O'REILLY®

Security

BUILD BETTER DEFENSES

oreillysecuritycon.com
#oreillysecurity

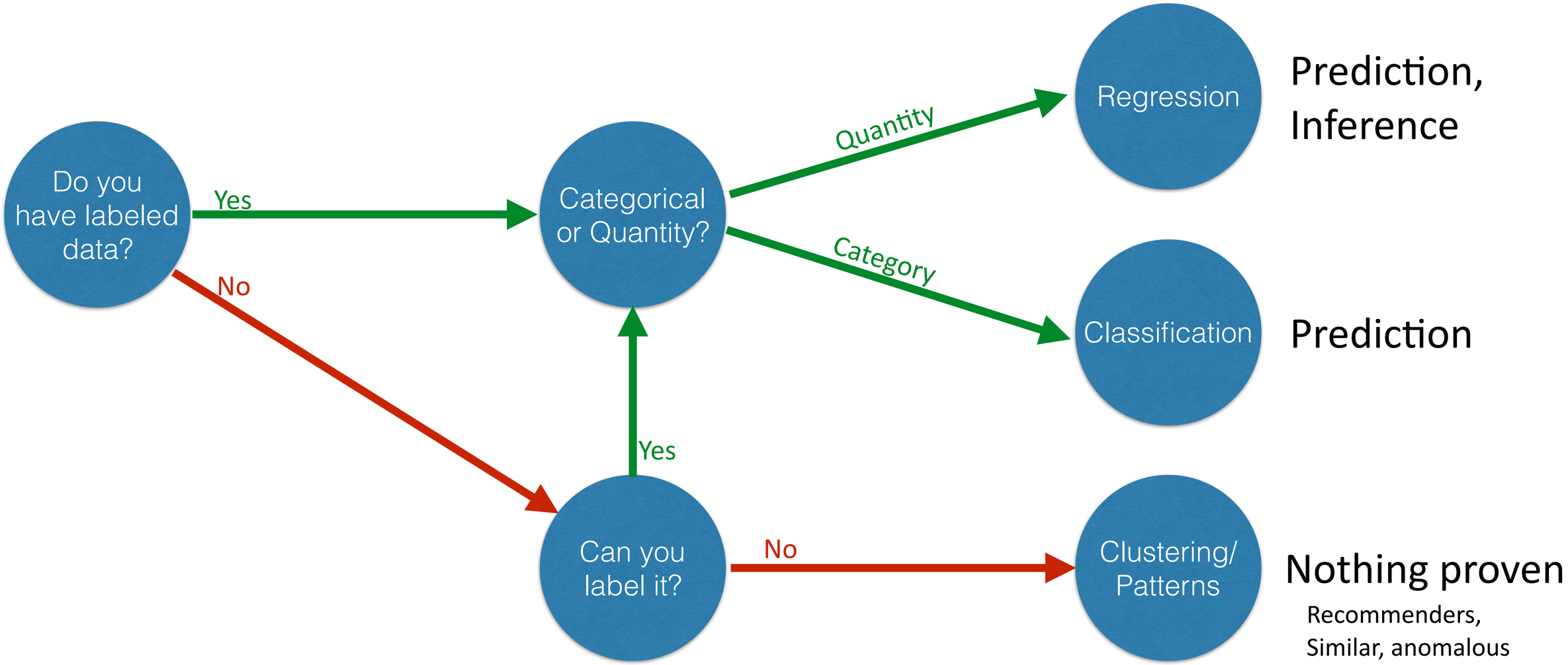
Foundations of Security Data Science

Machine Learning: Intro

Jay Jacobs
Charles Givre

What we learned yesterday

What Data Science kind of is…



Data Science Topics

Core Statistics
Descriptive (range, median, mean)
Confidence Intervals
Correlation
Tests (t.test, ks.test, fitting)

Regression/Classification
Labeled Data
Can falsify claims
Prediction, maybe inference
No free lunch

Clustering/Unsupervised
Unlabeled data
Cannot falsify claims
Clustering
Anomoly Detection

Visualization
Visual cues, coord. system, scale, context
Pre-attentive processing, saccadic
Working and Long-term memory
Accuracy in Decoding, mumbling

Feature Engineering

1. Define the Object of Measurement
   These will become the rows in your data (one per object)

2. Define one or more “features” that describe each object
   These will become the columns in your data

3. Run Algorithm (more on this later)

4. Optimally, measure feature contribution and model performance

Group Discussion

Instructions: In groups, discuss the scenarios and devise a plan for how you would go about implementing a prototype. DO NOT WRITE CODE. Focus on questions like: What data would I need? What are possible obstacles or challenges? What modeling techniques could I use?

- **Scenario 1:** You have been tasked with prototyping a system to identify the use of stolen credentials. How might you approach this task?

- **Scenario 2:** You are working for a major restaurant chain and you would like to employ data science to improve efficiency at your restaurants. What are some things you could do?

- **Scenario 3:** You have been tasked with building a system to identify fraudulent credit card transactions. How would you approach this problem?

Measuring Distance

(precursor to clustering)

Unsupervised Clustering Algorithm

1. Select Features
2. Calculate a distance measure
3. Apply a clustering algorithm
4. Validate?

Which Departments are Similar?

	Malware events
Dept1	6
Dept2	1
Dept3	8

Which Departments are Similar?

	Malware events	Phishing
Dept1	6	6
Dept2	1	2
Dept3	8	1

Which Departments are Similar?

	Malware events	Phishing	Open Tickets
Dept1	6	6	3
Dept2	1	2	1
Dept3	8	1	9

Computing Distance

	Malware events
Dept1	6
Dept2	1
Dept3	8

Compare:

Dept1 to Dept2: | 6 - 1 | = 5

Dept2 to Dept3: | 1 - 8 | = 7

Dept1 to Dept3: | 6 - 8 | = 2

Two-Dimenionsal Distance

	Malware events	Phishing
Dept1	6	6
Dept2	1	2
Dept3	8	1

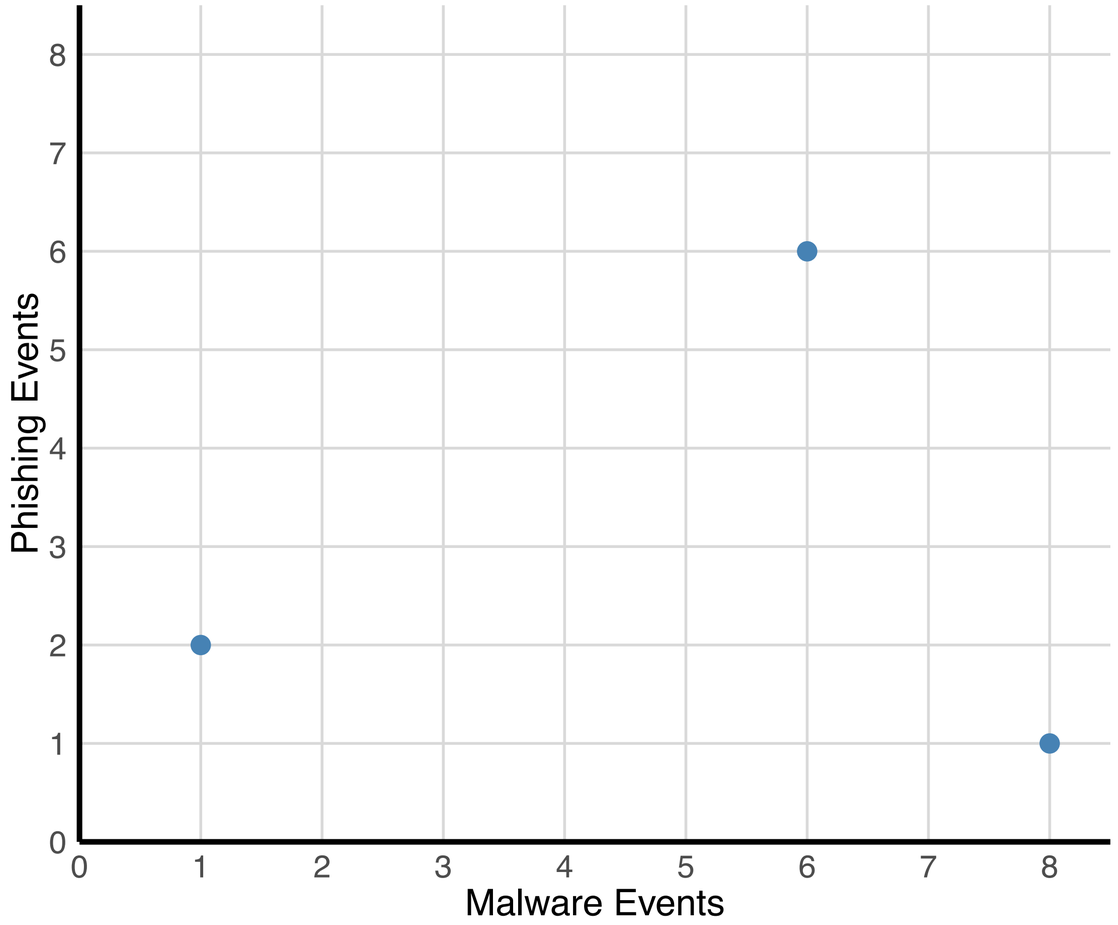
Multiple Distance methods

- Euclidean
- Manhattan
- Maximum
- Canberra
- Binary
- Minkowski

… (to name a few)

Two-Dimenionsal Distance

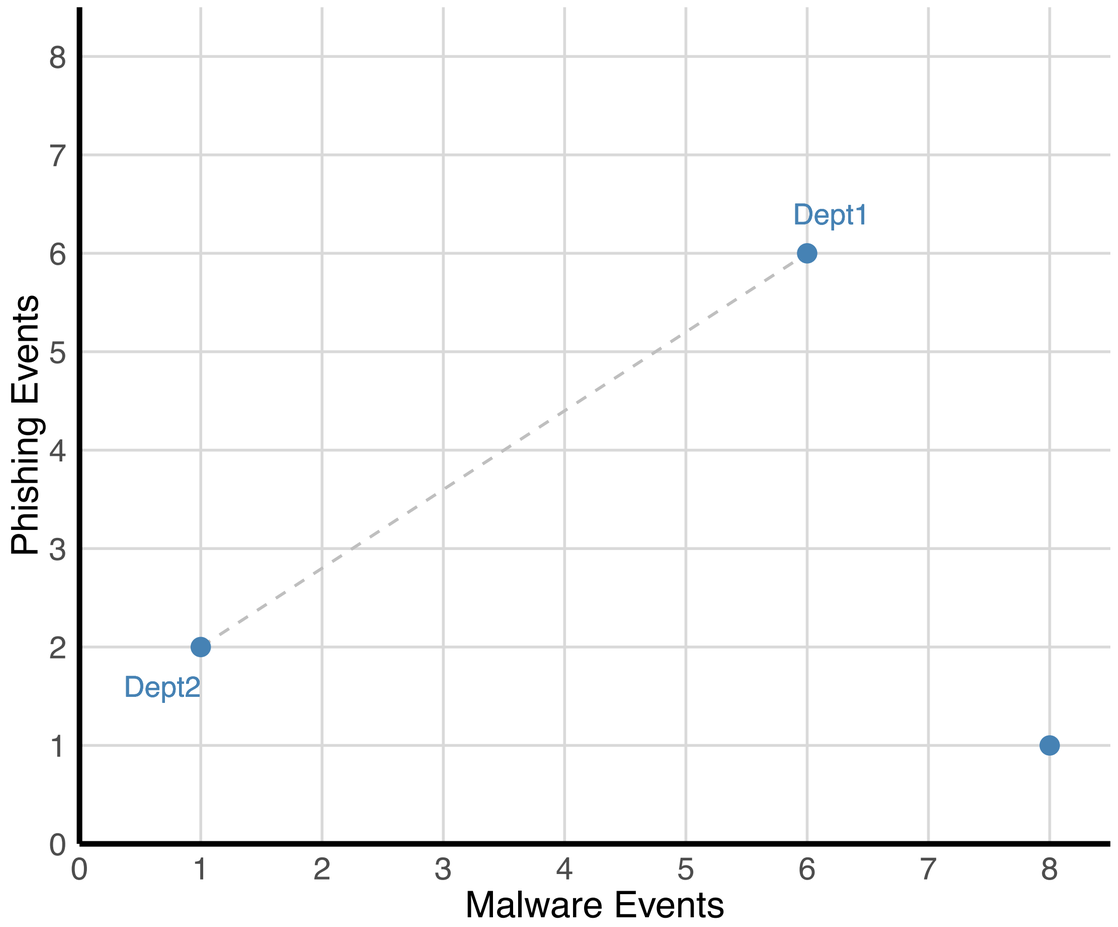
	Malware events	Phishing
Dept1	6	6
Dept2	1	2
Dept3	8	1



Two-Dimenionsal Distance

Euclidean very common and easy to grok

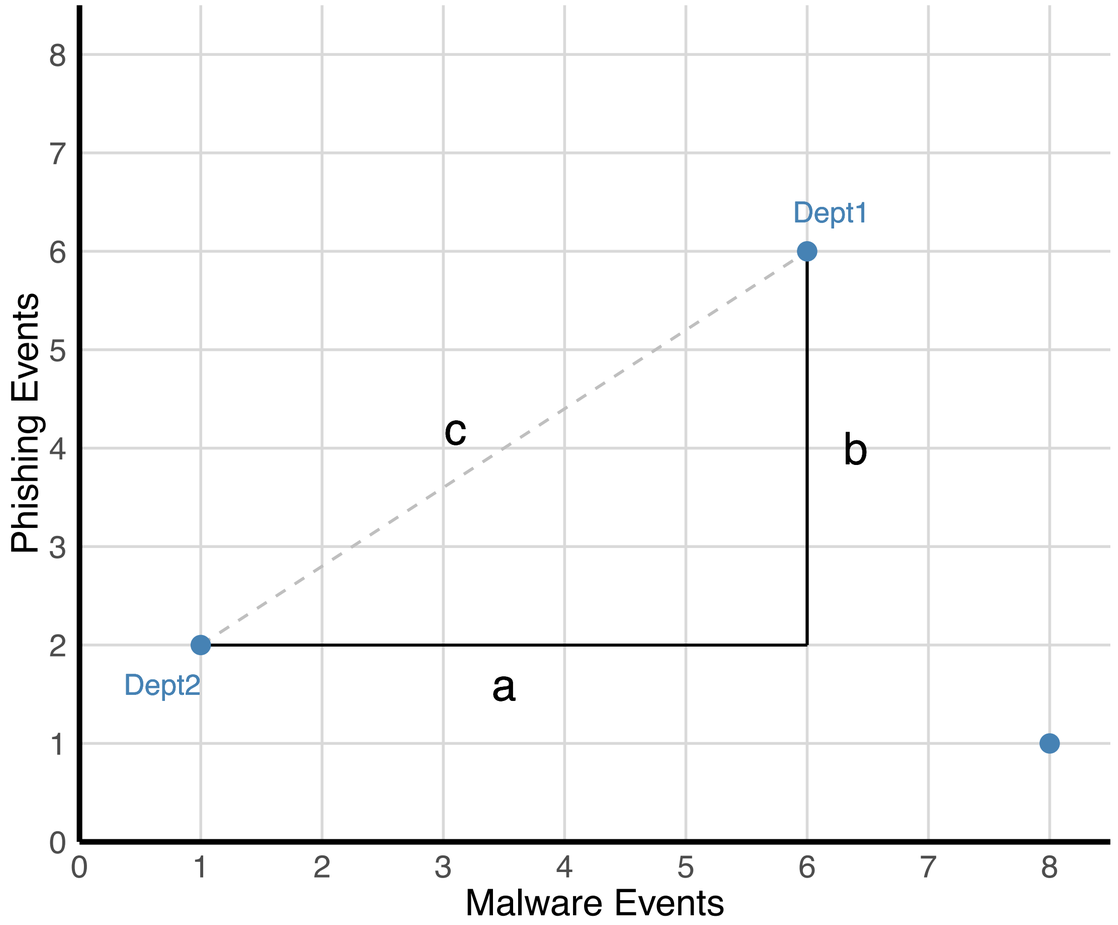
	Malware events	Phishing
Dept1	6	6
Dept2	1	2
Dept3	8	1



Two-Dimenionsal Distance

Euclidean very common and easy to grok: $a^2 + b^2 = c^2$

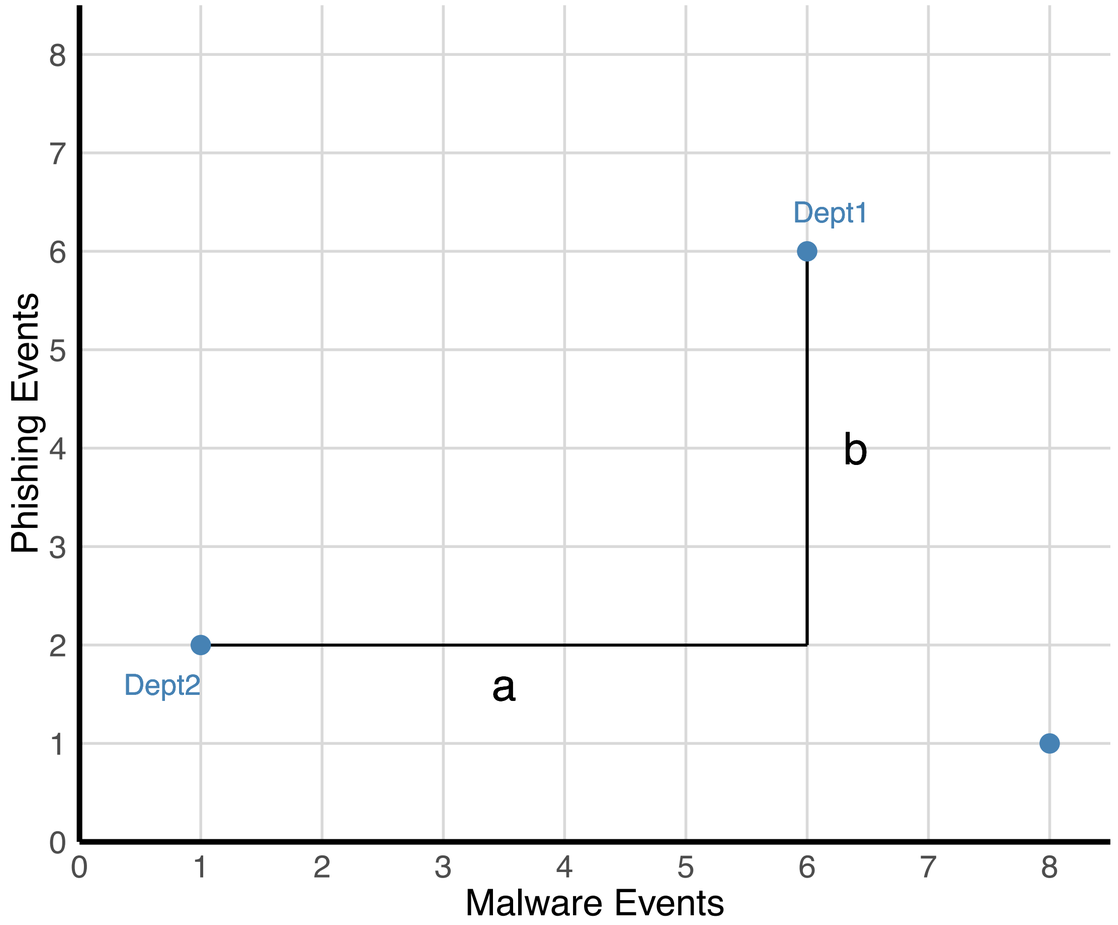
	Malware events	Phishing
Dept1	6	6
Dept2	1	2
Dept3	8	1



Two-Dimenionsal Distance

Manhattan also easy to comprehend: a + b

	Malware events	Phishing
Dept1	6	6
Dept2	1	2
Dept3	8	1



Computing Distance

	Malware events	Phishing
Dept1	6	6
Dept2	1	2
Dept3	8	1

Compare:

Dept1 to Dept2: sqrt((6-1)^2 + (6-2)^2) = **6.4**

Dept2 to Dept3: … = **7.1**

Dept1 to Dept3: … = **5.4**

Euclidean Distance calculations

```
def dist(x,y):
    return np.sqrt(np.sum((x-y)**2))

> mat = np.array([[ 6,6,3 ], [1,2,1], [8,1,9]])
> dist(mat[0], mat[1])
6.7082039324993694
> dist(mat[1], mat[2])
10.677078252031311
> dist(mat[0], mat[2])
8.0622577482985491
```

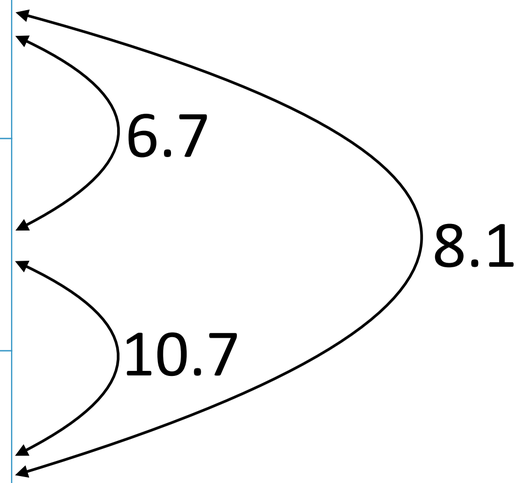
Python

```
> mat <- matrix(c(6,1,8,6,2,1,3,1,9), nrow = 3)
> mat
     [,1] [,2] [,3]
[1,]    6    6    3
[2,]    1    2    1
[3,]    8    1    9
> dist(mat) # default is euclidean
          1         2
2  6.708204
3  8.062258 10.677078
```

R

Which Departments are Similar?

	Malware events	Phishing	Open Tickets
Dept1	6	6	3
Dept2	1	2	1
Dept3	8	1	9



O'REILLY®
Security
BUILD BETTER DEFENSES

oreillysecuritycon.com
#oreillysecurity

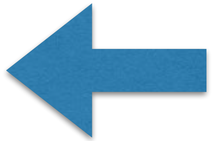
Applying Distances
(MDS)

So what now?

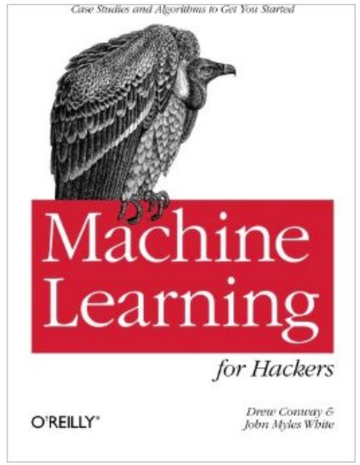
US Senator Similarity: 111th US Congress

cong	id	state	dist	lstate	party	eh1	eh2	name	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	…	…	…
111	49300	71	0	CALIFOR	100	0	1	FEINSTEIN	1	1	1	1	6	1	6	1	9	1	1			
111	29906	62	0	COLORAD	100	0	1	UDALL	1	1	1	1	6	1	6	1	1	1	1			
111	40500	62	0	COLORAD	100	1	1	SALAZAR	1	1	1	1	6	0	0	0	0	0	0			
111	40910	62	0	COLORAD	100	2	5	BENNET	0	0	0	0	0	0	0	0	0	0	1			
111	14213	1	0	CONNECT	100	0	1	DODD	1	1	1	1	6	1	6	1	1	1	1			
111	15704	1	0	CONNECT	100	0	1	LIEBERMAN	1	1	1	1	6	1	6	1	1	1	1			
111	14101	11	0	DELAWAR	100	1	1	BIDEN	9	9	9	1	6	0	0	0	0	0	0			
111	40901	11	0	DELAWAR	100	2	5	KAUFMAN	0	0	0	0	0	1	6	1	1	1	1			
111	40916	11	0	DELAWAR	100	3	2	COONS	0	0	0	0	0	0	0	0	0	0	0			
111	15015	11	0	DELAWAR	100	0	1	CARPER	1	1	1	1	6	1	6	1	1	1	1			

sen1	sen2	distance
...	...	...
CHAMBLISS (R)	SCHUMER (D)	45.287967
WICKER (R)	SCHUMER (D)	44.452222
WYDEN (D)	ENZI (R)	45.376205
BROWN (R)	LAUTENBERG (D)	33.120990
LEVIN CARL (D)	WICKER (R)	44.698993
BINGAMAN (D)	COBURN (R)	47.180504
LEVIN CARL (D)	CASEY (D)	12.727922
UDALL (D)	GOODWIN (D)	25.495098
DURBIN (D)	CARDIN (D)	9.433981
BROWN (R)	SPECTER (R)	21.213203
...	...	...



What can we do with these distances?

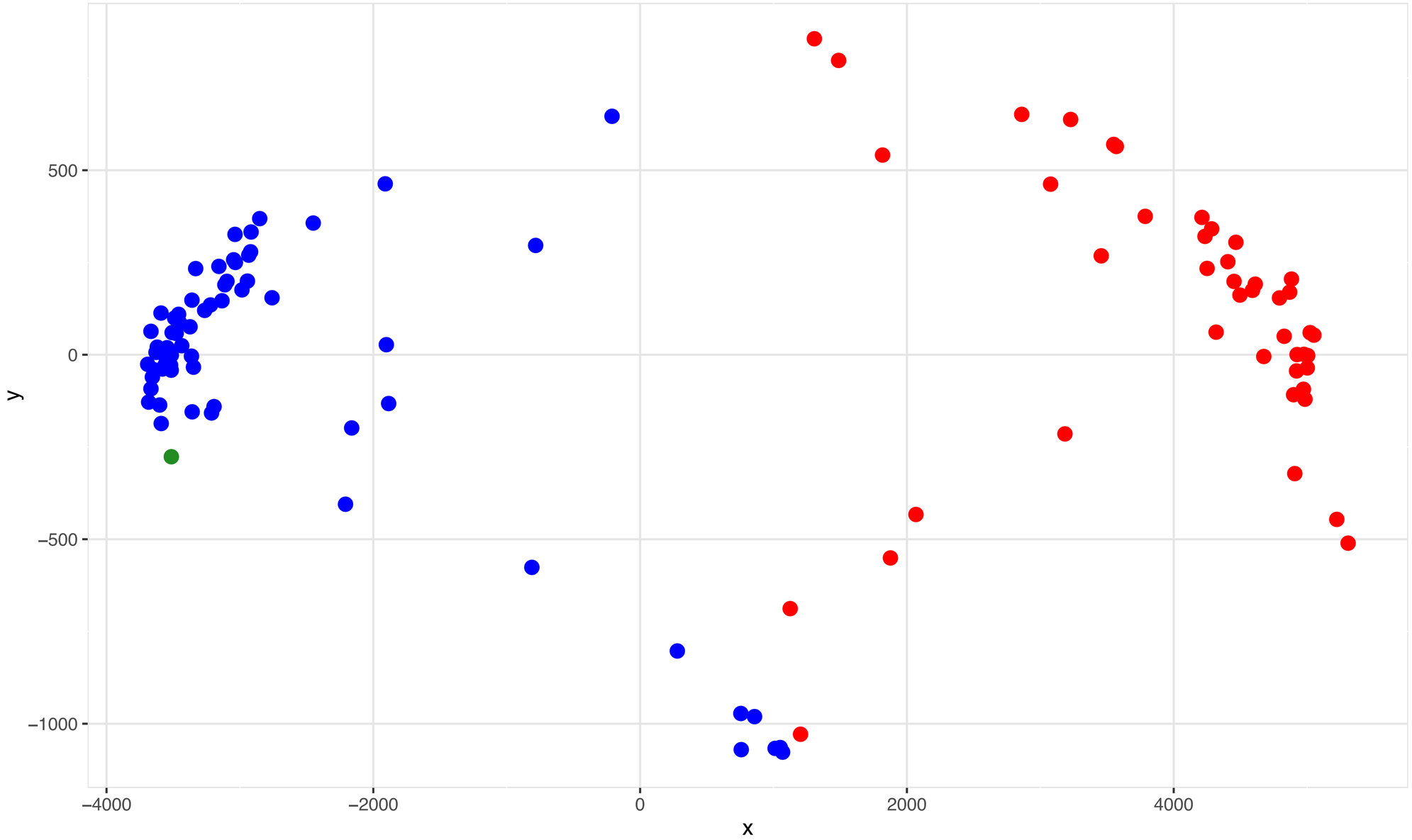


Multi-Dimensional Scaling (MDS)

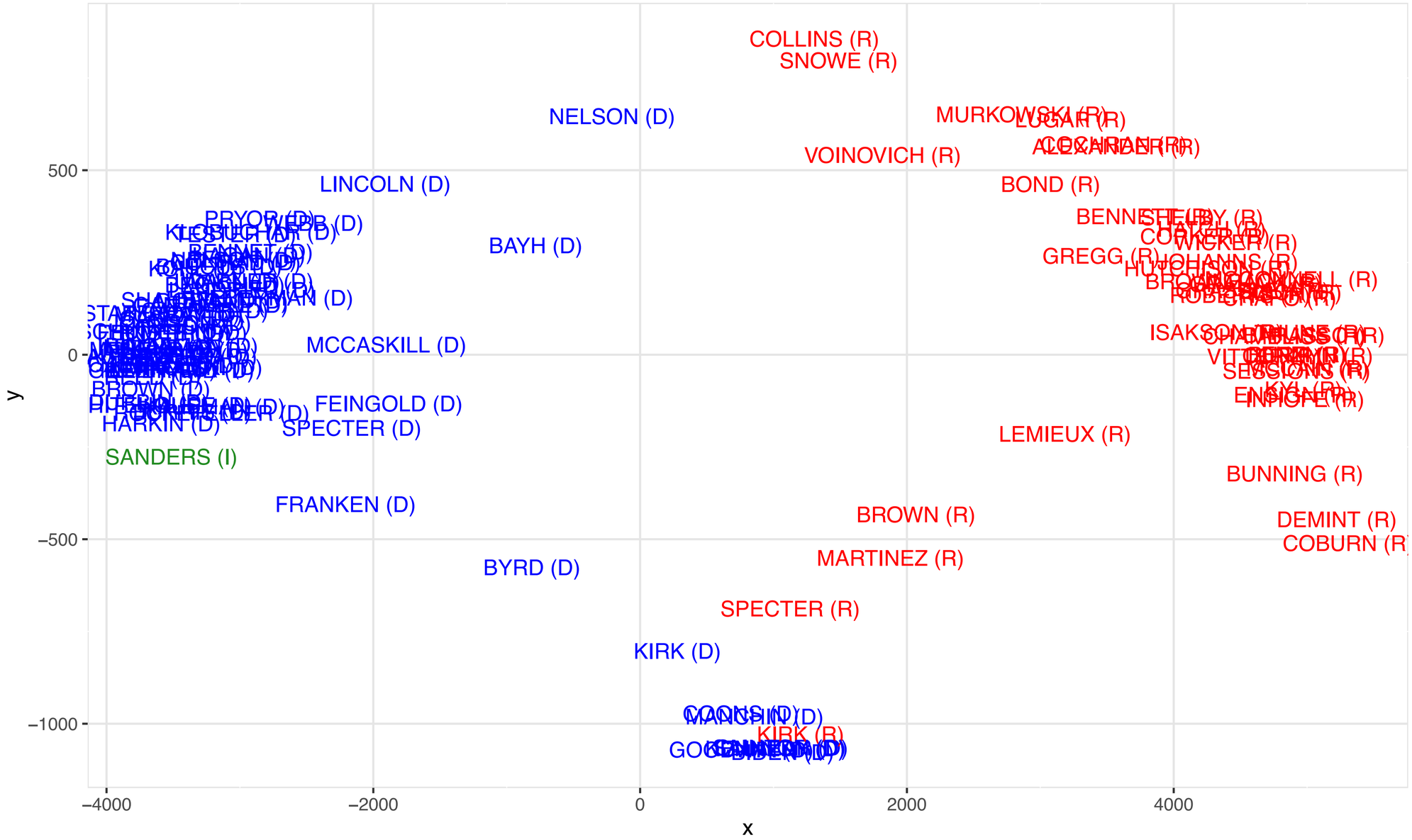
"An MDS algorithm aims to place each object in N-dimensional space such that the between-object distances are preserved as well as possible." (wikipedia)

x	y	name
...	...	...
-2759.2620	154.41524	LIEBERMAN (D)
-3520.0632	-28.39657	MERKLEY (D)
-1910.7153	463.20060	LINCOLN (D)
1202.5691	-1028.44244	KIRK (R)
-3691.4226	-25.88029	CARDIN (D)
-2943.8030	199.57122	WARNER (D)
-3459.6420	109.54234	MURRAY (D)
4972.5638	-93.60466	KYL (R)
-210.5705	646.20845	NELSON (D)
-3113.0023	189.57231	BEGICH (D)
...	...	...

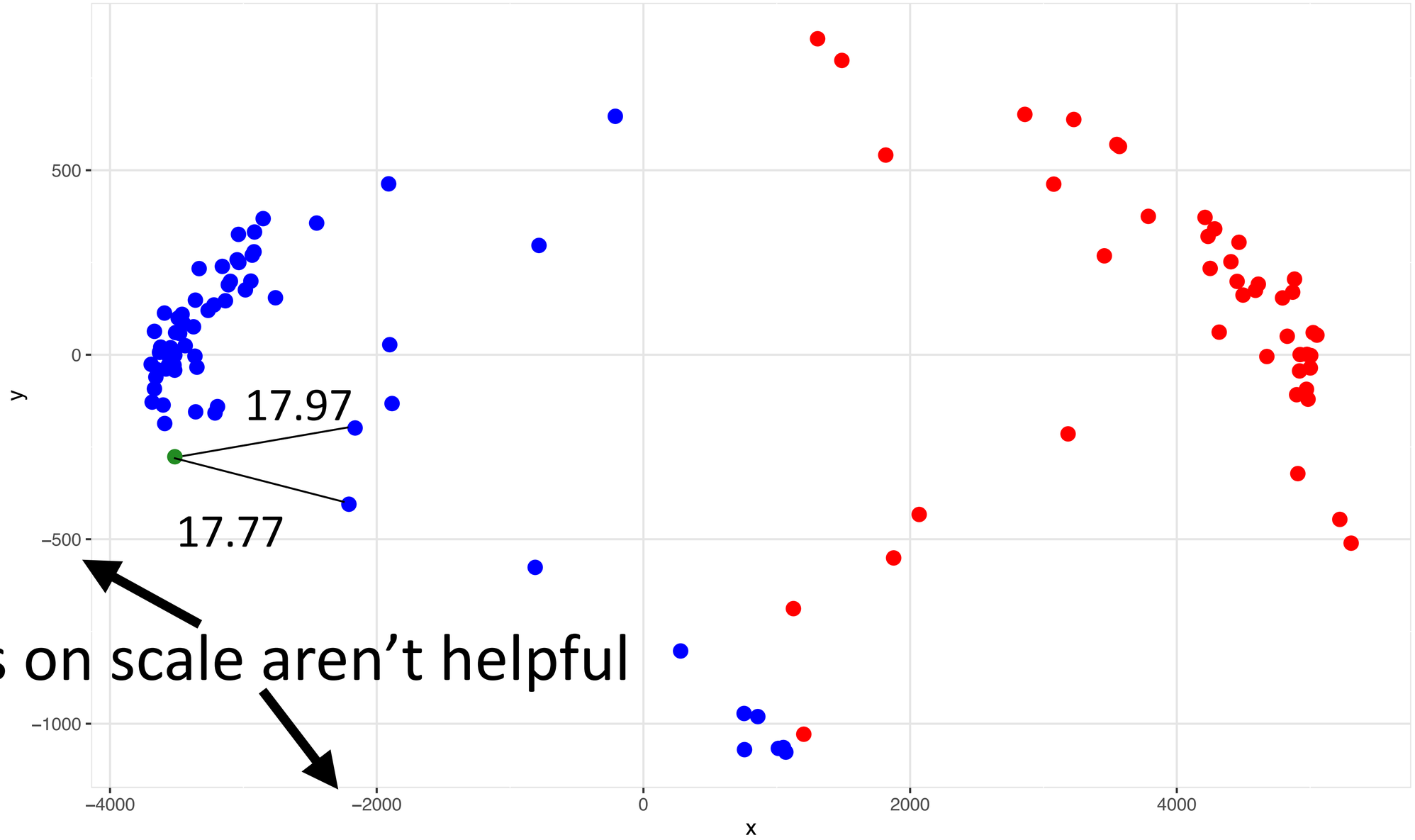
Multi-Dimensional Scaling (MDS)



Multi-Dimensional Scaling (MDS)

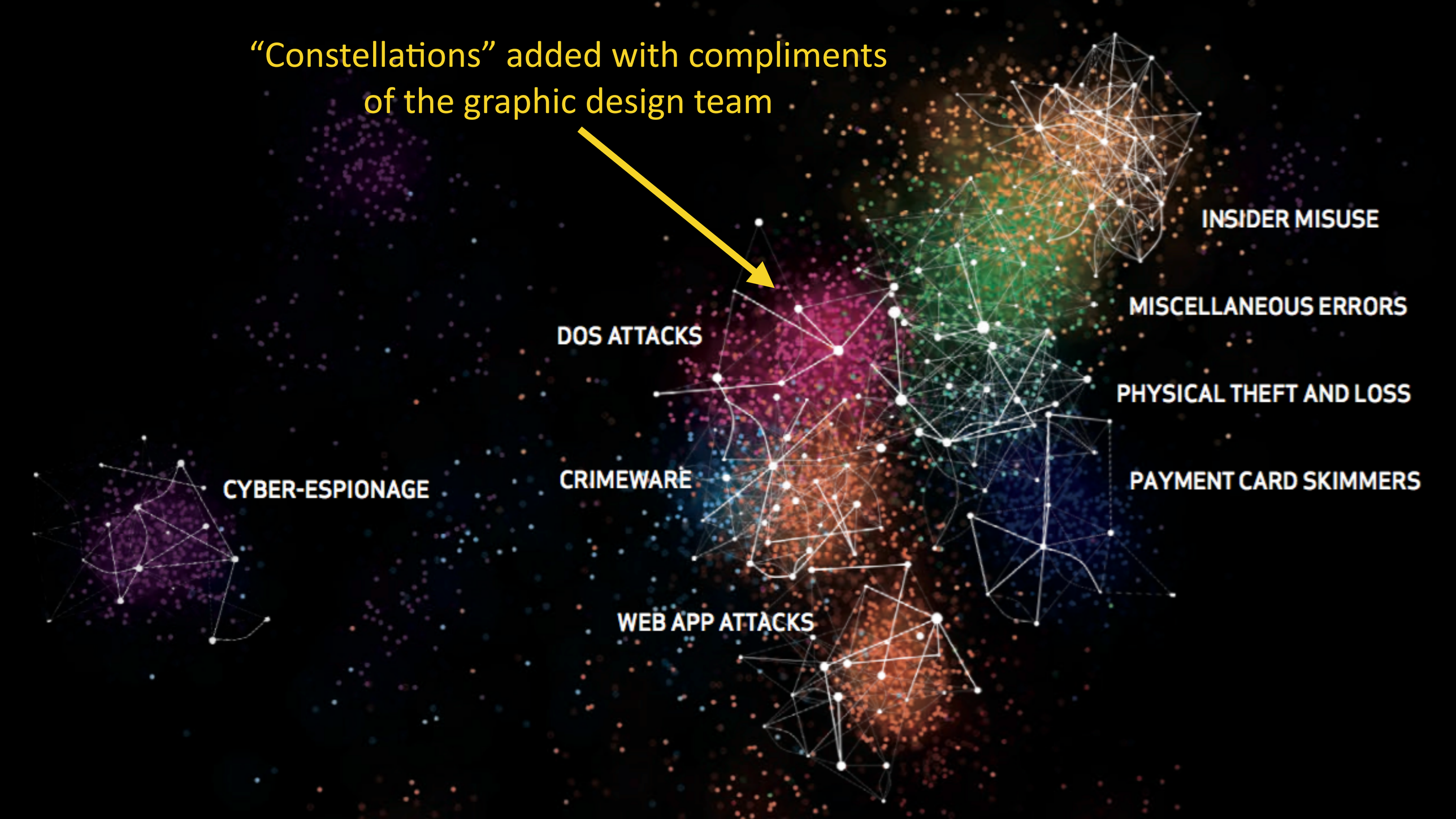


Multi-Dimensional Scaling (MDS)



Values on scale aren’t helpful

Recognize this?


“Constellations” added with compliments of the graphic design team
INSIDER MISUSE
MISCELLANEOUS ERRORS
DOS ATTACKS
PHYSICAL THEFT AND LOSS
CYBER-ESPIONAGE
CRIMEWARE
PAYMENT CARD SKIMMERS
WEB APP ATTACKS

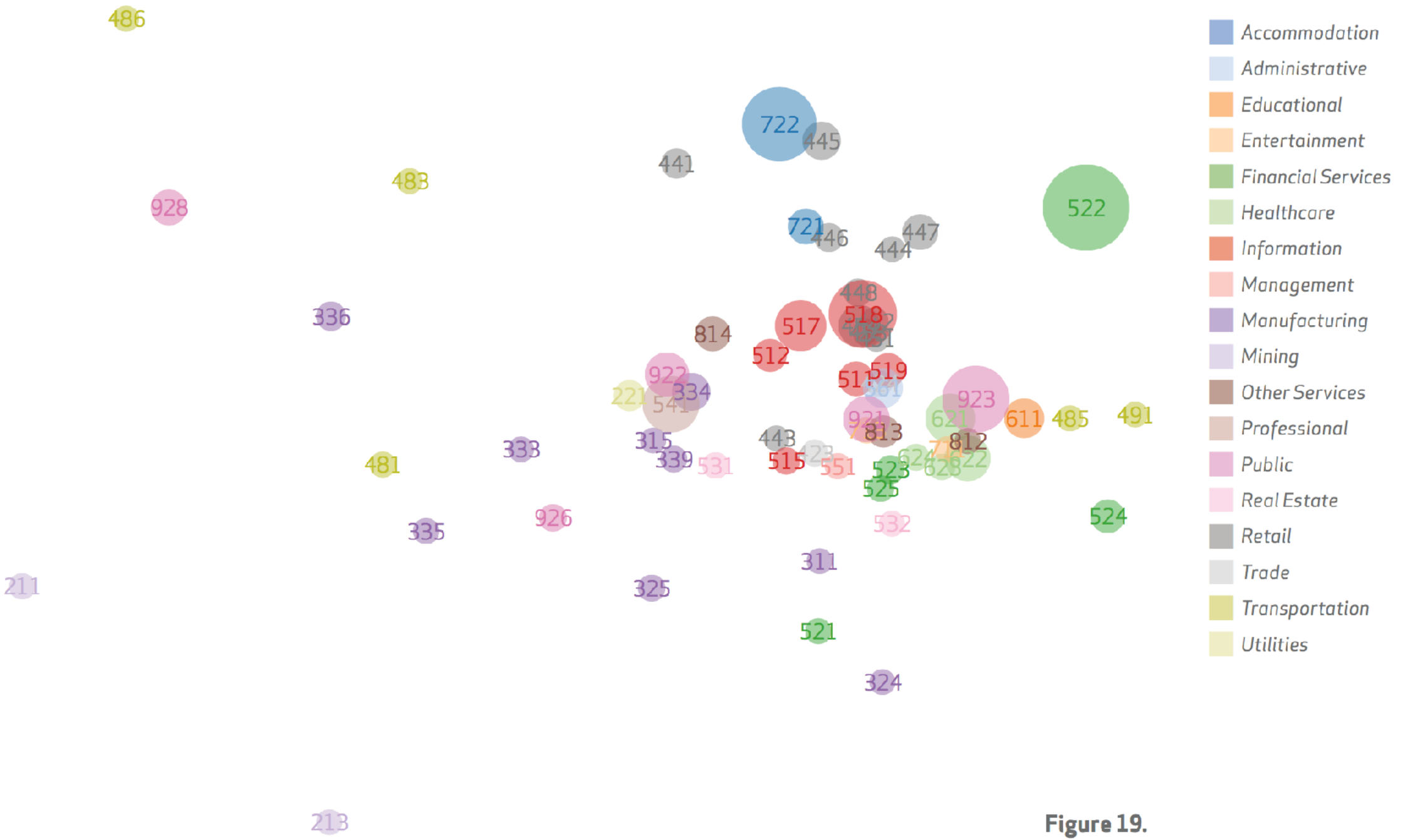


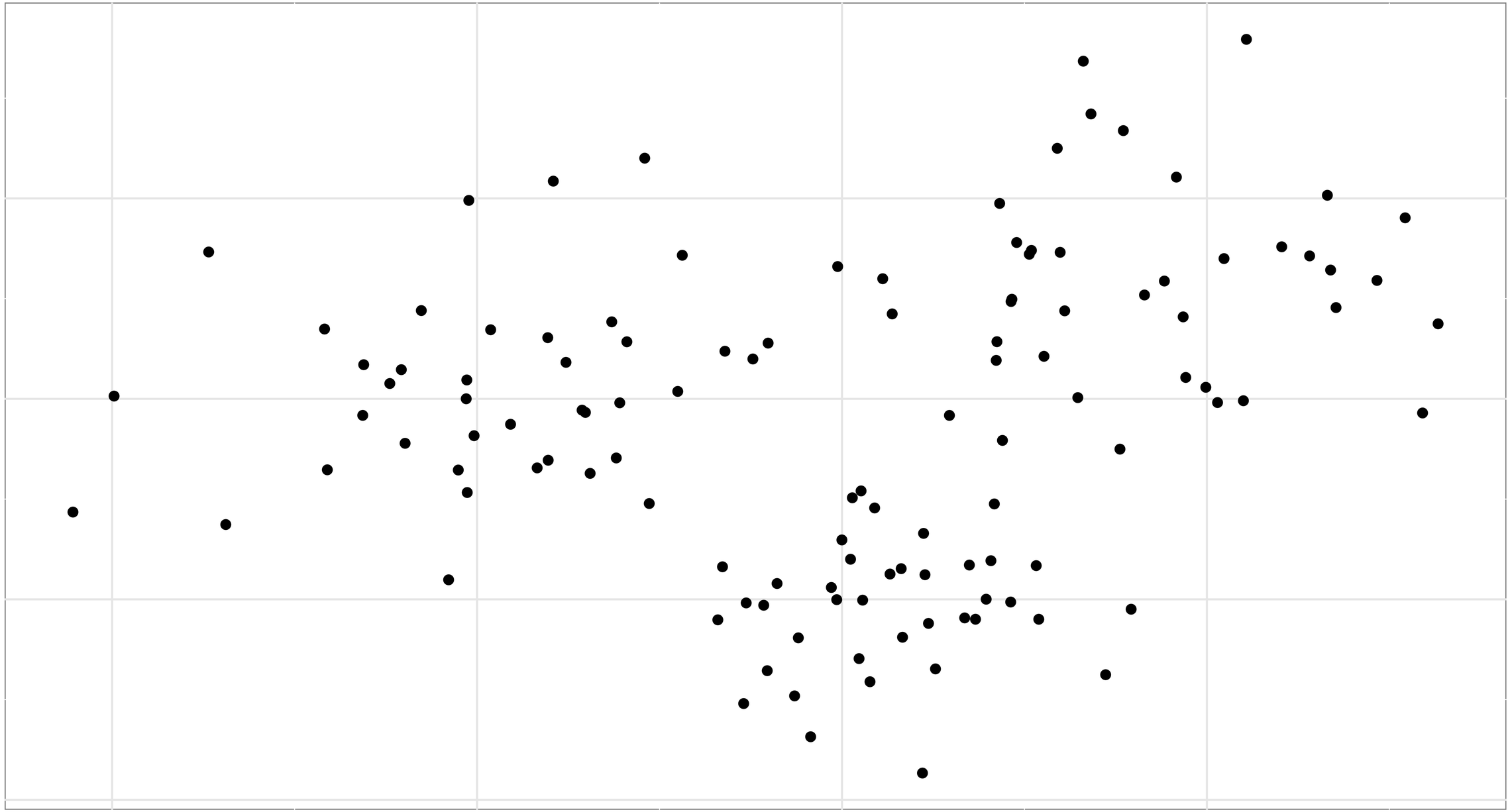
Figure 19.

Clustering on breach data across industries

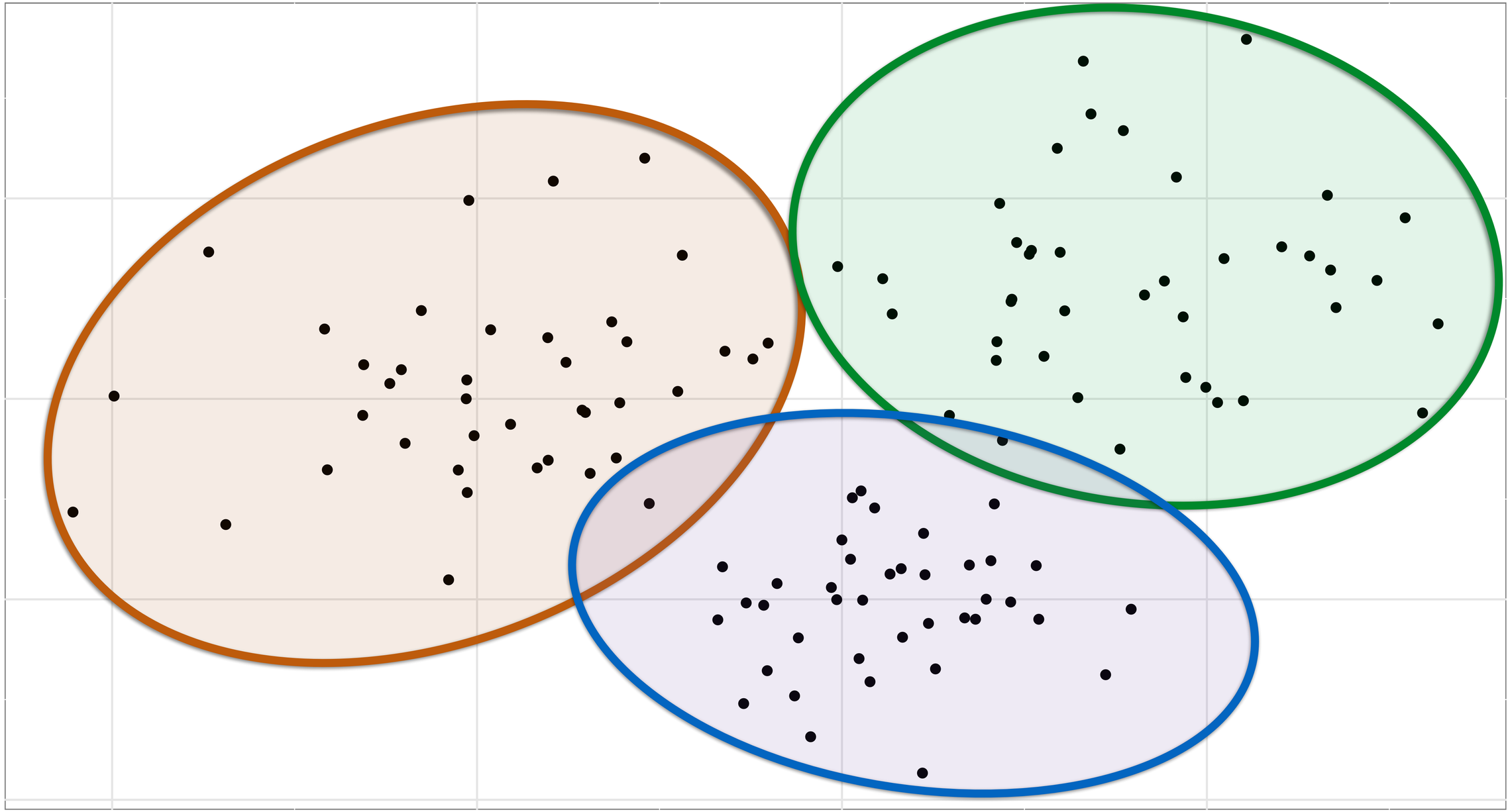
28 To look up the three-digit NAICS codes, visit: www.census.gov/eos/www/naics/index.html

Identifying membership of Clusters

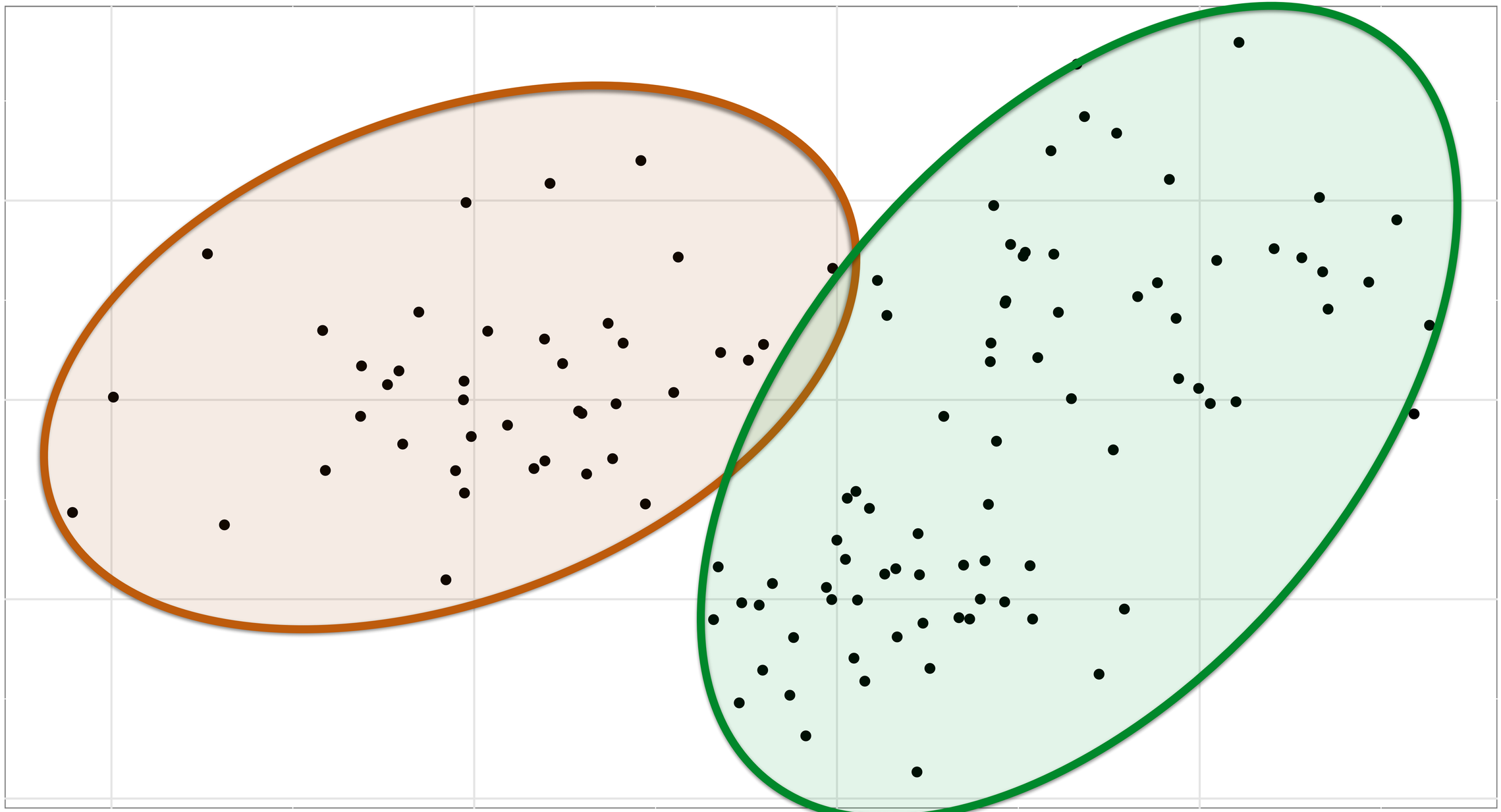
Clustering…



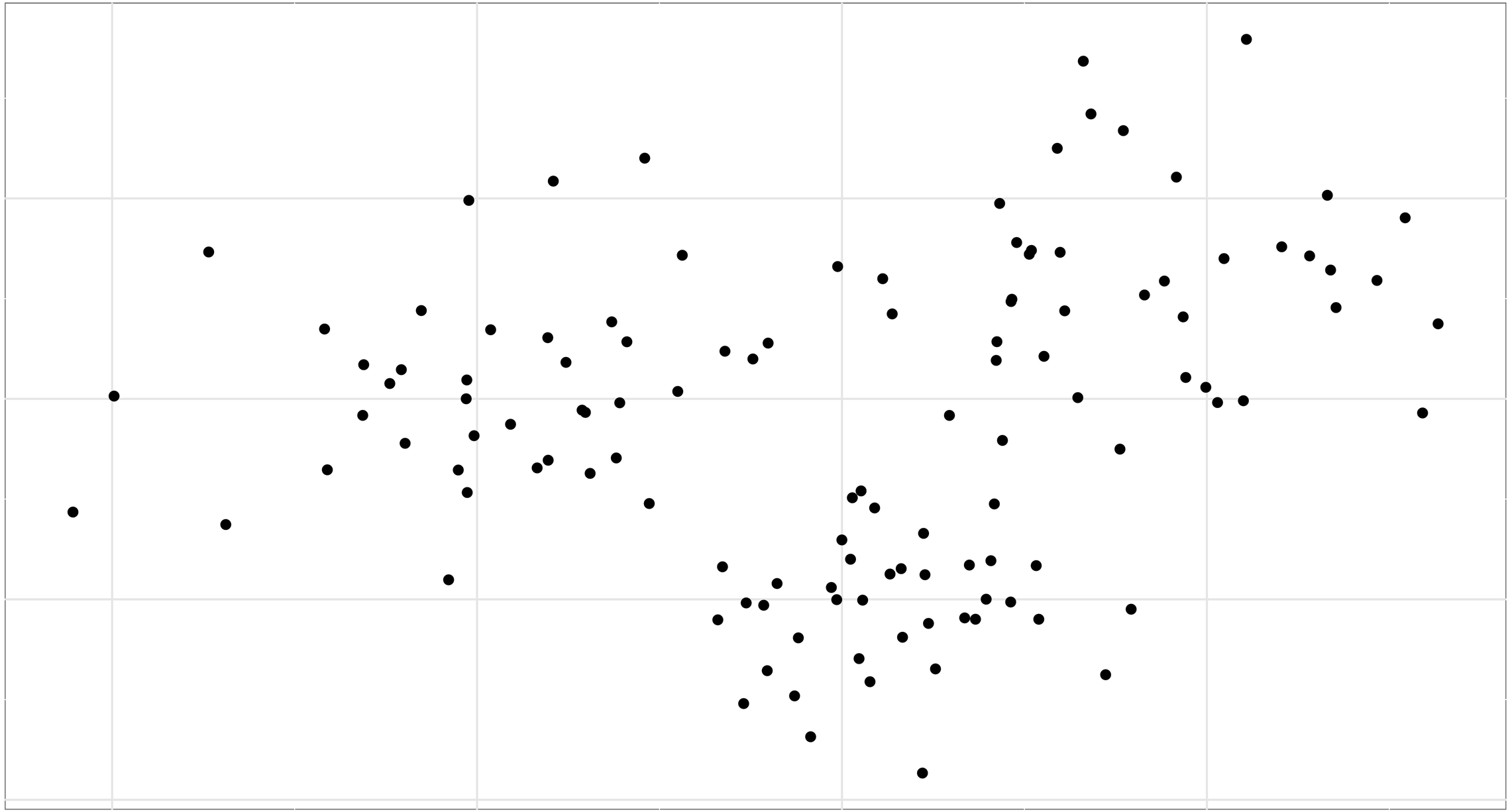
Clustering…



Clustering…



Clustering…

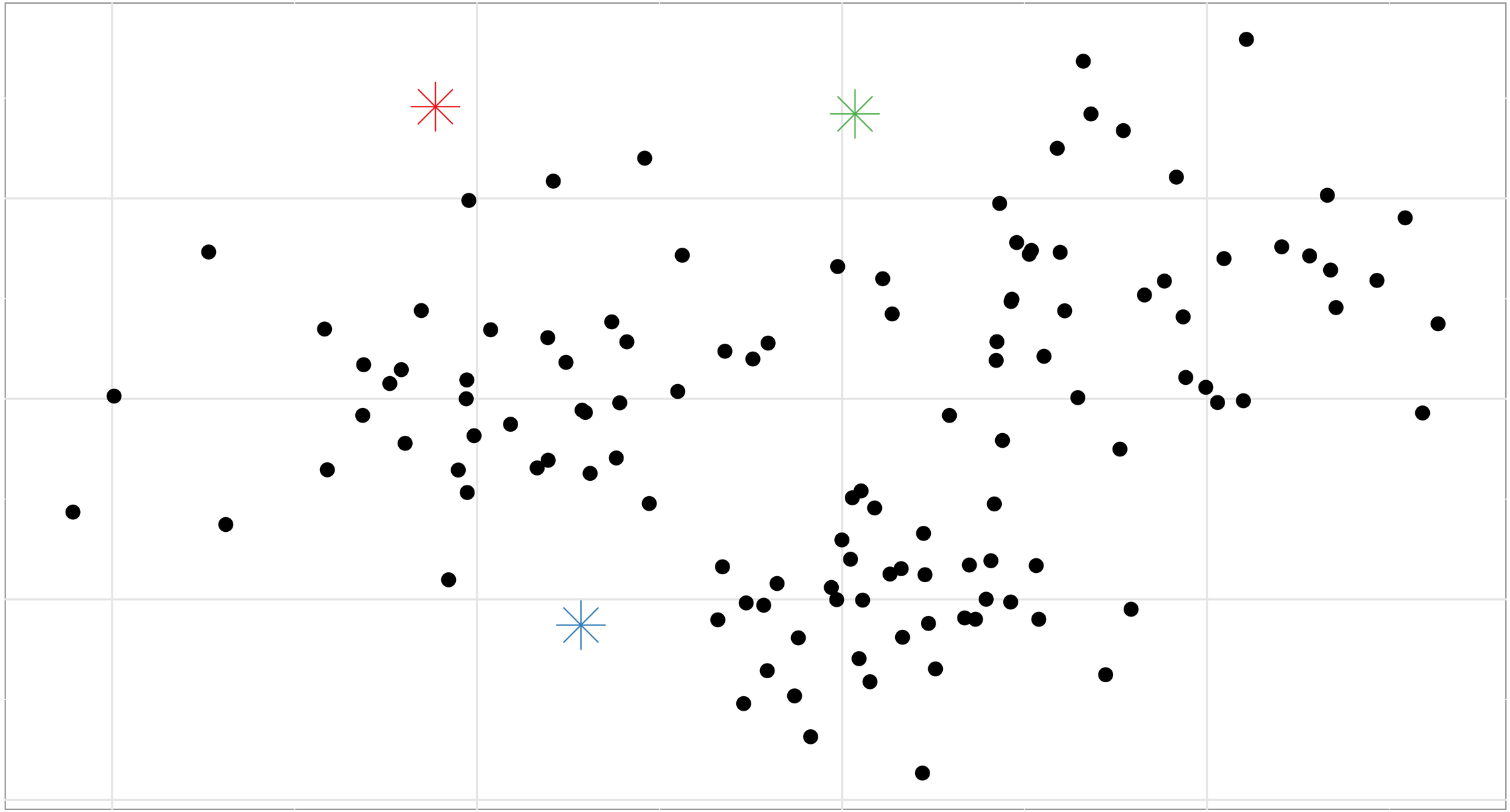


K-Means

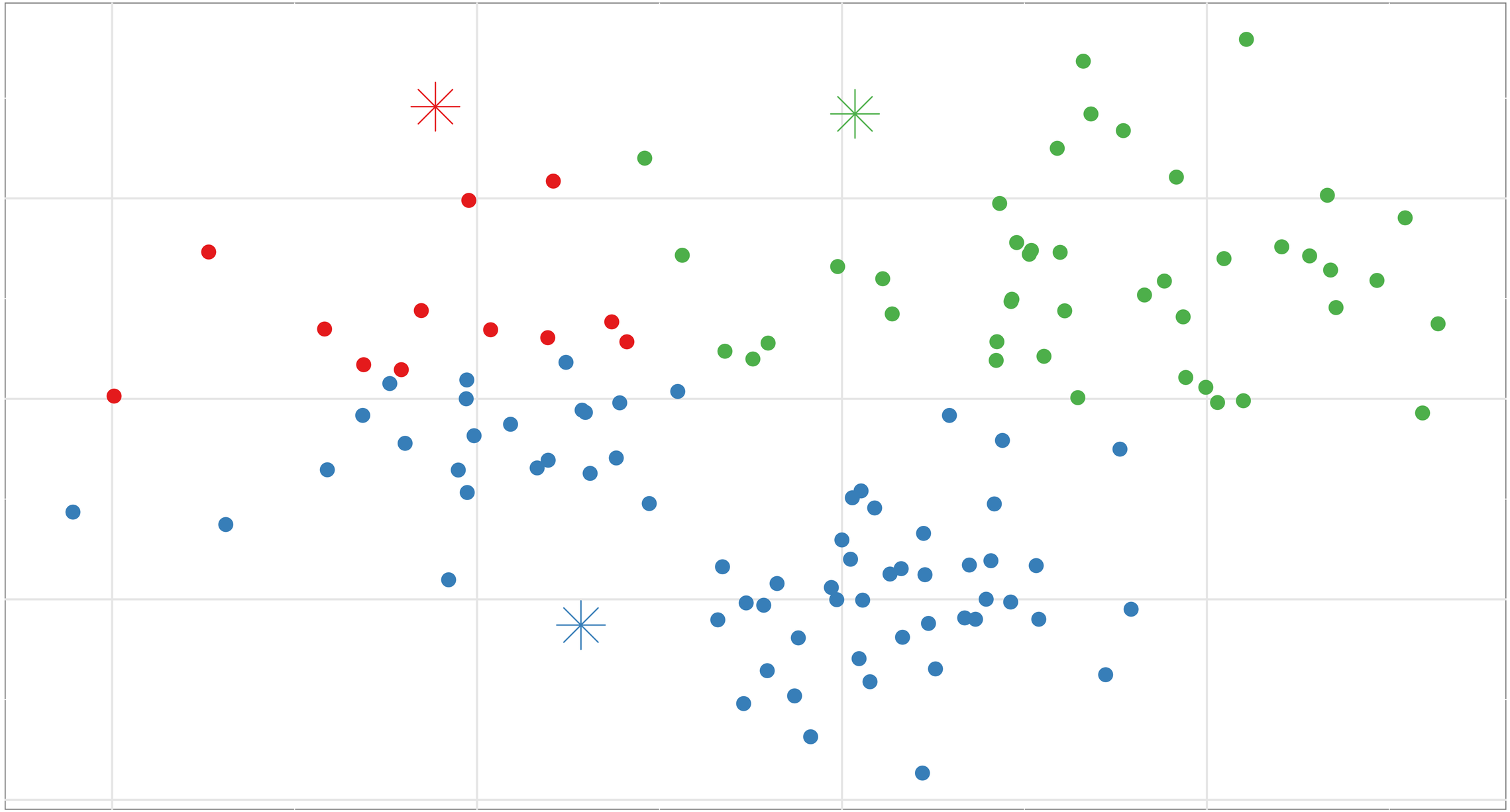
Before starting, pick the number of clusters, K

1. Pick K random centroids within data range
2. Assign each data point to the nearest centroid
3. Move centroid to center of assigned points
4. Repeat steps 2 and 3 until centroid stops shifting

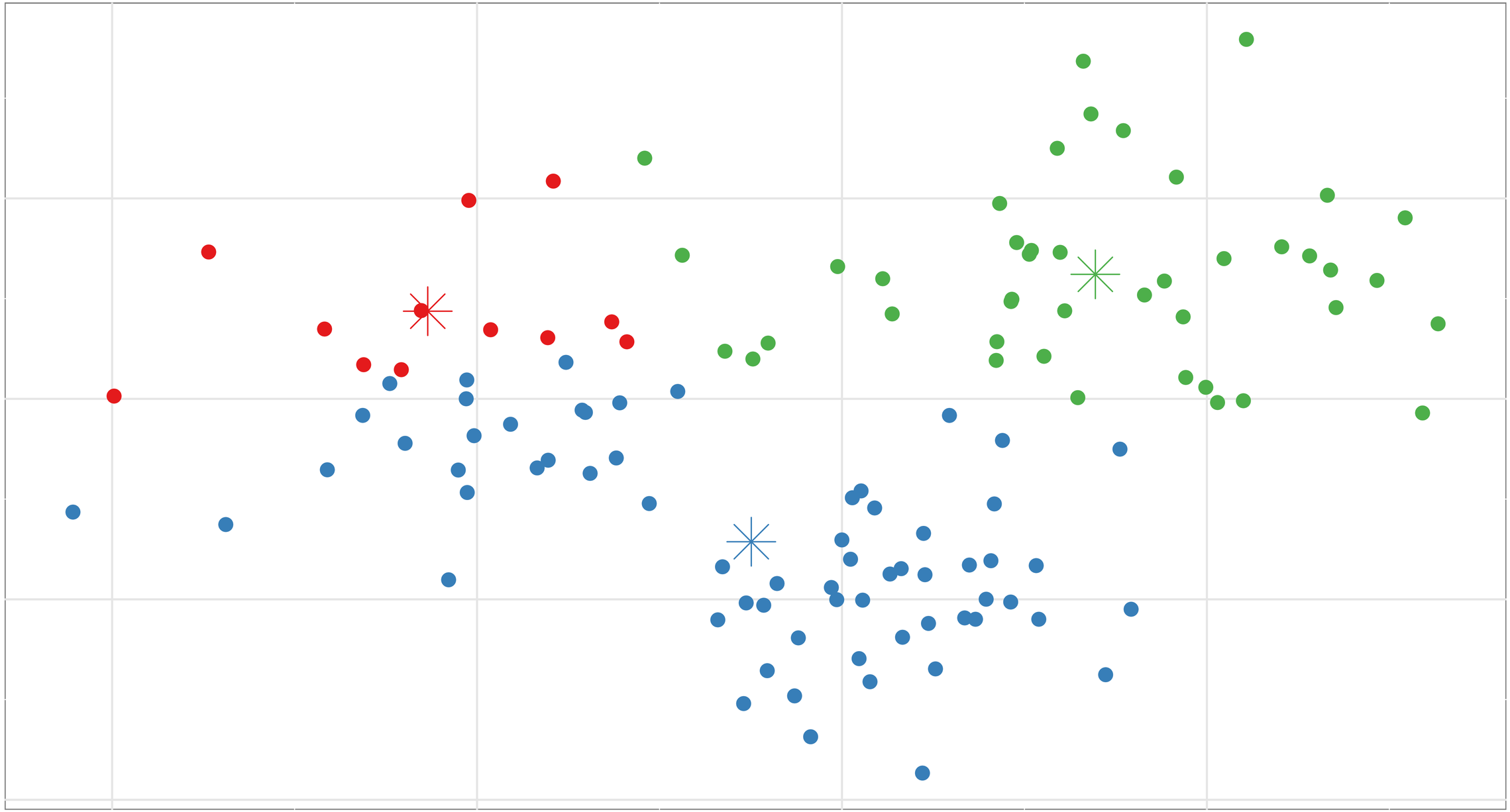
Step 1: Pick 3 random centroids within data range



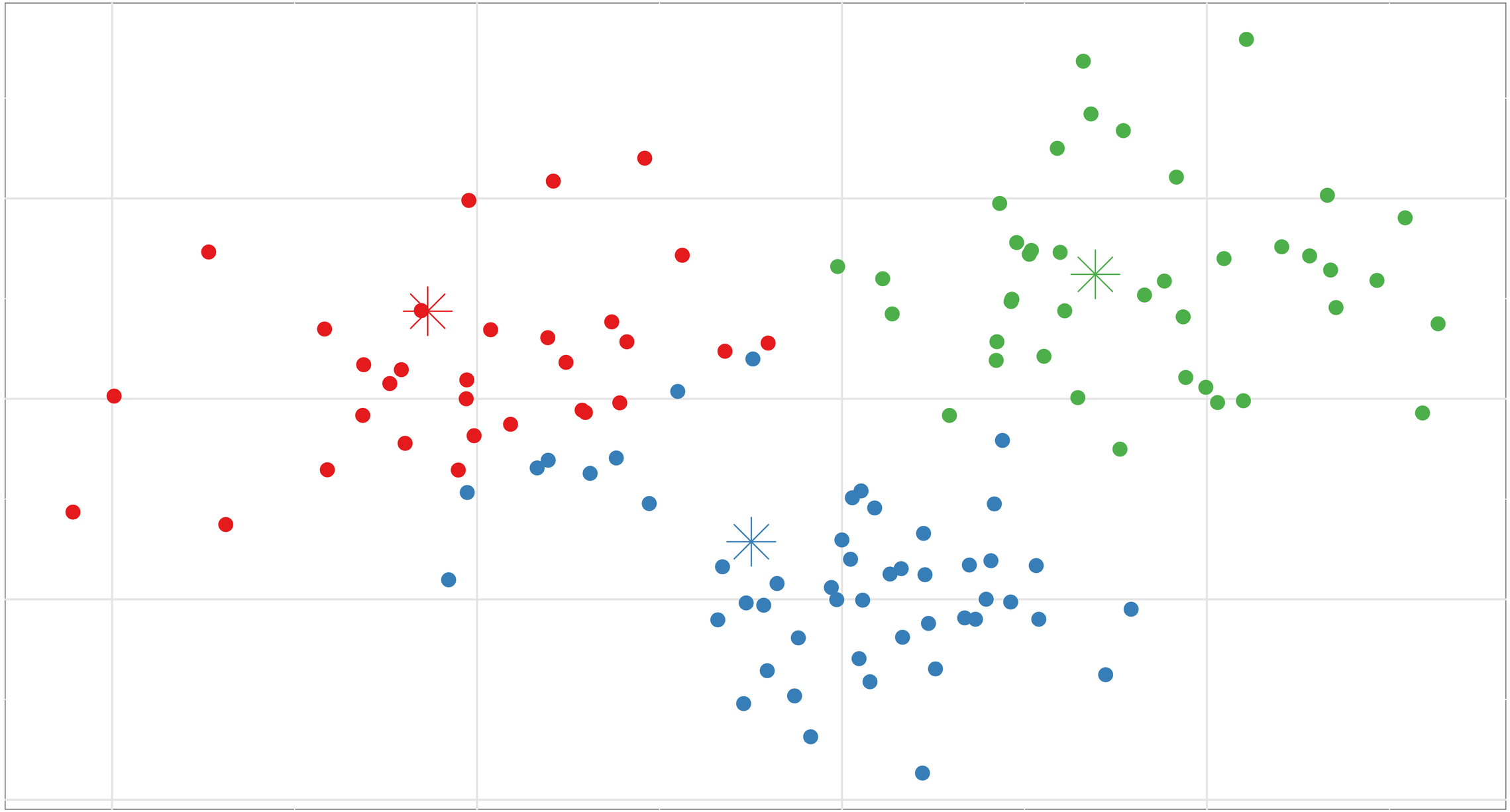
Step 2: Assign each data point to the nearest centroid (1)



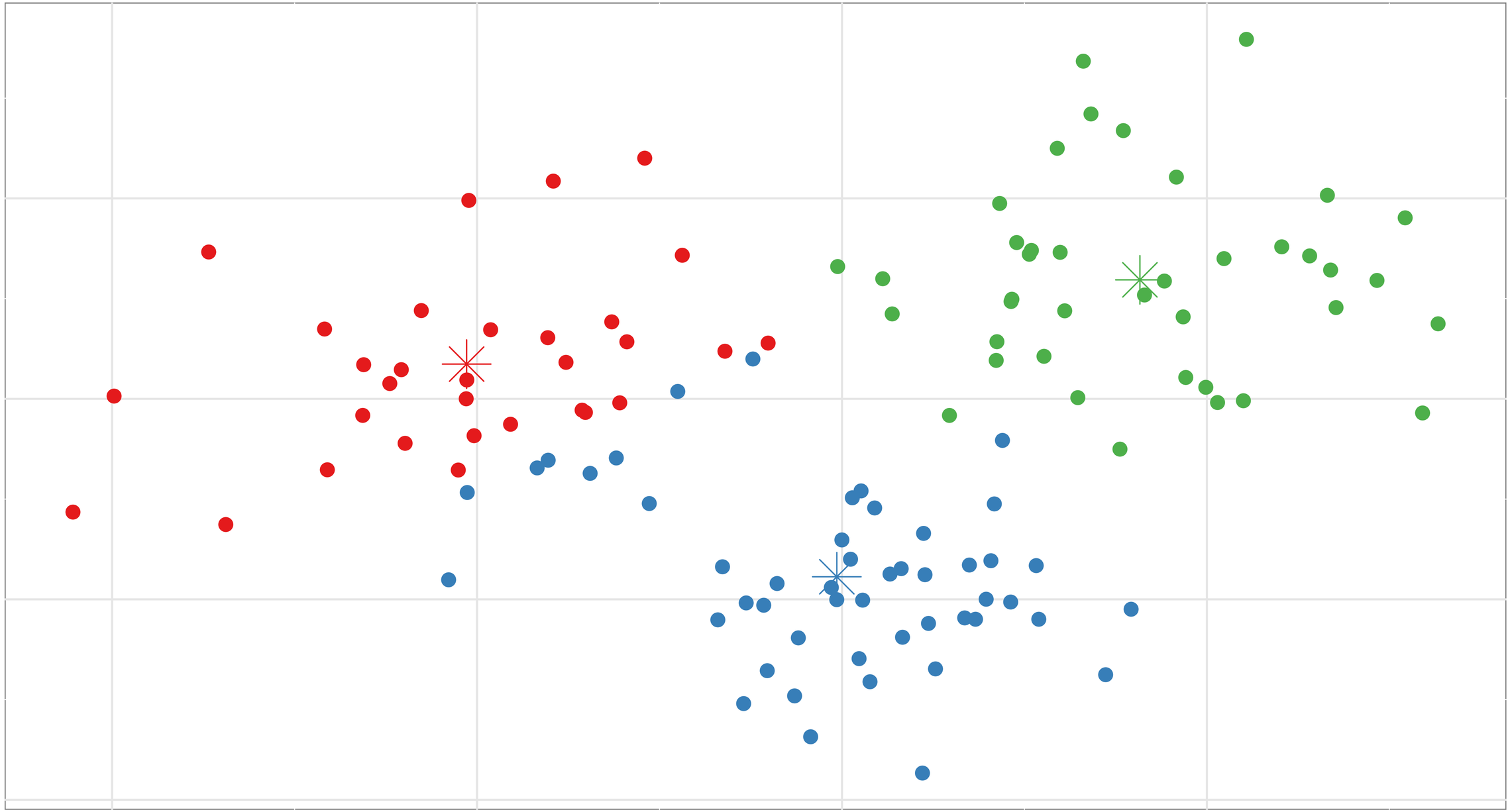
Step 3: Move centroid to center of assigned points (1)



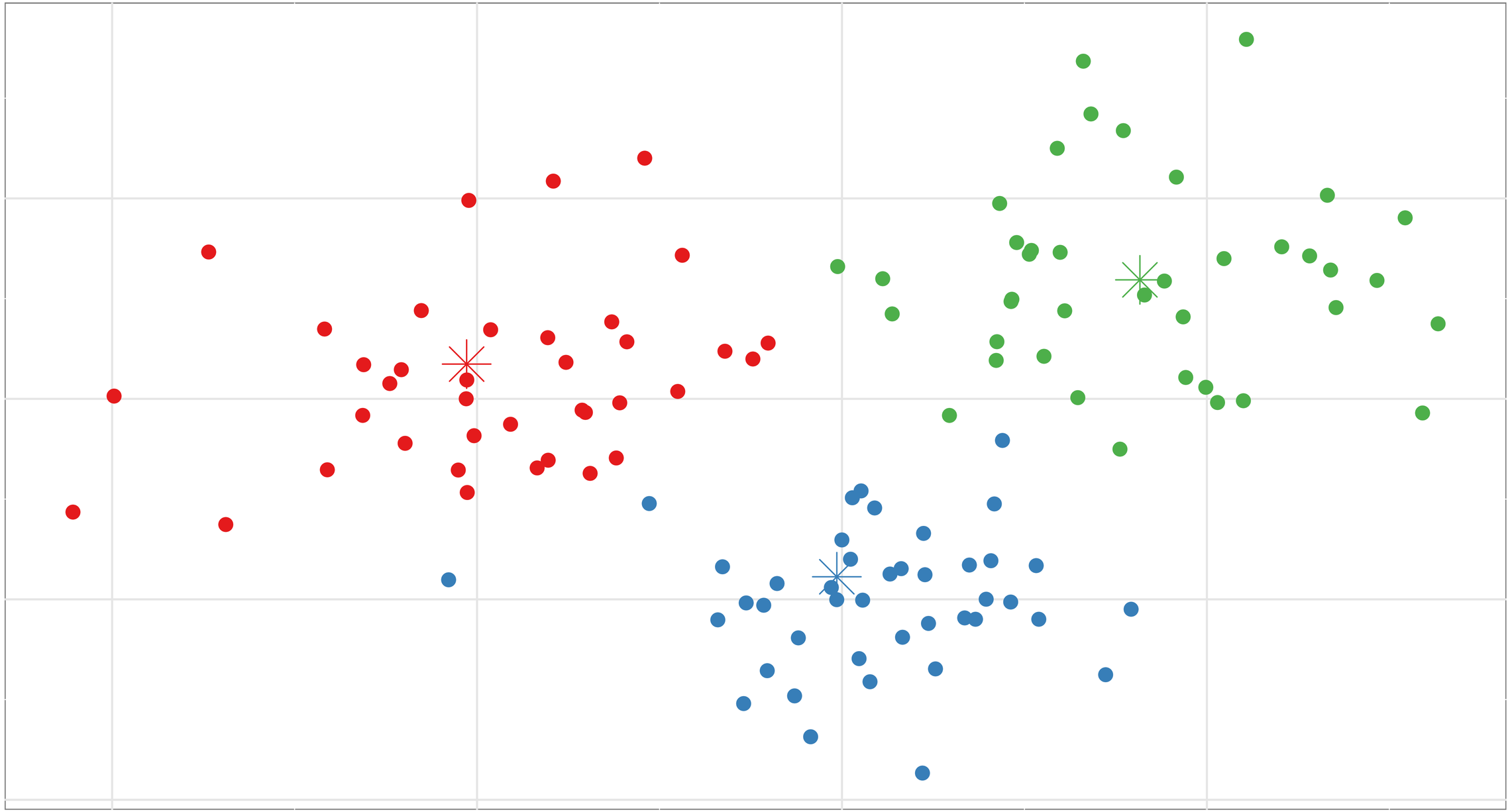
Step 2: Assign each data point to the nearest centroid (2)



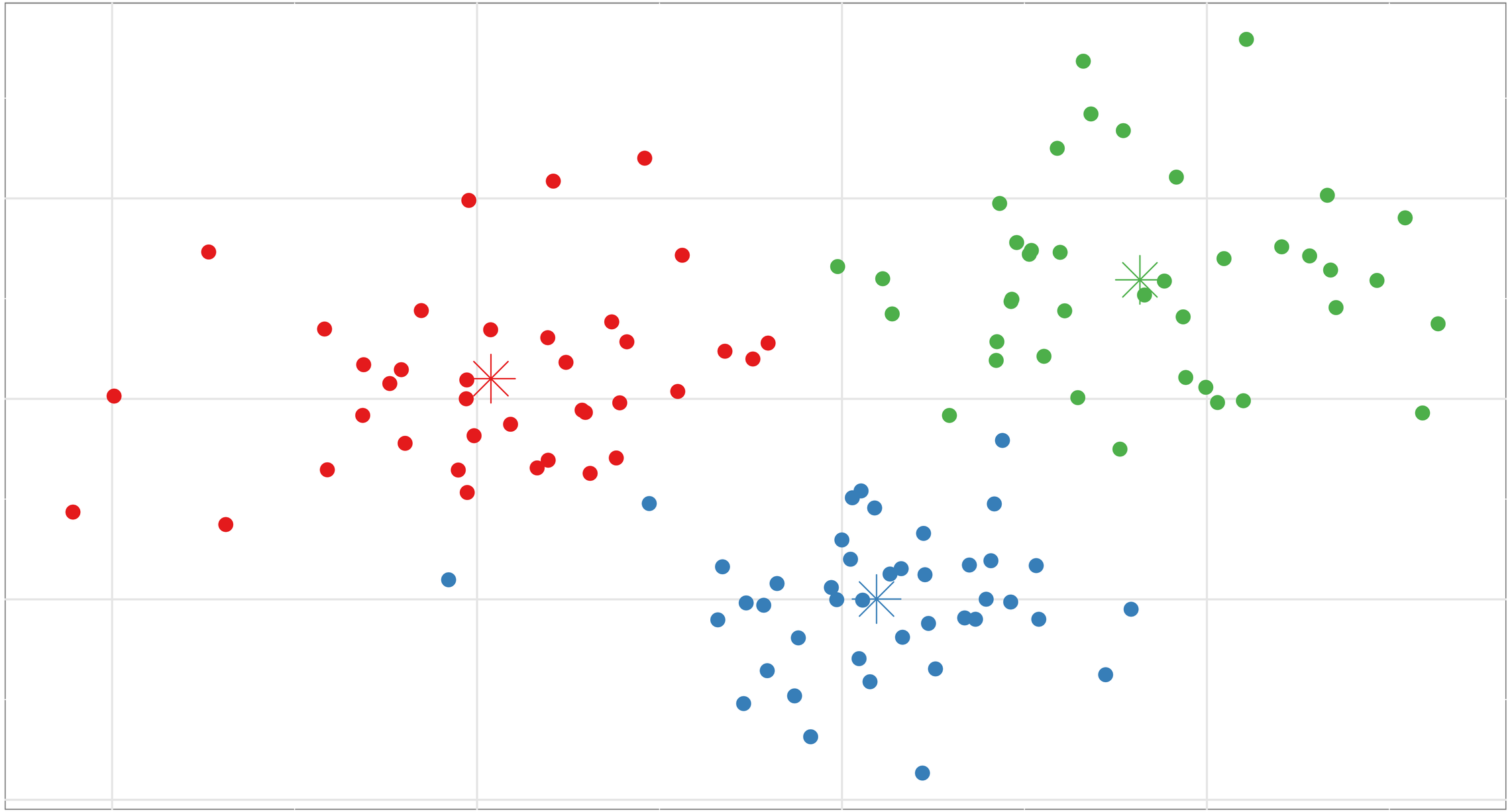
Step 3: Move centroid to center of assigned points (2)



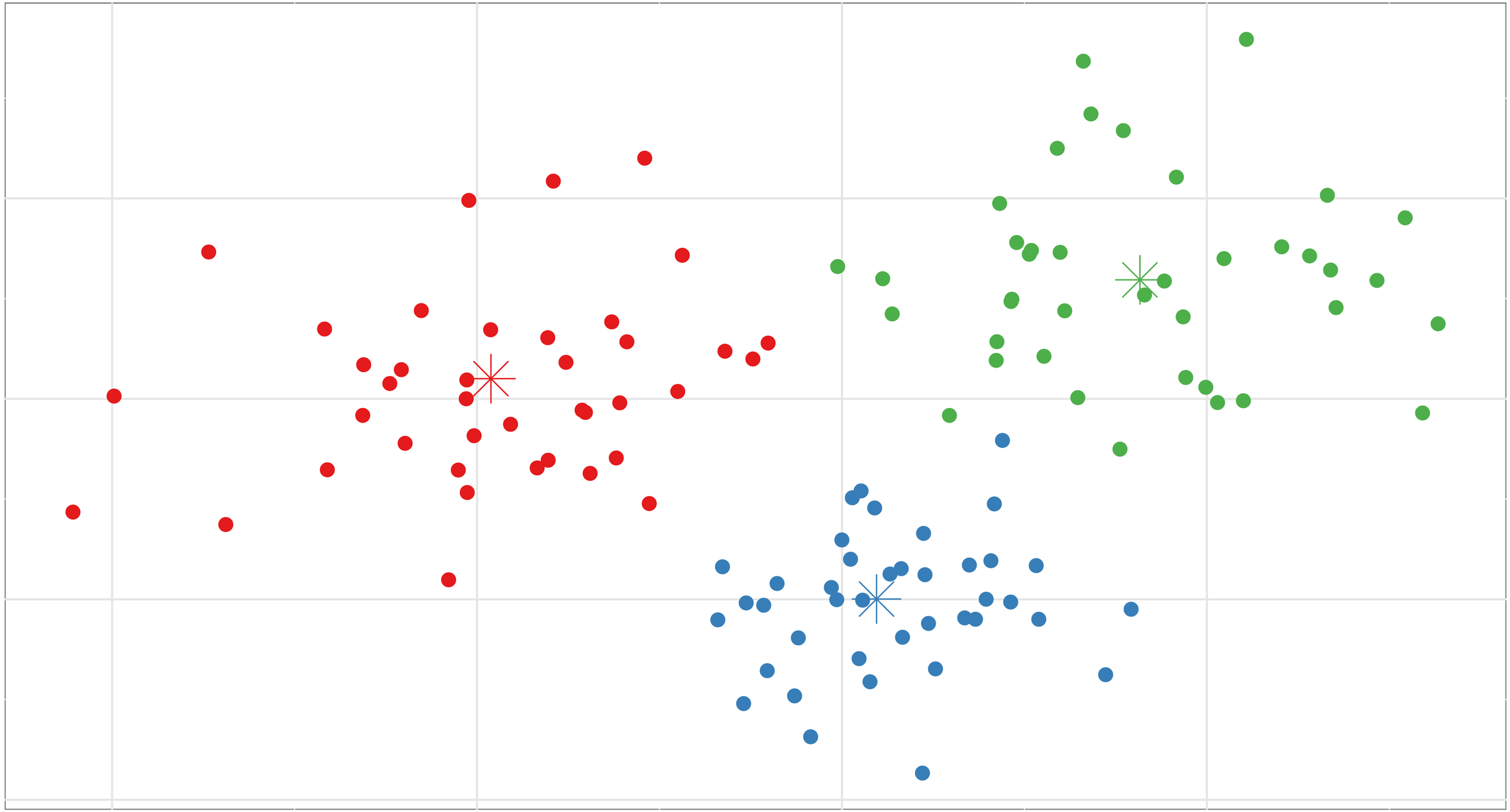
Step 2: Assign each data point to the nearest centroid (3)



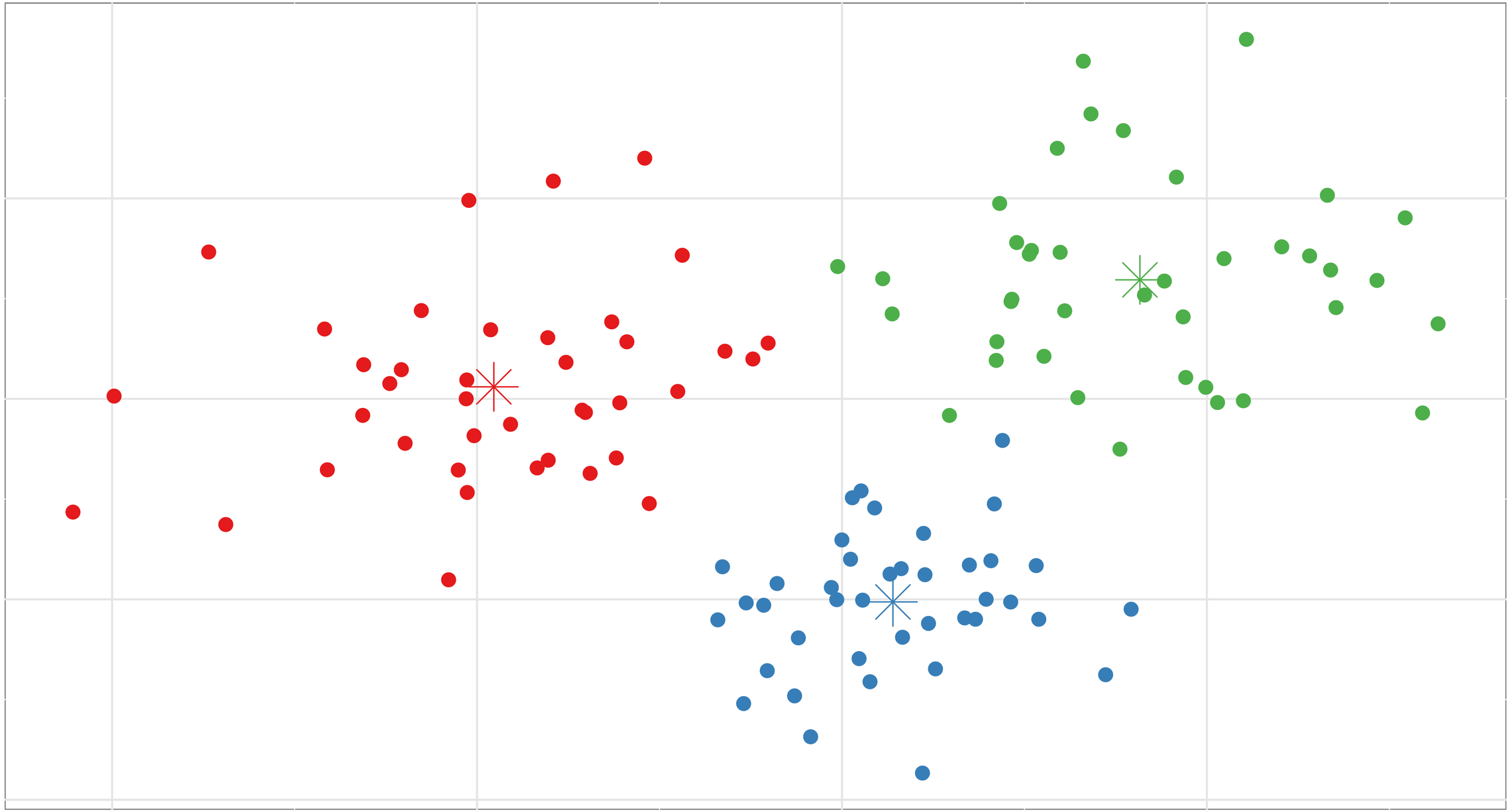
Step 3: Move centroid to center of assigned points (3)



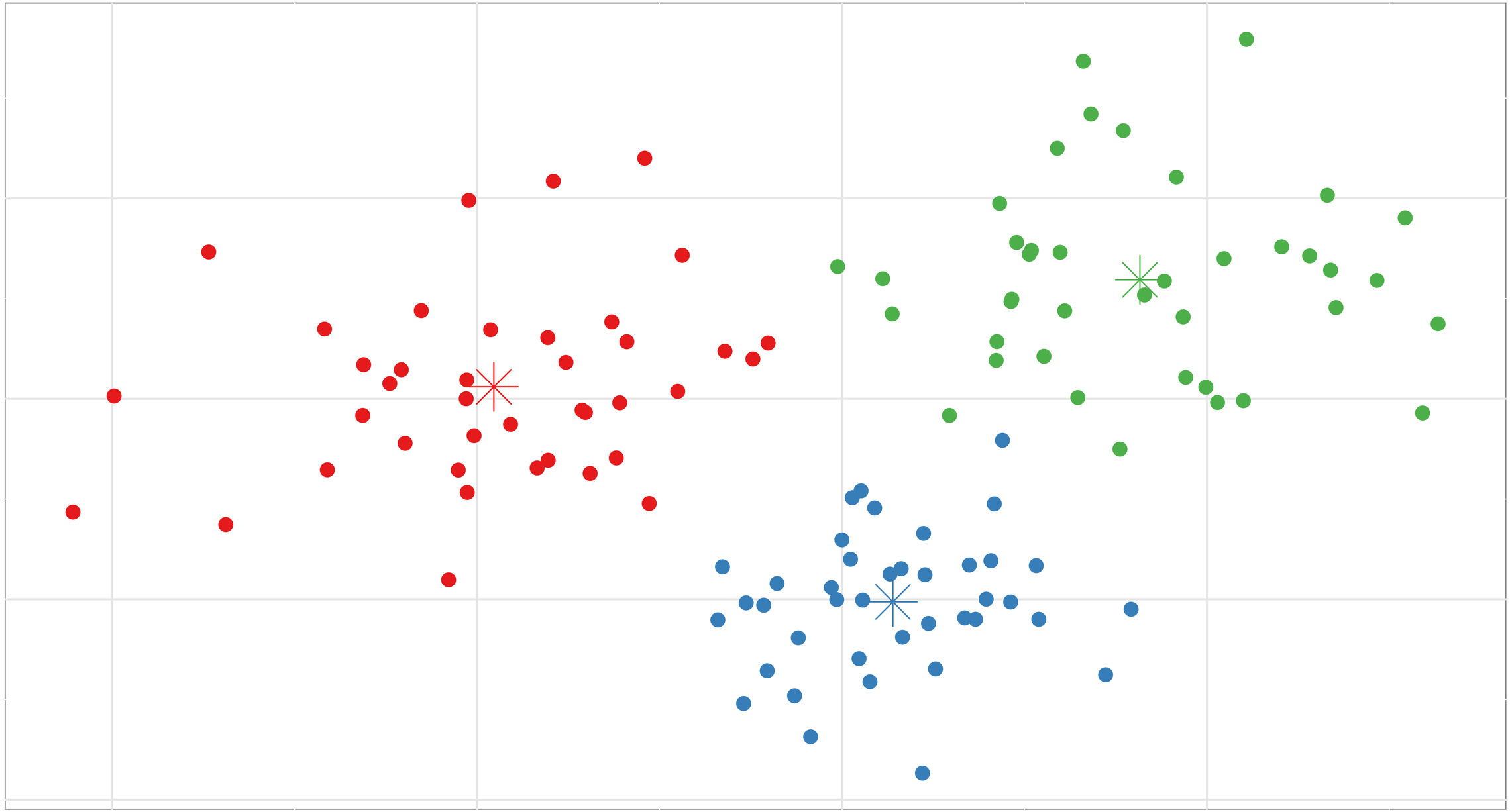
Step 2: Assign each data point to the nearest centroid (4)



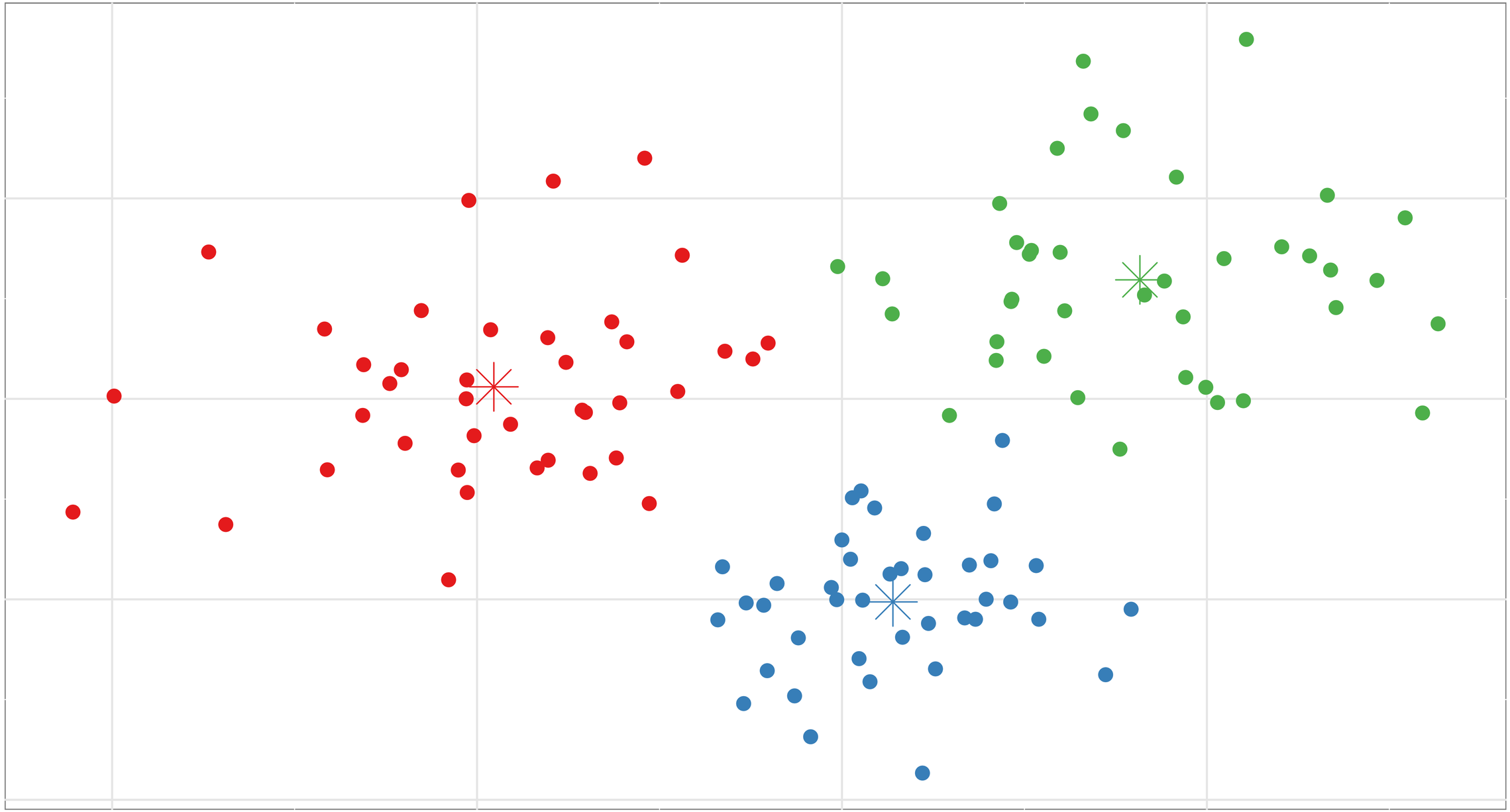
Step 3: Move centroid to center of assigned points (4)



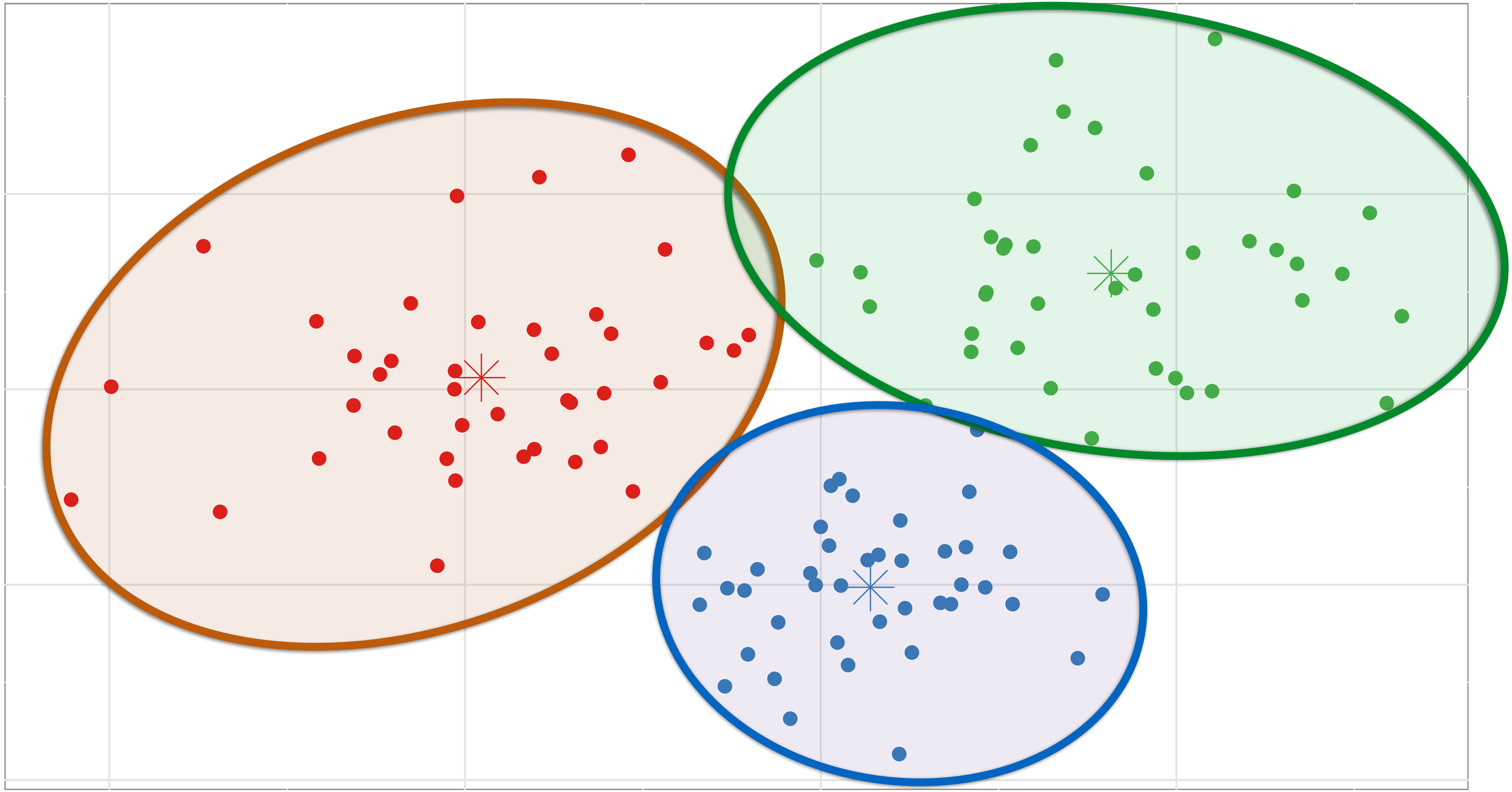
Step 2: Assign each data point to the nearest centroid (5)



Step 3: Move centroid to center of assigned points (5)



Step 3: Move centroid to center of assigned points (5)



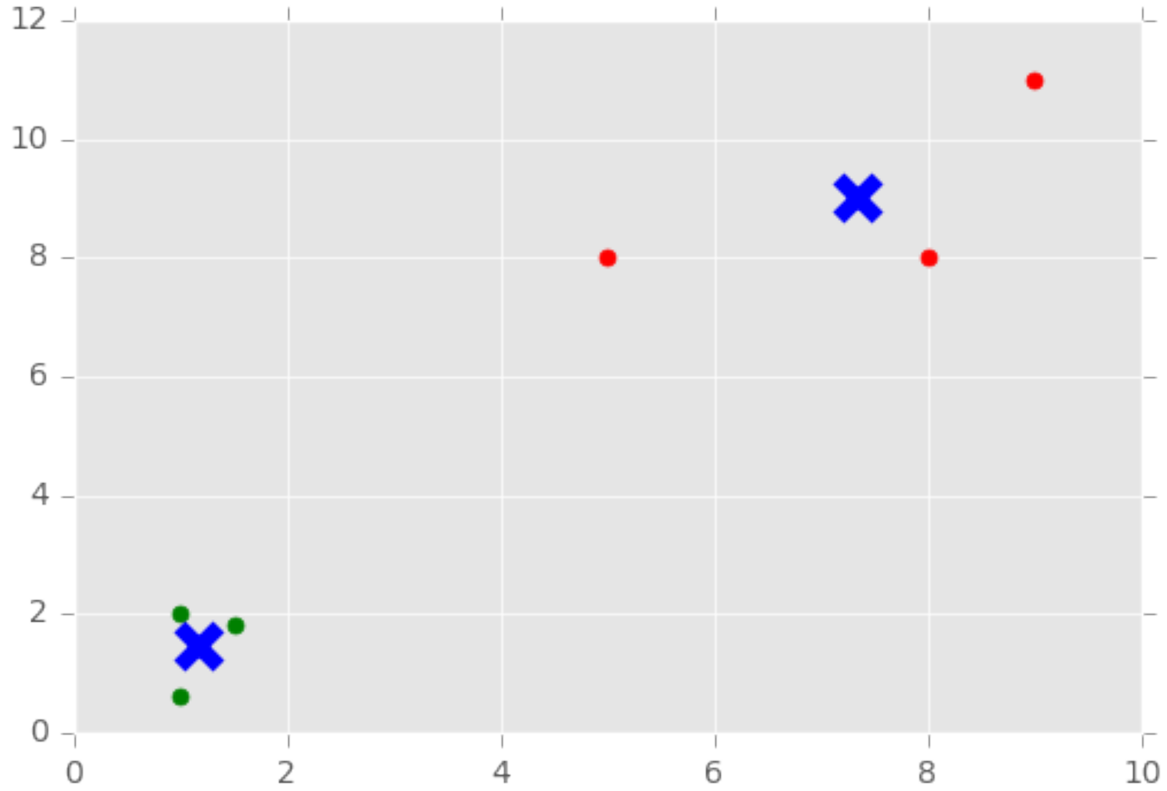
K-Means in practice (Python version)

Python

```
#Import from Scikit-learn
from sklearn.cluster import Means

kmeans = KMeans(n_clusters=2)
kmeans.fit(data)

centroids = kmeans.cluster_centers_
labels = kmeans.labels_f
```



Scikit-Learn Unsupervised Interface

Fundamentally, every unsupervised learning technique is coded in the same manner using Scikit-learn.

Step 1: Create the object and specify any tuning parameters.

```
means = KMeans( n_clusters = 4 )
```

Step 2: Call the .fit() method using your data.

```
means.fit(X)
```

K-Means in practice (R version)

```
> points <- data.frame(x=c(rnorm(50, mean=5), rnorm(50, mean=10)),
+                      y=c(rnorm(50, mean=5), rnorm(50, mean=10)))
> km.out <- kmeans(points, centers = 2)
> print(km.out)
K-means clustering with 2 clusters of sizes 50, 50

Cluster means:
         x         y
1 9.981532 10.073307
2 5.416039  4.917267

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
1 1 1 1 1
 [56] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Within cluster sum of squares by cluster:
[1]  73.6352 112.3886
 (between_SS / total_SS =  86.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
> points$cluster <- km.out$cluster
```

K-Means: Got a problem with it?

Before starting, pick the number of clusters, K

1. Pick K random centroids within data range
2. Assign each data point to the nearest centroid
3. Move centroid to center of assigned points
4. Repeat steps 2 and 3 until centroid stops shifting

K-Means: Got a problem with it?

Before starting, pick the number of clusters, K

Subjective

1. Pick K random centroids within data range

Not Repeatable

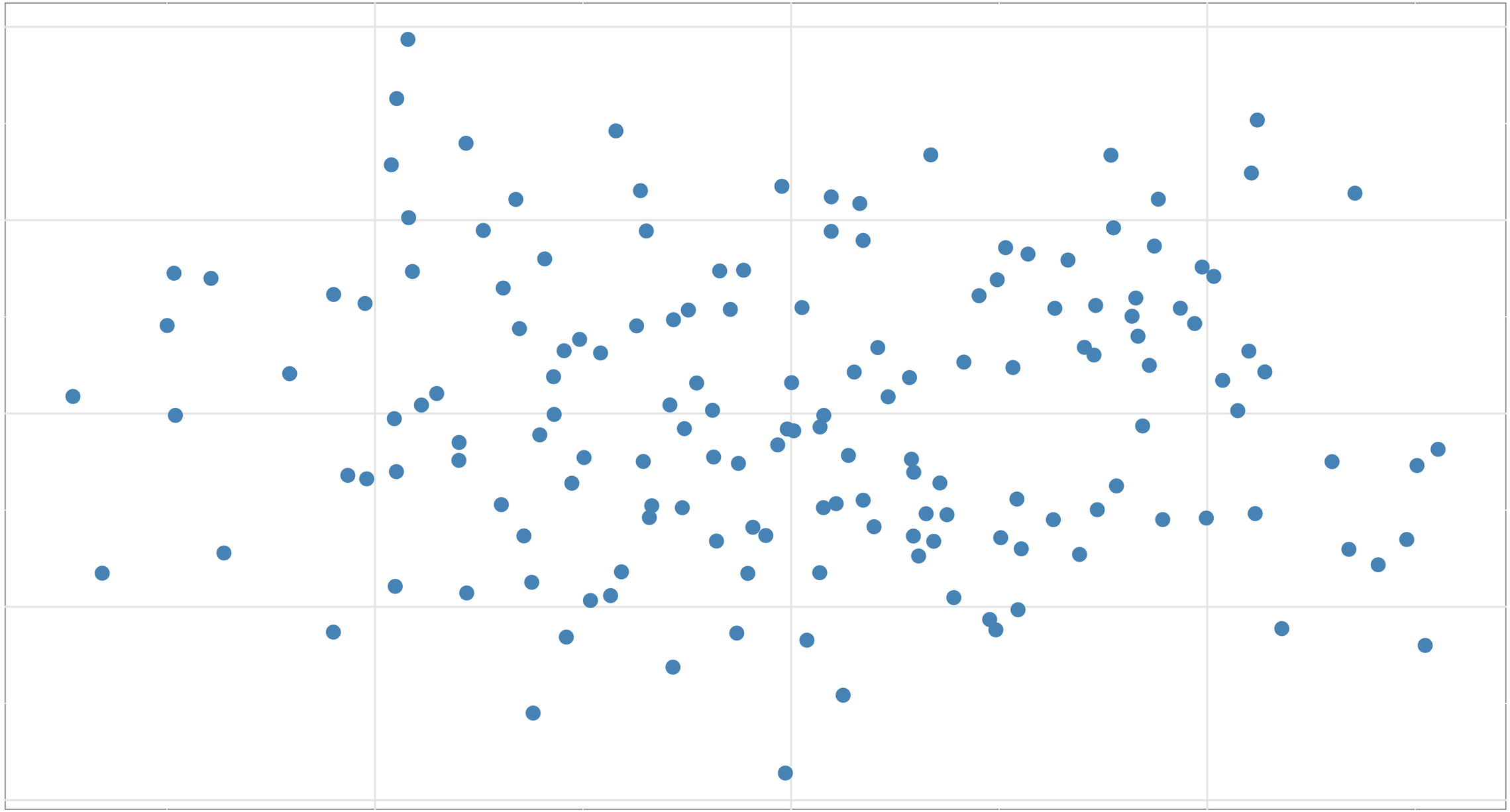
2. Assign each data point to the nearest centroid

Sensitive to Scale

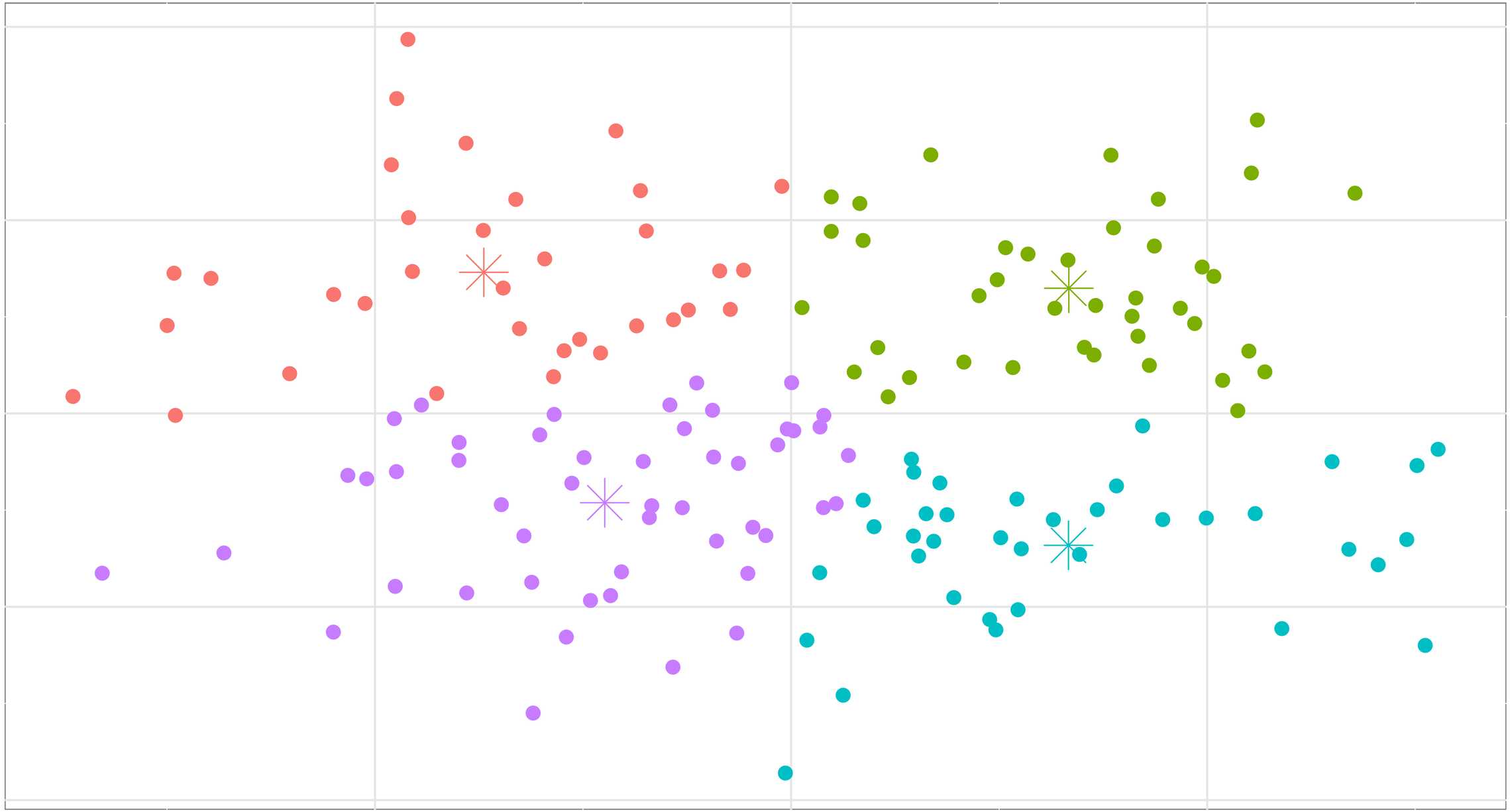
3. Move centroid to center of assigned points

4. Repeat steps 2 and 3 until centroid stops shifting

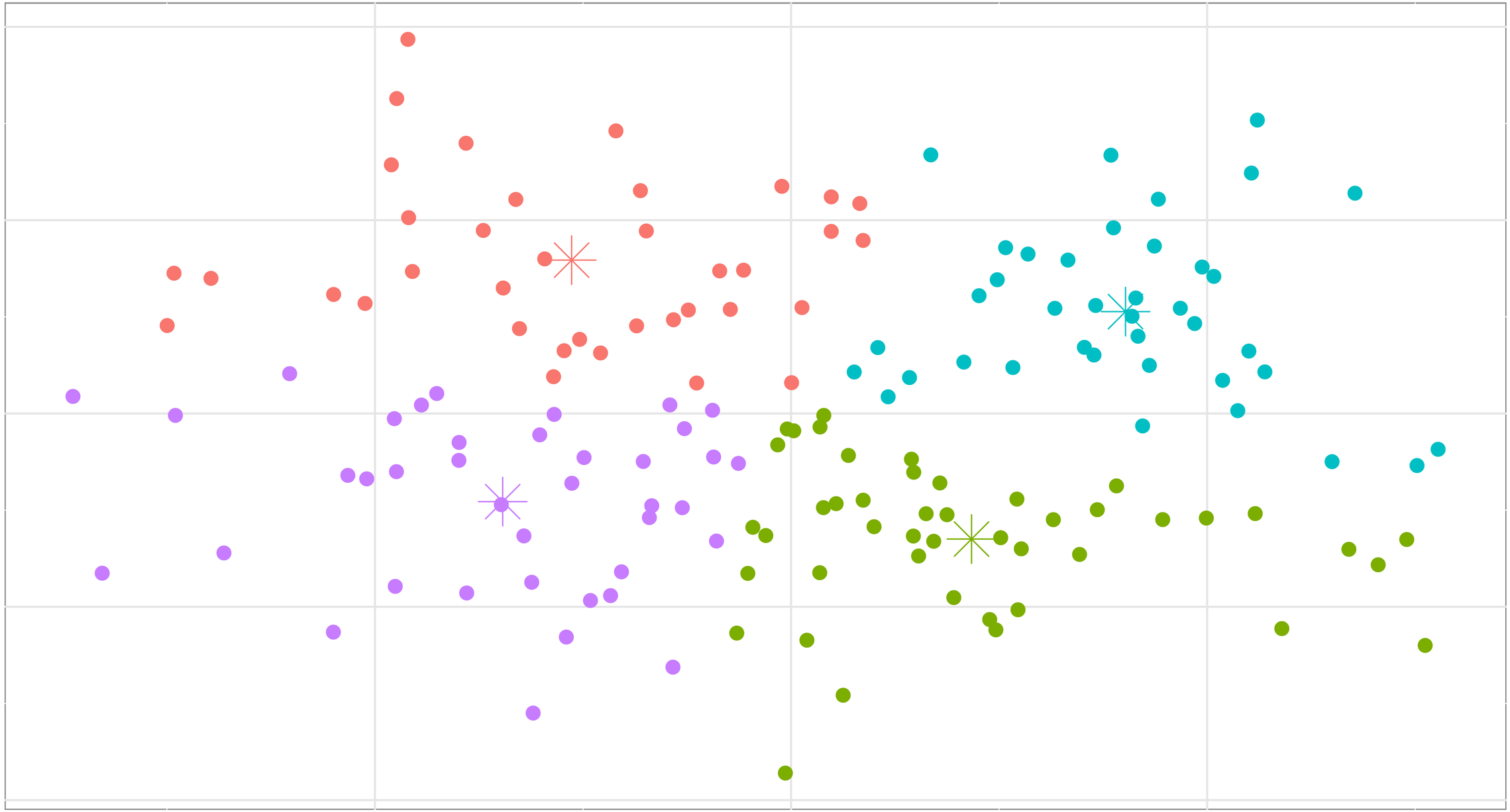
How many clusters?



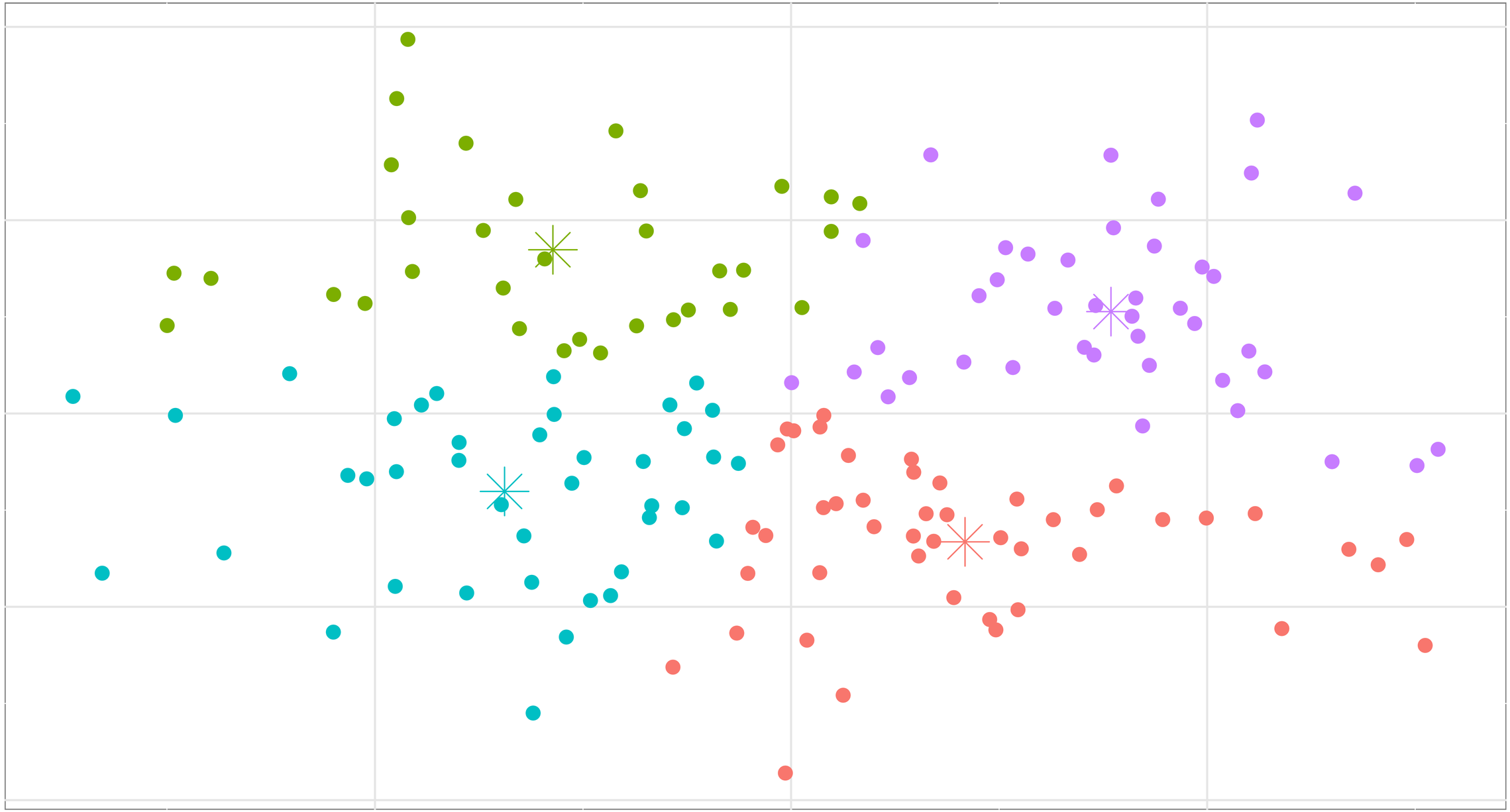
Random Start…



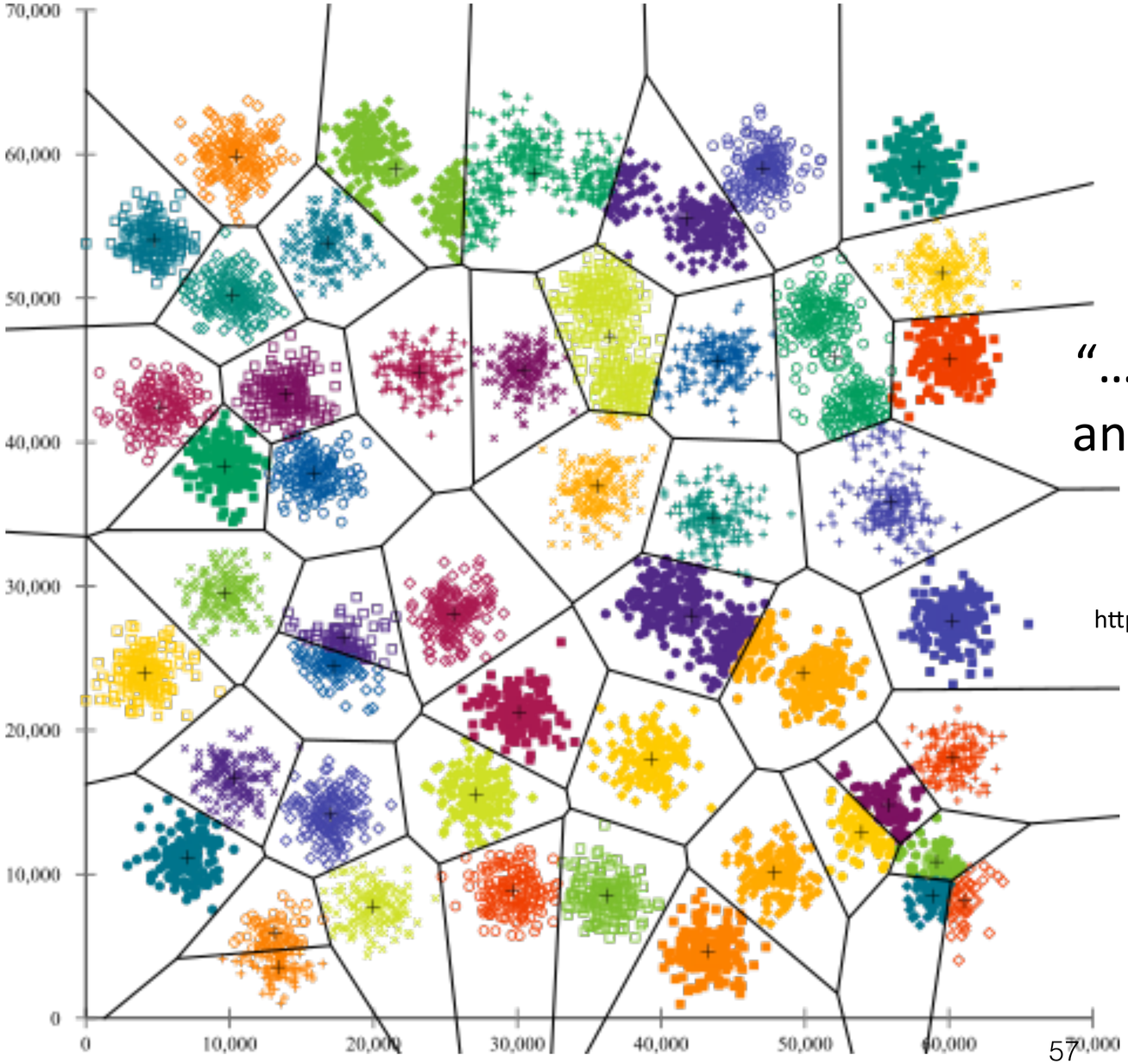
Random Start…



Random Start…



K-Means



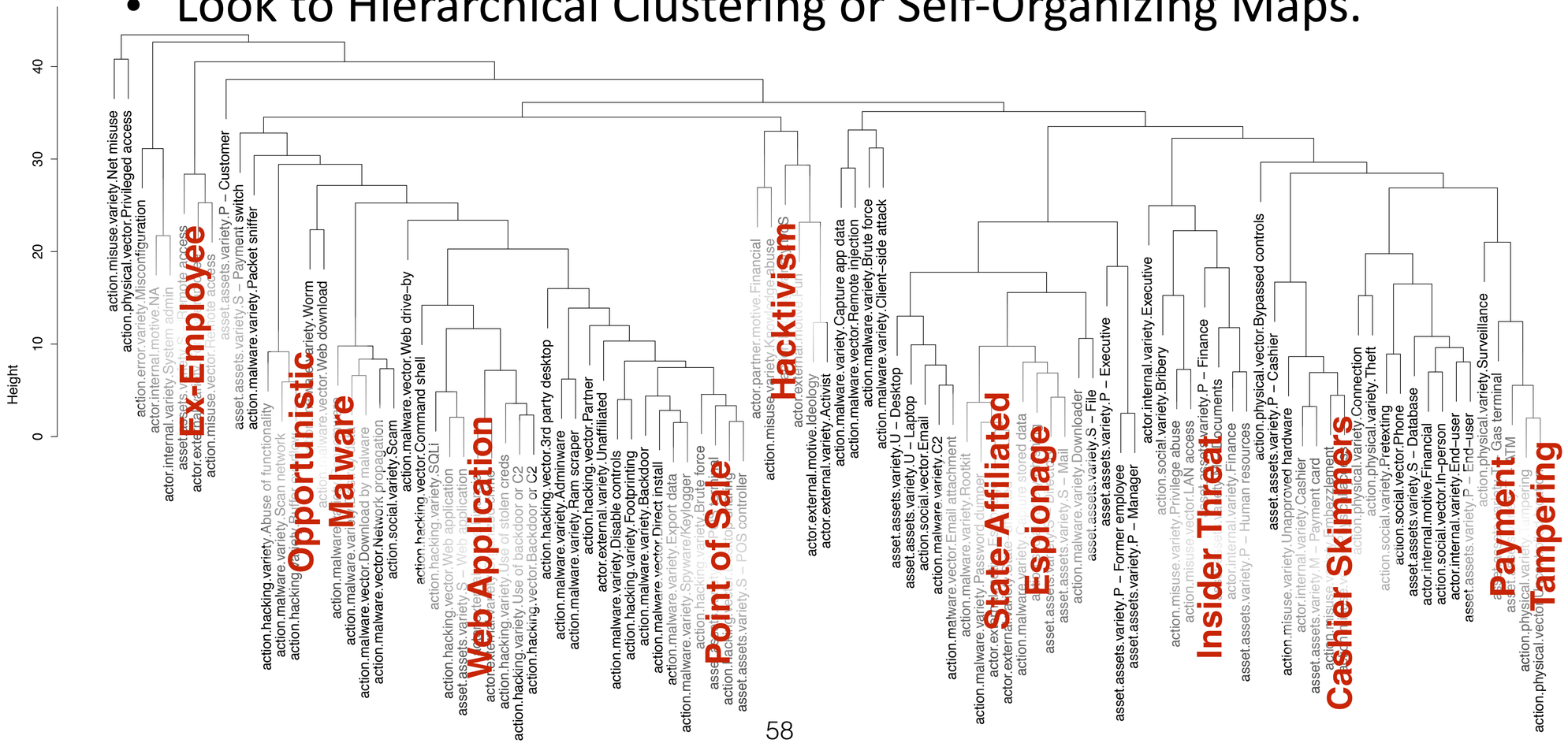
"…it's too easy to throw k-means on your data, and nevertheless get a result out (that is pretty much random, but you won't notice)."

— Anony-Mousse

http://stats.stackexchange.com/questions/133656/how-to-understand-the-drawbacks-of-k-means

After K-Means

- K-Means has many, many alternatives and corrections to the core shortcomings.
- Look to Hierarchical Clustering or Self-Organizing Maps.



Other Clustering and Dimensionality Reduction Algorithms:

- **DBSCAN - DBSCAN** stands for **D**ensity**-B**ased **S**patial **C**lustering of **A**pplications with **N**oise. Another clustering algorithm, but you specify a maximum distance between clusters and DBSCAN will find the optimum number of clusters.

- **PCA** - Principal Components Analysis, converts data into a set of values that are linearly uncorrelated. Each component is a projection of the highest possible variance.

- **t-SNE** - t-Distributed Stochastic Neighbor Embedding. Relatively new (2008+) approach, reduces to two or three dimensions

DBSCAN

DBSCAN does not allow you to specify how many clusters you want. Instead, you specify 2 parameters:

- **ϵ (epsilon)**: This is the maximum distance between two points to allow them to be neighbors

- **min_samples**: The number of neighbors a given point is allowed to have to be able to be part of a cluster

Any points that don't satisfy the criteria of being close enough to other points are labeled outliers and all fall into a single "cluster" (their cluster label by default is -1).

DBSCAN

DBSCAN works as follows:

1. Choose an arbitrary starting point in your dataset that has not been seen.

2. Retrieve this point's ϵ-neighborhood (all points that are within a distance $_{e}\epsilon$ from it), and if it contains at least ***min_samples**, a cluster is started.

3. Otherwise, the point is labeled as an outlier (-1). Note: This point might later be found in a sufficiently sized ε-environment of a different point and hence be made part of a cluster.

4. If a point is found to be a dense part of a cluster, its ϵ-neighborhood is also part of that cluster. All points that are found within the ϵ-neighborhood are added, as is their own ϵ-neighborhood when they are also dense.

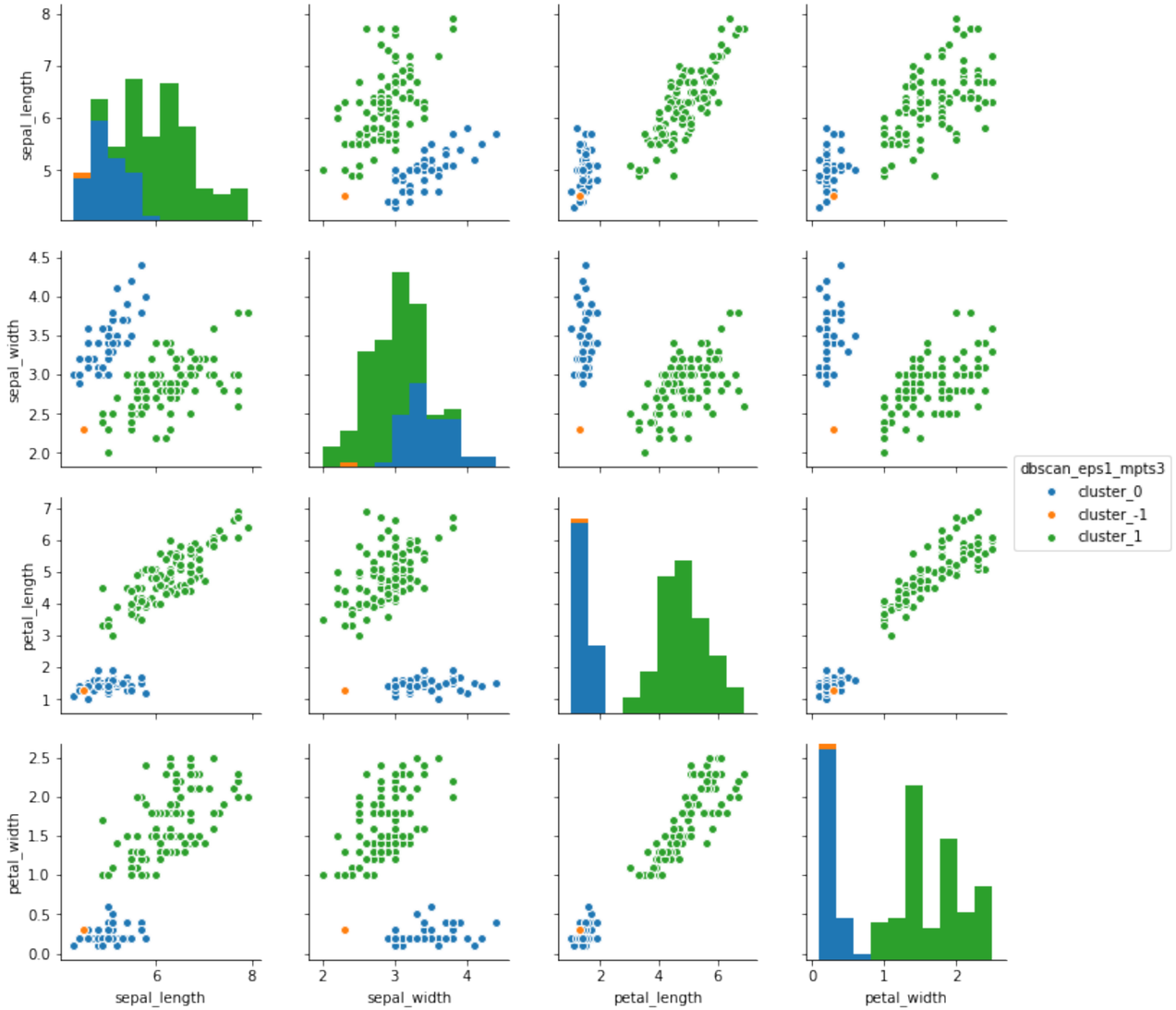
5. Continue until the density-connected cluster is completely found.

6. Find a new unvisited point to process, and repeat.

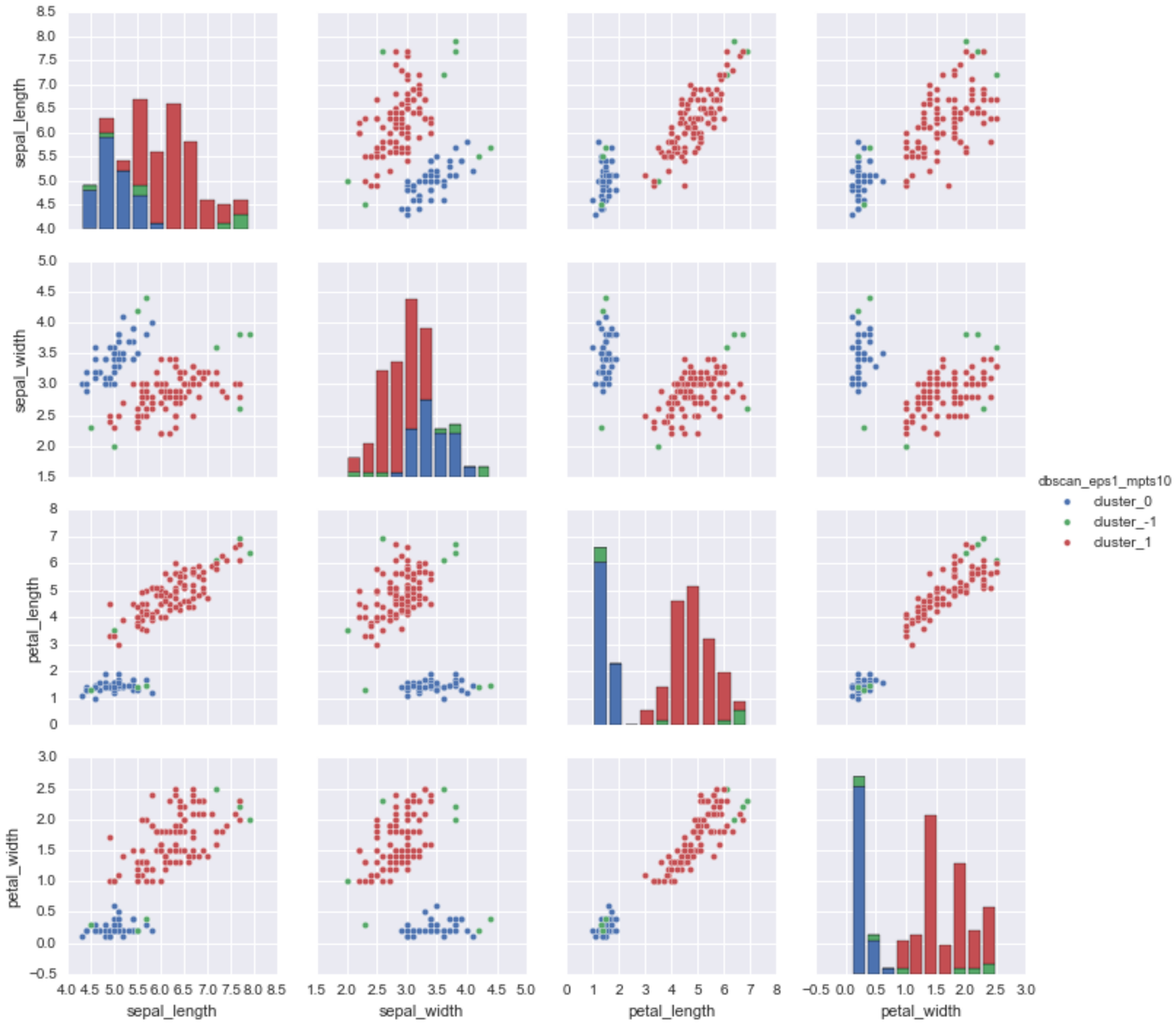
DBSCAN in Python

```
db = DBSCAN(eps=1, min_samples=3)
db.fit(data)
```

DBSCAN in Python

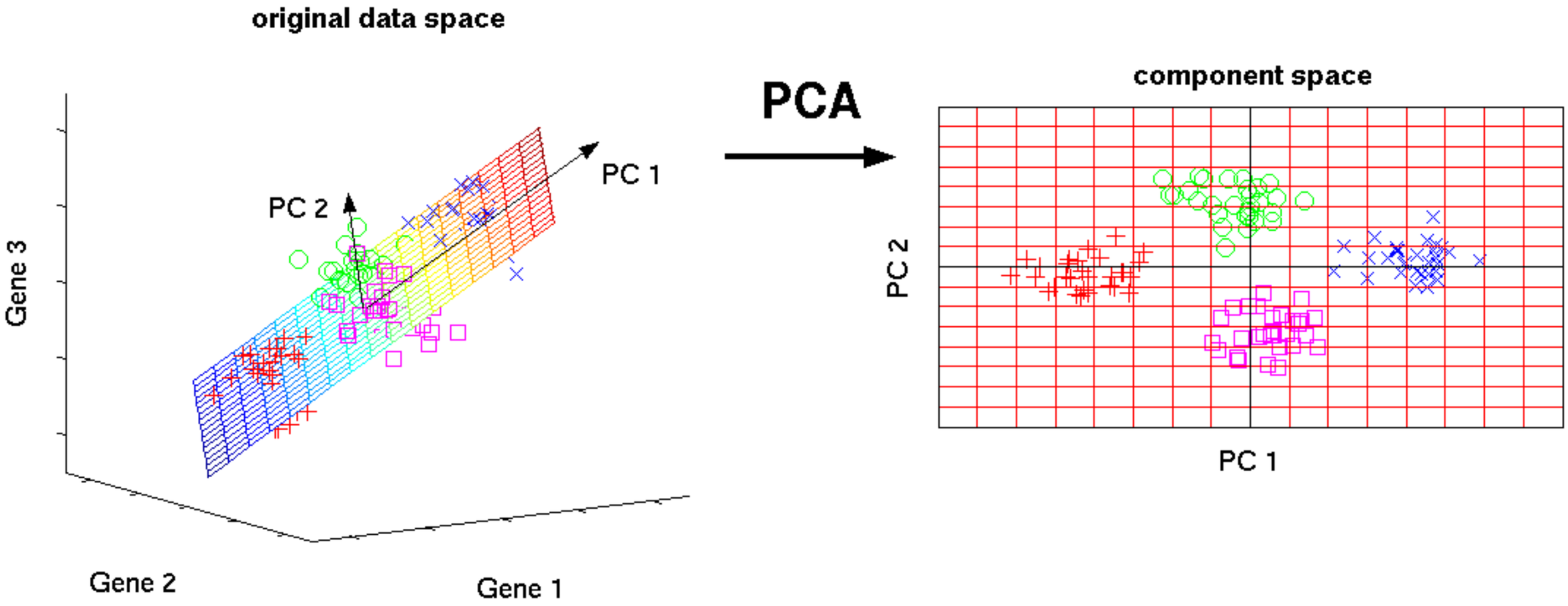


DBSCAN in Python



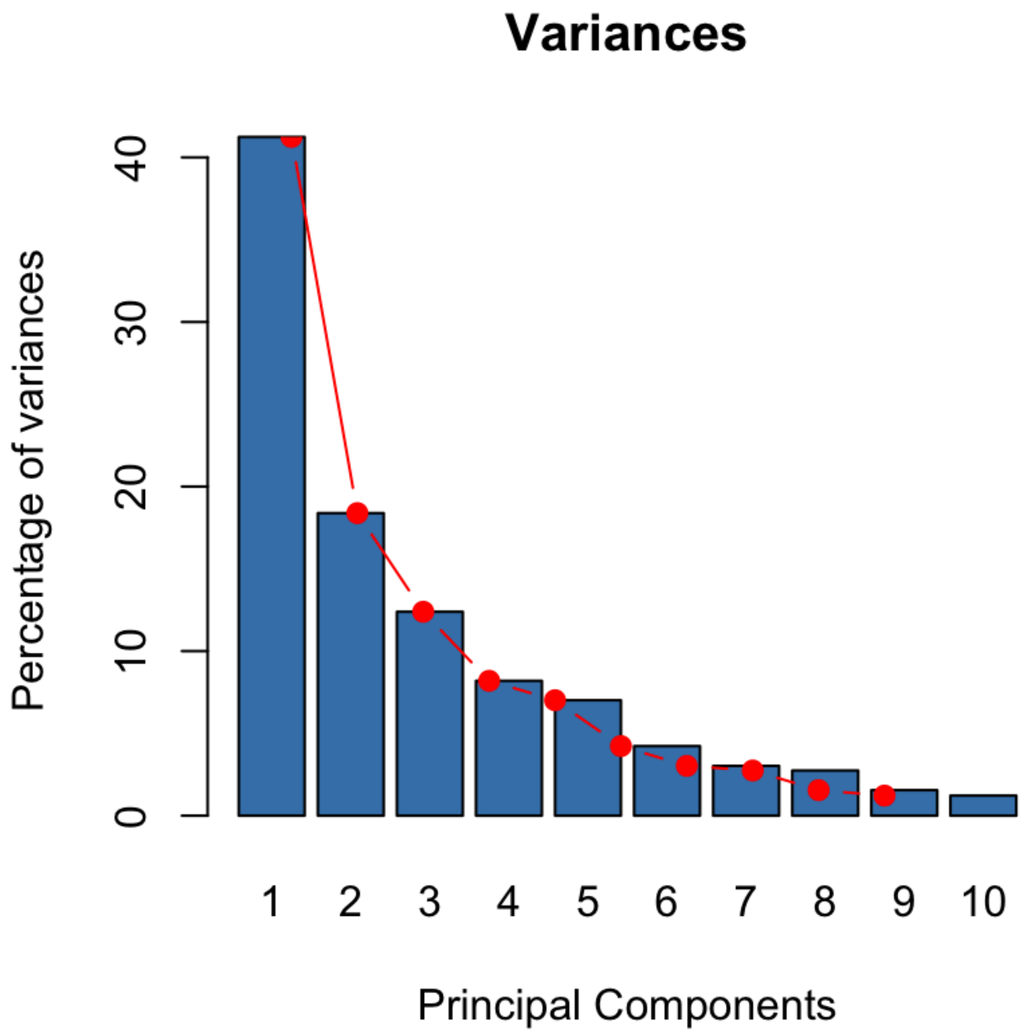
Principal Components Analysis

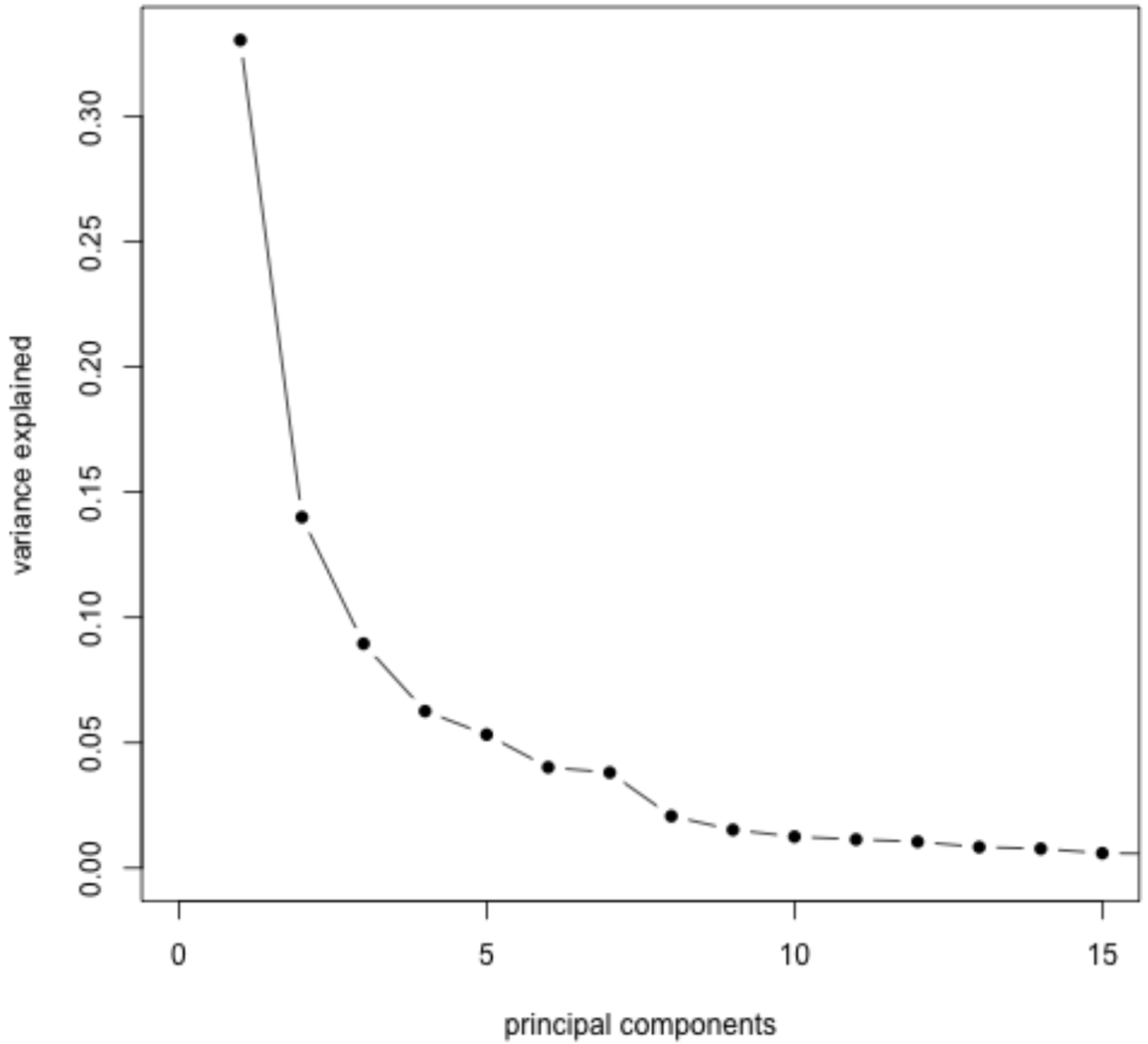
Example: Reducing 3D down to 2D:



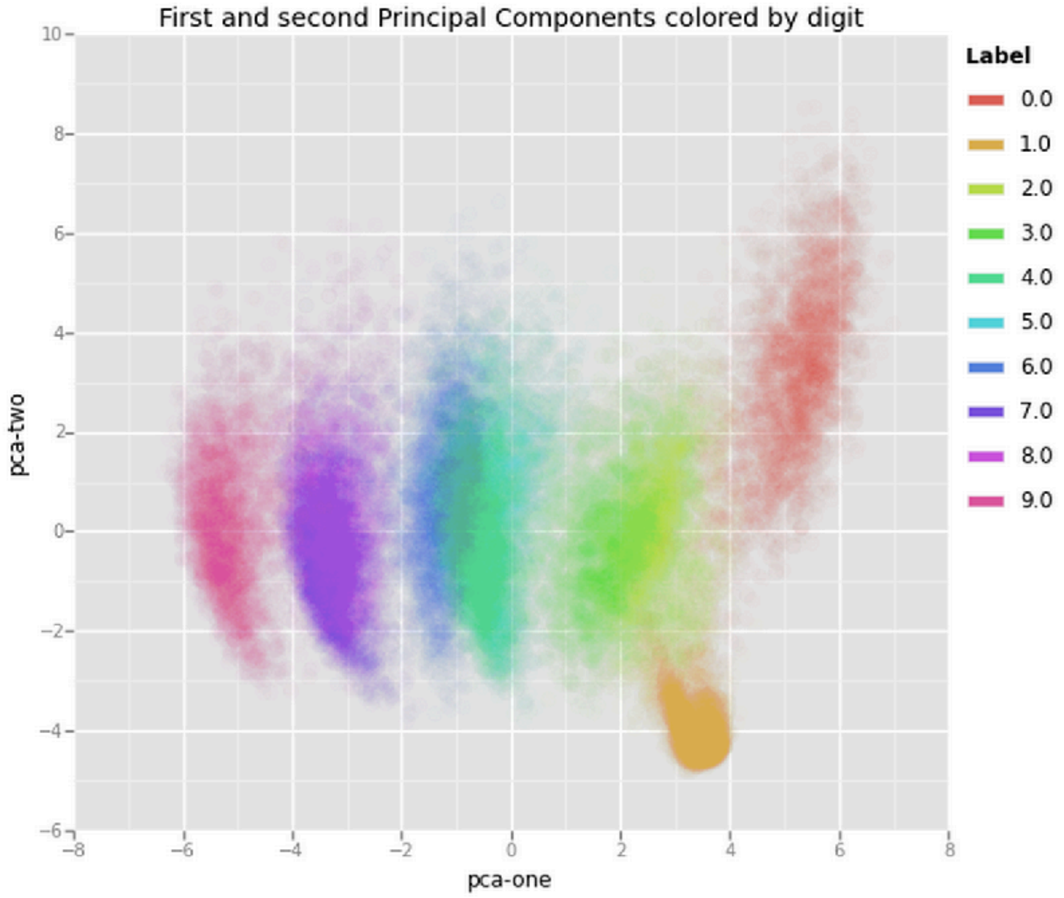
Source: https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/

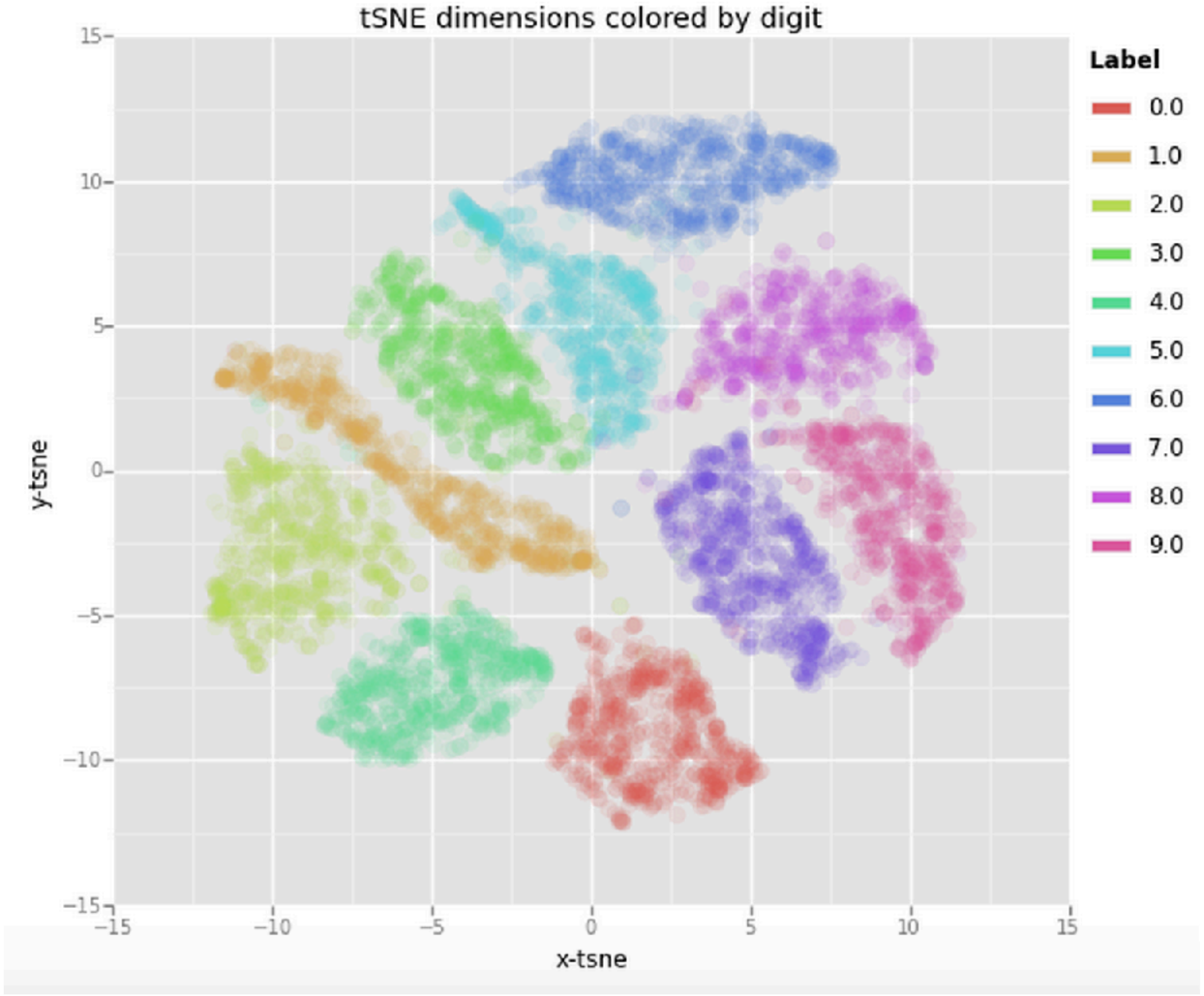
Principal Component Analysis





Dimensionality Reduction: PCA vs t-SNE





Source: https://medium.com/@luckylwk/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b

http://colah.github.io/posts/2014-10-Visualizing-MNIST/

{LAB} time

Please Complete “K-Means Worksheet”