Data Stream Clustering of Large Non-Streaming Data Sets

Matthew Bolaños, John Forrest & Michael Hahsler

Intelligent Data Analysis Group
Southern Methodist University

36th Annual Conference of the German Classification Society
Hildesheim, Germany
August 1–3, 2012

SMU | Bobby B. Lyle School of Engineering

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.

Table of Contents

1. Motivation
2. Conventional Clustering
3. Data Stream Clustering
4. R-extension package **stream**
5. Empirical Study
6. Conclusion

Motivation

Clustering large data sets is important for many applications:

- Cluster users to find groups (Amazon, Netflix, etc.)
- Cluster documents to organize the information space (PubMed, arXiv, etc.)
- Cluster meteorological data to categorize phenomena (e.g., storms)
- Cluster genetic sequences for classifying organisms.

Data is typically large with respect to

- Number of data points
- Number of expected clusters
- Dimensionality

Motivation

Clustering large data sets is important for many applications:

- Cluster users to find groups (Amazon, Netflix, etc.)
- Cluster documents to organize the information space (PubMed, arXiv, etc.)
- Cluster meteorological data to categorize phenomena (e.g., storms)
- Cluster genetic sequences for classifying organisms.

Data is typically large with respect to

- Number of data points
- Number of expected clusters
- Dimensionality

Problems: Conventional clustering methods become very expensive and large data is noisy.

Investigate data stream clustering methods.

Table of Contents

1 Motivation

2 Conventional Clustering

3 Data Stream Clustering

4 R-extension package **stream**

5 Empirical Study

6 Conclusion

Clustering

Clustering [8], the grouping of similar objects, is an important task in statistics/data mining/machine learning.

Hard clustering with noise

Algorithms cluster a data set $\mathbf{D}$ into k subsets

$$\zeta = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$$

with

$$\bigcup_{i=1}^{k} \mathcal{C}_i \subseteq \mathbf{D}$$

and

$$\mathcal{C}_i \cap \mathcal{C}_j = \emptyset \text{ for all } i, j \in \{1, 2, \ldots, k\} \text{ and } i \neq j$$

- The number k is often user-defined.
- Not all data points in $\mathbf{D}$ have to be assigned to clusters. Unassigned points are assumed to be noise.

k-means

Tries to find a grouping which minimizes the squared distance of each data point to the representative (center) of it's cluster.

$$\arg\min_{\zeta} \sum_{i=1}^{k} \sum_{x_j \in \mathcal{C}_i} ||x_j - \mu_i||_2$$

where μ_i is the mean of the points in $\mathcal{C}_i$ and $||x||_2$ is the Euclidean norm.

- Simple iterative heuristic.
- Small overhead in terms of space complexity (store centers).
- Time complexity: $O(nkb)$ where b is the number of iterations.
- Assumes spherical shaped clusters.
- Algorithm requires access to all data points in each iteration.
- Problem when data set does not fit into main memory.

Hierarchical Clustering

Agglomerative hierarchical clustering

1. Start with n sets containing one point each.
2. Iteratively combine the two closest sets till there is only one set left. Distance between sets is computed using a user-specified linkage function (single, average, complete link).

- Depending on the used linkage function, hierarchical clustering can find spherical clusters (average and complete link) or arbitrarily shaped clusters (single link).
- Problem: Distance computation takes $O(n^2)$ time and space.

Density-based Clustering

A typical density-based clustering algorithm is DBSCAN [6].

DBSCAN

1. Determines for each point if it is in a sufficiently dense area (MinPts withing a radius ϵ)
2. Join neighbors into clusters using the idea of reachability (shared neighbors)

- Like hierarchical clustering with single link, DBSCAN can also find arbitrary shaped clusters.
- Problem: Time complexity is $O(n\,log(n))$ or $O(n^2)$.

Clustering Large Data

- Iterative extensions of classical algorithms.
- Parallel/distributed computing (e.g., k-means with MapReduce)

- Cheap preprocessing of the data before clustering
  - Sampling (e.g., CLARA [9])
  - Single pass algorithms like BIRCH [17]
  - Canopy clustering [11]
  - Online component of data stream clustering algorithms.

BIRCH

- BIRCH [17] builds a height balanced tree (also known as a CF tree) to store information about subclusters.
- New data points are either added to an existing subcluster or create a new one.
- After all data points are added to the CF tree, the user is presented with a list of subclusters which can be recluster with another method like k-means.

- Time complexity is $O(n)$
- Space complexity is independent of n and only depends on k', the number of subclusters.
- Subclusters in the CF tree are very similar to micro-clusters in data stream clustering.

Table of Contents

1 Motivation

2 Conventional Clustering

3 Data Stream Clustering

4 R-extension package **stream**

5 Empirical Study

6 Conclusion

Data Stream Clustering

Over the last 10 years many algorithms for clustering data streams have been proposed [7, 1, 2, 4, 12, 13, 15, 14, 16].

Typical data stream clustering approach

1. **Online:** Summarize the data by a set of k' micro-clusters using a single pass over the data.
2. **Offline:** Recluster the k' micro-clusters into k final (macro-) clusters ($k \ll k'$). Micro-clusters are regarded as pseudo-points for a conventional clustering algorithm.

Note: Data stream clustering also deals with temporal aspects (e.g., concept drift) but we are here only interested in their ability to summarize large data.

CluStream and ClusTree

CluStream [1] was the first micro-cluster-based algorithm. A micro-cluster is a local summary typically represented by position, dispersion and weight.

CluStream

1. It collects data to creates an initial set of micro-clusters using k-means.
2. For each point: either add the point to a micro-cluster or create a new micro-cluster.
3. If a new micro-cluster was created then delete or merge micro-clusters to keep k' constant.

For a stable clustering with no merging and deletion of micro-clusters: Time complexity of $O(nk')$.

ClusTree [10] is similar to CluStream but maintains the micro-clusters in a tree structure for faster access.

DenStream

Similar to CluStream but DenStream [4] initializes the micro-cluster list using DBSCAN on an initial set of points and maintains two lists of micro-clusters.

- **Core-micro-clusters:** Dense clusters (defined by MinPoints)
- **Potential micro-clusters:** May develop into core-micro-clusters or be eventually deleted (outliers).

DenStream incurs more overhead compared to CluStream for maintaining the data structure and k' is not held constant.

Table of Contents

1 Motivation

2 Conventional Clustering

3 Data Stream Clustering

4 R-extension package **stream**

5 Empirical Study

6 Conclusion

The **stream** Package

The R-extension package **stream** provides an infrastructure for data stream mining.

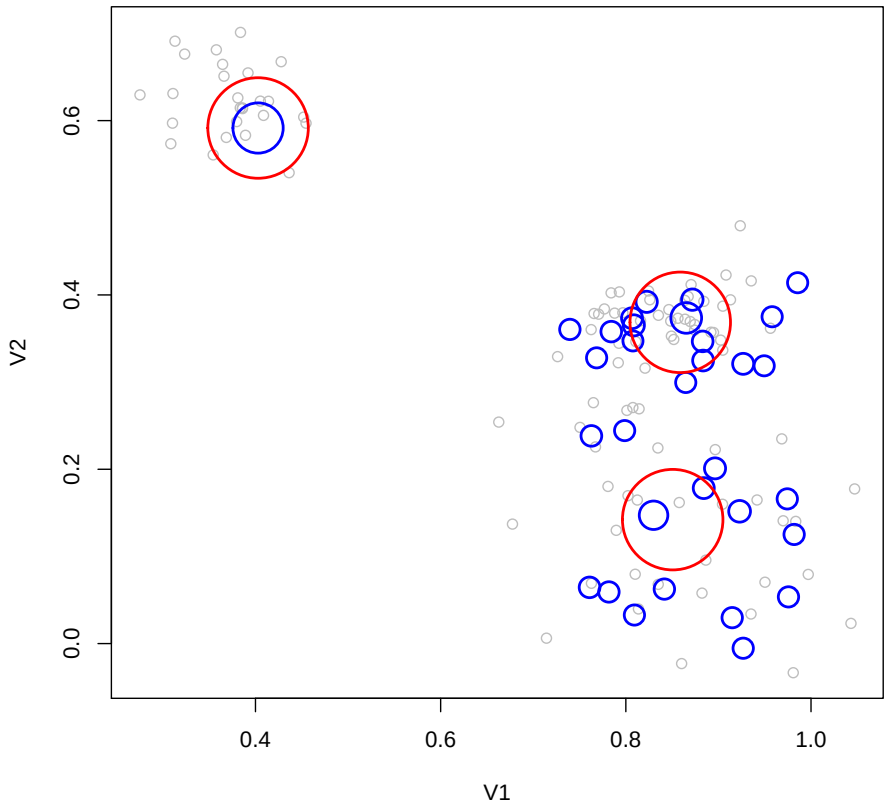
- Define and create synthetic data streams.
- Connect to real data streams (via connections)
- Implement data stream mining algorithms (currently clustering from MOA [3] and native in R)
- "Stream" interface to classical algorithms (k-means, sampling, etc.)
- Implement reclustering.
- Performance evaluation and support for systematic experimentation.

stream is currently under development and the development code is available at `http://R-Forge.R-Project.org/projects/clusterds/`.

Using the **stream** Package

```
library(stream)

### define a data stream
dsd <- DSD_Gaussian_Static(k=3)

### cluster using DenStream
micro <- DSC_DenStream(initPoints=100, horizon=1000)
cluster(micro, dsd, 1000)

### recluster using weighted k-means
macro <- DSC_KmeansW(k=3)
recluster(macro, micro)

### plot results
plot(get_points(dsd, 100), col="gray")
points(get_centers(micro), col="blue",
     cex=get_weights(micro, c(2,5)), lwd=2)
points(get_centers(macro), col="red", cex=10, lwd=2)
```

stream: Example Code Output




```
> get_evaluation(micro, dsd, n=100)
 recall     precision         f1        ssq       rand numCluster numClasses
 0.1000000  1.0000000  0.1818182  4.5551937  0.7776000 30.0000000  3.0000000
> get_evaluation(macro, dsd, n=100)
 recall     precision         f1        ssq       rand numCluster numClasses
 0.9876543  0.9912281  0.9894380  7.8401944  0.9874000  3.0000000  3.0000000
```

Table of Contents

1 Motivation

2 Conventional Clustering

3 Data Stream Clustering

4 R-extension package **stream**

5 Empirical Study

6 Conclusion

Empirical Study

How well do different "large data" preprocessing methods work?

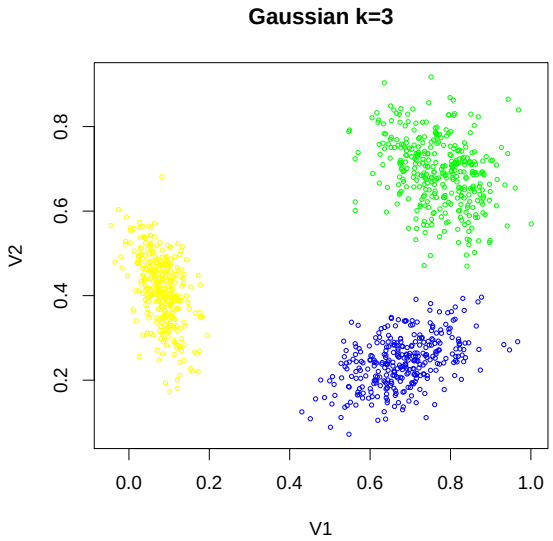
We will compare the quality of pure k-means (baseline) with

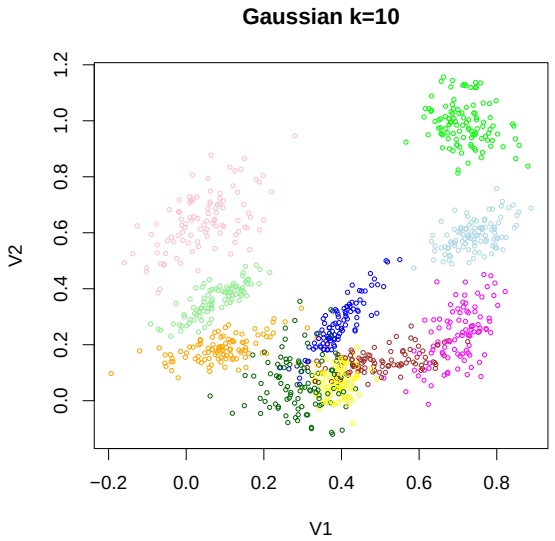
1. Sampling (reservoir sampling)
2. BIRCH (single pass)
3. CluStream
4. ClusTree
5. DenStream

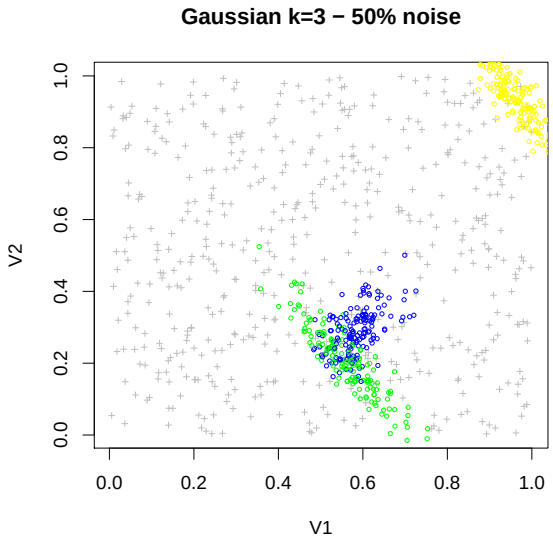
    + reclustering using k-means and hierarchical clustering.

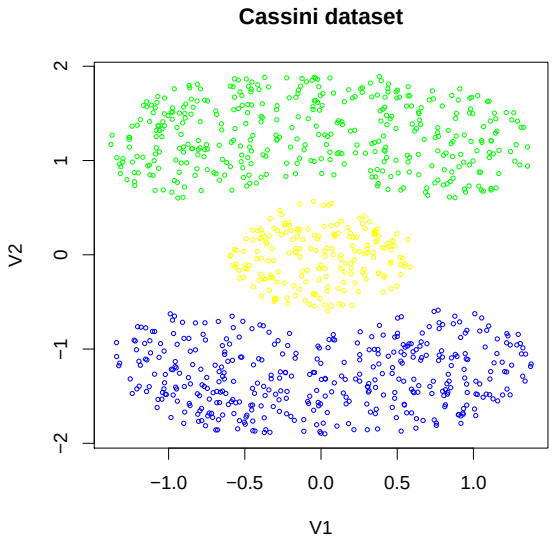
The experiments are performed using the R-extension package **stream**

Data Sets


Gaussian k=3
V2
V1
0.0
0.2
0.4
0.6
0.8
1.0


Gaussian k=10
V2
V1
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2


Gaussian k=3 − 50% noise
V2
V1
0.0
0.2
0.4
0.6
0.8
1.0


Cassini dataset
V2
V1
−2
−1
0
1
2
−1.0
−0.5
0.0
0.5
1.0

Setup

1. Data set $n = 1000$
2. Use each data stream clustering algorithm tuned to produce about 100 micro-clusters.
3. Recluster using k-means and hierarchical clustering using the known k.
4. Assign all points to the clusters (nearest neighbor) and report F1 [5] averaged over 10 runs.

$$\text{precision} = \frac{d}{b+d}$$

$$\text{recall} = \frac{d}{c+d}$$

$$\text{F1} = 2\,\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Actual/Pred.	Neg.	Pos.
Negative	a	b
Positive	c	d

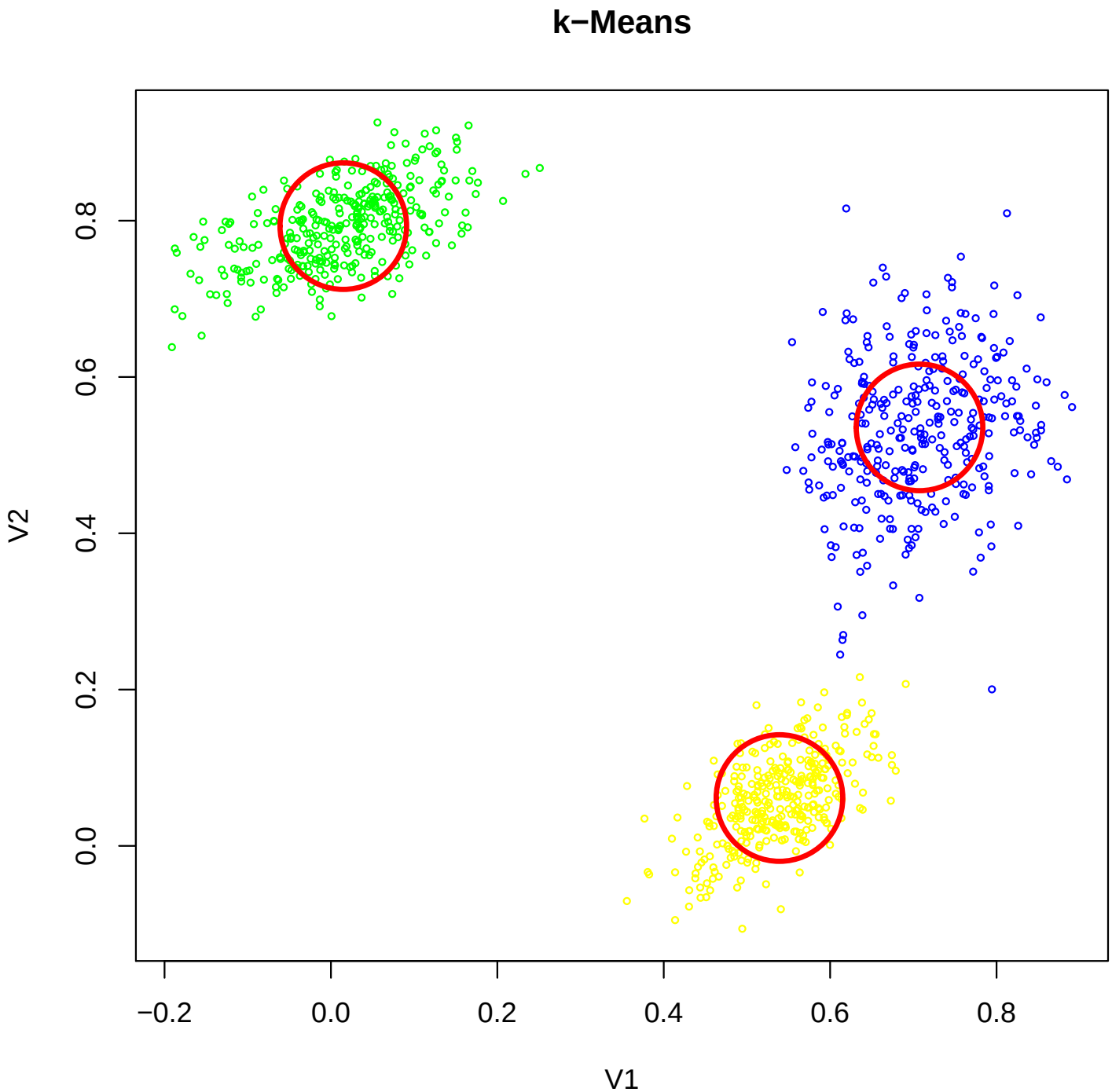
Table: Confusion Matrix

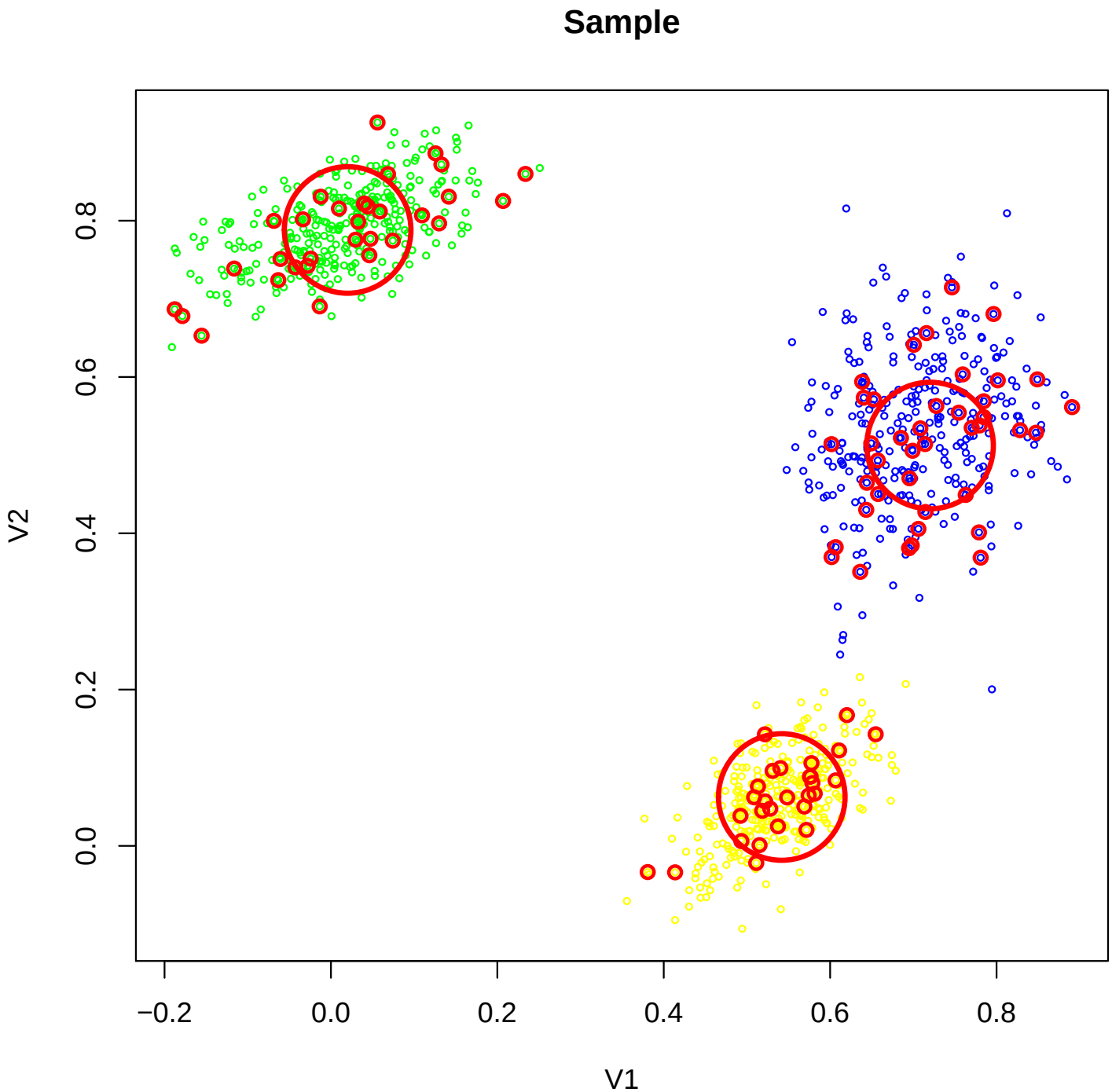
Results: Gaussian $k = 3$

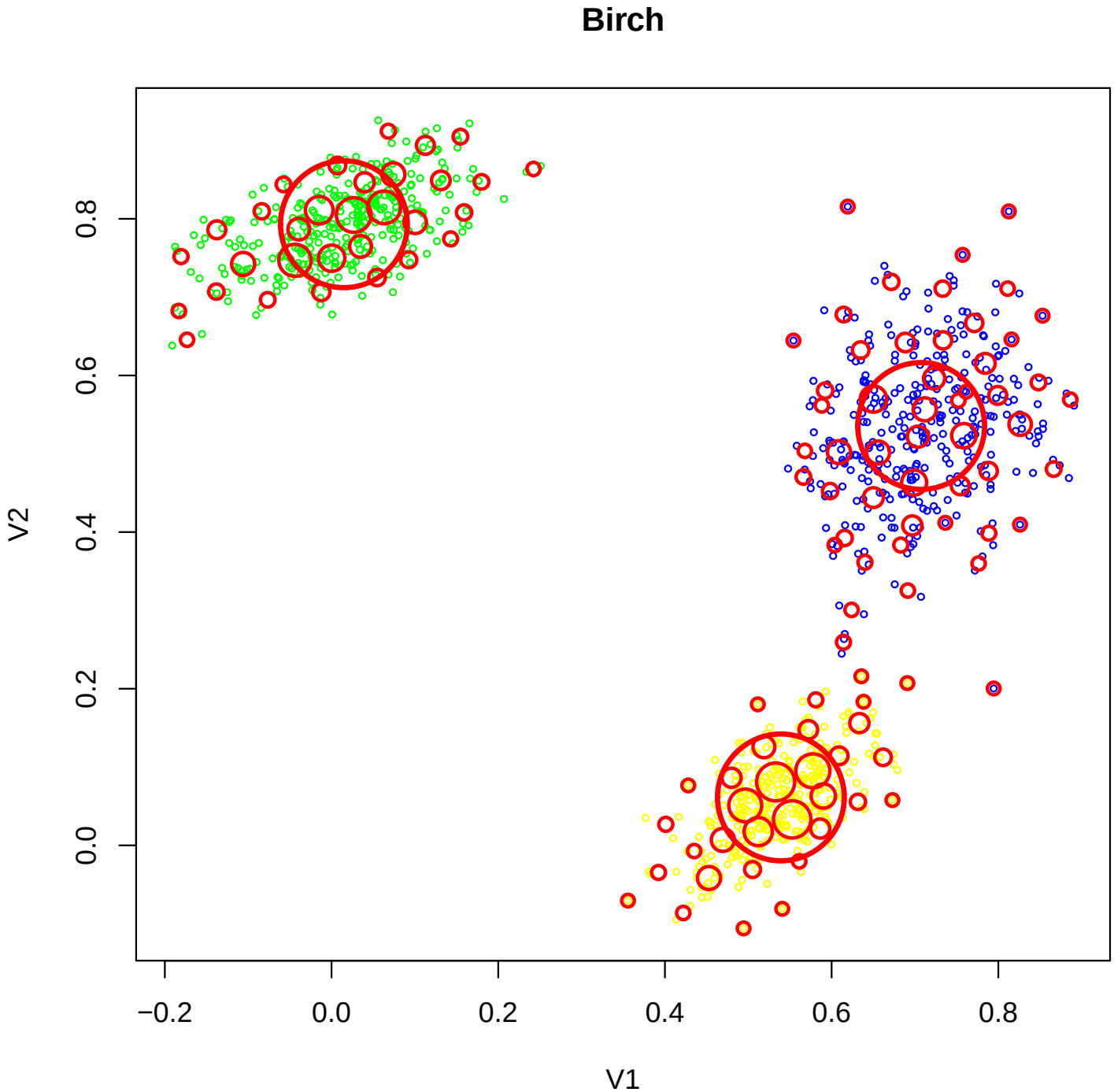
	kmeans	hc_complete	hc_average	hc_single
sample	0.934	0.866	0.894	0.786
birch	**0.936**	0.860	0.893	0.687
clustream	0.927	0.889	0.889	0.738
clustree	0.932	0.913	0.900	0.759
denstream	0.919	0.894	0.871	0.784

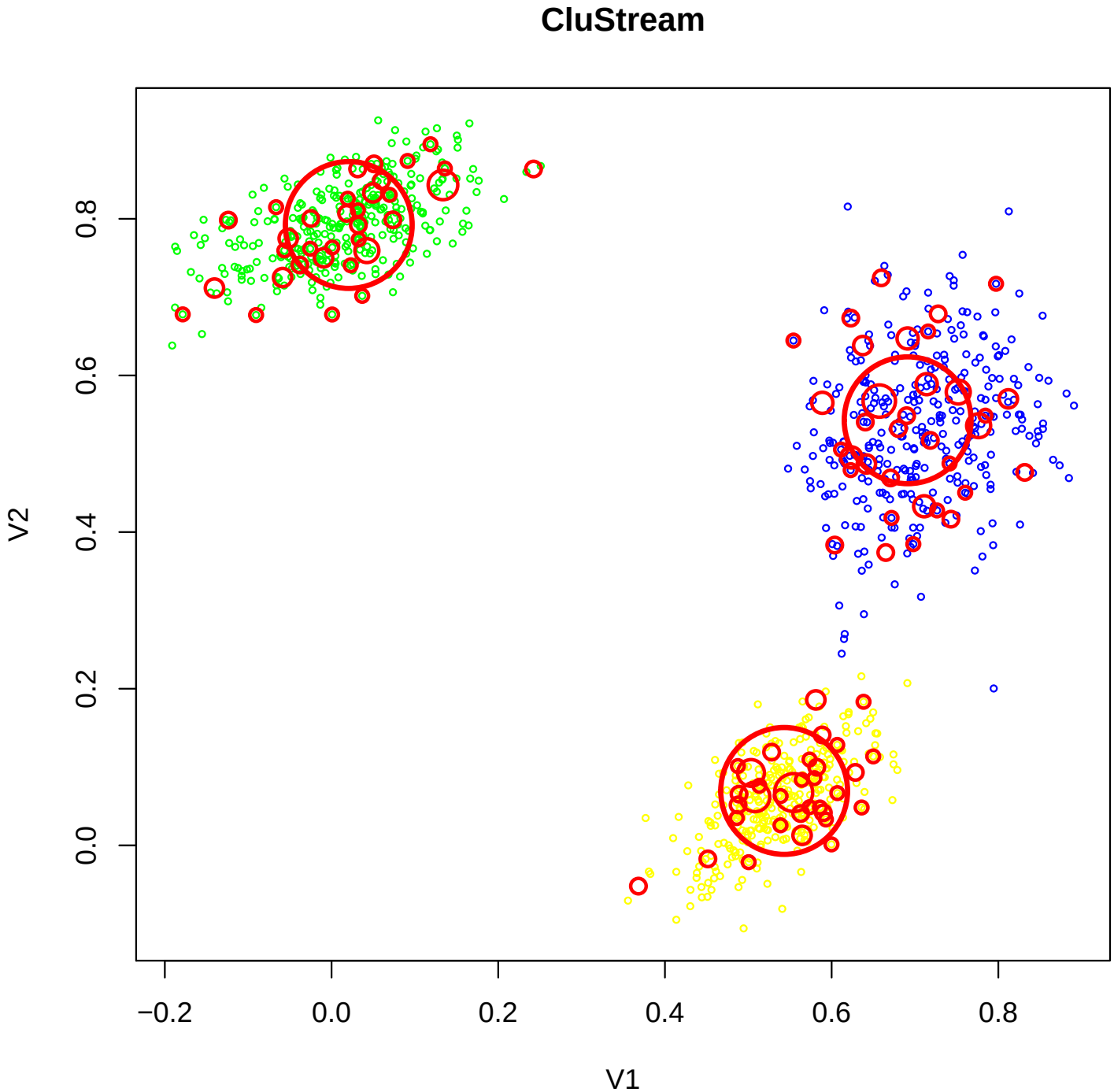
k-means: 0.912

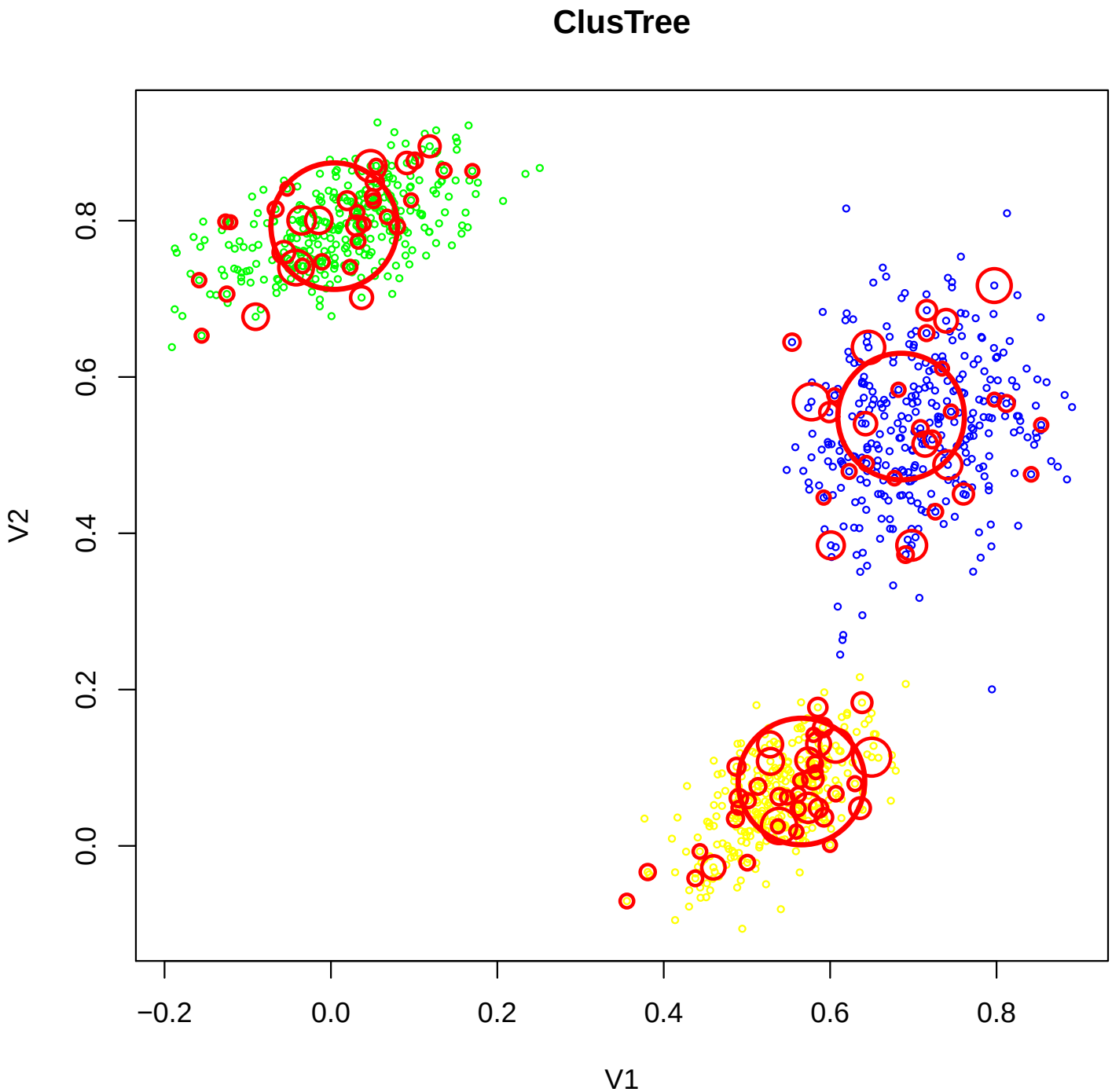
Almost no difference!

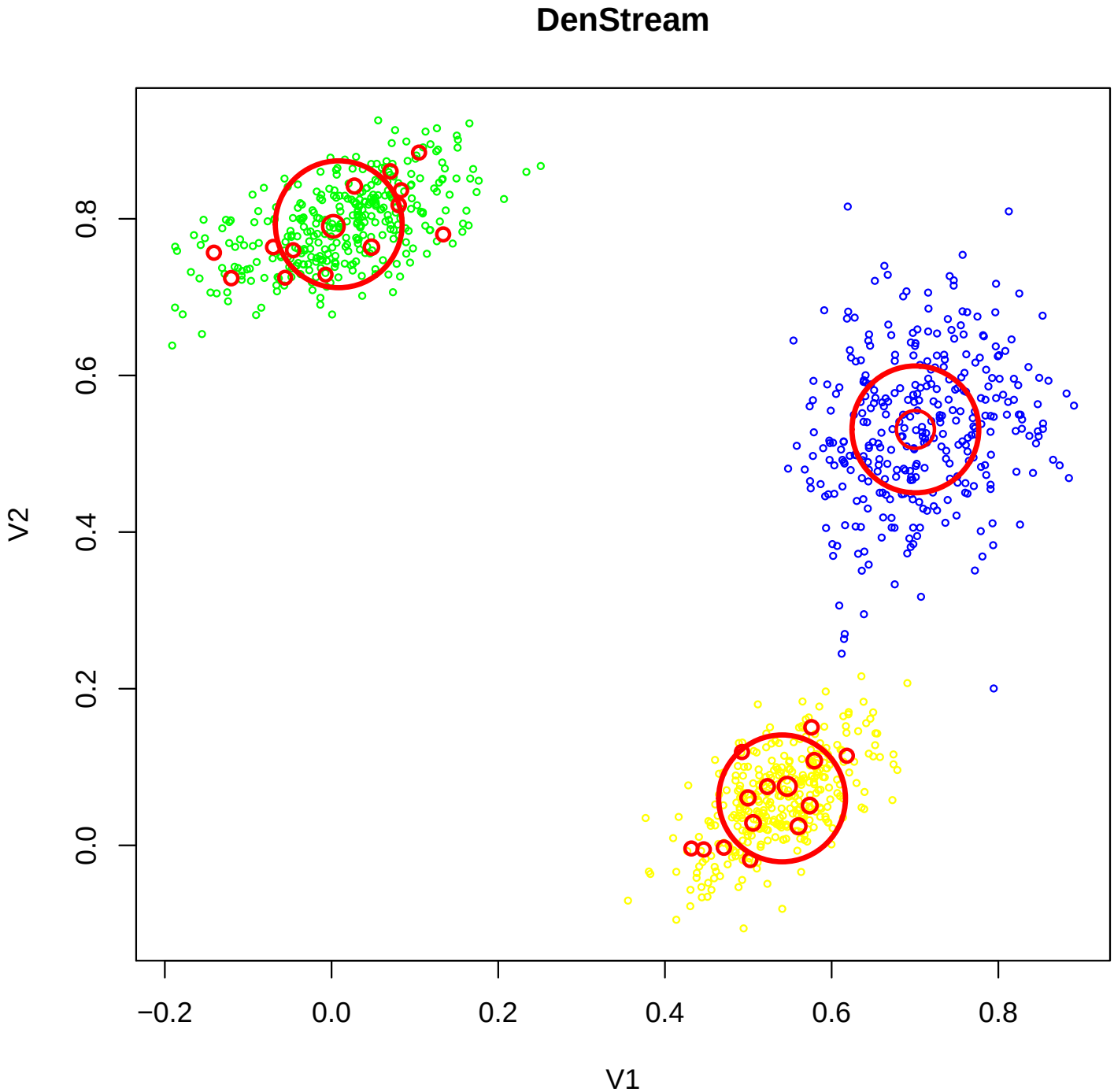

k−Means
V2
V1
0.0
0.2
0.4
0.6
0.8
−0.2
0.0
0.2
0.4
0.6
0.8


Sample
V2
V1
0.0
0.2
0.4
0.6
0.8
−0.2
0.0
0.2
0.4
0.6
0.8


Birch
V2
V1
0.0
0.2
0.4
0.6
0.8
−0.2
0.0
0.2
0.4
0.6
0.8


CluStream
V2
V1
0.0
0.2
0.4
0.6
0.8
−0.2
0.0
0.2
0.4
0.6
0.8


ClusTree
V2
V1
0.0
0.2
0.4
0.6
0.8
−0.2
0.0
0.2
0.4
0.6
0.8


DenStream
V2
V1
0.0
0.2
0.4
0.6
0.8
−0.2
0.0
0.2
0.4
0.6
0.8

Results: Gaussian $k = 10$

	kmeans	hc_complete	hc_average	hc_single
sample	0.759	0.747	0.739	0.662
birch	**0.785**	0.706	0.740	0.557
clustream	0.760	0.743	0.747	0.618
clustree	0.766	0.748	0.740	0.640
denstream	0.781	0.753	0.753	0.681

k-means: 0.777

Results: Gaussian $k = 10$, $d = 10$

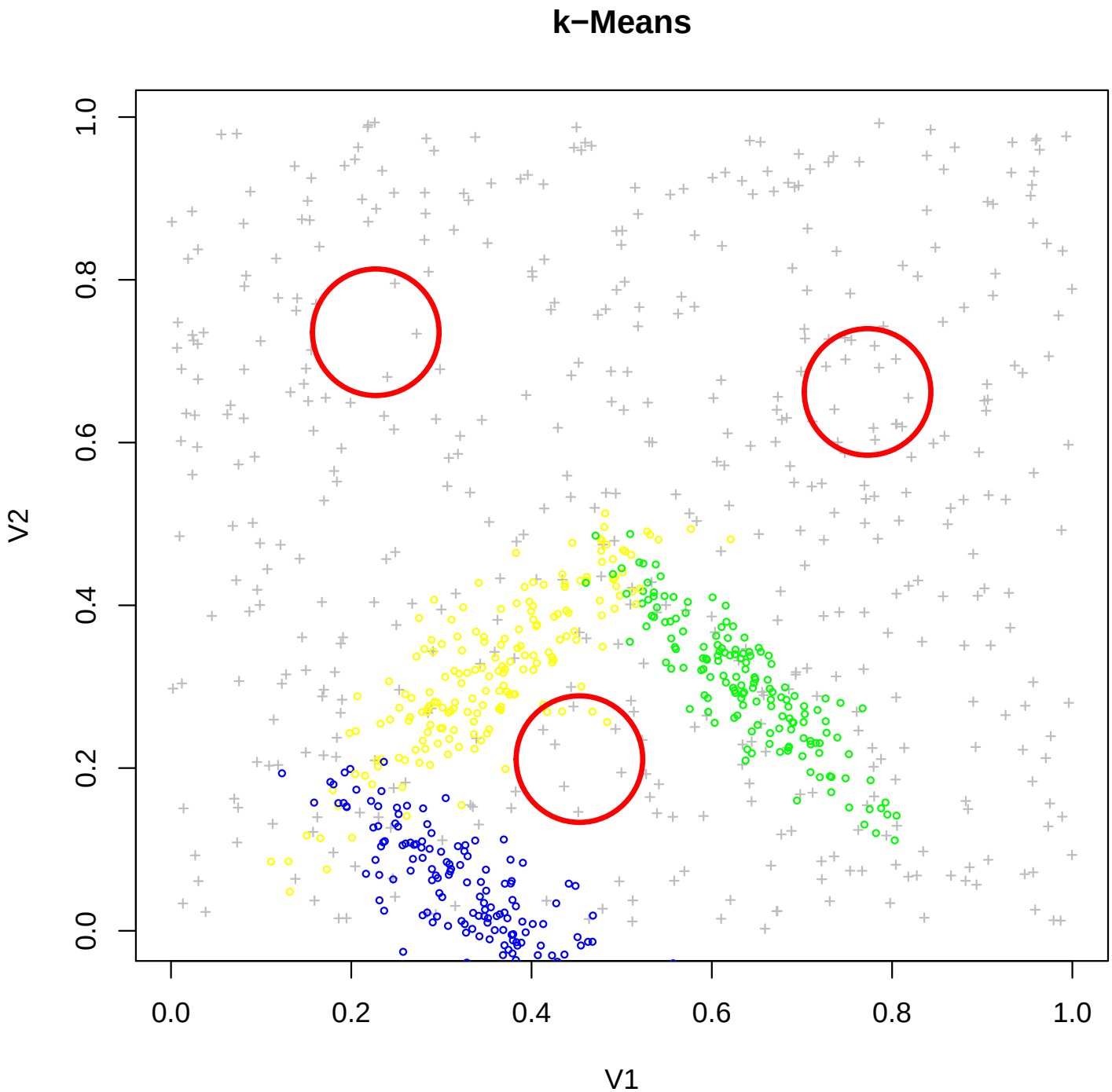
	kmeans	hc_complete	hc_average	hc_single
sample	0.954	**1.000**	**1.000**	**1.000**
birch	**1.000**	**1.000**	**1.000**	**1.000**
clustream	0.948	**1.000**	**1.000**	**1.000**
clustree	0.978	**1.000**	**1.000**	**1.000**
denstream	0.996	0.999	0.999	0.980

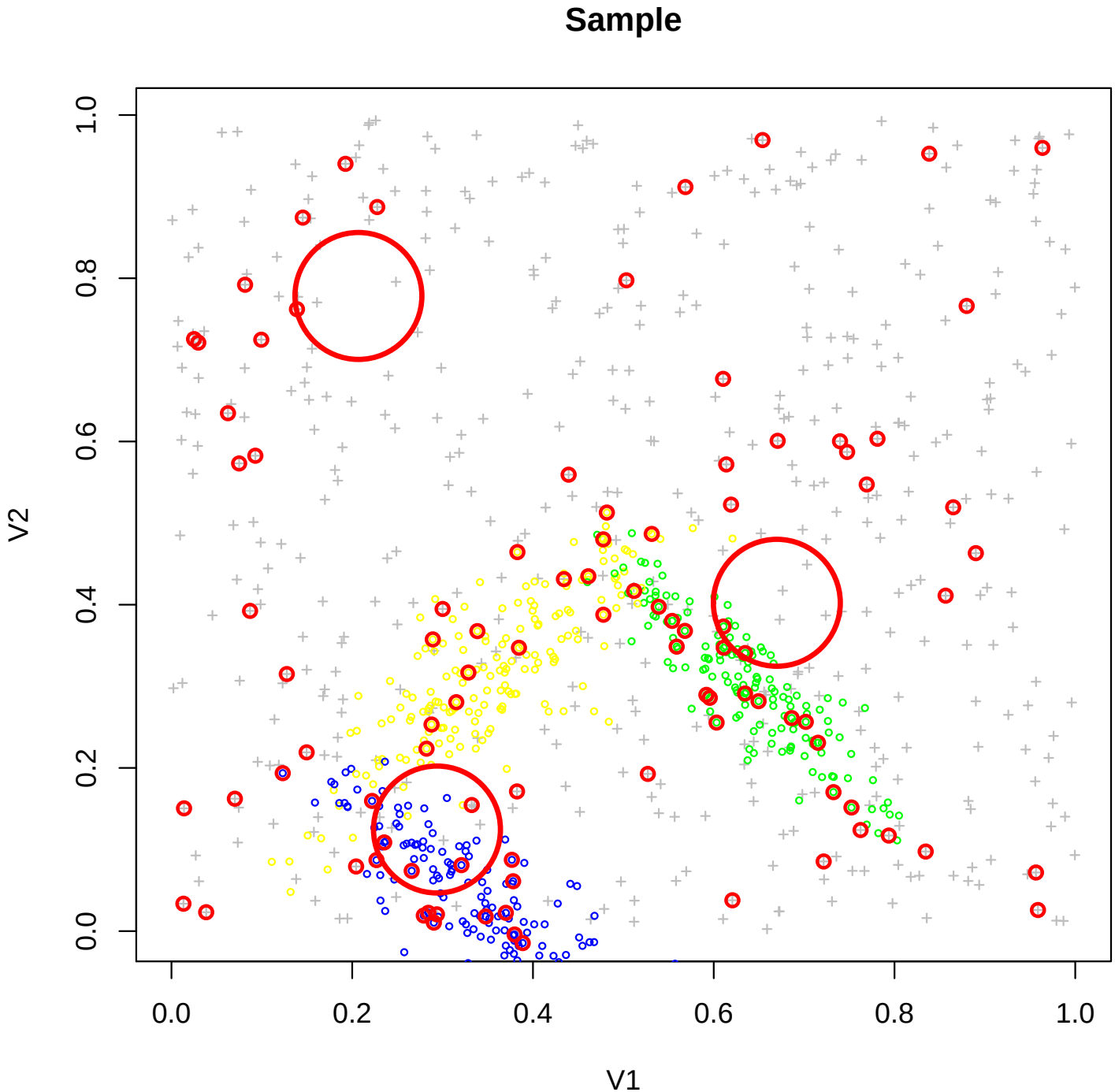
k-means: 0.910

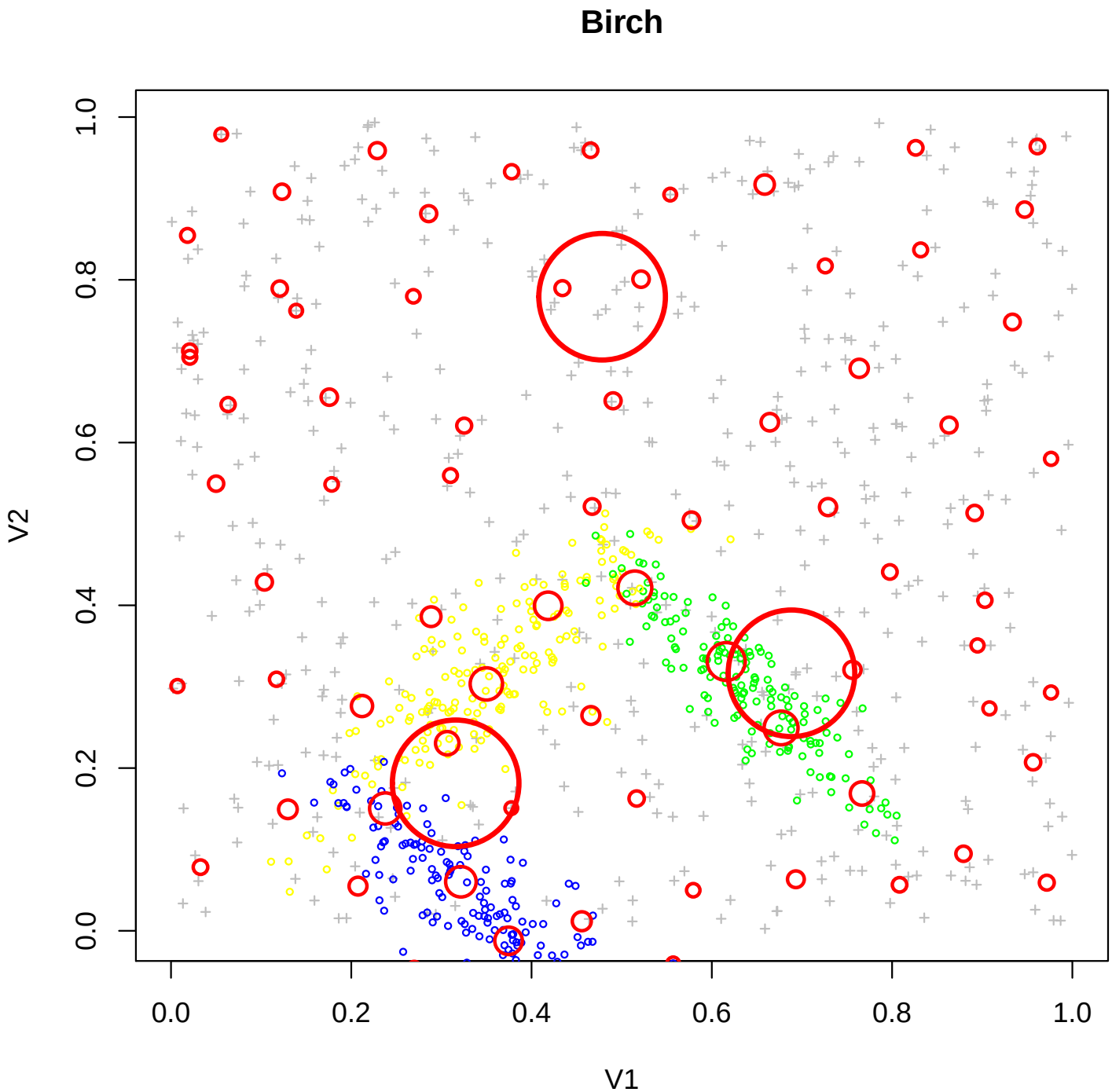
Results: Gaussian $k = 3$, 50% noise

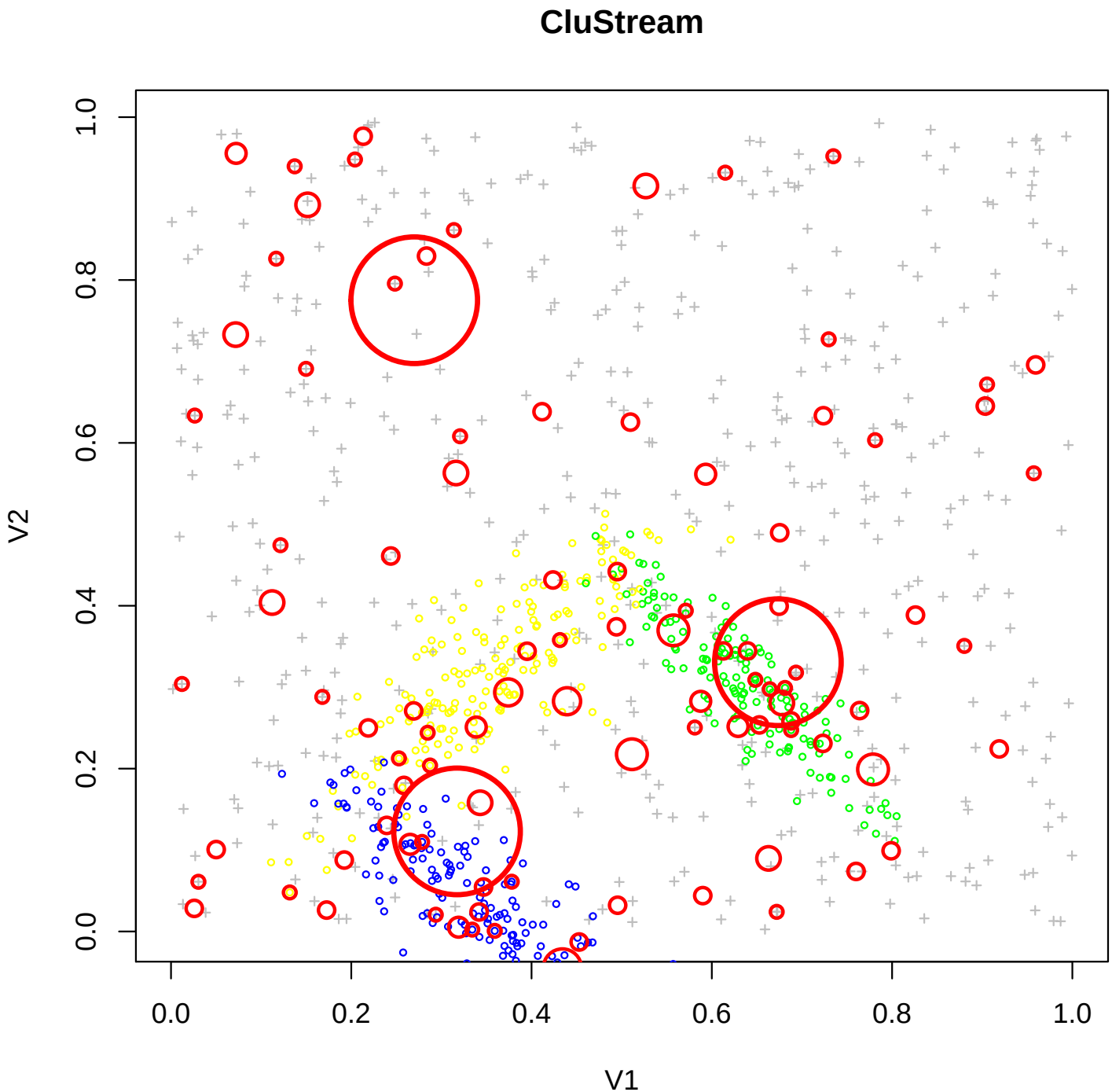
	kmeans	hc_complete	hc_average	hc_single
sample	0.672	0.679	0.575	0.351
birch	0.709	0.689	0.627	0.293
clustream	0.684	0.616	0.684	0.275
clustree	0.651	0.617	0.667	0.304
denstream	**0.784**	0.675	0.615	0.538

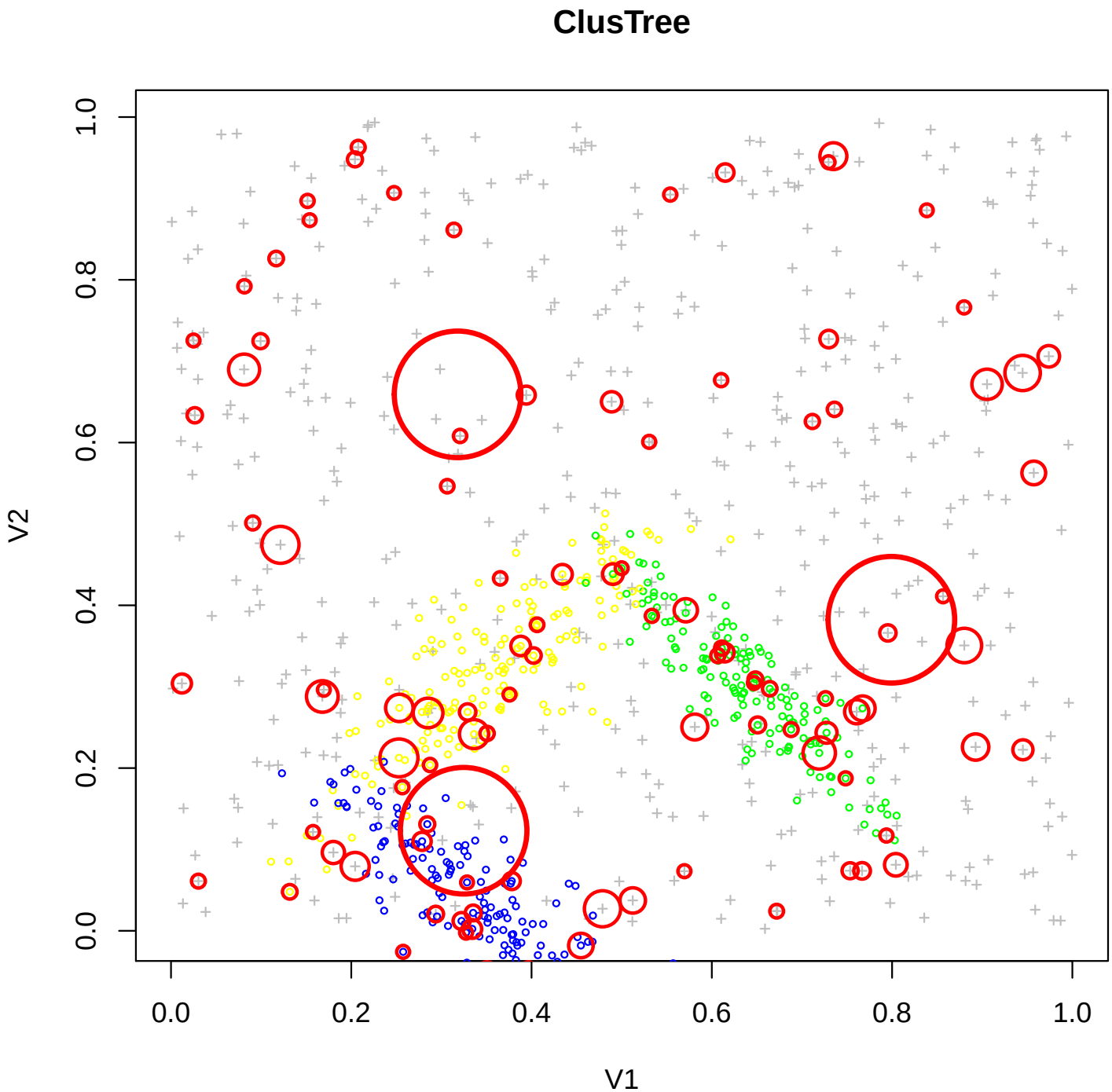
k-means: 0.734

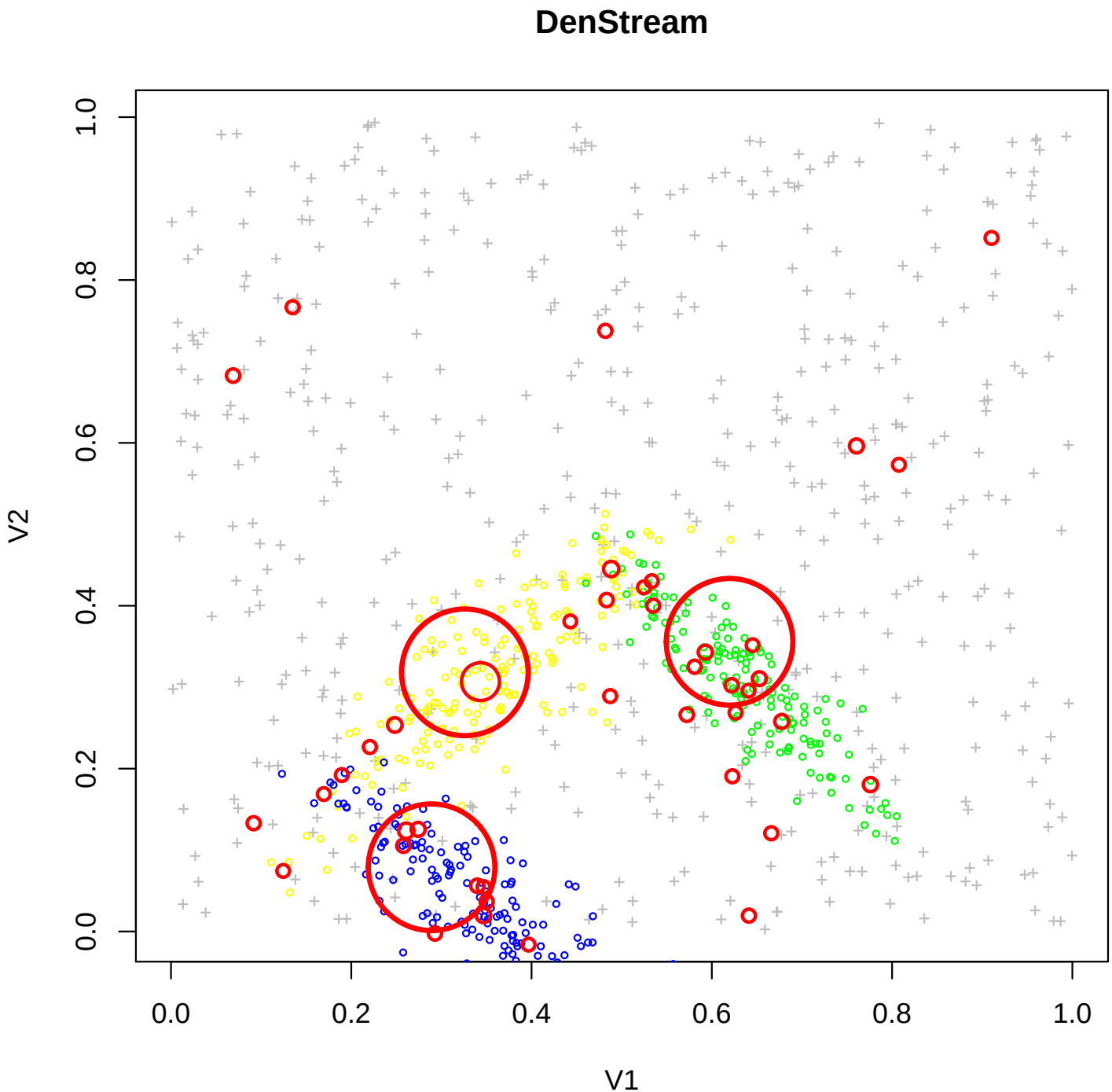

k−Means
V2
V1
0.0
0.2
0.4
0.6
0.8
1.0


Sample
V2
V1
0.0
0.2
0.4
0.6
0.8
1.0


Birch
V2
V1
0.0
0.2
0.4
0.6
0.8
1.0


CluStream
V2
V1
0.0
0.2
0.4
0.6
0.8
1.0


ClusTree
V2
V1
0.0
0.2
0.4
0.6
0.8
1.0


DenStream
V2
V1
0.0
0.2
0.4
0.6
0.8
1.0

Results: Gaussian $k = 10$, 50% noise

	kmeans	hc_complete	hc_average	hc_single
sample	0.659	0.639	0.600	0.346
birch	0.718	0.641	0.644	0.366
clustream	0.696	0.639	0.605	0.357
clustree	0.675	0.639	0.620	0.364
denstream	0.668	0.604	0.622	0.506

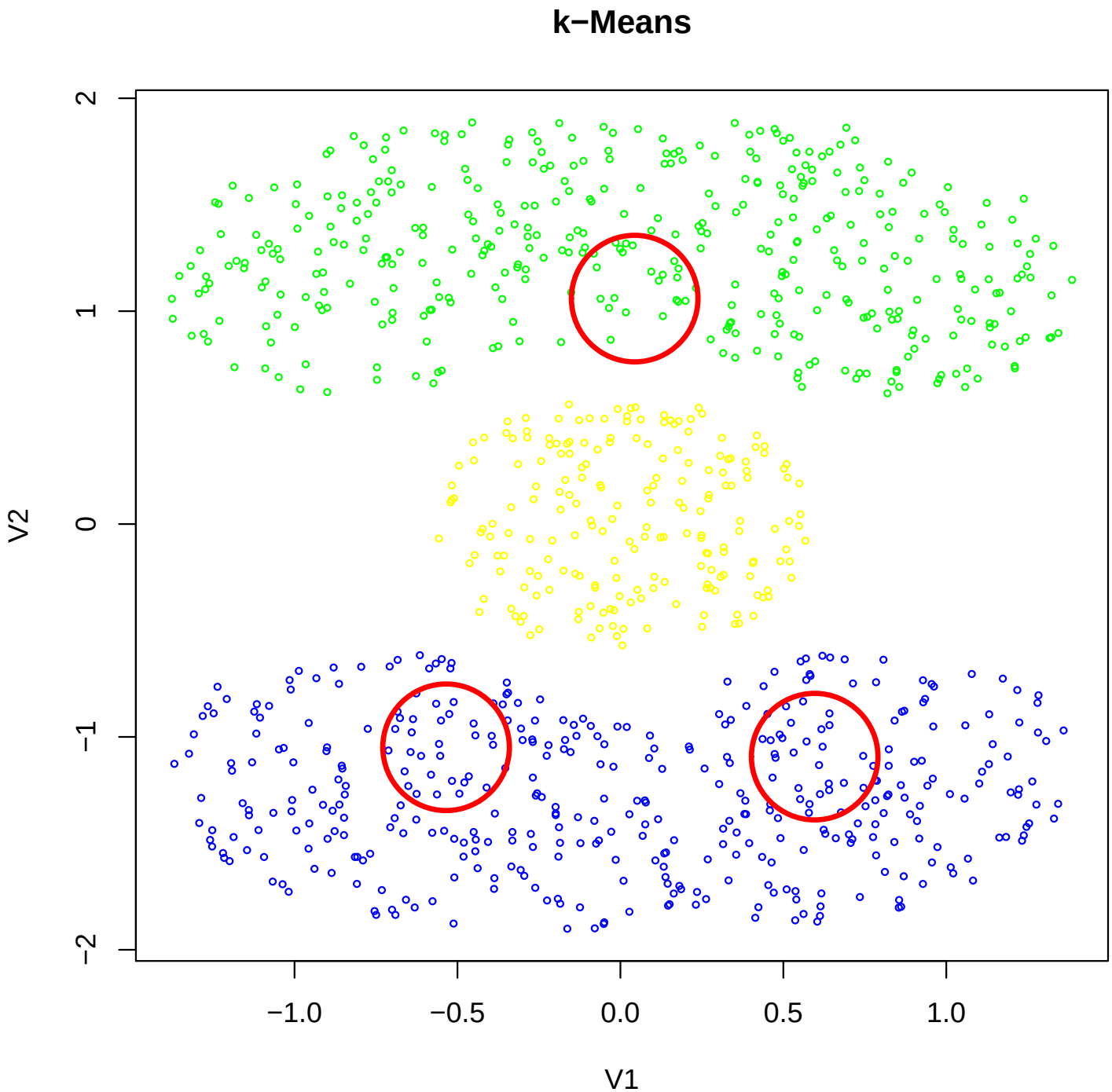
k-means: 0.688

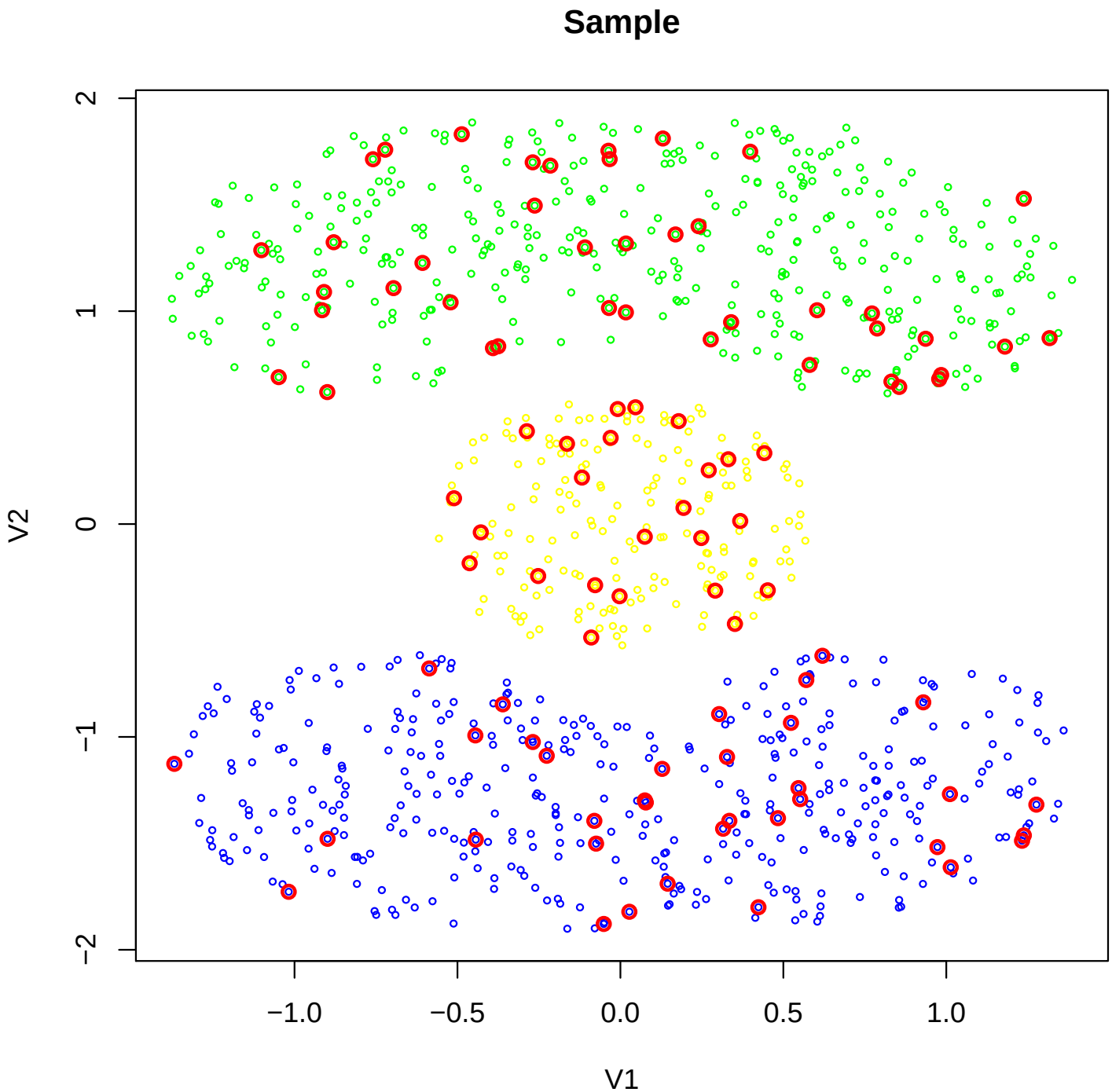
All methods perform poorly!

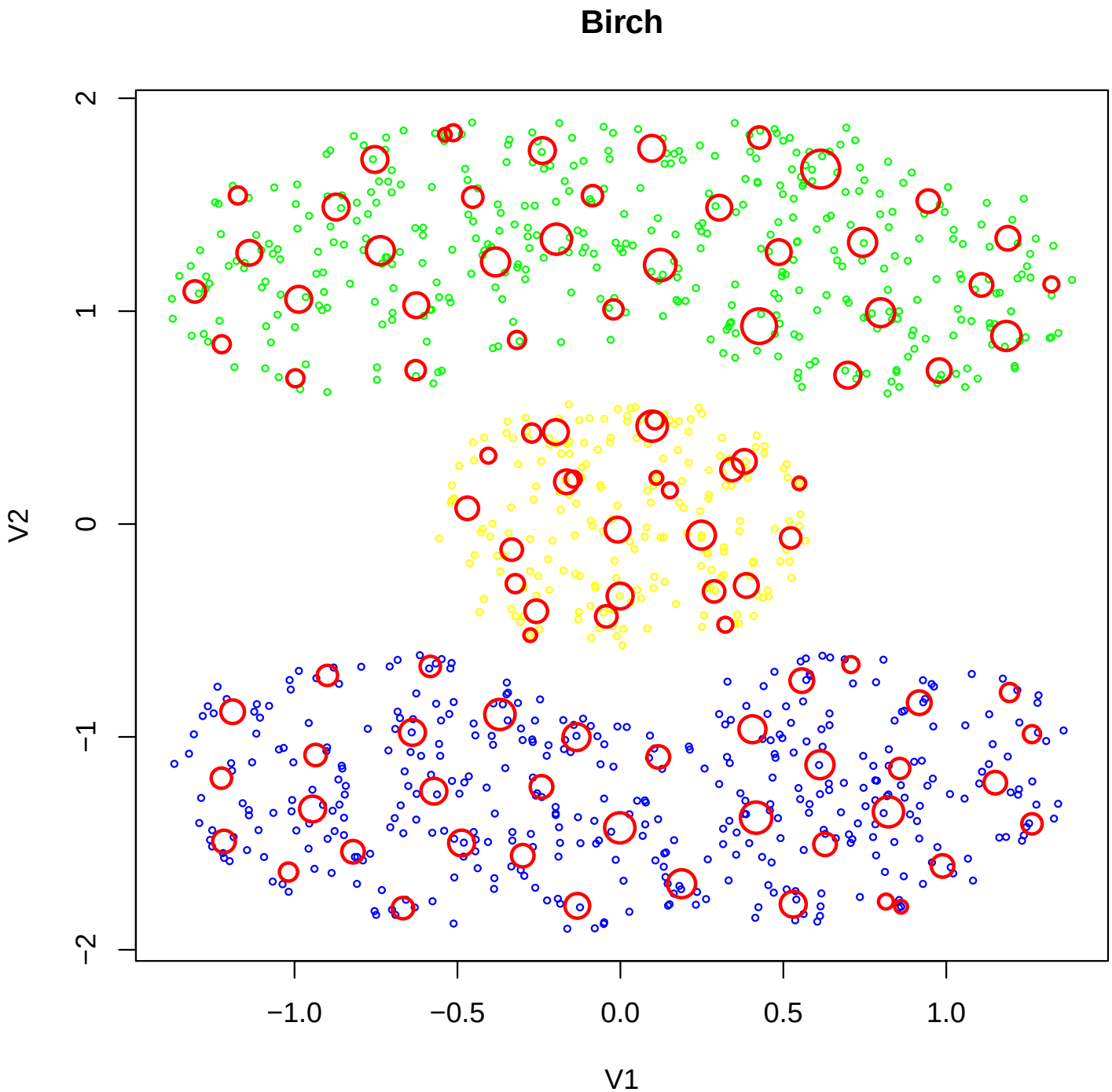
Results: Cassini

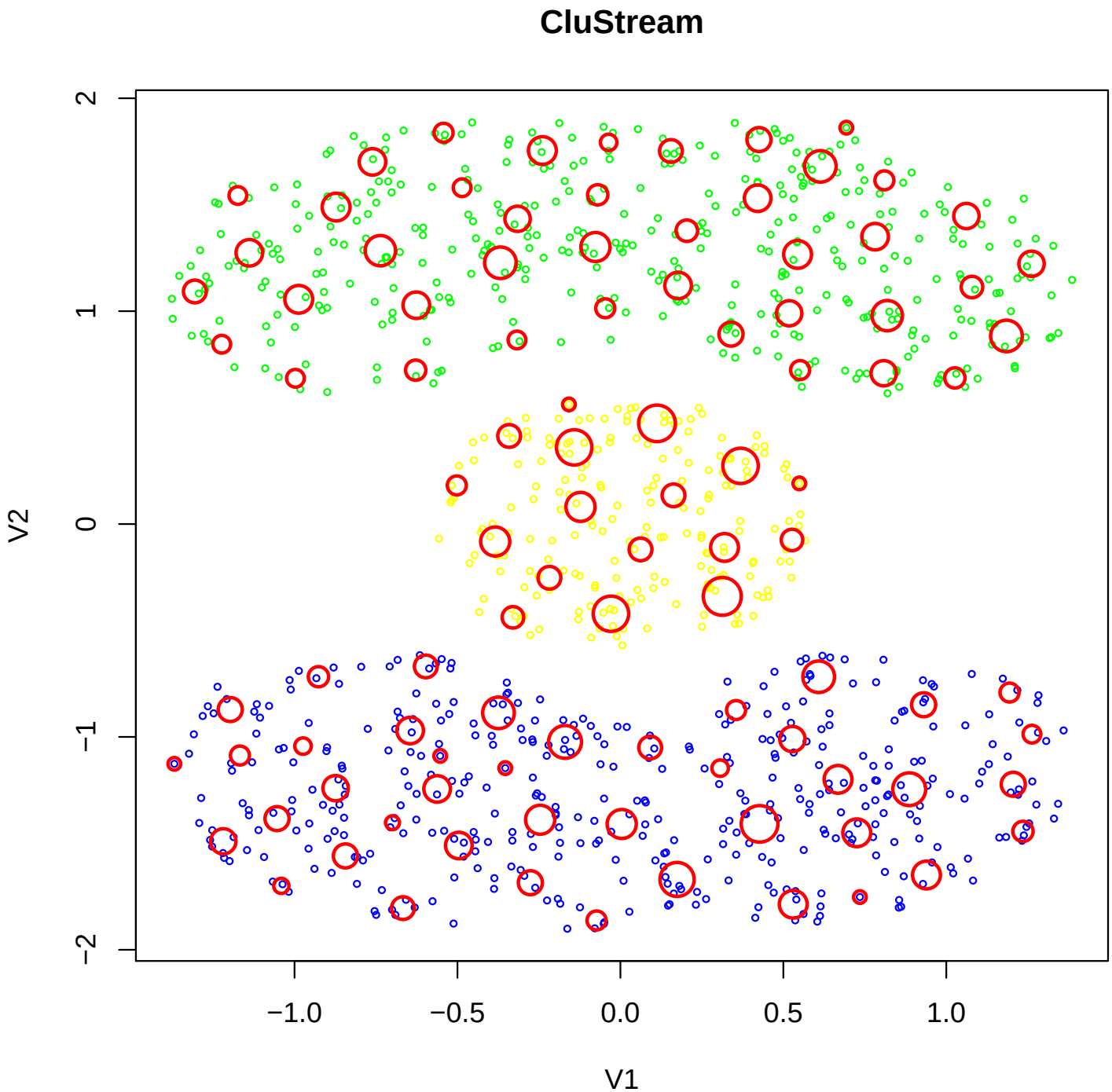
	kmeans	hc_complete	hc_average	hc_single
sample	0.783	0.776	0.826	0.637
birch	0.775	0.786	0.936	0.928
clustream	0.755	0.765	0.932	**0.952**
clustree	0.758	0.776	0.863	0.649
denstream	0.813	0.807	0.813	0.653

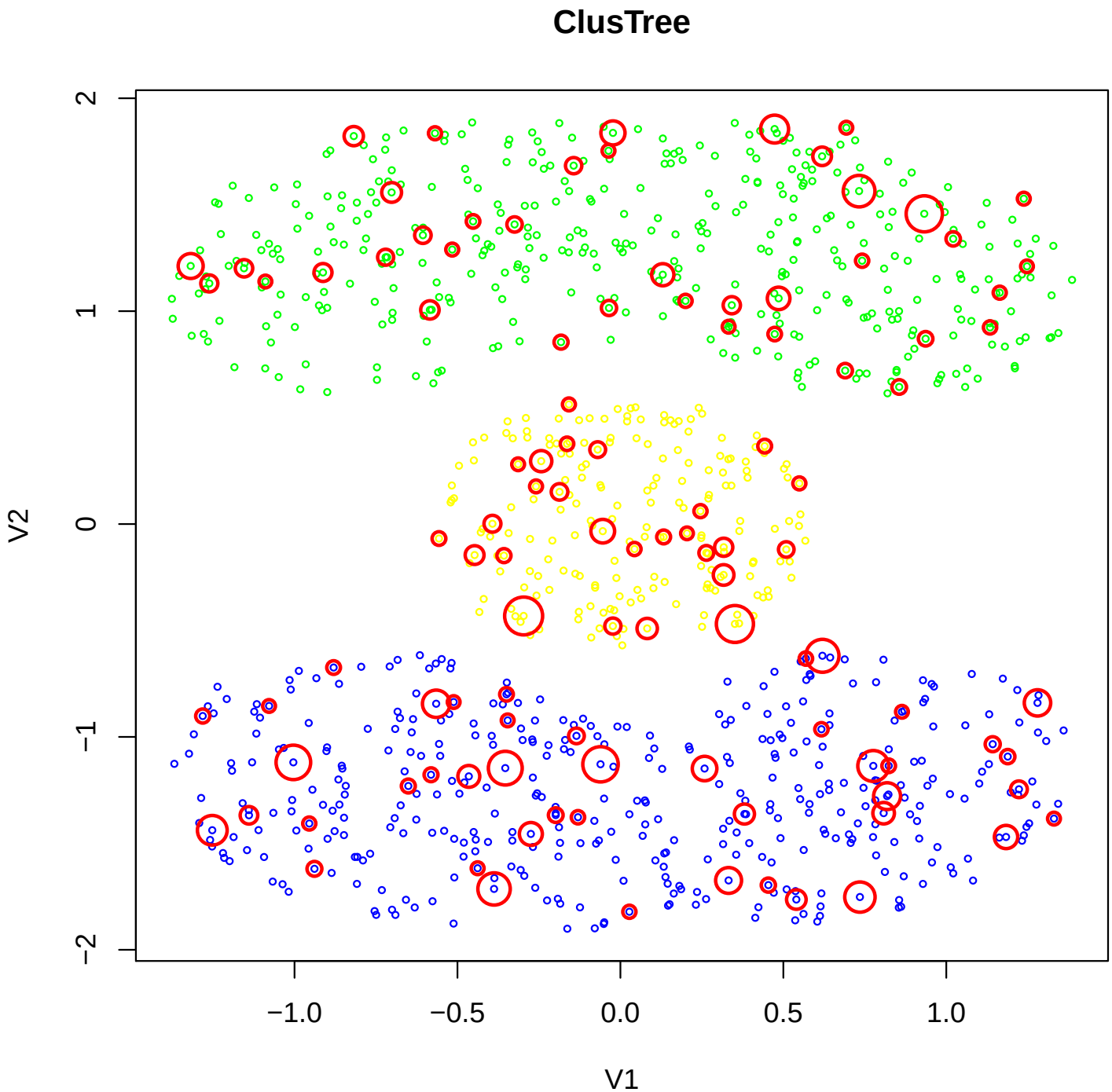
k-means: 0.737


k−Means
V2
V1
2
1
0
−1
−2
−1.0
−0.5
0.0
0.5
1.0


Sample
V2
2
1
0
−1
−2
−1.0
−0.5
0.0
0.5
1.0
V1


Birch
2
1
0
−1
−2
V2
−1.0
−0.5
0.0
0.5
1.0
V1


CluStream
V2
V1
2
1
0
−1
−2
−1.0
−0.5
0.0
0.5
1.0


ClusTree
V2
2
1
0
−1
−2
−1.0
−0.5
0.0
0.5
1.0
V1

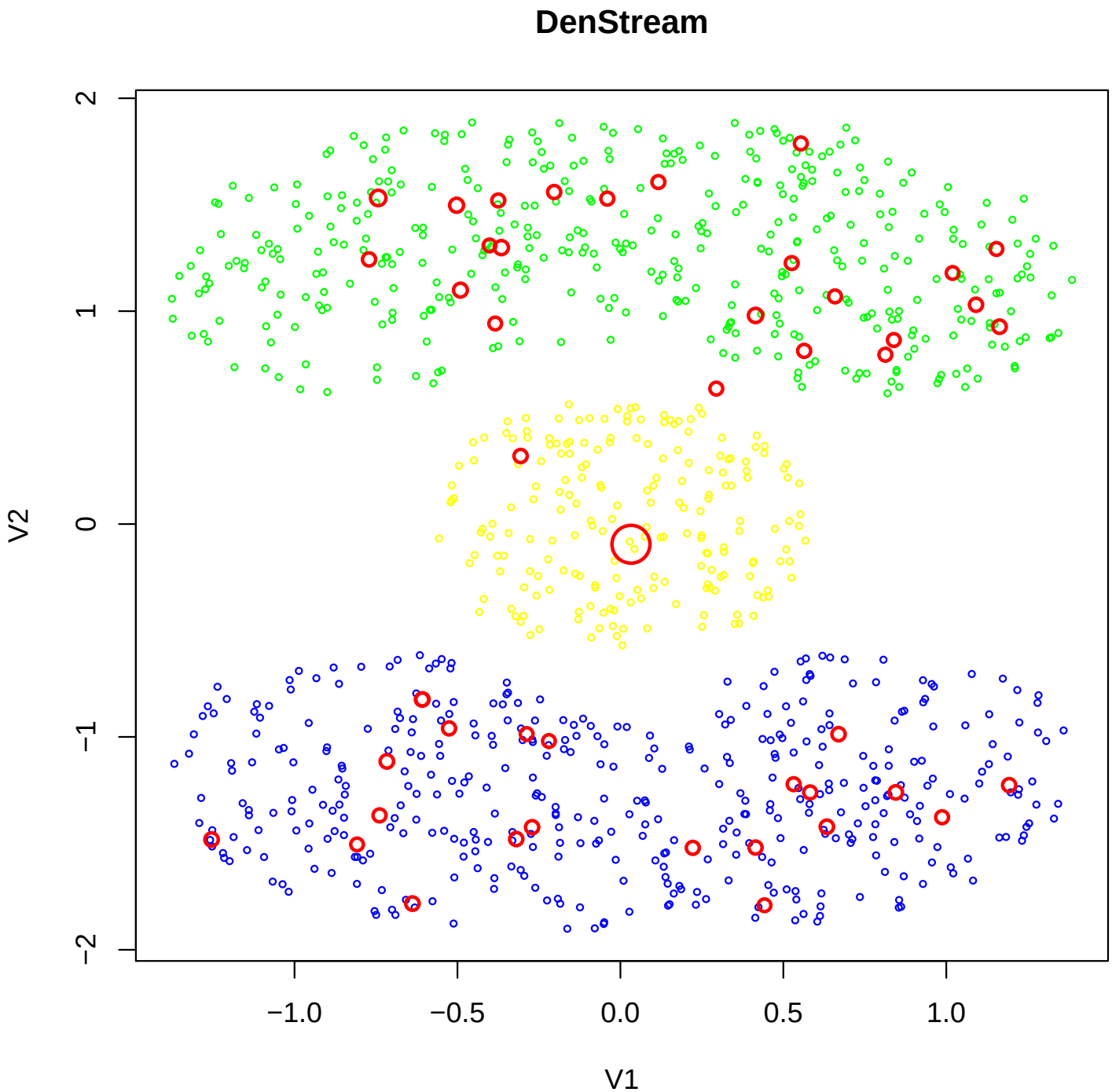

DenStream
V2
V1
2
1
0
−1
−2
−1.0
−0.5
0.0
0.5
1.0

Table of Contents

1. Motivation
2. Conventional Clustering
3. Data Stream Clustering
4. R-extension package **stream**
5. Empirical Study
6. Conclusion

Conclusion

- The R-extension package **stream** provides a framework to experiment with new algorithmic ideas and perform systematic experiments.
- Compared to k-means data stream clustering methods
  1. provide a preprocessing step suitable for large data,
  2. the quality of the clustering is comparable, and
  3. noise can be handled.

Future work

- Evaluation on large real data sets.
- Develop data stream clustering without reclustering.
- Extend stream for data stream classification and mining frequent itemsets in streams.

Thank you!

Visit us at

`http://lyle.smu.edu/IDA`

References I

C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu.

A framework for clustering evolving data streams.

In *Proceedings of the International Conference on Very Large Data Bases (VLDB '03)*, pages 81–92, 2003.

C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu.

A framework for projected clustering of high dimensional data streams.

In *Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB '04)*, pages 852–863, 2004.

A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl.

MOA: Massive online analysis, a framework for stream classification and clustering.

Framework, 2010.

F. Cao, M. Ester, W. Qian, and A. Zhou.

Density-based clustering over an evolving data stream with noise.

In *Proceedings of the 2006 SIAM International Conference on Data Mining*, pages 328–339. SIAM, 2006.

T. G. Dietterich.

Mach. Learn., 30(2-3), 1998.

M. Ester, H. peter Kriegel, J. S, and X. Xu.

A density-based algorithm for discovering clusters in large spatial databases with noise.

pages 226–231. AAAI Press, 1996.

S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan.

Clustering data streams: Theory and practice.

IEEE Transactions on Knowledge and Data Engineering, 15(3):515–528, 2003.

References II

A. K. Jain, M. N. Murty, and P. J. Flynn.

Data clustering: a review.
ACM Comput. Surv., 31(3):264–323, Sept. 1999.

L. Kaufman and P. J. Rousseeuw.

Finding groups in data: an introduction to cluster analysis.
John Wiley and Sons, New York, 1990.

P. Kranen, I. Assent, C. Baldauf, and T. Seidl.

The clustree: indexing micro-clusters for anytime stream mining.
Knowledge and Information Systems, 29(2):249–272, 2011.

A. McCallum, K. Nigam, and L. H. Ungar.

Efficient clustering of high-dimensional data sets with application to reference matching.
In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pages 169–178, New York, NY, USA, 2000. ACM.

D. Tasoulis, N. Adams, and D. Hand.

Unsupervised clustering in streaming data.
In *IEEE International Workshop on Mining Evolving and Streaming Data. Sixth IEEE International Conference on Data Mining (ICDM 2006)*, pages 638–642, Dec. 2006.

D. K. Tasoulis, G. Ross, and N. M. Adams.

Visualising the cluster structure of data streams.
In *Advances in Intelligent Data Analysis VII*, Lecture Notes in Computer Science, pages 81–92. Springer, 2007.

L. Tu and Y. Chen.

Stream data clustering based on grid density and attraction.
ACM Transactions on Knowledge Discovery from Data, 3(3):1–27, 2009.

References III

K. Udommanetanakit, T. Rakthanmanon, and K. Waiyamai.

E-stream: Evolution-based technique for stream clustering.
In *ADMA '07: Proceedings of the 3rd international conference on Advanced Data Mining and Applications*, pages 605–615. Springer-Verlag, Berlin, Heidelberg, 2007.

L. Wan, W. K. Ng, X. H. Dang, P. S. Yu, and K. Zhang.

Density-based clustering of data streams at multiple resolutions.
ACM Transactions on Knowledge Discovery from Data, 3(3):1–28, 2009.

T. Zhang, R. Ramakrishnan, and M. Livny.

BIRCH: An efficient data clustering method for very large databases.
In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 103–114. ACM, 1996.