Package 'stream'

April 3, 2011

Version 0.0

Date 2010-09-29

Title Infrastructure for Data Streams

Author John Forrest, Michael Hahsler

Maintainer John Forrest <jhforrest@smu.edu>

Description A framework for data stream modelling and associated data mining tasks such as clustering and classification.

Depends R (>= 2.10.0), clusterGeneration, proxy

Imports rJava (>= 0.6-3), MASS

SystemRequirements Java (>= 5.0)

License GPL-2

R topics documented:

cluster . . . 2
DSClusterer . . . 2
DSC_Clustream . . . 3
DSC_ClusTree . . . 4
DSC_CobWeb . . . 5
DSC_DenStream . . . 6
DSC_MOA . . . 7
DSC_StreamKM . . . 7
DSC_tNN . . . 8
DSData . . . 9
DSD_DataFrame . . . 10
DSD_Gaussian_Static . . . 11
DSD_MOA . . . 12
DSD_ReadStream . . . 13
get_centers . . . 14
get_points . . . 15
reset_stream . . . 16
write_stream . . . 16

Index **18**

cluster *Clustering data based with input algorithm and data stream*

Description

Clusters a number of input points from a data stream into a clustering object.

Usage

```
cluster(dsc, dsd, n = 1000)
```

Arguments

dsc	a DSC object
dsd	a DSD object
n	number of points to cluster

Details

Clusters input data. The underlying clustering in the DSC object is implicitly updated as if the object was 'passed by reference' in a traditional programming language.

Value

The updated DSC object is returned invisibly for reassignment.

Examples

```
dsc <- DSC_DenStream()
dsd <- DSD_MOA()
cluster(dsc, dsd, 2000)
plot(dsc)
```

DSClusterer *DSClusterer*

Description

The abstract base class for all DSC classes.

Details

The DSClusterer class cannot be instantiated, but it serves as a base class from which all DSC objects inherit from.

DSClusterer provides several generic functions that can operate on all DSC subclasses: get_centers(), which returns the centers of the micro-clusters generated; nclusters(), which returns the number of micro-clusters generated; print(), which prints general data about the DSC; and plot(), which will plot the centers of the micro-clusters.

See Also

DSClusterer, DSC_CobWeb, DSC_ClusTree, DSC_StreamKM, DSC_DenStream, DSC_Clustream, DSData

---

`DSC_Clustream` *DataStreamClusterer: Clustream*

Description

Creates a new DataStreamClusterer with the Clustream algorithm

Usage

```
DSC_Clustream(timeWindow = 1000, maxNumKernels = 100)
```

Arguments

`timeWindow`	Defines the time window to be used in Clustream
`maxNumKernels`	Defines the maximum number of kernels used in Clustream

Details

Clustream was originally introduced in Aggarwal's paper cited below. Like most data stream clustering algorithms, Clustream summarizes the incoming data points into micro-clusters. At any given time, it can be assumed there are q micro-clusters maintained by Clustream. This means that when a new micro-cluster is created, another one must be deleted, and the opposite is true for when two micro-clusters merge with one another. A significant amount of thought in Clustream is given to the temporal structure of data streams. Snapshots of calculated micro-clusters are stored in a "pyramidal time frame" pattern. This allows the user to turn back to a specific location in the data stream to mine information from a particular time period.

The `timeWindow` parameter changes the granularity at which the snapshots are taken for the micro-cluster summary maintanence. `maxNumKernels` is analogous to the q micro-clusters that are maintained within the algorithm.

Value

`description`	The name of the algorithm in the DSC object.
`options`	The CLI params defined when creating the DSC object.
`javaObj`	The underlying Java object associated with DSC_MOA objects.

References

Aggarwal CC, Han J, Wang J, Yu PS (2003). "A Framework for Clustering Evolving Data Streams." In "Proceedings of the International Conference on Very Large Data Bases (VLDB '03)," pp. 81-92.

Bifet A, Holmes G, Pfahringer B, Kranen P, Kremer H, Jansen T, Seidl T (2010). "MOA: Massive Online Analysis, a Framework for Stream Classification and Clustering." In "Journal of Machine Learning Research (JMLR) Workshop and Conference Proceedings,"

See Also

DSClusterer, DSC_MOA, DSC_DenStream, DSC_CobWeb, DSC_StreamKM, DSC_ClusTree

Examples

```
dsc <- DSC_Clustream()
dsd <- DSD_Gaussian_Static()
cluster(dsc, dsd, 5000)
plot(dsc)
```

DSC_ClusTree *DataStreamClusterer: ClusTree*

Description

Creates a new DataStreamClusterer with the ClusTree algorithm

Usage

```
DSC_ClusTree(timeWindow = 1000, maxHeight=8)
```

Arguments

timeWindow	The timeWindow parameter used in MOA.
maxHeight	The maximum height of the tree.

Details

The authors of ClusTree define it as a "parameter free algorithm". Their innovation automatically adjusts to the speed of the incoming points generated by a data stream. ClusTree is able to detect concept drift and outliers in the stream being operated upon.

Value

A list of class DSC, DSC_MOA, and DSC_ClusTree. The list contains the following items:

description	The name of the algorithm in the DSC object.
options	The CLI params defined when creating the DSC object.
javaObj	The underlying Java object associated with DSC_MOA objects.

References

Philipp Kranen, Ira Assent, Corinna Baldauf, and Thomas Seidl. 2009. Self-Adaptive Anytime Stream Clustering. In Proceedings of the 2009 Ninth IEEE International Conference on Data Mining (ICDM '09). IEEE Computer Society, Washington, DC, USA, 249-258. DOI=10.1109/ICDM.2009.47 http://dx.doi.org/10.1109/ICDM.2009.47

See Also

DSClusterer, DSC_MOA, DSC_DenStream, DSC_CobWeb, DSC_StreamKM, DSC_Clustream

Examples

```
dsc <- DSC_ClusTree()
dsd <- DSD_Gaussian_Static()
cluster(dsc, dsd, 5000)
plot(dsc)
```

DSC_CobWeb *DataStreamClusterer: CobWeb*

Description

Creates a new DataStreamClusterer with the CobWeb algorithm

Usage

```
DSC_CobWeb(acuity = 1, cutoff = 0.002, randomSeed = 1)
```

Arguments

acuity	The acuity used in the underlying implementation.
cutoff	The cut off.
randomSeed	The random seed used in the MOA implementation.

Value

A list of class DSC, DSC_MOA, and DSC_CobWeb. The list contains the following items:

description	The name of the algorithm in the DSC object.
options	The CLI params defined when creating the DSC object.
javaObj	The underlying Java object associated with DSC_MOA objects.

References

Douglas H. Fisher. 1987. Knowledge Acquisition Via Incremental Conceptual Clustering. Mach. Learn. 2, 2 (September 1987), 139-172.

See Also

DSClusterer, DSC_MOA, DSC_DenStream, DSC_Clustream, DSC_StreamKM, DSC_ClusTree

Examples

```
#dsc <- DSC_CobWeb()
#dsd <- DSD_Gaussian_Static()
#cluster(dsc, dsd, 1000)
#plot(dsc)
```

DSC_DenStream *DataStreamClusterer: DenStream*

Description

Creates a new DataStreamClusterer with the DenStream algorithm

Usage

```
DSC_DenStream(epsilon = 0.1, minPoints = 10, lambda = 0.006, beta = 0.001, mu =
```

Arguments

epsilon	-e (defines the epsilon neighborhood, range: 0 to 1)
minPoints	-p (min. num. points a cluster must have)
lambda	-l (range: 0 to 1)
beta	-b (range: 0 to 1)
mu	-m (range: 0 to max(double))
initPoints	-i (number of points to use for initialization)

Details

DSC_DenStream is based on the DenStream algorithm. DenStream is a density based clustering algorithm that summarizes the incoming data points into micro-clusters based upon the frequency of data points in a particular location within a defined boundary (i.e., their density). DenStream includes a fading function that causes the weight of data points to decrease exponentially as t increases. This means that temporal structures within the data stream are also visible using this algorithm.

The parameters are defined as follows: epsilon: refers to the neighborhood in which the density of each micro-cluster is calculated (in other words, the radius).

minPoints: is the minimum number of points a micro-cluster must have.

lambda: alters the fading function: the larger `lambda`, the lower the weight of the historic data.

beta: is the outlier threshold, and is used in conjunction with `mu`

mu: the minimum weight a micro-cluster must have within the `epsilon` neighborhood

epsilon: the neighborhood to be considered a core-micro-cluster (as opposed to an outlier or a potential-micro-cluster).

initPoints: is the number of data points to initialize the algorithm with.

Value

A list of class `DSC`, `DSC_MOA`, and `DSC_DenStream`. The list contains the following items:

description	The name of the algorithm in the DSC object.
options	The CLI params defined when creating the DSC object.
javaObj	The underlying Java object associated with DSC_MOA objects.

References

Cao F, Ester M, Qian W, Zhou A (2006). "Density-Based Clustering over an Evolving Data Stream with Noise." In "Proceedings of the 2006 SIAM International Conference on Data Mining," pp 326-337. SIAM.

Bifet A, Holmes G, Pfahringer B, Kranen P, Kremer H, Jansen T, Seidl T (2010). "MOA: Massive Online Analysis, a Framework for Stream Classification and Clustering." In "Journal of Machine Learning Research (JMLR) Workshop and Conference Proceedings,"

See Also

DSClusterer, DSC_MOA, DSC_Clustream, DSC_CobWeb, DSC_StreamKM, DSC_ClusTree

Examples

```
dsc <- DSC_DenStream()
dsd <- DSD_Gaussian_Static()
cluster(dsc, dsd, 5000)
plot(dsc)
```

DSC_MOA *DSC_MOA*

Description

An abstract class that inherits from the base DSC class, DSClusterer. All DSC objects that use underlying MOA code will inherit this class because they have a specific way of operating.

Details

DSC_MOA classes operate in a different way in that the centers of the micro-clusters have to be extracted from the underlying Java object. This is done by performing method calls directly in the JRI and unpackaging the multi-dimensional Java array into a local R data type.

See Also

DSClusterer, DSC_CobWeb, DSC_ClusTree, DSC_StreamKM, DSC_DenStream, DSC_Clustream, DSData

---

DSC_StreamKM *DataStreamClusterer: StreamKM*

Description

Creates a new DataStreamClusterer with the StreamKM++ algorithm

Usage

```
DSC_StreamKM(sizeCoreset=100, k=5, width=1000, randomSeed=1)
```

Arguments

sizeCoreset	The size of the coreset tree.
k	The number of clusters.
width	
randomSeed	The random seed used in the k-means algorithm.

Value

A list of class DSC, DSC_MOA, and DSC_StreamKM. The list contains the following items:

description	The name of the algorithm in the DSC object.
options	The CLI params defined when creating the DSC object.
javaObj	The underlying Java object associated with DSC_MOA objects.

References

Marcel R. Ackermann, Christiane Lammersen, Marcus M\"artens, Christoph Raupach, Christian Sohler, Kamil Swierkot. StreamKM++: A Clustering Algorithm for Data Streams. In Proceedings of the 12th Workshop on Algorithm Engineering and Experiments (ALENEX '10), pp. 173-187, Society for Industrial and Applied Mathematics, 2010.

See Also

DSClusterer, DSC_MOA, DSC_DenStream, DSC_CobWeb, DSC_Clustream, DSC_ClusTree

Examples

```
#dsc <- DSC_StreamKM()
#dsd <- DSD_Gaussian_Static()
#cluster(dsc, dsd, 5000)
#plot(dsc)
```

DSC_tNN *DataStreamClusterer: tNN*

Description

Creates a new DataStreamClusterer with the tNN (threshold Nearest Neighbor) algorithm

Usage

```
DSC_tNN(threshold = 0.2, measure = "euclidean",
centroids = identical(tolower(measure), "euclidean"), lambda=0)
```

Arguments

threshold	The threshold in the nearest neighborhood algorithm.
measure	The measure used to calculate cluster proximity.
centroids	The measure used to calculate centroid proximity.
lambda	The lambda used in the fading function.

Value

description	The name of the algorithm in the DSC object.
clusterFun	The cluster function specific to DSC_tNN
RObj	The underlying R object associated with DSC_R objects.

References

M.H. Dunham, Y. Meng, J. Huang (2004): Extensible Markov Model, In: ICDM '04: Proceedings of the Fourth IEEE International Conference on Data Mining, pp. 371-374.

M. Hahsler, M. H. Dunham (2010): rEMM: Extensible Markov Model for Data Stream Clustering in R, Journal of Statistical Software, 35(5), 1-31, URL http://www.jstatsoft.org/v35/i05/

See Also

DSClusterer, DSC_MOA, DSC_DenStream, DSC_CobWeb, DSC_StreamKM, DSC_Clustream, DSC_ClusTree

Examples

```
dsc <- DSC_tNN()
dsd <- DSD_Gaussian_Static()
cluster(dsc, dsd, 5000)
plot(dsc)
```

DSData *DSData*

Description

The abstract base class for all DataStreamData classes.

Details

The DSData class cannot be instantiated, but it serves as a base class from which all DSD objects inherit from. There is only one function that needs to be implemented: get_points(). DSData also provides a generic print() function that displays basic information about the class.

In addition to this function, each DSD class also needs a constructor specific to that class. The links below contain various DSD classes that have been implemented.

See Also

DSClusterer, DSD_Gaussian_Static,

DSD_ReadStream, DSD_DataFrame,

DSD_DataFrame *DataStreamData: DataFrame (a wrapper for data frames and matrices)*

Description

A DataStreamData class that wraps either a data frame or matrix that was created in R. The data can either be looped or replayed manually to give the exact same data to 2 different DataStreamTask objects.

Usage

```
DSD_DataFrame(df, k, loop=FALSE)
```

Arguments

df	A data frame or matrix with the data to be used in the stream.
k	Optional: The number of clusters
loop	A flag that tells the stream to loop or not to loop over the data frame.

Details

The DSD_DataFrame class is designed to be a wrapper class for data that is generated within R in either a data frame or matrix form. It removes the step of having to write the data to a file, then read in by a connection through DSD_ReadStream.

It works like the other DSD classes–by calling get_points() to retrieve data from the stream. The function reset_stream() can be used to move the counter back to the beginning of the stream. This is an important feature to be able to replay stream data for multiple clusters as shown in the example below.

The value returned is a list of class DSD, and DSD_DataFrame. The important items with the list are strm, the data frame being wrapped, state the environment which contains the counter, and d, the number of dimensions in the stream.

Value

description	The name of the class of the DSD object.
strm	The data frame or matrix that the stream is wrapping.
state	The environment variable which holds the counter for the data frame (accessed through state$counter).
d	The number of dimensions (ncol(strm).
k	The number of clusters (may not be defined).
loop	The flag that determines if looping is or is not enabled.

See Also

DSD_MOA, DSD_ReadStream,

DSD_Gaussian_Static, write_stream, reset_stream

Examples

```
# creating the DSD_DataFrame from other stream data
dsd <- DSD_Gaussian_Static(k=3, d=2)
replayer <- DSD_DataFrame(get_points(dsd, 100), k=3)

# creating 2 clusterers of different algorithms
dsc1 <- DSC_Clustream()
dsc2 <- DSC_CobWeb()

# clustering the same data in 2 DSC objects
cluster(dsc1, replayer, 100)
reset_stream(replayer) # resetting the dsd to its original state
cluster(dsc2, replayer, 100)
```

DSD_Gaussian_Static

DataStreamData: Static data stream

Description

A DataStreamData that generates random data based upon either a defined covariance matrix or a randomly generated covariance matrix.

Usage

```
DSD_Gaussian_Static(k=2, d=2, mu, sigma, p, noise=0)
```

Arguments

k	Determines the number of clusters.
d	Determines the number of dimensions.
mu	A matrix of means for each dimension of each cluster.
sigma	The covariance matrix.
p	A vector of probabilities that determines the likelihood of generated a data point from a particular cluster.
noise	Noise is generated in the stream based on this parameter.

Details

DSD_Gaussian_Static is a general purpose DSD generator for stream data. It has been implemented entirely in R, so there is no computational overhead with communicating to the Java Runtime Interface (JRI) or native C code. An important characteristic of DSD_Gaussian_Static is that once it has been initialized according to the input parameters defined, the defined clusters will not move (i.e., they are static). This means that DSD_Gaussian_Static is not an ideal DSD for examining the temporal structure of data streams for drastic changes such as the splitting or merging of clusters.

Its initialization function accepts 5 main parameters: the number of clusters k, the number of dimensions d, a matrix of means mu, a covariance matrix sigma, and a probability vector p.

By default, DSD_Gaussian_Static generates 2 dimensionality data in 2 different clusters, but the user is able to define any number of clusters with any number of dimensions. Additionally, the

user may define mu, sigma, and p, but if left undefined the constructor will generate these values automatically. When get_points() is called on DSD_Gaussian_Static, the data points are generated using the mvrnorm() function, making it important to seed the random number generator to reproduce the experiment. This can be done with set.seed().

Value

Returns a DSD_Gaussian_Static object which is a list of the defined params. The params are either passed in from the function or created internally. They include:

description	A brief description of the DSD object.
k	The number of clusters.
d	The number of dimensions.
mu	The matrix of means of the dimensions in each cluster.
sigma	The covariance matrix.
p	The probability vector for the cluters.
noise	A flag that determines if or if not noise is generated.

See Also

DSD_MOA, DSD_ReadStream,

DSD_DataFrame, write_stream

Examples

```
# other params will be generated internally
dsd <- DSD_Gaussian_Static(k=2, d=2)
dsc <- DSC_DenStream()

# obtaining some data
sample <- get_points(dsd, 500)

# clustering some data, this will take some time
cluster(dsc, dsd, 5000)

# plotting the data
plot(sample, xlab='x', ylab='y', pch=4, cex=.5)
points(get_centers(dsc),col='red', cex=3,lwd=2)
```

DSD_MOA *DataStreamData - MOA*

Description

A DataStreamData that generates random data based upon the RandomRBFGenerator implemented in MOA.

Usage

```
DSD_MOA(k=3, d=2, avgRadius=0, modelSeed=1, instanceSeed=1)
```

Arguments

k	The number of clusters.
d	The dimensionality of the data.
avgRadius	The average radius of the micro-clusters.
modelSeed	Random seed for the model.
instanceSeed	Random seed for the instances.

Value

description	The name of the class of the DSD object.
options	The CLI params defined when creating the DSD object.
javaObj	The underlying Java object associated with DSD_MOA objects.

References

Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Philipp Kranen, Hardy Kremer, Timm Jansen, Thomas Seidl. Journal of Machine Learning Research (JMLR) Workshop and Conference Proceedings. Volume 11: Workshop on Applications of Pattern Analysis (2010).

See Also

DSD_Gaussian_Static, DSD_ReadStream, DSD_DataFrame, write_stream

---

DSD_ReadStream *DataStreamData: ReadStream (from an open connection or a file name)*

Description

A DSD object that reads a data stream from an R connection.

Usage

```
DSD_ReadStream(x, sep=",", loop=FALSE)
```

Arguments

x	An open connection, or a file path/URL to be opened as a connection.
sep	The character string that separates dimensions in data points in the stream.
loop	If enabled, the object will loop through the stream when the end has been reached. If disabled, the object will warn the user upon reaching the end.

Details

DSD_ReadStream uses read.table() to read in data from an R connection. The connection is responsible for maintaining where the stream is currently being read from. In general, the connections will consist of files stored on disk but have many other possibilities (see connection).

The get_points() method can pass additional params to read.table() to alter how the reading is done. By default, the comment.char is set to an empty string for performance reasons.

Value

description	The name of the class of the DSD object.
con	The connection where the data stream is being read from.
sep	The character string that separates dimensions in data points in the stream.
loop	A flag that determines whether or not the stream will loop.

See Also

DSD_MOA, DSD_ReadStream, DSD_Gaussian_Static, DSD_DataFrame, write_stream reset_stream

Examples

```
# creating data and writing it to disk
dsd <- DSD_Gaussian_Static(k=3, d=5)
write_stream(dsd, "dsd_data.txt", n=100, sep=",")

# reading the same data back in
dsd2 <- DSD_ReadStream("dsd_data.txt", sep=",")
```

---

get_centers *get_centers*

Description

Gets the centers of micro-clusters (if available) from a DSC object

Usage

```
get_centers(x, ...)
```

Arguments

x	The DSC object the centers are being requested from.
...	Additional parameters to pass to get_centers().

Details

Each DSC object has a unique way for returning data points, but they all are called through the generic function, get_centers(). This is done by using the S3 class system.

All of the DSC_MOA classes use the same function that first checks for the existence of micro-clusters (as deemed in the MOA framework), extracts those micro-clusters, then manually extracts the center from each of those micro-clusters and packages them into a matrix.

Value

A matrix of data that contains the centers of the micro-clusters (these can be either the centroids or medoids depending on the algorithm). There are some DSC classes that do not generate micro-clusters–an error will be given in these cases.

The number of columns in the matrix is dependent on the data stream that was used as input data to the DSC object, and the number of rows will differ upon the algorithm being used.

See Also

DSClusterer, DSC_CobWeb, DSC_ClusTree, DSC_StreamKM, DSC_DenStream, DSC_Clustream, DSData

Examples

```
# setting up the objects
dsd <- DSD_Gaussian_Static()
dsc <- DSC_DenStream(initPoints=100)
cluster(dsc, dsd, 500)

# getting the centers
d <- get_centers(dsc)
```

---

get_points *get_points*

Description

Gets points from a DSD object.

Usage

```
get_points(x, n=1, ...)
```

Arguments

x	The DSD object.
n	The number of data points being requested.
...	Additional parameters to pass to get_points()

Details

Each DSD object has a unique way for returning data points, but they all are called through the generic function, get_points(). This is done by using the S3 class system. See the man page for the specific DSD class on the semantics for each implementation of get_points().

Value

Returns a matrix of x$d columns and n rows.

See Also

DSD_Gaussian_Static, DSD_ReadStream,

DSD_DataFrame, DSD_MOA, write_stream

Examples

```
dsd <- DSD_Gaussian_Static()
d <- get_points(dsd, 100)
```

reset_stream *reset_stream*

Description

Resets the counter in a DSD object to the beginning.

Usage

```
reset_stream(x)
```

Arguments

x	Either a DSD_ReadStream or DSD_DataFrame object.

Details

Resets the counter of the stream object that is passed in. For DSD_ReadStream objects, this is done by calling seek() on the underlying connection. For DSD_DataFrame, the counter stored in the environment variable is moved back to 1.

See Also

DSD_ReadStream, DSD_DataFrame, write_stream

Examples

```
# initializing the objects
dsd <- DSD_Gaussian_Static(k=3, d=2)
replayer <- DSD_DataFrame(get_points(dsd, 100), k=3)
dsc1 <- DSC_Clustream()
dsc2 <- DSC_CobWeb()

# clustering the same data in 2 DSC objects
cluster(dsc1, replayer, 100)
reset_stream(replayer) # resetting the dsd to its original state
cluster(dsc2, replayer, 100)
```

write_stream *write_stream*

Description

Writes points to a connection from a DSD object.

Usage

```
write_stream(dsd, con, n=100, sep=",", col.names=FALSE,
row.names=FALSE, ...)
```

Arguments

dsd	The DSD object that will generate the data points for output.
con	The R connection to be written to.
n	The number of data points to be written to the connection.
sep	The character that will seperate attributes in a data point.
col.names	A flag that determines if column names will be output.
row.names	A flag that determines if row names will be output.
...	Additional parameters that are passed to write.table().

Value

There is no value returned from this operation.

See Also

write.table, DSD_Gaussian_Static, DSD_ReadStream, DSD_DataFrame, DSD_MOA,

Examples

```
# creating data and writing it to disk
dsd <- DSD_Gaussian_Static(k=3, d=5)
write_stream(dsd, "dsd_data.txt", n=100, sep=",")
```

Index

cluster, 2
connection, *13*

DSC_Clustream, *3*, 3, *4*, *5*, *7–9*, *15*
DSC_ClusTree, *3*, *4*, 4, *5*, *7–9*, *15*
DSC_CobWeb, *3*, *4*, 5, *7–9*, *15*
DSC_DenStream, *3–5*, 6, *7–9*, *15*
DSC_MOA, *4*, *5*, *7*, 7, *8*, *9*
DSC_StreamKM, *3–5*, *7*, 7, *9*, *15*
DSC_tNN, 8
DSClusterer, 2, *3–5*, *7–9*, *15*
DSD_DataFrame, *9*, 10, *12–17*
DSD_Gaussian_Static, *9*, *10*, 11, *13–15*, *17*
DSD_MOA, *10*, *12*, 12, *14*, *15*, *17*
DSD_ReadStream, *9*, *10*, *12*, *13*, 13, *14–17*
DSData, *3*, *7*, 9, *15*

get_centers, 14
get_points, 15

reset_stream, *10*, *14*, 16

write.table, *17*
write_stream, *10*, *12–16*, 16