Package: streamMOA (via r-universe)

October 14, 2026

Version 1.4.0

Date 2026-10-13

Encoding UTF-8

Title Interface for MOA Stream Clustering Algorithms

Description Interface for data stream clustering algorithms implemented in the MOA (Massive Online Analysis) framework (Albert Bifet, Geoff Holmes, Richard Kirkby, Bernhard Pfahringer (2010). MOA: Massive Online Analysis, Journal of Machine Learning Research 11: 1601-1604).

Depends stream (>= 2.0-0), rJava (>= 1.0-1)

Imports graphics, stats, methods, proxy

Suggests testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

SystemRequirements Java (>= 8)

BugReports https://github.com/mhahsler/streamMOA/issues

License GPL-3

Copyright MOA code in inst/java/moa.jar is Copyright (C) The University of Waikato and distributed under the Apache License, version 2.0. All other code is Copyright (C) Matthew Bolanos, John Forrest and Michael Hahsler

Config/testthat/edition 3

URL https://github.com/mhahsler/streamMOA, https://michael.hahsler.net/streamMOA/

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Michael Hahsler [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-2716-1405>), John Forrest [aut, cph], Matthew Bolanos [ctb], Matthias Carnein [ctb], Dalibor Krleža [ctb]

Maintainer Michael Hahsler <mhahsler@lyle.smu.edu>

Config/pak/sysreqs default-jdk

Repository https://cran.r-universe.dev

Date/Publication 2026-10-13 22:20:02 UTC

RemoteUrl https://github.com/cran/streamMOA

RemoteRef HEAD

RemoteSha affc482a05feabdf9f0338a3ac46bc4622e4a4da

Contents

DSC_BICO_MOA 2
DSC_CluStream 4
DSC_ClusTree 5
DSC_DenStream 6
DSC_DStream_MOA 8
DSC_MCOD 9
DSC_MOA 11
DSC_StreamKM 12
DSD_MOA 13
DSD_RandomRBFGeneratorEvents 14

Index **17**

`DSC_BICO_MOA` *BICO: Fast computation of k-means coresets in a data stream*

Description

This is an interface to the MOA implementation of BICO. The original BICO implementation by Fichtenberger et al is also available as stream::DSC_BICO.

Usage

```
DSC_BICO_MOA(
  Cluster = 5,
  Dimensions,
  MaxClusterFeatures = 1000,
  Projections = 10,
  k = NULL,
  space = NULL,
  p = NULL
)
```

Arguments

`Cluster`	Number of desired centers.
`Dimensions`	Number of dimensions in the input stream; this must be specified in advance.
`MaxClusterFeatures`	Maximum number of cluster features in the coreset.
`Projections`	Number of random projections used for the nearest neighbor search.
`k`	Alias for `Cluster`.
`space`	Alias for `MaxClusterFeatures`.
`p`	Alias for `Projections`.

Details

BICO maintains a tree which is inspired by the clustering tree of BIRCH, a SIGMOD Test of Time award-winning clustering algorithm. Each node in the tree represents a subset of these points. Instead of storing all points as individual objects, only the number of points, the sum and the squared sum of the subset's points are stored as key features of each subset. Points are inserted into exactly one node.

Author(s)

Matthias Carnein

References

Hendrik Fichtenberger, Marc Gille, Melanie Schmidt, Chris Schwiegelshohn, Christian Sohler: BICO: BIRCH Meets Coresets for k-Means Clustering. ESA 2013: 481-492

See Also

Other DSC_MOA: `DSC_CluStream()`, `DSC_ClusTree()`, `DSC_DStream_MOA()`, `DSC_DenStream()`, `DSC_MCOD()`, `DSC_MOA()`, `DSC_StreamKM()`

Examples

```
# data with 3 clusters and 2 dimensions
set.seed(1000)
stream <- DSD_Gaussians(k = 3, d = 2, noise = 0.05)

# cluster with BICO
bico <- DSC_BICO_MOA(Cluster = 3, Dimensions = 2)
update(bico, stream, 100)
bico

# plot micro and macro-clusters
plot(bico, stream, type = "both")
```

DSC_CluStream *CluStream Data Stream Clusterer*

Description

Implements the CluStream algorithm for data streams (Aggarwal et al., 2003).

Usage

```
DSC_CluStream(m = 100, horizon = 1000, t = 2, k = 5)
```

Arguments

m	Maximum number of micro-clusters.
horizon	Time window used by CluStream.
t	Maximum boundary factor used to decide whether a new point belongs to a micro-cluster. The boundary is t times the root mean square deviation from the micro-cluster center.
k	Number of macro-clusters produced by weighted k-means.

Details

This is an interface to the MOA implementation of CluStream.

If k is specified, then CluStream applies a weighted k-means algorithm for reclustering (see Examples section below).

Value

An object of class DSC_CluStream (subclass of stream::DSC_Micro, DSC_MOA and stream::DSC).

Author(s)

Michael Hahsler and John Forrest

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).

See Also

Other DSC_MOA: DSC_BICO_MOA(), DSC_ClusTree(), DSC_DStream_MOA(), DSC_DenStream(), DSC_MCOD(), DSC_MOA(), DSC_StreamKM()

Examples

```
# data with 3 clusters and 5% noise
set.seed(1000)
stream <- DSD_Gaussians(k = 3, d = 2, noise = .05)

# cluster with CluStream
clustream <- DSC_CluStream(m = 50, horizon = 100, k = 3)
update(clustream, stream, 500)
clustream

plot(clustream, stream, type = "both")
```

DSC_ClusTree *ClusTree Data Stream Clusterer*

Description

Interface to the MOA implementation of the ClusTree data stream clustering algorithm (Kranen et al., 2009).

Usage

```
DSC_ClusTree(horizon = 1000, maxHeight = 8, lambda = NULL, k = NULL)
```

Arguments

horizon	Length of the time window.
maxHeight	Maximum height of the tree.
lambda	Value used to override the computed decay parameter.
k	If specified, use k-means with k clusters for reclustering.

Details

ClusTree uses a compact, self-adaptive index structure to maintain stream summaries. Kranen et al. (2009) suggest EM or k-means for reclustering.

Value

An object of class DSC_ClusTree (subclass of stream::DSC, DSC_MOA, stream::DSC_Micro).

Author(s)

Michael Hahsler and John Forrest

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

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).

See Also

Other DSC_MOA: DSC_BICO_MOA(), DSC_CluStream(), DSC_DStream_MOA(), DSC_DenStream(), DSC_MCOD(), DSC_MOA(), DSC_StreamKM()

Examples

```
# data with 3 clusters
set.seed(1000)
stream <- DSD_Gaussians(k = 3, d = 2, noise = 0.05)

clustree <- DSC_ClusTree(maxHeight = 3)
update(clustree, stream, 500)
clustree

plot(clustree, stream)

# Use the k-means reclusterer with k = 3 to create macro-clusters
clustree <- DSC_ClusTree(maxHeight = 3, k = 3)
update(clustree, stream, 500)
clustree

plot(clustree, stream, type = "both")
```

DSC_DenStream *DenStream Data Stream Clusterer*

Description

Interface for the DenStream cluster algorithm for data streams implemented in MOA.

Usage

```
DSC_DenStream(
  epsilon,
  mu = 1,
  beta = 0.2,
  lambda = 0.001,
  initPoints = 100,
```

```
  offline = 2,
  processingSpeed = 1,
  recluster = TRUE,
  k = NULL
)
```

Arguments

epsilon	Maximum radius of a micro-cluster. Must be between 0 and 1.
mu	Minimum weight required for a core micro-cluster.
beta	Weight multiplier used to identify outlier micro-clusters. Must be between 0 and 1.
lambda	Decay constant.
initPoints	Number of points used to initialize the algorithm with DBSCAN.
offline	Multiplier applied to epsilon for reachability reclustering. Must be between 2 and 20.
processingSpeed	Number of incoming points per time unit, used for decay. Must be between 1 and 1000.
recluster	If TRUE, apply offline reachability reclustering.
k	If specified, choose a reachability threshold to produce this number of macro-clusters.

Details

DenStream reclusters micro-clusters using DBSCAN-style reachability. The threshold is epsilon * offline (with offline = 2 by default).

If k is specified, single-link hierarchical clustering chooses a reachability threshold that produces k macro-clusters.

Value

An object of class DSC_DenStream (subclass of stream::DSC, DSC_MOA, stream::DSC_Micro) or, for recluster = TRUE, an object of class stream::DSC_TwoStage.

Author(s)

Michael Hahsler and John Forrest

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).

See Also

Other DSC_MOA: DSC_BICO_MOA(), DSC_CluStream(), DSC_ClusTree(), DSC_DStream_MOA(), DSC_MCOD(), DSC_MOA(), DSC_StreamKM()

Examples

```
# data with 3 clusters and 5% noise
set.seed(1000)
stream <- DSD_Gaussians(k = 3, d = 2, noise = 0.05)

# use Den-Stream with reachability reclustering
denstream <- DSC_DenStream(epsilon = .05)
update(denstream, stream, 500)
denstream

# plot macro-clusters
plot(denstream, stream, type = "both")

# plot micro-clusters
plot(denstream, stream, type = "micro")

# reclustering: Choose reclustering reachability threshold automatically to find 4 clusters
denstream2 <- DSC_DenStream(epsilon = .05, k = 4)
update(denstream2, stream, 500)
plot(denstream2, stream, type = "both")
```

DSC_DStream_MOA *D-Stream Data Stream Clustering Algorithm*

Description

This is an interface to the MOA implementation of D-Stream. A C++ implementation (including reclustering with attraction) is available as stream::DSC_DStream.

Usage

```
DSC_DStream_MOA(decayFactor = 0.998, Cm = 3, Cl = 0.8, Beta = 0.3)
```

Arguments

decayFactor	Decay factor applied to grid-cell density.
Cm	Threshold for classifying grid cells as dense.
Cl	Threshold for classifying grid cells as sparse.
Beta	Adjusts the window of protection for renaming previously deleted grids as sporadic.

Details

D-Stream creates an equally spaced grid and estimates the density in each grid cell using the count of points falling in the cells. Grid cells are classified based on density into dense, transitional and sporadic cells. The density is faded after every new point by a decay factor.

Notes

- The implementation uses a 1-by-1 grid, so the example expands the data range.
- The MOA implementation does not currently return micro-clusters.

Author(s)

Matthias Carnein

References

Yixin Chen and Li Tu. 2007. Density-based clustering for real-time stream data. In Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '07). ACM, New York, NY, USA, 133-142.

Li Tu and Yixin Chen. 2009. Stream data clustering based on grid density and attraction. ACM Transactions on Knowledge Discovery from Data, 3(3), Article 12 (July 2009), 27 pages.

See Also

Other DSC_MOA: DSC_BICO_MOA(), DSC_CluStream(), DSC_ClusTree(), DSC_DenStream(), DSC_MCOD(), DSC_MOA(), DSC_StreamKM()

Examples

```
set.seed(1000)
stream <- DSD_Gaussians(k = 3, d = 2, noise = 0.05, space_limit = c(0, 10))

# cluster with D-Stream
dstream <- DSC_DStream_MOA(Cm = 3)
update(dstream, stream, 1000)
dstream

# plot macro-clusters
plot(dstream, stream, type = "macro")
```

DSC_MCOD *Micro-cluster Continuous Outlier Detector (MCOD)*

Description

Interface to the MOA implementation of the MCOD algorithm for distance-based data stream outlier detection.

Usage

```
DSC_MCOD(r = 0.1, t = 50, w = 1000, recheck_outliers = FALSE)

DSOutlier_MCOD(r = 0.1, t = 50, w = 1000, recheck_outliers = TRUE)

get_outlier_positions(x, ...)

recheck_outlier(x, outlier_correlated_id, ...)

clean_outliers(x, ...)
```

Arguments

r	Radius used to search for neighbors.
t	Minimum number of neighbors required for a point not to be an outlier.
w	Sliding window width in data points.
recheck_outliers	If TRUE, allow detected outliers to be checked again.
x	A DSC_MCOD object.
...	Further arguments (currently ignored).
outlier_correlated_id	Identifier of the outlier to check again.

Details

The algorithm detects density-based outliers. An object x is defined to be an outlier if there are less than t objects lying at distance at most r from x.

Outliers are stored and can be retrieved with `get_outlier_positions()` and checked again with `recheck_outlier()`.

Note: The implementation updates the clustering when `predict()` is called.

Value

An object of class DSC_MCOD (subclass of stream::DSC_Micro, DSC_MOA and stream::DSC).

Functions

- `get_outlier_positions()`: Returns spatial positions of all current outliers.
- `recheck_outlier()`: Re-check whether the outlier identified by `outlier_correlated_id` is still an outlier. Returns TRUE if it is.
- `clean_outliers()`: Forget detected outliers (currently not implemented).

Author(s)

Dalibor Krleža

References

Kontaki M, Gounaris A, Papadopoulos AN, Tsichlas K, and Manolopoulos Y (2016). Efficient and flexible algorithms for monitoring distance-based outliers over data streams. *Information Systems,* Vol. 55, pp. 37-53. doi:10.1109/ICDE.2011.5767923

See Also

Other DSC_MOA: DSC_BICO_MOA(), DSC_CluStream(), DSC_ClusTree(), DSC_DStream_MOA(), DSC_DenStream(), DSC_MOA(), DSC_StreamKM()

Examples

```
# Example 1: Clustering with MCOD
stream <- DSD_Gaussians(k = 3, d = 2, noise = 0.05)
mcod <- DSC_MCOD(r = .1, t = 3, w = 100)
update(mcod, stream, 100)
mcod

plot(mcod, stream, n = 100)

# Example 2: Predict outliers (have a class label of NA)
stream <- DSD_Gaussians(k = 3, d = 2, noise = 0.05)
mcod <- DSOutlier_MCOD(r = .1, t = 3, w = 100)
update(mcod, stream, 100)

plot(mcod, stream, n = 100)

# Retrieve detected outlier positions.
get_outlier_positions(mcod)

# Example 3: evaluate on a stream
evaluate_static(mcod, stream, n = 100, type = "micro",
  measure = c("crand", "noisePrecision", "outlierjaccard"))
```

DSC_MOA *DSC_MOA Class*

Description

An abstract class that inherits from the base class stream::DSC and provides the common functions needed to interface MOA clusterers.

Usage

```
DSC_MOA(...)
```

Arguments

...	Further arguments (currently ignored).

Details

DSC_MOA is a subclass of stream::DSC for MOA-based clusterers. 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 using **rJava** to perform method calls directly in the JRI and converting the multi-dimensional Java array into a local R data type.

Note: The formula interface is currently not implemented for MOA-based clusterers. Use stream::DSF to select features instead.

Author(s)

Michael Hahsler and John Forrest

References

Albert Bifet, Geoff Holmes, Richard Kirkby, Bernhard Pfahringer (2010). MOA: Massive Online Analysis, Journal of Machine Learning Research 11: 1601-1604

See Also

Other DSC_MOA: DSC_BICO_MOA(), DSC_CluStream(), DSC_ClusTree(), DSC_DStream_MOA(), DSC_DenStream(), DSC_MCOD(), DSC_StreamKM()

DSC_StreamKM *streamKM++*

Description

This is an interface to the MOA implementation of streamKM++.

Usage

```
DSC_StreamKM(sizeCoreset = 10000, numClusters = 5, length = 100000L, ...)
```

Arguments

sizeCoreset	Size of the coreset.
numClusters	Number of clusters to compute.
length	Maximum number of points in the data stream.
...	Further arguments (currently ignored).

Details

streamKM++ uses a tree-based sampling strategy to build a small weighted sample of the stream (a coreset). The MOA implementation applies k-means++ to find the requested number of centers in the coreset.

Notes

- The clusterer can process at most length points. Processing more points causes an ArrayIndexOutOfBoundsExceptio
- The coreset is not exposed as micro-clusters; only macro-clusters can be requested.

Author(s)

Matthias Carnein

References

Marcel R. Ackermann, Christiane Lammersen, Marcus Maertens, 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)*, 2010.

See Also

Other DSC_MOA: DSC_BICO_MOA(), DSC_CluStream(), DSC_ClusTree(), DSC_DStream_MOA(), DSC_DenStream(), DSC_MCOD(), DSC_MOA()

Examples

```
set.seed(1000)
stream <- DSD_Gaussians(k = 3, d = 2, noise = 0.05)

# cluster with streamKM++
streamkm <- DSC_StreamKM(sizeCoreset = 100, numClusters = 3, length = 1000)
update(streamkm, stream, 100)
streamkm

# plot macro-clusters (no access to micro-clusters)
plot(streamkm, stream)
```

DSD_MOA *Base class for MOA-based data stream generators*

Description

Abstract base class for MOA-based data stream generators. It inherits directly from stream::DSD.

Usage

```
DSD_MOA(...)
```

Arguments

...	Further arguments (currently ignored).

Value

The abstract class cannot be instantiated and produces an error.

Author(s)

Michael Hahsler

References

Bifet A, Holmes G, Pfahringer B, Kranen P, Kremer H, Jansen T, and Seidl T (2010). MOA: Massive Online Analysis, a Framework for Stream Classification and Clustering. *Journal of Machine Learning Research*, 11, 1601–1604.

See Also

Other DSD_MOA: `DSD_RandomRBFGeneratorEvents()`

Examples

```
## Not run:
DSD_MOA()

## End(Not run)
```

`DSD_RandomRBFGeneratorEvents` *Random RBF Generator Events Data Stream Generator*

Description

Generates random data using MOA's `RandomRBFGeneratorEvents` stream.

Usage

```
DSD_RandomRBFGeneratorEvents(
  k = 3,
  d = 2,
  numClusterRange = 3L,
  kernelRadius = 0.07,
  kernelRadiusRange = 0,
  densityRange = 0,
  speed = 100L,
  speedRange = 0L,
  noiseLevel = 0.1,
  noiseInCluster = FALSE,
  eventFrequency = 30000L,
  eventMergeSplitOption = FALSE,
  eventDeleteCreate = FALSE,
  modelSeed = NULL,
  instanceSeed = NULL
)
```

Arguments

k	Average number of centroids in the model.
d	Number of dimensions in the generated stream.
numClusterRange	Range for the number of clusters.
kernelRadius	Average radius of the micro-clusters.
kernelRadiusRange	Range of variation in micro-cluster radii.
densityRange	Range of variation in cluster density.
speed	Number of points between kernel movements.
speedRange	Range of variation in kernel speed.
noiseLevel	Proportion of noise points.
noiseInCluster	If TRUE, allow noise points inside clusters.
eventFrequency	Number of points between concept-drift events.
eventMergeSplitOption	If TRUE, enable cluster merge and split events.
eventDeleteCreate	If TRUE, enable cluster deletion and creation events.
modelSeed	Random seed for the cluster model.
instanceSeed	Random seed for generated instances.

Details

Only a subset of the parameters supported by the underlying MOA generator is exposed. If modelSeed or instanceSeed is NULL, a seed is sampled from R's random-number generator. Set these arguments explicitly to reproduce a stream; call set.seed() to make the generated default seeds reproducible.

By default, the generator creates three clusters with concept drift. Cluster locations move over time, and clusters may merge.

Value

An object of class DSD_RandomRBFGeneratorEvents (subclass of DSD_MOA, stream::DSD).

Author(s)

Michael Hahsler and John Forrest

References

Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Philipp Kranen, Hardy Kremer, Timm Jansen, Thomas Seidl. MOA: Massive Online Analysis, a Framework for Stream Classification and Clustering *Journal of Machine Learning Research (JMLR)*, 2010.

See Also

Other DSD_MOA: DSD_MOA()

Examples

```
stream <- DSD_RandomRBFGeneratorEvents()
get_points(stream, 10)

if (interactive()) {
animate_data(stream, n = 5000, horizon = 100, xlim = c(0, 1), ylim = c(0, 1))
}
```

Index

∗ **DSC_MOA**
  DSC_BICO_MOA, 2
  DSC_CluStream, 4
  DSC_ClusTree, 5
  DSC_DenStream, 6
  DSC_DStream_MOA, 8
  DSC_MCOD, 9
  DSC_MOA, 11
  DSC_StreamKM, 12
∗ **DSD_MOA**
  DSD_MOA, 13
  DSD_RandomRBFGeneratorEvents, 14
∗ **DSOutlier_MOA**
  DSC_MCOD, 9

clean_outliers *(*DSC_MCOD*)*, 9
CluStream *(*DSC_CluStream*)*, 4
clustream *(*DSC_CluStream*)*, 4
ClusTree *(*DSC_ClusTree*)*, 5
clustree *(*DSC_ClusTree*)*, 5

DenStream *(*DSC_DenStream*)*, 6
denstream *(*DSC_DenStream*)*, 6
DSC_BICO_MOA, 2
DSC_BICO_MOA(), *4*, *6*, *8*, *9*, *11–13*
DSC_CluStream, 4
DSC_CluStream(), *3*, *6*, *8*, *9*, *11–13*
DSC_CluStream_MOA *(*DSC_CluStream*)*, 4
DSC_ClusTree, 5
DSC_ClusTree(), *3*, *4*, *8*, *9*, *11–13*
DSC_DenStream, 6
DSC_DenStream(), *3*, *4*, *6*, *9*, *11–13*
DSC_DenStream_MOA *(*DSC_DenStream*)*, 6
DSC_DStream_MOA, 8
DSC_DStream_MOA(), *3*, *4*, *6*, *8*, *11–13*
DSC_MCOD, 9
DSC_MCOD(), *3*, *4*, *6*, *8*, *9*, *12*, *13*
DSC_MCOD_MOA *(*DSC_MCOD*)*, 9
DSC_MOA, *4*, *5*, *7*, *10*, 11
DSC_MOA(), *3*, *4*, *6*, *8*, *9*, *11*, *13*
DSC_StreamKM, 12
DSC_StreamKM(), *3*, *4*, *6*, *8*, *9*, *11*, *12*
DSD_MOA, 13, *15*
DSD_MOA(), *16*
DSD_RandomRBFGeneratorEvents, 14
DSD_RandomRBFGeneratorEvents(), *14*
DSOutlier_MCOD *(*DSC_MCOD*)*, 9
DSOutlier_MCOD_MOA *(*DSC_MCOD*)*, 9

get_outlier_positions *(*DSC_MCOD*)*, 9
get_outlier_positions(), *10*

MCOD *(*DSC_MCOD*)*, 9

predict(), *10*

recheck_outlier *(*DSC_MCOD*)*, 9
recheck_outlier(), *10*

stream::DSC, *4*, *5*, *7*, *10–12*
stream::DSC_BICO, *2*
stream::DSC_DStream, *8*
stream::DSC_Micro, *4*, *5*, *7*, *10*
stream::DSC_TwoStage, *7*
stream::DSD, *13*, *15*
stream::DSF, *12*
StreamKM *(*DSC_StreamKM*)*, 12
streamkm *(*DSC_StreamKM*)*, 12