INSTITUT
TUTTE
INSTITUTE

Clustering Data

A Guide for the Perplexed

Leland McInnes
John Healy

INSTITUT TUTTE INSTITUTE
Tutte Institute
for Mathematics and Computing

leland.mcinnes@gmail.com
jchealy@gmail.com

What is clustering?

Find groups of data that are all similar

What that means depends on your application

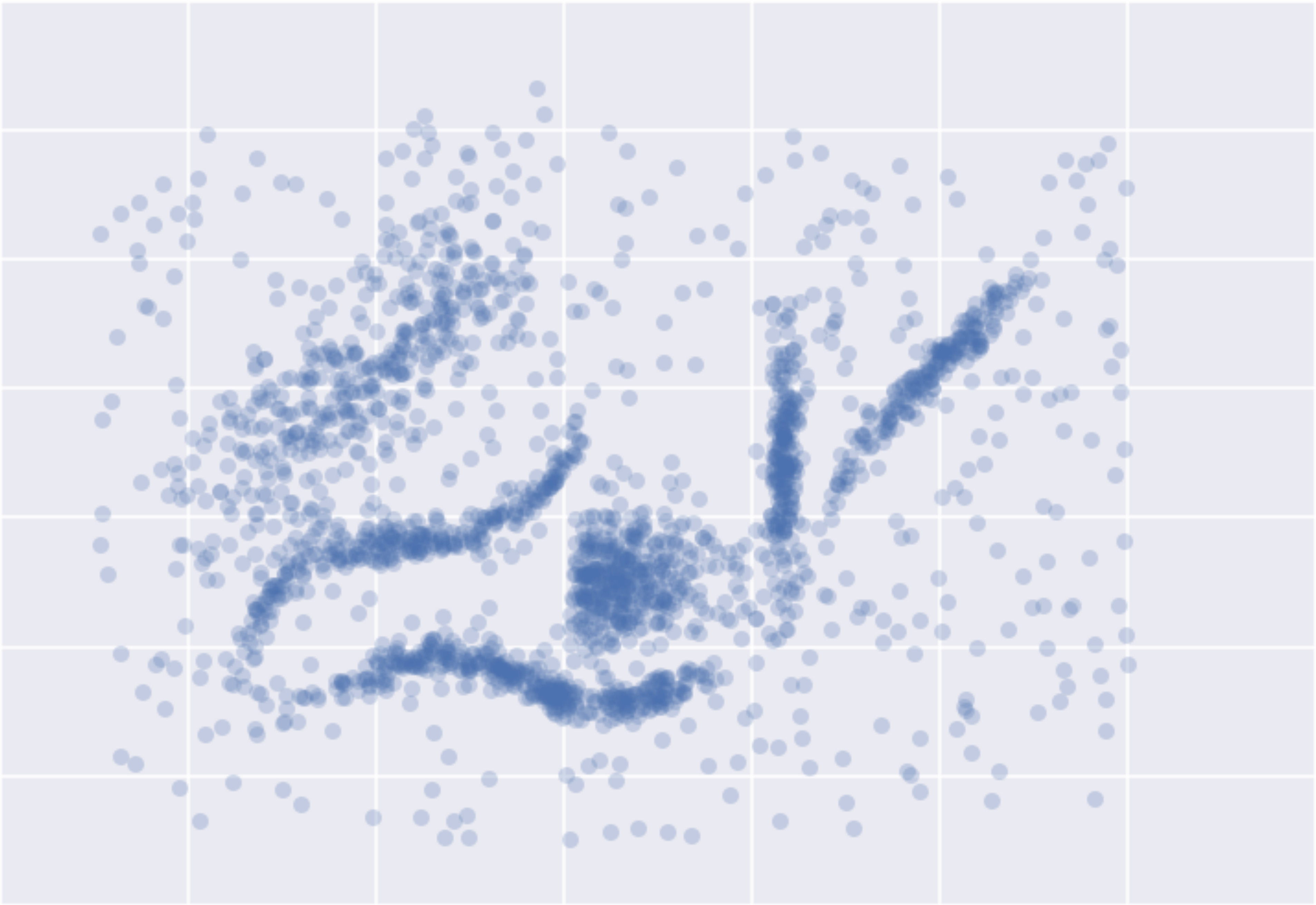
Goals

- Partition your data
- Summarize your data
- Explore your data
- Embed in a vector space
- Find patterns in your data
- …

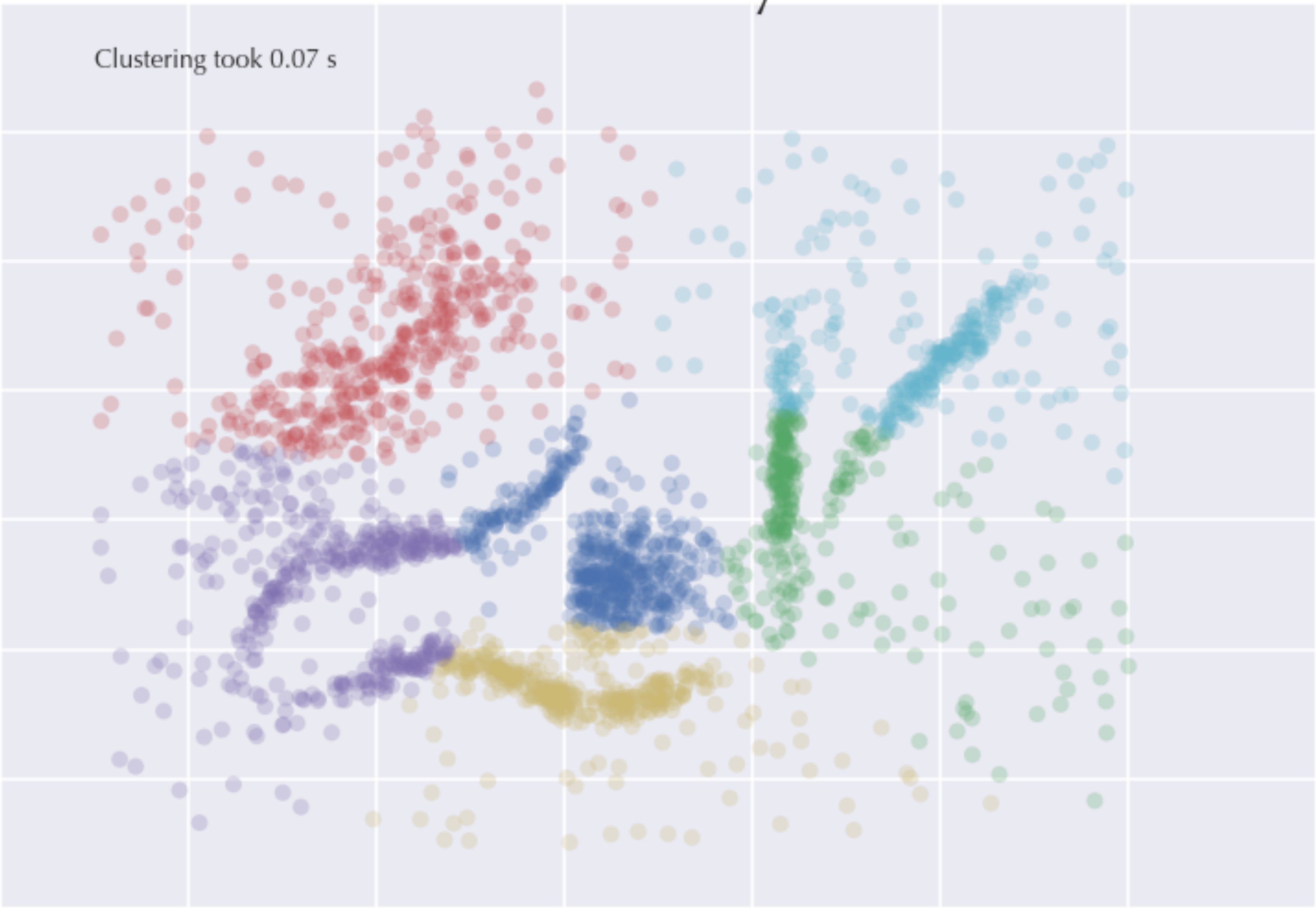
Find groups of data that are all similar

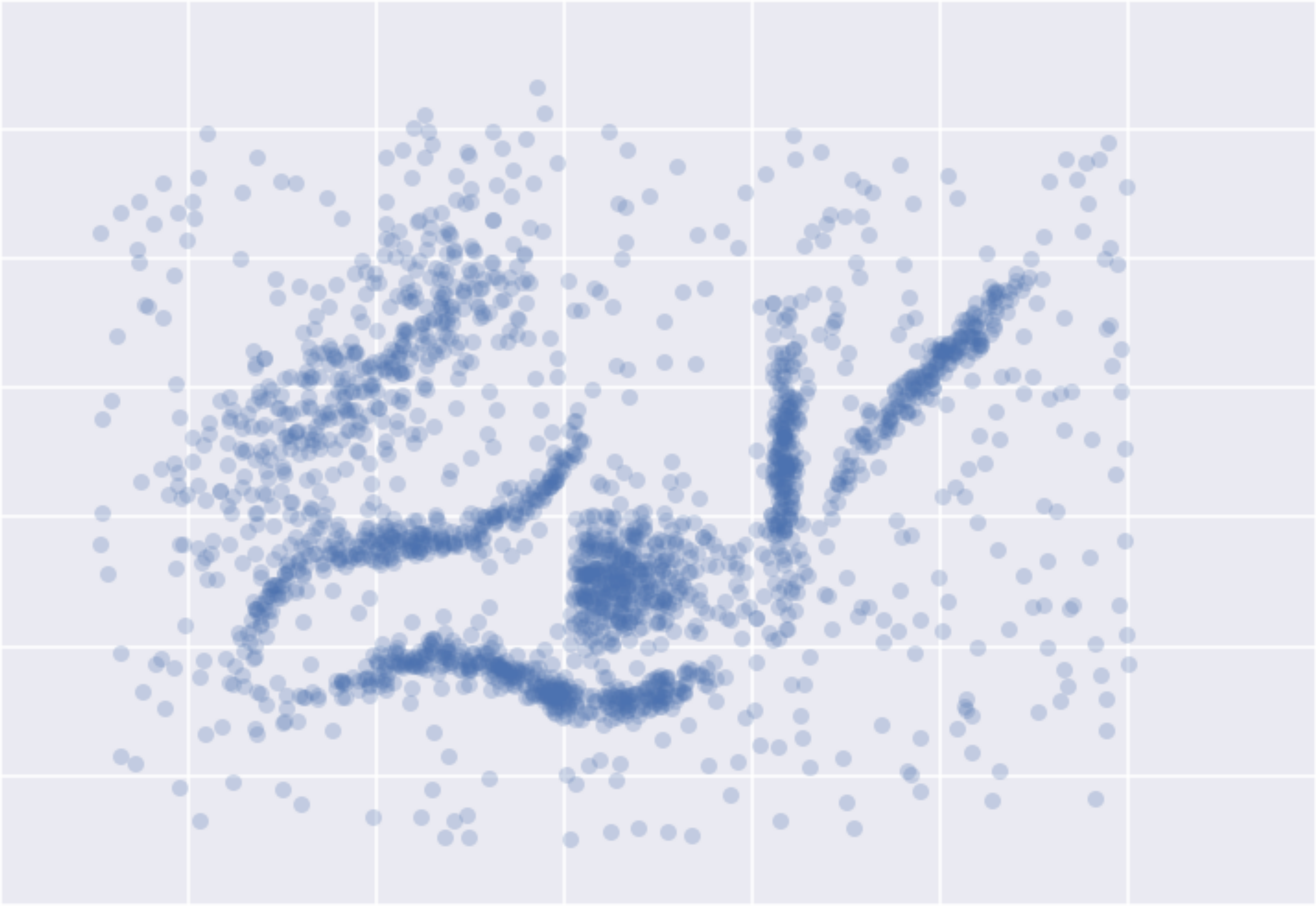
Easy in theory

Hard in practice

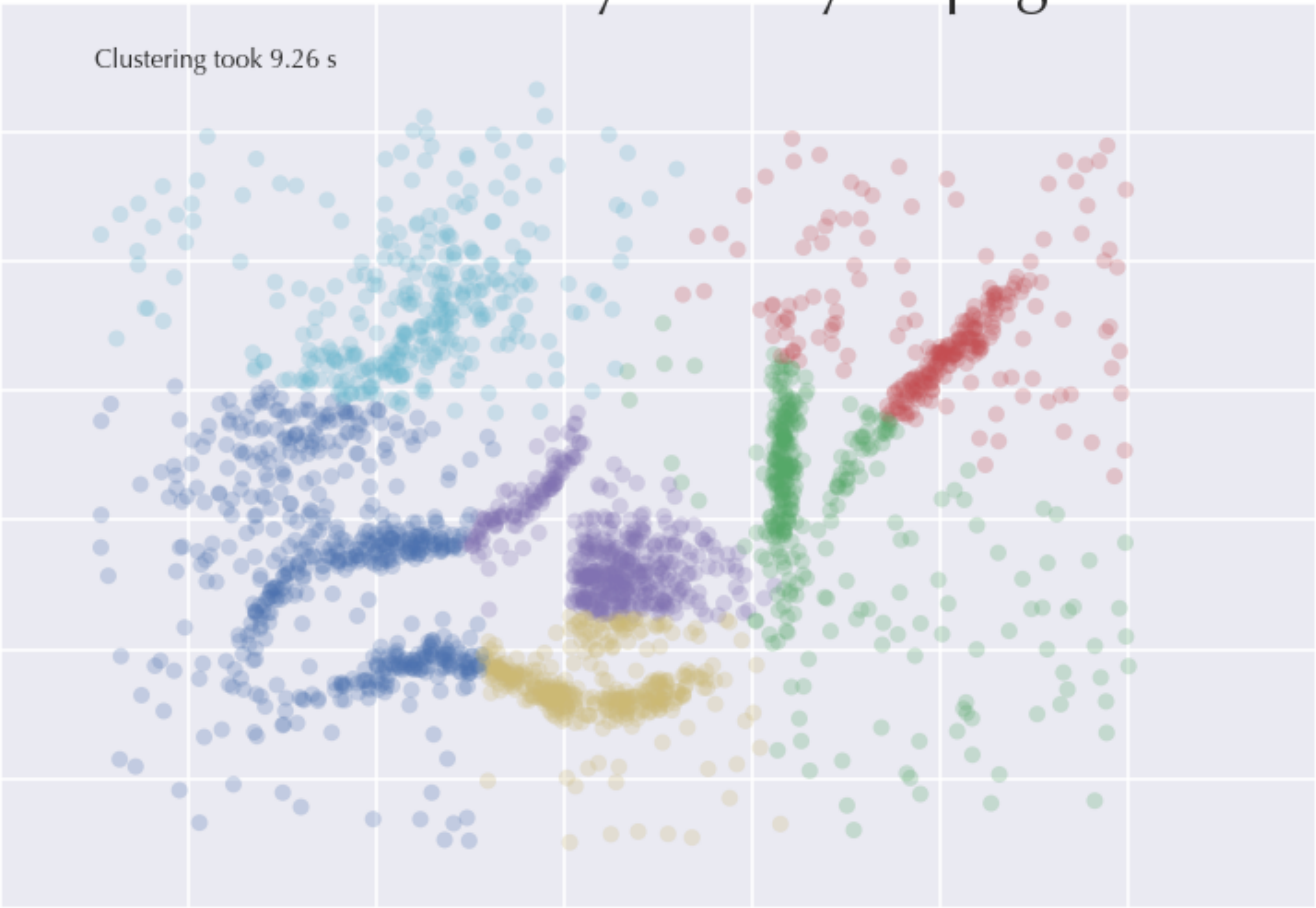


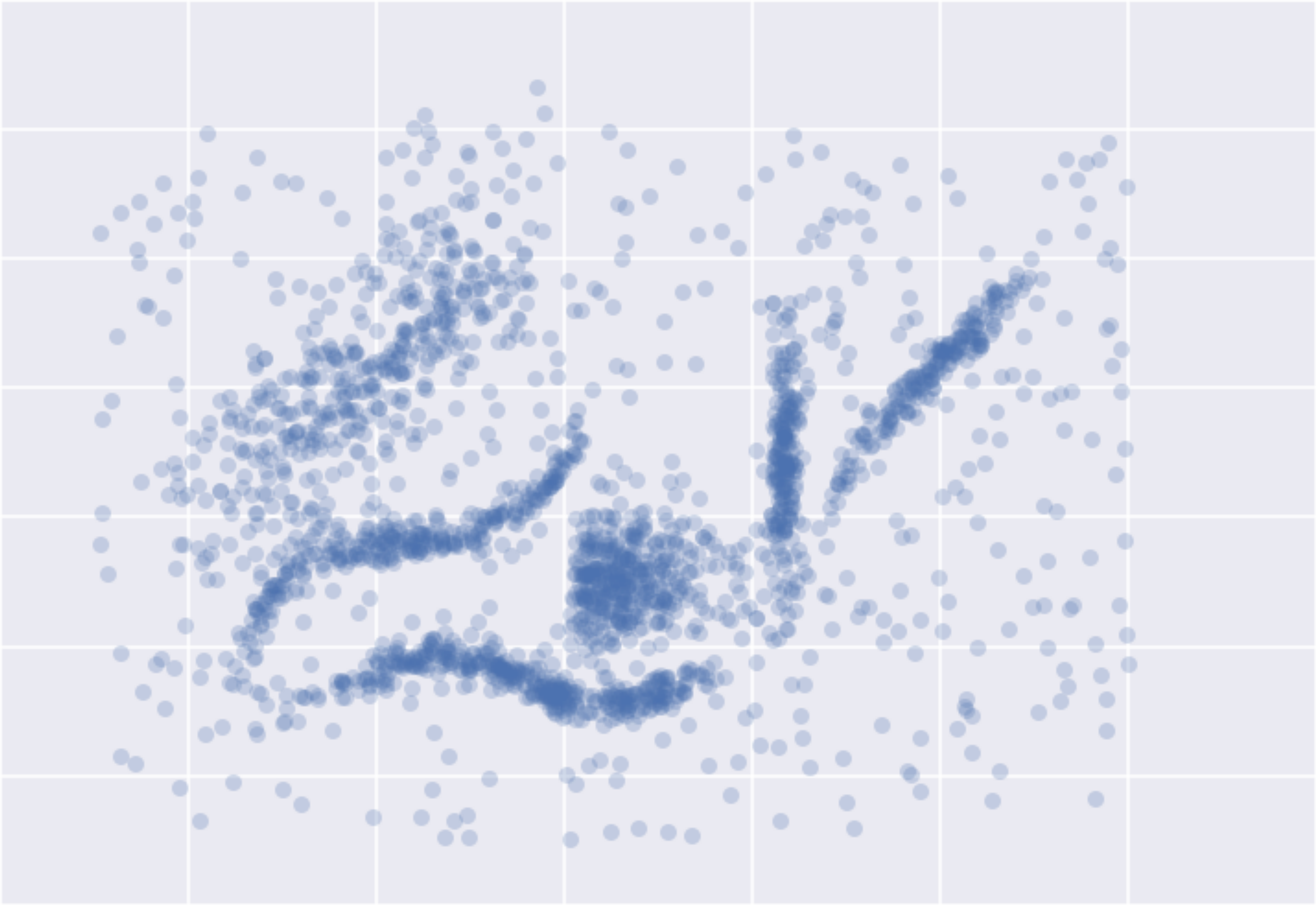
Clusters found by KMeans



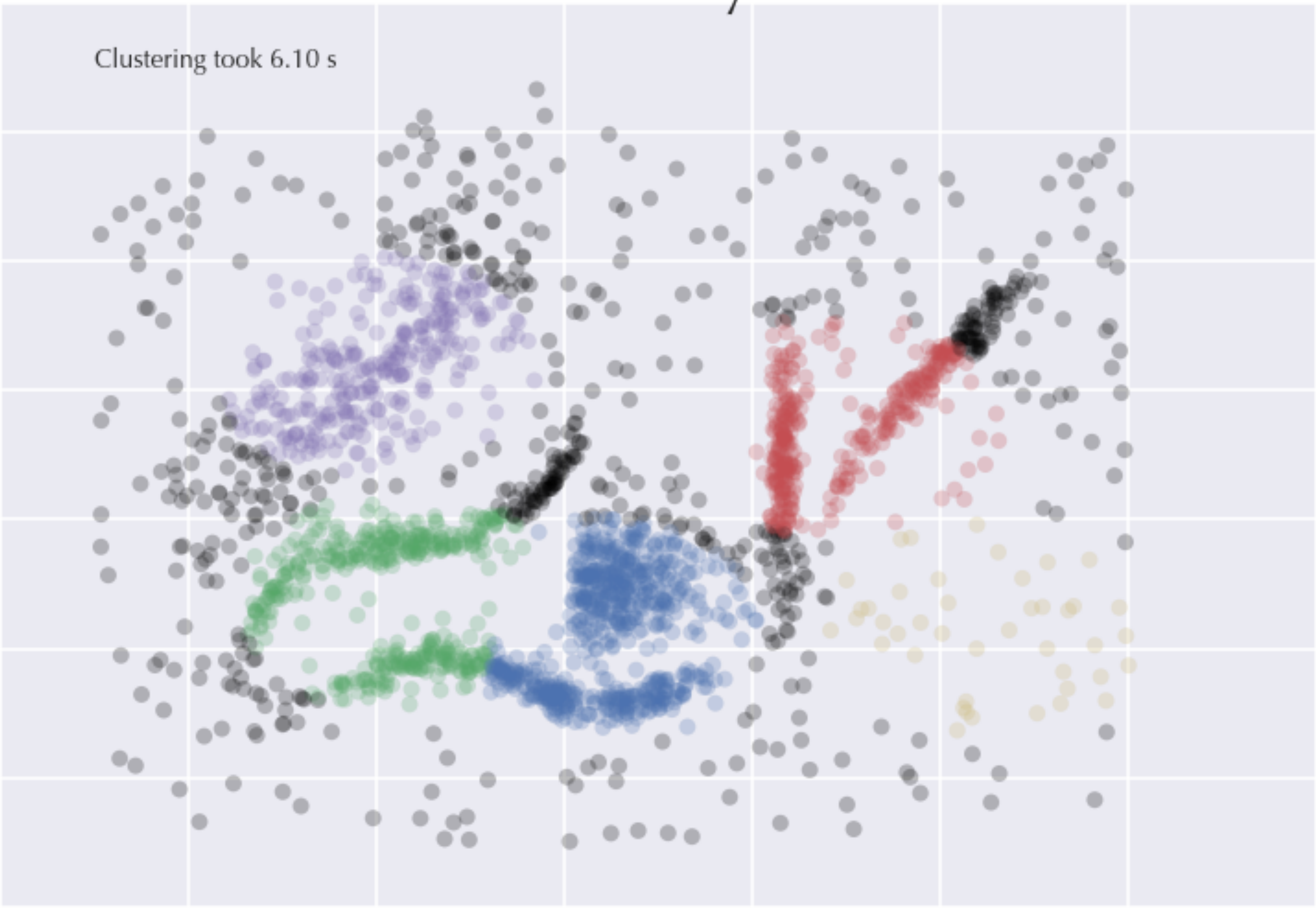


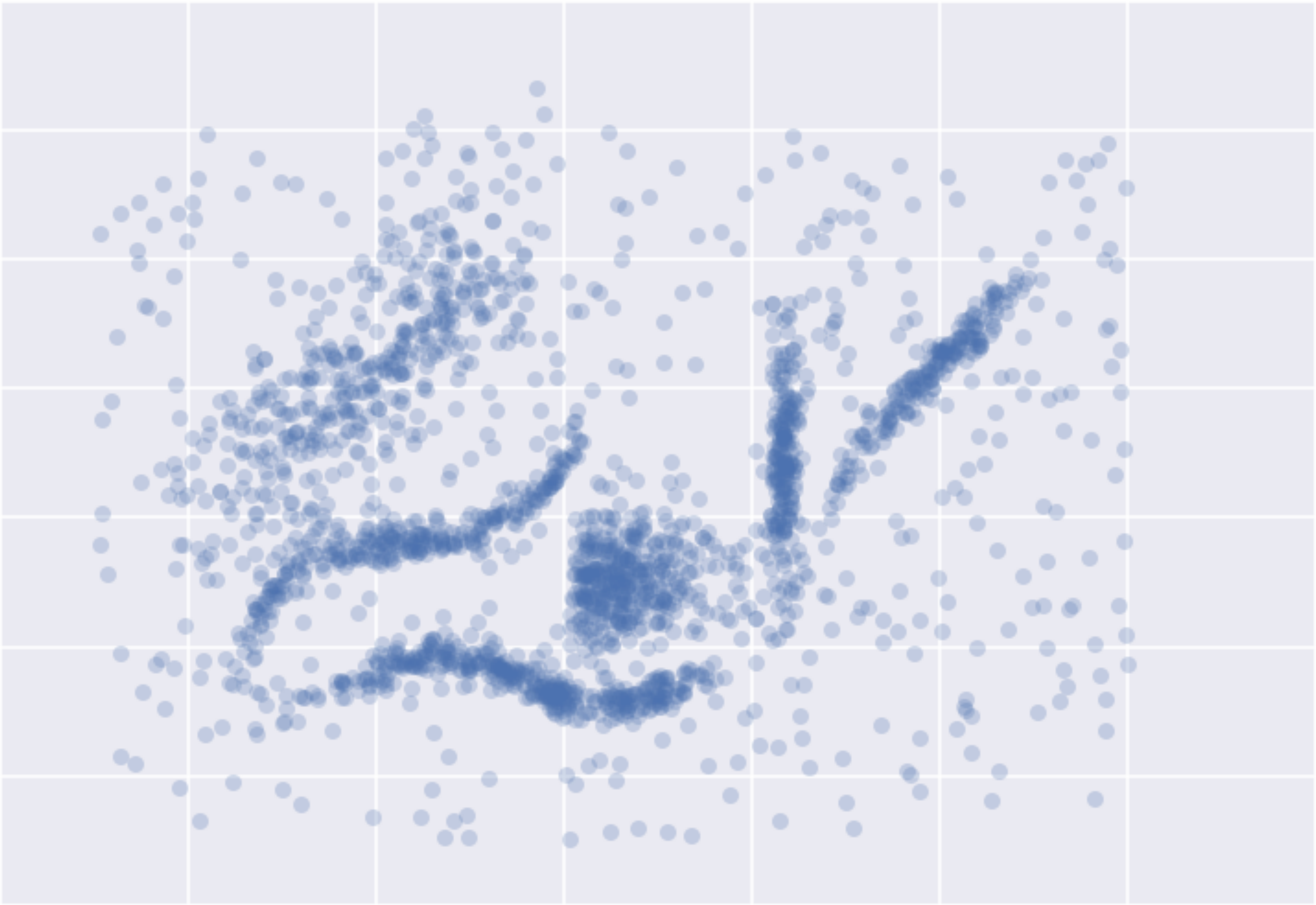
Clusters found by AffinityPropagation



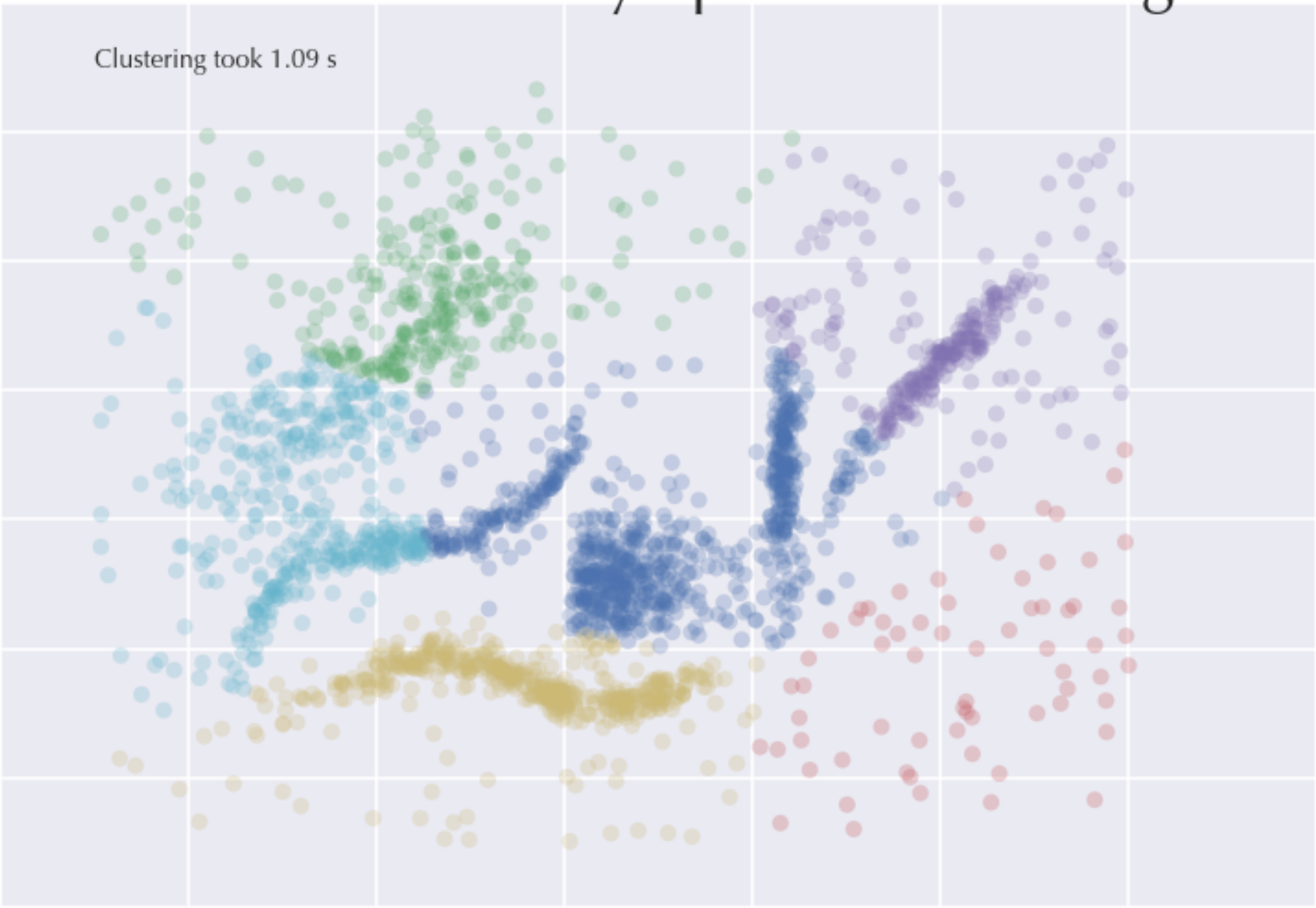


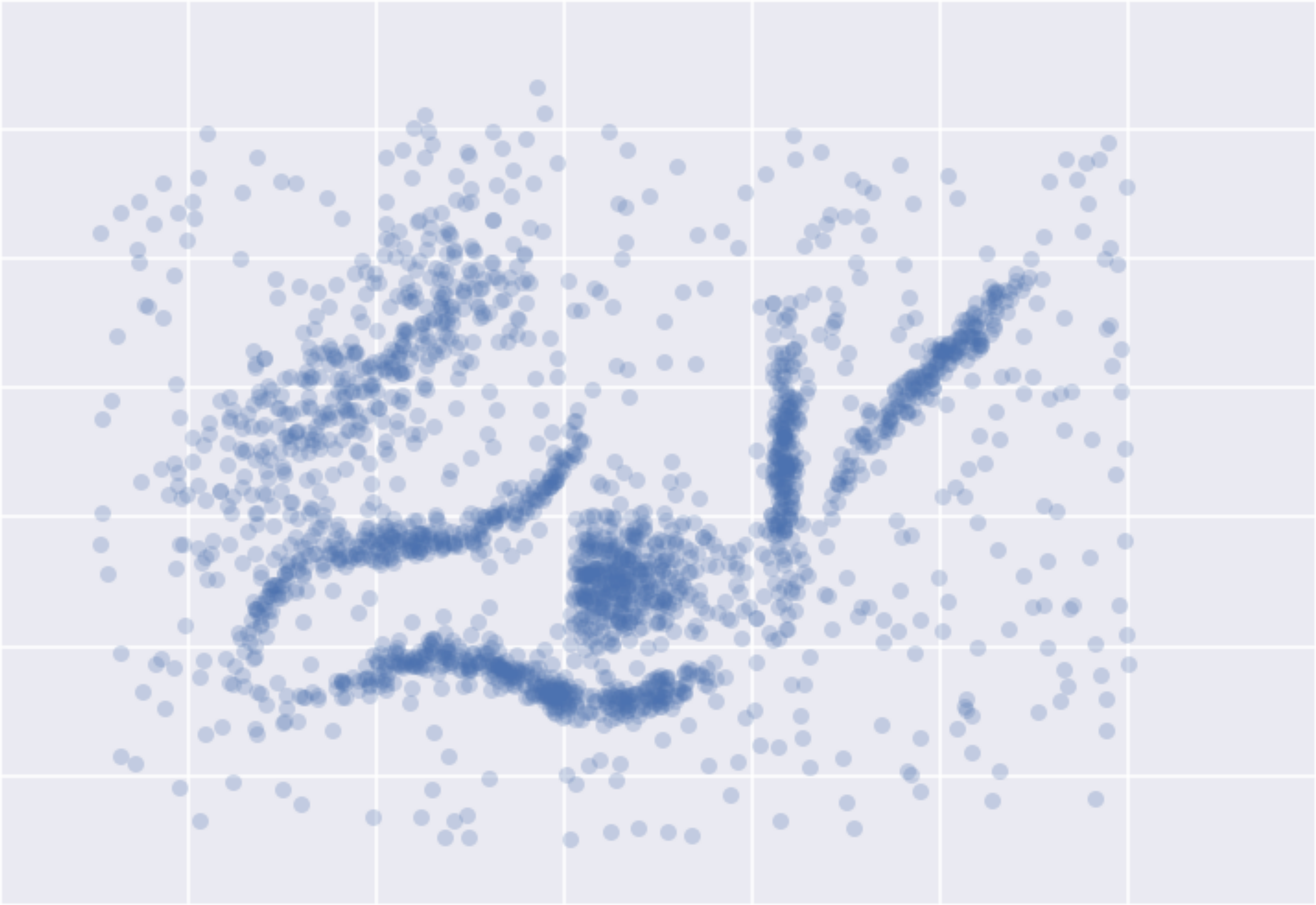
Clusters found by MeanShift



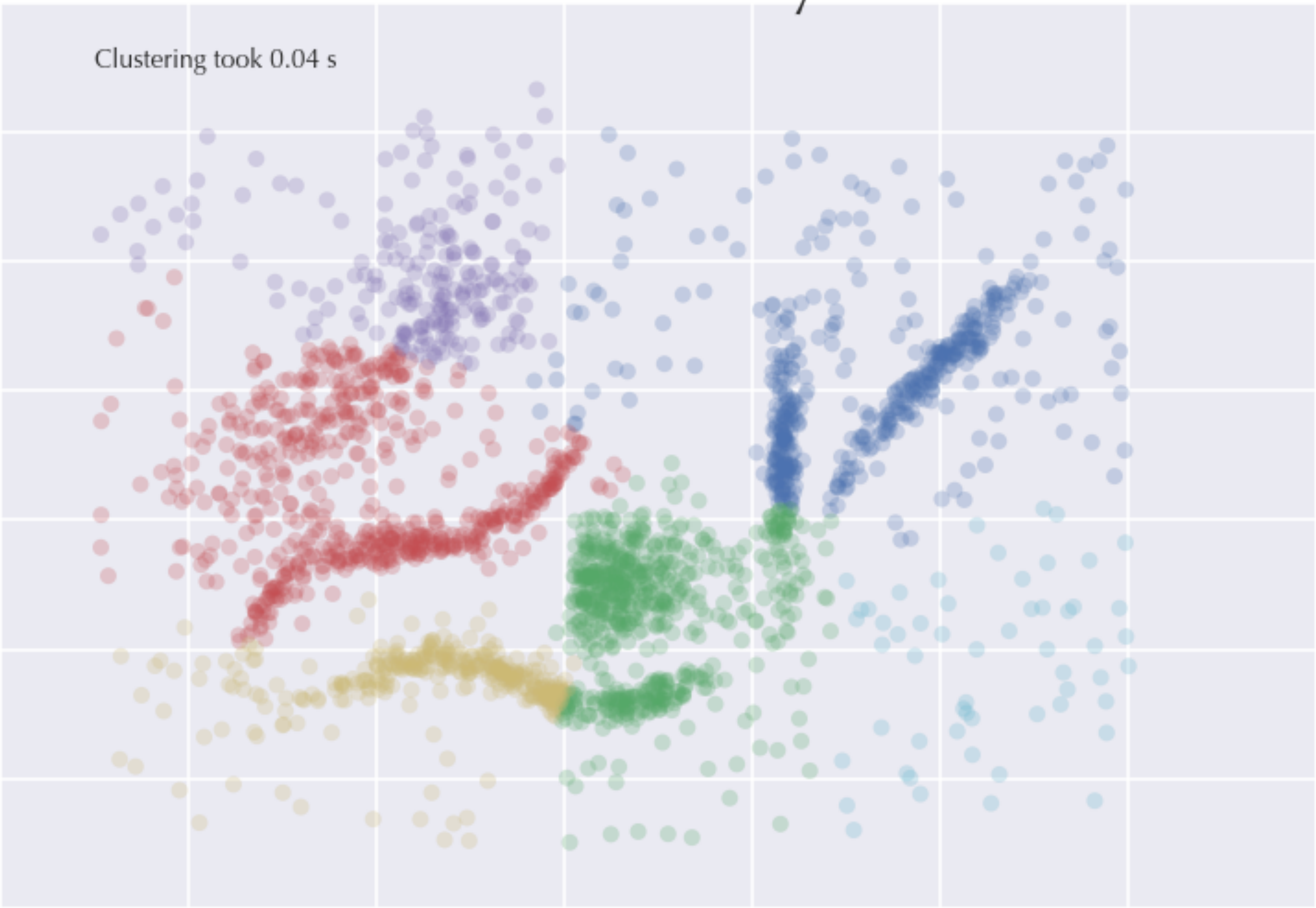


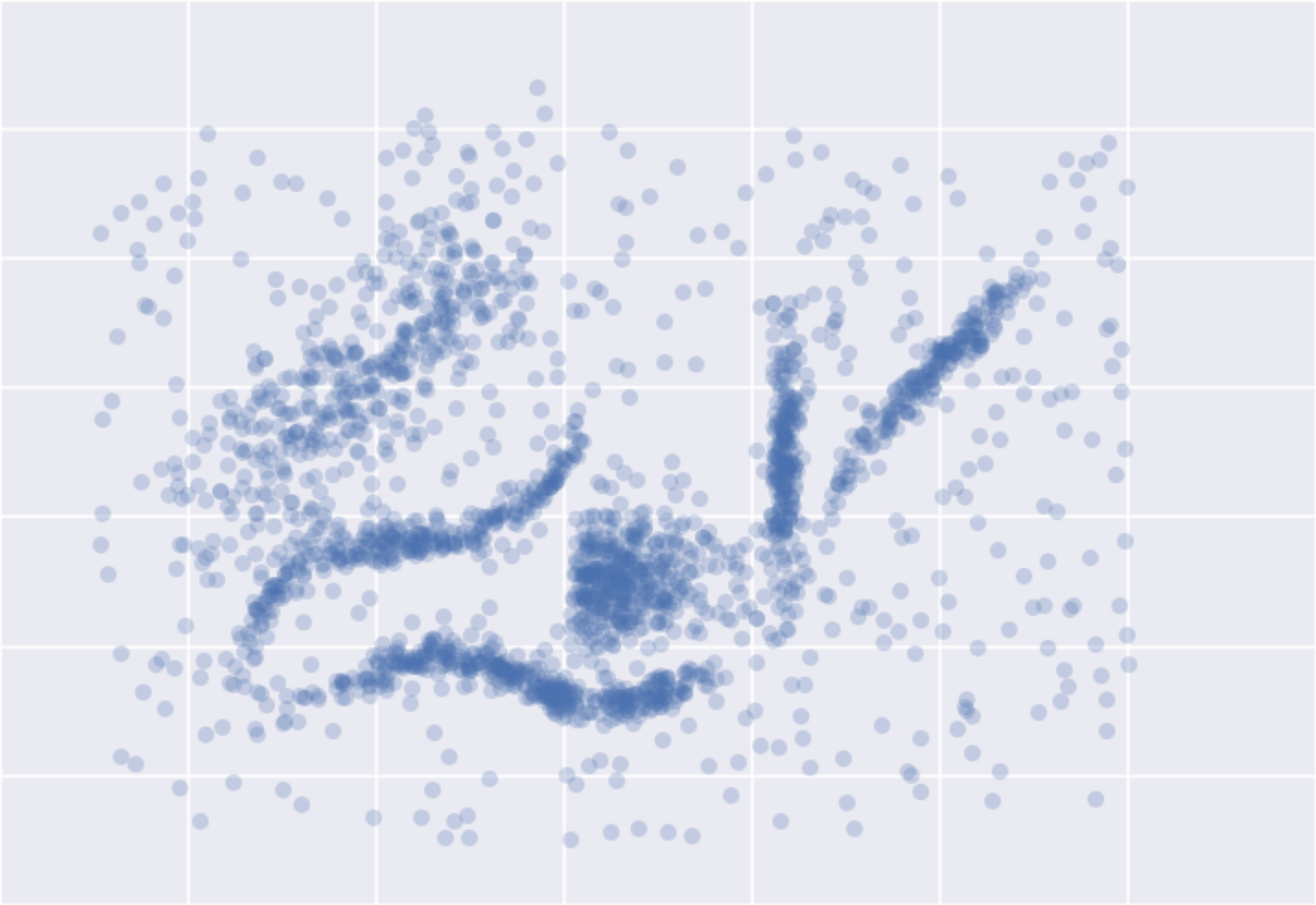
Clusters found by SpectralClustering



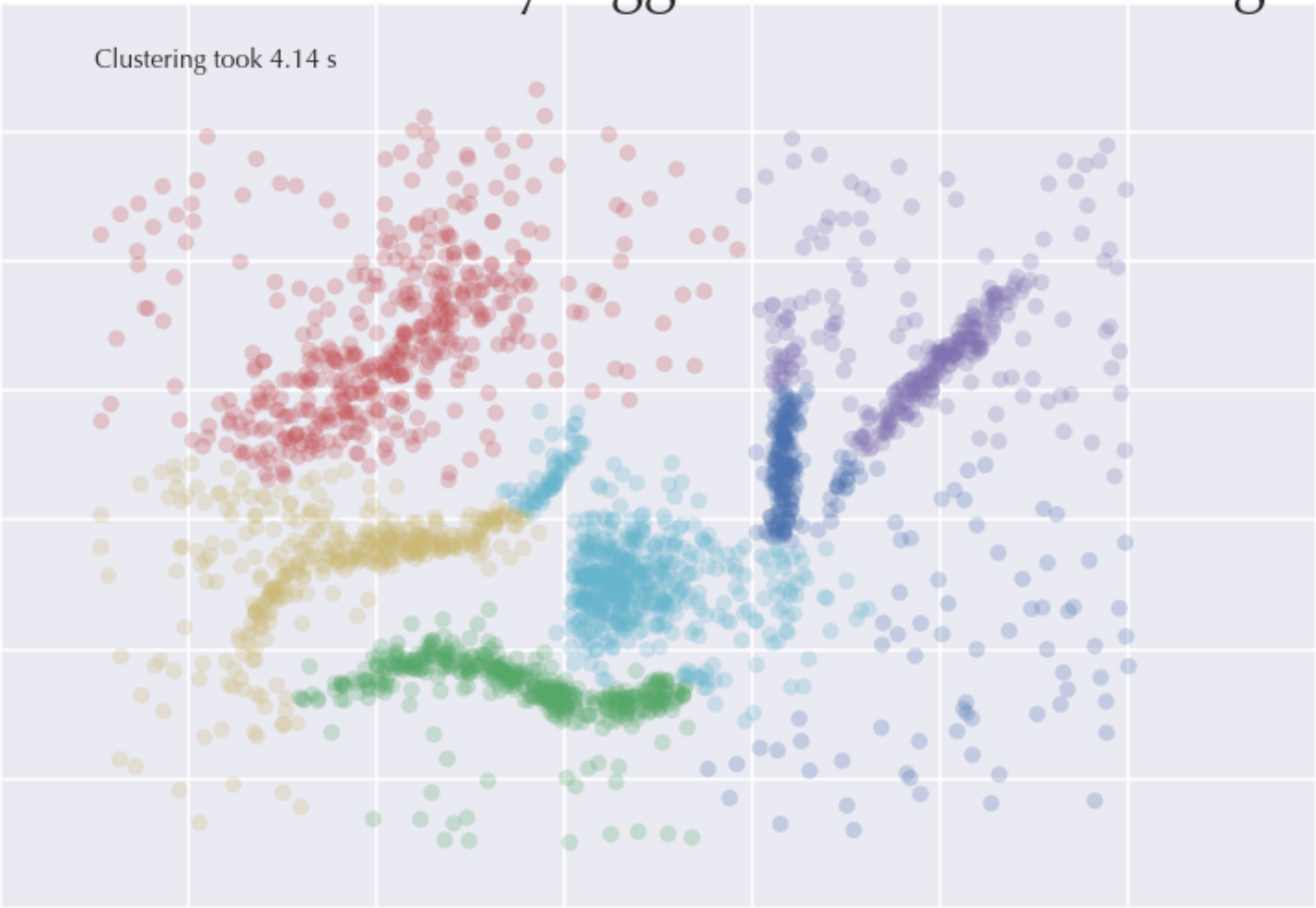


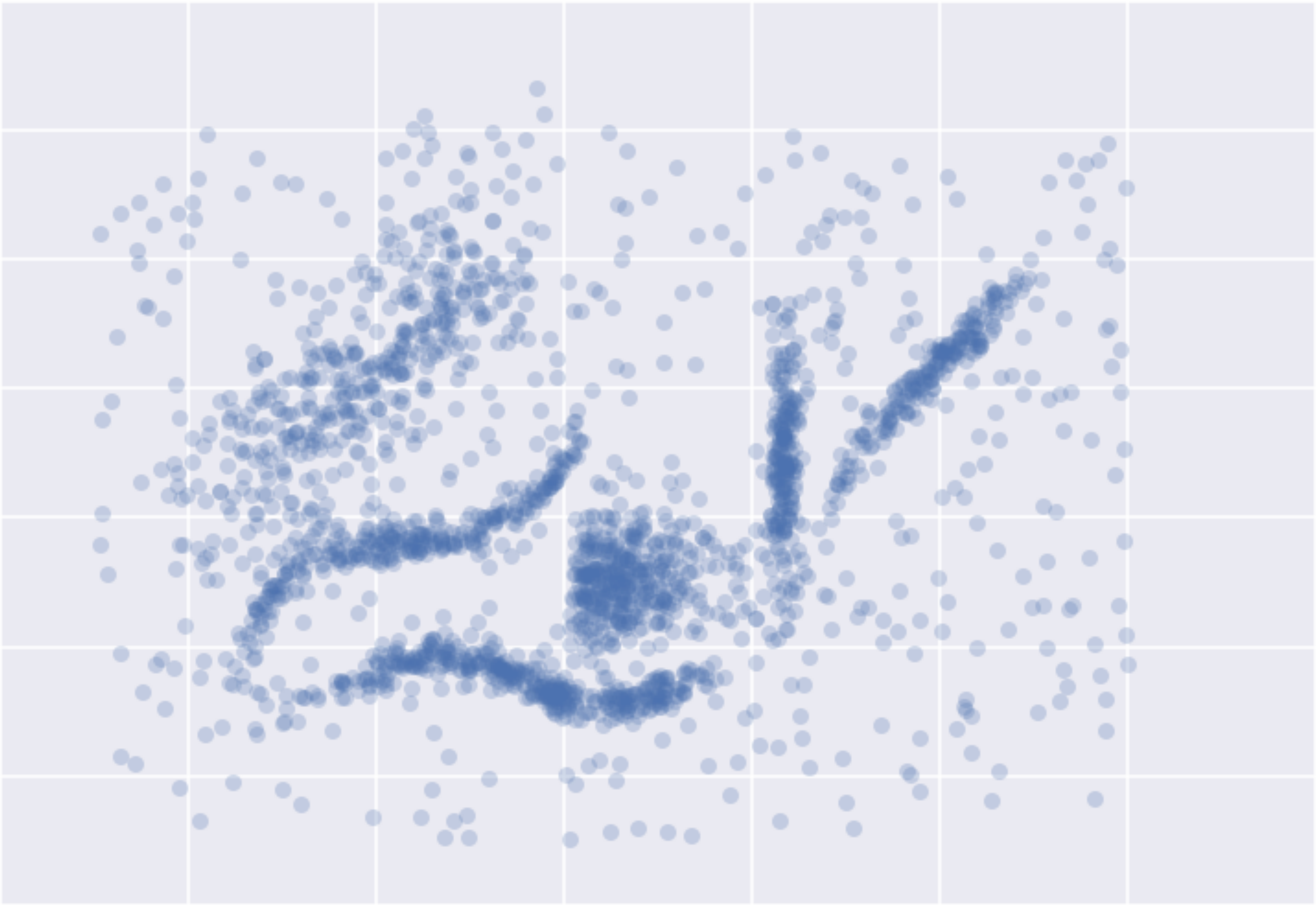
Clusters found by Birch



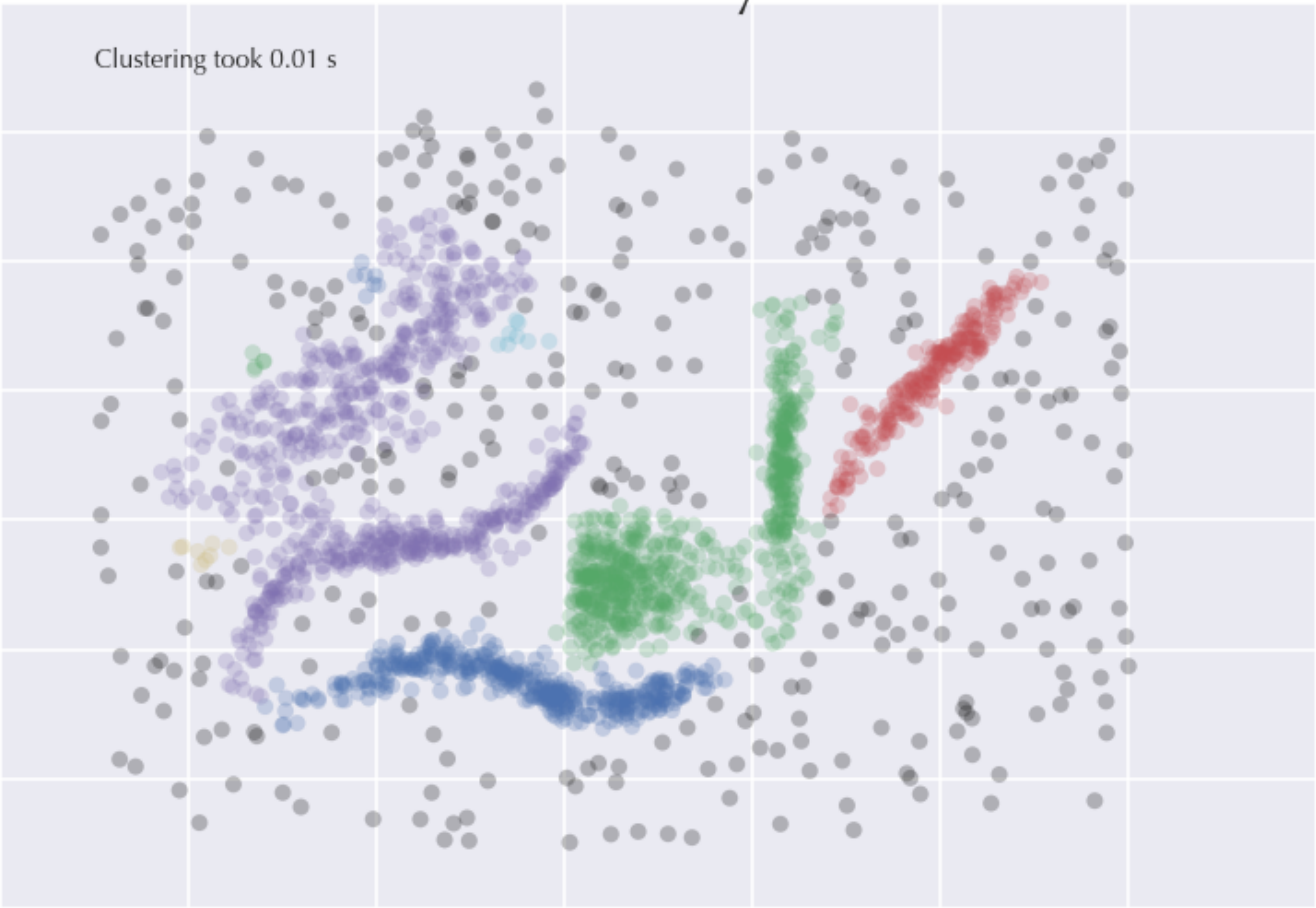


Clusters found by AgglomerativeClustering





Clusters found by DBSCAN

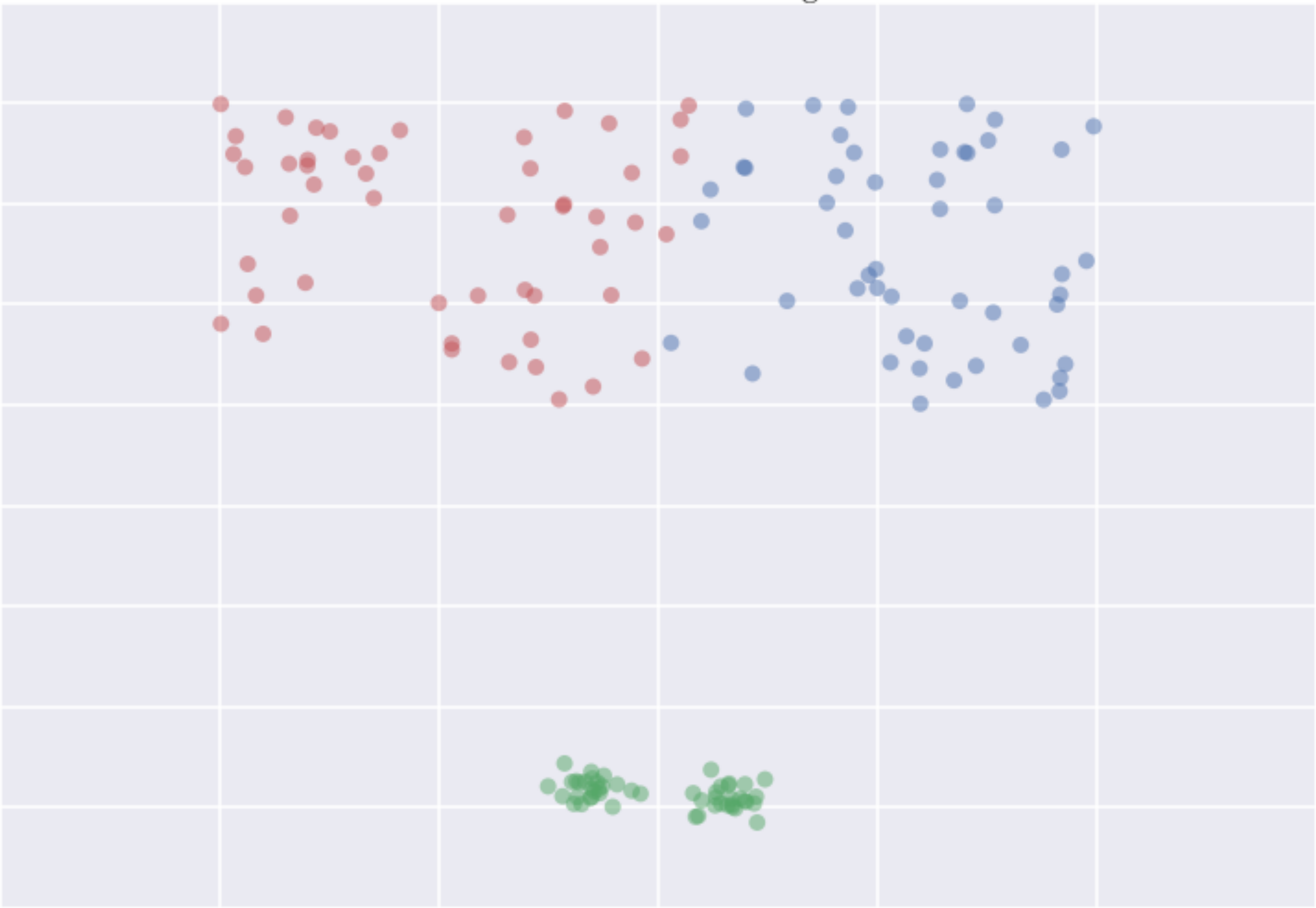


What makes it so hard?

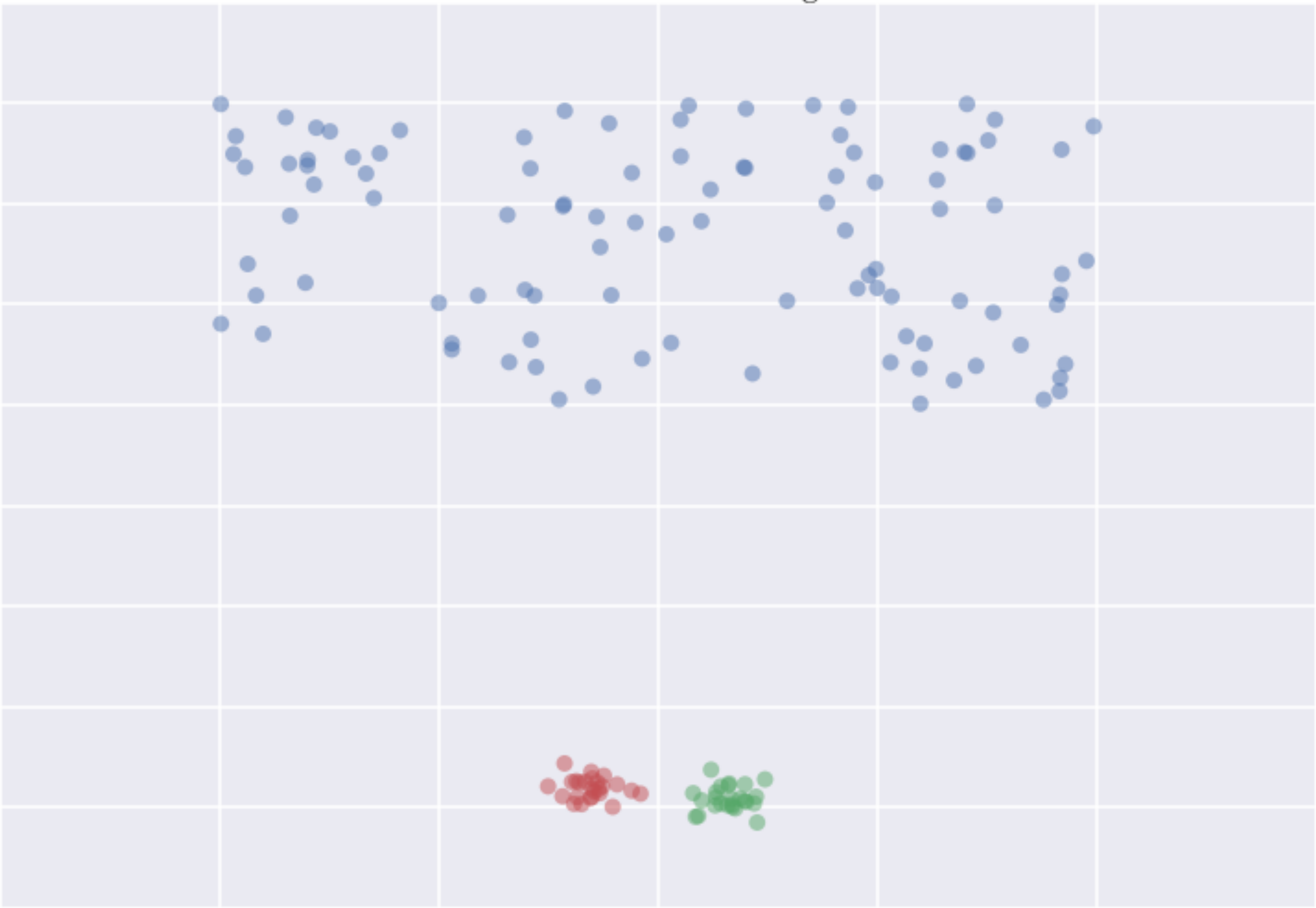
What makes a clustering good? How do you tell?

Which clustering is better?

A Possible Clustering



A Possible Clustering

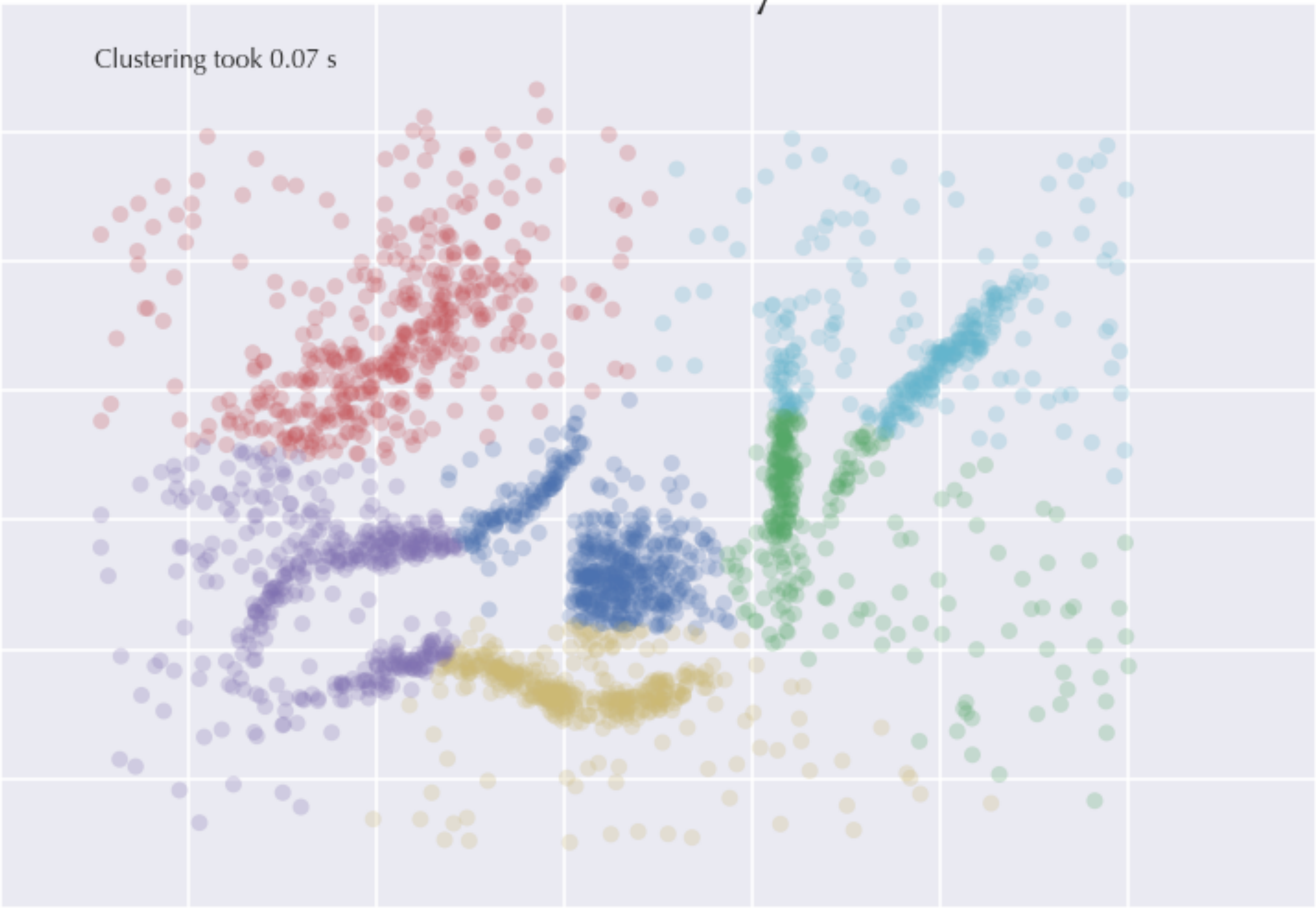


What is a cluster?

What do I mean by a cluster?

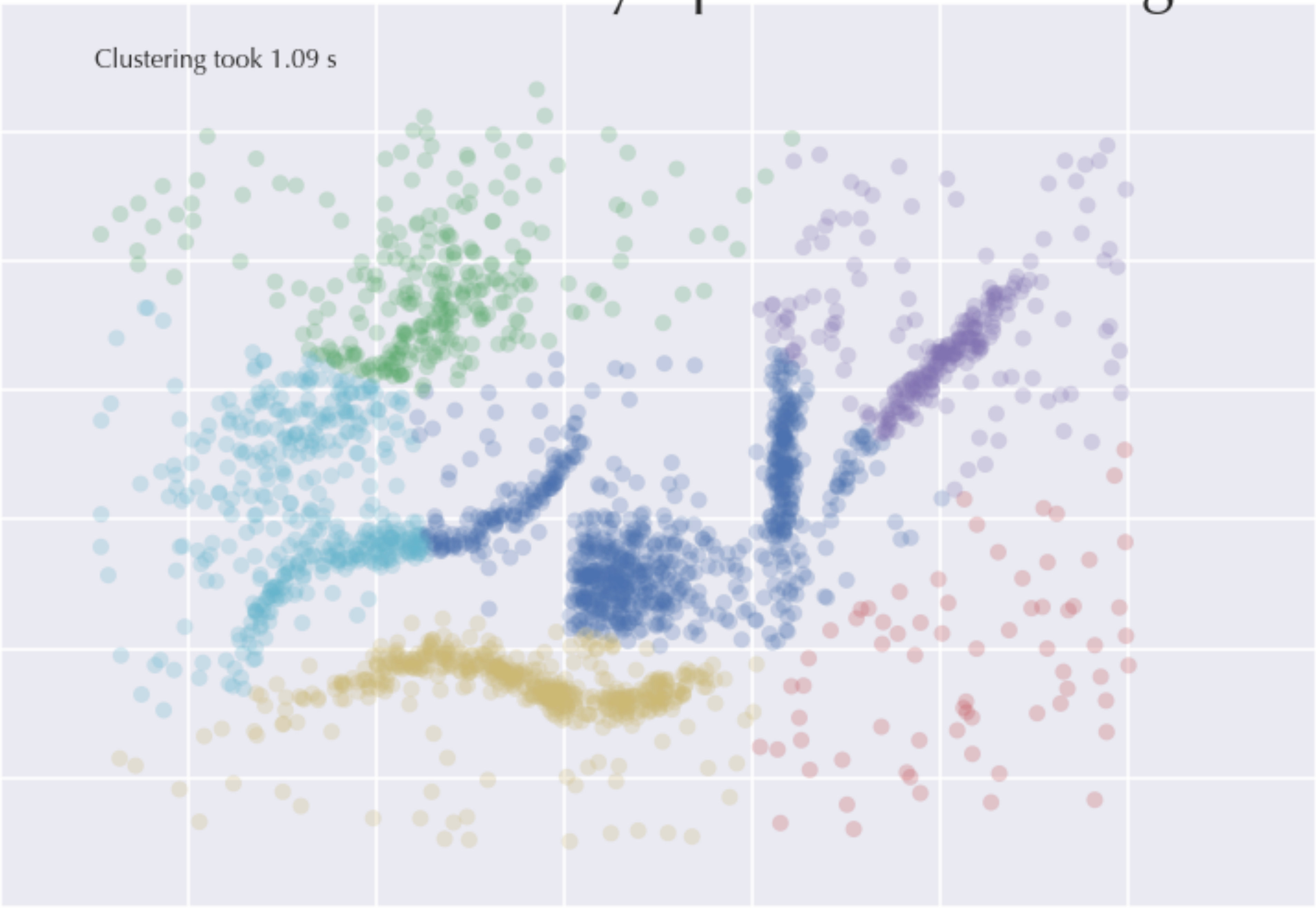
Let’s start by looking at what not to do

Clusters found by KMeans



Clusters need not be balls!

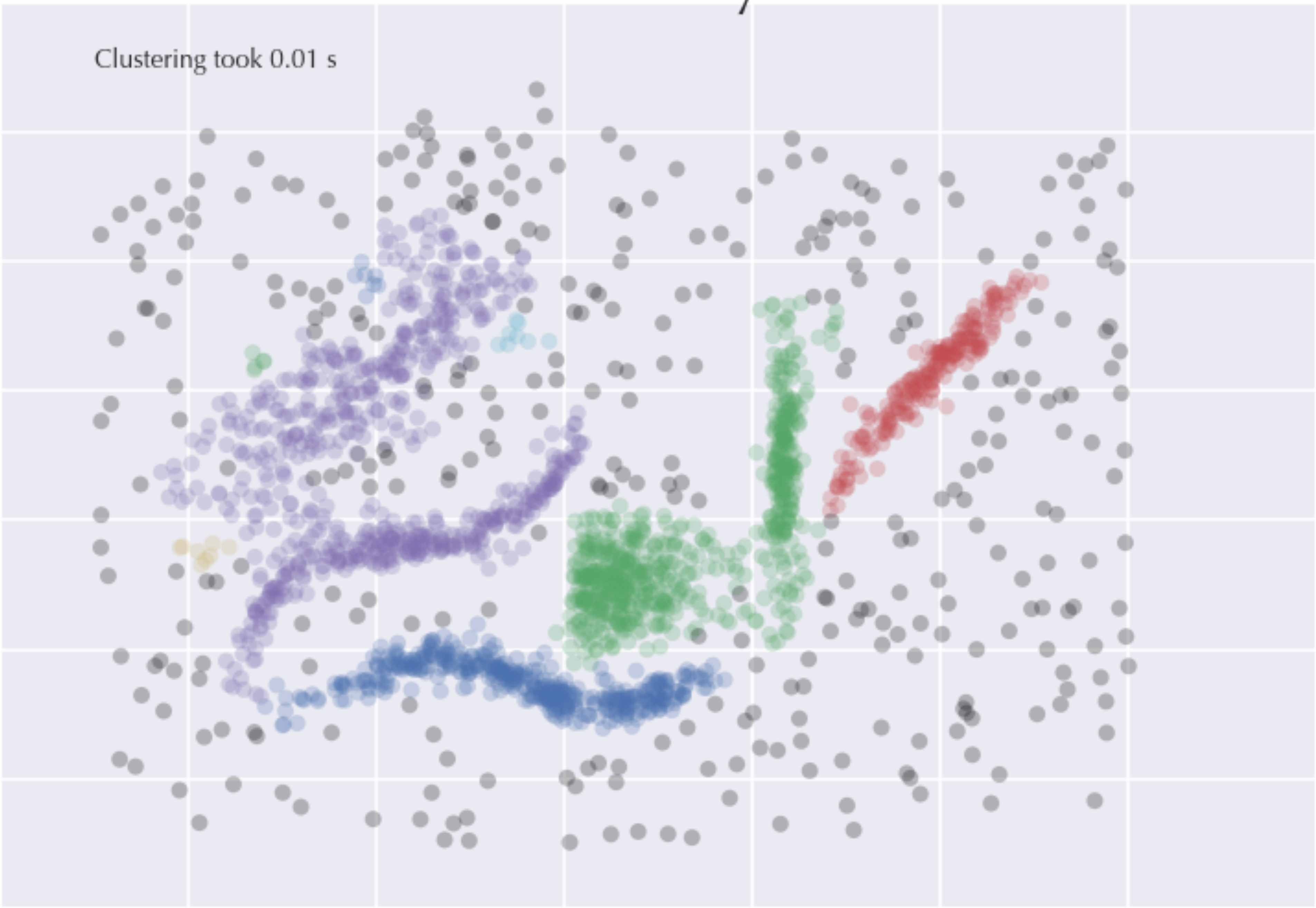
Clusters found by SpectralClustering



Not every point is in a cluster

Real data has noise!

Clusters found by DBSCAN



Clusters are dense areas?

Separated by less dense areas?

That’s what I mean when I talk about clustering

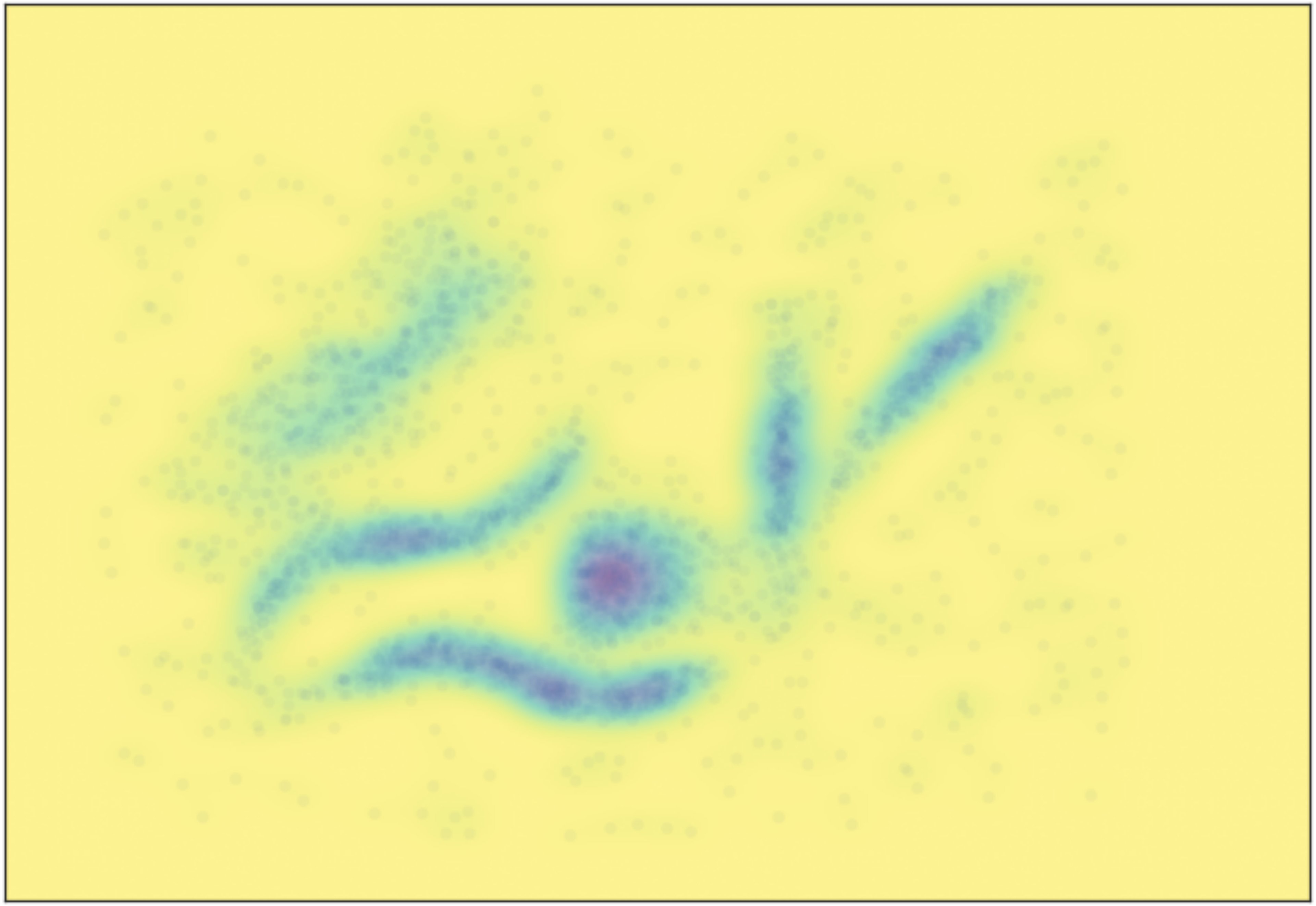
You should think hard about what you mean before doing any clustering

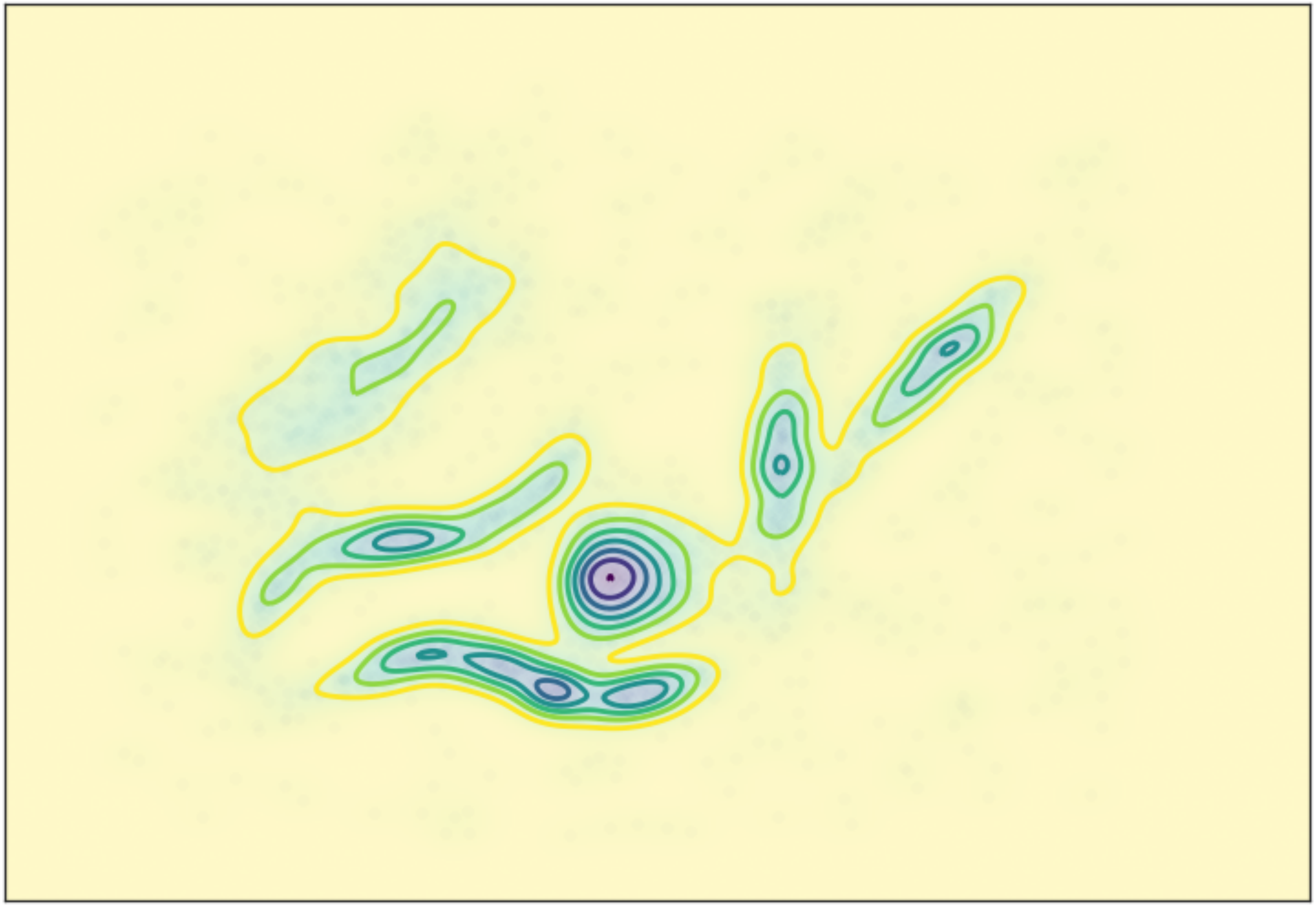
Can we be more specific about density-based clusters?

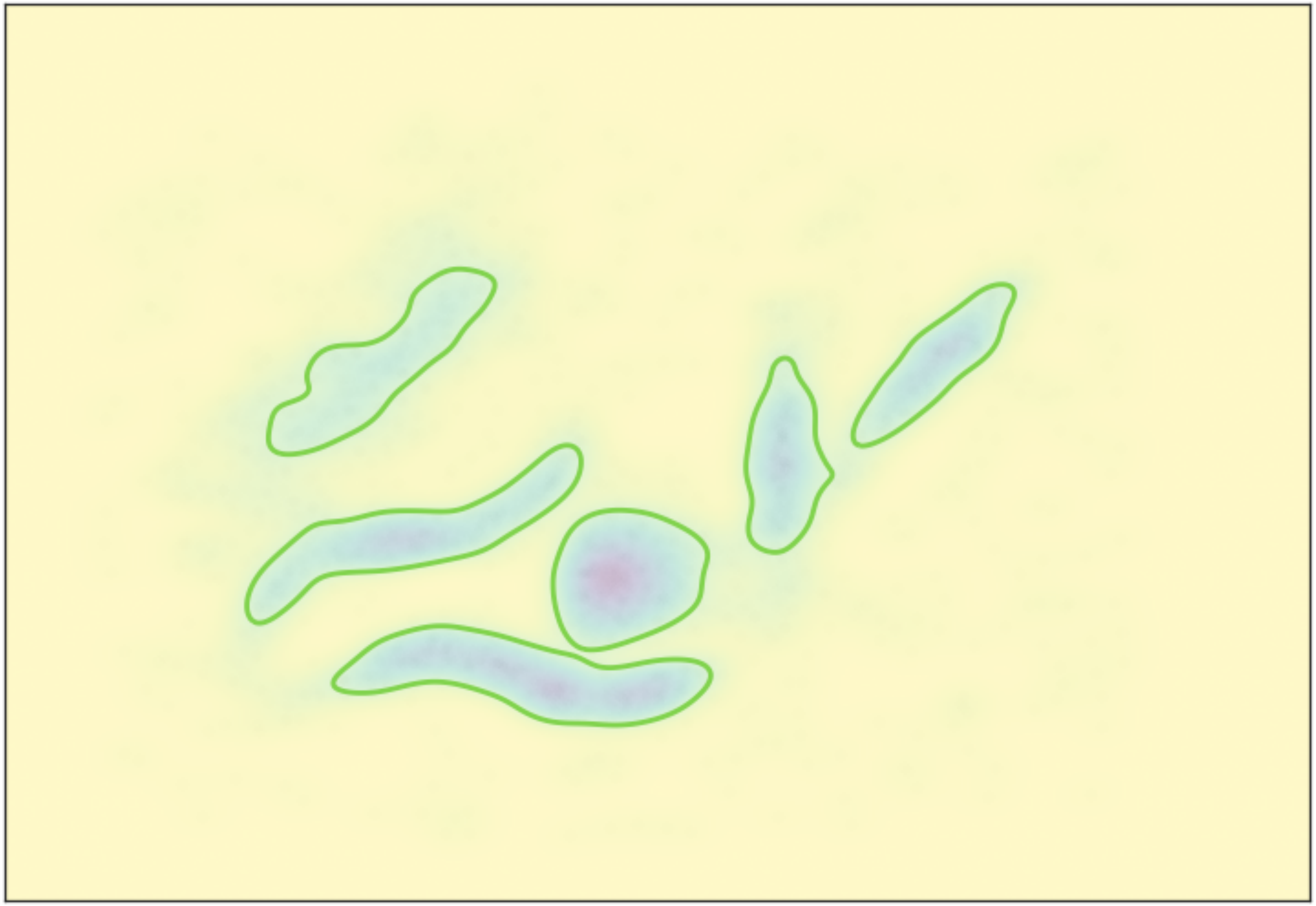
A cluster is …

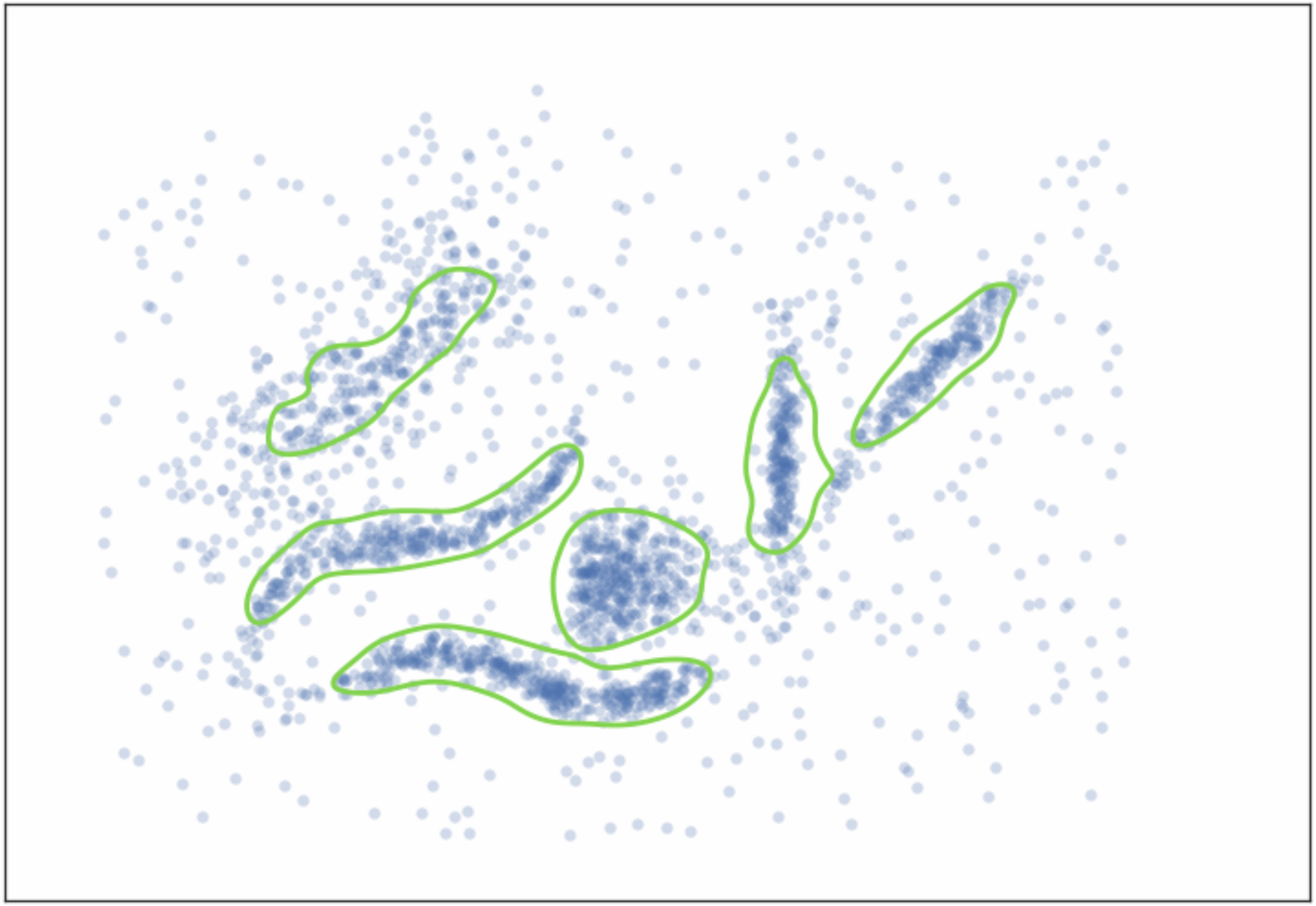
A connected component of a level set of the probability density function of the underlying (and unknown) distribution from which our data samples are drawn.

What?







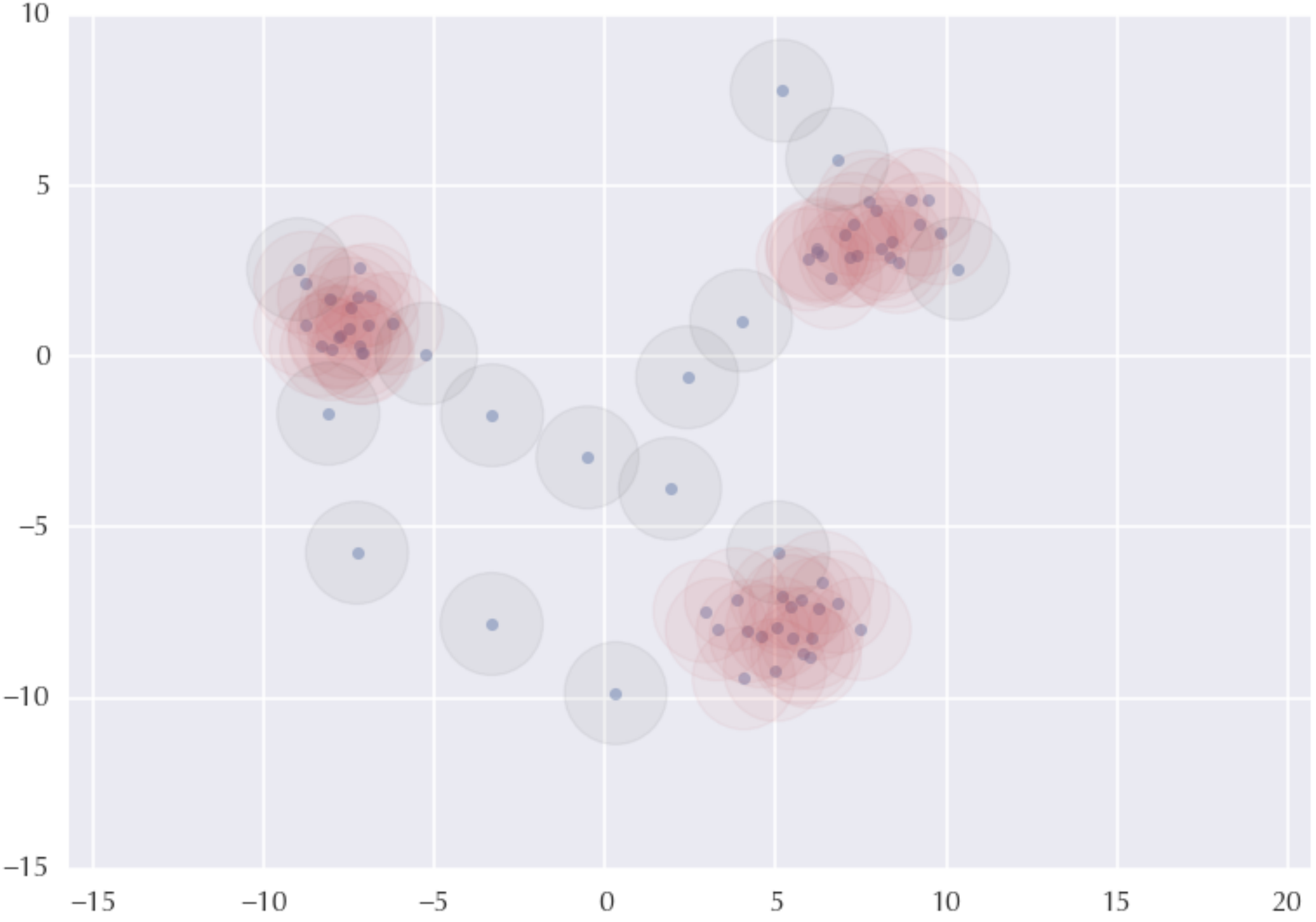


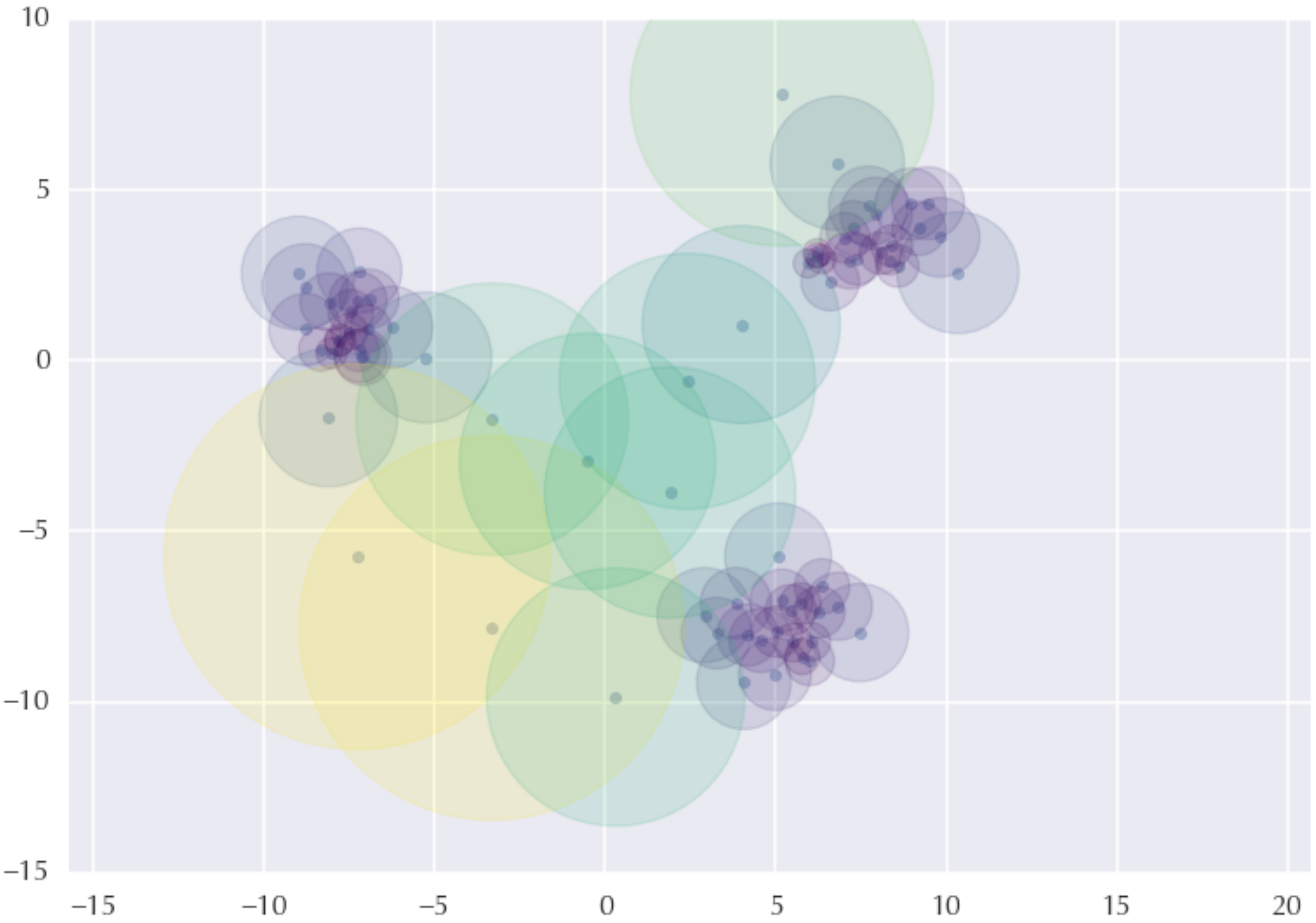
Is that process a clustering algorithm?

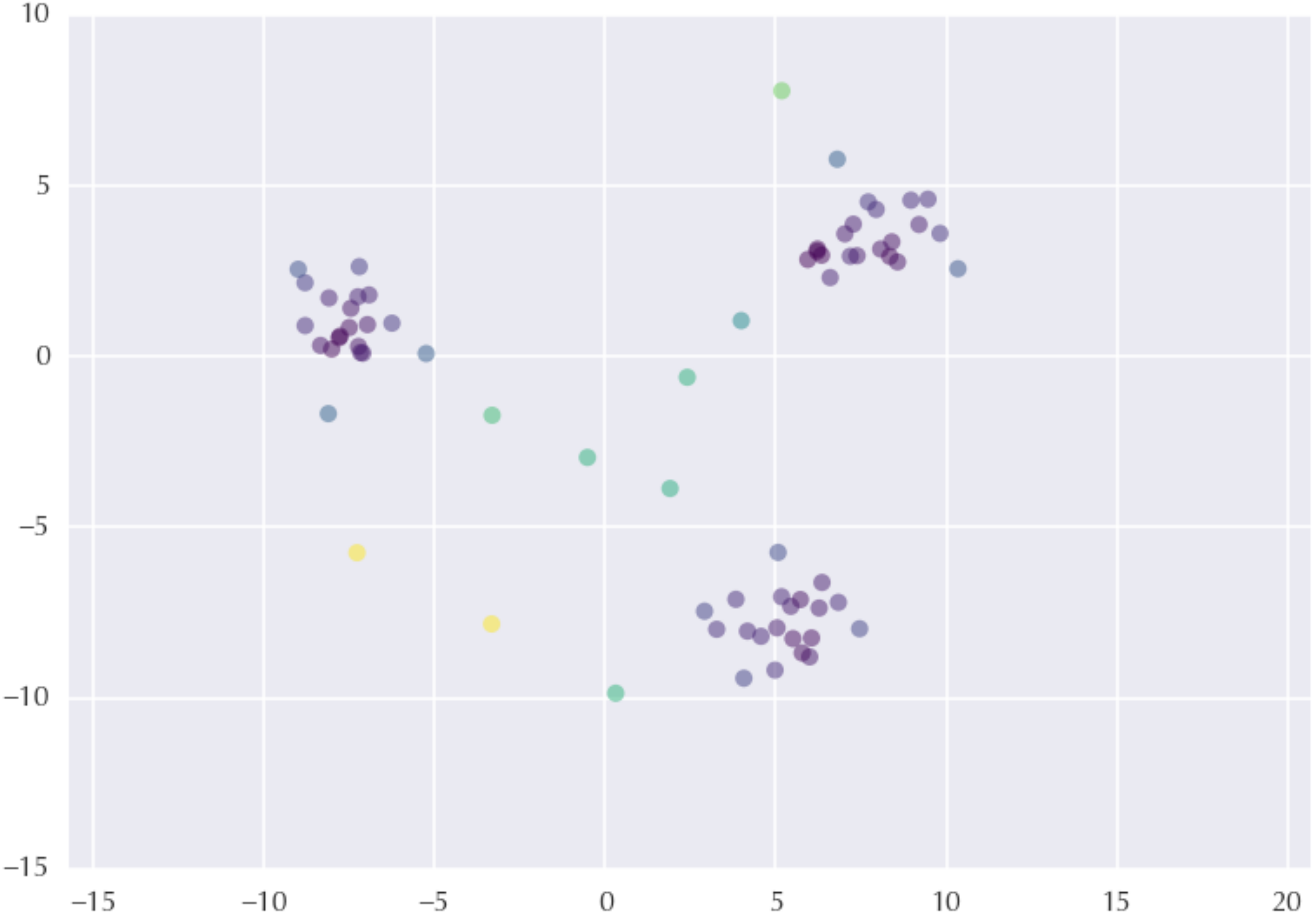
1. We don’t know the PDF
2. Which level sets to choose?
3. Computational complexity

What can we do?

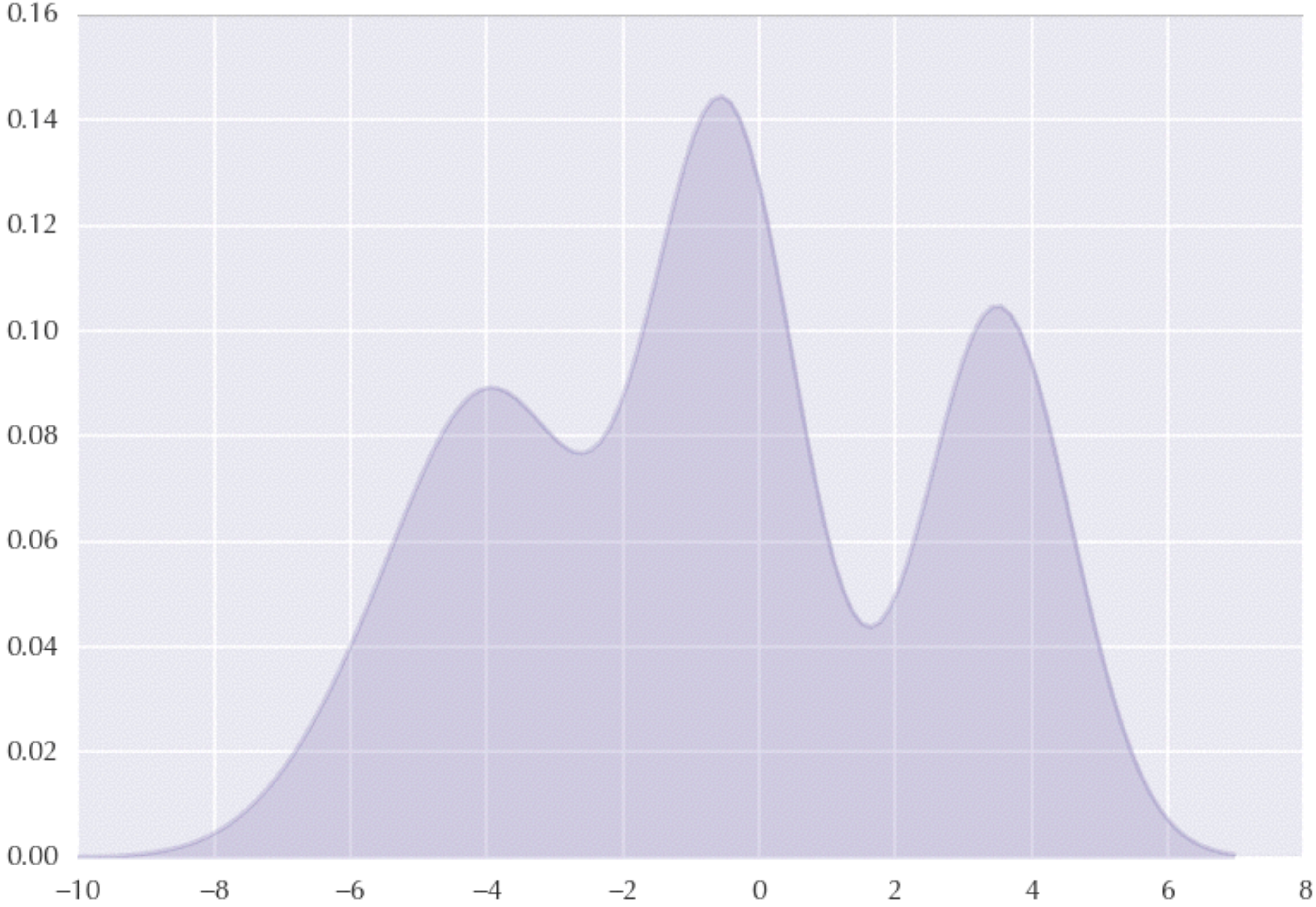
Locally approximate the density

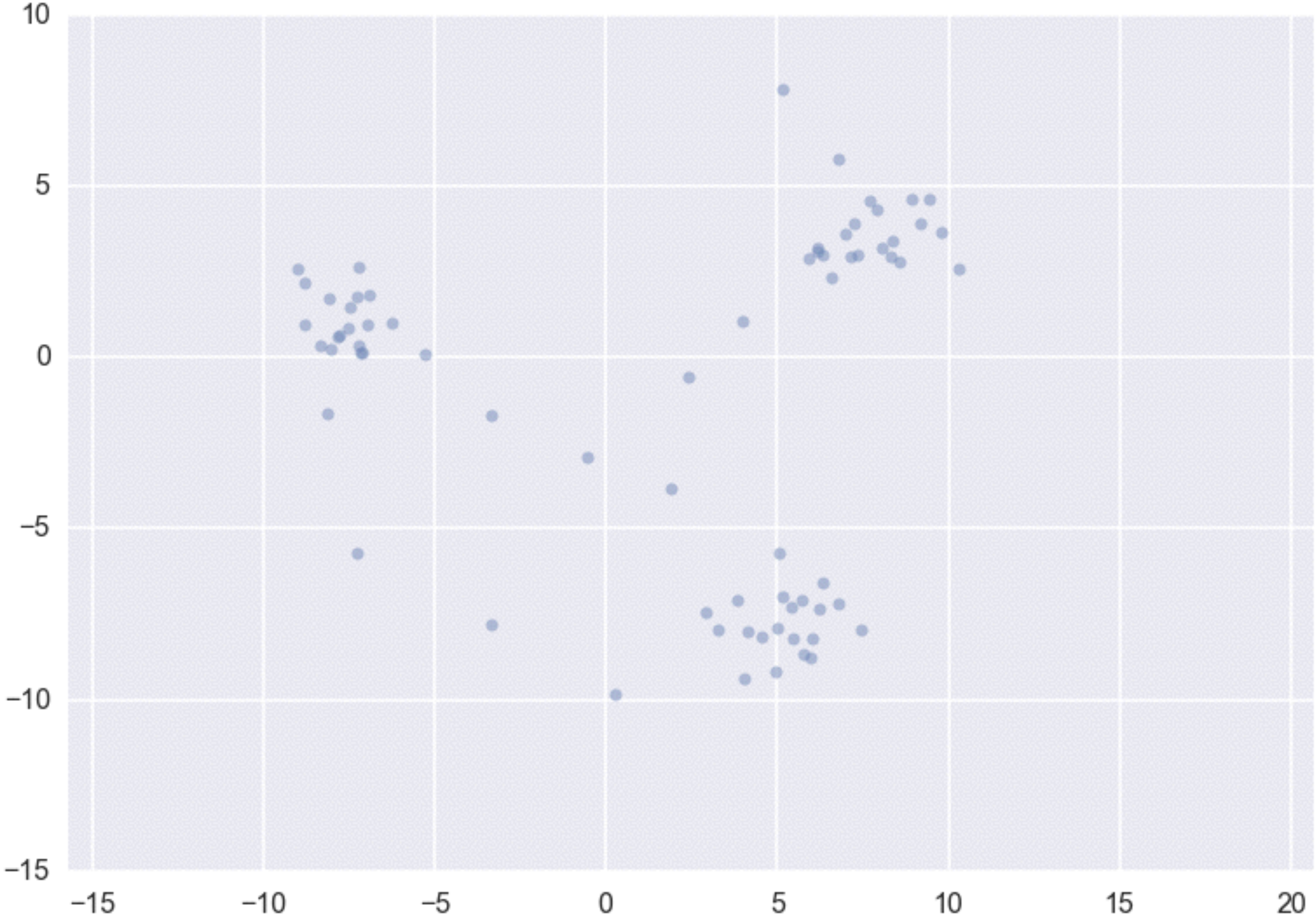

10
5
0
−5
−10
−15
−15
−10
−5
0
5
10
15
20


10
5
0
−5
−10
−15
−15
−10
−5
0
5
10
15
20


10
5
0
−5
−10
−15
−15
−10
−5
0
5
10
15
20

The connected components of level sets form a tree

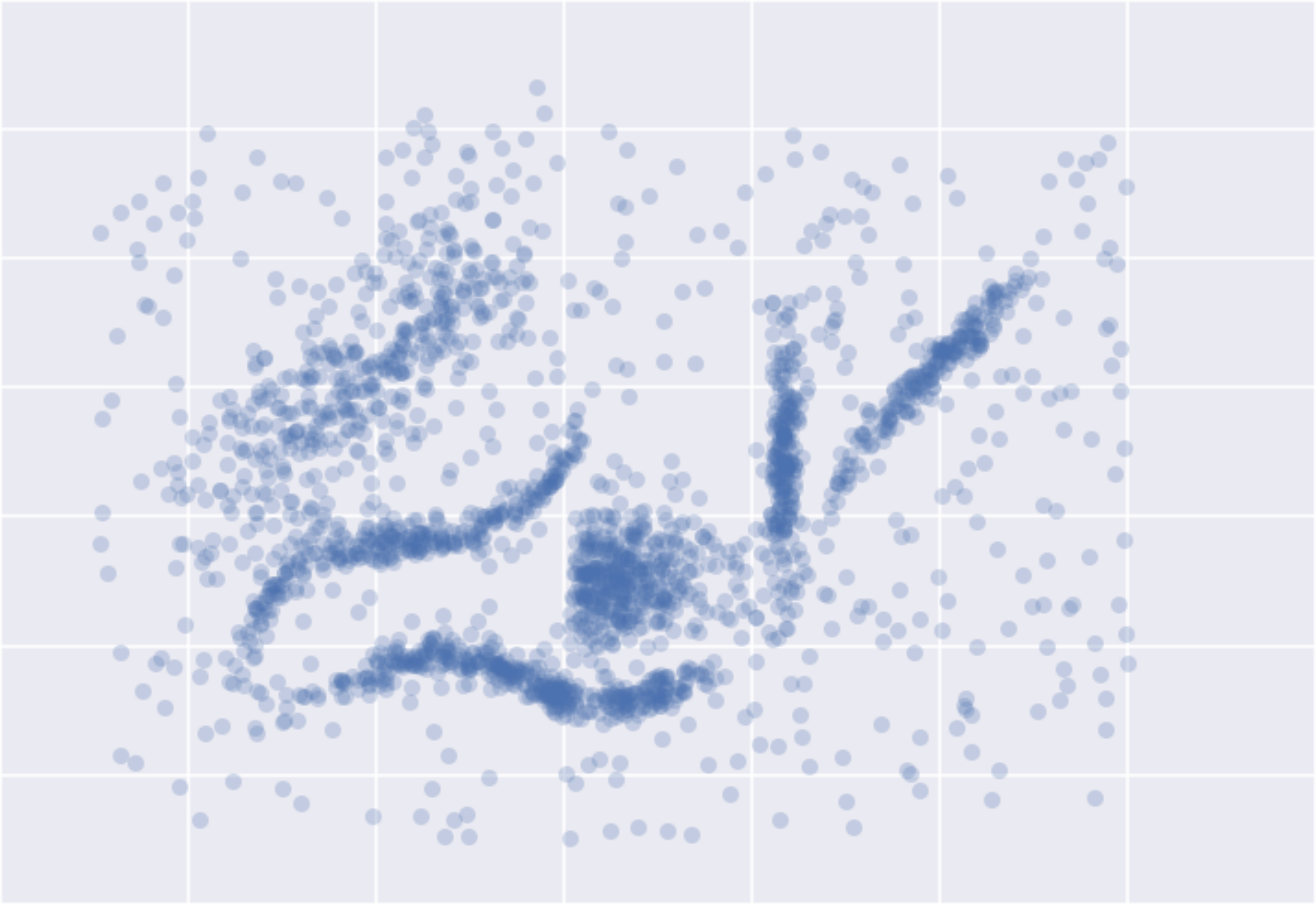

0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
−10
−8
−6
−4
−2
0
2
4
6
8

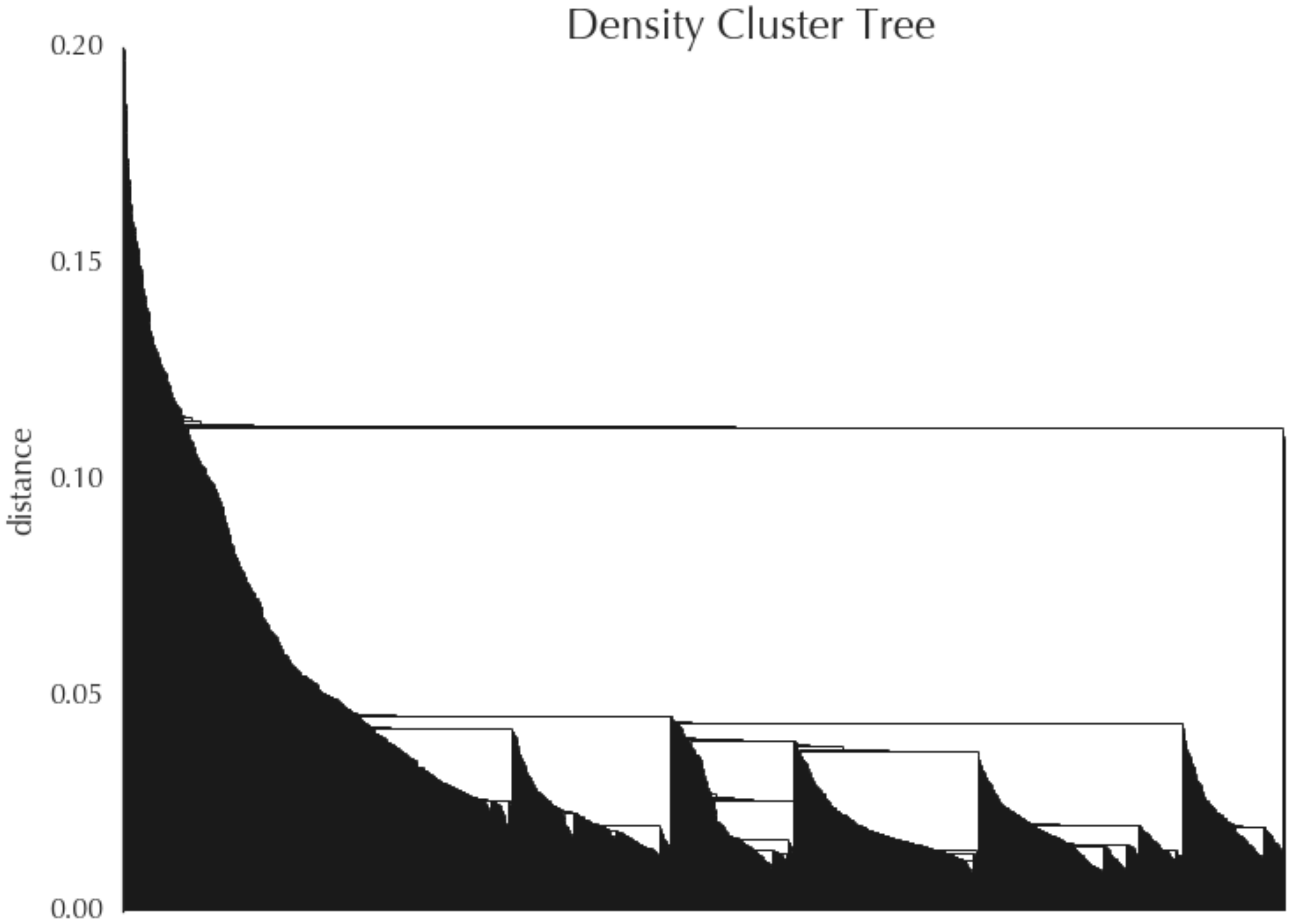


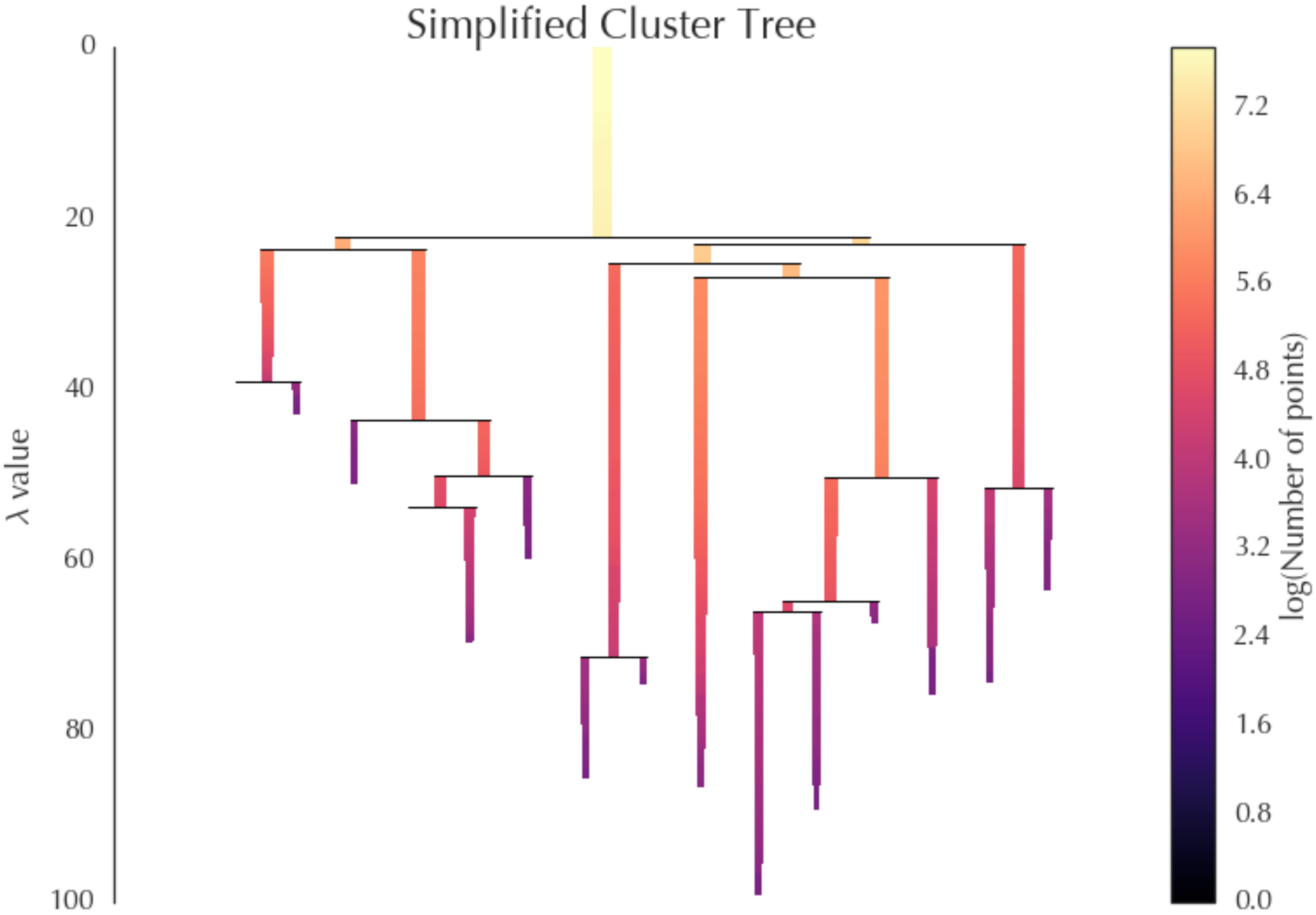
10
5
0
−5
−10
−15
−15
−10
−5
0
5
10
15
20

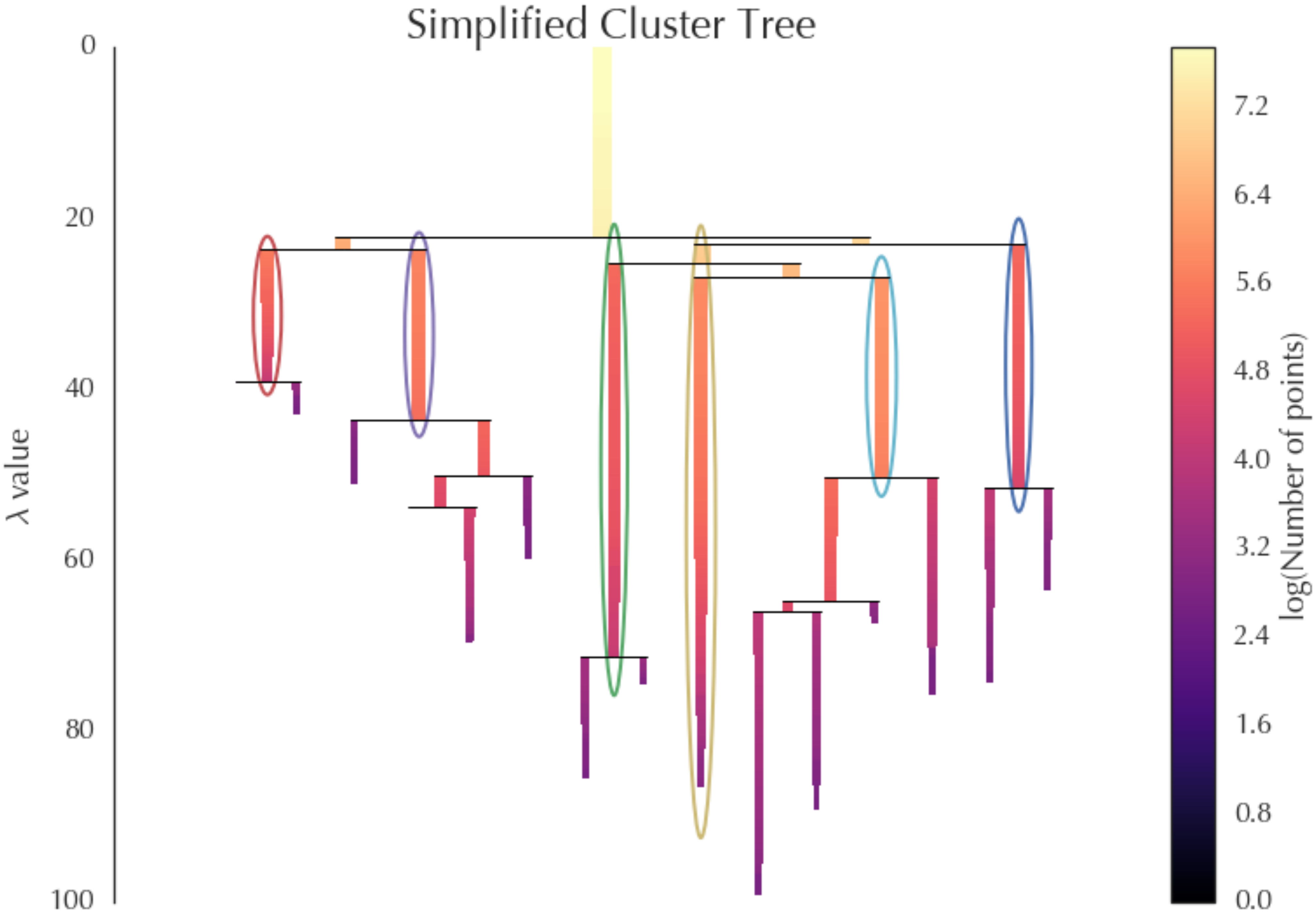
Approximate the level set tree, and use 'Excess of Mass' to select clusters

What does this look like in practice?



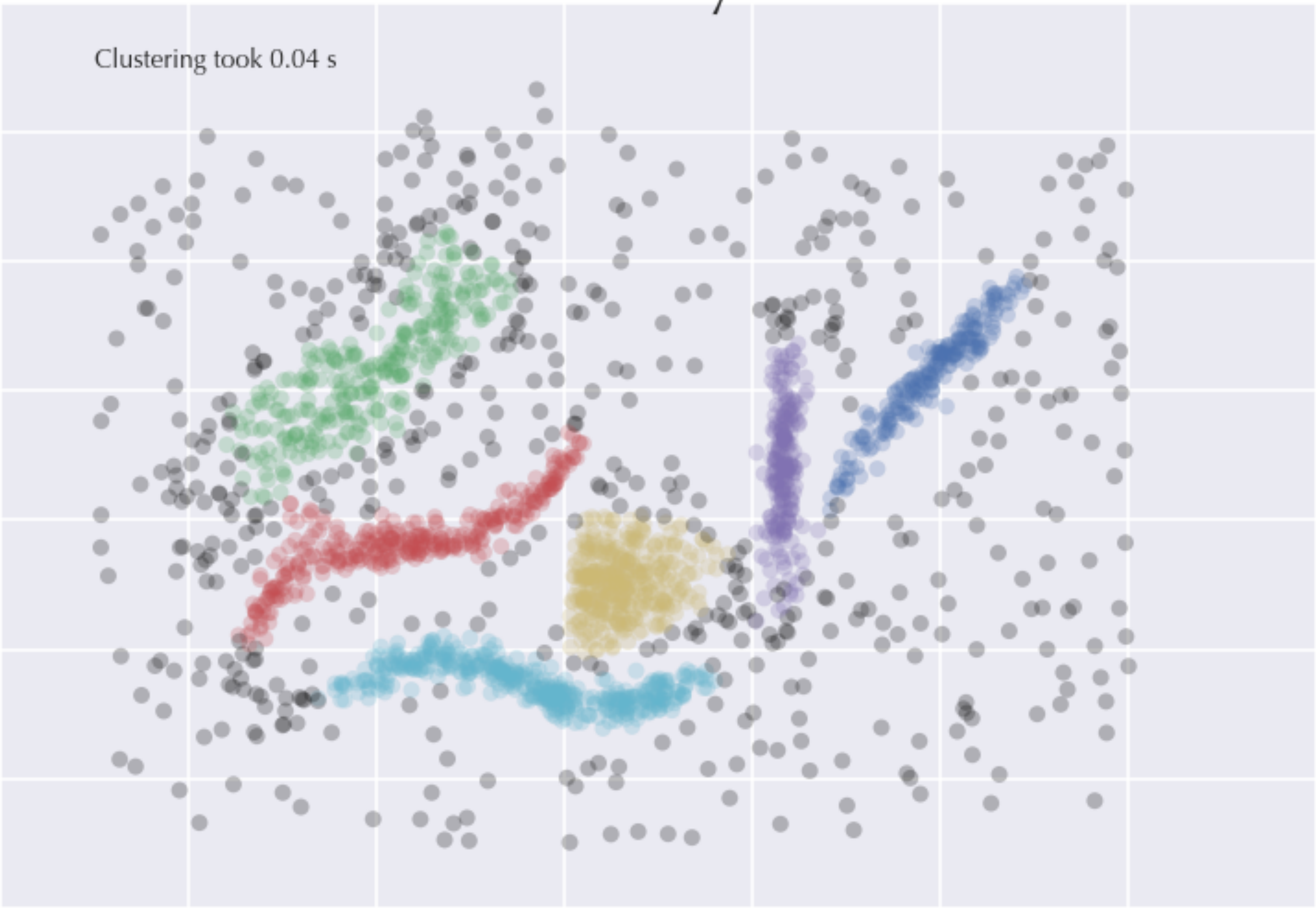

Density Cluster Tree
0.20
0.15
0.10
0.05
0.00
distance


Simplified Cluster Tree
0
20
40
60
80
100
λ value
7.2
6.4
5.6
4.8
4.0
3.2
2.4
1.6
0.8
0.0
log(Number of points)


Simplified Cluster Tree
0
20
40
60
80
100
λ value
7.2
6.4
5.6
4.8
4.0
3.2
2.4
1.6
0.8
0.0
log(Number of points)

And the resulting clusters…

Clusters found by HDBSCAN



INSTITUT TUTTE INSTITUTE

This is the HDBSCAN* clustering algorithm

Campello, Moulavi, Sander 2013
Campello, Moulavi, Sander, Zimek 2015

But what about performance?

We don’t want to run connected components for every possible epsilon!

Minimum spanning trees!

The weighted graph is complete!

We can use spatial indexing to compute fewer distances

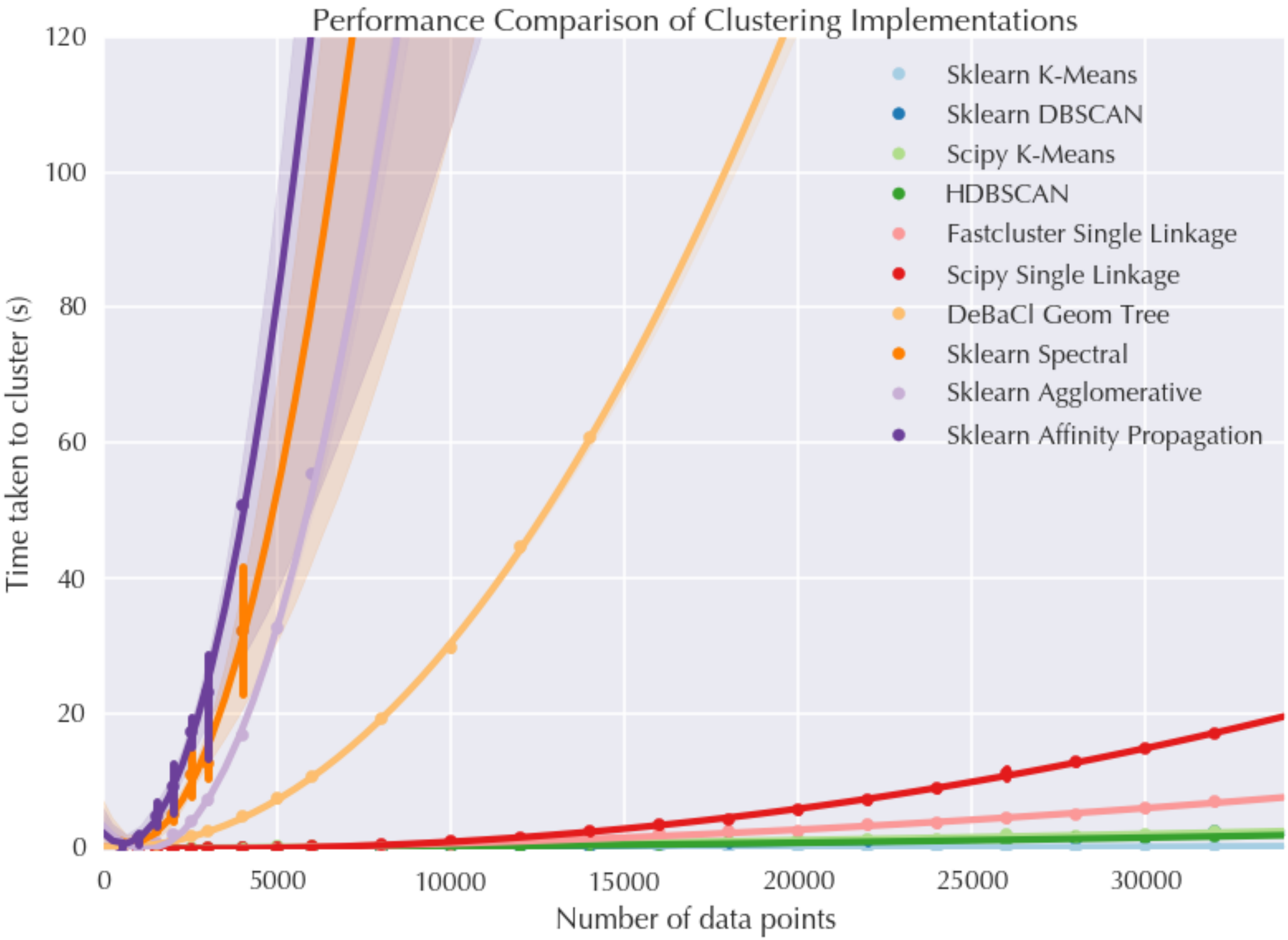
Spatial indexing is great for neighbour queries

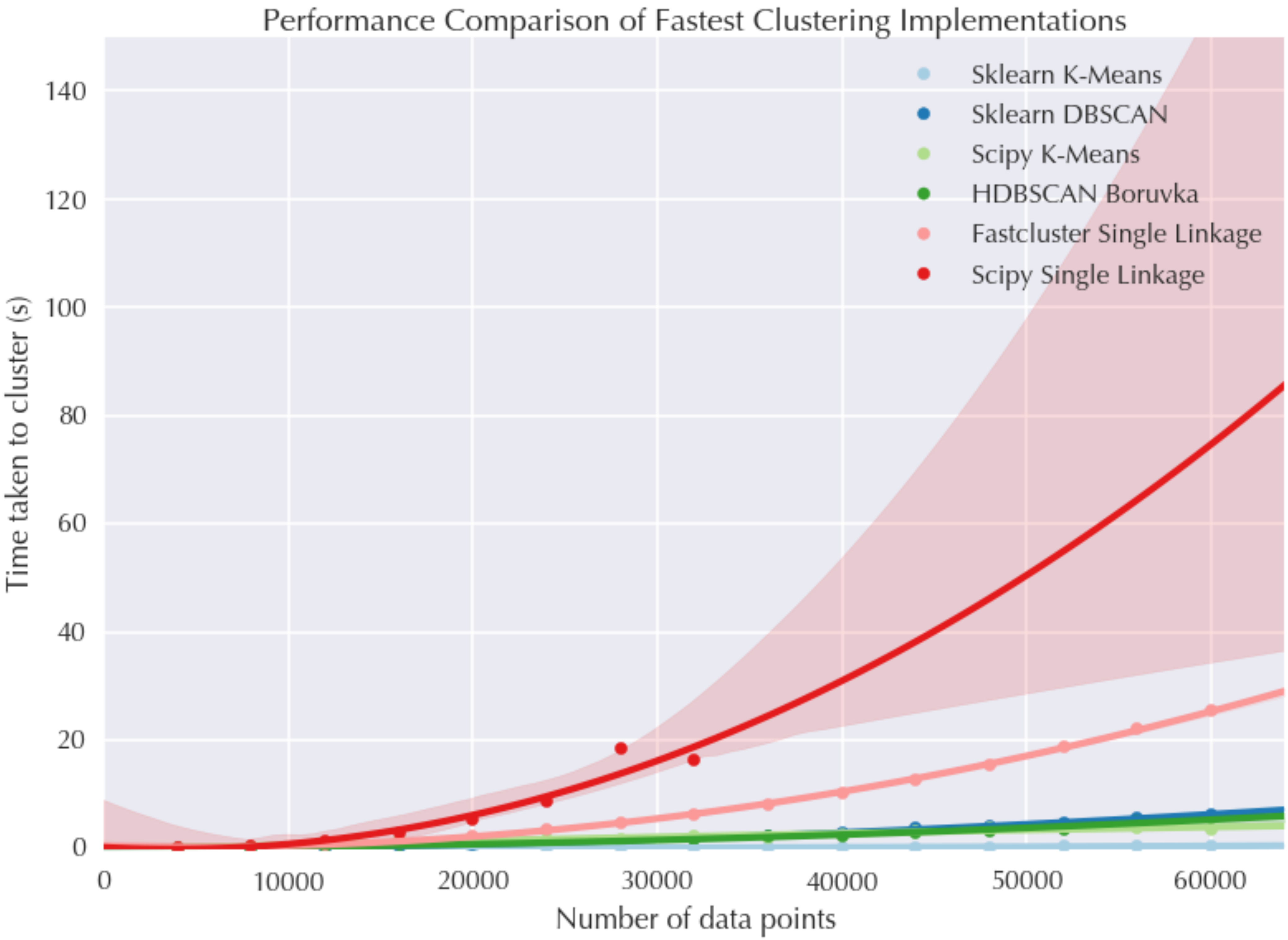
If we use Boruvka's algorithm we can reduce minimum spanning tree computations to repeated neighbour queries

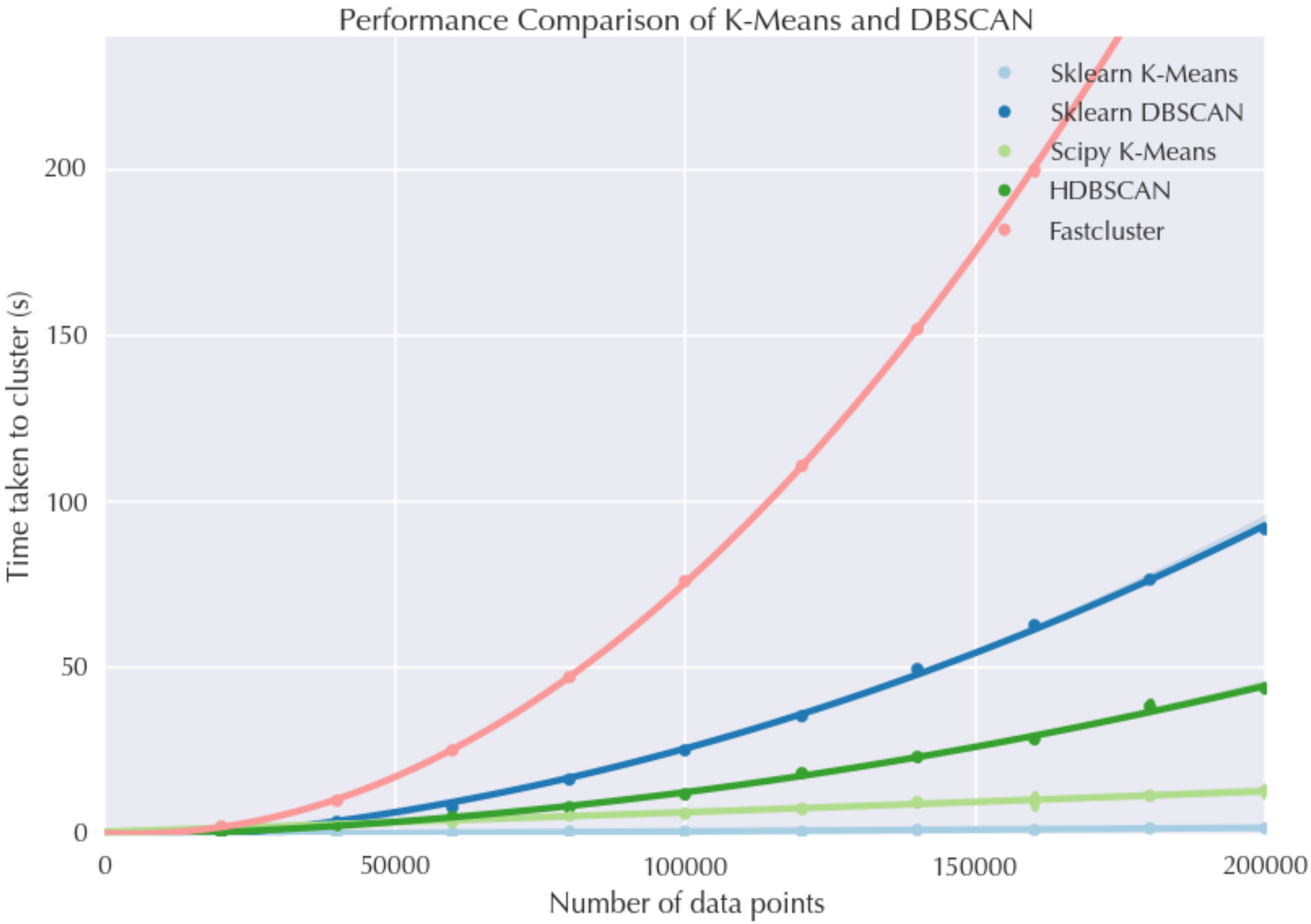
INSTITUT TUTTE INSTITUTE

A modified Dual Tree Boruvka algorithm provides O(n log n) performance

March, Ram, Gray 2010


Performance Comparison of Clustering Implementations
Time taken to cluster (s)
0
20
40
60
80
100
120
Number of data points
0
5000
10000
15000
20000
25000
30000
Sklearn K-Means
Sklearn DBSCAN
Scipy K-Means
HDBSCAN
Fastcluster Single Linkage
Scipy Single Linkage
DeBaCl Geom Tree
Sklearn Spectral
Sklearn Agglomerative
Sklearn Affinity Propagation


Performance Comparison of Fastest Clustering Implementations
Sklearn K-Means
Sklearn DBSCAN
Scipy K-Means
HDBSCAN Boruvka
Fastcluster Single Linkage
Scipy Single Linkage
Time taken to cluster (s)
140
120
100
80
60
40
20
0
0
10000
20000
30000
40000
50000
60000
Number of data points


Performance Comparison of K-Means and DBSCAN
Sklearn K-Means
Sklearn DBSCAN
Scipy K-Means
HDBSCAN
Fastcluster
Time taken to cluster (s)
0
50
100
150
200
Number of data points
0
50000
100000
150000
200000

	Number of data points
Interactive	100,000
Over coffee	500,000
Over lunch	1,000,000
Over night	5,000,000

INSTITUT
TUTTE
INSTITUTE

Better Clustering?

We still have to choose a “number of points” value for the density estimate

Can’t we vary that the way we varied epsilon?

Topology to the rescue!

Building the initial HDBSCAN* tree can be described in terms of persistent homology

Multi-dimensional persistent homology allows for persistence over multiple variables

Unfortunately the result is not a tree

We can re-interpret HDBSCAN* using sheaves instead of trees

The resulting algorithm can be generalized to the multi-dimensional case!

INSTITUT TUTTE INSTITUTE

This (will be) Persistent Density Clustering

More robust

Fewer parameters

Similar performance

INSTITUT TUTTE INSTITUTE

Conclusions

K-Means shouldn't be your first choice

K-Means probably shouldn’t be your second choice either

Think hard about what “cluster” means for your application

You may as well run HDBSCAN* while you’re thinking!

https://github.com/scikit-learn-contrib/hdbscan

```
conda install -c conda-forge hdbscan
```

```
pip install hdbscan
```