Unsupervised learning

R workshops, 2021
Baruch college

Víctor Peña, January 2021

Unsupervised learning

- **Descriptive analysis** of the dataset (without making predictions)
- Most often, unsupervised learning algorithms are either
  - **Dimensionality reduction algorithms:** Reduce the dimensionality of a large dataset to something that we can visualize, trying to lose as little information as possible [The goal is usually to find the "best" 2- or 3-dimensional representation of a dataset with many variables.]
    - Examples: principal component analysis, multidimensional scaling, ...
  - **Clustering algorithms:** Creating groups of interesting observations or variables
    - Examples: k-means clustering, hierarchical clustering, ...

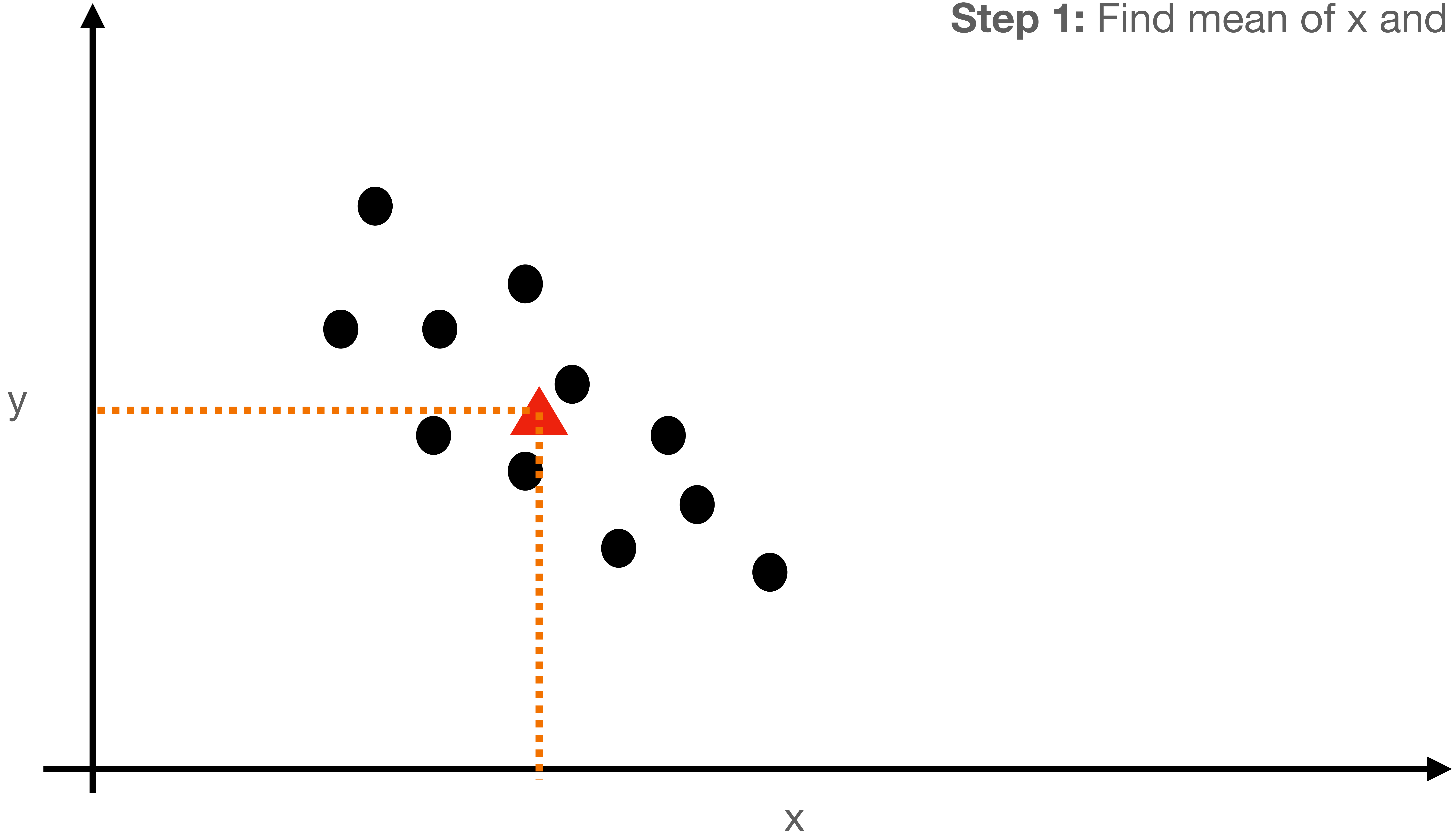
Principal component analysis

Principal Component Analysis (PCA)

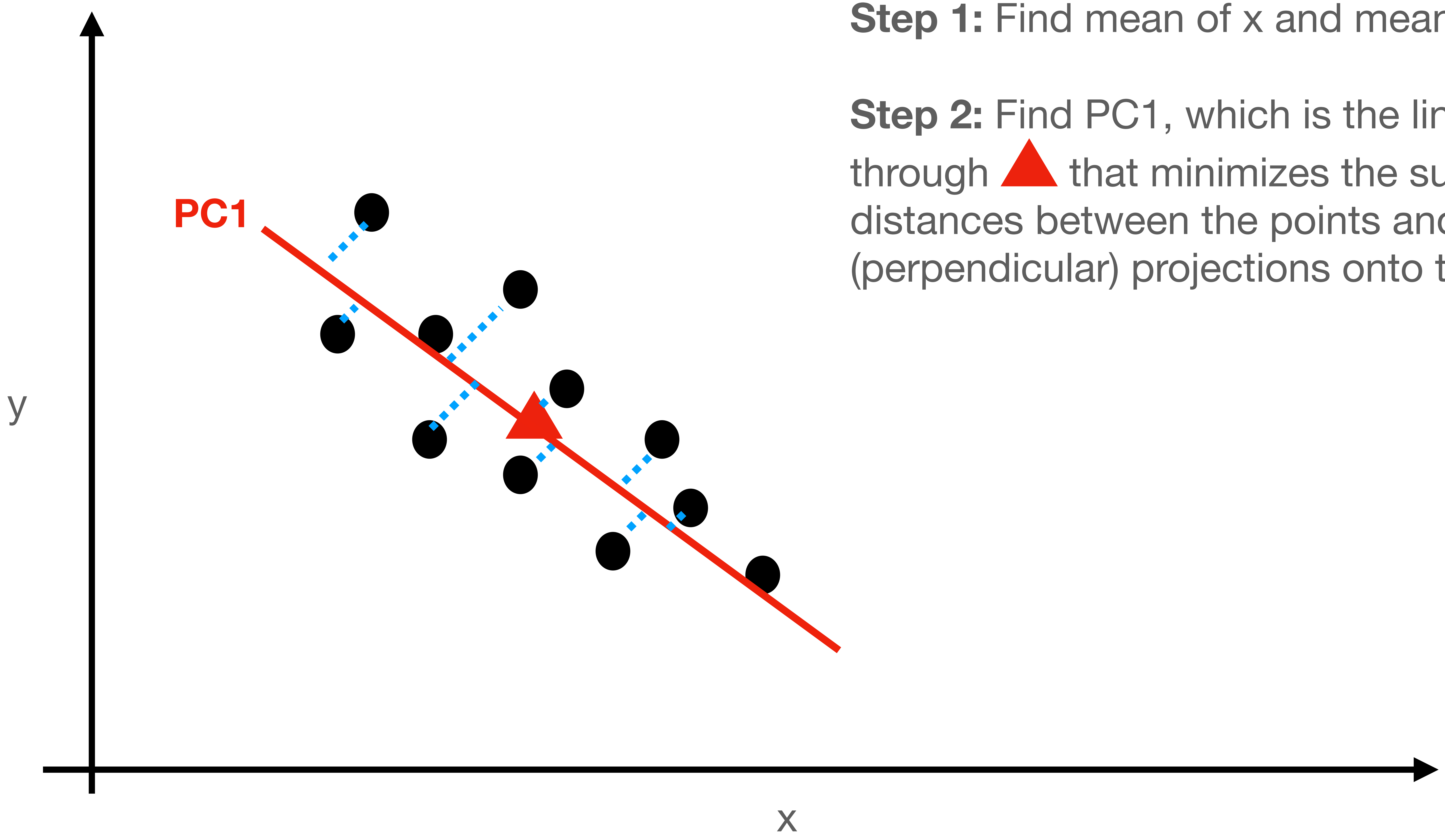
- **Goal:** Find lower-dimensional summaries of higher-dimensional datasets
- **Why?**
  - We might be interested in studying which variables and rows of the dataset "go together" and which ones do not
  - Why not plot all possible combinations of 2 variables?
    1. Tedious: If we have 20 variables, we'd have to look at 190 plots
    2. Misses higher-order dependence: Pairwise plots can capture pairwise dependence, but more intricate patterns might exist

PCA in 2D

Step 1: Find mean of x and mean of y (▲)



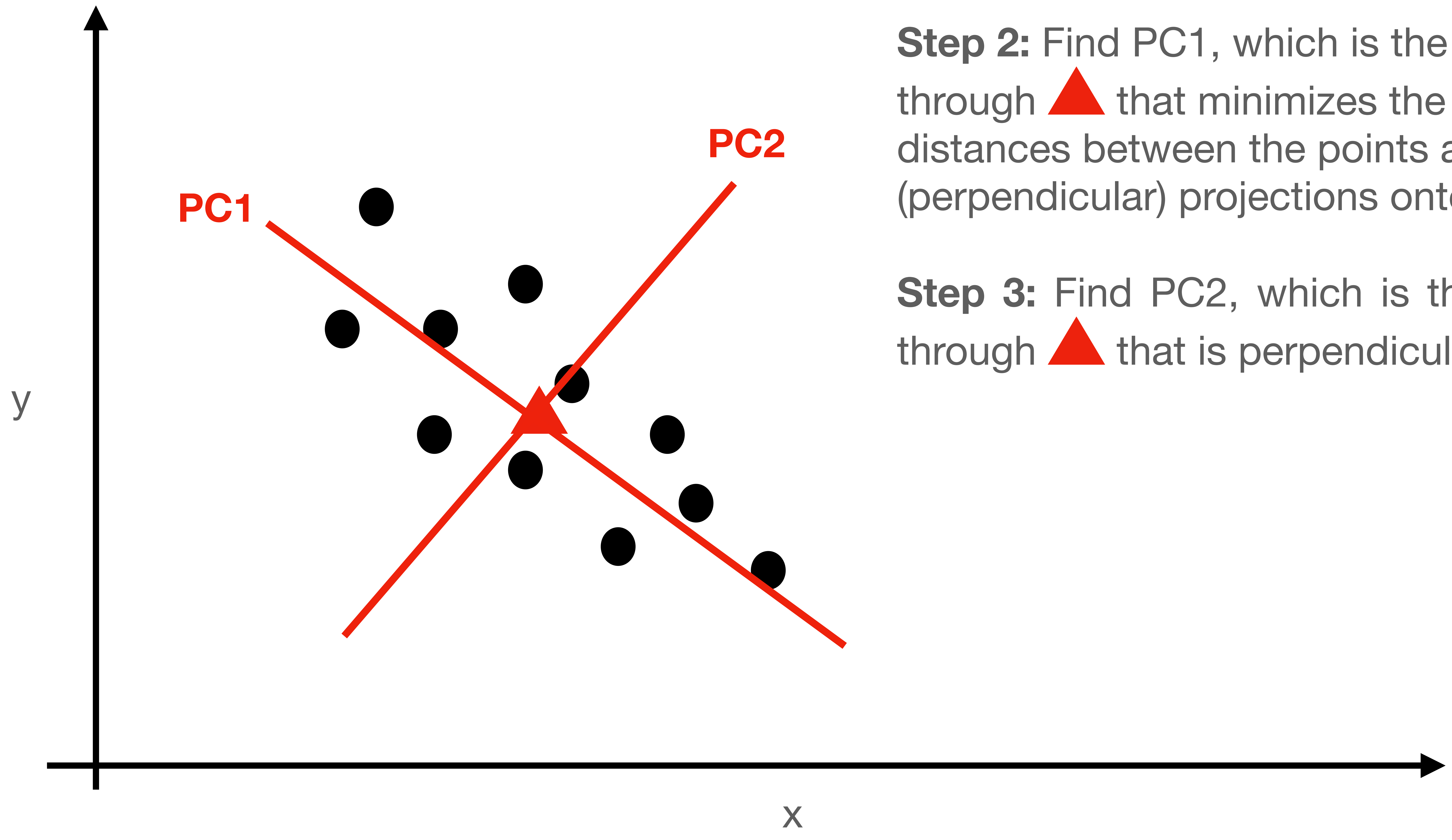
PCA in 2D



Step 1: Find mean of x and mean of y (▲)

Step 2: Find PC1, which is the line that goes through ▲ that minimizes the sum of squared distances between the points and their (perpendicular) projections onto the line

PCA in 2D



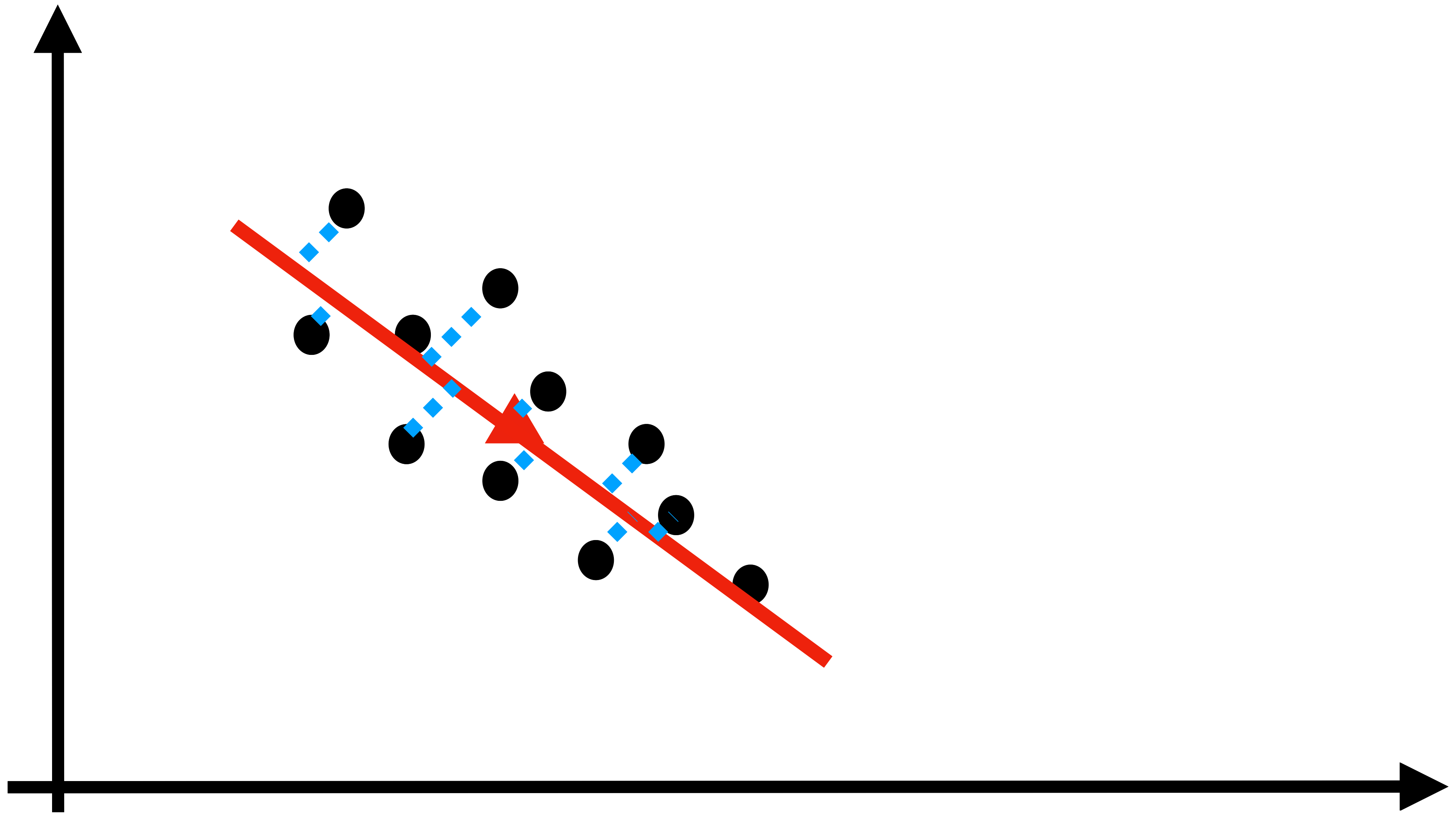
Step 1: Find mean of x and mean of y ( )

Step 2: Find PC1, which is the line that goes through that minimizes the sum of squared distances between the points and their (perpendicular) projections onto the line

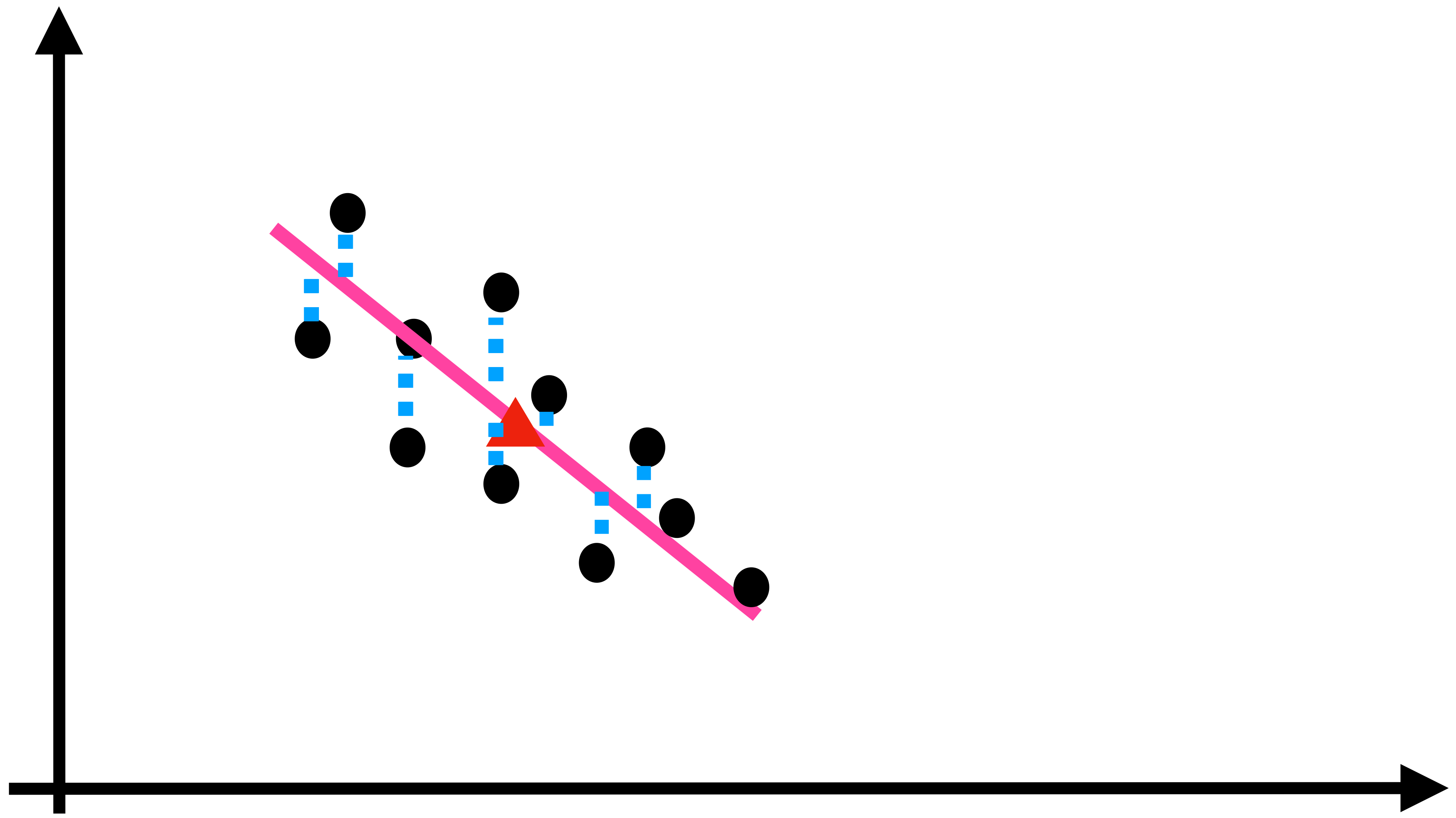
Step 3: Find PC2, which is the line that goes through that is perpendicular to PC1

PC1 vs ordinary least squares (OLS)

PC1 minimizes perpendicular distances



OLS minimizes perpendicular distances



Both lines go through ▲

PCA in higher dimensions

- Sort of the same thing
- For example, in 3D, with variables X1, X2, X3:
  1. Find means of X1, X2, X3, call that 3D point M
  2. Find PC1, which is the line that goes through M that minimizes sum of (point - perp. projection)^2
  3. Find PC2, the line that goes through M and is perpendicular to PC1 that minimizes sum of (point - perp. projection)^2
  4. Find PC3, the line that goes through M and is perpendicular to both PC1 and PC2

Why is this useful at all?

- Suppose we have **many** variables (think 1000 or more)
- We can project our points onto PC1 and PC2 and get a 2-dimensional summary of the dataset
- Points that are close in our 2D plot will tend to be similar in the original 1000-dimensional space... ***if PCA works well***
- How to quantify how well PCA works?
  - There are many different approaches to doing this, as you may imagine!
    - One popular option: quantifying what % of the total variability in the data is captured by the PCs [Same idea as R^2 in regression, but with the PCs.]

Some odds and ends

- **Can we interpret the PCs?**
  - Yes. They're combinations of the original variables. Going back to our example with variables X and Y, we can see that, in some sense, PC1 can be written as a*X + b*Y, where a and b are numbers we can get from R
- **What if I have categorical variables or, in my application, measuring distances using the usual notion of distance doesn't make sense?**
  - Good question. There are different options here. A popular approach is multi-dimensional scaling (MDS), which is doing PCA using distances that differ from our usual notion of distance

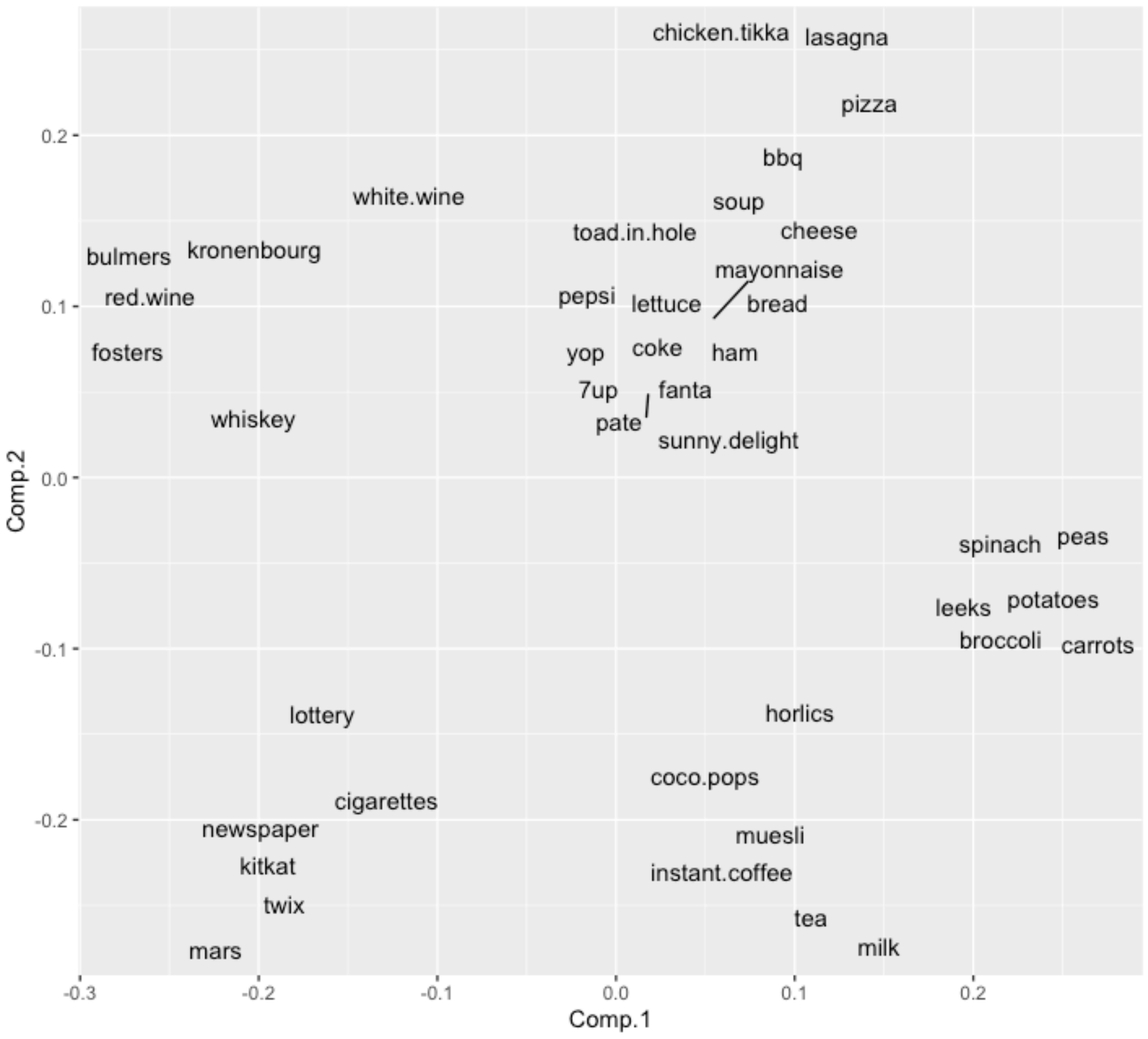
Time to practice

- We'll play around with some basket analysis data

Clustering algorithms

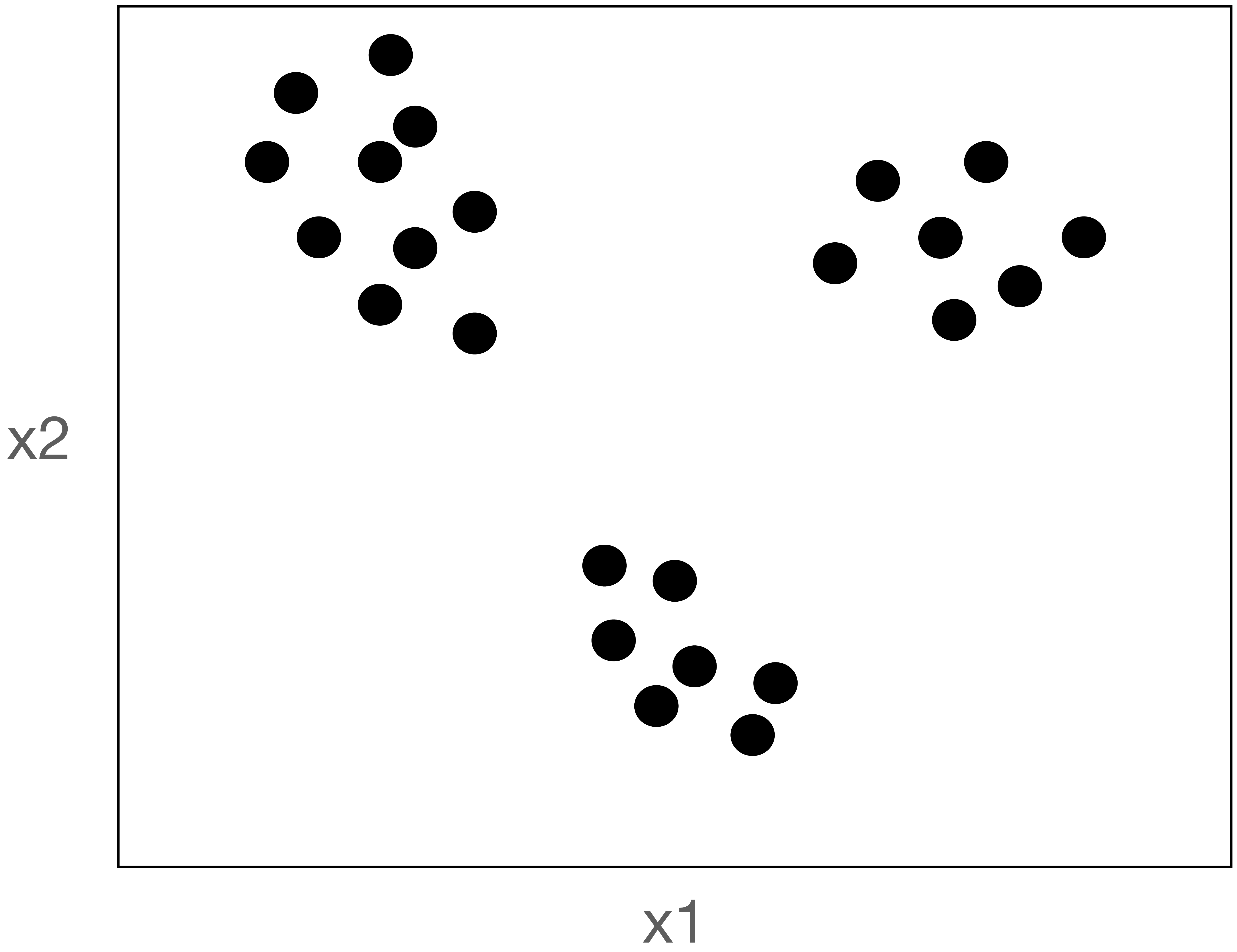
What is clustering?

- **Goal:** Identifying groups of observations or variables that seem to "go together" **automatically**
- In our basket analysis example, we might want to identify groups of items that tend to be bought together
- Today, we'll cover
  - K-means
  - Hierarchical clustering
  - Can find more in Hands on Machine Learning in R, by Boehmke



k-means

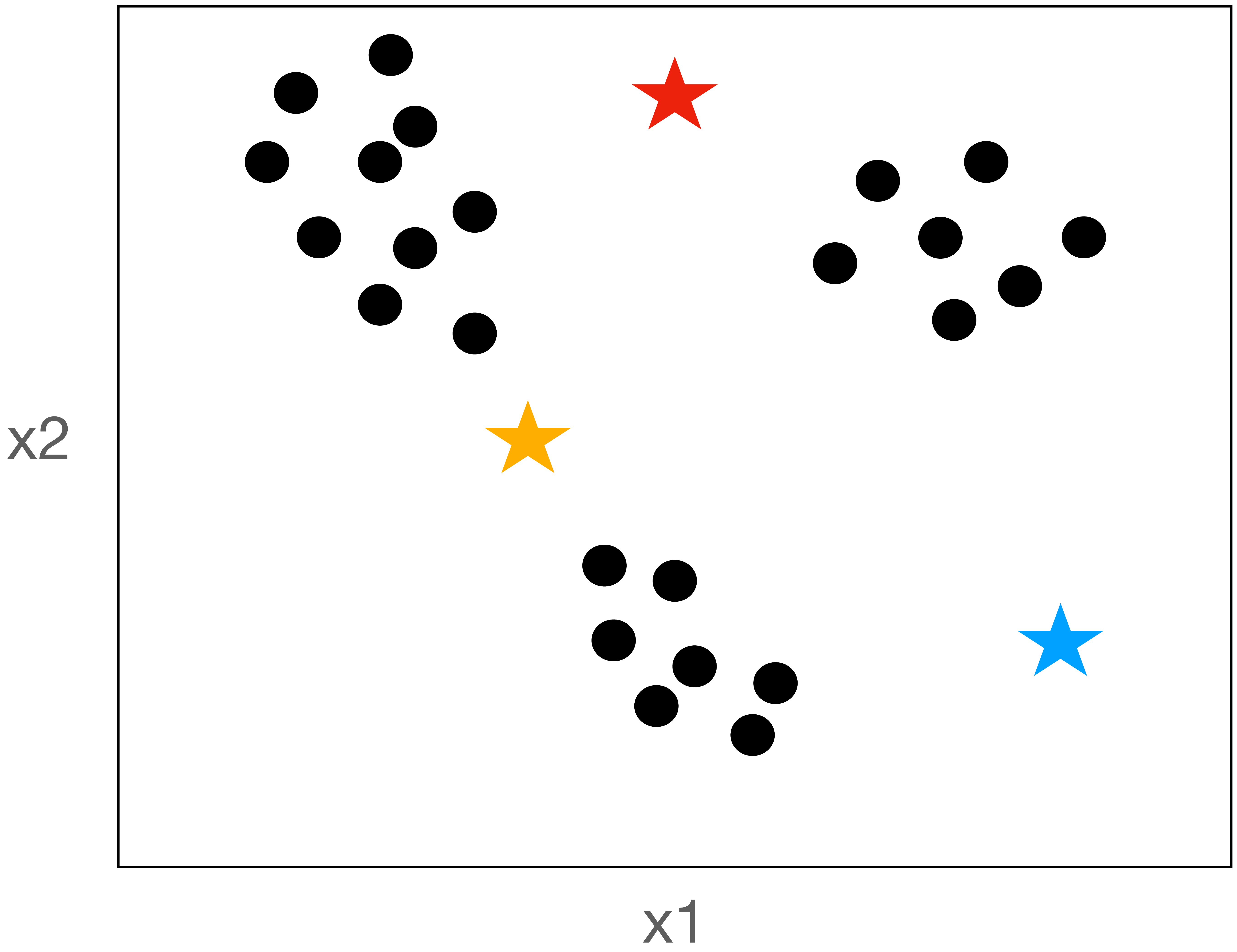
- **Step 1:** Select k, the number of clusters we want. Here, k = 3 makes sense.



k-means

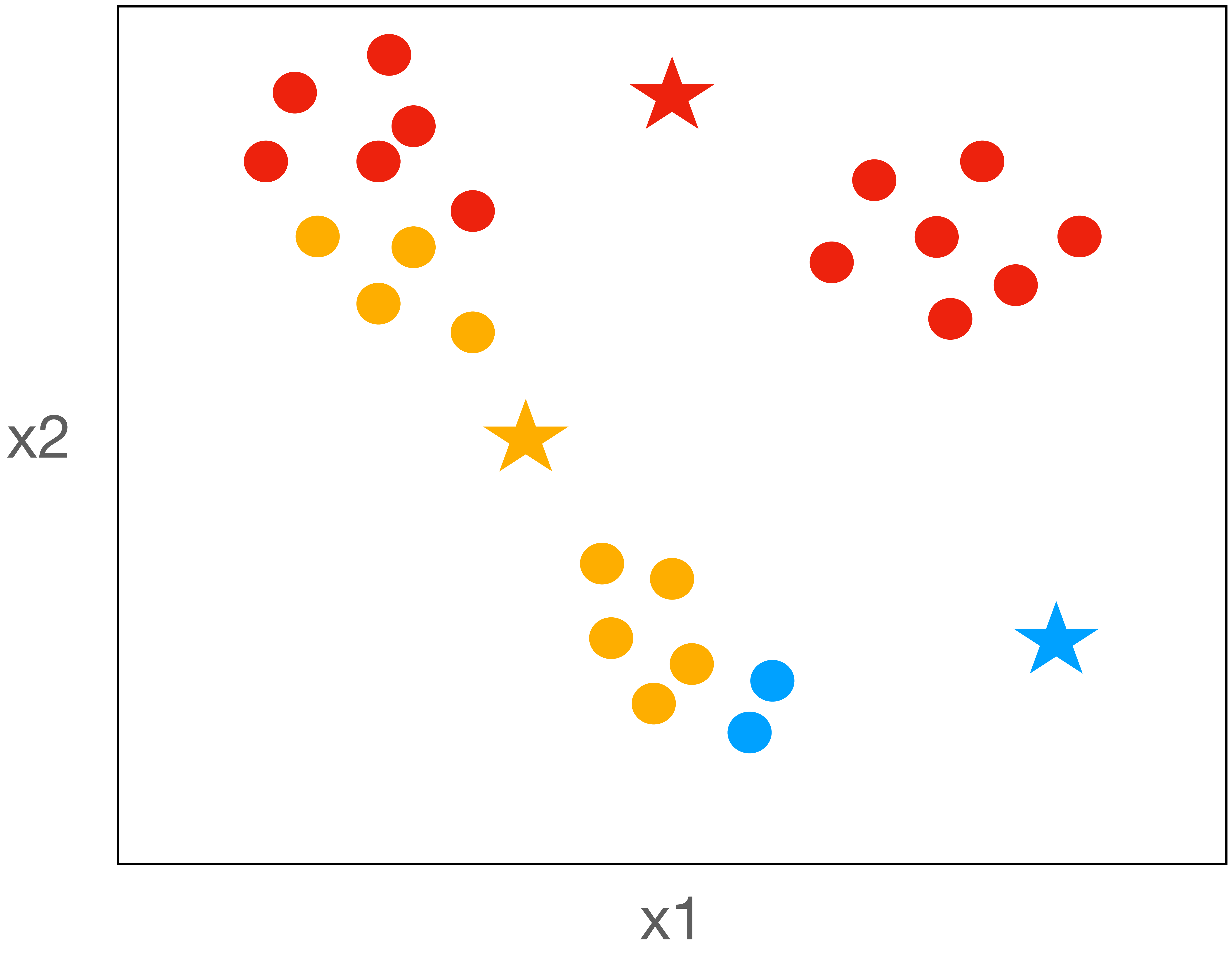
- **Step 1:** Select k, the number clusters we want. Here, k = 3 makes sense.

- **Step 2:** Assign k = 3 cluster centroids at random.



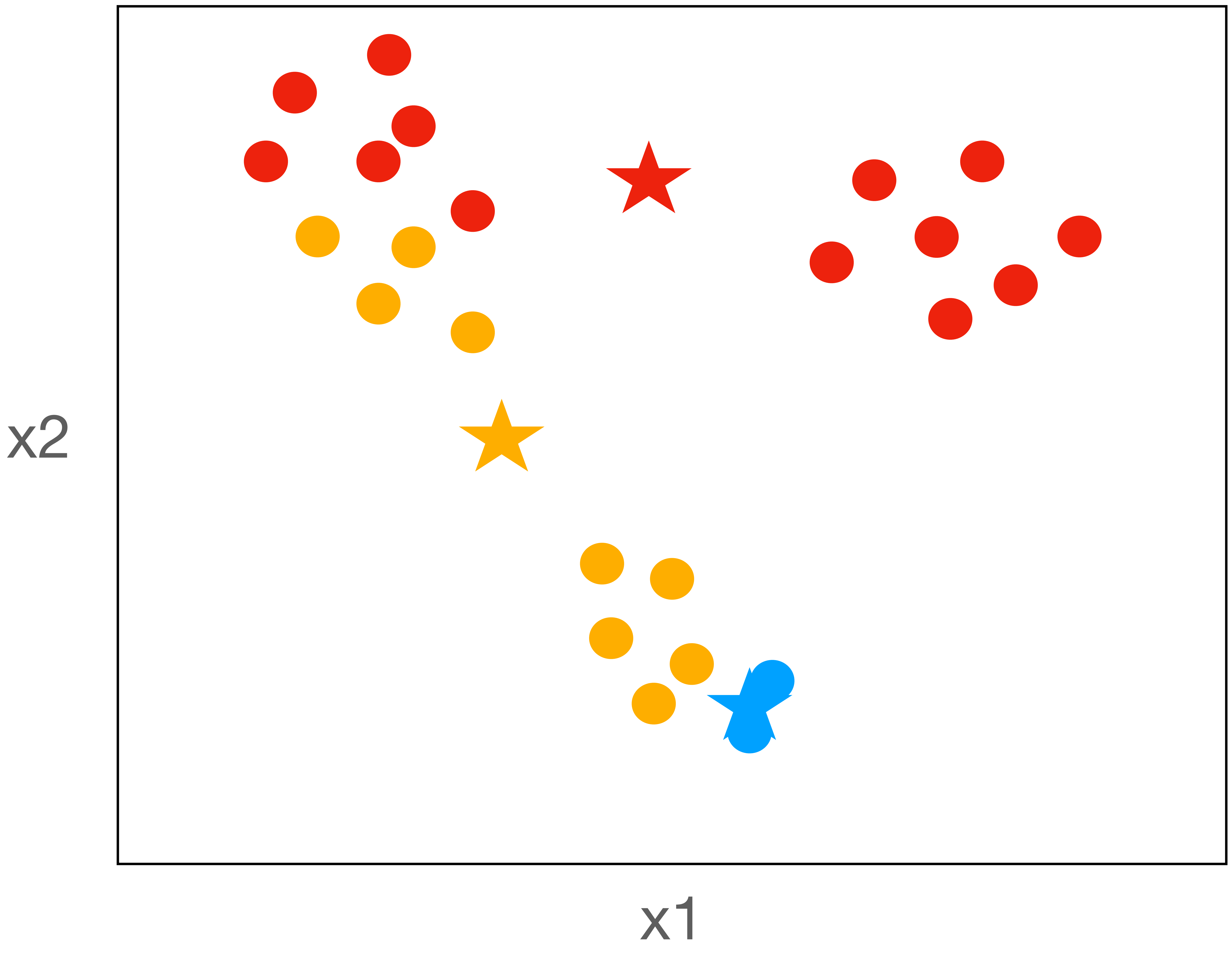
k-means

- **Step 1:** Select k, the number clusters we want. Here, k = 3 makes sense.
- **Step 2:** Place k = 3 cluster centroids at random.
- **Step 3:** Assign observations to clusters by looking at which centroid is closest



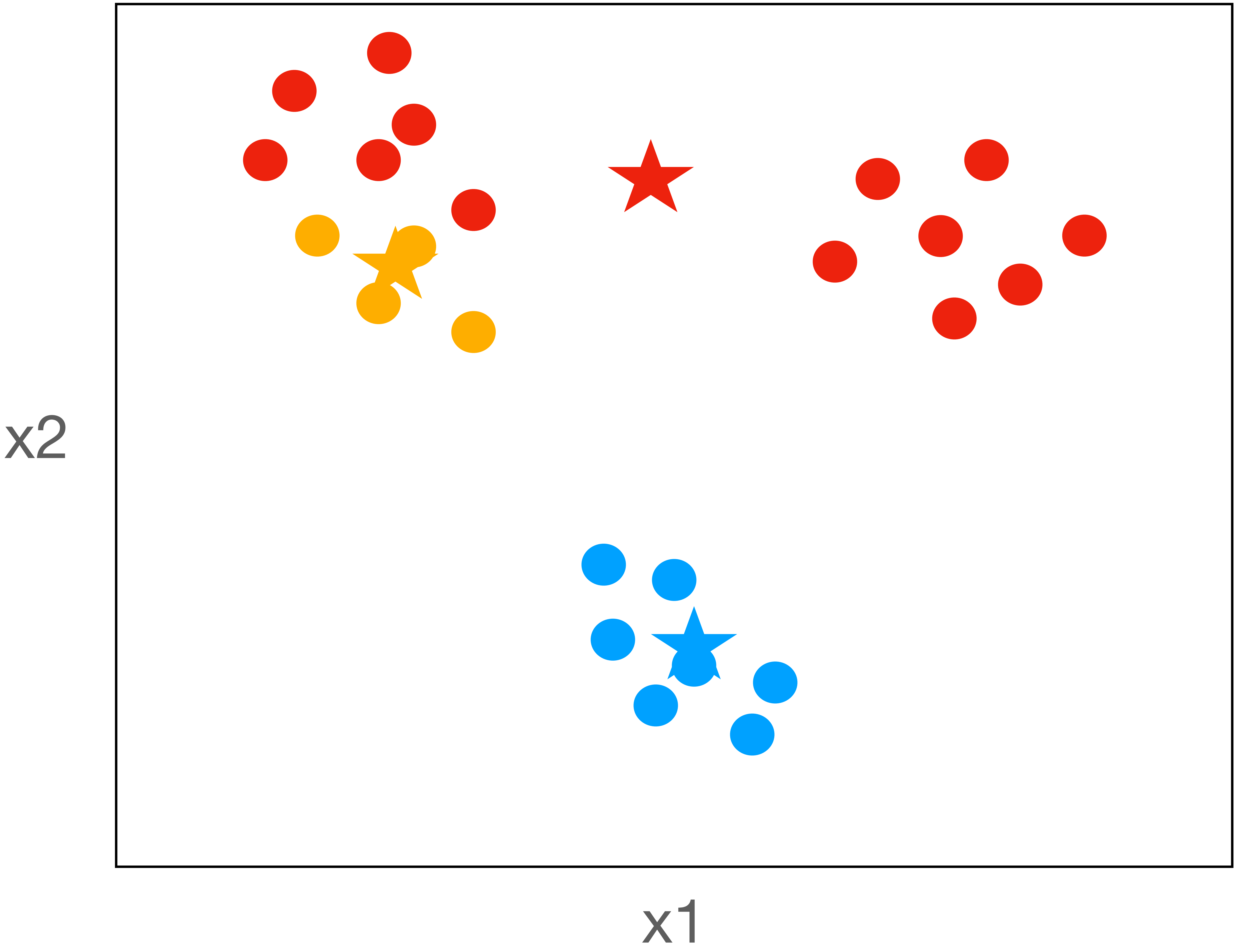
k-means

- **Step 1:** Select k, the number clusters we want. Here, k = 3 makes sense.
- **Step 2:** Place k = 3 cluster centroids at random.
- **Step 3:** Assign observations to clusters given by closest centroid
- **Step 4:** Find new centroids as the average of the points assigned to clusters



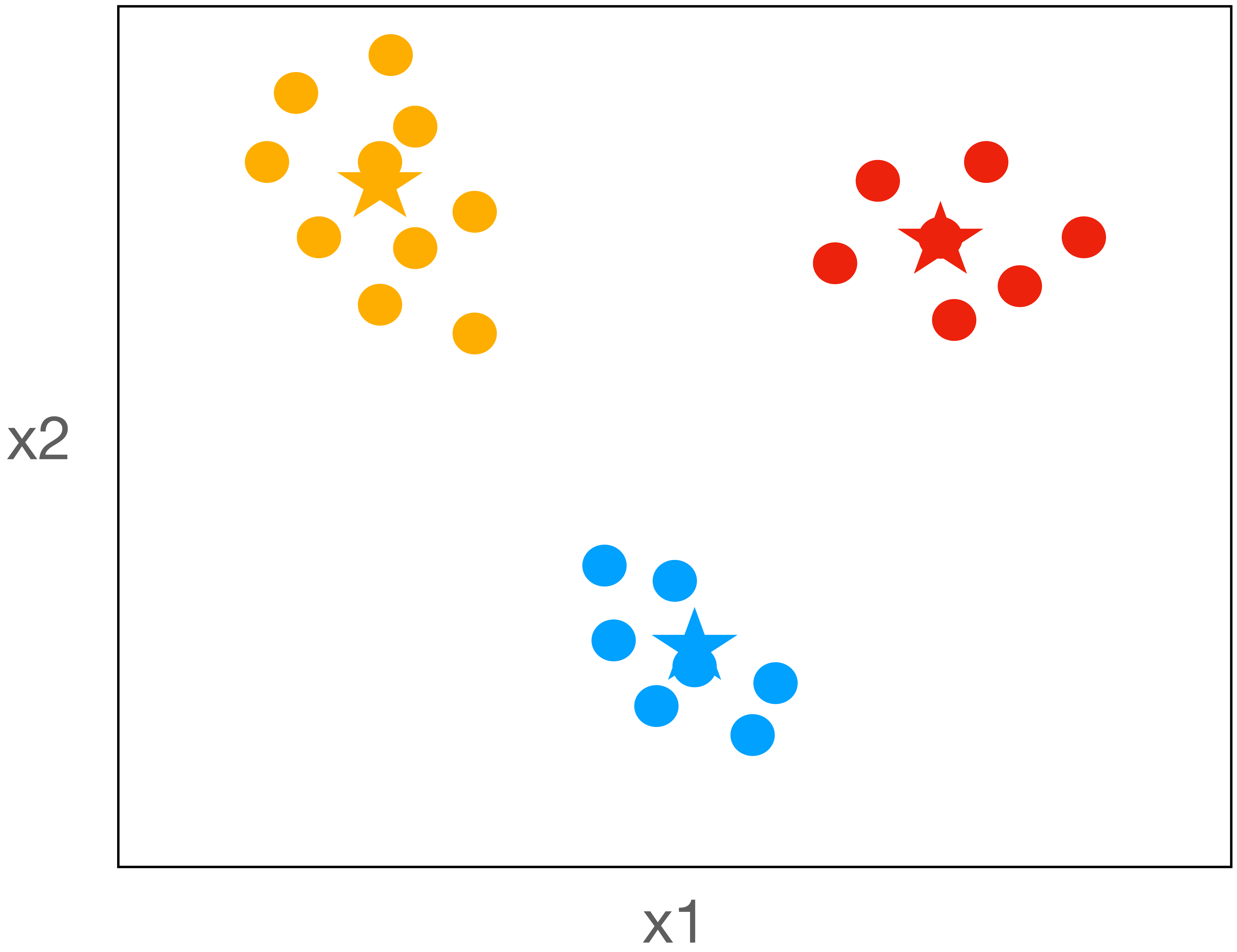
k-means

- **Step 1:** Select k, the number clusters we want. Here, k = 3 makes sense.
- **Step 2:** Place k = 3 cluster centroids at random.
- **Step 3:** Assign observations to clusters given by closest centroid
- **Step 4:** Find new centroids by averaging data assigned to clusters
- **Step 5:** Repeat Step 3 and Step 4 until cluster assignment doesn't change



k-means

- **Step 1:** Select k, the number clusters we want. Here, k = 3 makes sense.
- **Step 2:** Place k = 3 cluster centroids at random.
- **Step 3:** Assign observations to clusters given by closest centroid
- **Step 4:** Find new centroids by averaging data assigned to clusters
- **Step 5:** Repeat Step 3 and Step 4 until cluster assignment doesn't change



Odds and ends

- The *output of the algorithm can depend on the initial location of the centroids*. To mitigate this issue, you can run the algorithm a few times and average the runs, somehow.

- How many clusters should we pick? In our example, it was clear that k = 3 made most sense. Sometimes, it isn't clear how many clusters we want in advance.

    - We can plot the within-cluster variability for different values of k and pick a value of k after which "adding more clusters doesn't seem to help much"

Hierarchical clustering

- **Example:** Want to cluster variables x1, x2, x3, x4
- **Step 0:** Define a notion of distance between variables that makes sense in context
- **Step 1:** Compute sum of square of differences between pairs of variables (SS)
    - SS(x1, x2) = [(1-1)^2+(1-0)^2+(2-1)^2+(0-2)^2]
    - SS(x1, x3) = [(1-1)^2+(1-0)^2+(2-1)^2+(0-1)^2]
    - SS(x1, x4)
    - ...
- Combine two variables with smallest SS into a cluster
- Other notions of distance are possible (e.g. take absolute values instead)

x1	x2	x3	x4
1	1	1	1
1	0	0	0
2	1	1	1
0	2	1	0

Hierarchical clustering

- **Example:** Want to cluster variables x1, x2, x3, x4
- **Step 1:** Compute sum of square of differences between pairs of variables (SS)
    - SS(x1, x2) = [(1-1)^2+(1-0)^2+(2-1)^2+(0-2)^2]
    - SS(x1, x3) = [(1-1)^2+(1-2)^2+(2-1)^2+(0-1)^2]
    - SS(x1, x4)
    - ...
- Combine two variables with smallest SS into a cluster
- In this case, it would be x3 and x4

x1	x2	x3	x4
1	1	1	1
1	0	0	0
2	1	1	1
0	2	1	0

Hierarchical clustering

- **Step 2:** Do the same thing, treating the cluster (x3, x4) as a "variable"...
- How do we compute the distance between a variable and a cluster (or, in general, distance between clusters?)

x1	x2	x3	x4
1	1	1	1
1	0	0	0
2	1	1	1
0	2	1	0

Hierarchical clustering

- Computing distances between clusters:
  - **Complete linkage:** distance between variables in clusters that are farthest away
    - define the distance as the worst possible distance for observations within the clusters
  - **Single linkage:** distance between variables in clusters that are closest
  - **Centroid method:** distance between the centroids (means) of the clusters
  - ...

x1	x2	x3	x4
1	1	1	1
1	0	0	0
2	1	1	1
0	2	1	0

Hierarchical clustering

- **Step 3:** Keep repeating the process until all variables are put together in a single cluster

x1	x2	x3	x4
1	1	1	1
1	0	0	0
2	1	1	1
0	2	1	0

How to choose the number of clusters?

- Can use the same strategy we used for k-means
- Track within-cluster variability for different number of clusters
- Pick value after which creating a new cluster doesn't seem to help much
- There are many other methods you can use to select the number of clusters. For examples, you can take a look at Chapter 21 of Hands-on Machine learning with R, by Boehmke

Odds and ends

- Here, I explained what is known as "agglomerative" clustering
- Divisive clustering starts with all variables clustered together, and at each step we split up into clusters, aiming at "maximizing distance"

References

- Hands-on Machine Learning with R, by Bradley Boehmke
- StatQuest, by Josh Starmer