ifelse), 57
- initialize, H2OObject-method (H2OObject-class), 125
- is.factor, 9
- is.factor, H2OFrame-method, 128
- is.na, H2OFrame-method (H2OS4groupGeneric), 126
- is.numeric, H2OFrame-method, 128
- length, 68
- length, H2OFrame-method (h2o.length), 67
- levels, 68
- log, H2OFrame-method (H2OS4groupGeneric), 126
- match, 73
- match, H2OFrame-method (h2o.match), 72
- Math, H2OFrame-method (H2OS4groupGeneric), 126
- Math2, H2OFrame-method (H2OS4groupGeneric), 126
- mean, 73
- mean, H2OFrame-method (h2o.mean), 73
- median, H2OFrame-method (h2o.median), 75
- ModelAccessors, 129
- month (h2o.month), 78
- na.omit, H2OFrame-method, 130

names,H2OFrame-method  
     (colnames<-,H2OFrame,H2OFrame-method), 121  
     10  
 names<-,H2OFrame-method  
     (colnames<-,H2OFrame,H2OFrame-method), 123  
     10  
 ncol,H2OFrame-method (h2o.ncol), 82  
 Node (Node-class), 130  
 Node-class, 130  
 nrow, 82  
 nrow,H2OFrame-method (h2o.nrow), 82  
  
 Ops,character,H2OFrame-method  
     (H2OS4groupGeneric), 126  
 Ops,H2OFrame,character-method  
     (H2OS4groupGeneric), 126  
 Ops,H2OFrame,H2OFrame-method  
     (H2OS4groupGeneric), 126  
 Ops,H2OFrame,missing-method  
     (H2OS4groupGeneric), 126  
 Ops,H2OFrame,numeric-method  
     (H2OS4groupGeneric), 126  
 Ops,missing,H2OFrame-method  
     (H2OS4groupGeneric), 126  
 Ops,numeric,H2OFrame-method  
     (H2OS4groupGeneric), 126  
  
 predict, 22  
 predict.H2OModel, 35, 42, 49, 92, 131  
 print.H2OTable, 131  
  
 quantile, 89  
  
 range,H2OFrame-method, 132  
 rbind, 92  
  
 sapply,H2OFrame-method, 132  
 sd, 99  
 sd,H2OFrame-method (h2o.sd), 99  
 show,ASTNode-method (ASTNode-class), 10  
 show,H2OAutoEncoderMetrics-method  
     (H2OModelMetrics-class), 124  
 show,H2OBinomialMetrics-method  
     (H2OModelMetrics-class), 124  
 show,H2OClusteringMetrics-method  
     (H2OModelMetrics-class), 124  
 show,H2OConnection-method  
     (H2OConnection-class), 120  
 show,H2ODimReductionMetrics-method  
     (H2OModelMetrics-class), 124  
  
 show,H2OFrame-method (H2OFrame-class), 121  
 show,H2OGrid-method (H2OGrid-class), 122  
 show,H2OModel-method (H2OModel-class), 123  
 show,H2OModelMetrics-method  
     (H2OModelMetrics-class), 124  
 show,H2OMultinomialMetrics-method  
     (H2OModelMetrics-class), 124  
 show,H2ORawData-method  
     (H2ORawData-class), 125  
 show,H2ORegressionMetrics-method  
     (H2OModelMetrics-class), 124  
 str.H2OFrame, 133  
 subset, 108  
 summary, 108  
 Summary,H2OFrame-method  
     (H2OS4groupGeneric), 126  
 summary,H2OFrame-method (h2o.summary), 108  
 summary,H2OGrid-method, 133  
 summary,H2OModel-method, 134  
  
 t,H2OFrame-method (H2OS4groupGeneric), 126  
 tail,H2OFrame-method (h2o.head), 54  
 transform, 114  
 trunc,H2OFrame-method  
     (H2OS4groupGeneric), 126  
  
 var, 115  
 var,H2OFrame-method (h2o.var), 115  
  
 week (h2o.week), 116  
 within, 114  
  
 year (h2o.year), 118  
  
 zzz, 134