# streamMOA: Interface to Algorithms from MOA for stream

**Michael Hahsler**
Southern Methodist University

**John Forrest**
Microsoft

**Matthew Bolaños**
Microsoft

---

**Abstract**

This packages provides an interface for several algorithms from the Massive Online Analysis (MOA) framework to be used in **stream**. This vignette contains some examples.

*Keywords*: data stream, data mining, clustering, MOA.

---

## 1. Introduction

Please refer to the vignette in package **stream** for an introduction to data stream mining in R. In this vignette we give two examples that show how to use the **stream** framework being used from start to finish. The examples encompasses the creation of data streams, preparation of data stream clustering algorithms, the online clustering of data points into micro-clusters, reclustering and finally evaluation. The first example shows how compare a set of data stream clustering algorithms on a static data set. The second example shows how to perform evaluation on a data stream with concept drift (clusters evolve over time).

## 2. Experimental Comparison on Static Data

First, we set up a static data set. We extract 1500 data points from the Bars and Gaussians data stream generator with 5% noise and put them in a `DSD_Memory`. The wrapper is used to replay the same part of the data stream for each algorithm. We will use the first 1000 points to learn the clustering and the remaining 500 points for evaluation.

```
R> library("streamMOA")

R> stream <- DSD_BarsAndGaussians(noise=0.05) %>% DSD_Memory(n = 5500)
R> stream

Memorized Stream for Bars and Gaussians (d = 2, k = 4)
Class: DSD_Memory, DSD_R, DSD
Contains 5500 data points - currently at position 1 - loop is FALSE
```

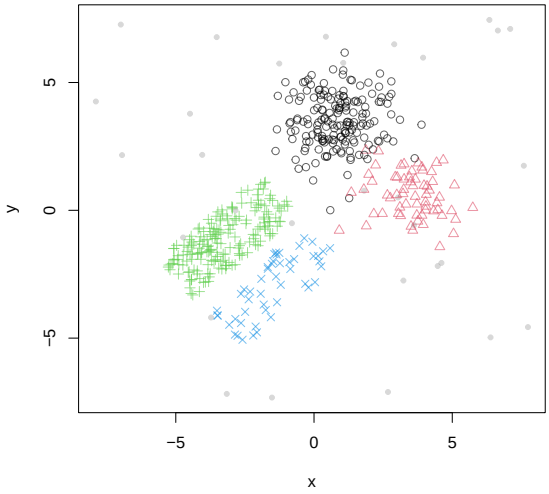


Figure 1: Bar and Gaussians data set.

```
R> plot(stream)
```

Figure 1 shows the structure of the data set. It consists of four clusters, two Gaussians and two uniformly filled rectangular clusters. The Gaussian and the bar to the right have 1/3 the density of the other two clusters.

We initialize a k-means on a sample and multiple clusterers from **streamMOA**. We choose the parameters experimentally so that the algorithm produce each (approximately) 100 micro-clusters.

```
R> algorithms <- list(
+   'Sample + k-means' = DSC_TwoStage(micro = DSC_Sample(k = 100),
+                                     macro = DSC_Kmeans(k = 4)),
+   'DenStream'     = DSC_DenStream(epsilon = .5, mu = 1),
+   'cluStream'     = DSC_CluStream(m = 100, k = 4),
+   'Bico'          = DSC_BICO_MOA(Cluster = 4, Dimensions = 2, MaxClusterFeatures = 100)
+ )
```

We store the algorithms in a list for easier handling and then cluster the same 1000 data points with each algorithm. Note that we have to reset the stream each time before we cluster.

```
R> for (a in algorithms) {
+   reset_stream(stream)
+   update(a, stream, 1000)
+ }
```

We use `nclusters()` to inspect the number of micro-clusters.

```
R> sapply(algorithms, nclusters, type = "micro")

Sample + k-means        DenStream        cluStream             Bico
             100               39              100               78
```

All algorithms except DenStream produce around 100 micro-clusters. We were not able to adjust DenStream to produce more than around 50 micro-clusters for this data set.

To inspect micro-cluster placement, we plot the calculated micro-clusters and the original data.

```
R> op <- par(no.readonly = TRUE)
R> layout(mat = matrix(1:4, ncol = 2))
R> for (a in algorithms) {
+   reset_stream(stream)
+   plot(a, stream, main = description(a), type = "micro")
+ }
R> par(op)
```

Figure 2 shows the micro-cluster placement by the different algorithms. Micro-clusters are shown as red circles and the size is proportional to each cluster's weight. Reservoir sampling and the sliding window randomly place the micro-clusters and also a few noise points (shown as grey dots). Clustream and BICO also do not suppress noise and places even more micro-clusters on noise points since it tries to represent all data as faithfully as possible. DenStream suppresses noise and concentrate the micro-clusters on the real clusters. DenStream produces one heavy micro-cluster on one cluster, while using a large number of micro clusters for the others. It also has problems with detecting the rectangular low-density cluster.

It is also interesting to compare the assignment areas for micro-clusters created by different algorithms. The assignment area is the area around the center of a micro-cluster in which points are considered to belong to the micro-cluster. In case that a point is in the assignment area of several micro-clusters, the closer center is chosen. To show the assignment area we add `assignment = TRUE` to plot. We also disable showing micro-cluster weights to make the plot clearer.

```
R> op <- par(no.readonly = TRUE)
R> layout(mat = matrix(1:4, ncol = 2))
R> for (a in algorithms) {
+   reset_stream(stream)
+   plot(
+     a,
+     stream,
+     main = description(a),
+     assignment = TRUE,
+     weight = FALSE,
+     type = "micro"
+   )
+ }
R> par(op)
```

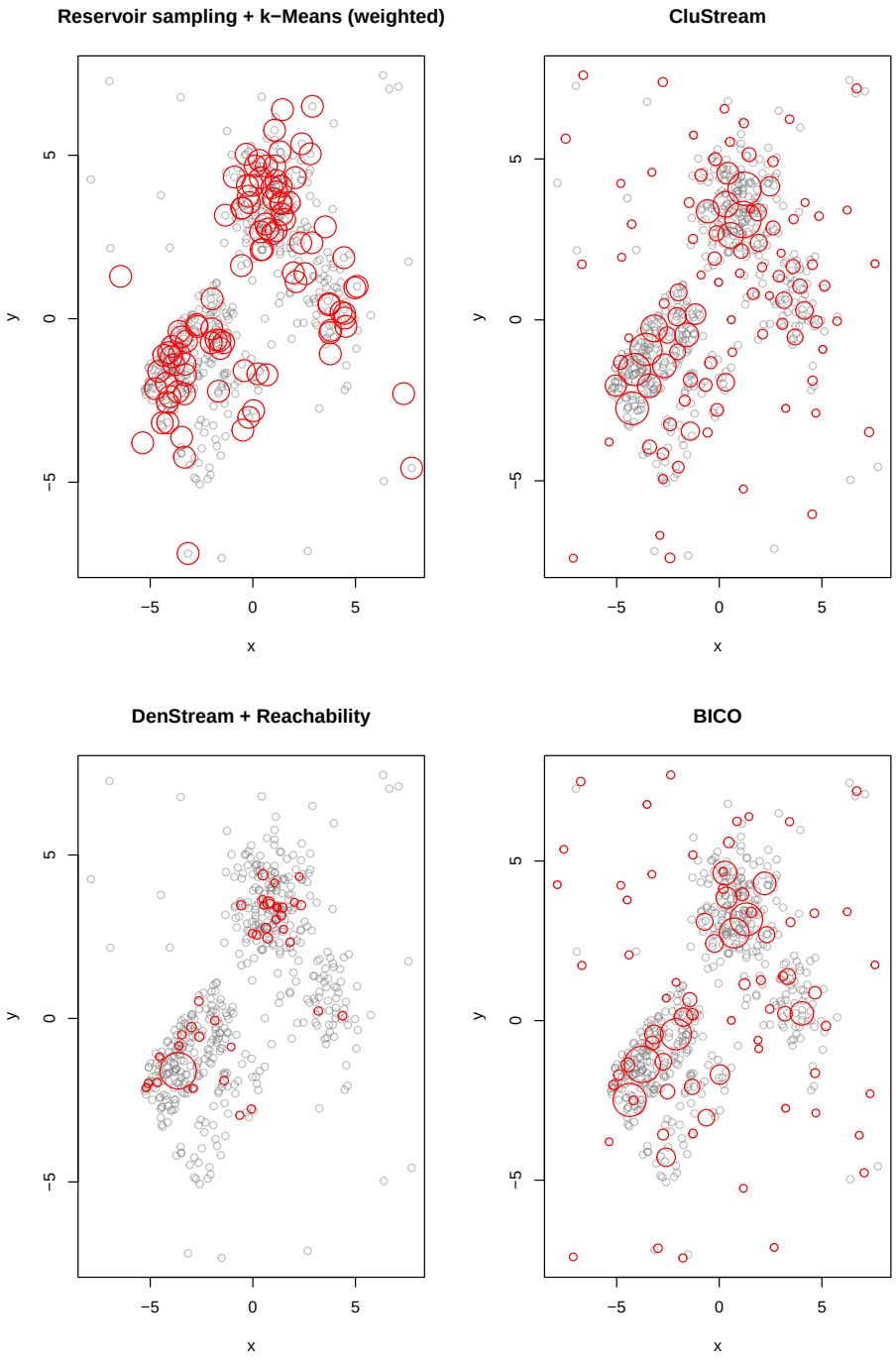


Figure 2: Micro-cluster placement for different data stream clustering algorithms.

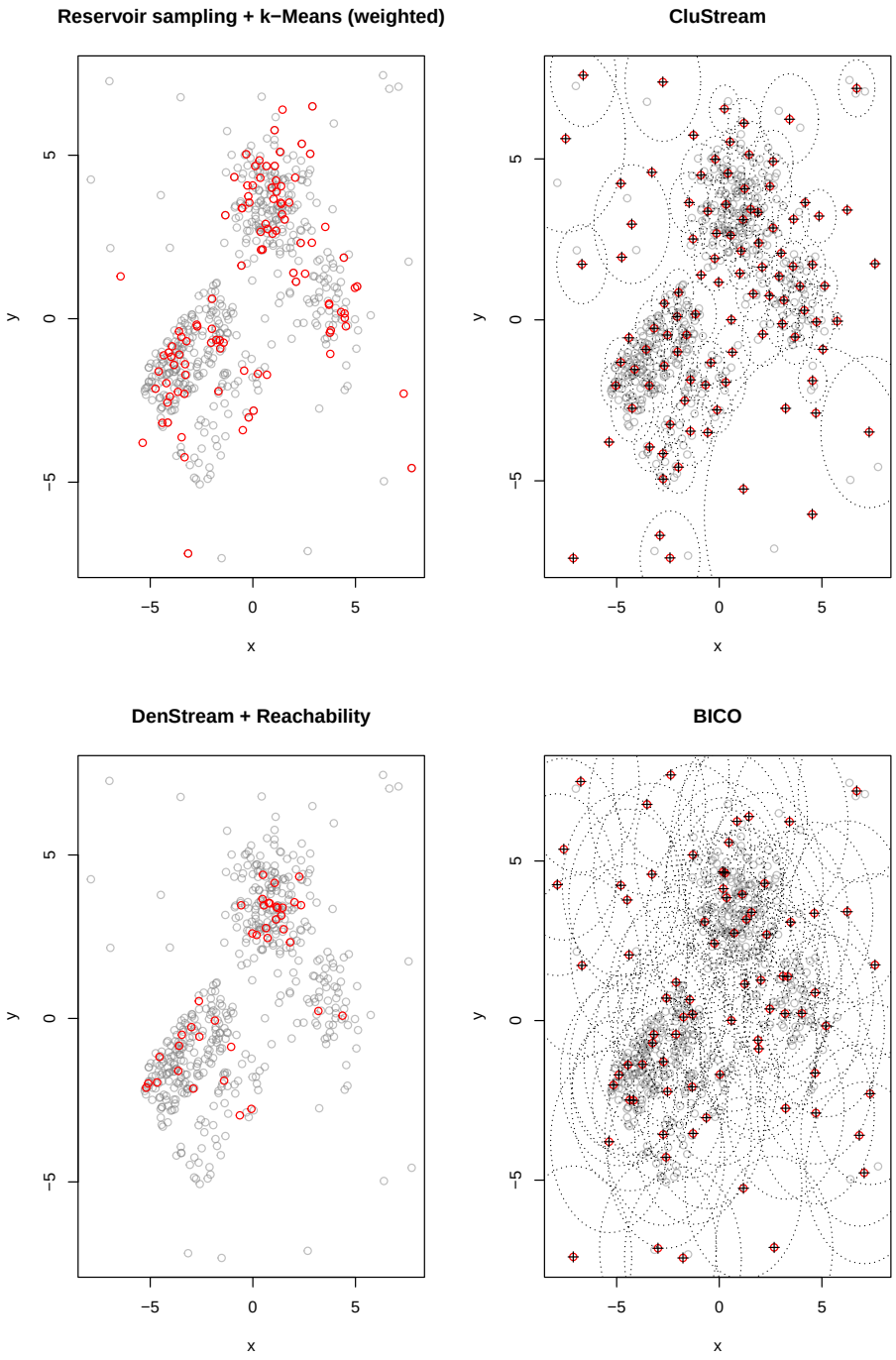


Figure 3: Micro-cluster assignment areas for different data stream clustering algorithms.

Figure 3 shows the assignment areas as dotted circles around micro-clusters. Not all algorithms provide assignment ares.

```
R> sapply(
+   algorithms,
+   FUN = function(a) {
+     reset_stream(stream, 1001)
+     evaluate_static(
+       a,
+       stream,
+       measure = c("numMicroClusters", "purity", "SSQ", "silhouette"),
+       n = 500,
+       assignmentMethod = "auto",
+       type = "micro"
+     )
+ })
```

```
                 Sample + k-means DenStream cluStream    Bico
numMicroClusters          100.000  39.00000   100.000  78.000
purity                      0.976   0.95105     0.973   0.958
SSQ                       114.768 269.80727    72.674 122.497
silhouette                  0.141  -0.00623     0.209   0.245
```

We need to be careful with the comparison of these numbers, since the depend heavily on the number of micro-clusters with more clusters leading to a better value. Therefore, a comparison with DenStream is not valid. We can compare the measures, of the other algorithms since the number of micro-clusters is close. Sampling produces very good values for purity, BICO achieves the highest average silhouette coefficient and CluStream produces the lowest sum of squares. For better results more data and cross-validation could be used.

## 3. Outlier detection

To support the outlier detection area, **streamMOA** contains a wrapper to the MOA implementation of the Micro-cluster Continuous Outlier Detector (MCOD). To demonstrate a synergy of outlier detection capabilities between **stream** and **streamMOA** packages, we bring two basic examples. First, we create a fixed data stream with noise outliers that are well separated from the clusters using Mahalanobis distance.

```
R> library(stream)
R> set.seed(1000)
R> stream <- DSD_Gaussians(k = 3, d = 2,
+             variance_limit = c(0.1, 1),
+             space_limit = c(0, 30),
+             noise = .01,
+             noise_limit = c(0, 30),
+             noise_separation = 6,
```

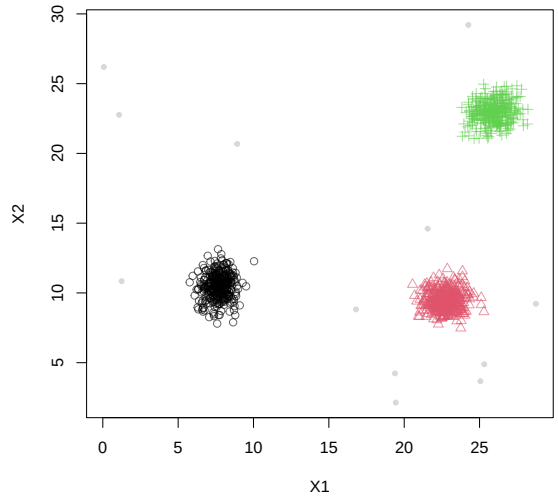


Figure 4: Data points from `DSD_Gaussians` having 3 clusters and 1% is outliers (noise).

```
+             separation_type = "Mahalanobis"
+   ) %>% DSD_Memory(n = 1000)

R> plot(stream, n = 1000)
```

The generated stream can be seen in Figure 4.

Then we define a `DSC_MCOD` clusterer. Since this is a single-pass clusterer `DSC_SinglePass`, we do not need to update the model first, we can immediately call evaluation.

```
R> reset_stream(stream)
R> mic_c <- DSOutlier_MCOD(r = 2, w = 1000)
R> evaluate_static(
+   mic_c,
+   stream,
+   n = 1000,
+   type = "micro",
+   measure = c("crand", "outlierjaccard")
+ )

Evaluation results for micro-clusters.
Points were assigned to micro-clusters.

         cRand outlierJaccard
        0.4476         0.0361
attr(,"type")
[1] "micro"
attr(,"assign")
[1] "micro"
```

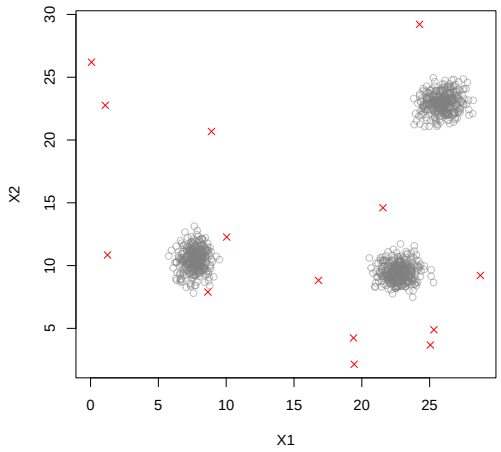


Figure 5: MCOD outlier detection.

```
R> reset_stream(stream)
R> plot(mic_c, stream, n = 1000)
```

In Figure 5 we can see micro-cluster and outlier assignments for the generated data stream.

## Acknowledgments

This work is supported in part by the U.S. National Science Foundation as a research experience for undergraduates (REU) under contract number IIS-0948893 and by the National Human Genome Research Institute under contract number R21HG005912.

### Affiliation:

Michael Hahsler
Engineering Management, Information, and Systems
Lyle School of Engineering
Southern Methodist University
P.O. Box 750122
Dallas, TX 75275-0122
E-mail: mhahsler@lyle.smu.edu
URL: http://michael.hahsler.net

John Forrest
Microsoft Corporation
E-mail: jforrest@microsoft.com

Matthew Bolaños
Microsoft Corporation
E-mail: mbolanos@curiouscrane.com