IBM Big R Package Specification

February 10, 2015

R topics documented:

An Introduction to Big R

An Introduction to Big R

Description

Big R (bigr) provides an end-to-end integration of R within IBM InfoSphere BigInsights. This makes it easy to write and execute R programs that operate on big data.

Details

Using Big R, an R user can explore, transform, and analyze big data hosted in a BigInsights cluster using familiar R syntax and paradigm. All of theses capabilities are accessible from a standard R client. The goals of Big R are two-fold:

(1) Enable the use of R as a query language for big data: Big R hides many of the complexities pertaining to the underlying Hadoop / MapReduce framework. Using classes such as bigr.frame,

bigr.vector and bigr.list, a user is presented with an API that is heavily inspired by R's foundational API on data.frames, vectors and frames.

(2) Enable the pushdown of R functions such that they run right on the data: Via mechanisms such as groupApply, rowApply and tableApply, user-written functions composed in R can be shipped to the cluster. BigInsights transparently parallelizes exection of these function and provides consolidated results back to the user. Almost any R code, including most packages available on open-source repositories such as CRAN, can be run using this mechanism.

To use Big R, one needs to take the following steps:

(1) Install the "bigr" package on the client. Big R requires several pre-requisite packages including rJava, data.table, and base64enc.

(2) To enable function pushdown via the "Apply" functions, each node of a BigInsights cluster needs to have the R interpreter installed on it. In addition, each node also needs the same "bigr" package installed.

(3) Ensure that Big SQL server is running on the BigInsights cluster. As Big R statements are executed, they are transparently converted into corresponding SQL and JaQL statements, and these are executed by the Big SQL server.

The builtin package documentation describes the package API and includes numerous illustrative examples. Many of the examples refer to the "airline" dataset. This data originally comes from US DOT/RITA (http://www.rita.dot.gov) and a cleansed version is also available at http://stat-computing.org/dataexpo/2009/the-data.html. This data serves as an excellent vehicle for describing the capabilities of Big R. A small sample of the original data is bundled with the Big R package itself.

Examples

```
## Not run:

# In order to try out any example, first run the following steps to upload
# the aforementioned dataset to a BigInsights cluster.
bigr.connect(host="hostname", port=7052,
          user="username", password="password")
airfile <- system.file("extdata", "airline.zip", package="bigr")
airfile <- unzip(airfile, exdir = tempdir())
airR <- read.csv(airfile, stringsAsFactors=F)

# Upload the data to the BigInsights server. This may take 15-20 seconds
air <- as.bigr.frame(airR)
air <- bigr.persist(air, dataSource="DEL", dataPath="airline_demo.csv",
                header=T, delimiter=",", useMapReduce=F)

# Once uploaded, one merely needs to instantiate a big.frame object,
# commonly referenced as "air" in the examples, to access the dataset via
# the Big R API.
air <- bigr.frame(dataPath = "airline_demo.csv",
              dataSource = "DEL",
              delimiter=",", header = T,
              coltypes = ifelse(1:29 %in% c(9,11,17,18,23),
                           "character", "integer"),
              useMapReduce = F)
```

```
## End(Not run)
```

Arithmetic operators   +, -, *, /, ^

Description

Big R supports standard operations (+, -, *, /, ^) on two bigr.vectors, or on a bigr.vector and a single numeric value. The bigr.vector(s) must have an underlying data type of "numeric" or "integer".

Usage

```
"+"(e1, e2)
```

Arguments

x	a bigr.vector
y	a bigr.vector or a numeric value

Value

a bigr.vector that holds result of the arithmetic operation

Usage

```
x operator y
```

Examples

```
## Not run:

# Compute difference between actual and scheduled time
diff <- air$CRSElapsedTime - air$ActualElapsedTime

# Compute average speed in miles per hour
speed <- air$Distance / (air$ActualElapsedTime / 60)
## End(Not run)
```

as.bigr.frame *Turn an R data.frame to a bigr.frame*

Description

as.bigr.frame uploads the contents of an R data.frame to a temporary dataset on a BigInsights cluster, and presents the resulting dataset as a bigr.frame object. This function creates a temporary file in the /tmp/bigr directory. Once the bigr.frame object that references the temporary file goes out of scope, and assuming that a valid connection to BigInsights still exists, the file is deleted. To provide a different name to the file, use bigr.persist().

Usage

```
as.bigr.frame(df)
```

Arguments

df	a data.frame

Value

a bigr.frame

Caveats

This mechanism is only designed to transfer small files to HDFS.

See Also

as.bigr.vector, bigr.persist, Temporary Files

Examples

```
## Not run:

irisbf <- as.bigr.frame(iris)
irisbf <- bigr.persist(irisbf, dataSource="DEL", dataPath="iris.csv",
                       header=T, del=",")

## End(Not run)
```

as.bigr.vector *Turn an R vector into a bigr.vector*

Description

as.bigr.vector uploads the contents of an R vector to a temporary dataset on a BigInsights cluster, and presents the resulting dataset as a bigr.vector object. This function creates a temporary file in the /tmp/bigr directory. Once the bigr.vector object that references the temporary file goes out of scope, and assuming that a valid connection to BigInsights still exists, the file is deleted.

Usage

```
as.bigr.vector(v)
```

Arguments

v	a vector

Value

a bigr.vector

Caveats

This mechanism is only designed to transfer small datasets to HDFS.

See Also

as.bigr.frame, Temporary Files

Examples

```
## Not run:

irisbv <- as.bigr.vector(unique(iris$Species))

## End(Not run)
```

as.data.frame — *Download data from the server into a data.frame*

Description

This function downloads the contents of a bigr.frame into a data.frame. Since R data.frames are held in memory, ensure that you have enough memory on your system to accommodate the contents.

Usage

```
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	a bigr.frame

Value

a data.frame

See Also

as.vector

Examples

```
## Not run:

airdf <- as.data.frame(air)
airha <- as.data.frame(air[air$UniqueCarrier == "HA",])

## End(Not run)
```

as.list — *as.list*

Description

See bigr.pull

Usage

```
as.list(x, ...)
```

as.vector — *Download data from the server into a vector*

Description

This function downloads the contents of a bigr.vector into an R vector. Since R vectors are held in memory, ensure that you have enough memory on your system to accommodate the contents.

Usage

```
as.vector(x, mode = "any")
```

Arguments

x	a bigr.vector

Value

a vector containing the data in the bigr.vector

See Also

as.data.frame

Examples

```
## Not run:

carriers <- as.vector(air$UniqueCarrier)

years <- as.vector(unique(air$Year))

## End(Not run)
```

attach — *Attach bigr.frame to search path*

Description

The specified bigr.frame is attached to the R search path. This means that the bigr.frame is searched by R when evaluating a variable, so columns in the bigr.frame can be accessed by simply giving their names.

Usage

```
attach(what, pos = 2L, name = deparse(substitute(what)),
  warn.conflicts = TRUE)
```

Arguments

what	(bigr.frame) The frame to attach
pos	(integer) Specify position in search() where to attach.
name	(character) Name to use for the attached database. Names starting with package: are reserved for library.
warn.conflicts	(logical) If TRUE, warnings are printed about conflicts from attaching the database, unless that database contains an object

See Also

detach

Examples

```
## Not run:

attach(air)

print(Distance)

# How many flights were flown by specific airlines?
length(UniqueCarrier[UniqueCarrier %in% c("UA", "DL")])

# Number of flights delayed by more than 15 minutes?
length(DepDelay[DepDelay >= 15])

## End(Not run)
```

bigr.bivariateStats *Bivariate statistics*

Description

This function computes bivariate statistics from a bigr.matrix, given two sets of columns. The statistics can be calculated for the entire dataset or in a stratified fashion.

Usage

```
bigr.bivariateStats(data, cols1, cols2, strata = NULL)
```

Arguments

data	(bigr.matrix) the datasets which the statistics will be calculated from
cols1	(character) the names of the first set of columns
cols2	(character) the names of the second set of columns
strata	(character/bigr.vector) the strata column as a character or a bigr.vector Please refer to the algorithm reference documentation for further details about the output of this function.

bigr.connect *Connect to BigInsights*

Description

bigr.connect attempts to establish a JDBC connection to BigInsights. For the call to be successful, there needs to be a Big SQL server running at the specified host. In addition, the user must have the proper credentials. As Big R statements are executed, they are transparently converted into corresponding SQL and JaQL statements, and these are executed within the context of the established connection.

Usage

```
bigr.connect(host, port, database, user, password, driverPath)
```

Arguments

host	(character) IP address of the Big SQL server
port	(integer) port of the Big SQL server
database	(character) not used, may be omitted
user	(character) user name for the connection
password	(character) password of the user
driverPath	(character) Optional location of the Big SQL JDBC driver JAR file. This option is typically not specified, and Big R uses the default driver that is bundled with the package itself. This option is used to override the default driver. Alternatively, a driver may also be specified via the CLASSPATH environment variable.

Details

Once connected, it is recommended that you use bigr.set.server.option() to set server-side options. Specifically, when running large-scale machine learning algorithms such as bigr.lm, bigr.svm, etc., set "jaql.fence.jvm.parameters" to adjust the size of JVM memory. If not specified, "jaql.fence.jvm.parameters" defaults to "-Xmx2048m -Xms2048m -Xmn256m"

Value

TRUE if the connection was successful, FALSE otherwise.

See Also

bigr.disconnect, bigr.reconnect, is.bigr.connected, bigr.set.server.option

Examples

```
## Not run:
bigr.connect(host="131.45.112.15", port=7052,
         user="biadmin", password="xxx")
## End(Not run)
```

bigr.disconnect *Disconnect from BigInsights*

Description

Disconnect from BigInsights

Usage

```
bigr.disconnect()
```

Value

a boolean indicating whether or not the connection was terminated.

See Also

bigr.connect, bigr.reconnect, is.bigr.connected

Examples

```
## Not run:
bigr.disconnect()
## End(Not run)
```

bigr.execute *Execute the DML script*

Description

Execute the specified DML script

Usage

```
bigr.execute(dmlScript, ...)
```

Arguments

dmlScript	(character) The name of the DML script. The script file must be in the $BIGINSIGHTS_HOME/machine-learning/algorithms path.
...	(optional) The pairs of parameter=value to be passed to the dmlScript

bigr.extractFeaturesFromText

This function uses BigInsights Text Analytics to extract feature vectors from a collection of documents stored on the distributed filesystem. a given bigr.vector.

Description

The extractFeatures() function will only accept the input collection format documented in the BigInsights Information Center under Analyzing big data>Analyzing big data with Text Analytics>Text Analytics overview>Data collection formats>UTF-8 encoded JSON files in Hadoop text input format. This JSON-based collection format supports documents with arbitrary sets of document fields, as well as external views.

Usage

```
bigr.extractFeaturesFromText(docCollectionURI, aqlModulesURI, aqlModuleName,
  aqlViewName)
```

Arguments

docCollectionURI	HDFS, GPFS, or file: URI that points to either a single JSON text file or to a directory that contains multiple such files.
aqlModulesURI	HDFS, GPFS, or file: URI that points to a directory containing one or more compiled AQL modules (.tam files).
aqlModuleName	Name of the AQL module whose .tam file is located in the specified directory, along with any other modules required to run the target module.
aqlViewName	Name of an AQL view from the specified module.

Details

The extractFeatures() function will attempt to read the first line of input and check that the first line is correctly-formatted JSON data according to the documented format. If the first line does not conform to the format, then the function will exit with an appropriate error message. All other errors encountered interpreting the input data will result in less user-friendly Jaql error messages.

The extractFeatures() function will start by instantiating the module specified in the aqlModuleName parameter, using the value of the aqlModulesURI parameter as the only entry in the AQL module path. Then the function will read the output schema of the view named by the aqlViewName parameter. The function will use this schema information to generate the fieldOrder argument to the flattenTuples() function. Then the function will iterate over the documents in the collection. For each document, the function will pass the document, plus any external views, to the Jaql function systemT::annotateDocument(), then pass the resulting JSON record, along with the generated fieldOrder argument, to the flattenTuples() function described in the previous section. The extractFeatures() function will return a stream (array) of arrays, where each sub-array represents a single line of output.

The extractFeatures() function will guarantee that the order of rows in the output data exactly corresponds to the order of documents and to the order of outputs per document.

Value

a BigR frame containing the extracted feature vectors

---

bigr.frame *Contruct a bigr.frame object*

Description

bigr.frame objects are proxies for tabular datasets residing in a BigInsights cluster. These datasets could be delimited files, tables in a Big SQL server, or flat files containing JSON records. For JSON files, the format must be such each line represents a complete JSON record, and all lines contain records with identical schemas.

Usage

```
bigr.frame(dataSource, dataPath, delimiter = ifelse(dataSource == "DEL", ",",
  ""), colnames = NULL, coltypes = NULL, header = FALSE, na.string = "",
  useMapReduce = TRUE, quoted = TRUE, ddquote = TRUE, escape = FALSE,
  doubleq = TRUE)
```

Arguments

dataSource	(character) Type of data source. Supported sources are "DEL", "BIGSQL" and "JSON"
dataPath	(character) Complete name or path of the dataset
delimiter	(character) Delimiter if dataSource is set to "DEL"
colnames	(character vector) Names of the table columns.
coltypes	(character vector) an array of column types.
header	(logical) whether or not the dataset has headers. This only applies if dataSource is "DEL"
na.string	(character) a character vector of strings which are to be interpreted as NA values. Blank fields are also considered to be missing values.
useMapReduce	(logical) If set to FALSE, Big R will execute all statements on a single-node. This is the node where the Big SQL server is running, and it is typically the BigInsights console node. By using useMapReduce = FALSE, one experiences lower execution-time latencies. This mode of operation can be very useful when experimenting with smaller files, especially those that are created by sampling large datasets.
quoted	(logical) When dataSource is "DEL", indicate whether the column content is quoted or not.

ddquote	(logical) When dataSource is "DEL", indicate whether double quotes is used as the escape character.
escape	(logical) When dataSource is "DEL", indicate whether the character with the escape character should be escaped or not.
doubleq	(logical) When dataSource is "DEL", indicate whether the column string is quoted by " or ' quotation mark.

Value

a newly created bigr.frame

Examples

```
## Not run:

# Construct a bigr.frame atop a delimited file
air <- bigr.frame(dataPath = "airline_demo.csv",
                  dataSource = "DEL",
                  delimiter=",", header = T,
                  coltypes = ifelse(1:29 %in% c(9,11,17,18,23),
                                    "character", "integer"),
                  useMapReduce = F)

# Construct a bigr.frame over a Big SQL catalog table
bfcat <- bigr.frame(dataSource = "BIGSQL", dataPath = "syscat.tables")

# Construct a bigr.frame over a Big SQL user table
bfhbase <- bigr.frame(dataSource = "BIGSQL", dataPath = "hbase.emp")


## End(Not run)
```

---

bigr.get.server.option *Get Hadoop/MapReduce options*

Description

List the values of one or more Hadoop/MapReduce options that apply to the current Big R session.

Usage

```
bigr.get.server.option(option)
```

Arguments

option	(optional) If specified, returns the value of the specified option. This option may have previously been set by the user via bigr.set.server.option(), and if so, that value is returned. If the value of the option hasn't been set, then the value from the default Hadoop JobConf is returned. If "option" is not specified, return all options that were previously set via bigr.set.server.option().

See Also

bigr.set.server.option

Examples

```
## Not run:
bigr.get.server.option()
bigr.get.server.option("mapred.reduce.tasks")
## End(Not run)
```

---

bigr.getRowLimit *Control printing of rows and elements*

Description

See Control printing of rows and elements

Usage

```
bigr.getRowLimit()
```

---

bigr.glm *Generalized Linear Models*

Description

Trains a Generalized Linear Model for a given bigr.matrix with the specified parameters.

Usage

```
bigr.glm(formula, data, family, BernoulliResponse = 0, intercept = FALSE,
  shiftAndRescale = FALSE, lambda = 0, tolerance = 1e-06,
  dispersion = 0, outer.iter.max = 200, inner.iter.max = 0,
  directory = "")
```

Arguments

formula	(formula) a formula in the form Y ~ . indicating the response variable.
data	(bigr.matrix) the training dataset
family	(family) the distribution family and link function. Supported families are "gaussian", "poisson", "Gamma", and "inverse.gaussian". Supported link functions are "identity", "inverse", "log", "sqrt", "1/mu^2", "logit", "probit", "cloglog", and "cauchit". By defaut, gaussian(canonical) is used.
BernoulliResponse	(numeric) response value for Bernoulli "No" label, usually 0.0 or -1.0

intercept	(logical) a boolean value indicating wheter intercept should be used or not
shiftAndRescale	(logical) a boolean value indicating whether shifting and rescaling should be done or not
lambda	(numeric) regularization parameter for L2 regularization
tolerance	(numeric) tolerance (epsilon)
dispersion	(numeric) (Over-)dispersion value, or 0.0 to estimate it from data
outer.iter.max	(integer) maximum number of outer (Newton / Fisher Scoring) iterations
inner.iter.max	(integer) maximum number of inner (Conjugate Gradient) iterations, 0 = no maximum
directory	(character) the HDFS/GPFS path where the model will be stored

bigr.histogram *Renders a histogram for a given bigr.vector*

Description

This function renders a histogram for a given dataset.

Usage

```
bigr.histogram(formula, nbins = 10, data = NULL)
```

Arguments

formula	a formula or a bigr.vector specifying the data source
nbins	the number of bins (optional). The default is 10.
data	optional bigr.frame. If specified, this serves as the environment for column references in the formula that don't have an explicit bigr.frame.

Details

If a formula is passed in as parameter, it must specify (1) A bigr.frame as the data origin, (2) A bigr.vector with the target column for the histogram, and (3) One or more grouping columns separated by the + operator

This function depends on the "ggplot2" package.

Value

a data.frame with the computed statistics

Examples

```
## Not run:

# Render a histogram for the DepDelay column
bigr.histogram(air$DepDelay, nbins=100)

# Render a histogram for the DepDelay column for each distinct combination
# of UniqueCarrier and Year
bf <- air[air$UniqueCarrier %in% c("UA", "HA")
        & air$Year %in% c(2004, 2005), ]
bigr.histogram(bf$DepDelay ~ bf$UniqueCarrier + bf$Year)

## End(Not run)
```

---

bigr.histogram.stats *Compute histogram statistics*

Description

This function computes statistics to render a histogram. These are centroids and counts. See bigr.histogram on usage.

Usage

```
bigr.histogram.stats(formula, nbins, data = NULL)
```

Arguments

formula	a formula or a bigr.vector specifying the data source
nbins	the number of bins (optional)
data	optional bigr.frame. If specified, this serves as the environment for column references in the formula that don't have an explicit bigr.frame.

Value

a data.frame with the computed statistics

See Also

bigr.histogram

bigr.kmeans *K-Means Clustering*

Description

Perform k-means clustering on a bigr.matrix

Usage

```
bigr.kmeans(data, centers, runs = 100, iter.max = 1000, tolerance = 1e-06,
  samp = 10, writeY = F, directory)
```

Arguments

data	(bigr.matrix) A bigr.matrix object
centers	Number of centroids
runs	Number of runs (with different initial centroids)
iter.max	Maximum number of iterations per run
tolerance	Tolerance (epsilon) for WCSS change ratio
samp	Average number of records per centroid in data samples
writeY	(logical) Whether to write the cluster assignments
directory	Directory to save the output Kmeans model

Details

Kmeans

Value

An object of class 'bigr.kmeans'

bigr.list *class bigr.list*

Description

Class bigr.list

Details

bigr.list is a collection of one or more R objects that are generated by a groupApply() function call. It is loosely modeled after R's "list" class. bigr.list only holds references to its constituents. The objects themselves are stored in a serialized form on HDFS. These objects can be referenced individually or selectively using the "$" or "[" methods. They can also be pulled to the client using the as.list() or bigr.pull() functions.

See Also

groupApply, dimnames, [, $, as.list, bigr.pull,

Examples

```
## Not run:

# See groupApply for examples on using bigr.list objects
#
## End(Not run)
```

---

bigr.listColumns *List Big SQL database columns*

Description

Returns a data.frame describing database columns

Usage

```
bigr.listColumns(fulltablename = "*.*", expand = T)
```

Arguments

fulltablename	(character) Name of the table. Can be a simple regular expression such as "syscat.*"
expand	(logical) Controls whether the regular expression is expanded. Defaults to TRUE.

See Also

bigr.listColumns

Examples

```
## Not run:
bigr.listColumns("system.dual")
bigr.listColumns("system.*")
bigr.listColumns("*.dual")
bigr.listColumns("*.*")
## End(Not run)
```

bigr.listfs — *List files on the BigInsights file system*

Description

List files on the BigInsights file system

Usage

```
bigr.listfs(pathExpression, directoryOnly = FALSE, compact = TRUE)
```

Arguments

pathExpression	Name of a file/directory, or a simple glob. Hidden files, i.e. files whose names start with a period (".")) are not returned
directoryOnly	Should only the directory be returned (TRUE), or should the directory contents be returned (FALSE)? Default is FALSE.
compact	Return only selected fields (TRUE), or all of them (FALSE)?

Examples

```
## Not run:
bigr.listfs("/tmp")
bigr.listfs("/user/biadmin")
## End(Not run)
```

bigr.listTables — *List Big SQL database tables*

Description

Returns a data.frame describing database tables

Usage

```
bigr.listTables(fulltablename = "*.*", expand = T)
```

Arguments

fulltablename	(character) Name of the table. Can be a simple regular expression such as "syscat.*"
expand	(logical) Controls whether the regular expression is expanded. Defaults to TRUE.

See Also

bigr.listColumns

Examples

```
## Not run:
bigr.listTables()
bigr.listTables("system.dual")
bigr.listTables("system.*")
bigr.listTables("*.dual")

## End(Not run)
```

---

bigr.lm *Linear Regression*

Description

bigr.lm is used to fit a linear regression model.

Usage

```
bigr.lm(formula, data, method = "direct-solve", intercept = F,
  shiftAndRescale = F, tolerance = 0.001, iter.max = 100, lambda = 1,
  directory)
```

Arguments

formula	(formula) A formula in the form Y ~ ., indicating the response variable. NOTE: the response variable must be nominal.
data	(bigr.matrix) The bigr.matrix that includes both the X and Y parts
method	(character) Either "direct-solve" or "interative" methods are supported
intercept	(logical) intercept value, either TRUE or FALSE
shiftAndRescale	(logical) Whether to shift and rescale
tolerance	(numeric) tolerance
iter.max	(numeric) Number of iteractions
lambda	(numeric) lambda
directory	(character) The directory to save the outpu Linear Regression model

Details

Generic construct method to initialize the bigr.lm class Linear Regression

Value

An object of ’bigr.lm’ class representing the Linear Regression model just learned

bigr.logistic.regression *Learn a Logistic Regression model based on the given labelled dataset*

Description

With this method one can learn a Logistic Regression model for a given labeled dataset.

Usage

```
bigr.logistic.regression(formula, xy, directory, intercept = FALSE,
  shiftAndRescale = FALSE, reg = 0, tol = 1e-06, maxOuterIter = 100,
  maxInnerIter = 0)
```

Arguments

formula	(formula) : A formula of the type "label~." specifying the label column
xy	(bigr.matrix) : A Labeled dataset with a label column of integer values and name specified in parameter formula
directory	(character) : Path to store the learned model
intercept	(boolean) : A boolean value specifying whether or not to include an intercept term in the model
shiftAndRescale	(boolean) : A boolean controlling whether to shift & rescale X columns to mean = 0, variance = 1 or not
reg	(numeric) : regularization parameter (lambda = 1/C); intercept is not regularized
tol	(numeric) : tolerance ("epsilon")
maxOuterIter	(numeric): max. number of outer (Newton) iterations
maxInnerIter	(numeric) : max. number of inner (conjugate gradient) iterations, 0 = no max

Value

a bigr.logistic.regression object representing the learned Logistic Regression model

Usage

bigr.logistic.regression(formula, xy, directory, intercept = FALSE, shiftAndRescale = FALSE, reg = 0.0, tol = 0.000001, maxOuterIter = 100, maxInnerIter = 0)

bigr.logs *Display output generated by *Apply functions*

Description

Returns a numbered list of log files or shows the log file content for the specified log file produced through the *Apply function.

Usage

```
bigr.logs(...)
```

Arguments

...	(optional) A bigr.frame or bigr.list object whose log files are to be displayed. If this parameter is not specified, log files of the the most recently executed *Apply call are displayed.
...	(optional) A numeric vector identifying the number(s) of the log file(s) whose contents are to be displayed. If not specified, merely a list of log files is returned

Details

Each instance of an *Apply function (groupApply, rowApply and tableApply) that executes on a BigInsights cluster can write output to "stdout" or "stderr". Big R captures this output and preserves it in the form of log files. bigr.logs() can be used to show a numbered list of log files generated by an *Apply call. In addition, the same function can be used to examine the detailed output in any specific log file. In the case of groupApply(), the log files are tagged with the grouping columns, and this makes it easier to correlate groups with their corresponding log files. In the case of tableApply(), only one log file is produced.

See Also

groupApply, rowApply, tableApply

bigr.matrix *bigr.matrix*

Description

bigr.matrix objects are proxies for numeric tabular datasets residing in a file on HDFS/GPFS.

Usage

```
bigr.matrix(dataPath = "", delimiter = ",", colnames = "",
  header = FALSE, na.string = bigr.env$DEFAULT_NA_STRING,
  useMapReduce = !bigr.env$DEFAULT_LOCAL_PROCESSING, transformPath = "")
```

Arguments

dataPath	(character) Location of the dataset on HDFS/GPFS
delimiter	(character) The delimiter is optional. By default, the delimiter is set to comma.
colnames	(character vector) Names of the table columns.
header	(logical) whether or not the dataset has headers.
na.string	(character) a character vector of strings which are to be interpreted as NA values. Blank fields are also considered to be missing values.
useMapReduce	(logical) If set to FALSE, Big R will execute all statements on a single-node. This is the node where the Big SQL server is running, and it is typically the BigInsights console node. By using useMapReduce = FALSE, one experiences lower execution-time latencies. This mode of operation can be very useful when experimenting with smaller files, especially those that are created by sampling large datasets.
transformPath(character)	Location of transform related data, if exists (optional).

Value

a newly created bigr.matrix

---

bigr.naive.bayes *Naive Bayes classifier*

Description

This method trains a Naive Bayes classifier for a given dataset with the specified parameters. The model is automatically stored on the given location.

Usage

```
bigr.naive.bayes(formula, xy, directory, laplace = DEFAULT_LAPLACE_CORRECTION)
```

Arguments

formula	(formula) : A formula in the form Y ~ . where Y is the response variable
xy	(bigr.matrix) : The training set
directory	(character) : Path on HDFS to store the learned model
laplace	(numeric) : Laplace correction term

Value

a bigr.naive.bayes object encapsulating the trained model

Usage

bigr.naive.bayes(formula, xy, directory, laplace)

bigr.persist — *Copy a Big R object to a named persistent dataset.*

Description

Copy the contents of a Big R object (a bigr.vector, bigr.frame, or a bigr.list) into a named persistent object, and return a second Big R object that references the newly created dataset.

Usage

```
bigr.persist(dataset, dataSource = "DEL", dataPath, header = FALSE,
  delimiter = ",", useMapReduce)
```

Arguments

dataset	a bigr.frame, bigr.vector or bigr.list
dataSource	the output data source. Only relevant for bigr.vectors and bigr.frames, and if specified, the value must be "DEL".
dataPath	the output data location
header	whether or not the output will contain headers. Only relevant for bigr.frames
delimiter	the delimiter character. Only relevant for bigr.frames.
useMapReduce	If TRUE, the newly created object will use MapReduce as the access mechanism.

Value

A newly created bigr object of the same class as the original

Examples

```
## Not run:

# Save contents of a derived bigr.frame to a file
air2 <- air[air$UniqueCarrier %in% c("UA", "HA"), c(1,2,3,5:9)]
bf <- bigr.persist(air2, dataSource = "DEL", dataPath = "/tmp/myair.csv",
                header = T, delim = ",", useMapReduce = F)

# Save list of unique airports to a file
bf <- bigr.persist(unique(air$Origin),
                dataPath="/tmp/airportcodes.csv",
                useMapReduce = F)

## End(Not run)
```

bigr.pull *Pull cluster-resident objects to the R client.*

Description

Pull one or more cluster-resident objects to the client

Usage

```
bigr.pull(x)
```

Arguments

x	(bigr.list)

Details

A bigr.list is constructed via a call to groupApply(). It holds references to one or more R objects that are stored on the cluster in a serialized form. With bigr.pull(), these objects can be transferred to the client, where they are deserialized and presented as a regular R list, or as a singleton object.

as.list is similar to bigr.pull. The only difference is that as.list always returns a list even for singleton objects.

Value

A single R object, or a list of R objects

See Also

bigr.list, groupApply, as.list

Examples

```
## Not run:

# See groupApply for examples on using bigr.pull.
#
## End(Not run)
```

bigr.random *Generate a random number*

Description

Generate a random number between 0 (inclusive) and 1 (exclusive)

Arguments

x	a bigr.vector

Value

a numeric bigr.vector

Examples

```
## Not run:

# Add a new column that has 5 randomly generated values (0 through 4)
bf <- air
bf$batch <- as.integer(bigr.random() * 5)

## End(Not run)
```

bigr.readLogisticRegressionFromHDFS *Read a Logistic Regression object stored on a given location on HDFS*

Description

With this method one can read a Logistic Regression model stored on HDFS. See bigr.logistic.regression for more on learning and storing a Logistic Regression model on HDFS

Usage

```
bigr.readLogisticRegressionFromHDFS(directory)
```

Arguments

directory	(character) A string representing the path to the stored model directory

Value

A bigr.logistic.regression object

Usage

lr <- bigr.readLogisticRegressionFromHDFS(directory)

---

bigr.readNaiveBayesFromHDFS
Read a Naive Bayes object stored on a given location on HDFS

Description

With this method one can read a Naive Bayes model stored on HDFS. See bigr.naive.bayes for more on learning and storing a Naive Bayes model on HDFS

Usage

```
bigr.readNaiveBayesFromHDFS(directory)
```

Arguments

directory	(character) A string representing the path to the stored SVM model directory

Value

A bigr.svm object

Usage

nb <- bigr.readNaiveBayesFromHDFS(directory)

---

bigr.readSvmFromHDFS *Read an SVM object stored on a given location*

Description

With this method one can read an SVM model stored on HDFS. See bigr.svm for more on learning and storing an SVM model on a cluster

Usage

```
bigr.readSvmFromHDFS(directory)
```

Arguments

directory	(character) A string representing the path to the stored SVM model directory

Value

A bigr.svm object

Usage

```
svm <- bigr.readSvmFromHDFS(directory)
```

---

bigr.reconnect *Reconnect to BigInsights*

Description

Attempt to re-establish a JDBC connection to Big SQL using the same credentials that were specified in an earlier bigr.connect() call. If no prior call to bigr.connect() had been made, an exception is raised. If there exists an active connection, it is dropped and a new connection is attempted.

Usage

```
bigr.reconnect()
```

Value

an invisible boolean indicating whether or not the connection was successful

See Also

bigr.connect, bigr.disconnect, is.bigr.connected

Examples

```
## Not run:
bigr.reconnect()
## End(Not run)
```

---

bigr.rmfs *Remove a file or directory from cluster*

Description

Remove a file or directory from cluster

Usage

```
bigr.rmfs(path, force = F, recursive = T)
```

Arguments

`path`	(character) the path to the file or directory on the cluster's hadoop file system force (logical) if TRUE remove without warning recursive (logical) if TRUE recursively remove all subdirectories

---

`bigr.sample` *Random sampling*

Description

Generate one or more samples from a bigr.frame

Usage

```
bigr.sample(data, perc)
```

Arguments

`data`	(bigr.frame) The data to sample from
`perc`	(numeric) For random sampling, an atomic value between (0, 1) that represents the sampling percentage. For partitioned sampling, a vector of numerics in the interval (0, 1), such that their sum is exactly 1.

Details

Two sampling methods are supported:

1. Random sampling: Generate a random subset of the given bigr.frame with the specified size in the form of a percentage.

2. Partitioned sampling: Split the given bigr.frame into the specified number of randomly generated non-overlapping subsets.

Value

For random sampling, a single bigr.frame is returned. For partitioned sampling, a list of bigr.frames is returned, and each bigr.frame in the list represents a partition.

See Also

bigr.persist

Examples

```
## Not run:

# Generate a 10% random sample of data
airs <- bigr.sample(air, 0.1)
bigr.persist(airs, dataSource = "DEL", dataPath = "airline_10percent.csv",
           header = T, delimiter = ",")

# Randomly split the data into training (70%) and test (30%) sets
airsplit <- bigr.sample(air, c(0.7, 0.3))
nrow(airsplit[[1]]) / nrow(air)
nrow(airsplit[[2]]) / nrow(air)

# Randomly split the data for 10-fold cross-validation
aircv <- bigr.sample(air, rep(0.1, 10))

## End(Not run)
```

---

bigr.set.server.option *Set a Hadoop/MapReduce option*

Description

Big R statements are translated into JaQL and Big SQL statements, which are then turned into MapReduce jobs. Execution of these jobs is influenced by numerous Hadoop / MapReduce options. These options are usually specified in site-wide configuration files on the BigInsights cluster. They can also be specified on a per-job basis. This function provides an easy way to set these options from Big R itself, and these setting only apply to the current Big R session.

Usage

```
bigr.set.server.option(option, value)
```

Arguments

option	the option string
value	the value

See Also

bigr.get.server.option

Examples

```
## Not run:
bigr.set.server.option("mapred.reduce.tasks", 10)
## End(Not run)
```

bigr.setRowLimit *Control printing of rows and elements*

Description

See Control printing of rows and elements

Usage

```
bigr.setRowLimit(rowLimit)
```

bigr.svm *Train an SVM model*

Description

This method allows to train a Support Vector Machine from a given bigr.matrix using the specified parameters. The model is automatically exported to the given path and it can be reloaded in a different R session.

Usage

```
bigr.svm(formula, xy, directory, binaryClass = FALSE, intercept = FALSE,
  tolerance = 0.001, maxIterations = 100, regularizer = 1)
```

Arguments

formula	(formula) : A formula in the form "Y ~ ." specifying the response variable in the left side
xy	(bigr.matrix) : The training dataset
directory	(character) : Input or output directory for the SVM model. If the user provides both xy and formula, a new model will be built and stored on directory. Otherwise, the model on directory will be read and returned as a bigr.svm object.
intercept	(logical) : A logical value specifying whether or not to include an intercept term in the model
tolerance	(numeric) : A tolerance value to control the termination of the learning algorithm
maxIterations	(numeric) : The number of iterations that the learning algorithm will execute for
regularizer	(numeric) The regularization parameter

Value

a bigr.svm object

bigr.transform	*Clean and transform data before invoking statistical algorithms. Transforms bigr.frame into a bigr.matrix with recode, missing value imputation, dummycode and binning.*

Description

Performs various data cleaning and transformation operations on a bigr.frame. The user can choose from a variety of operations including imputation of missing values, recoding attributes, binning attributes, dummy-coding attributes and scaling transformations. Besides returning a new bigr.frame containing the transformed data, bigr.transform also attaches the transformPath to the returned bigr.matrix. The transformPath can be subsequently used to perform the same transformations on other bigr.frames.

Usage

```
bigr.transform(bf, outData, transformPath, applyTransformPath = NULL,
  recodeAttrs = NULL, missingAttrs = NULL, imputationMethod = NULL,
  binningAttrs = NULL, numBins = NULL, binningMethod = NULL,
  dummycodeAttrs = NULL, scalingAttrs = NULL, scalingMethod = NULL)
```

Arguments

bf	(bigr.frame) The data to be transformed
outData	(character) Path on HDFS where the transformed data will be stored
transformPath	(character) Path on HDFS where all files related to this transformation (e.g., recode maps, dummy-code maps etc.) will be stored
applyTransformPath	(character) Path of the previously created transformation related files to be used to transform bf
recodeAttrs	(character) List of attributes to be recoded
missingAttrs	(character) List of attributes containing missing values that the user would like to fill-in using imputation
imputationMethod	(character) String value or list containing the imputation methods to be used to fill-in missing values. Note that the order of imputation methods should coincide with the order of the attributes specified in missingAttrs. In case a single string is provided, then that method will be used to fill-in missing values in all attributes specified in missingAttrs. Current options for imputation method include "global_mean" and "mode"
binningAttrs	(character) List of (scale) attributes to be binned
numBins	(integer) Integer or list of integers denoting the number of bins to use. Note that the order of integers specified should coincide with the order of the attributes specified in binningAttrs. In case a single integer is provided, then that bin count will be used to bin all attributes specified in binningAttrs.

binningMethod	(character) String or list specifying binning methods to be used. Note that the order of binning methods should coincide with the order of the attributes specified in binningAttrs. In case a single string is provided, then that method will be used to bin all attributes specified in binningAttrs. Current options for binning method includes "equi-width"
dummycodeAttrs	(character) List of attributes to dummy-code
scalingAttrs	(character) List of attributes to transform using scaling
scalingMethod	(character) String or list specifying the scaling methods to be used. Note that the order of scaling methods should coincide with the order of the attributes specified in scalingAttrs. In case a single string is provided, then that method will be used to scale all attributes specified in scalingAttrs. Current options for scaling methods include "mean-subtraction" and "z-score"

Details

Method to transform a bigr.frame to a matrix that can be read by DML

Each attribute in inData can be named at most once in any of the input arguments that take a list of attributes (missingAttrs, recodeAttrs, binningAttrs, dummycodeAttrs, scalingAttrs). Besides creating a bigr.frame that contains the transformed data, bigr.transform also attaches the transformPath to the returned bigr.matrix. This helps maintain a lineage of the transformations that were performed. It can be passed to any model that is learnt from the transformed bigr.frame. This allows access to both model coefficients and the transformations via the same model object.

Value

bigr.matrix Points to the recoded data and corresponding transformPath.

---

bigr.univariateStats *Univariate statistics*

Description

This method computes univariate statistics for a given `bigr.matrix`. Different statistics are calculated for each column according to their data types.

For scalar attributes, the following statistics will be computed: Minimum, maximum, range, mean, variance, sandard deviation, standard error of mean coefficient of variation, skewness, kurtosis, standard error of skewness, standard error of kurtosis, median, and interquartile mean.

For nominal attributes, the following statistics will be computed: Number of categories and number of modes.

Usage

```
bigr.univariateStats(data)
```

Arguments

`data`	(bigr.matrix) The dataset which statistics will be calculated from.

Value

A `data.frame` with the computed statistics.

---

bigr.vector — *Class bigr.vector*

Description

This class represents a single column of data. It is not meant to be instantiated directly by the user, and is instead derived from a bigr.frame.

Examples

```
## Not run:

origins <- air$Origin
print(class(origins))

## End(Not run)
```

---

coef — *Get the coefficients of a learned model*

Description

With this method we can get the coefficients of a learned model

Usage

```
## S4 method for signature 'bigr.svm'
coef(object)
```

Arguments

`object`	(bigr.svm)

Value

A data.frame representing the coefficients of a learned SVM model

Usage

```
coef(svm)
```

colnames, colnames<- *Set and get column names of a bigr.frame*

Description

With this pair of methods, one can get and set column names of a bigr.frame.

Usage

```
colnames(x, do.NULL = TRUE, prefix = "col")
```

Arguments

x	(bigr.frame)
value	(character) A vector that has the same length as the number of columns of the bigr.frame

Value

For colnames, return a character vector holding the column names. For colnames<-, return an updated object.

Usage

```
colnames(x)
names(x)
colnames(x) <- value
names(x) <- value
```

Examples

```
## Not run:

# Get column names
colnames(air)

air2 <- air[air$UniqueCarrier == "DL", c("UniqueCarrier", "Distance")]
colnames(air2) <- c("Code", "Dist")
str(air2)

## End(Not run)
```

coltypes, coltypes<- *Set and get column types of a bigr.frame*

Description

With this pair of methods, one can get and set column types of a bigr.frame.

Usage

```
coltypes(x)
```

Arguments

x	(bigr.frame)
value	(character) A vector that has the same length as the number of columns of the bigr.frame

Details

Legal column types are "character", "integer", "numeric" and "logical". coltypes<- can only be used to assign types on bigr.frame objects created directly on delimited files (dataSource = "DEL".) One cannot alter column types of bigr.frames created on Big SQL tables (dataSource = "BIGSQL"), JSON files (dataSource = "JSON") or frames that have been derived from other frames.

Value

For coltypes, return a character vector holding the column types. For coltypes<-, return an updated object.

Usage

```
coltypes(x)
coltypes(x) <- value
```

Examples

```
## Not run:

# Get column types
coltypes(air)

# Set column types
coltypes(air) <- ifelse(1:29 %in% c(9,11,17,18,23),
                        "character", "integer")

# Illegal column type assignment on derived big.frame
air2 <- air[air$UniqueCarrier == "DL", c("UniqueCarrier", "Distance")]
coltypes(air2) <- c("character", "numeric")                    # error
```

```
## End(Not run)
```

Column assignment operator
Add a new column to a bigr.frame

Description

$<- adds a new column to a given bigr.frame. The new column to be appended must have the same data origin as the original bigr.frame.

Usage

```
$(x, name) <- value
```

Arguments

x	(bigr.frame)
name	(character) name of the new column
value	(scalar value or bigr.vector) an expression representing the new column

Value

a new bigr.frame with the appended column

Examples

```
## Not run:

bf <- air
bf$FlightDelayed <- ifelse(bf$DepDelay >= 15, 1, 0)

## End(Not run)
```

Comparison operators *Comparison operators (>, >=, <, <=, ==, !=)*

Description

Big R supports relational operators (>, >=, <, <=, ==, !=) on two bigr.vectors, or on a bigr.vector and an atomic R value. The atomic values can be of type character, logical, numeric, or integer. These operators are typically used to construct filtering conditions on bigr.frames and bigr.vectors.

Arguments

x	a bigr.vector or an atomic R value
y	a bigr.vector or an atomic R value

Value

a bigr.vector of boolean values that represents the result of the comparison operation between each element in x and each corresponding element in y. If y is an R atomic data type, it is effectively replicated into a vector that has the same length as x.

Usage

```
x op y
```

Examples

```
## Not run:
air[air$UniqueCarrier == "HA",]

length(air$DepDelay[air$DepDelay >= 15])

## End(Not run)
```

Control printing of rows and elements

Control printing of rows and elements

Description

Control printing of rows and elements

Details

Get and set the maximum number of rows/elements to be displayed by the print() and show() functions on bigr.vectors and bigr.frames.

bigr.getRowLimit() gets the present value. The default is 50.

bigr.setRowLimit(rowLimit) sets the value to rowLimit.

See Also

print, show

cor *cor*

Description

See Variance, Covariance and Correlation

cov *cov*

Description

See Variance, Covariance and Correlation

Data coercion functions *Data coercion functions*

Description

Big R supports several data coercion functions that operate on bigr.vector objects.

Details

as.character turns a bigr.vector to a character bigr.vector.

as.numeric turns a bigr.vector to a numeric bigr.vector. If a value cannot be coerced into a numeric, an NA is returned.

as.integer turns a bigr.vector to an integer bigr.vector. If a value cannot be coerced into an integer, an NA is returned.

as.logical turns a bigr.vector to a logical bigr.vector. If a value cannot be coerced into an logical, an NA is returned.

dataPath *Get the "dataPath" property of a bigr.frame*

Description

Return the data path of a bigr.frame.

Usage

```
dataPath(bf)
```

Arguments

bf	a bigr.frame

Details

The data path is specified when the frame is first constructed.

Value

a character value representing the path

See Also

bigr.frame

dataSource *Get the "dataSource" property of a bigr.frame*

Description

Return the data source of a bigr.frame

Usage

```
dataSource(bf)
```

Arguments

bf	a bigr.frame

Details

The data source is specified when the frame is first constructed. For base frames, the value is either "DEL" or "BIGSQL". For frames that are derived from other bigr.frames, a value of "TRANSF" is returned. JSON frames also use a value of "TRANSF" because these frames are created as a result of query transformations on the underlying JSON files.

Value

"DEL", "BIGSQL" or "TRANSF"

See Also

bigr.frame

---

delimiter *Get the "delimiter" property of a bigr.frame*

Description

Return the delimiter used by a bigr.frame

Usage

```
delimiter(bf)
```

Arguments

bf	a bigr.frame

Details

The delimiter may be specified when the frame is first constructed.

Value

the delimiter of a bigr.frame

See Also

bigr.frame

dim *Dimemsions of a bigr.frame*

Description

Returns the dimensions of a bigr.frame

Usage

```
dim(x)
```

Arguments

x	a bigr.frame

Value

A two-element vector representing the number of rows and columns respectively.

dimnames *Dimension names of a bigr.list*

Description

Dimension names of a bigr.list

Usage

```
dimnames(x)
```

Arguments

x	(bigr.list) object

Value

a data.frame representing the dimension names

See Also

bigr.list

groupApply *Partitioned execution via column groups*

Description

groupApply partitions its input and applies an R function to each partition. The partitioning of the data and execution of the functions all happen on the server and results are retained on the server as well.

Usage

```
groupApply(data, groupingColumns, rfunction, signature, ..., numBatches)
```

Arguments

data	bigr.frame object
groupingColumns	a bigr.vector or list of bigr.vector objects
rfunction	function to apply to each partition
signature	If specified, this is a data.frame that has the same structure as the return value of "rfunction". This only needs to be specified when returning tabular data.
...	(optional) parameters that are passed to 'rfunction'
numBatches	If specified, indicates the # of partitions to divide the entire input (groupingColumns not specified), or each group of the input (groupingColumns also specified)

Details

There are several mechanisms to partition the data: via one or more columns, via a fixed number of batches, or both. To process the partitions, an instance of R is invoked for each partition and data from the partition is passed to R. The function is invoked on the data and results are consolidated. Depending on the configuration of the underlying cluster, the partitions can be processed in parallel on different nodes of the cluster. Results generated from each individual partition could be tabular (i.e. returned as a data.frame) or complex (i.e. any arbitrary R object). Tabular data is consolidated and presented as a bigr.frame, while R objects are assembled in a bigr.list. When returning tabular data, Big R needs to know the shape of the output, and this information can be provided via an input "signature".

Value

For tabular data, the return value is a bigr.frame. For non-tabular data, the return value is a bigr.list.

Configuration

groupApply partitions the underlying data source and attempts to process each partition in a Hadoop "reducer". How many partitions will be actually run concurrently depends on the configuration of the underlying cluster. Specifically, the options "mapred.reduce.tasks" and "mapred.tasktracker.reduce.tasks.maximum" play a big role. A user can set these settings from Big R itself. Depending on how long your R functions are designed to run, there may be a need to increase the MapReduce timeout via "mapred.task.timeout".

Error handling and diagnostics

groupApply can fail for several reasons. The most common error scenario faced by beginning users is when the R interpreter or the "bigr" package itself are not properly installed on all nodes of the BigInsights cluster. In addition, groupApply may also fail if any of the underying MapReduce jobs fail to execute. Should any of these events happen, groupApply issues a stack trace that describes the symptoms of the problem. In some cases, it may be necessary to examine the Big SQL server logs (bigsql.loq) to obtain additional information on the problems. Should there be a problem in executing MapReduce jobs, it may be necessary to examine the job logs via services such as Hadoop JobTracker.

In many cases, one or more instances of the user-specified R function raise errors. These errors, along with all output that the R function(s) may produce on stdout and stderr, are captured. These can be accessed by bigr.logs().

IMPORTANT - Partition sizes and memory considerations

Given that Big R invokes the R interpreter for each partition, it is important to ensure that each partition is only as big as can be handled by the R instance. For partitions that are too large, R may run out of memory, and this will cause execution of the partition to fail. In such cases, you may considing using sampling or other means to reduce partition size. In addition, depending on average partition sizes, one may want to tune the degree of parallelism (see Configuration above.)

See Also

bigr.logs, bigr.frame, bigr.list, bigr.set.server.option

Examples

```
## Not run:

# Only select two airlines
air <- air[air$UniqueCarrier %in% c("UA", "HA"),]

# Returning tabular-data as a bigr.frame
means <- groupApply(air, air$UniqueCarrier,
                    function(df) {
                        return (data.frame(df$UniqueCarrier[1],
                                    mean(df$Distance,
                                    na.rm=T)))
                    },
                    signature = data.frame(carrier = "anystring",
                                    avg = 1.0))
```

```
print(means)

bigr.logs()

bigr.logs(1)

# Returning non-tabular data such as complex R objects
models <- groupApply(data = air,
                 groupingColumns=list(air$UniqueCarrier),
                 rfunction = function(df) {
                  predcols <- c('ArrDelay', 'DepDelay', 'DepTime',
                               'CRSArrTime', 'Distance')
                   m <- lm(ArrDelay ~ ., df[,predcols])
                   coef(m)
                 })
print(models)

model.ha <- bigr.pull(models$HA)

print(model.ha)

bigr.logs(models, 1:2)


## End(Not run)
```

head *Return the first part of an object*

Description

head returns the first n elements of a bigr.vector or a bigr.frame. If n is not specified, a default value 6 is used.

Usage

```
head(x, ...)
```

Arguments

x	(bigr.vector or bigr.frame)
n	(integer) the number of elements to be returned

Details

If the dataset is derived from an original bigr.frame via transformations such as filtering, projections, arithmetic operations, etc.), any ordering inherent in the dataset as stored on disk may be lost. In such an event, the rows that are returned may not necessarily be the first n rows.

Value

Another bigr.vector or bigr.frame that is a subset of the original object

Usage

```
head(x, n = 6)
```

---

ifelse *Conditional element selection*

Description

ifelse accepts a logical bigr.vector, "test", and returns another bigr.vector which is filled with elements selected from either "yes" or "no" depending on whether the element of "test" is TRUE or FALSE.

Usage

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

Arguments

test	a logical bigr.vector specifying a condition
yes	a bigr.vector or scalar value that is chosen if test is TRUE
no	a bigr.vector or scalar value that is chosen if test is FALSE

Details

ifelse essentially allows recoding elements of a bigr.vector based on a given condition. The result is a new bigr.vector while the original bigr.vector remains intact. Nested ifelse constructs are also supported, and recode conditions may contain different columns from the same bigr.frame.

Value

a recoded bigr.vector

Usage

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

Examples

```
## Not run:
# Make a working copy of a bigr.frame and add a new column to it
bf <- air
bf$FlightDelayed <- ifelse(bf$DepDelay >= 15, 1, 0)

## End(Not run)
```

is.bigr.connected *Check connection to BigInsights*

Description

Check whether an active connection still exists to Big SQL server

Usage

```
is.bigr.connected()
```

Value

a boolean indicating whether the connection is active or not.

See Also

bigr.connect, bigr.disconnect, bigr.reconnect

is.na *Determine missing elements in a bigr.vector*

Description

is.na compares each element of a bigr.vector to NA, and returns another bigr.vector that has TRUE or FALSE elements depending on whether the original element was NA or not.

Usage

```
is.na(x)
```

Arguments

x	a bigr.vector

Value

another bigr.vector

length *Length of a bigr object*

Description

Return the length of a Big R object. The length of a bigr.frame is the number of its columns. The length of a bigr.vector is the number of its elements. The length of a bigr.list is the number of objects held in the list

Arguments

x	a bigr.vector, bigr.frame or bigr.list

Usage

```
length(x)
```

Examples

```
## Not run:

# length of a bigr.frame (i.e. number of columns)
length(air)

# length of a bigr.vector (i.e. number of elements)
length(air[,"UniqueCarrier"])

## End(Not run)
```

Logical operators *Logical operators (&, |, !)*

Description

Big R supports logical operators (&, |, !). & indicates logical AND, | indicates logical OR, and ! indicates logical negation. & and | are supported on two bigr.vectors, or on a bigr.vector and an atomic R value of type logical. ! is a unary operator.

Arguments

e1	a bigr.vector or an R atomic data type of type "logical"
e2	a bigr.vector or an R atomic data type of type "logical"

Value

a bigr.vector of type "logical"

Examples

```
## Not run:

# Select all flights that were delayed by more than 15 minutes at
# departure or arrival.
airSubset <- air[! (air$Cancelled > 0)
                 & (air$DepDelay >= 15 | air$ArrDelay >= 15),]

## End(Not run)
```

Math functions — *Math functions*

Description

Big R supports several mathematical functions that operate on bigr.vector objects.

Usage

```
## S4 method for signature 'bigr.vector'
Math(x)
```

Arguments

x	a numeric bigr.vector

Usage

```
abs(x)
sign(x)
sqrt(x)
ceiling(x)
floor(x)
trunc(x)
log(x)
log10(x)
exp(x)
expm1(x)
cos(x)
sin(x)
tan(x)
acos(x)
asin(x)
atan(x)
```

---

mean *mean*

Description

See Mean and standard deviation

Mean and standard deviation
mean, sd

Description

Calculate the mean and standard deviation of a bigr.vector, or columns of a bigr.frame. When called on bigr.vector objects, the underlying datatype of the object must be numeric. Similarly, when called on bigr.frame objects, all columns must be numeric.

Arguments

x	An object of class bigr.frame or bigr.vector

Usage

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

sd(x, na.rm = TRUE)
```

NOTE: Unlike the identically-named R functions that operate on vector and data.frame objects, these functions do not consider rows or elements with NA values. Any "na.rm" parameter specification is ignored by these functions.

Examples

```
## Not run:

mean(air)

mean(air[,c("ArrDelay", "DepDelay")])

sd(air[,c("ArrDelay", "DepDelay", "DepTime", "ArrTime", "Distance")])


## End(Not run)
```

merge *Merge two bigr.frames*

Description

Merge two bigr.frames by common columns, or do other versions of database join operations.

Usage

```
merge(x, y, ...)
```

Arguments

x	the first bigr.frame to be joined
y	the second bigr.frame to be joined
by	a character vector specifying the join columns. If by is not specified, the common column names in x and y will be used.
by.x	a character vector specifying the joining columns for x. If by.x is not specified, the common column names in x and y will be used.
by.y	a character vector specifying the joining columns for y. If by.y is not specified, the common column names in x and y will be used.
all.x	a boolean value indicating whether all the rows in x should be including in the join
all.y	a boolean value indicating whether all the rows in y should be including in the join If all.x and all.y are set to FALSE, a natural join will be returned. If all.x is set to TRUE and all.y is set to FALSE, a left outer join will be returned. If all.x is set to FALSE and all.y is set to TRUE, a right outer join will be returned. If all.x and all.y are set to TRUE, a full outer join will be returned. In this case, columns in by.x and by.y will not be shown duplicated, but they will be combined discarding non-existing values.

Details

CAUTION: This is an experimental method that does not perform well when using MapReduce! In order to join large datasets, it is advisable to use Big SQL and materializing the merged data into a persistent dataset.

Value

The merged bigr.frame

Migration and backward compatibility
Migration and backward compatibility

Description

Each release of Big R contains a mix of defect fixes, performance enhancements and functional changes. Unless otherwise noted, these changes are backward compatible. However, some internal changes made to R object structures can render previously saved R objects incompatible. Any old objects that were saved using the prior versions of the product must be discarded. These objects could have been explicitly saved via save() and save.image() functions. They could also have been saved in .RData files that can be created when a user exits the R environment. Using these saved objects from a prior release would result in errors.

---

na.exclude *Remove missing elements from an object*

Description

See na.omit

Usage

```
na.exclude(object, ...)
```

---

na.omit *Omit missing elements from an object*

Description

Omit missing elements from an object. This is identical to na.exclude.

Usage

```
na.omit(object, ...)
```

Arguments

object	a bigr.frame or bigr.vector

Details

Given a bigr.vector, this function returns a new bigr.vector without any NA elements.

Given a bigr.frame, this function returns a new bigr.frame without any rows that contain NA values.

Value

Another bigr.vector or bigr.frame with NAs removed

See Also

na.exclude

---

na.string — *Get the "na.string" property of a bigr.frame*

Description

Return the na.string value of a bigr.frame

Usage

```
na.string(bf)
```

Arguments

bf	a bigr.frame

Details

na.string may be specified when the frame is first constructed.

Value

a character vector

See Also

bigr.frame

names — *Dimension names of a bigr.list*

Description

Dimension names of a bigr.list

Usage

```
names(x)
```

Arguments

x	(bigr.list) object

Value

a data.frame representing the dimension names

See Also

bigr.list

ncol — *Number of columns of a bigr.frame*

Description

Return the number of columns of a bigr.frame

Usage

```
ncol(x)
```

Arguments

x	a bigr.frame

nrow — *Number of rows of a bigr.frame*

Description

Return the number of rows of a bigr.frame

Usage

```
nrow(x)
```

Arguments

x	a bigr.frame

Details

The first time this method is called on a bigr.frame, the return value is cached internally. Subsequent calls return the cached value.

predict — *Model Predictions*

Description

Model Predictions

Usage

```
predict(object, ...)
```

See Also

predict.logistic.regression

predict.bigr.svm

predict.bigr.naive.bayes

predict.bigr.glm

predict.bigr.lm

predict.bigr.kmeans

predict.bigr.glm *Scoring/Testing Generalized Linear Models*

Description

This method allows to score/test a GLM model for a given bigr.matrix. If the test set is labeled (label column is present), testing will be done. Otherwise, scoring will be performed.

Usage

```
predict.bigr.glm(object, data, directory, family, dispersion = 1)
```

Arguments

object	(bigr.glm) a trained model
data	(bigr.matrix) the testing set
directory	(character) a HDFS/GPFS path where the results of scoring/testing will be stored
family	(family) the distribution family and link function. Supported families are "gaussian", "poisson", "Gamma", and "inverse.gaussian". Supported link functions are "identity", "inverse", "log", "sqrt", "1/mu^2", "logit" "probit", "cloglog", and "cauchit"
dispersion	(Over-) dispersion value

Value

Refer to the algorithm reference documentation for further details about the output of this function.

predict.bigr.kmeans *Testing and scoring of a Kmeans model*

Description

This method allows to score/test a Kmeans model for a given bigr.matrix.

Usage

```
predict.bigr.kmeans(object, data, directory)
```

Arguments

object	(bigr.kmeans) The Kmeans model
data	(bigr.matrix) The data to be tested or scored.
directory	Name of the output directory

Value

Return a data.frame that computes statistics

predict.bigr.lm *Scoring/Testing Linear Models*

Description

This method allows to score/test a linear regression model for a given bigr.matrix. If the testing set has the label column, testing will be done. Otherwise, scoring will be performed.

Usage

```
predict.bigr.lm(object, data, directory)
```

Arguments

object	(bigr.lm) The linear regression model
data	(bigr.matrix) The data to be tested or scored.
directory	(character) The directory to keep the predict output.

Value

A list of two bigr.frame objects containing the prediction values and the statistics, respectively.

predict.bigr.naive.bayes *Scoring/testing a Naive Bayes model*

Description

This method allows to score/test a Naive Bayes model on a given bigr.matrix. It computes the probabilities of each class for each example. If the given testing set is already labeled, the confusion matrix and overall accuracy are also computed.

Usage

```
predict.bigr.naive.bayes(object, data, outputDir, returnProbabilities)
```

Arguments

object	(bigr.naive.bayes) : Naive Bayes model
data	(bigr.matrix) : Testing set with an optional label column
outputDir	(character) : Path on HDFS/GPFS to store the model
returnProbabilities	(boolean) : A boolean value indicating whether to return each class' probabilities for each example

Value

a list containing probabilities, accuracy and ctable

predict.bigr.svm *Scoring/testing an SVM model*

Description

This method allows to score/test an SVM model for a given bigr.matrix. If the testing set has the label column, testing will be done. Otherwise, scoring will be performed.

Usage

```
predict.bigr.svm(object, data, outputDir, returnScores)
```

Arguments

object	(bigr.svm) : A trained SVM model
data	(bigr.matrix) : The testing dataset
outputDir	(character) :
returnScores	(logical) : A logical value to control whether to return the scores or not
directory	the HDFS/GPFS path where the model will be stored

Value

a list containing scores, accuracy and ctable

predict.logistic.regression *Score a dataset using Logistic Regression model*

Description

Predict the probabilities of the rows of a given dataset belonging to different classes of labels

Usage

```
predict.logistic.regression(object, data, outputDir)
```

Arguments

object	(bigr.logistic.regression) : Logistic Regression model
data	(bigr.matrix) : Dataset with an optional label column
outputDir	(character) : Path to store the results

Value

a list containing probabilities the data rows of being various labels and their statistics

print *Print an object*

Description

Print a small subset of the contents of the object. For objects of class bigr.vector and bigr.frame, the number of rows printed can be controlled by calling bigr.setRowLimit().

Arguments

object	Any Big R object

Value

an invisible NULL.

Usage

```
print(object)
```

See Also

show

Examples

```
## Not run:
bf <- bigr.frame(dataPath = "syscat.tables", dataSource = "BIGSQL")
print(bf)
## End(Not run)
```

rowApply *Partitioned execution via batches of rows*

Description

rowApply partitions its input into batches of rows, and applies an R function to each batch. The batching of the data and execution of the functions all happen on the server and results are retained on the server as well. Big R attempts to assign the same # of rows to each batch. In some cases, however, the number of rows in a batch may be less than the requested number. This can happen when a batch happens to be the last one in an "input split."

Usage

```
rowApply(data, rfunction, signature, ..., numRowsPerBatch = 10000)
```

Arguments

data	bigr.frame object
rfunction	function to apply to each partition
signature	a data.frame that has the same structure as the return value of "rfunction".
...	(optional) parameters that are passed to 'rfunction'
numRowsPerBatch	Indicates the # of rows per batch. In some cases, the # of rows in the batch may be less that this parameter.

Details

The semantics of rowApply() make it an ideal for scenarios wherein one wants to loosely partition data, and each row is generally independently processed by the user-specified R function. On application that fits this profile involves "model scoring".

To process the batches, an instance of R is invoked for each batch and data from the batch is passed to R. The function is invoked on the data and results are consolidated. Depending on the configuration of the underlying cluster, the batches can be processed in parallel on different nodes of the cluster. Unlike groupApply, results generated from rowApply are always tabular. The user-supplied R function is expected to return a data.frame, and tabular data from all batches is consolidated and presented as a bigr.frame. Big R needs to know the shape of the output, and this information can be provided via an input "signature".

Value

a bigr.frame.

Configuration

Unlike groupApply, rowApply contructs the batches (i.e. partitions) in the Hadoop "mappers". How many partitions will be actually run concurrently depends on the configuration of the underlying cluster. Specifically, the options "mapred.map.tasks" and "mapred.tasktracker.map.tasks.maximum" play a big role. A user can set these settings from Big R itself. Depending on how long your R functions are designed to run, there may be a need to increase the MapReduce timeout via "mapred.task.timeout".

In the current implementation, rowApply first constructs the batches in memory inside the JaQL environment, and then passes the batches to R. As such, one ought to set the 'numRowsPerBatch' carefully. Too high a number may cause MapReduce jobs to run out of memory. Too low a number will spawn a large # of R instances (i.e. one per batch) and increase execution overhead.

Error handling and diagnostics

rowApply can fail for several reasons. The most common error scenario faced by beginning users is when the R interpreter or the "bigr" package itself are not properly installed on all nodes of the BigInsights cluster. In addition, rowApply may also fail if any of the underying MapReduce jobs fail to execute. Should any of these events happen, rowApply issues a stack trace that describes the symptoms of the problem. In some cases, it may be necessary to examine the Big SQL server logs (bigsql.log) to obtain additional information on the problems. Should there be a problem in executing MapReduce jobs, it may be necessary to examine the job logs via services such as Hadoop JobTracker.

In many cases, one or more instance of the user-specified R function raise errors. These errors, along with all output that the R function(s) may produce on stdout and stderr, are captured. These can be accessed by bigr.logs().

See Also

bigr.logs, bigr.frame, bigr.set.server.option

Examples

```
## Not run:

air2 <- air[air$UniqueCarrier %in% c("UA", "HA"), c(1,2,3,5:9)]
bf <- rowApply(air2,
            function(df) {
                data.frame(nrow(df))
            },
            signature=data.frame(nrow = 1),
            numRowsPerBatch = 10000)

## End(Not run)
```

sd *sd*

Description

See Mean and standard deviation

show *Generic function to return generic function to output the bigr.lm model*

Description

Generic function to return generic function to output the bigr.lm model

Usage

```
show(object)
```

show — *Show an object*

Description

Print a small subset of the contents of the object. For objects of class bigr.vector and bigr.frame, the number of rows printed can be controlled by calling bigr.setRowLimit().

Arguments

object	Any Big R object

Value

an invisible NULL.

Usage

```
show(object)
```

See Also

print

Examples

```
## Not run:
bf <- bigr.frame(dataPath = "syscat.tables", dataSource = "BIGSQL")
show(bf)
## End(Not run)
```

sort — *Sort a bigr.frame*

Description

Sort a bigr.frame by the given column(s).

Arguments

bf	a bigr.frame to be sorted
by	a list of columns (i.e., bigr.vector) to sort by, or a vector of character values containing the column names to sort by
decreasing	an array of logical values specifying whether each column sort should be sorted in increasing or decreasing order

Value

a sorted bigr.frame

Examples

```
## Not run:

sort(air, by = list(air$Origin, air$Dest), dec=c(T,F))[,c("Origin","Dest")]

## End(Not run)
```

---

sort *Sort a bigr.vector*

Description

Sorts a `bigr.vector` in increasing or decreasing order.

Arguments

x	a `bigr.vector`
decreasing	(logical) If TRUE, the bigr.vector should be sorted in decreasing order.

Value

a sorted bigr.vector

---

stats *Get the statistics data of a learned model*

Description

With this method we can get the statistics of a learned model represented by the object

Usage

```
## S4 method for signature 'bigr.lm'
stats(object)
```

Arguments

object	(bigr.lm) The linear regression model

Value

A data.frame with the statistics data for the learned model

---

str *Compactly display the structure of a bigr.frame*

Description

Display the structure of a bigr.frame including column names, column types as well as a small sample of rows. The total number of rows may also be displayed as long as that value was previously computed and cached.

Usage

```
str(object, ...)
```

Arguments

x	a bigr.frame

---

str *Compactly display the structure of a bigr.vector*

Description

Display the structure of a bigr.vector including data type as well as a small sample of elements.

Usage

```
str(object, ...)
```

Arguments

x	a bigr.vector

String functions *String functions*

Description

Big R supports several string manipulation functions that operate on bigr.vector objects.

Details

toString turns a bigr.vector to a character bigr.vector.

toupper and tolower turn characters in big.vectors to uppercase and lowercase respectively.

nchar takes a character bigr.vector as an argument and returns a bigr.vector whose elements contain the sizes of the corresponding elements of x.

substr and substring extract substrings from a character bigr.vector. Unlike equivalently named functions in R, these functions are 0-indexed.

grepl searches for matches of the argument pattern within each element of a character vector.

sub and gsub performs replacement of all matches of a character string.

Examples

```
## Not run:

toString(air$DepDelay)

tolower(air$Origin)

# Determine number of characters in each destination
nchar(air$Dest)

# Pick first character of UniqueCarrier
substr(air$UniqueCarrier, 0, 1)

# Select all destinations that start with "LA"
air[grepl("LA.*", air$Dest), "Dest"]

# Substitute "LAX" with "LAS"
gsub("LAX", "LAS", air$Dest[air$Dest == "LAX", "Dest"])


## End(Not run)
```

summary — *Compute descriptive statistics*

Description

This method computes basic statistics on a bigr.vector or a bigr.frame object. It has several overloaded variants. In its simplest form, the method accepts a single parameter of class bigr.vector or bigr.frame, and produces several key statistics such as max, min and count. The method can also be called with a formula that specifies the exact set of aggregate statistics on one or more columns of a bigr.frame.

Arguments

object	(bigr.vector or bigr.frame) The data itself
formula	(formula) Optionally describes a set of columns, aggregate functions to be calculated on each column, as well any grouping columns. A formula can only be specified when object is of class "bigr.frame"

Details

The variants of the method that accept one parameter compute a fixed set of statistics.

The "formula" variant, however, allows for greater flexibility as it supports the ability to compute grouped aggregate functions for the specified columns. For a given formula LHS ~ RHS, LHS should contain the aggregate functions that need to be computed whereas RHS should specify any grouping columns. If the aggregate functions are to be computed for the entire dataset (i.e., without grouping), RHS should be set to a dot (.) symbol. A column alone can be specified in the LHS, and that serves as a shorthand for a collection of aggregate functions that apply to the column's type.

Supported aggregate functions are: min, max, count, sum, avg, var and sd. Each function requires a single-parameter that corresponds to a column name in the bigr.frame. Column names cannot be the same as function names. NA values are implicitly ignored when computing the aggregate statistics on columns.

The count aggregate has two flavors. count(column) only counts rows where the column is not NA. In addition, count(.) can be used to count the total number of rows in a bigr.frame, or group. Note that count(column1) and count(column2) may yield different values because of differences in NA values across column1 and column2.

Value

a data.frame with the computed aggregate functions

Usage

```
summary(object)            # object is a bigr.vector or bigr.frame
summary(object, formula)   # object is a bigr.frame
```

Examples

```
## Not run:

#' Summarize a bigr.frame
summary(air[,c("DepDelay", "ArrDelay")])

#' Summarize a bigr.vector
summary(air$Distance)

# Count total # of rows (flights) in the entire dataset
summary(air, count(.) ~ .)

# Count # of flights where Distance != NA
summary(air, count(Distance) ~ .)

# Compute basic descriptive statistics (count, min and max) on a
# non-numeric column.
summary(air, UniqueCarrier ~ .)

# Compute basic descriptive statistics (count, min, max, sum, mean) on a
# numeric column.
summary(air, air$Distance ~ .)

# Compute mean and standard deviation of distance flown by each airline
summary(air, mean(Distance) + sd(Distance) ~ UniqueCarrier)

# Compute a mix of statistics on various columms, grouped
# by multiple columns (UniqueCarrier and Year)
summary(air, max(Distance) + mean(DepDelay) + ArrDelay
               ~ UniqueCarrier + Year)

## End(Not run)
```

---

table *Compute a contingency table for a bigr.vector*

Description

Computes a contingency table for a given bigr.vector. This is essentially a set of distinct values along with the number of occurrences of each value

Usage

```
table(..., exclude = if (useNA == "no") c(NA, NaN), useNA = c("no", "ifany",
  "always"), dnn = list.names(...), deparse.level = 1)
```

Arguments

x	a bigr.vector

Value

a vector with the contingency table of x

Examples

```
## Not run:

# Determine number many flights are flown by each carrier.
table(air$UniqueCarrier)

## End(Not run)
```

---

tableApply *Embedded execution on entire dataset*

Description

tableApply applies an R function to an entire bigr.frame.

Usage

```
tableApply(data, rfunction, signature, ...)
```

Arguments

data	bigr.frame object
rfunction	function to apply to the data
signature	If specified, this is a data.frame that has the same structure as the return value of "rfunction". This only needs to be specified when returning tabular data.
...	(optional) parameters that are passed to 'rfunction'

Details

The contents of the bigr.frame are passed to the R function as a single data.frame. The function execution happens on the cluster, and the results are retained on the server as well.

Conceptually, tableApply is similar to groupApply with just one group. As such, please see groupApply for important considerations regarding return values, error handling, diagnostics, and memory sizes.

Value

For tabular data, the return value is a bigr.frame. For non-tabular data, the return value is a bigr.list.

See Also

groupApply, bigr.logs, bigr.frame, bigr.list, bigr.set.server.option

Examples

```
## Not run:
air2 <- air[air$UniqueCarrier %in% c("UA", "HA"), ]

bl <- tableApply(air2,
                 function(df) {
                     cor(df[, c("ArrDelay", "DepDelay")],
                         use="complete.obs")
                 })

cormatrix <- bigr.pull(bl)
print(cormatrix)

## End(Not run)
```

---

tail *Return the last part of an object*

Description

tail returns the last n elements of a bigr.vector or a bigr.frame. If n is not specified, a default value 6 is used.

Usage

```
tail(x, ...)
```

Arguments

x	(bigr.vector or bigr.frame)
n	(integer) the number of elements to be returned

Details

If the dataset is derived from an original bigr.frame via transformations such as filtering, projections, arithmetic operations, etc.), any ordering inherent in the dataset as stored on disk may be lost. In such an event, the rows that are returned may not necessarily be the last n rows.

Value

Another bigr.vector or bigr.frame that is a subset of the original object

Usage

```
tail(x, n = 6)
```

Temporary Files — *Temporary Files and garbage collection*

Description

During the execution of Big R functions, temporary files may be generated. Some of these files are created on HDFS, while others are created on the local file system of the data nodes. All temporary files on HDFS are placed under the /tmp/bigr directory. Some of the functions that create these files are as.bigr.frame, groupApply, rowApply, tableApply, and bigr.sample. In many cases, the temporary files have a prefix that identifies the function that created them. The lifetime of these temporary files is dependent on the lifetime of the R objects that refer to them. For e.g., as.bigr.frame() creates a temporary file on HDFS, and the resulting bigr.frame object references this file. If this R object is no longer accessible via the R environment, it is garbage collected and the underlying temporary file is deleted. However, this mechanism doesn't work if the connection to BigInsights is lost. In such a case, the temporary files are left behind on the cluster, and they need to be manually cleaned up. Should there be a need to manually clean up such files on HDFS, one only needs to examine the /tmp/bigr directory.

unique — *Compute unique elements in a bigr.vector*

Description

Computes the set of distinct values in a given bigr.vector

Usage

```
unique(x, incomparables = FALSE, ...)
```

Arguments

x	a bigr.vector

Value

another bigr.vector with only the distinct values of x

Examples

```
## Not run:

# Compute the set of unique airport codes
unique(air$Origin)

## End(Not run)
```

useMapReduce — *Get the "useMapReduce" property of a bigr.frame*

Description

Indicates whether the specified bigr.frame is using the MapReduce mode.

Usage

```
useMapReduce(bf)
```

Arguments

bf	a bigr.frame

Details

The "useMapReduce" property is specified when the frame is first constructed.

Value

a logical

See Also

bigr.frame

Value Matching (%in%) — *Match values in a list*

Description

Compares each element of a `bigr.vector` with every element in a given `vector`. The given `bigr.vector` and the given `vector` must have the same data type.

Usage

```
x %in% table
```

Arguments

x	a bigr.vector
table	a vector of values to be compared with

Value

a bigr.vector of boolean values indicating whether each element in x belongs to table

Examples

```
## Not run:

air[air$UniqueCarrier %in% c("UA", "HA"), ]


## End(Not run)
```

---

var *var*

Description

See Variance, Covariance and Correlation

Variance, Covariance and Correlation *var, cov, cor*

Description

These functions compute the variance, covariance and correlation matrices respectively.

Arguments

x	An object of class bigr.frame or bigr.vector

Usage

```
var(x, y = NULL, na.rm = TRUE)
cov(x, y = NULL, use = "complete.obs", method = "pearson")
cor(x, y = NULL, use = "complete.obs", method = "pearson")
```

var computes the variance of a bigr.vector, or of columns in a bigr.frame When called on bigr.vector objects, the underlying datatype of the object must be numeric.

cov and cor compute the covariance and correlation matrices between columns of a specified bigr.frame. All columns of the specified bigr.frame object must be numeric.

NOTE: Unlike the identically-named R functions that operate on vector and data.frame objects, these functions do not consider rows or elements with NA values. var() ignores the "na.rm" parameter specification. cov() and cor() implicitly assume the "use" parameter to be "complete.obs", and "method" to be "pearson".

Examples

```
## Not run:

var(air$ArrDelay)

cov(air[,c("ArrDelay", "DepDelay")])

cor(air[,c("ArrDelay", "DepDelay", "DepTime", "ArrTime", "Distance")])


## End(Not run)
```

---

with — *Evaluate an R expression in an environment constructed from a bigr.frame*

Description

with() allows access to columns of a bigr.frame by simply referring to their name. It appends every column of a bigr.frame into a new environment. Then, the given expression is evaluated in this new environment.

Usage

```
with(data, expr, ...)
```

Arguments

data	(bigr.frame)
expr	(expression) the expression to be evaluated

Examples

```
## Not run:

# Determine number of flights delayed by more than 15 minutes
with(air, length(DepDelay[DepDelay >= 15]))

## End(Not run)
```

[*Filter rows and project columns of a bigr.frame*

Description

Filter rows and project columns of a bigr.frame

Usage

```
"["(x, i, j, ..., drop = TRUE)
```

Arguments

x	(bigr.frame) the object being operated on
i	(bigr.vector) a logical operation that represents the filtering condition
j	(character or integer) a vector representing columns to be projected. These could be column ids (i.e. integers) or column names (i.e. characters)

Value

the derived bigr.frame or bigr.vector

See Also

bigr.frame

Examples

```
## Not run:

air[air$UniqueCarrier %in% c("UA", "HA"), c(1,2,3,5:9)]

air[, c("Origin", "Dest")]

air[air$Dest == "SFO", 17]

## End(Not run)
```

[	*Filter a bigr.list*

Description

Filter a bigr.list via positionally-specified criteria and return the derived list.

Usage

```
"["(x, i, j, ..., drop = TRUE)
```

Arguments

x	(bigr.list) the parent object
i,j,...	(character) one or more values of the grouping column, specified positionally

Value

filtered bigr.list object

See Also

bigr.list, groupApply

Examples

```
## Not run:

air2 <- air[air$UniqueCarrier %in% c("UA", "HA")
            & air$Origin %in% c("SFO", "LAX"), ]

summary(air2[, c("UniqueCarrier", "Origin")], count(.) ~ UniqueCarrier + Origin)

biglist <- groupApply(air2, list(air2$UniqueCarrier, air$Origin),
                      function(df) {
                          nrow(df)
                      })

# Select all objects for carriers UA and HA, departing from LAX
biglist[c("UA", "HA"), "LAX"]

# Select all UA flights
biglist["UA", ]

## End(Not run)
```

[*Filter a bigr.vector*

Description

Filter a bigr.vector

Usage

```
"["(x, i, j, ..., drop = TRUE)
```

Arguments

x	(bigr.vector) the object being operated on
i	a bigr.vector with a logic operation that represents the the filtering condition. i and x must have the same data origin.

Value

the derived bigr.vector

See Also

bigr.vector

Examples

```
## Not run:

# Create a bigr.vector ...
air2 <- air[air$UniqueCarrier %in% c("UA", "HA"), "Origin"]

# ... and filter it
length(air2[air2 == "SFO"])


## End(Not run)
```

$ *Project a column of a bigr.frame*

Description

Project a column of a bigr.frame

Usage

```
"$"(x, name)
```

Arguments

x	(bigr.frame) object being operated upon
name	(character) name of the column being projected

Value

a bigr.vector

Examples

```
## Not run:
print(air$Dest)

## End(Not run)
```

$ *Filter a bigr.list*

Description

Filter a bigr.list and return a derived list back to the caller. This only works when original bigr.list has one grouping column.

Usage

```
"$"(x, name)
```

Arguments

x	(bigr.list) object being operated upon
name	(character) name of the grouping column

Value

the derived bigr.list

See Also

groupApply

Examples

```
## Not run:
# To see an example, check the documentation on groupApply()

## End(Not run)
```

Index