K-means clustering

- In K-means clustering, have to specify K, the number of clusters we want.
- The aim of K-means is then to choose the K clusters so that the total *within-cluster variation* is *minimised*.
  - a simple objective to state, but rather difficult to obtain precisely – there are almost K^n ways to cluster the observations!

- Let $C_1, \ldots, C_K$ denote the K disjoint (non-overlapping) clusters, i.e. sets containing the indices of the observations in each cluster.

$$\min_{C_1,\ldots,CK} \left\{ \sum_{k=1}^{K} W(CK) \right\},$$

where $W(CK)$ is a measure of within-cluster variation

- The idea behind is a good clustering is one for which the *within-cluster variation* is as *small* as possible
- See also ISLR 10.3.1

Within-cluster variation

- By far the most common measure of within-cluster variation is based on the **squared Euclidean distance**:

$$W(C_k) = \frac{1}{|C_k|}\sum_{i,i' \in C_k}(x_i - x_{i'})^2 = \frac{1}{|C_k|}\sum_{i,i' \in C_k}\sum_{j=1}^{p}(x_{ij} - x_{i'j})^2,$$

where x_{ij} is the j*th* component of x_i, i.e. the value of feature j for observation i, and $|C_k|$ is the number of observations in cluster k

- Sum of all of the pairwise squared Euclidean distances between the observations in the *kth* cluster, divided by the total number of observations in the *kth* cluster
- The squared Euclidean distance is a measure of dissimilarity between pairs of observations

$$W(C_k) = \frac{1}{|C_k|}\sum_{i,i' \in C_k}(x_i - x_{i'})^2 = 2\sum_{i \in C_k}(x_i - \mathcal{U}_k)^2$$

where $\mathcal{U}_k = \frac{1}{|C_k|}\sum_{i \in C_k} x_i$ is the **centroid** of cluster k.

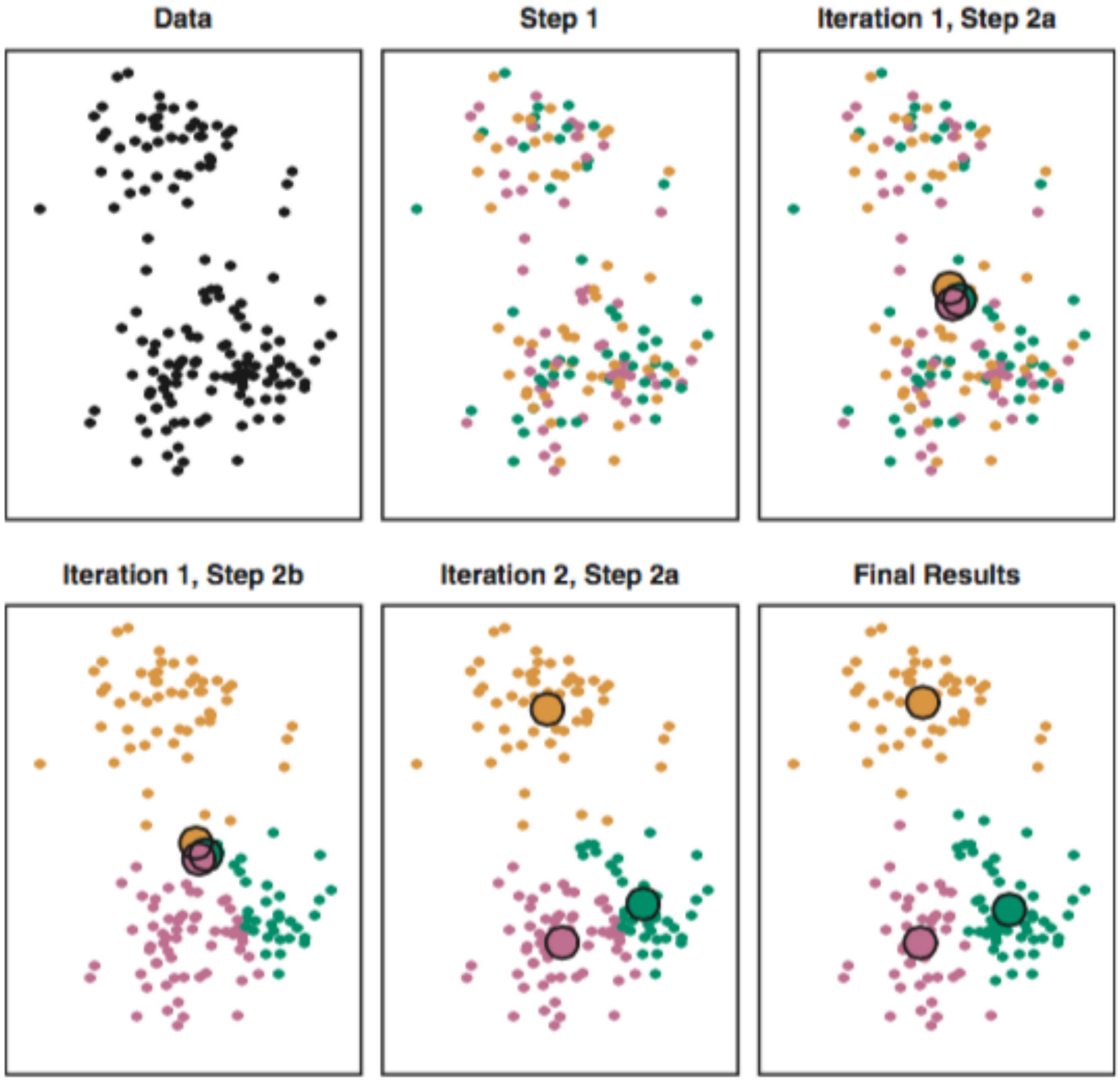
K-means algorithm

Main steps of K-means:

- Initialise $C_1, \ldots, C_K$ by randomly assigning each observation a number from 1 to K
- Repeat until the the cluster assignments don't change:
  - (a) Compute the *centroid* for each cluster
  - (b) Assign each observation to the cluster whose *centroid* is *closest* in Euclidean distance
- Algorithm 10.1 of ISLR

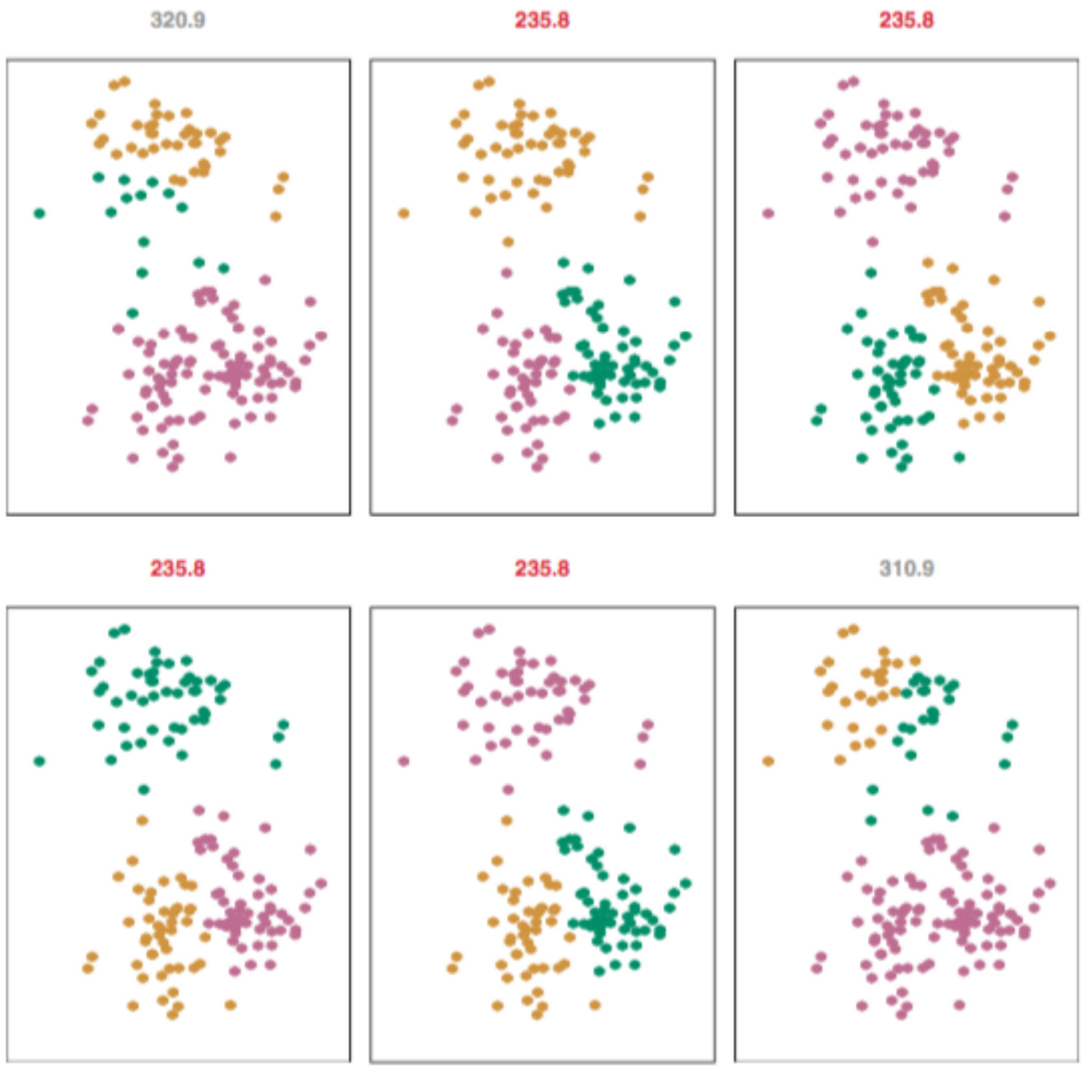
- The algorithm finds a *local minimum* of the objective function $\sum_{k=1}^{K} W(C_k)$.

K-means algorithm



ISLR Figure 10.6: K-means algorithm in operation

K-means algorithm



ISLR Figure 10.7: Different starting points can lead to different local minima

Iris example

- The Iris dataset is a famous dataset. The data is labelled so it's usually used as a common example for classification

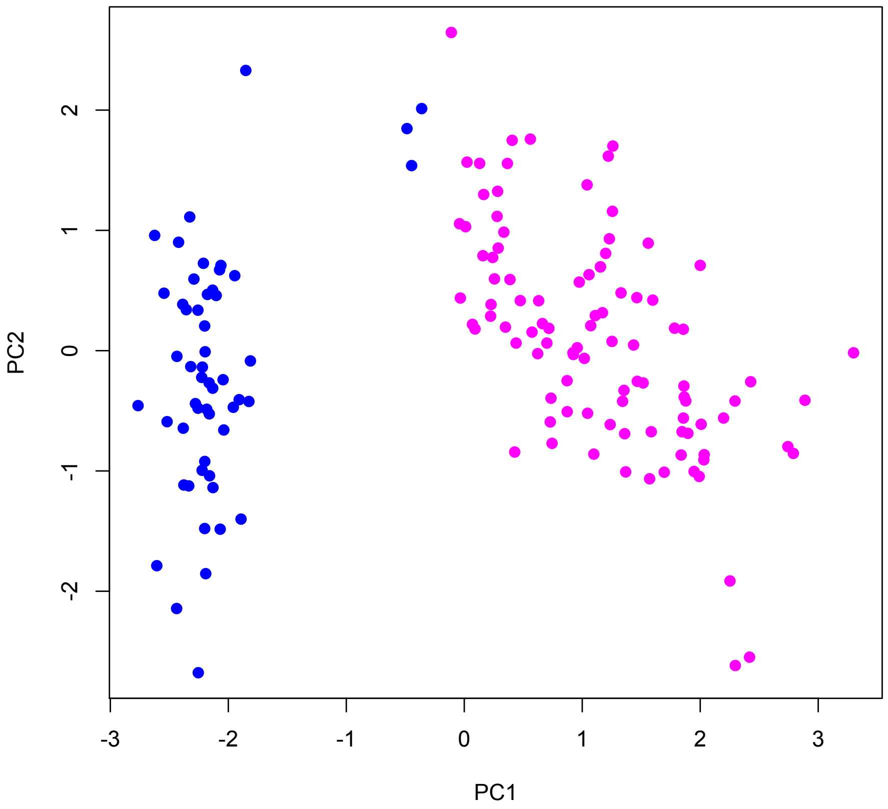
```
> data(iris)
> summary(iris)

  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

- Note that $n = 150$ and $p = 4$
- We'll see how clustering performs on this dataset

Iris example

- The function kmeans performs K-means clustering in R. First, let's ask for 2 clusters:

```
> km = kmeans(iris[,1:4],2)
```



$K = 2$ clustering shown for the first 2 principal components

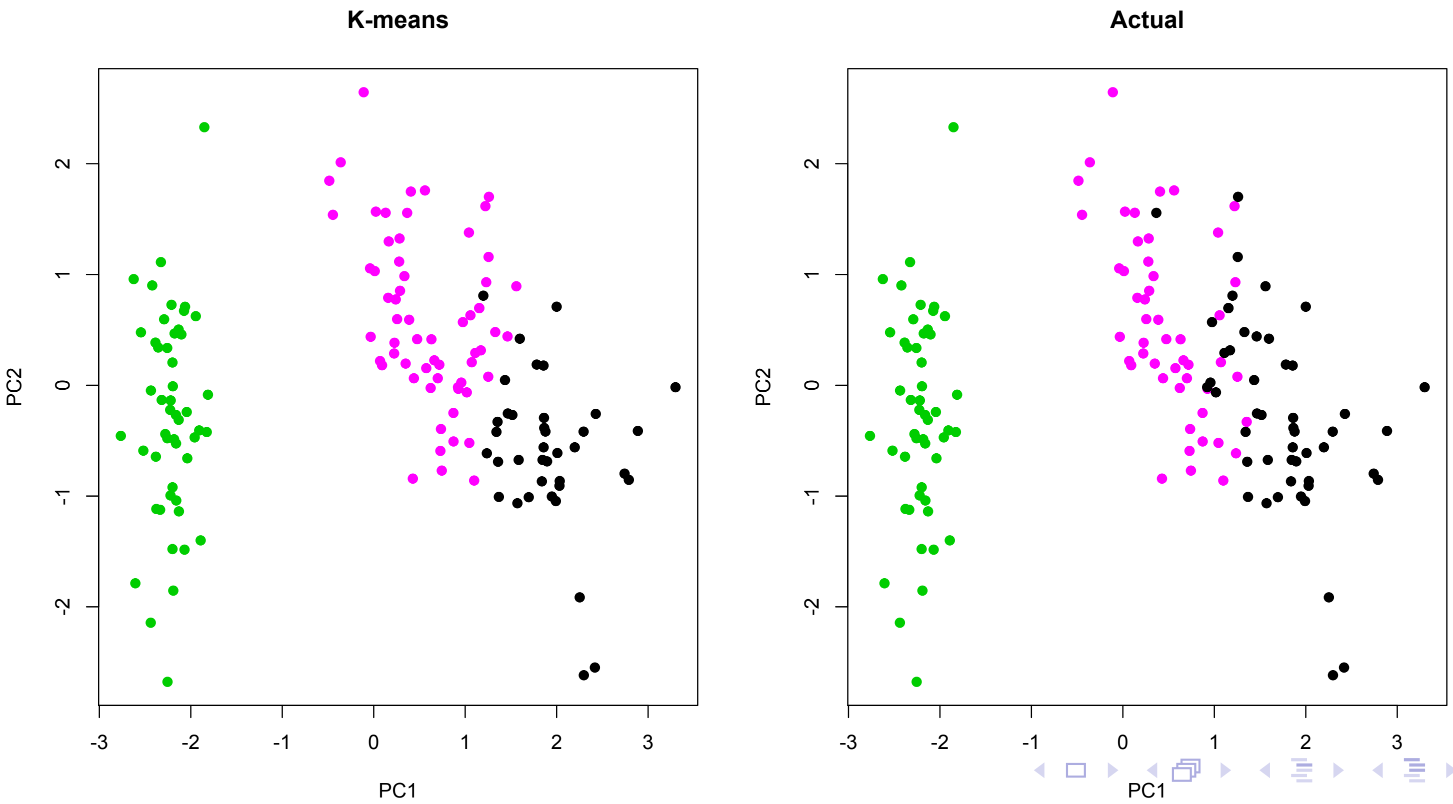
Iris example

- Now try $K = 3$, since we know there are 3 classes! We'll also ask for the best clustering from 50 random initialisations

```
> km = kmeans(iris[,1:4],3,nstart=50)
> km

K-means clustering with 3 clusters of sizes 50, 62, 38

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.006000    3.428000     1.462000    0.246000
2     5.901613    2.748387     4.393548    1.433871
3     6.850000    3.073684     5.742105    2.071053

Clustering vector:
  [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
 [41] 1 1 1 1 1 1 1 1 1 1 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2
 [81] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3 3 3 2 2 3 3 3 3 2
[121] 3 2 3 2 3 3 2 2 3 3 3 3 3 2 3 3 3 3 2 3 3 3 2 3 3 3 2 3 3 2

Within cluster sum of squares by cluster:
[1] 15.15100 39.82097 23.87947
 (between_SS / total_SS =  88.4 %)
```

Iris example

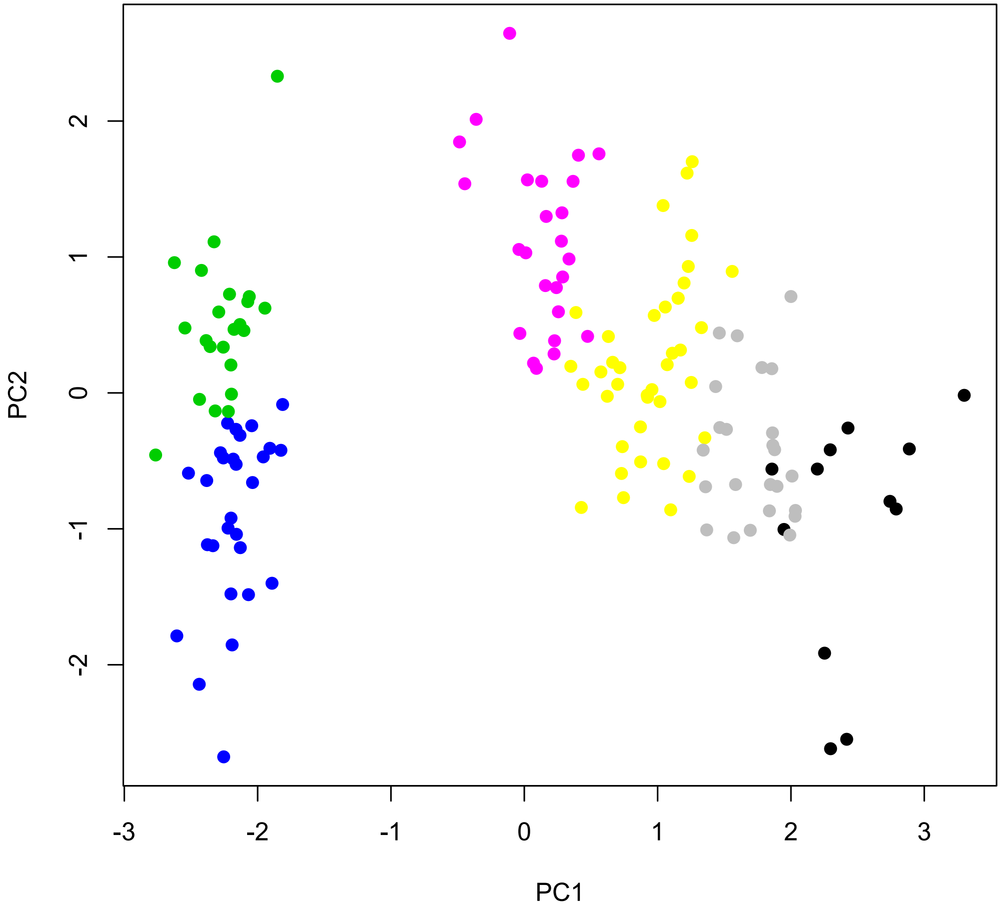
```
> table(km$cluster,iris[,5])

    setosa versicolor virginica
  1     50          0         0
  2      0         48        14
  3      0          2        36
```



Iris example

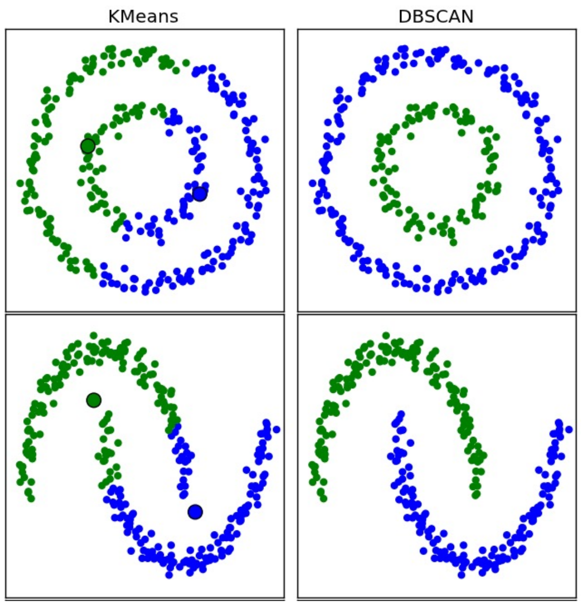
- Of course, if we ask for more clusters, K-means will find them:

```
> km = kmeans(iris[,1:4],6,nstart=50)
```



Comments on K-means

- Have to predefine K: no guidance on how to choose K
- K-means is based on *spherical clusters*, which might not always be appropriate.
- Sensitive to initial seeds, local minima



- Sensitive to outliers
- Generalising the distance function is possible, e.g. K-medians clustering defines centroids via *component-wise median* and assignment to a cluster is in terms of the *Manhattan* distance (aka taxicab geometry, l_1-norm)
- Care needs to be taken in *high dimensions; irrelevant* features can conceal information about clusters. Idea of distance also breaks down – curse of dimensionality again.
  - Dimension reduction prior to clustering is a good idea