Clustering

Everything Data

CompSci 216 Spring 2015

DUKE
COMPUTER SCIENCE

Announcements (Tue. Feb 24)

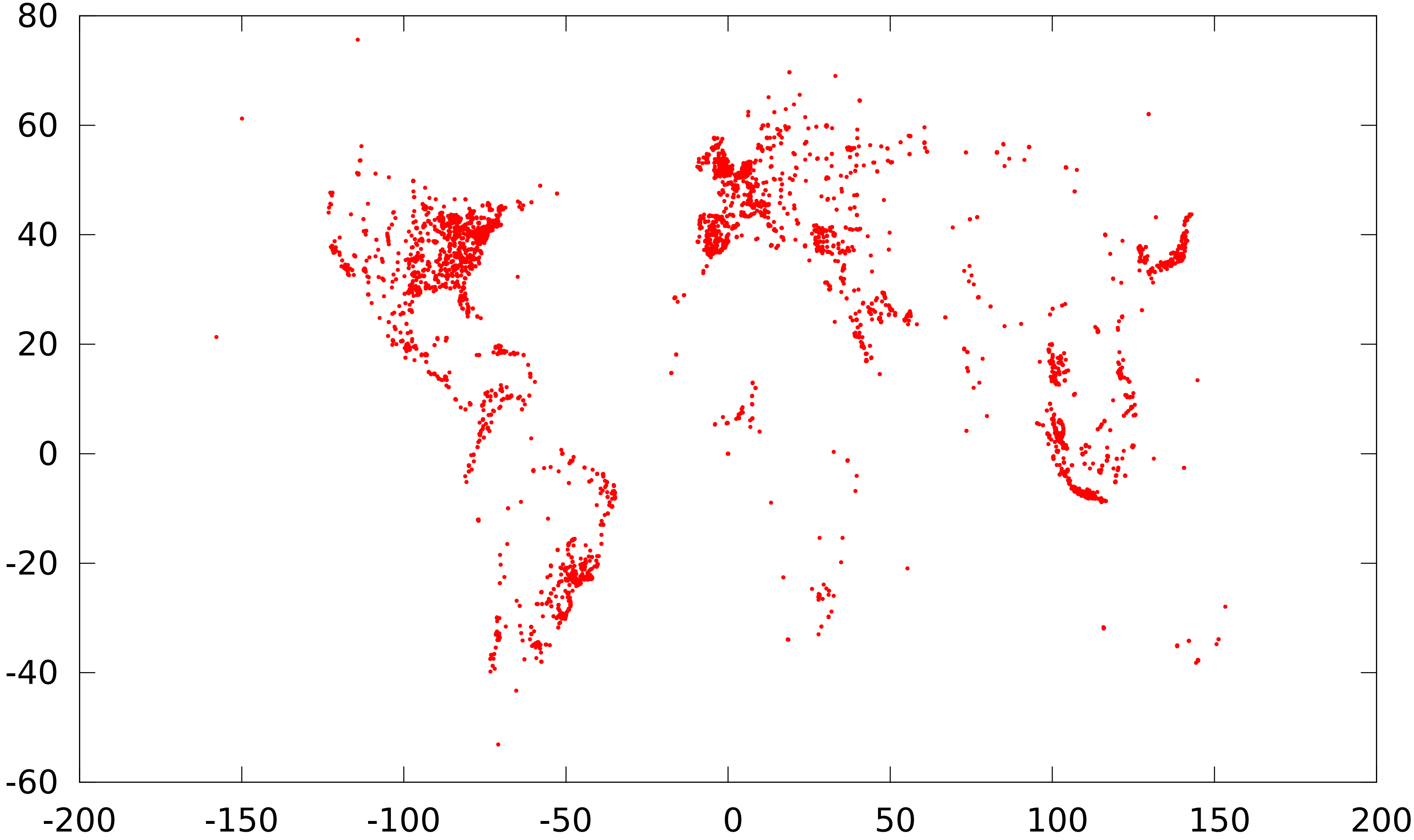
- **Homework #7** will be posted before tomorrow.

- **Office Hours:**
  - **Jun: Thursday 2:45-3:45 pm**
  - **Ashwin: Friday 3:30-4:30 pm**

Announcements (Tue. Feb 24)

- **Project presentations on Monday**
  - 4 minutes per team
  - Introduce your team members
  - Describe problem, dataset and how you will quantify success
  - You may use 1-2 slides (PDF format)
    - Submit your slides **before** the lab.

Geo-tags of tweets

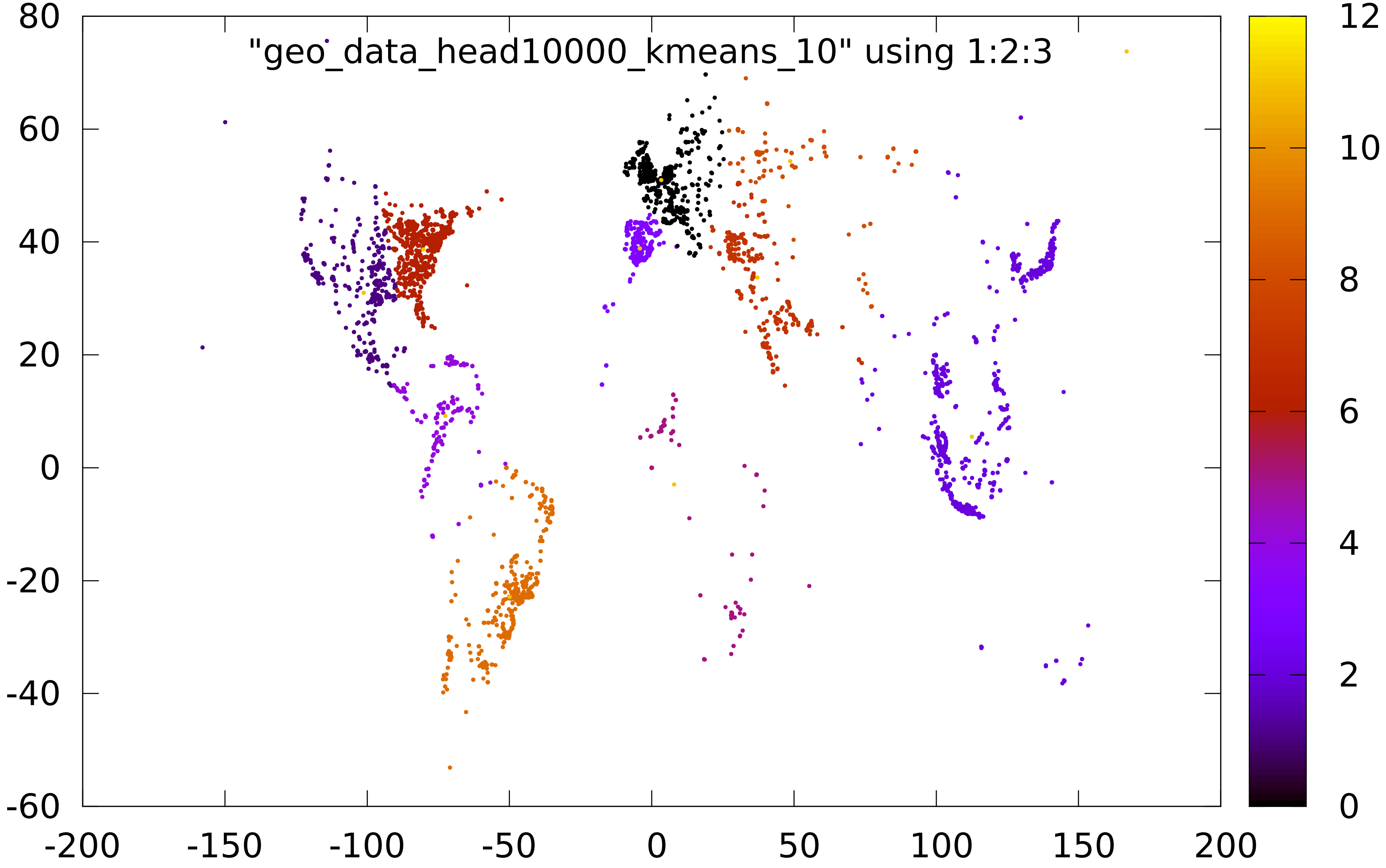


Trending topics

- How would you compute trending topics?
  - Most frequent hashtags
  - Frequent keywords or phrases (which are not stopwords)
  - …

- But interesting trends in one region may not represent interesting trends in another.

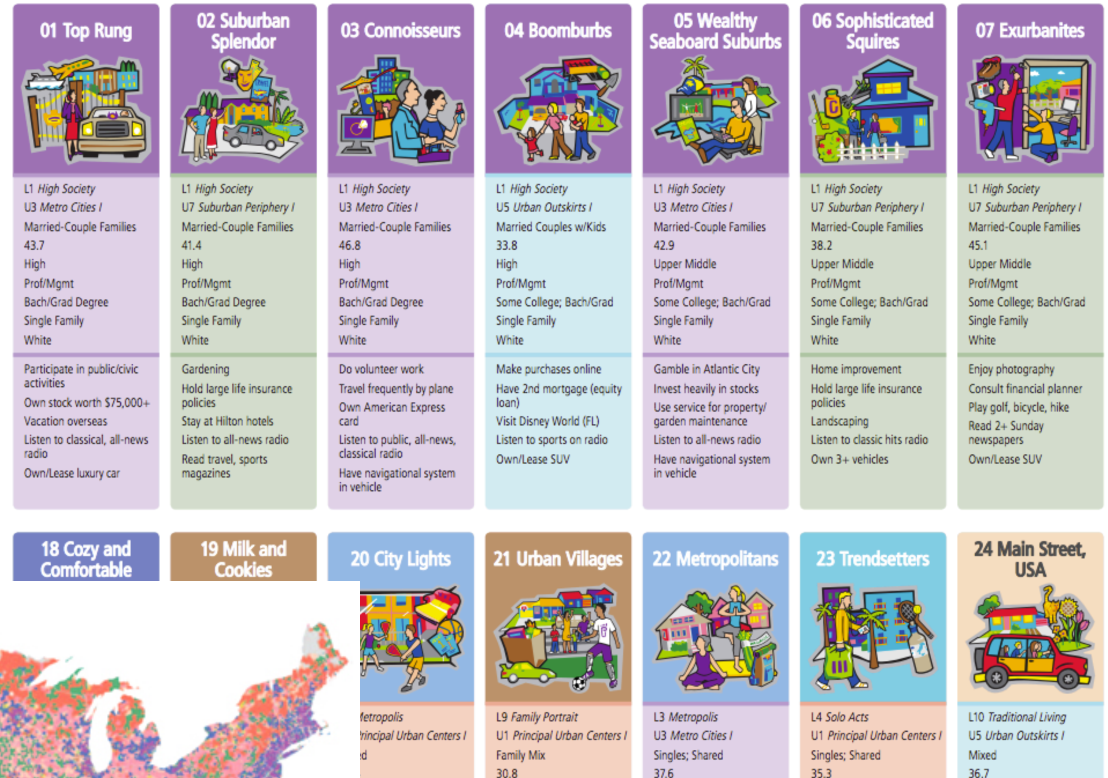
Idea: Cluster tweets by geography

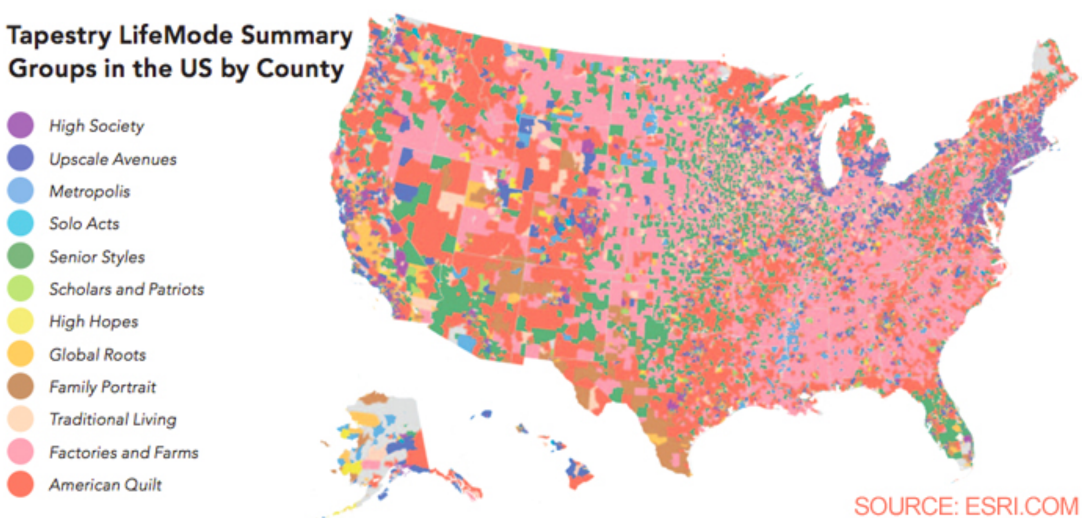


Trending topics by geography

- We can now compute trending topics within each cluster (region).

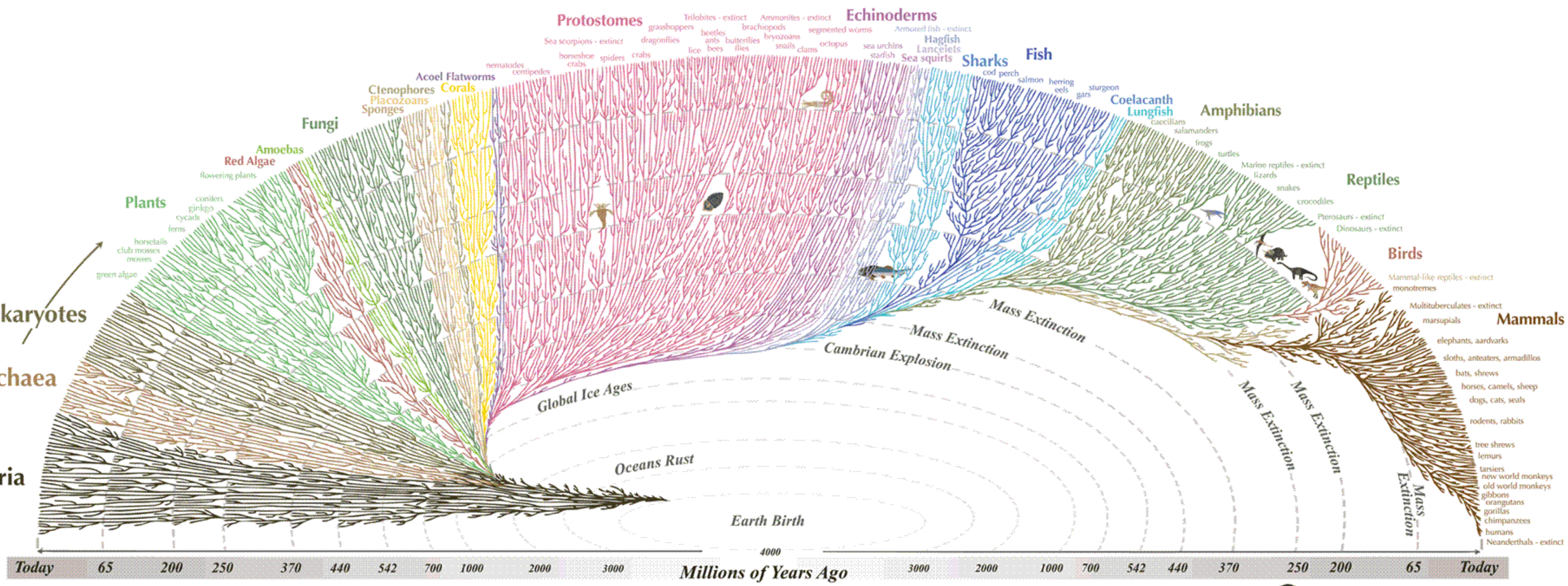
Example: Market Segmentation





http://www.esriro.ro/library/fliers/pdfs/tapestry_segmentation.pdf#page=2

Example: Phylogenetic Trees



All the major and many of the minor living branches of life are shown on this diagram, but only a few of those that have gone extinct are shown. Example: Dinosaurs - extinct

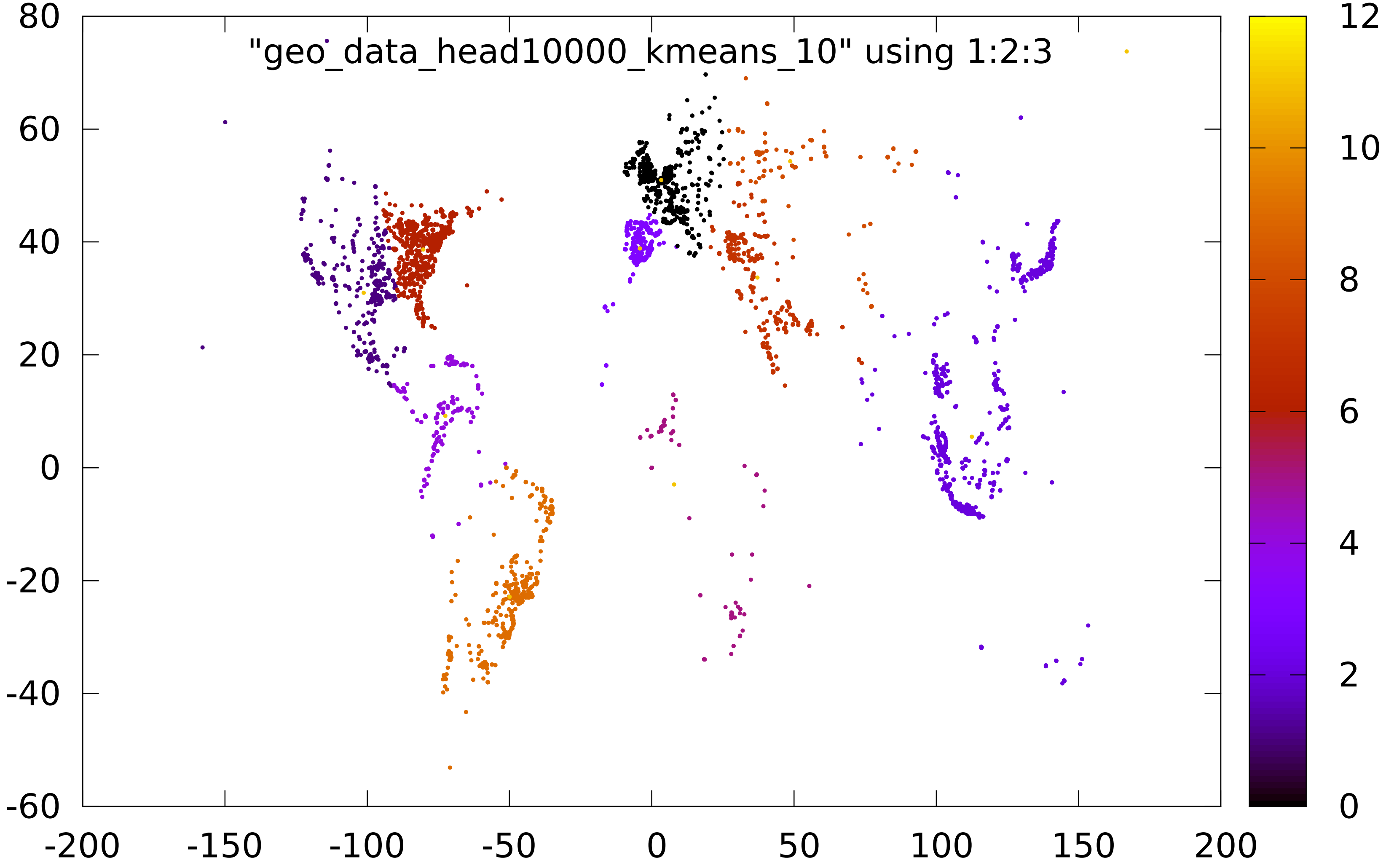
Other Examples

- Image segmentation
- Document clustering
- De-duplication …

Outline

- K-means Clustering
- Distance Metrics
- Using distance metrics for clustering
  - K-medoids
  - Hierarchical Clustering

How did we create 10 clusters?



Can compare apples vs oranges …

- … if they are in the same ***feature space***.
- $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ is a dataset
- Each $\boldsymbol{x}_i$ is assumed to be a point in some d-dimensional space
  - $\boldsymbol{x}_i = [x_{i1}, x_{i2}, \ldots, x_{id}]$
  - Each dimension represents a feature

K-means

- Partition a set of points $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ into k partitions $\boldsymbol{C} = \{C_1, C_2, \ldots, C_k\}$ that minimizes

$$RSS(\boldsymbol{C}) = \sum_{i=1}^{k} \sum_{j=1}^{n} a_{ij} \left\| \boldsymbol{x}_j - \boldsymbol{\mu}_i \right\|_2^2$$

a_{ij} is 1 if x_j is assigned to cluster C_i

Assignment Function

K-means

- Partition a set of points $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ into k partitions $\boldsymbol{C} = \{C_1, C_2, \ldots, C_k\}$ that minimizes

$$RSS(\boldsymbol{C}) = \sum_{i=1}^{k}\sum_{j=1}^{n} a_{ij}\left\|\boldsymbol{x}_j - \boldsymbol{\mu}_i\right\|_2^2$$

$$\boldsymbol{\mu}_i = [\mu_{i1}, \mu_{i2}, \ldots, \mu_{id}]$$

$$\mu_{ij} = \sum_{x \in C_i} \frac{x_j}{|C_i|}$$

μ_i is the mean of points in cluster C_i.

Cluster Representative

K-means

- Partition a set of points $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ into k partitions $\boldsymbol{C} = \{C_1, C_2, \ldots, C_k\}$ that minimizes

$$RSS(\boldsymbol{C}) = \sum_{i=1}^{k}\sum_{j=1}^{n} a_{ij}\left\|\boldsymbol{x}_j - \boldsymbol{\mu}_i\right\|_2^2$$

Square of the straight line distance between x_j and its center μ_i.

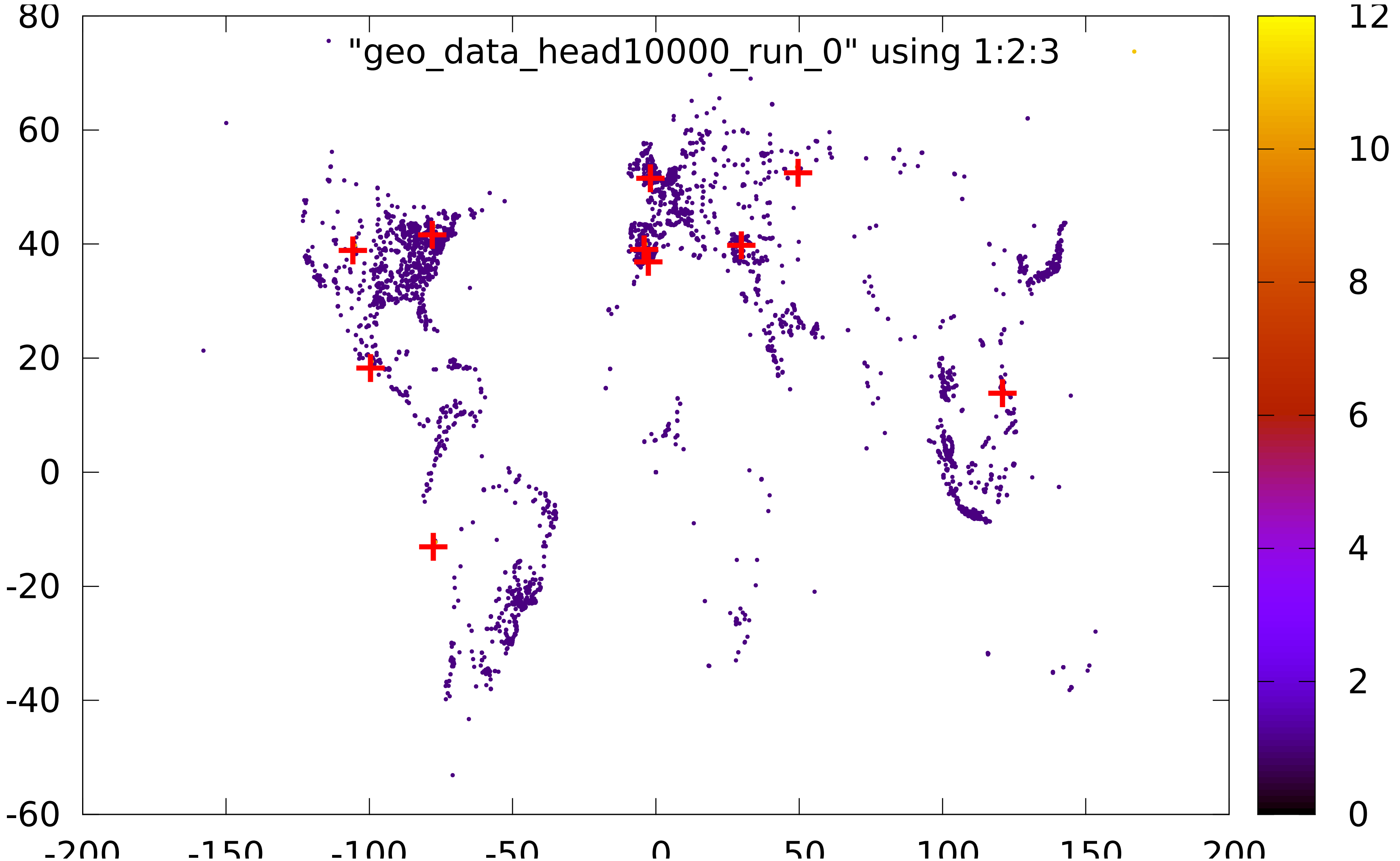
Chicken-and-Egg problem

- How do we minimize $RSS(\mathbf{C})$?
  - If we know the cluster representatives (or the means), then it is easy to find the assignment function (which minimizes $RSS(\mathbf{C})$)
    - *Assign point to the closest cluster representative*
  - If we know the assignment function, computing the cluster representatives is easy
    - *Compute the mean of the points in the cluster*

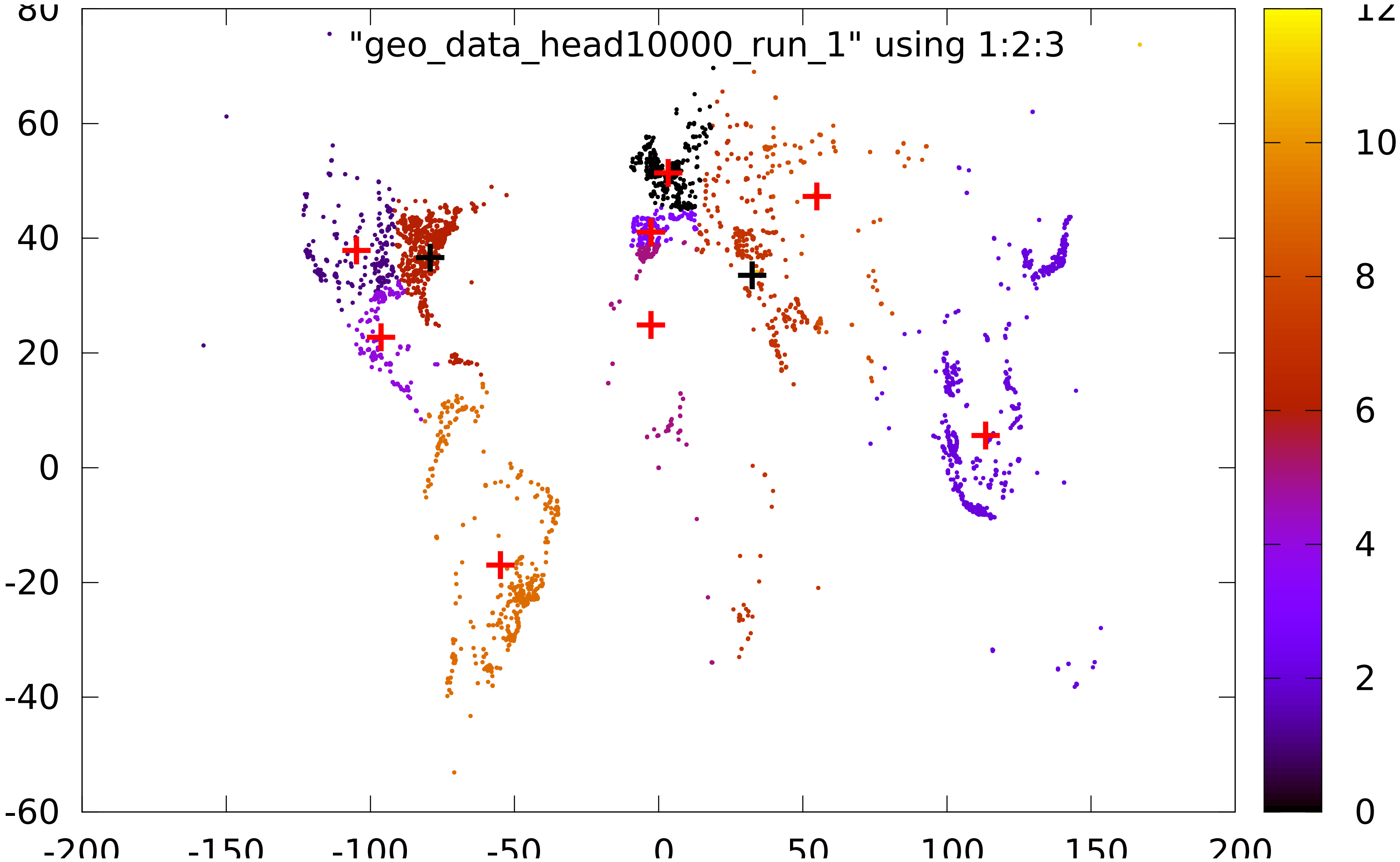
K-means Algorithm

- Idea: Alternate these two steps.
  - Pick some initialization for cluster representatives $\boldsymbol{\mu}^0$.
  - **E-step:**
    Assign points to the closest representative in $\boldsymbol{\mu}^i$.
  - **M-step:**
    Recompute the representatives $\boldsymbol{\mu}^{i+1}$ as means of the current clusters.

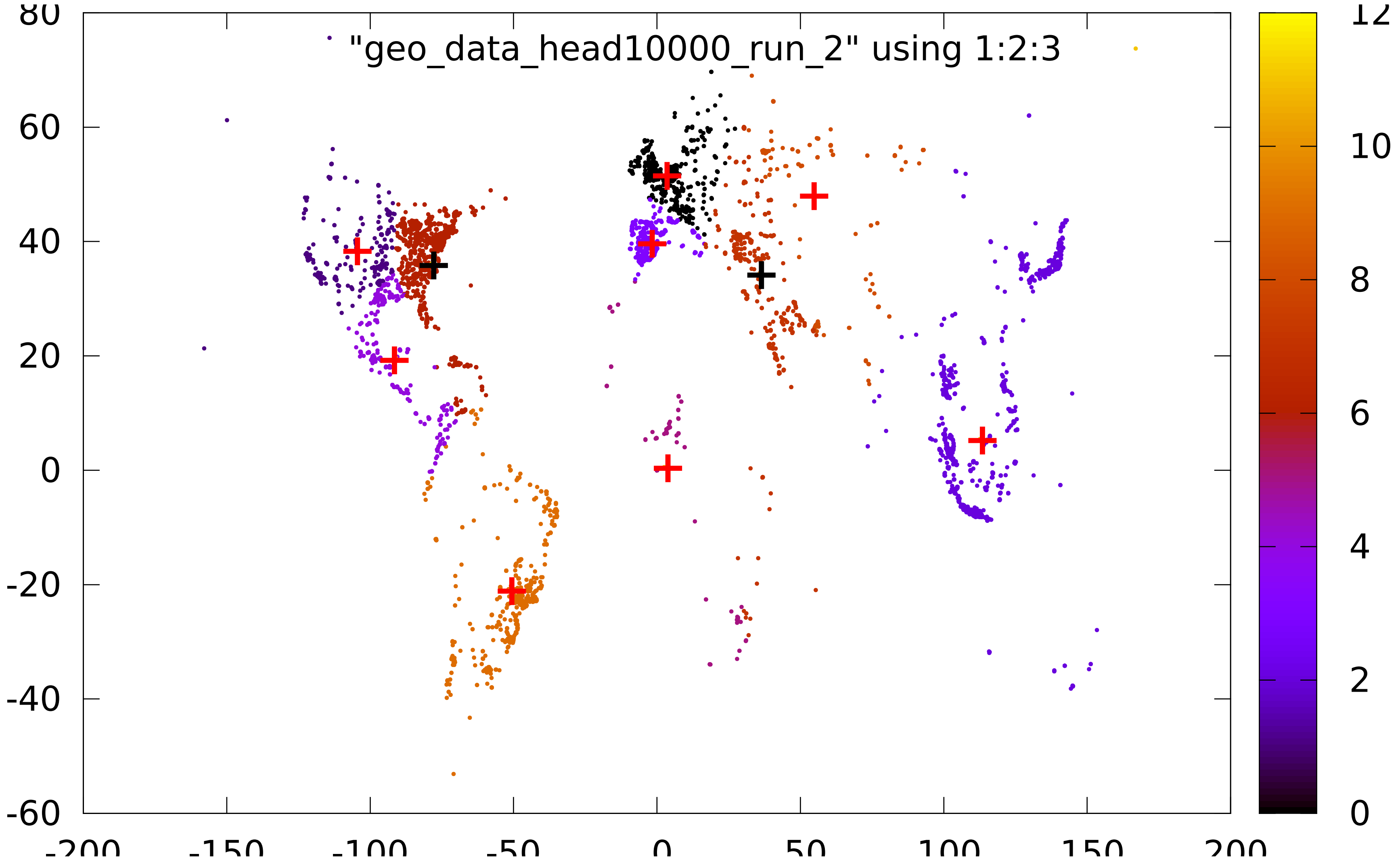
K-means: Initialization



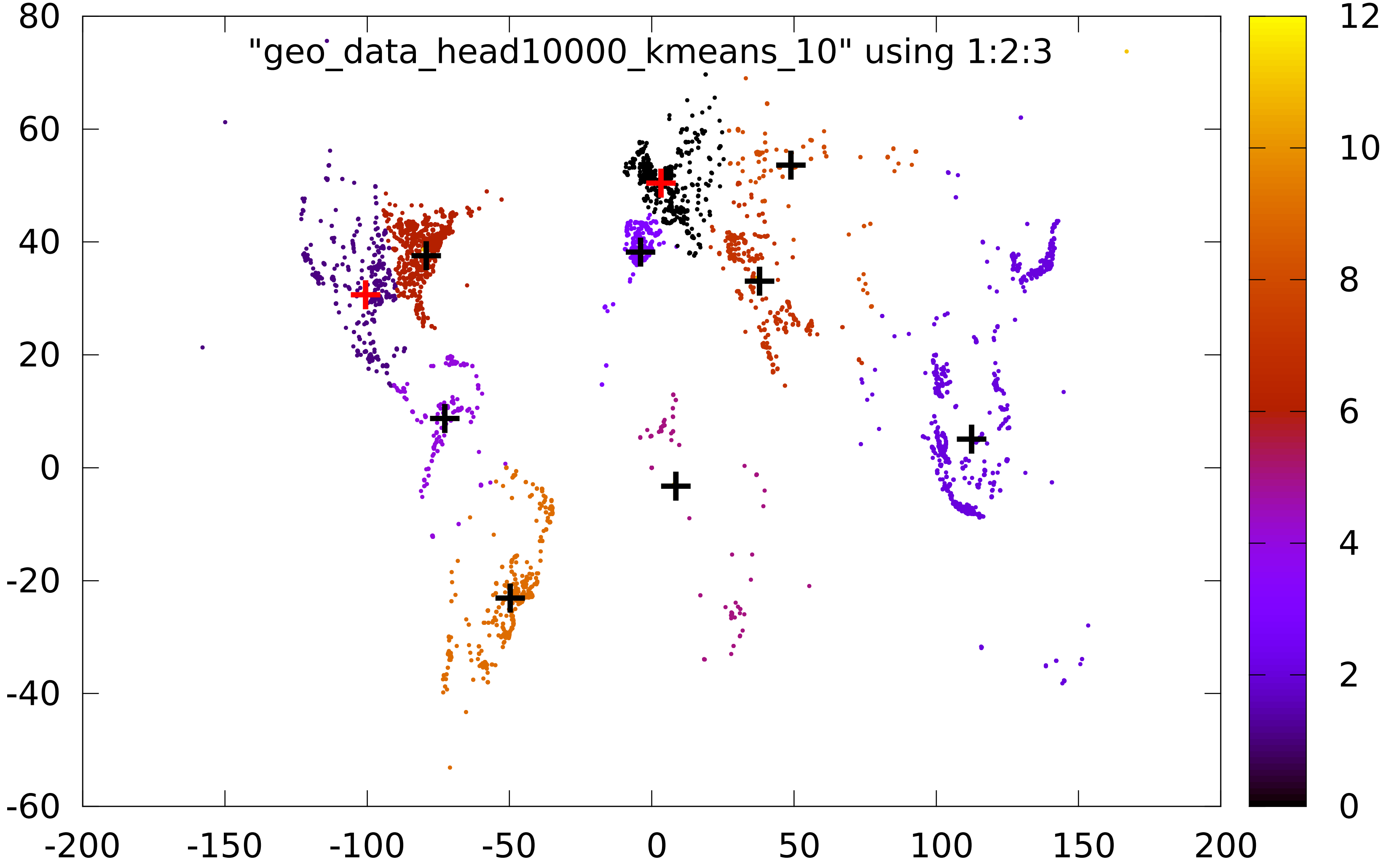
K-means: Iteration 1



K-means: Iteration 2



K-means: Iteration 10



Initialization

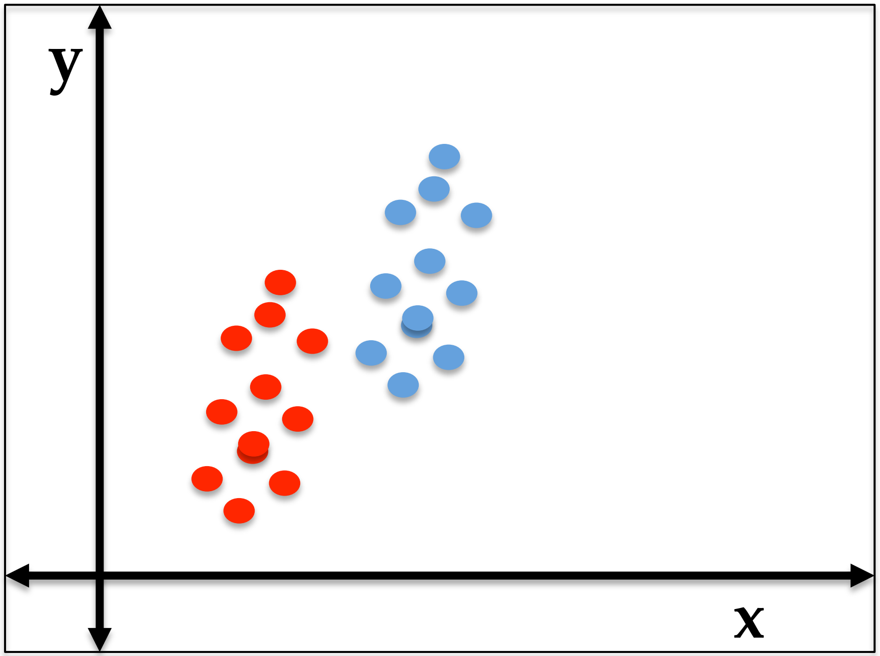
- Many heuristics
  - *Random*: K random points in the dataset
  - *Farthest First*:
    - Pick the first center at random
    - Pick the i^{th} center as the point "farthest away" from the last (i-1) centers
  - *K-means++*: (see paper)
    - Nice theoretical guarantees on quality of clustering

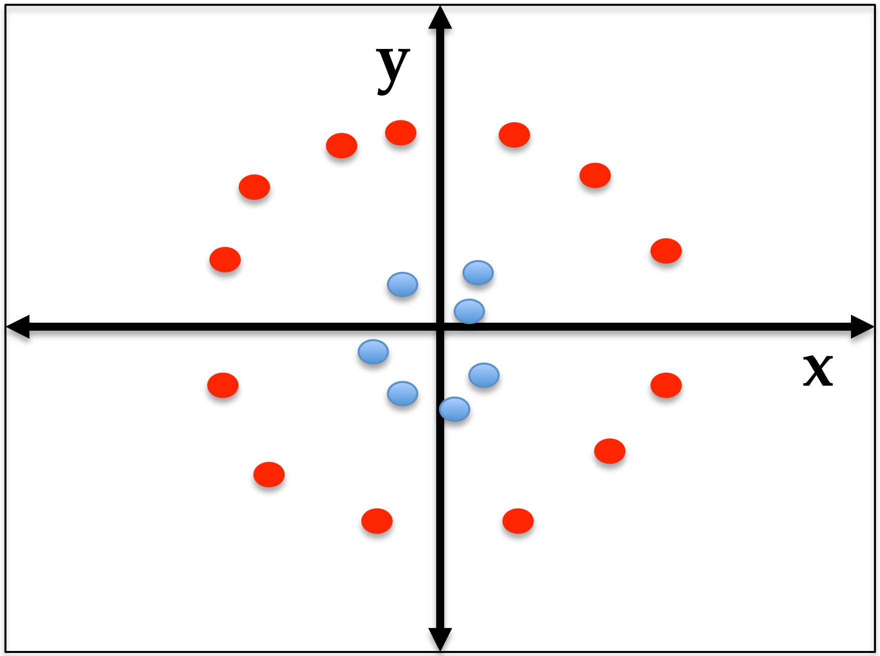
Stopping

- Alternate E and M steps till the cluster representatives do not change.

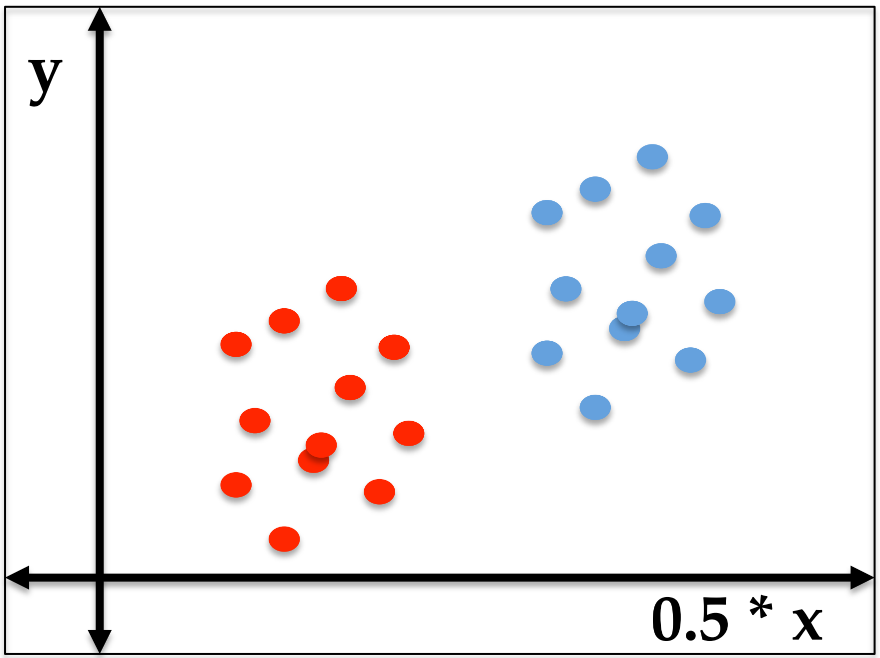
- Guaranteed to converge
  - To a **local optima** …
  - … but **not** necessarily to a **global optima**
    - *Finding the optimal solution (with least RSS(**C**)) is NP-hard, even for 2 clusters.*

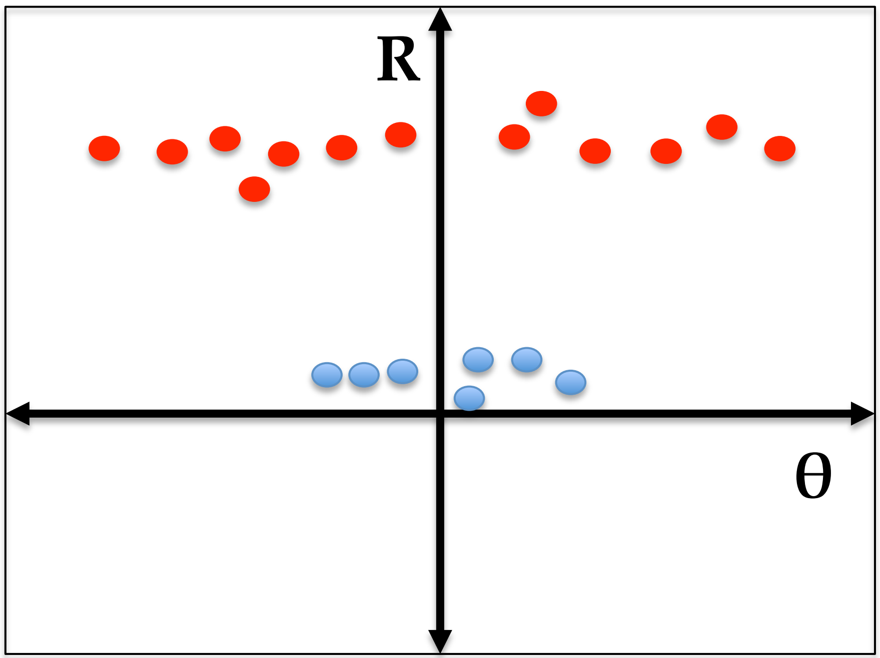
Where k-means fails …





Scaling / changing features can help





Limitations of k-means

- Scaling/changing the feature space can change the solution.
- Cluster points into spherical regions.

- Number of clusters should be known apriori

Outline

- K-means Clustering
- Distance Metrics
- Using distance metrics for clustering
  - K-medoids
  - Hierarchical Clustering

Distance Metrics

- Function d that maps pairs of points $\boldsymbol{x}$, $\boldsymbol{y}$ to real numbers (usually between 0, 1)

- *Symmetric:* $d(\boldsymbol{x},\boldsymbol{y}) = d(\boldsymbol{y},\boldsymbol{x})$
- *Triangle Inequality:* $d(\boldsymbol{x},\boldsymbol{y}) + d(\boldsymbol{y},\boldsymbol{z}) \geq d(\boldsymbol{x},\boldsymbol{z})$

- Choice of distance metric is usually application dependent

Euclidean Distance

$$\|\boldsymbol{x} - \boldsymbol{y}\|_2 = \sqrt{\left(\sum_i (x_i - y_i)^2\right)}$$

- Straight line distance between two points $\mathbf{x} = [x_1, x_2, \ldots, x_d]$ and $\mathbf{y} = [y_1, y_2, \ldots, y_d]$

- *K-means minimizes the sum of the Euclidean distances between the points and the centers*
  - *We use the mean as a center*

Minkowski (L_p) Distance

$$L_p = \left(\sum_i |x_i - y_i|^p\right)^{1/p}$$

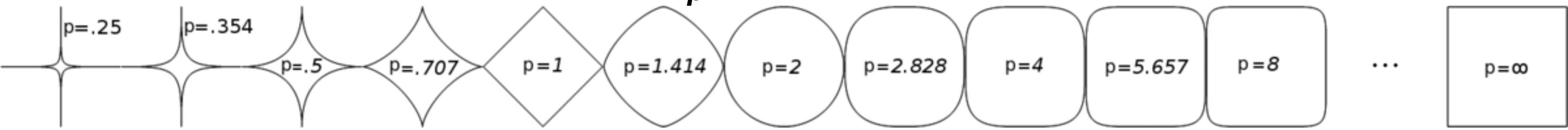
- L_2 = ?

Minkowski (L_p) Distance

$$L_p = \left(\sum_i |x_i - y_i|^p\right)^{1/p}$$

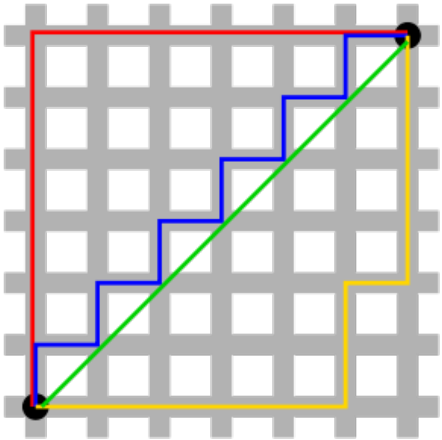
- L_2 = Euclidean
- L_1 = ?

Minkowski (L_p) Distance



$$L_p = \left(\sum_i |x_i - y_i|^p\right)^{1/p}$$

- L_1 = city block / Manhattan
- L_∞ = ?



Vector-based Similarities

- Cosine Similarity (inverse of a distance)

$$\frac{\sum_i x_i \cdot y_i}{\sqrt{\sum_i {x_i}^2}\sqrt{\sum_i {y_i}^2}}$$

Dot Product

L2 Norm

  – *can be used in conjunction with TFIDF scores*

Vector-based Similarities

- Pearson's Correlation Coefficient
  - *cosine similarity on mean normalized vectors*

Mean of x_i's

Dot Product

$$\frac{\sum_i (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2} \sqrt{\sum_i (y_i - \bar{y})^2}}$$

L2 Norm

Set-based Distances

- Let A and B be two sets.

$$\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

  – *Again, a measure of similarity (inverse of distance)*

Scaling / Changing features …

- … can be thought of as using a different distance function.

- How do we cluster for general distance functions?

Outline

- K-means Clustering
- Distance Metrics
- Using distance metrics for clustering
  - K-medoids
  - Hierarchical Clustering

K-means for general distance functions?

- Mean of a set of points does not always make sense.
  - *Mean of a set of movies or a set of documents?*

- Mean m of a set of points P minimizes the sum of Euclidean distances between m and every point in P
  - *Best cluster representative under Euclidean Distance*

  - The above is not true for a general distance metric.

K-medoids

- Allows a general distance metric $d(x,y)$.

- Same algorithm as K-means …
- … but we don't pick the new centers using mean of the cluster.

K-medoids

- Pick some initialization for cluster representatives $\boldsymbol{\mu}^0$.
- **E-step:**
  Assign points to the closest representative in $\boldsymbol{\mu}^i$.
- **M-step:**
  Recompute the representatives $\boldsymbol{\mu}^{i+1}$ as the *medoid*, or **one of the points in the cluster with the minimum distance from all the other points.**

Medoid

- m is the medoid of a set of points P if

$$m = \underset{x \in P}{\operatorname{argmin}} \left(\sum_{y \in P} d(x, y) \right)$$

Point that minimizes the sum of distances to all other points in the set.

Computing the medoid

$$m = \underset{x \in P}{\operatorname{argmin}} \left(\sum_{y \in P} d(x, y) \right)$$

- Need to compute all $|P|^2$ distances.

- In comparison, computing the mean in k-means only requires computing d averages involving $|P|$ numbers each.

K-medoids summary

- Same algorithm as K-means, but uses medoids instead of means
- Centers are always points that appear in the original dataset
- Can use any distance measure for clustering.
- *Still need to know the number of clusters a priori…*

Outline

- K-means Clustering
- Distance Metrics
- Using distance metrics for clustering
  - K-mediods
  - Hierarchical Clustering

Hierarchical Clustering

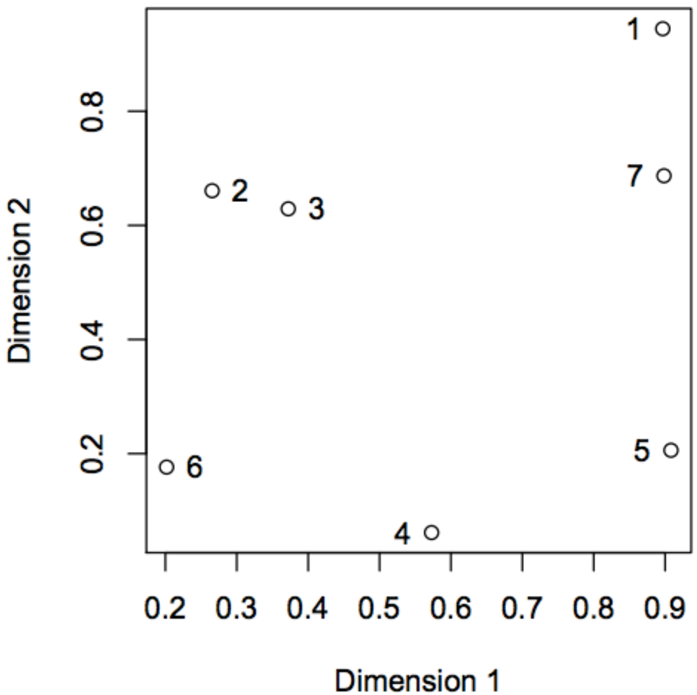
- Rather than compute a single clustering, compute a family of clusterings.
- Can choose the clusters *a posteriori*.

Agglomerative Clustering

- Initialize each point to its own cluster

- Repeat:
  - Pick the two clusters that are *closest*
  - Merge them into one cluster
  - Stop when there is only one cluster left

Example



Step 1: {1} {2} {3} {4} {5} {6} {7}

Step 2: {1} {2, 3} {4} {5} {6} {7}

Step 3: {1, 7} {2, 3} {4} {5} {6}

Step 4: {1, 7} {2, 3} {4, 5} {6}

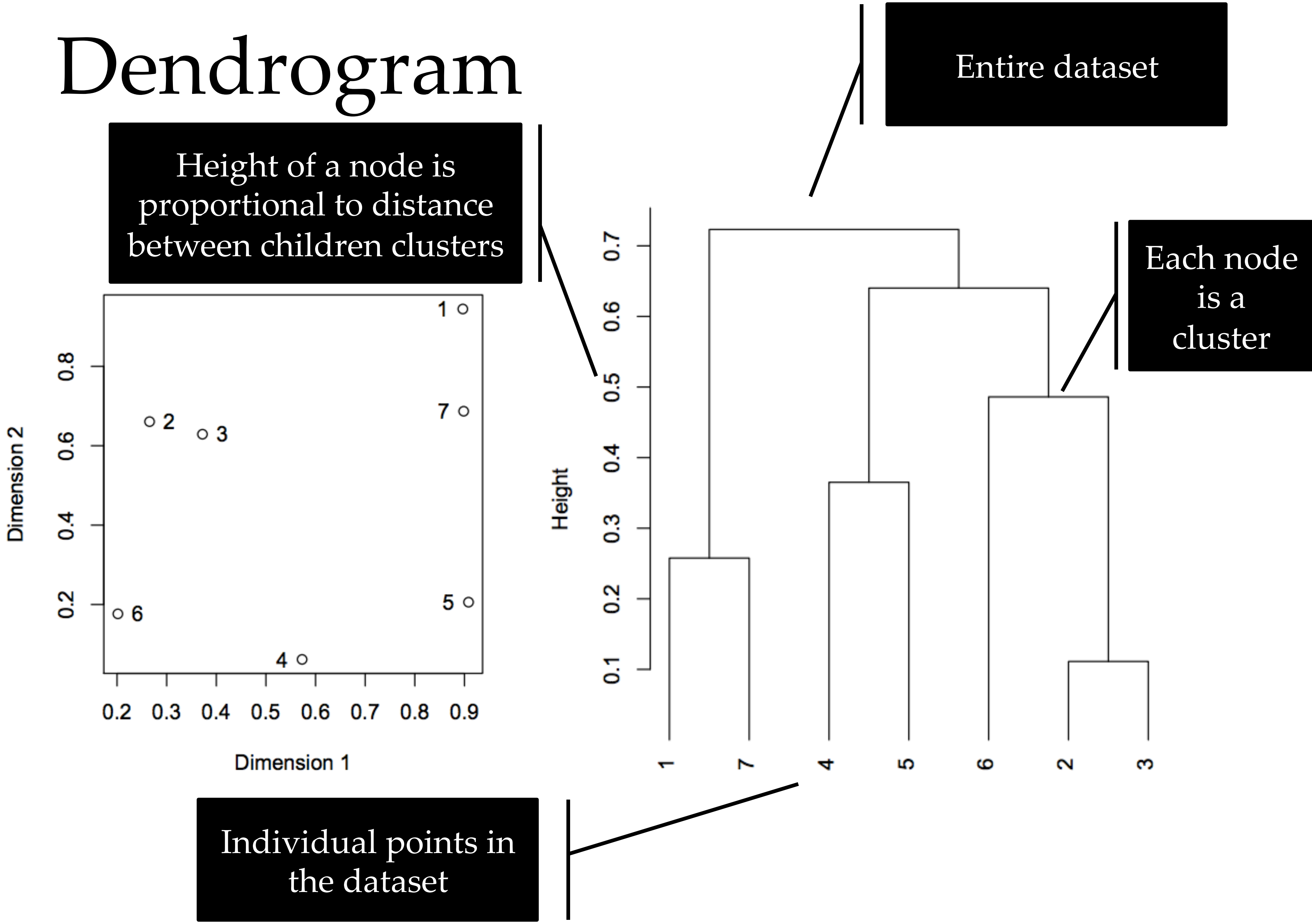
Step 5: {1, 7} {2, 3, 6} {4, 5}

Step 6: {1, 7} {2, 3, 4, 5, 6}

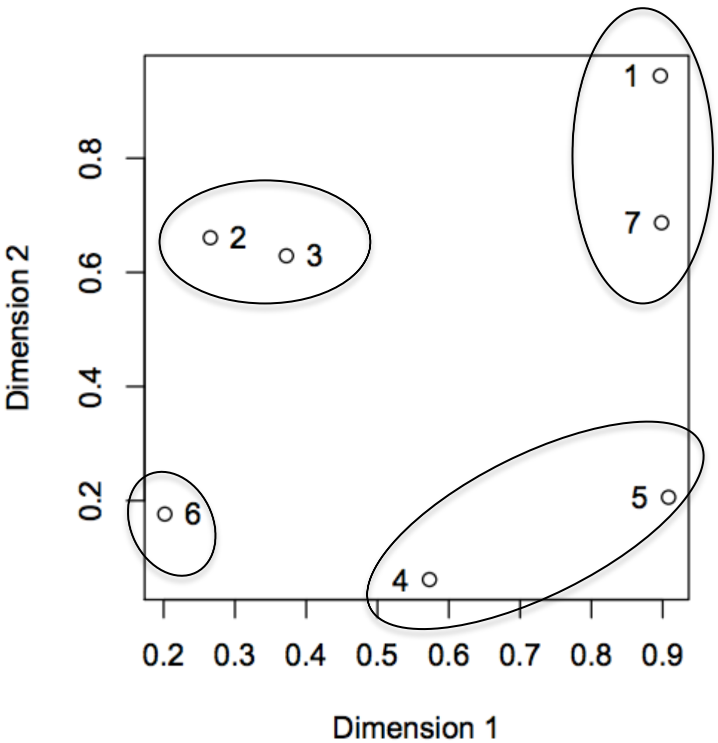
Step 7: {1, 2, 3, 4, 5, 6, 7}

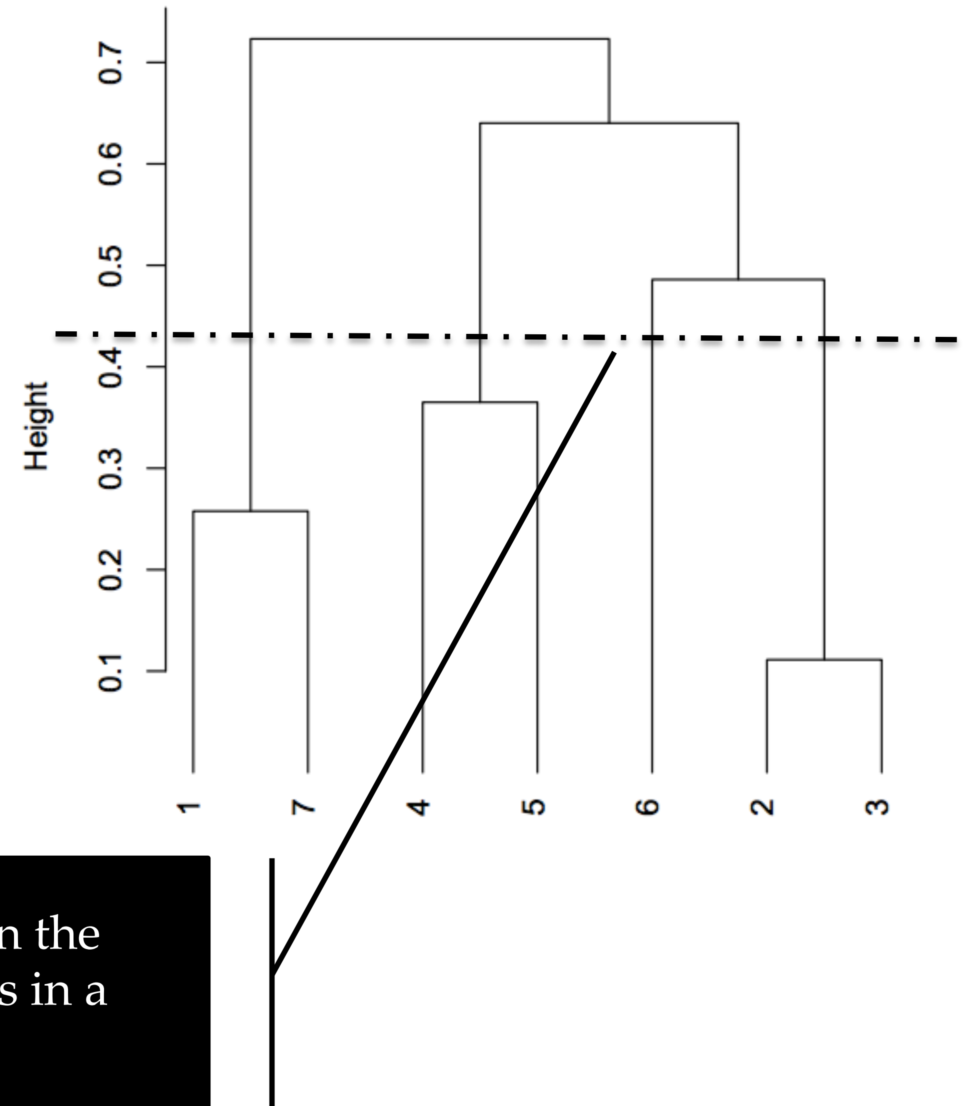
Example based on Ryan Tibshirani's slides

Dendrogram


Height of a node is proportional to distance between children clusters
Entire dataset
Each node is a cluster
Individual points in the dataset
Dimension 1
Dimension 2
0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
0.2 0.4 0.6 0.8
1
2
3
4
5
6
7
Height
0.1 0.2 0.3 0.4 0.5 0.6 0.7
1 7 4 5 6 2 3

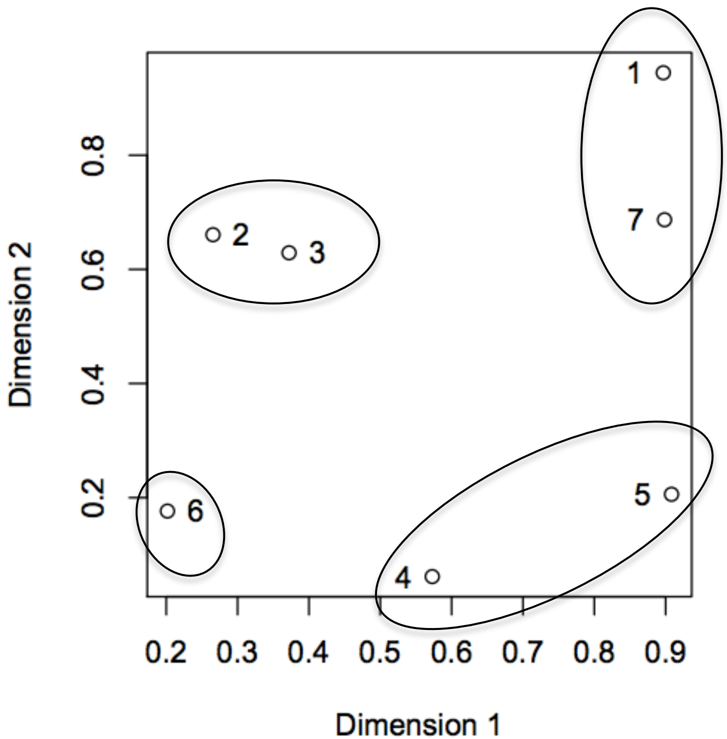
Dendrogram





A horizontal cut in the dendogram results in a clustering

Distance between clusters



Step 1: {1} {2} {3} {4} {5} {6} {7}

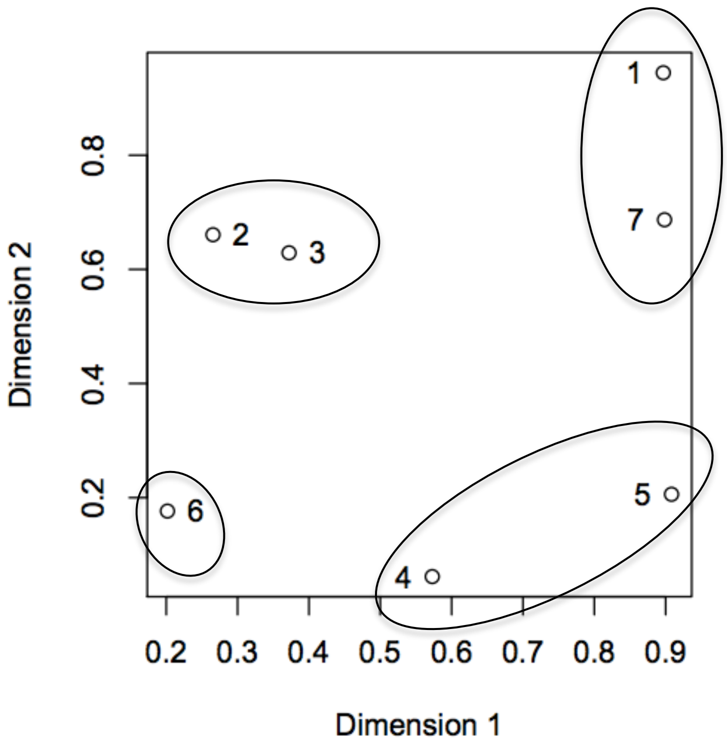
Step 2: {1} {2, 3} {4} {5} {6} {7}

Step 3: {1, 7} {2, 3} {4} {5} {6}

Step 4: {1, 7} {2, 3} {4, 5} {6}

What are the next two closest clusters?

Single Linkage

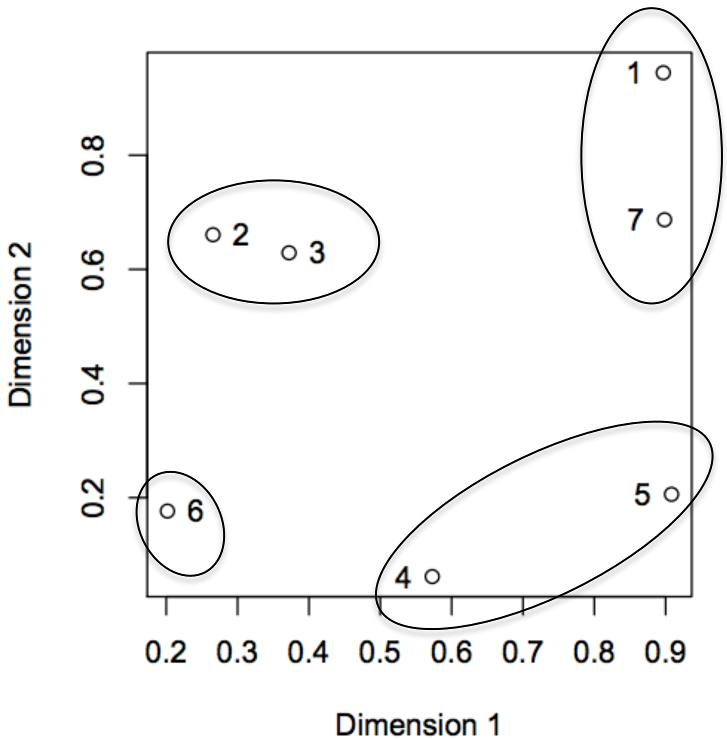


$$d_{single}(C_1, C_2) = \min_{x \in C_1, y \in C_2} d(x, y)$$

Distance between two clusters is the distance between the two ***closest*** *points in the clusters.*

{6} is closer to {4,5} than {2,3} according to single linkage

Complete Linkage

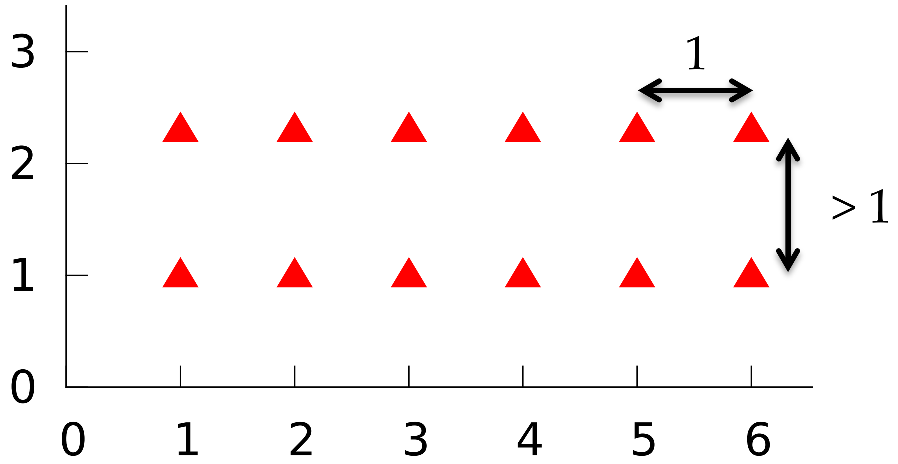


$$d_{complete}(C_1, C_2) = \max_{x \in C_1, y \in C_2} d(x, y)$$

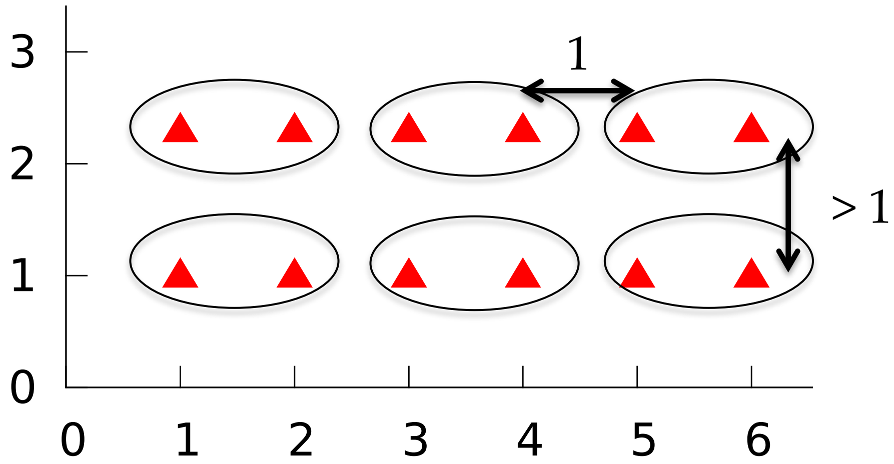
*Distance between two clusters is the distance between the two **farthest** points in the clusters.*

{6} is closer to {2,3} than {4,5} according to complete linkage

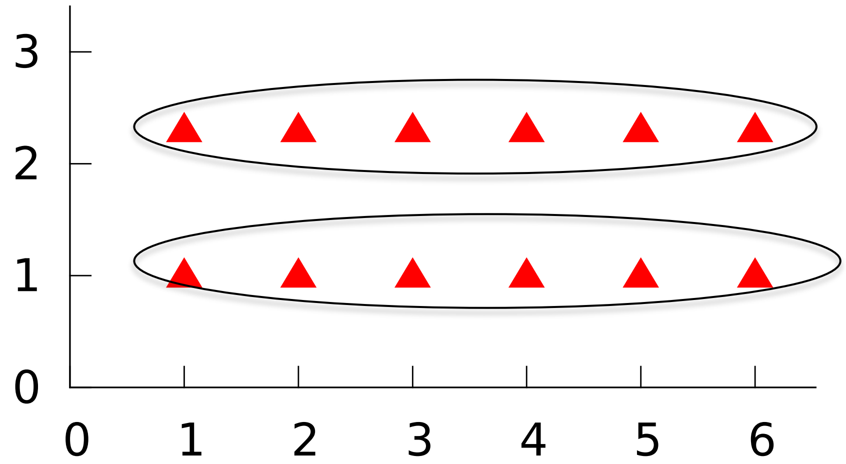
Single vs Complete Linkage



Single Linkage

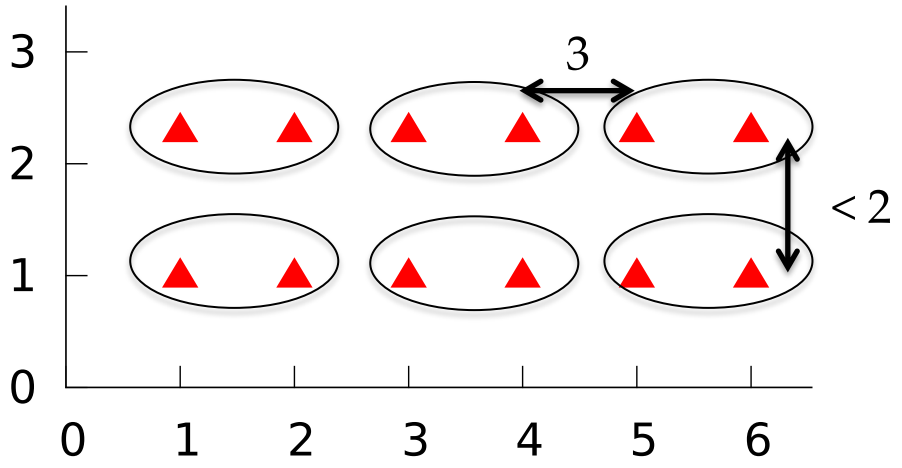


Single Linkage

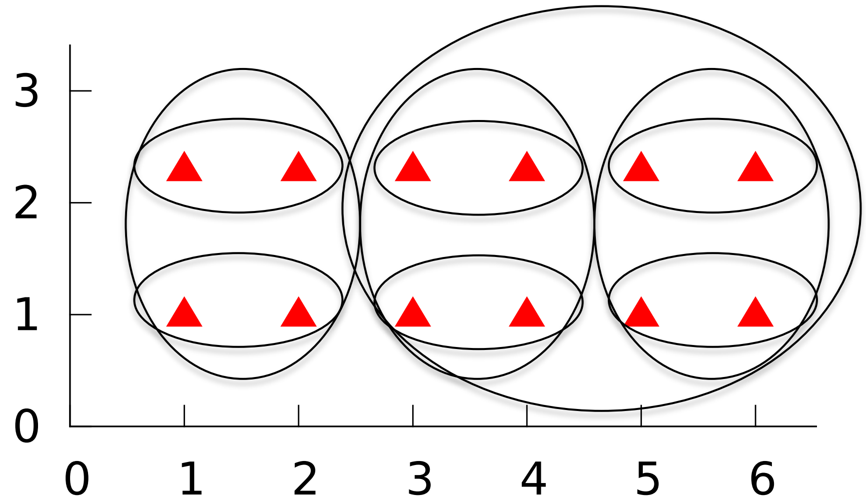


Chaining: Single linkage can result in clusters that are spread out and not compact

Complete Linkage

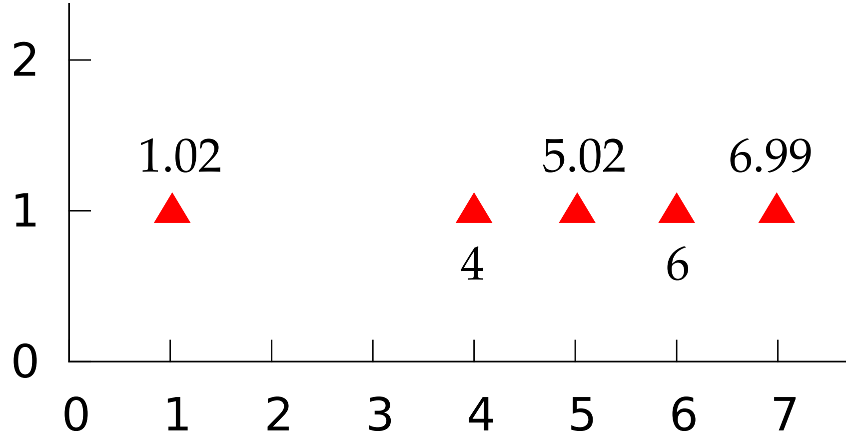


Complete Linkage

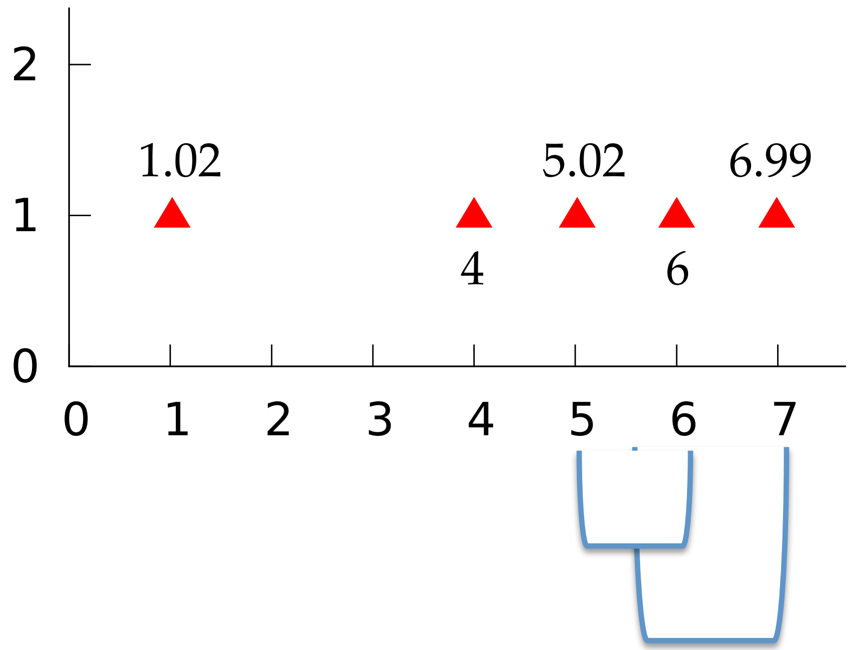


Complete linkage returns more compact clusters in this case.

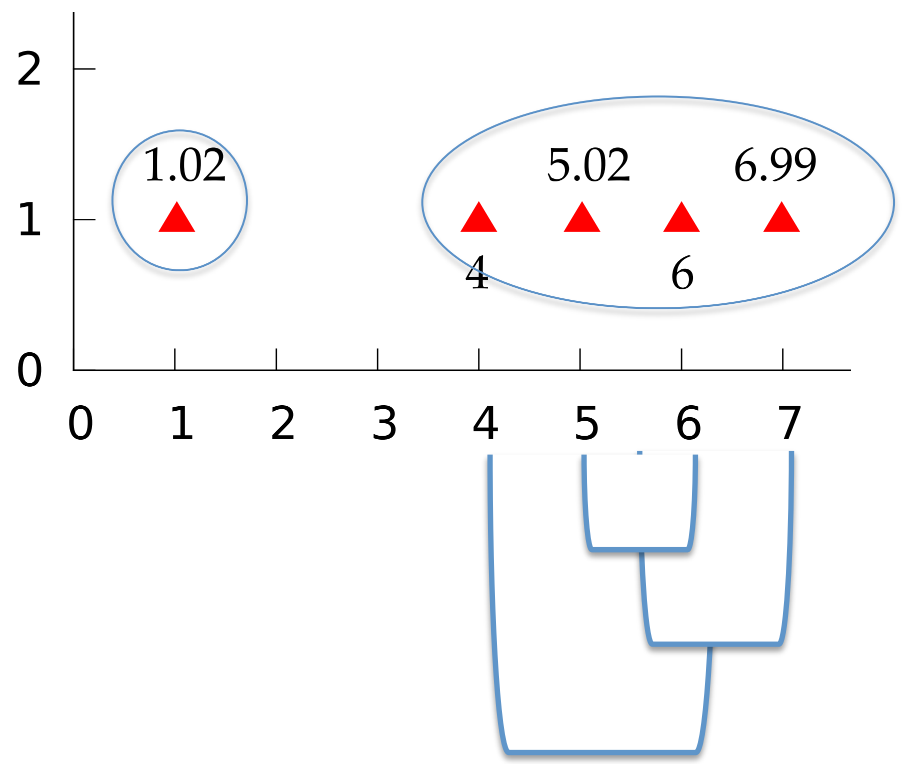
Single vs Complete Linkage



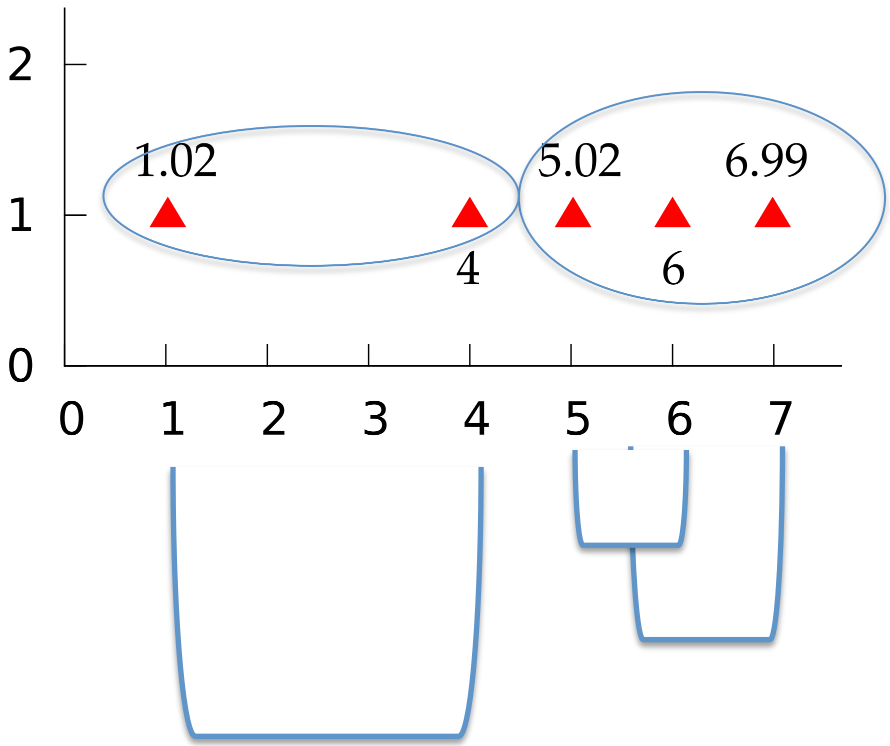
In both cases …



Single Linkage



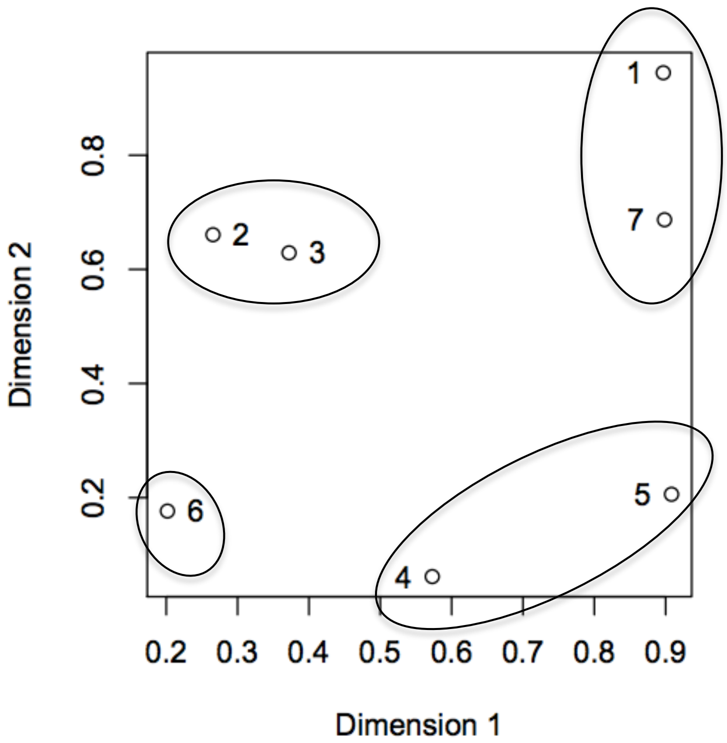
Complete Linkage



Complete Linkage is ***sensitive to outliers.***

Average Linkage

$$d_{avg}(C_1, C_2) = \frac{\sum_{x \epsilon C_1, y \epsilon C_2} d(x, y)}{|C_1| \cdot |C_2|}$$



*Distance between two clusters is the **average** distance between every pair of points in the clusters.*

Hierarchical Clustering summary

- Create a family of hierarchical clusterings
  - Visualized using a dendrograms
  - Users can choose number of clusters *after* clustering is done.
- Can use any distance function
- Different choices for measuring distance between clusters.