CS 696 Intro to Big Data: Tools and Methods
Fall Semester, 2016
Doc 15 Clustering2
Oct 13, 2016

Picking Number of Clusters

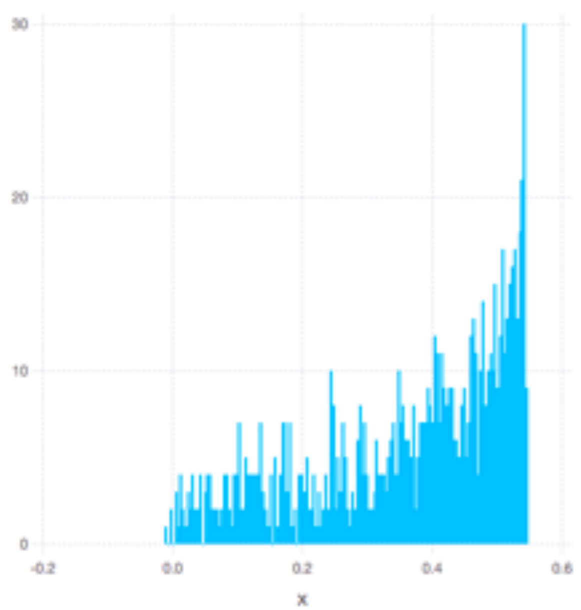
Silhouette coefficient

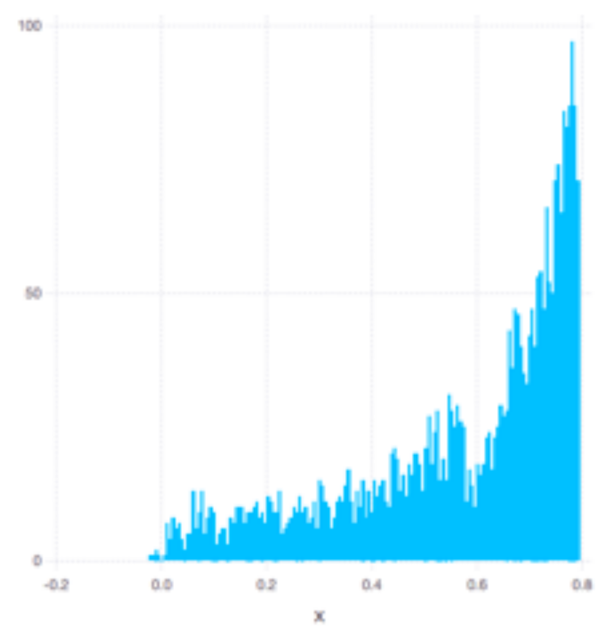
Davies-Bouldin Index

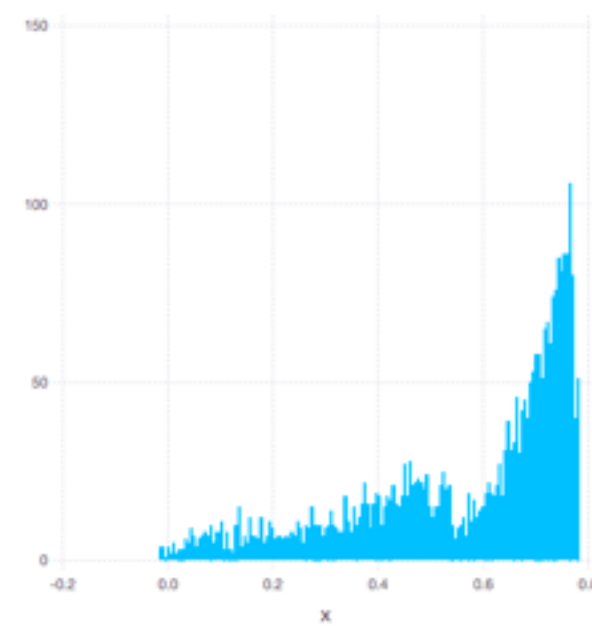
Dunn Index

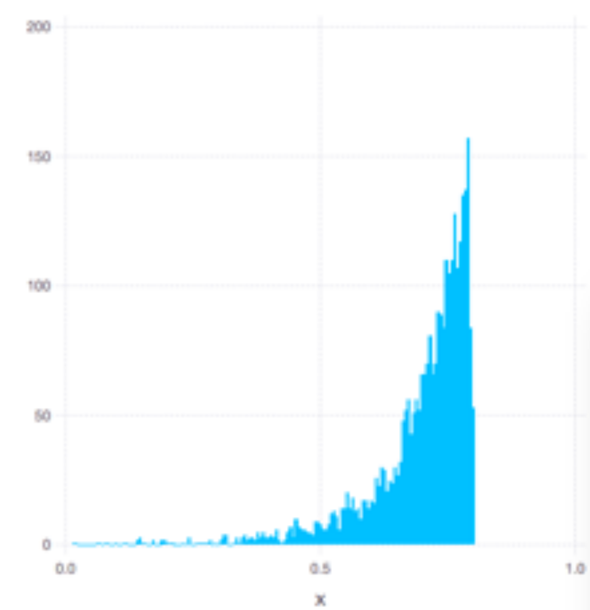
Silhouettes

High values indicate









Dunn Index

$$DI_m = \frac{\min\limits_{1 \le i < j \le k} \delta(C_i, C_j)}{\max\limits_{1 \le m \le k} S_m}$$

δ(Ci,Cj) distance between centers of cluster i & j

Sm size of the cluster

- Maximum distance between two points in cluster
- Mean distance between points in the cluster
- Mean distance between all points from center

Larger values are better

One spread out cluster can produce low value

Apache Mahout implements Dunn Index

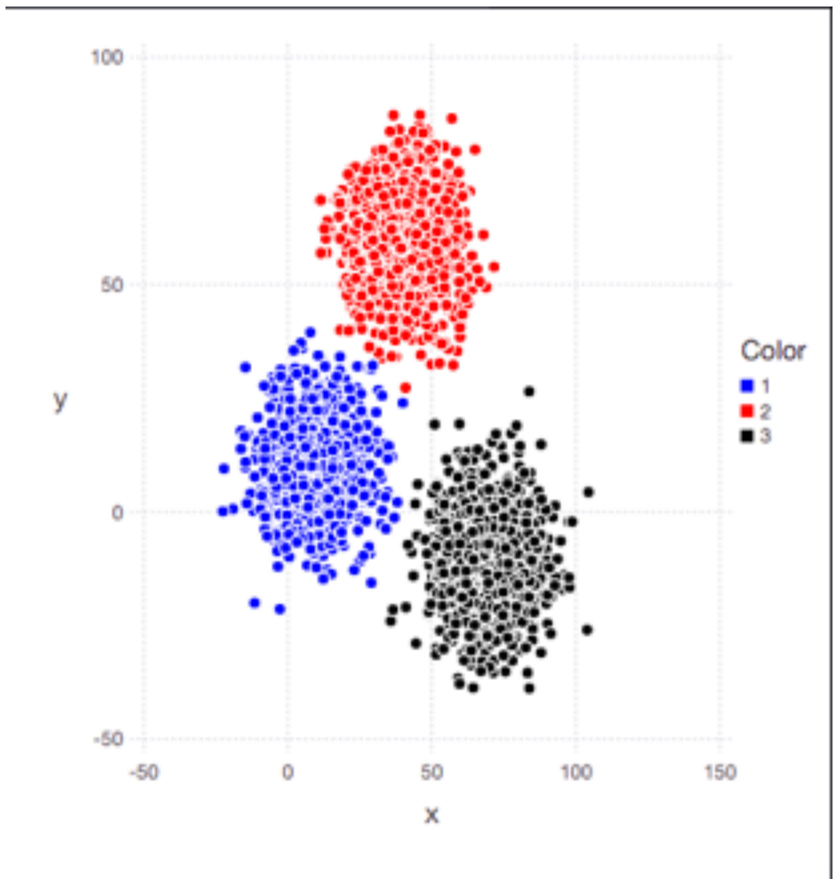
Dunn Index

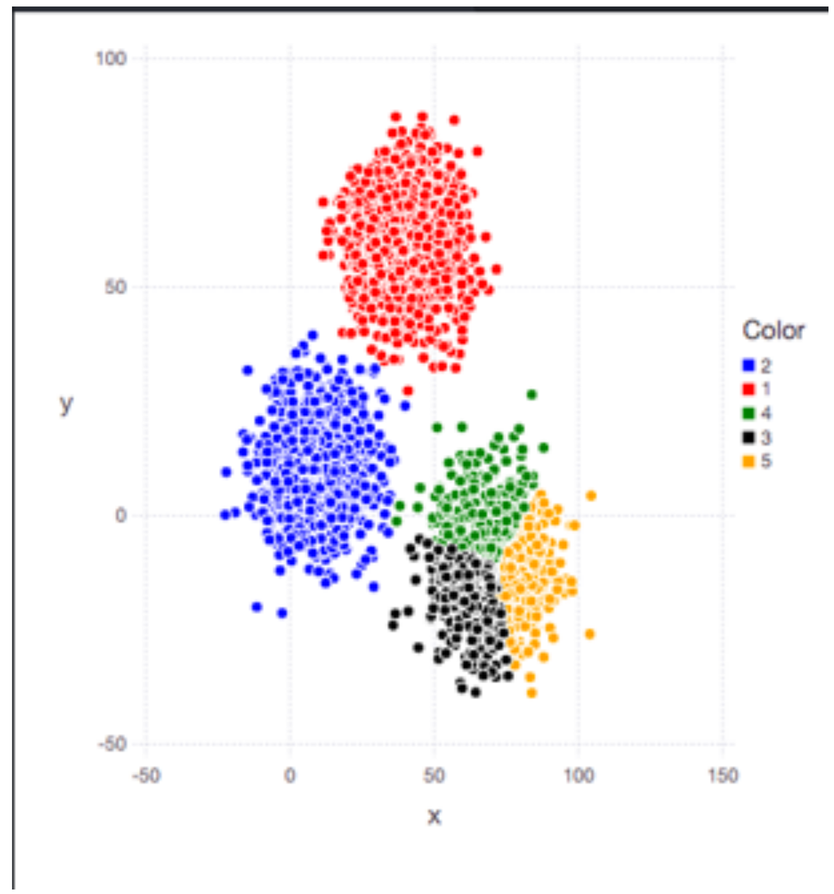
$$DI_m = \frac{\min\limits_{1 \le i < j \le k} \delta\left(C_i, C_j\right)}{\max\limits_{1 \le m \le k} S_m}$$

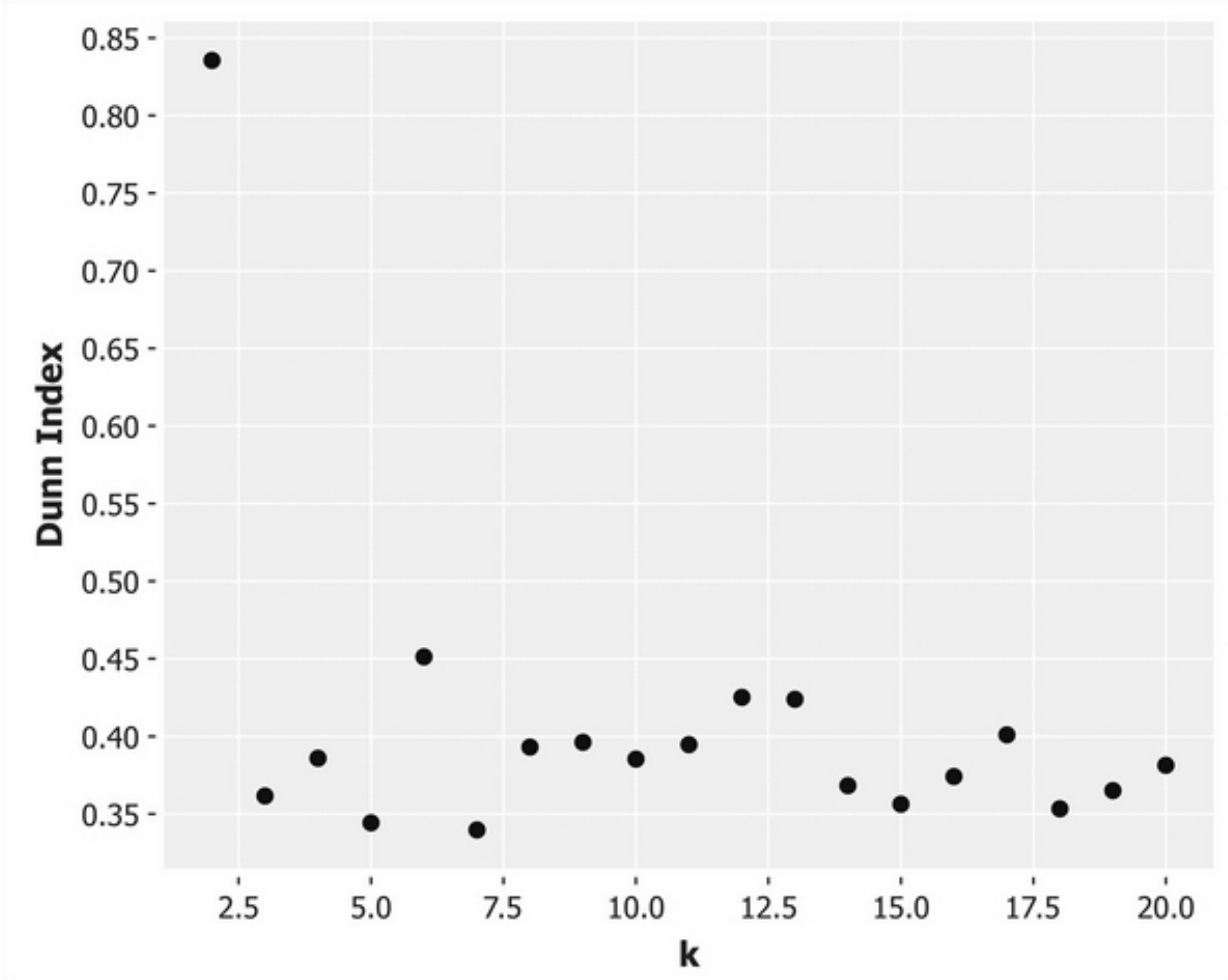
$\delta(C_i,C_j)$ distance between centers of cluster i & j

S_m size of the cluster

- Maximum distance between two points in cluster
- Mean distance between points in the cluster
- Mean distance between all points from center







Same data using k-means cluster with k = 2 to 20

Davies-Bouldin index

$$D_i = \max_{i \neq j} \frac{S_i + S_j}{\delta(C_i, C_j)}$$

$$\mathrm{DB} = \frac{1}{n} \sum_{i=1}^{n} D_i$$

δ(Ci,Cj) distance between centers of cluster i & j

Sm size of the cluster

- Maximum distance between two points in cluster
- Mean distance between points in the cluster
- Mean distance between all points from center

Lower values are better

Davies-Bouldin index

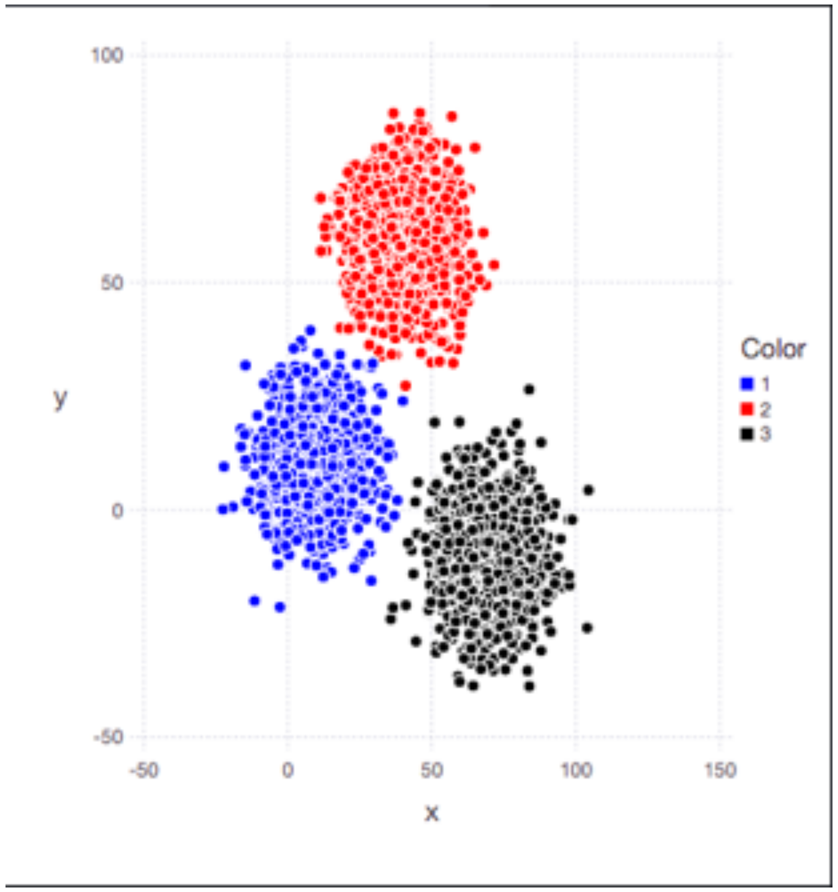
$$D_i = \max_{i \neq j} \frac{S_i + S_j}{\delta\left(C_i, C_j\right)}$$

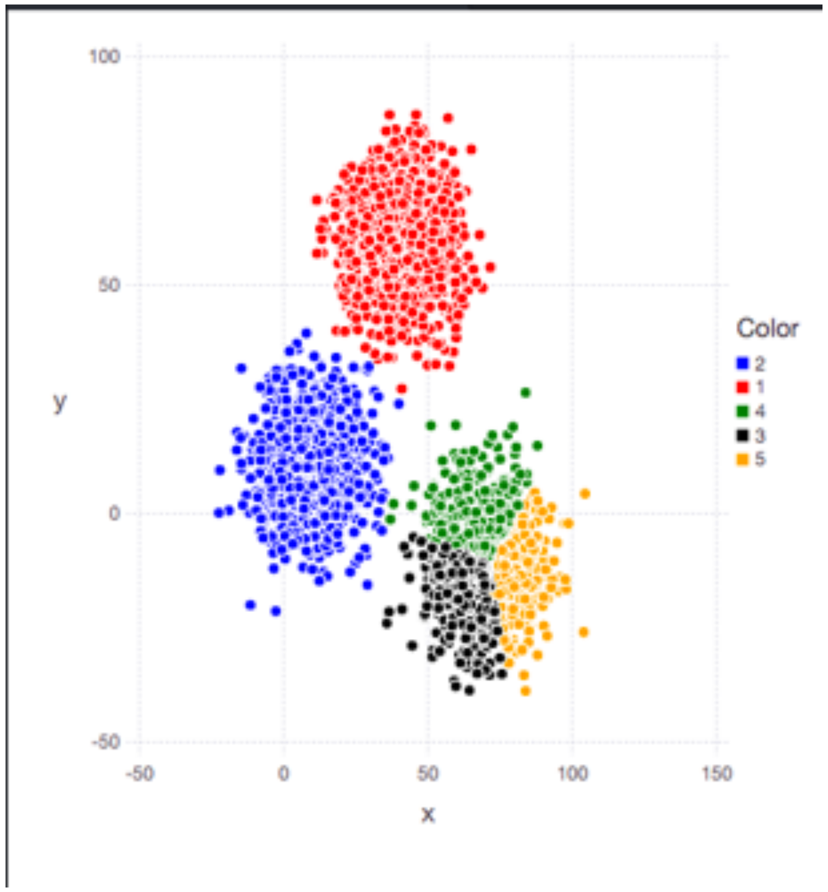
$$\mathrm{DB} = \frac{1}{n} \sum_{i=1}^{n} D_i$$

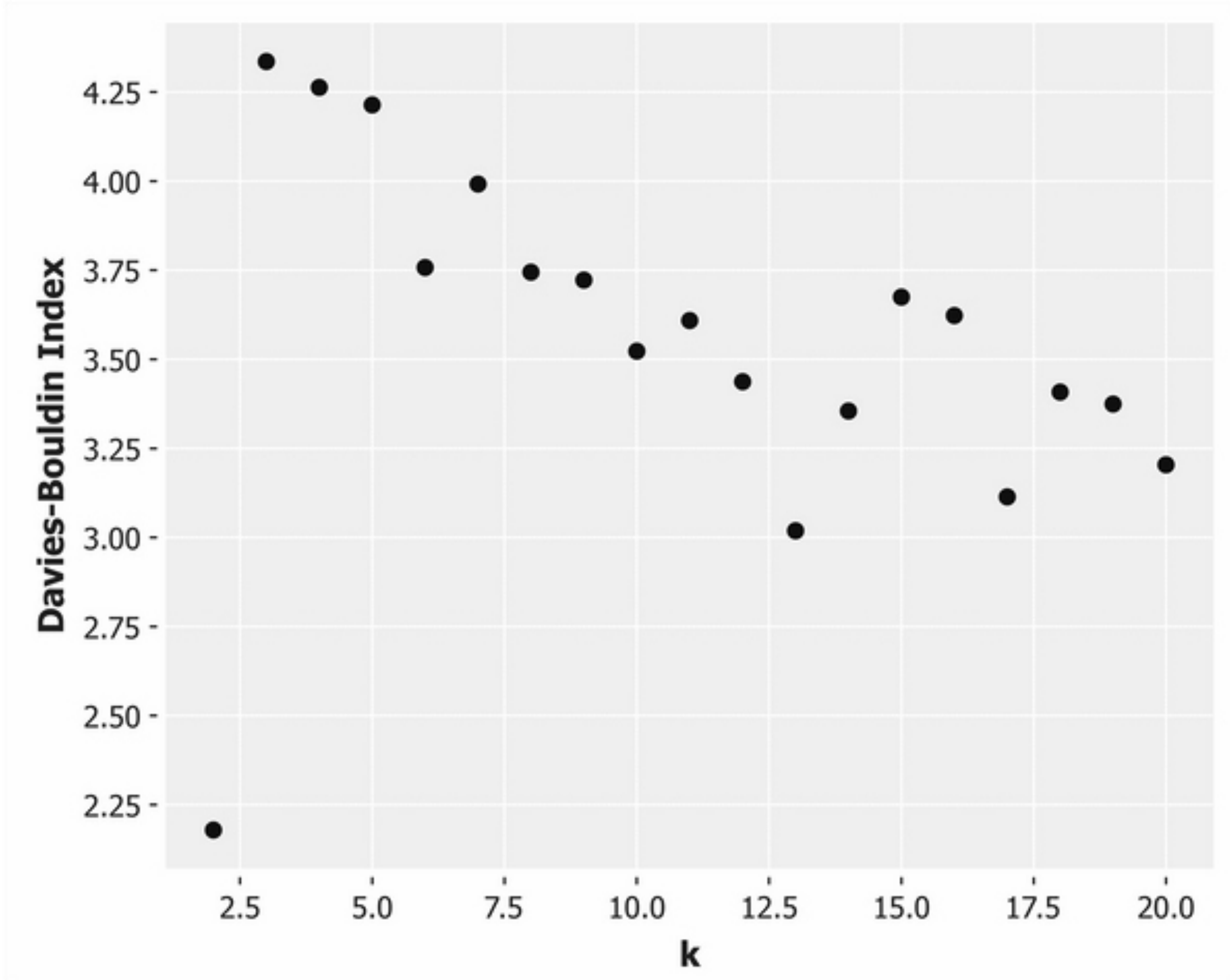
δ(Ci,Cj) distance between centers of cluster i & j

Sm size of the cluster

- Maximum distance between two points in cluster
- Mean distance between points in the cluster
- Mean distance between all points from center







Same data using k-means cluster with k = 2 to 20

DBSCAN

Density-based spatial clustering of applications with noise

Groups points together that are closely packed together

Developed in 1996

One of most commonly used clustering algorithms

Most cited in scientific literature

Terms

Parameters ϵ- distance

minPts

p is a core point if

There are minPts within distance ϵ of p including p

Directly reachable points

All points within distance ϵ of a core point p are directly reachable from p

q is reachable from p if

There is a path $p_1, ..., p_n$ with $p_1 = p$ and $p_n = q$,

p_{i+1} is directly reachable from p_i

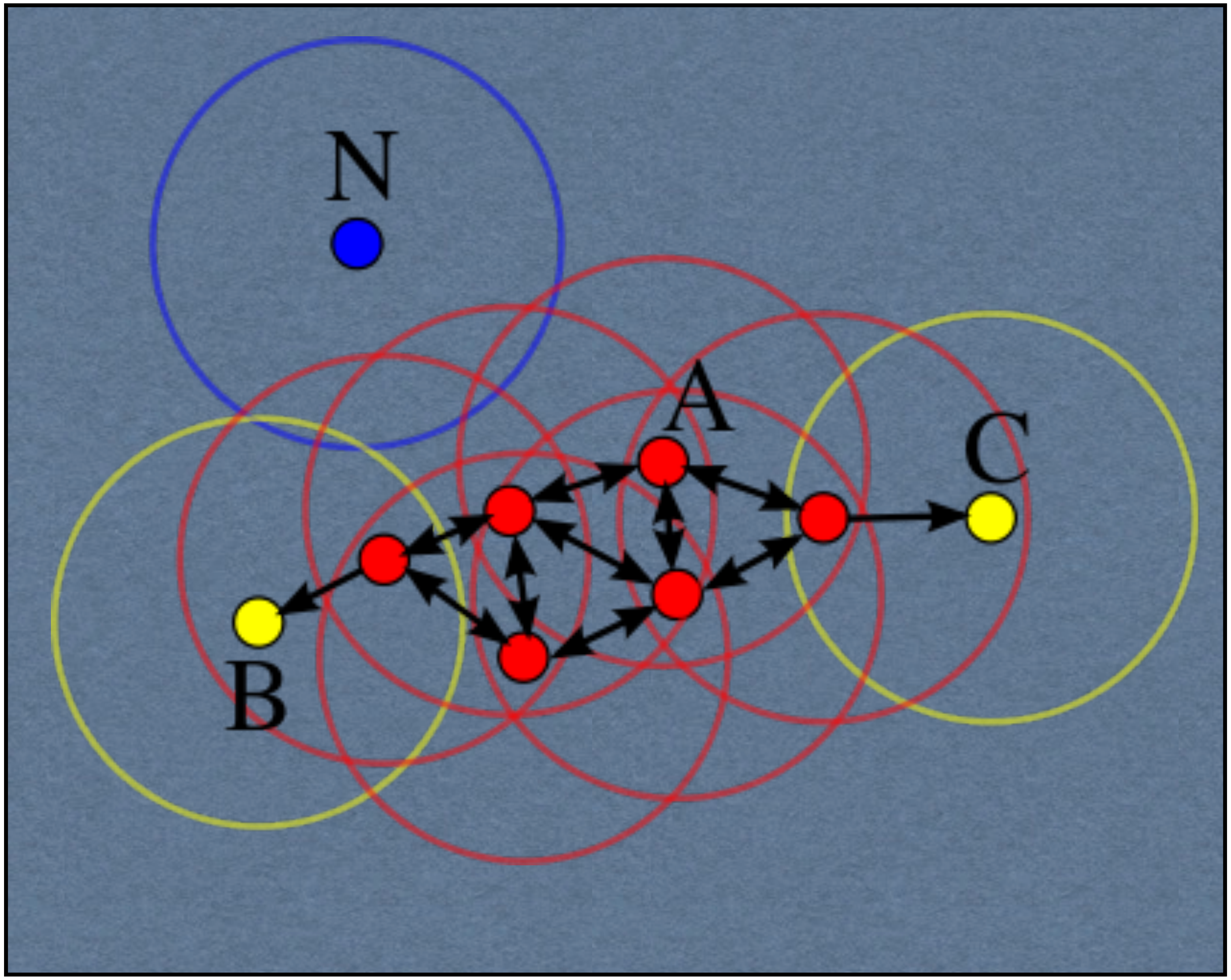
Outlier

Points not reachable from any other points

A core point and all points reachable from it form a cluster

https://en.wikipedia.org/wiki/DBSCAN

Example - minPts = 4



DBSCAN Issues

ϵ & minPts determine the clusters

No need to determine number of clusters

Robust to outliers

Can be implemented with runtime O(n log n)

Can not handle data with varying densities

High demensional data causes problems with selecting ϵ & minPts

Clustering.jl DBSAN

Two algorithms

dbscan(D, eps, minpts)

D - distance matrix

eps - raduis of neighborhood

minpts - points needed for core (density)

Following is faster and used less memory using KDTree

dbscan(points, radius, leafsize=20, min_neighbors=1, min_cluster_size=1)

points - data (column based)

radius = eps = ϵ

leafsize - size of leaf in KDTree

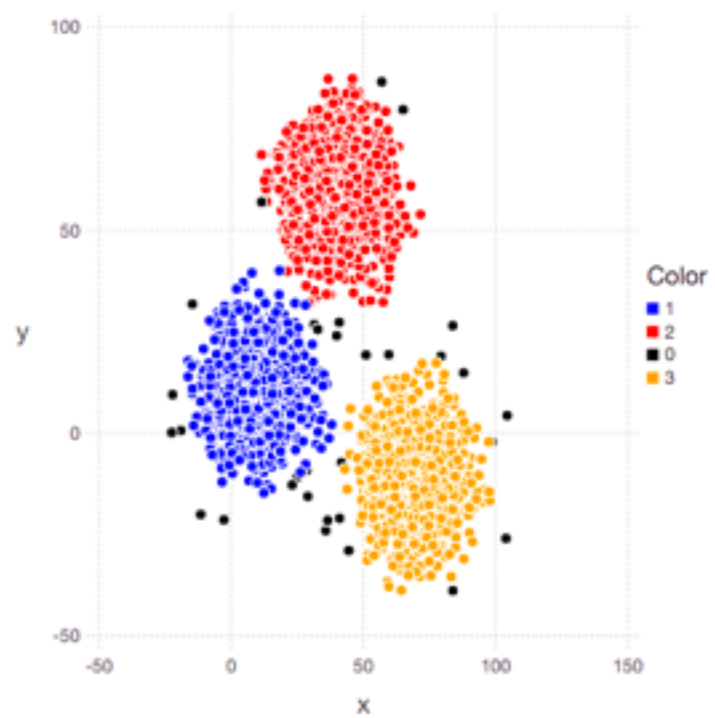
min_neighbors = minpts

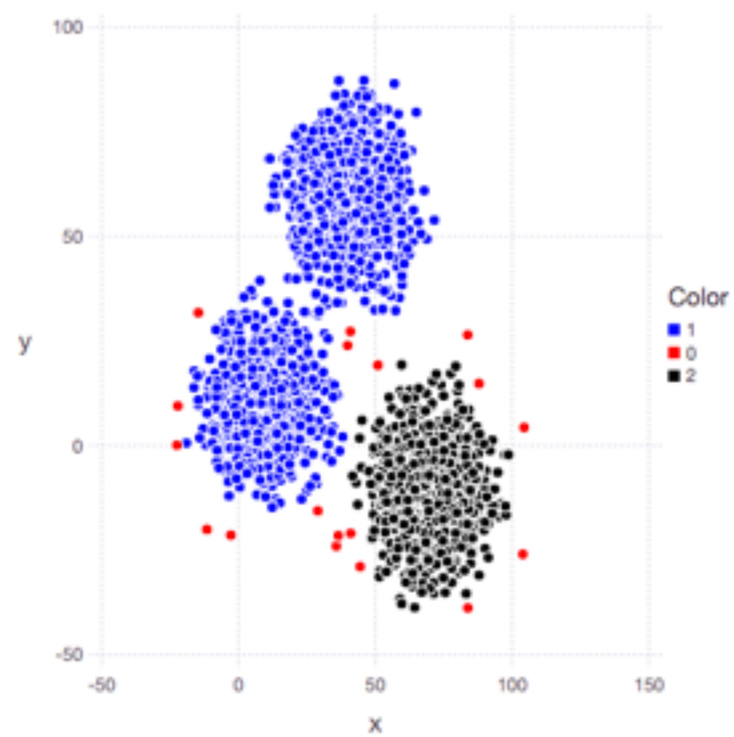
min_cluster_size - number of points needed to be a cluster

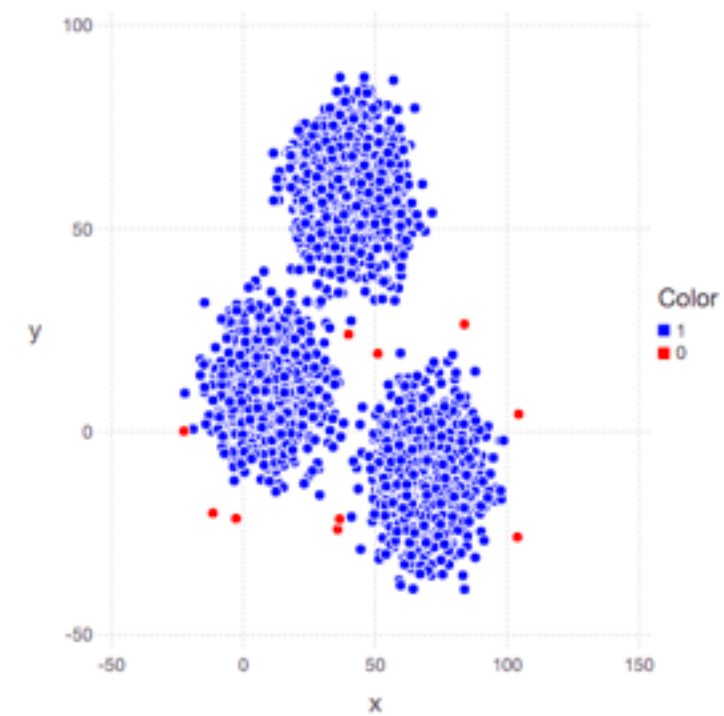
Sample Run

```
xclara_df = dataset("cluster", "xclara")
names!(xclara_df, [Symbol(i) for i in ["x", "y"]])

xclara_array = convert(Array, xclara_df);
xclara_array = xclara_array'

xclara_euclid = pairwise(Euclidean(),xclara_array)

xclara_dbscan = dbscan(xclara_euclid, 8, 10)

plot(x = xclara_array[1,:], y = xclara_array[2,:], color = assignments(xclara_dbscan),
        Geom.point(),
     Scale.color_discrete_manual("blue", "red", "black", "orange","green"))
```

DBSCAN with varying eps





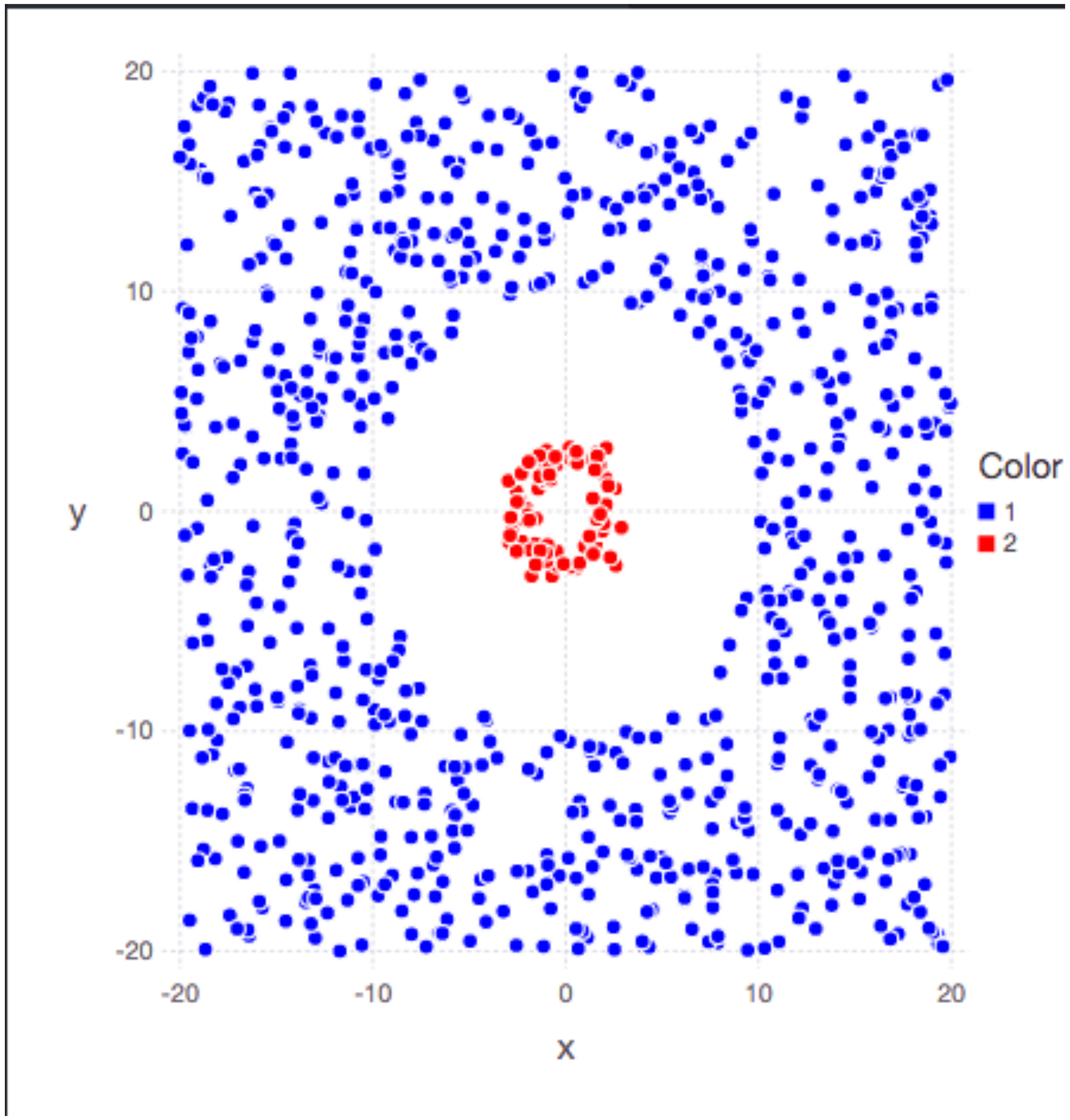


eps = 6
minpts = 10

eps = 7
minpts = 10

eps = 8
minpts = 10

DBSCAN & Non centered clusters



Reuters News Articles Dataset

Reuters-21578

21578 news articles published in 1987

Create simple recommondation system

Create clusters based on articles

Given an new article read by someone

Find what cluster is nearest

Select a few articles from that cluster as recommentaions

Sample Run

reuter_dir contains 200 articles

```
using TextAnalysis
using Clustering

# Read articles
crps = DirectoryCorpus(reuter_dir)
standardize!(crps, StringDocument)

#Clean up text
remove_case!(crps)
stem!(crps)

# Find all unique words
update_lexicon!(crps)

# translate articles into vectors
m = DocumentTermMatrix(crps)
D = dtm(m, :dense)
T = tf_idf(D)

cluster = kmeans(T, 5)
```

Issues

All at high level so what really happened?

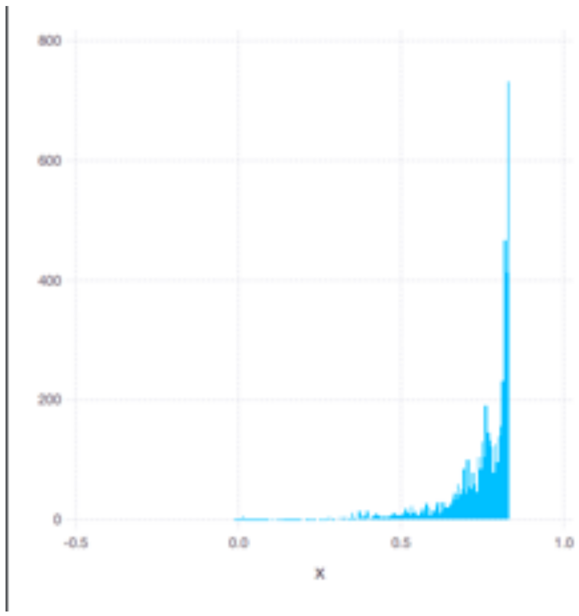
- What words were removed?
- What words remained

Contained 4962 unique words -> dealing with 4962 dimensional space

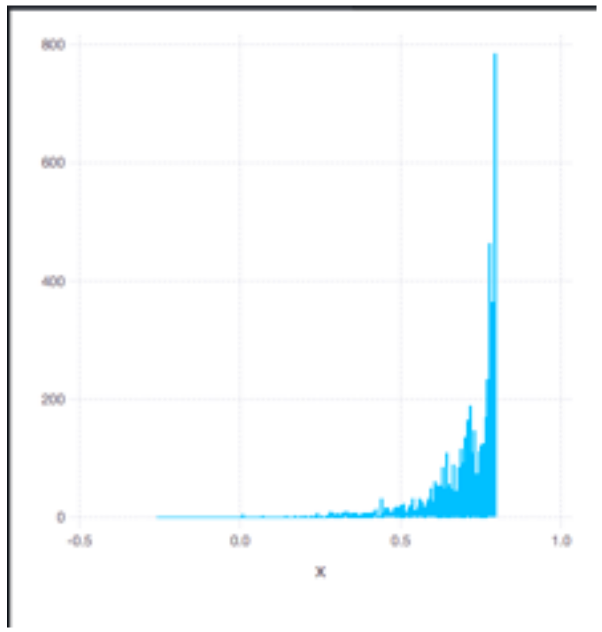
- Full data set contains about 130,000 unique words

How to interpret the results?

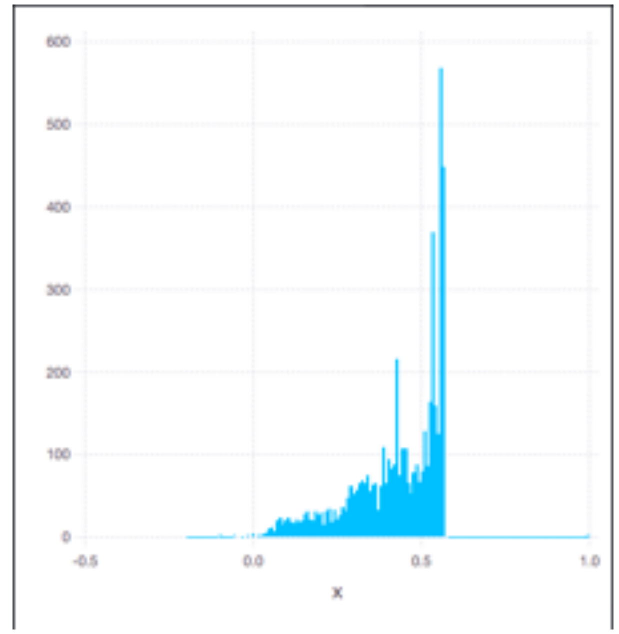
Silhouettes



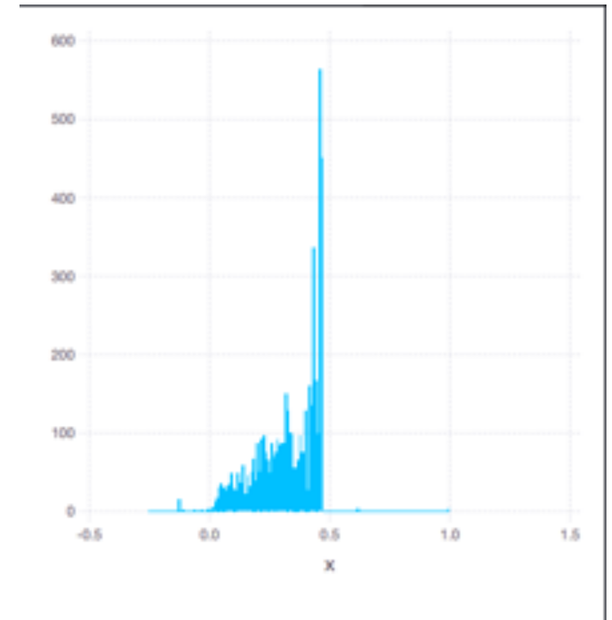
k=2



k=3



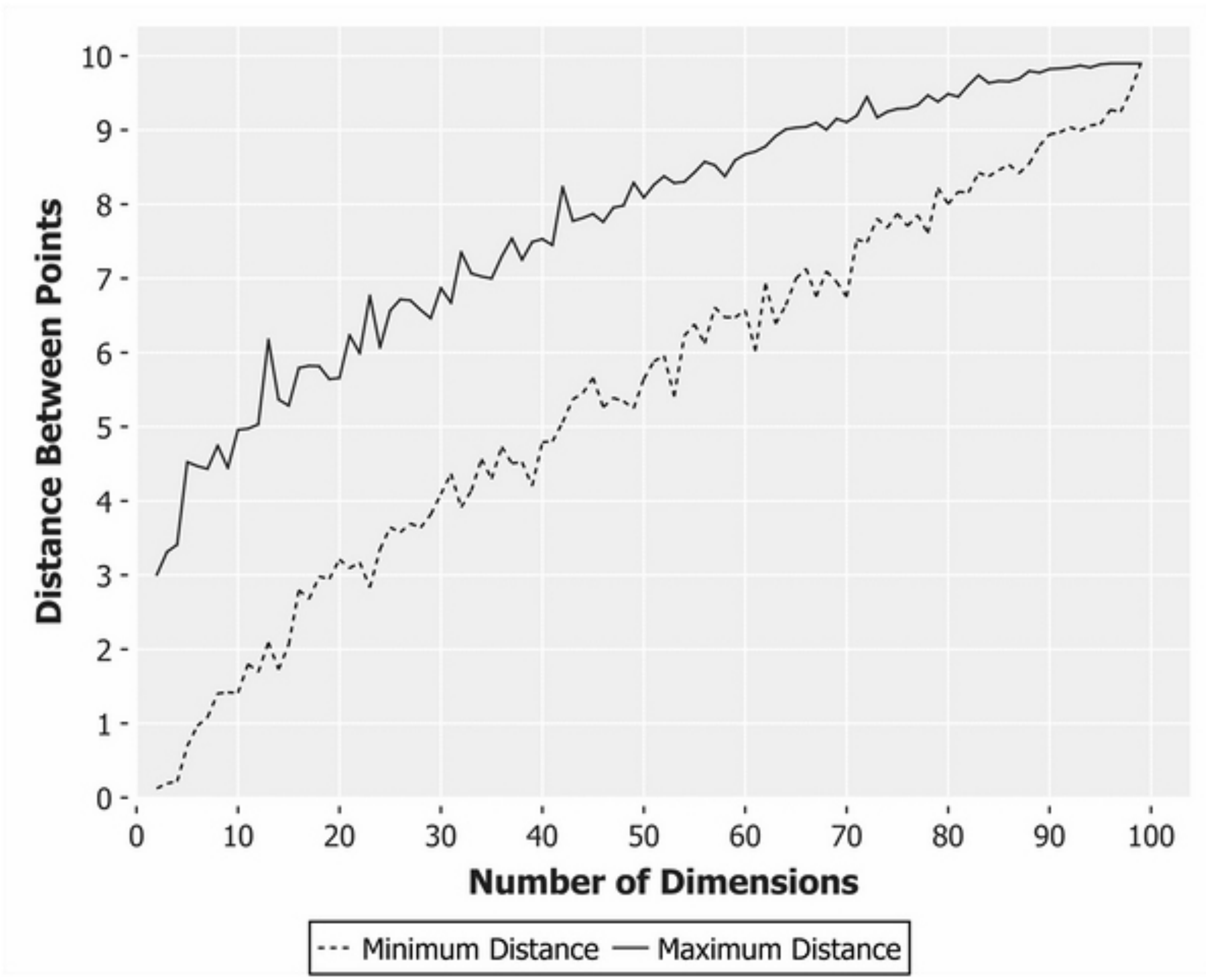
k=5



k=10

Curse of Dimensionality

As dimensions rise every point tends to become equally far from every other point

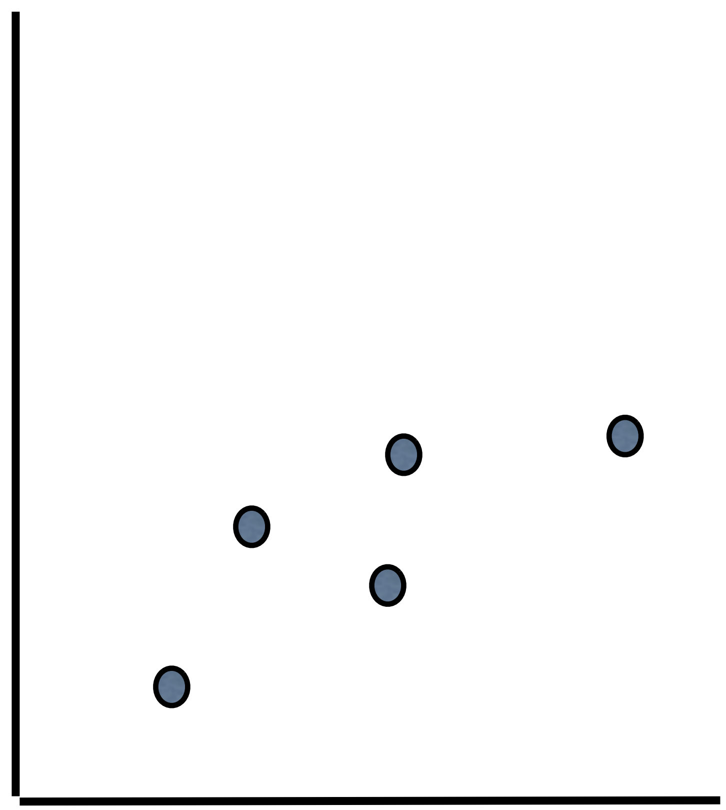


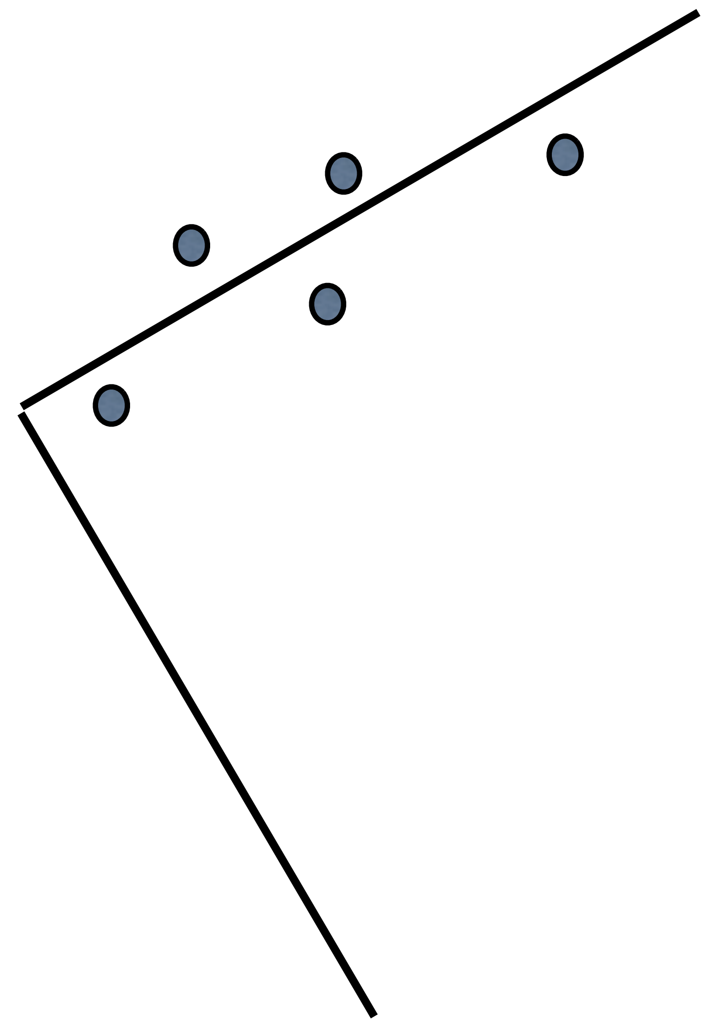
Reducing Dimensions

Some dimensions in a data set have less variation that others

So contribute less

These dimensions may not be the ones given in the data





PCA - Principle Component Analysis

Used to reduce the dimensionality of data

Changes the dimension of the data so

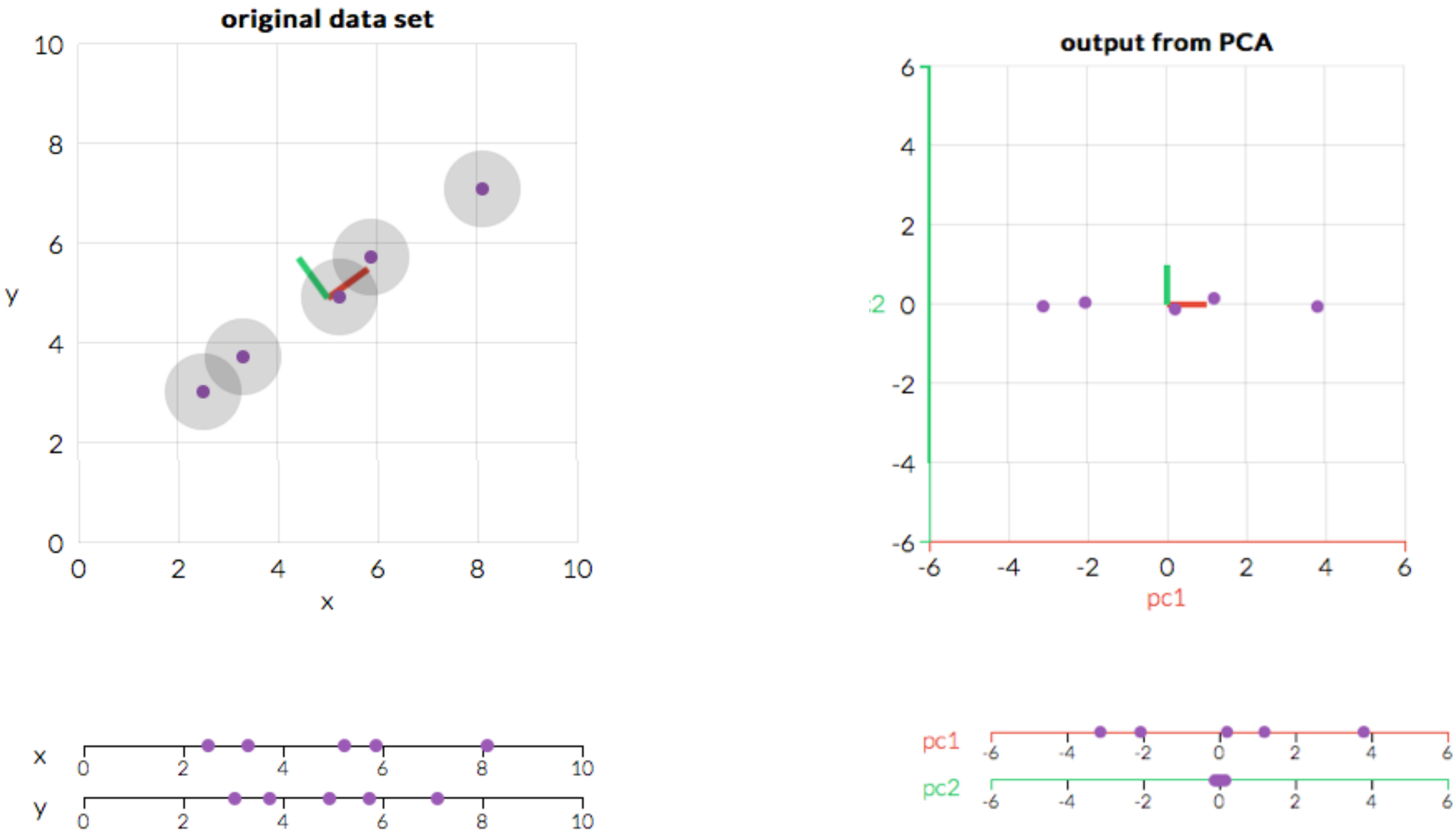
First dimension has the greatest variance
Second dimesion has second greatest variance
...

Can then select first K dimensions to work with

Data is transformed into different corrdinate system

Example

http://setosa.io/ev/principal-component-analysis/



http://setosa.io/ev/principal-component-analysis/

MultivariateStats.jl

Implements PCA

fit(PCA,data; options)

options

pration = ratio of variances preserved

maxoutdim

transform(PCA_result,data)

returns data in new coordinates

Example

```
using MultivariateStats
data = [1.0 1.0; 2.0 2.2; 3.1 3.2; 4.0 3.9]

pca_model = fit(PCA, data')          # data needs to be in columns not row - ‘ = transpose

show(pca_model)                      #PCA(indim = 2, outdim = 1, principalratio = 0.99777)

transformed_data = transform(pca_model,data')'
```

```
 2.19109
 0.638135
-0.84779
-1.98144
```