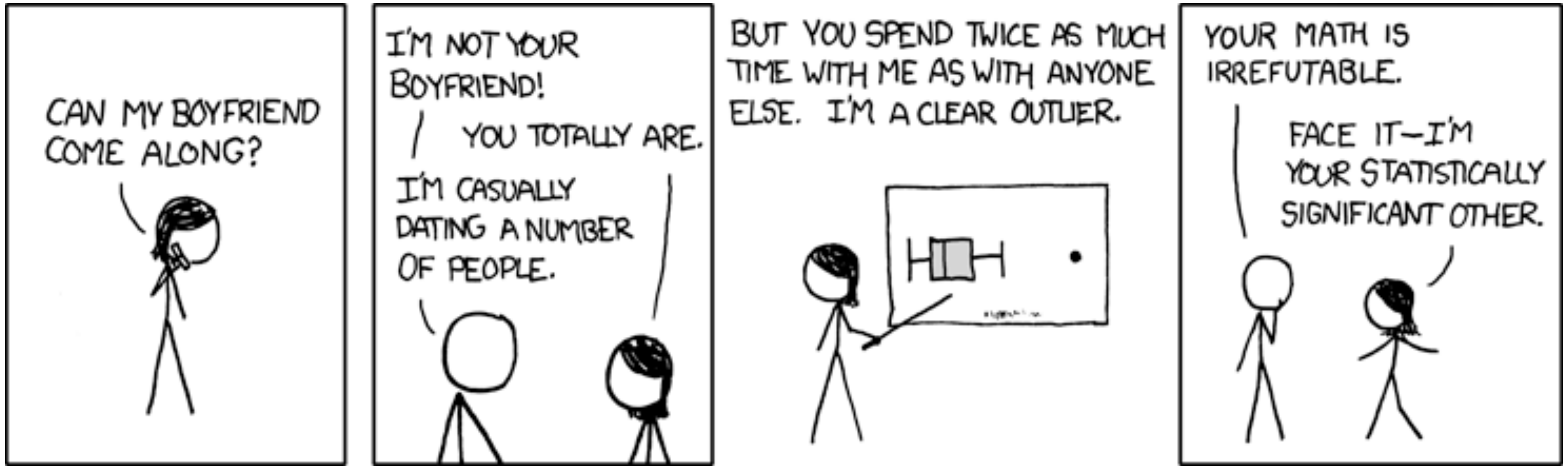
Introduction to Data Science
CS 5963 / Math 3900
Clustering

Alexander Lex
alex@sci.utah.edu

Braxton Osting
osting@math.utah.edu



[xkcd]

Unsupervised Learning

Up to now, we've only used labeled data for supervised learning:

- Train an algorithm on a training set, hope that it generalizes.

What if we don't have labels?

- Which customers have similar behavior?
- Which genes are co-regulated?
- Which images are similar?

Labels can be expensive or impossible to obtain!

Clustering

Almost every interesting dataset has some grouping structure

- E.g., 24 year old college students might have similar sleeping behavior

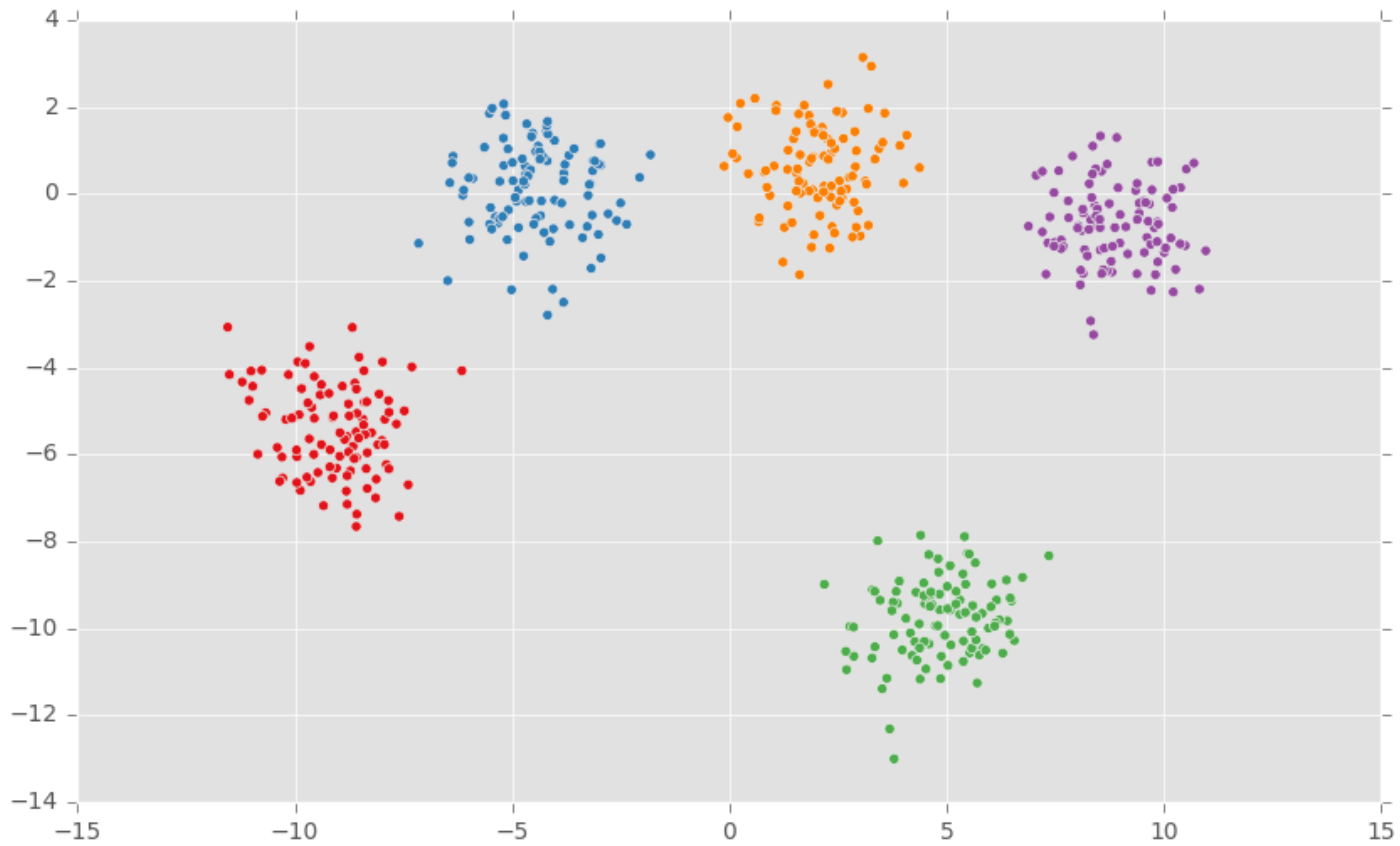
Clustering can try to tease out these groups

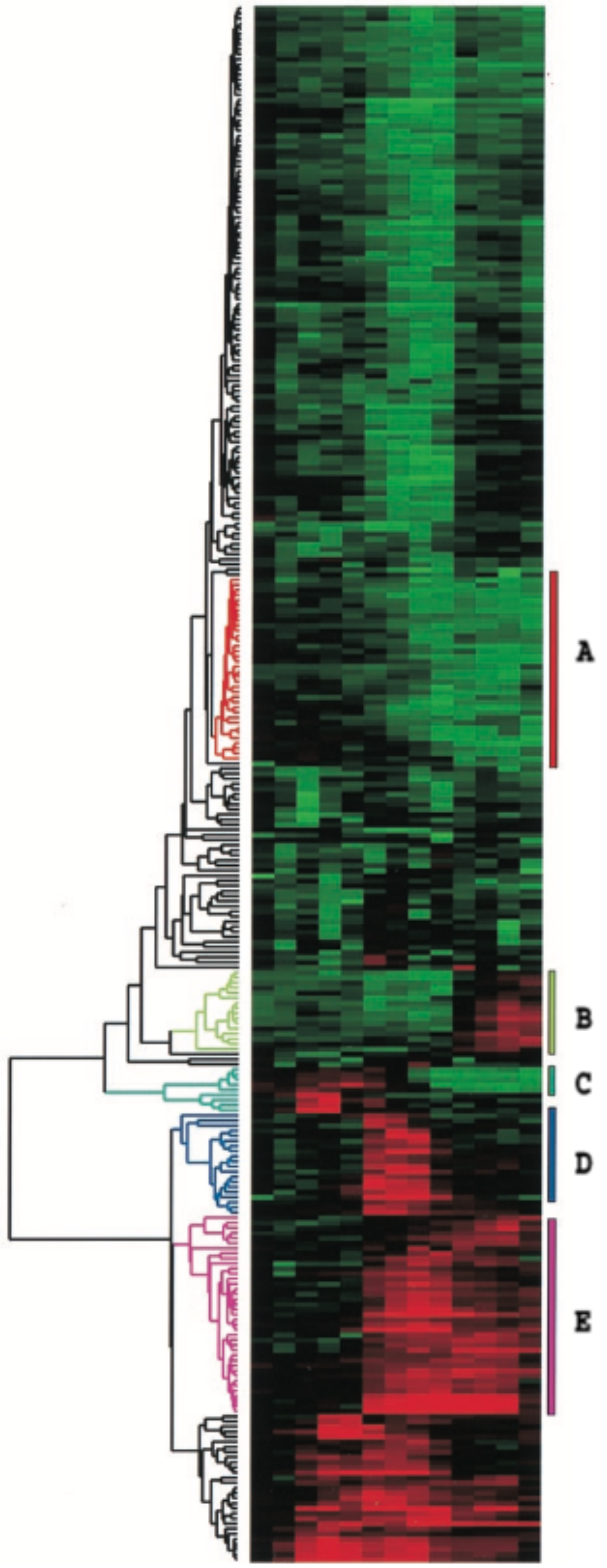
Clustering often doesn’t have “correct” results

- often necessary to compare various clusters

Visualization

Important to judge cluster quality





Clustering

The groups in cluster’s aren’t automatically labeled

- we know they’re similar, but that doesn’t mean it’s easy to describe it

Clustering is based on a “distance” or a “similarity” function

- it is often not obvious what the best function is
  - Euclidean distance, Pearson correlation, Manhattan distance, weighted distances, ….

Types of Clustering Algorithms

Partitional Algorithms

- divide data into set of bins
- # bins either manually set (e.g., k-means) or automatically determined (e.g., affinity propagation)

Hierarchical Algorithms

- Produce "similarity tree" – dendrogram
- discrete cluster can be produced by "cutting" a dendrogram

Bi-Clustering

- Clusters dimensions & records

Fuzzy clustering

- probabilistic cluster assignment
- allows occurrence of elements in multiples clusters

K-Means

Goal

Minimizes aggregate intra-custer distance (*inertia*)

$$\underset{C}{argmin} \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

total squared distance from point to center of its cluster

for euclidian distance: this is the variance

measure of how internally coherent clusters are

Lloyd's Algorithm

Input: set of records $x_1 \ldots x_n$, and k (nr clusters)

Pick k starting points as centroids $c_1 \ldots c_k$

While not converged:

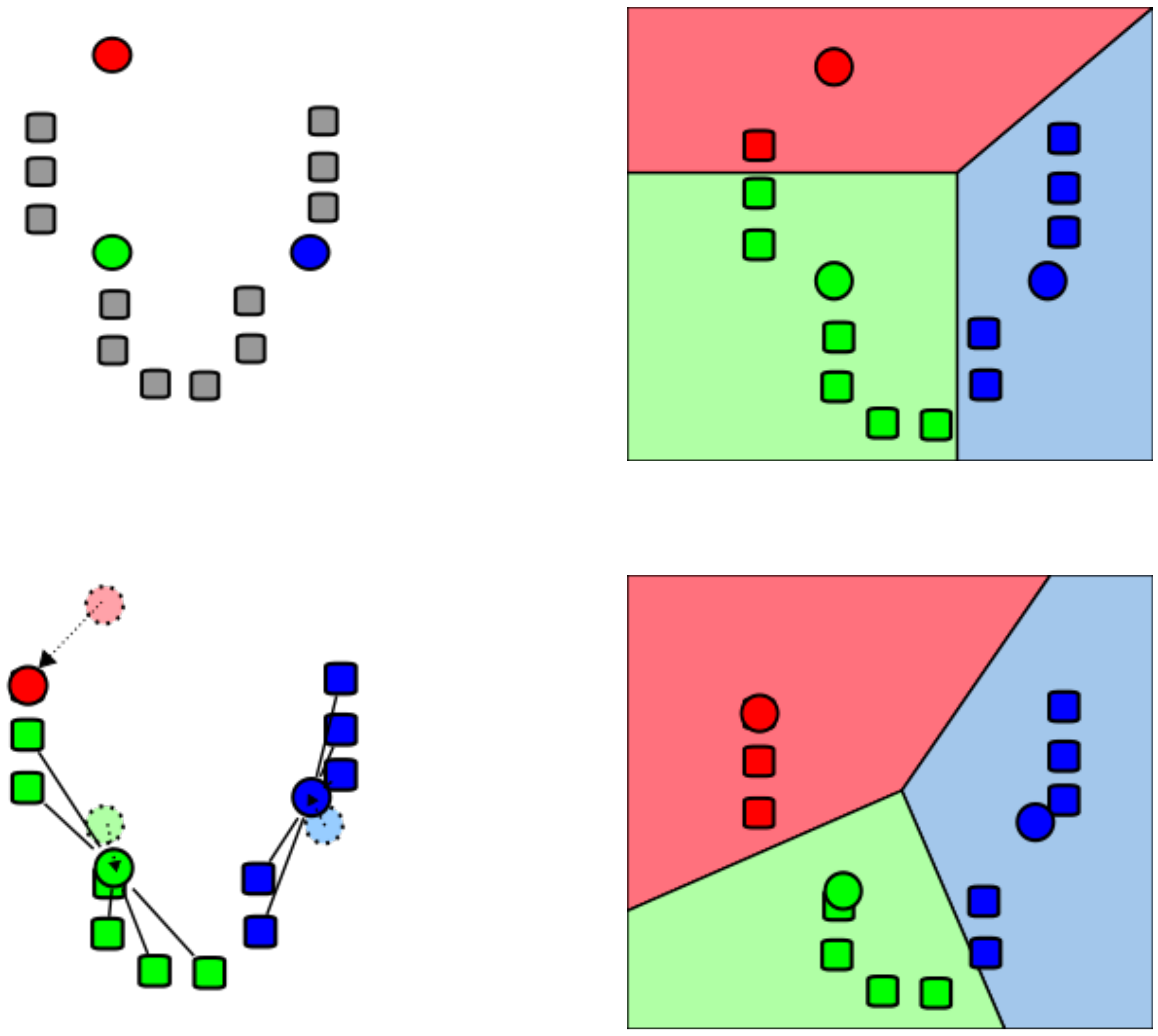
1. for each point x_i find closest centroid c_j
   - for every c_j calculate distance $D(x_i, c_j)$
   - assign x_i to cluster j defined by smallest distance
2. for each cluster j, compute a new centroid c_j
   by calculating the average of all x_i assigned to cluster j

Repeat until convergence, e.g.,

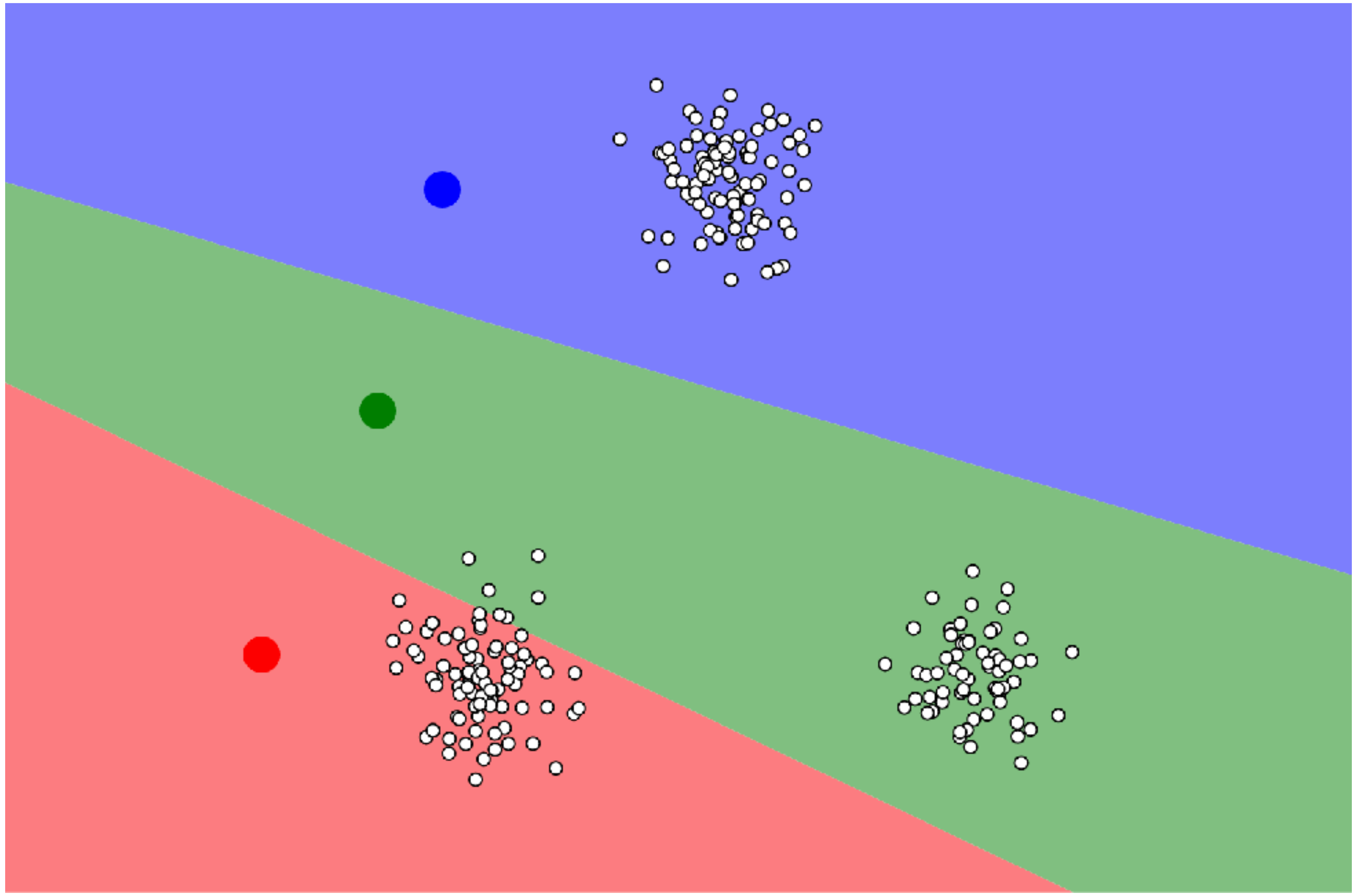
no point has changed cluster

distance between old and new centroid below threshold

number of max iterations reached



Illustrated



https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

Performance

The performance is $O(n*k*d*i)$

where n is the number of records,

k is the number of clusters

d is the number of dimensions

i is the number of iterations needed until convergence

For data that has clusters, i is usually small

In practice: very fast

Properties

Lloyds algorithm doesn’t find a global optimum

Instead it finds a local optimum

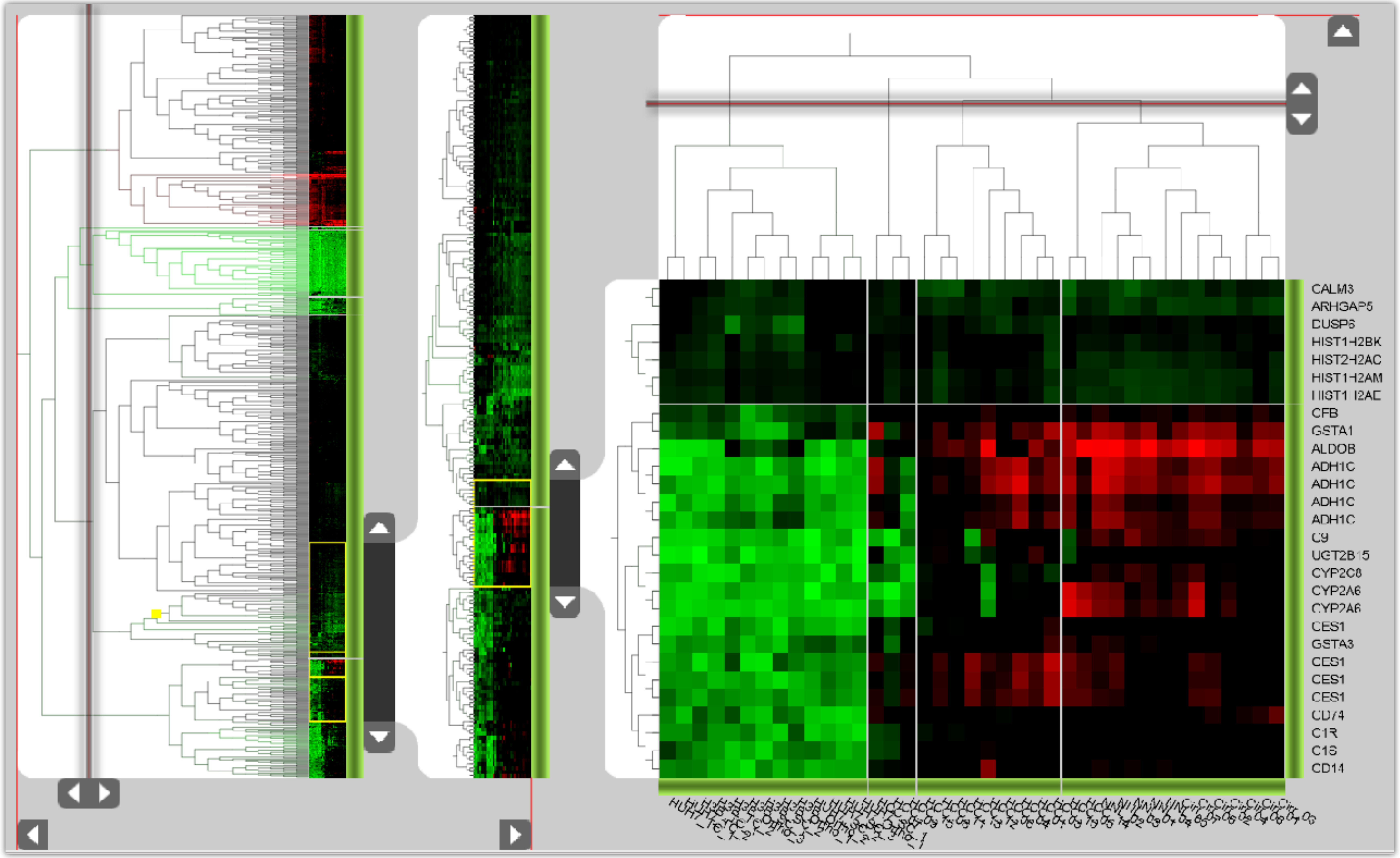
It is very fast:

- common to run multiple times and pick the solution with the minimum inertia

Let’s look at scikit learn k-means

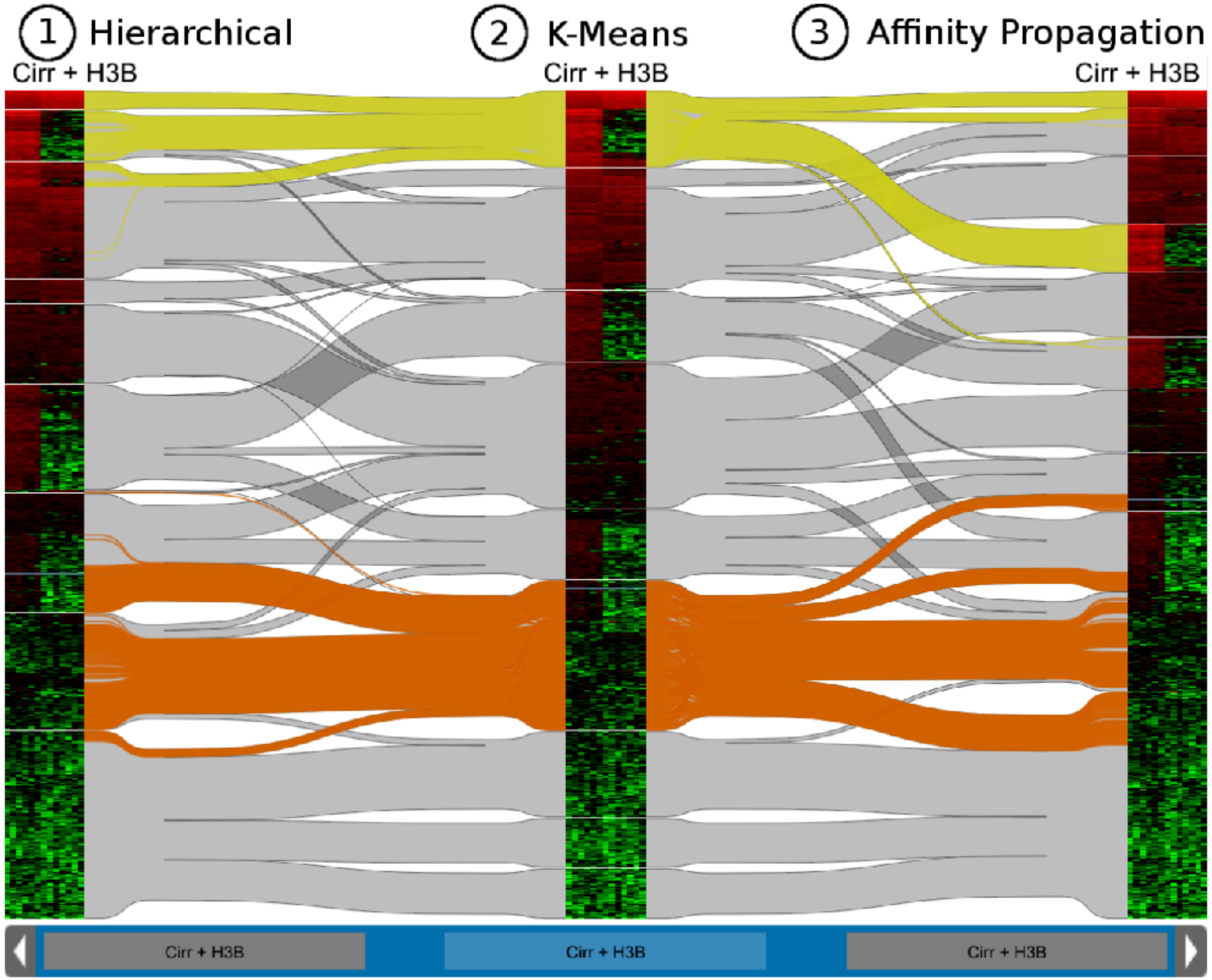
Applications and Visualization

F+C Approach, with Dendrograms

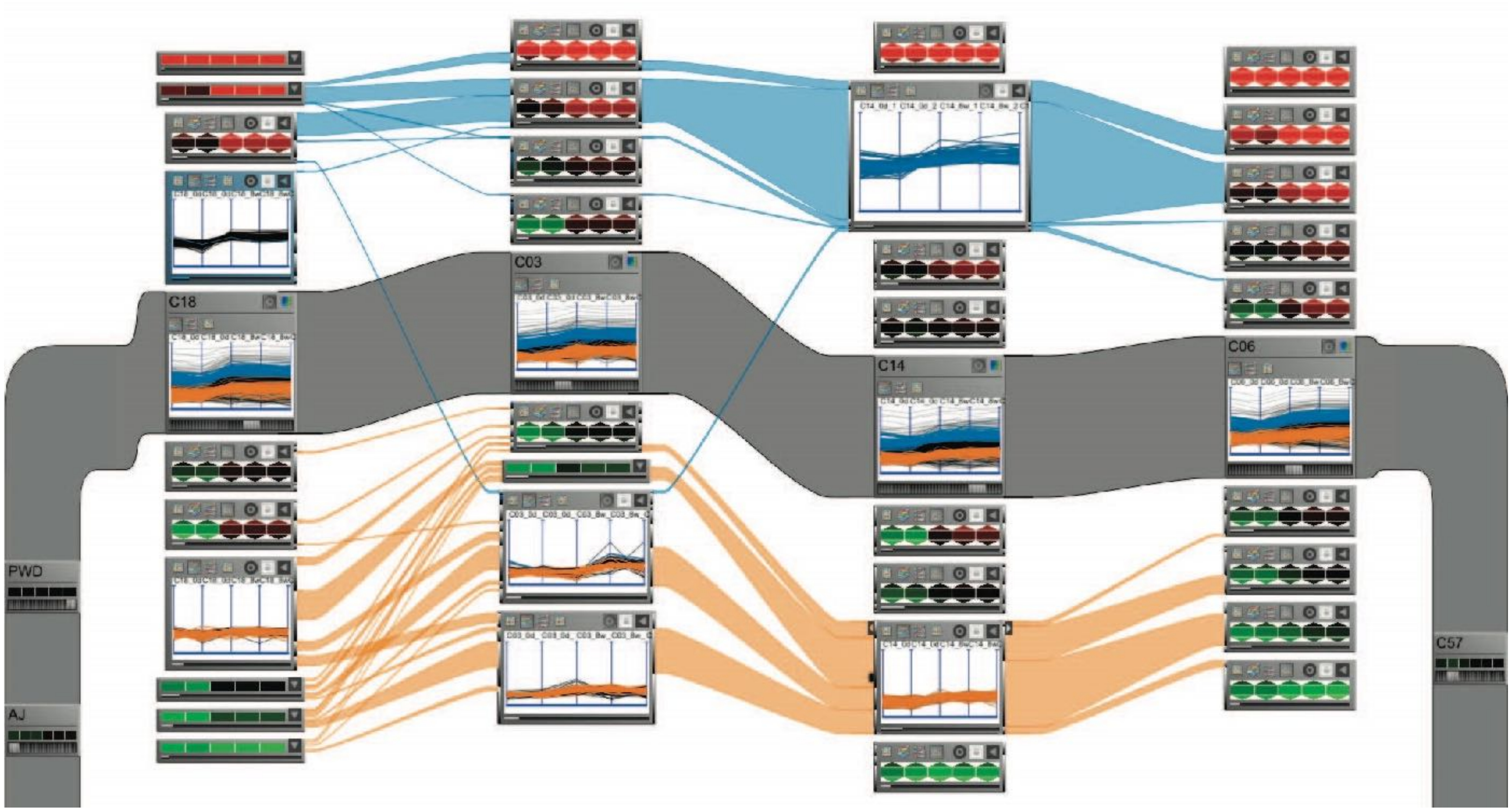


[Lex, PacificVis 2010]

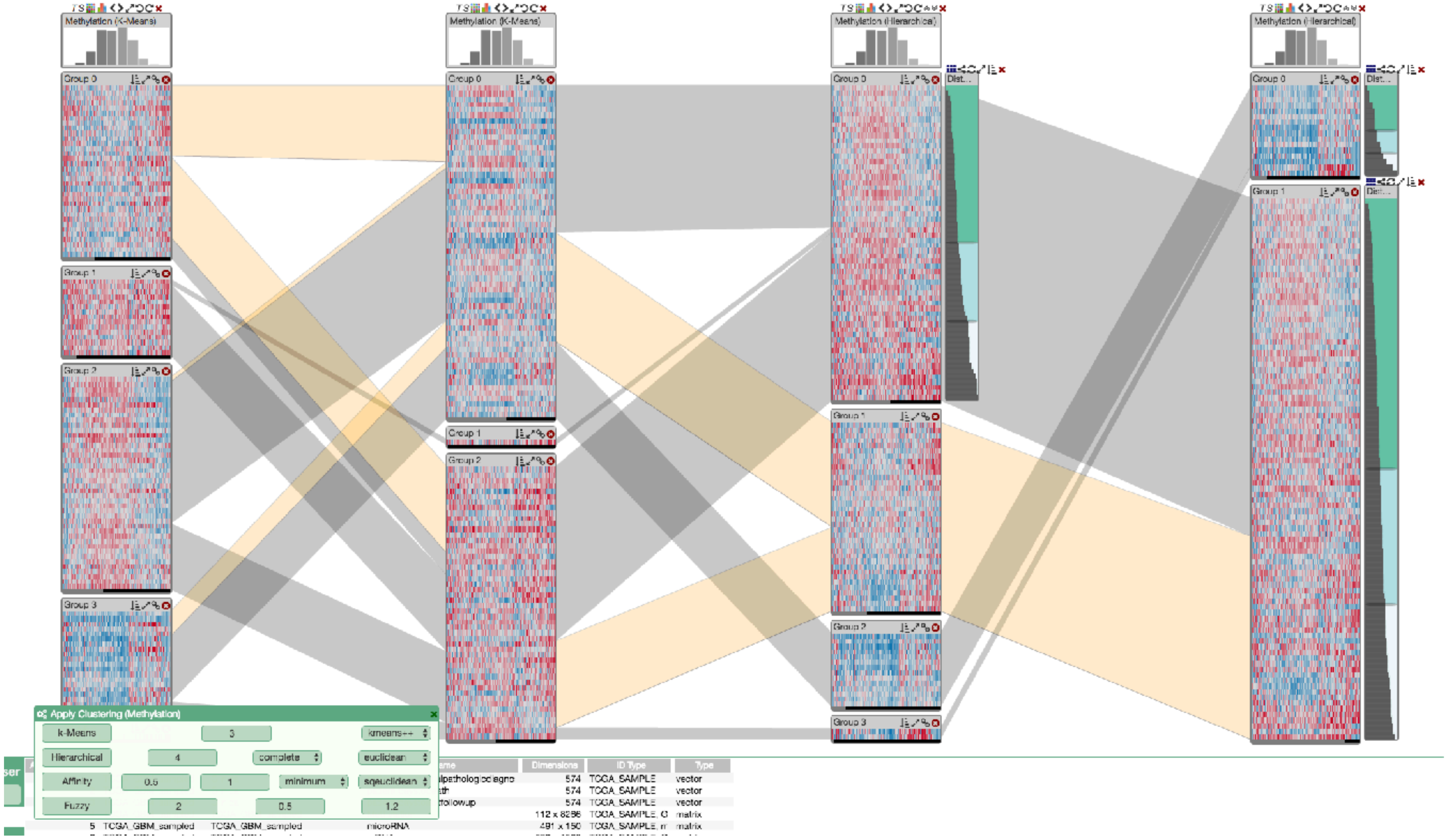
Cluster Comparison



Aggregation



Interactive Exploration



http://ec2-52-58-105-31.eu-central-1.compute.amazonaws.com/stratomex_clustering/