04 - Clustering

Data Mining

SYS 6018 | Fall 2019

03-density.pdf

Contents

1 Clustering Intro

1.1 Required R Packages

We will be using the R packages of:

- `tidyverse` for data manipulation and visualization
- `mixtools` for mixture modeling
- `mclust` for model-based clustering
- `MASS` for the crabs data

Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.

1.2 Clustering

Cluster analysis divides data into groups (clusters) that are meaningful, useful, or both. [ITDM 7]

- If meaningful, then clusters should capture the natural structure of the underlying data generating process.
  - biological taxonomy
- Alternatively, clustering can be useful for summarizing or reducing the size/complexity of the data.
  - market segmentation for targeted marketing

1.2.1 Clustering: Two Views

1. Find homogeneous subgroups
   - Partition data into groups such that objects in the same group are *similar* to each other, while objects in different groups are *dissimilar*
2. Statistical Clustering
   - Partition data into groups such that objects in the same groups are from the same *distribution*

1.2.2 The Field of clustering

There are probably more approaches/algorithms for clustering than any other area of data mining.

- Different criteria on which to base cluster analysis
  - E.g., Cluster plants on color, size, shape, geography
- Different ways to evaluate a clustering solution
- Different goals of clustering: understanding, data reduction, etc.
- The large variety of data types which can be clustered
  - rectangular, network, time series, functional, etc

We will cover the three most popular methods (in my opinion) which are also the building blocks of the more complex and targeted methods.

1.3 *Leptograpsus variegatus*

The *Leptograpsus variegatus*, or purple rock crab, can take on a blue or orange coloring in some parts of Australia.

Campbell, N.A. and Mahon, R.J. (1974) collected 5 morphological measurements on 200 crabs from the species *Leptograpsus variegatus*. These are recorded in the `crabs` dataset in the `MASS` R package.

```
crabs = MASS::crabs
```

sp	sex	index	FL	RW	CL	CW	BD
B	M	25	15.0	11.9	32.5	37.2	13.6
O	M	25	16.3	11.6	31.6	34.2	14.5
O	F	20	17.1	14.5	33.1	37.2	14.6
B	F	19	12.0	11.1	25.4	29.2	11.0
B	M	17	13.1	10.6	28.2	32.3	11.0
O	M	19	14.7	11.1	29.0	32.1	13.1

- columns
  - `sp` B=blue, O=orange
  - `sex` M=male, F=female
  - `FL`, `RW`, `CL`, `CW`, `BD` morphological measurements
  - `index` not important for us
  - see `?MASS::crabs` for details

We will construct clustering based on the five morphological features and use the color and sex as the group labels (which we will pretend are unknown).

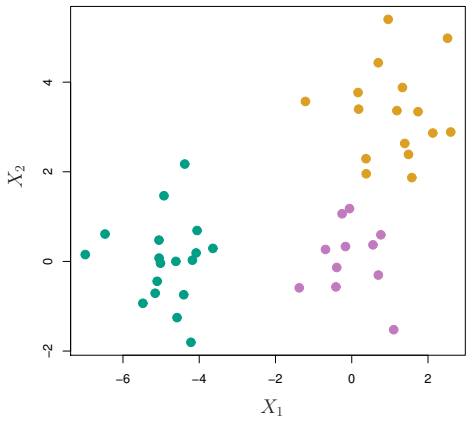
2 Hierarchical Clustering

Hierarchical clustering creates a set of *hierarchical* (or nested) clusters based on a *dissimilarity/distance* metric.

The resulting structure is represented by a dendrogram (*tree-drawing*), which resembles a (usually upside-down) tree.

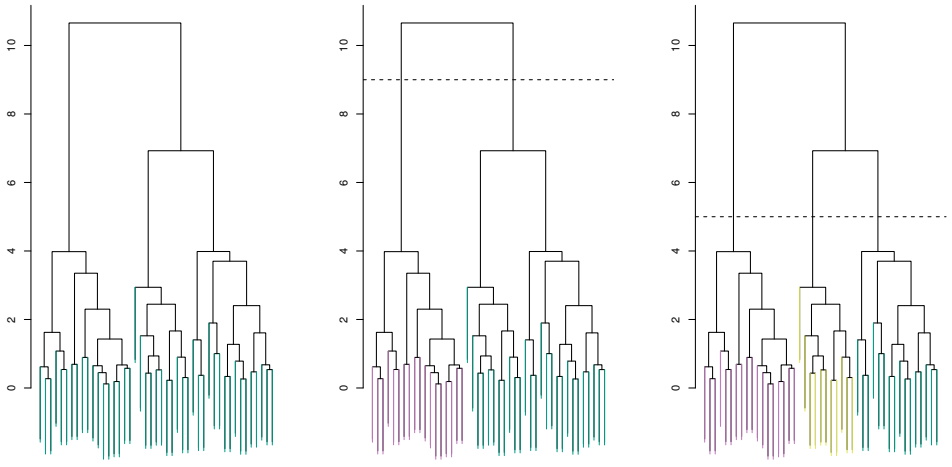
2.1 Dendrogram

Below are 45 observations, in 2D space, from 3 different classes (shown by color).



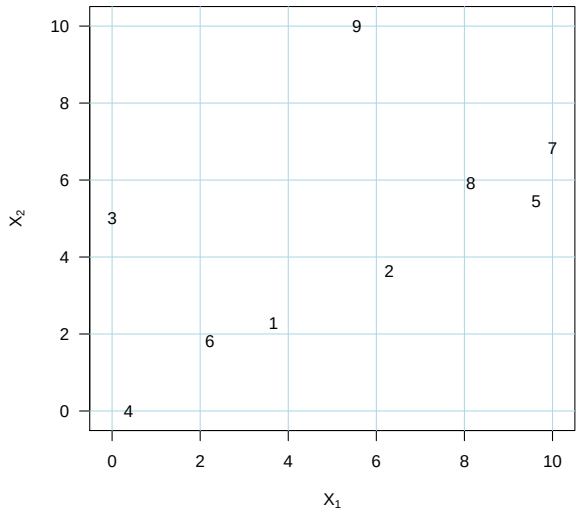
The following dendrogram is constructed from *agglomerative hierarchical clustering*:

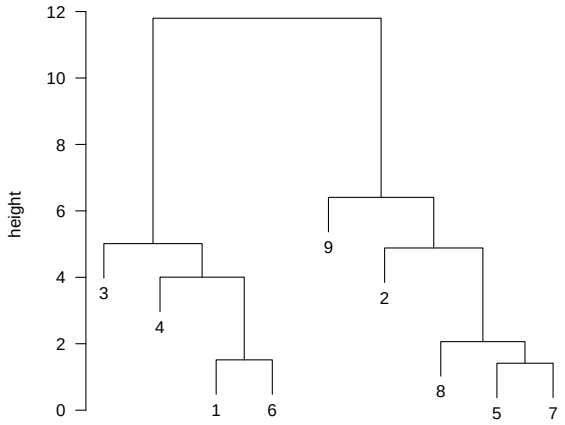
- The *pairwise dissimilarity* is defined as the Euclidean distance between points.
- The *cluster dissimilarity* is defined as the largest dissimilarity between two clusters (complete linkage).



Left: Cut at height of 11 ($K = 1$ cluster); Middle: Cut at height of 9 ($K = 2$ clusters); Right: Cut at height of 5 ($K = 3$ clusters).

2.1.1 Dendrogram Interpretation





Distance Matrix

	1	2	3	4	5	6	7	8
2	2.962							
3	4.570	6.442						
4	4.004	6.953	5.014					
5	6.759	3.797	9.640	10.746				
6	1.514	4.461	3.881	2.595	8.252			
7	7.796	4.883	10.164	11.799	1.413	9.246		
8	5.771	2.932	8.199	9.768	1.550	7.201	2.063	
9	7.955	6.407	7.474	11.264	6.104	8.835	5.466	4.843

2.2 Agglomerative (Greedy) Hierarchical Clustering Algorithm

The basic hierarchical clustering algorithm takes a *greedy*, *sequential* approach to forming the nested groups.

Algorithm: Agglomerative Hierarchical Clustering

Initialize

1. Begin with n observations and treat each observation as a unique cluster. Set the number of clusters $k = n$.
2. Calculate the *dissimilarity* between all $\binom{n}{2}$ clusters.

Iterate: For $k = n - 1, n - 2, \ldots, 2$:

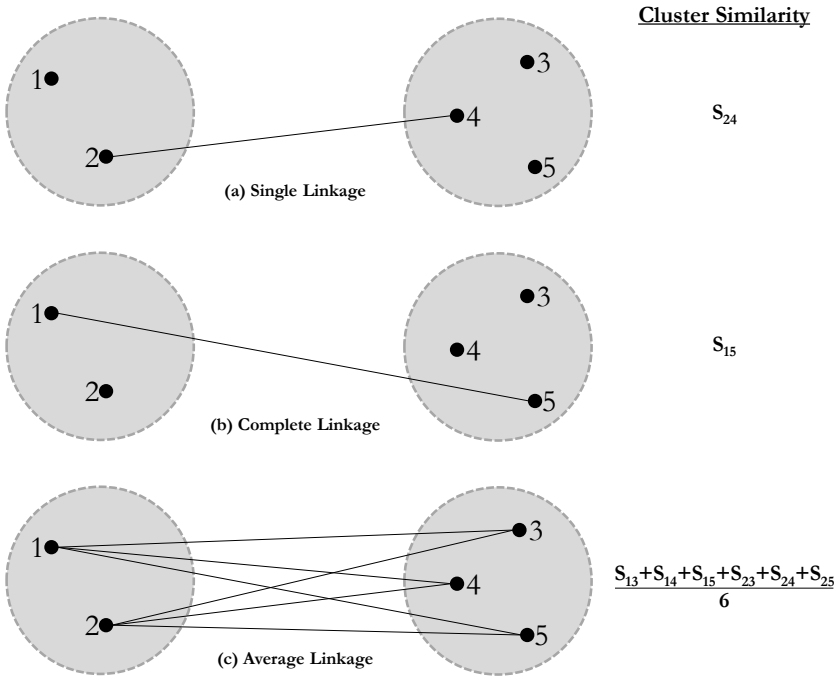
3. Merge the most *similar* (or least *dissimilar*) clusters.
4. Update the pairwise dissimilarity between the new cluster and all existing clusters.
5. Set $k = k - 1$

2.2.1 Cluster Dissimilarity

There are a few common approaches to calculate the dissimilarity between *clusters* (or sets).

The first three are based on the pairwise dissimilarity/similarity between observations.

- **Single Linkage/Nearest Neighbor**: The cluster dissimilarity is the *smallest* dissimilarity between pairs in the two sets
- **Copmlete Linkage/Farthest Neighbor**: The cluster dissimilarity is the *largest* dissimilarity between pairs in the two sets
- **Average Linkage**: The cluster dissimilarity is the *average* dissimilarity between pairs in the two sets



Another common approach is to base the cluster dissimilarity score on the *centroid* and *compactness* of the observations in the cluster.

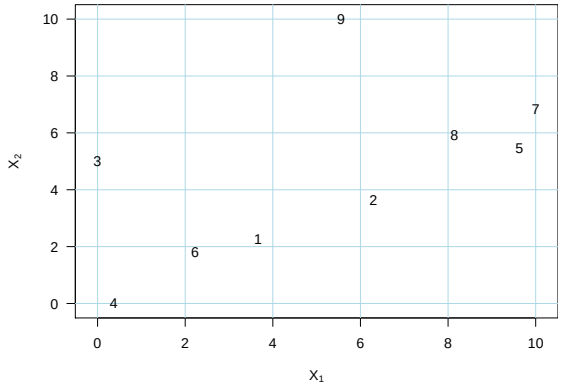
- **Centroid Linkage**: The cluster dissimilarity is the *distance between the cluster centroids*.
- **Ward's Linkage**: The cluster dissimilarity is the *increase in sum of squares* if the clusters were merged.

Let A and B be two clusters. Ward's linkage uses the dissimilarity score

$$\begin{aligned} W(A,B) &= \sum_{i\in A\cup B} \|x_i - m_{A\cup B}\|^2 - \sum_{i\in A} \|x_i - m_A\|^2 - \sum_{i\in B} \|x_i - m_B\|^2 \\ &= \frac{n_A n_B}{n_A + n_B} \|m_A - m_B\|^2 \end{aligned}$$

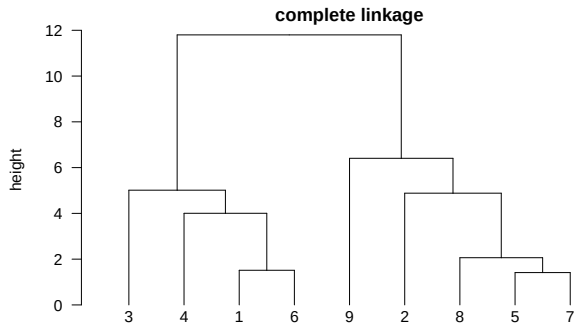
- $x = (x_1, \ldots, x_p)$
- m_A is the centroid of set A
- L_2 norm: $\|x\| = \left(\sum_j x_j^2\right)^{1/2}$
  - $\|x\|^2$ is the *squared* Euclidean distance
- See (Murtagh and Legendre 2011) for more details and R implementation (`ward.D` and `ward.D2`)

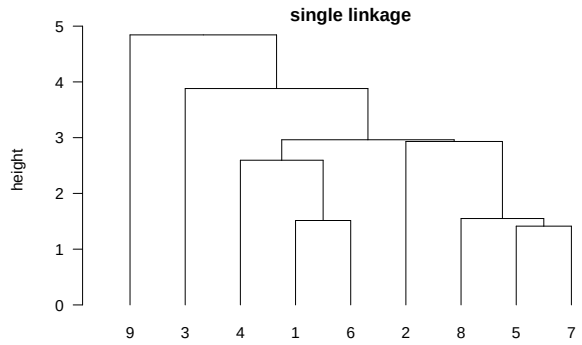
2.2.2 Example

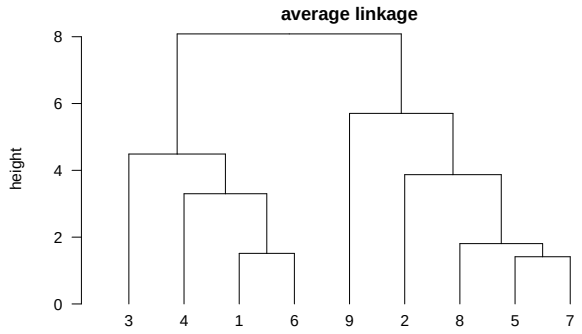


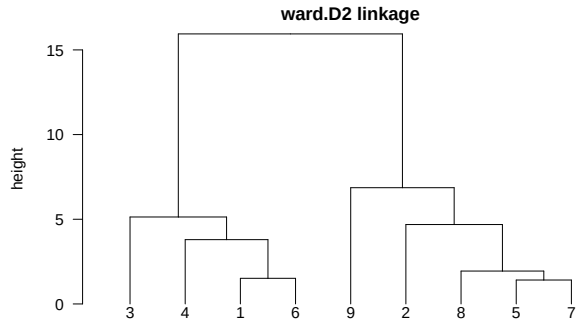
Distance Matrix

	1	2	3	4	5	6	7	8
2	2.96							
3	4.57	6.44						
4	4.00	6.95	5.01					
5	6.76	3.80	9.64	10.75				
6	1.51	4.46	3.88	2.60	8.25			
7	7.80	4.88	10.16	11.80	1.41	9.25		
8	5.77	2.93	8.20	9.77	1.55	7.20	2.06	
9	7.95	6.41	7.47	11.26	6.10	8.83	5.47	4.84







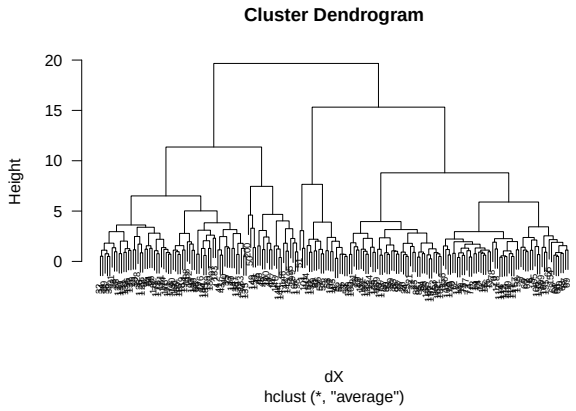


2.2.3 R Implementation

```
crabs = MASS::crabs                                  # get crabs data
crabsX = dplyr::select(crabs, FL, RW, CL, CW, BD)  # extract features
crabsY = paste(crabs$sp, crabs$sex, sep=":")         # get true labels
```

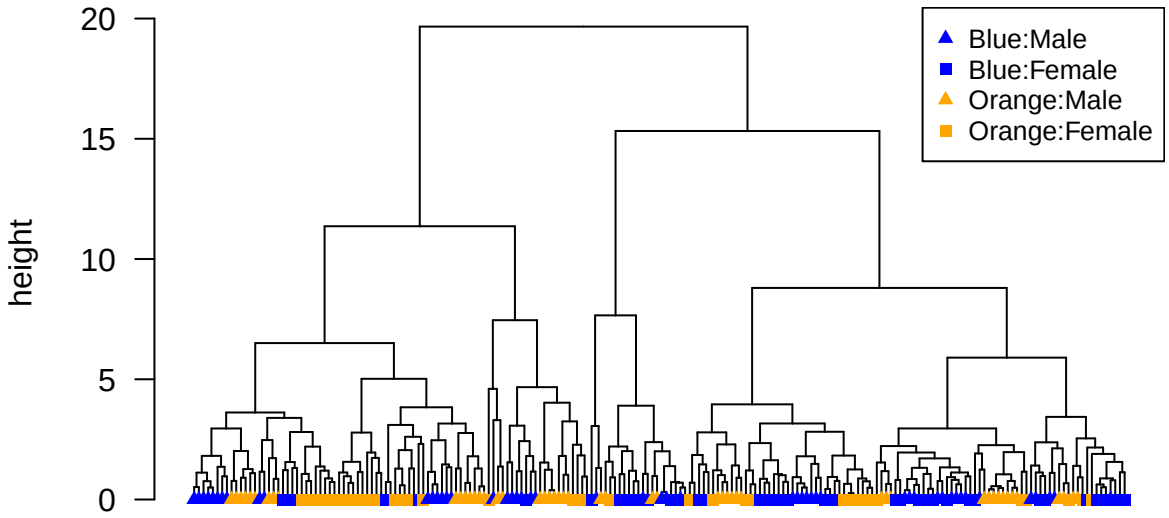
- The R function `hclust()` will run basic Hierarchical Clustering.
- It takes a *distance* object. A distance object is generated by calling `dist()` on a matrix or data frame.

```
dX = dist(crabsX, method="euclidean")   # calculate distance
hc = hclust(dX, method="average")       # average linkage
plot(hc, las=1, cex=.6)
```



- Some additional visualization is available if `hc` is converted to a `dendrogram` object

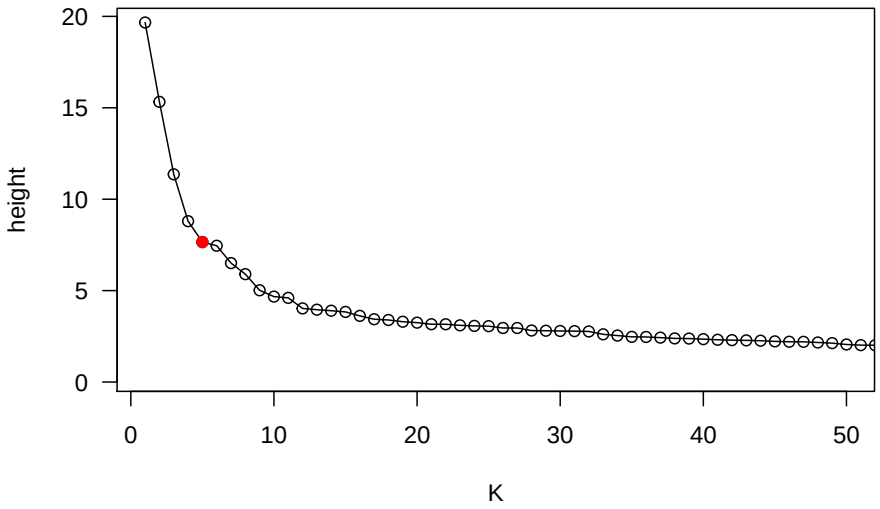
```
plot(as.dendrogram(hc), las=1, leaflab="none", ylab="height")
ord = hc$order
labels = crabsY[ord]
colors = ifelse(str_detect(labels, "B"), "blue", "orange")
shapes = ifelse(str_detect(labels, "M"), 17, 15)
n = nrow(crabsX)
points(1:n, rep(0, n), col=colors, pch=shapes, cex=.8)
legend("topright", c("Blue:Male", "Blue:Female", "Orange:Male", "Orange:Female"),
       pch=c(17, 15, 17, 15), col=c("blue", "blue", "orange", "orange"), cex=.8)
```



2.3 Choosing the number of clusters, K

- There are **many** approaches; see ITDM Chapter 7.5 for some approaches.
- One approach is to look for regions in the dendrogram where gaps/changes appear in the height of merges.
  - The height corresponds to the dissimilarity between the merged clusters, so a large jump in height corresponds to a high dissimilarity between the solutions for k and $k-1$ clusters
  - For Ward's method this corresponds to the change in the Sum of Squared Errors (SSE) for a merge
  - For Complete Linkage, the height corresponds to the dissimilarity between the most dissimilar points in the two clusters

```
n = length(hc$height)      # get number of merges
plot(n:1, hc$height, type='o', xlab="K", ylab="height", las=1,
     xlim=c(1, 50))
points(5, hc$height[n-4], col="red", pch=19) # K=5
```



Your Turn #1

1. Where do you have to cut the dendrogram to get $K = 5$ clusters?
2. Since we have labels, how could we evaluate how well the clustering performed?

- The function `cutree()` will extract the membership vector for a given *k* clusters or *h* height.

```
yhat = cutree(hc, k=5)
head(yhat)
#> 1 2 3 4 5 6
#> 1 1 1 1 1 2
```

- Confusion Matrix

```
table(est=yhat, true=crabsY)
#>    true
#> est B:F B:M O:F O:M
#>   1  10   5   2   4
#>   2  18  15   6  15
#>   3  14   7  13   6
#>   4   7  16  24  16
#>   5   1   7   5   9
```

2.4 Details and Considerations

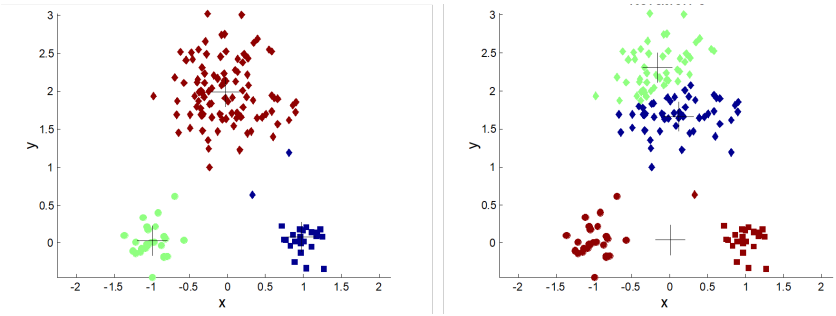
- Choice of dissimilarity/distance can be crucial
- Should the variables/features be standardized? E.g.,
  - Scale so all features have mean of zero and standard deviation of one. In R, this is done with `scale()` function.
  - Scale to be between $[0, 1]$
  - Scale by quantile.

- Should all the features/variables be used in to calculate the distance? Or more generally, should the features/variables be weighted?
  - See ESL 14.3.3 for an interesting discussion on feature weights
- Other transformations
  - PCA (Principal component analysis). See ISL chapter 10.1.
  - Autoencoders (based on ANNs)
- What type of linkage should be used?
- What value of K to use?
  - There are several, more quantitative, methods to select K. They all make strong assumptions.
  - See Gap Statistic paper for an approach that compares each solution to what is obtained in a *null* model (of no clustering)
  - Could consider a resampling (e.g., bootstrap or cross-validation) approach
- **These can have a substantial impact on your resulting analysis**
- View clustering results with skepticism
  - Results may not have much stability. Vary the data just a little and can get a very different solution
  - Most clustering results that you will see are based on trying many different K, distance/dissimilarity, and linkage methods, to get a pleasing solution
  - this hunting (p-hacking) does not usually lead to repeatable patterns in the data

3 K-means clustering

3.1 Prototype Methods

- Instead of seeking a hierarchical clustering structure, *prototype methods* seek a set of K points $(m_1, \ldots, m_K)$ that best represent the n data points.
  - The prototypes don't have to be existing observations
  - The K prototypes can be thought of as representing K *clusters*
  - Prototype methods can be used for data reduction (see ESL 14.3.9)



- The prototypes ($\{m_k\}$) should be determined by optimization. The prototypes should *best represent* the data.
- *Best* is, of course, determined by the application
  - For data compression, the choice of K and $\{m_k\}$ is based on the trade-off between fidelity and storage size.
  - For clustering, the choice of prototypes can give us different insights into the data structure
- The following concepts emerge:
  1. Each point should be represented by the prototype that *best represents* it.
  2. The prototypes should be determined so that they *best represent* the points assigned to it.

3.2 K-means

- K-means formalizes these concepts into an algorithm
- In K means, the K cluster *centroids* are the prototypes
  1. The kth centroid represents the n_k observations assigned to that cluster
  2. A point is assigned to the centroid that is *nearest*

Algorithm: K means

Initialize

1. Choose K initial centroids
   - $\{m_k\}_{k=1}^K$

Repeat until convergence:

2. Assign the observations to the *nearest* centroid (using squared Euclidean distance).

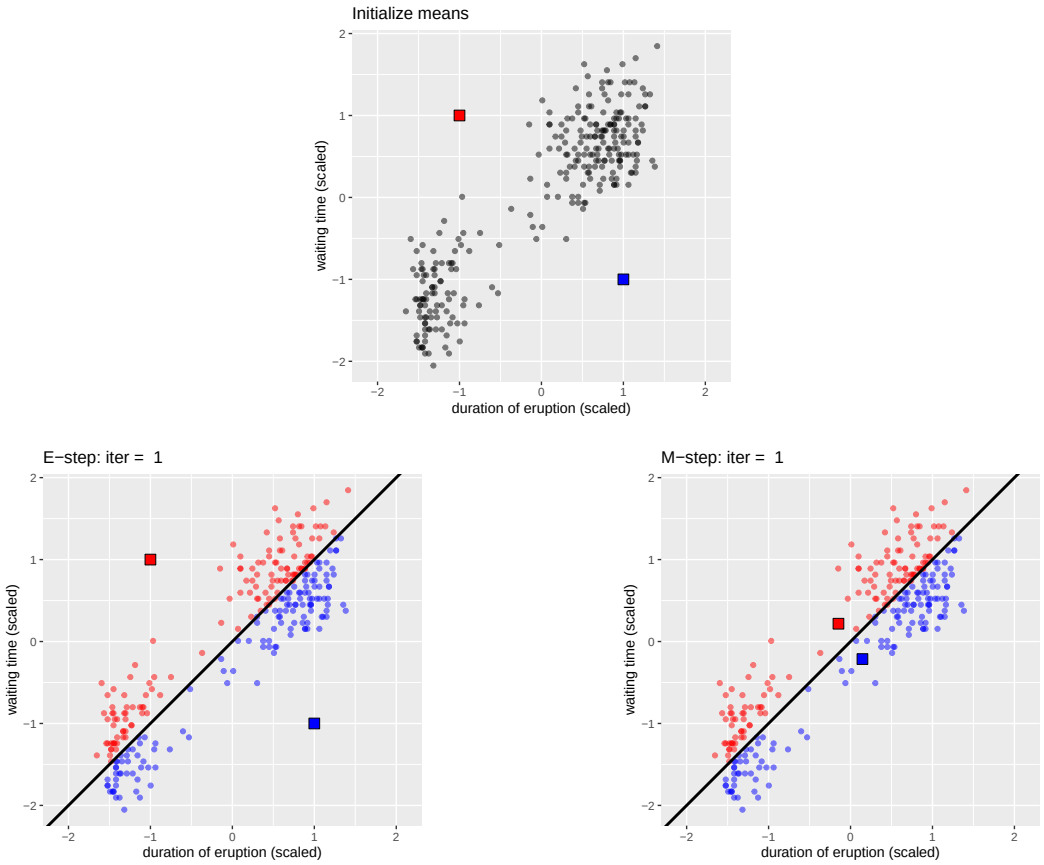
$$g_i = \underset{1 \le k \le K}{\arg\min} \|x_i - m_k\|^2$$

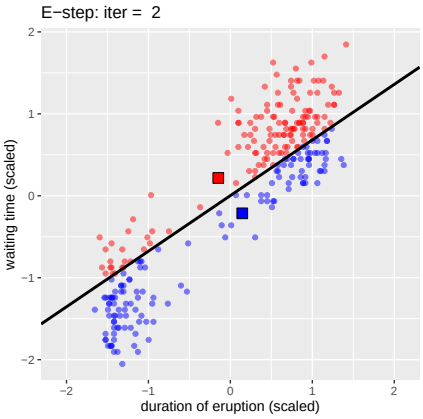
3. Update the centroids according to the points assigned to them.

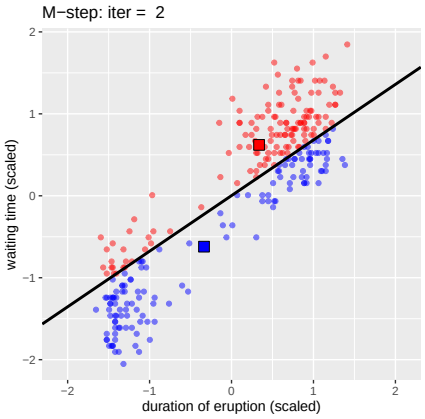
$$m_k = \arg\min_m \sum_{i:g_i=k} \|x_i - m\|^2$$

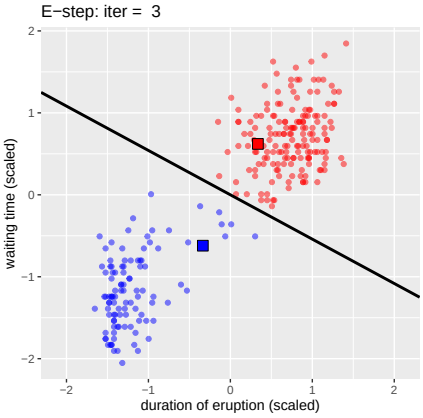
- Notation:
  - $x_i \in \mathbf{R}^p$ is the i^{th} observation (point in p dimensional Euclidean space)
  - $m_k \in \mathbf{R}^p$ is the k^{th} prototype
  - $g_i \in 1, 2, \ldots, K$ is the cluster label for observation i
  - The L_2 norm: $\|x - y\| = \left(\sum_{j=1}^p (x_j - y_j)^2\right)^{1/2}$

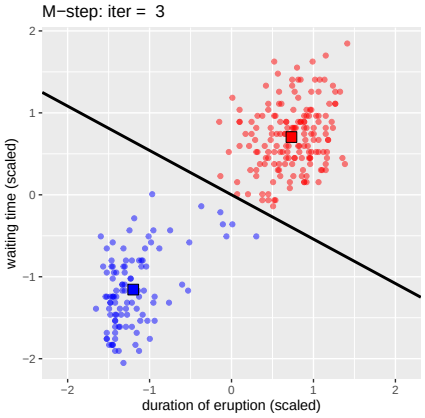
3.2.1 Example

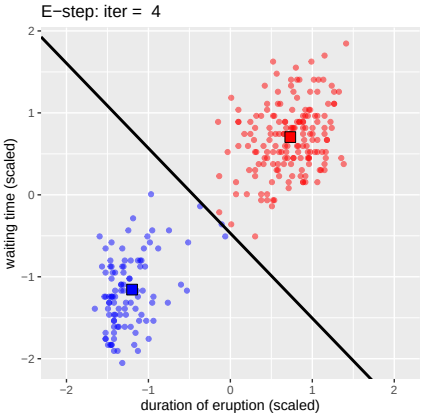


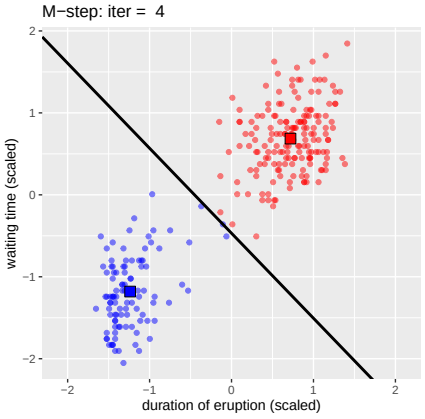

E−step: iter = 2
waiting time (scaled)
duration of eruption (scaled)

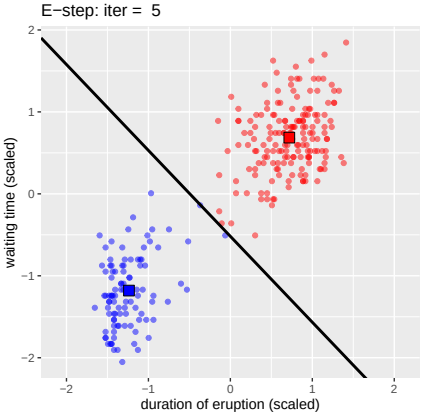

M−step: iter = 2
waiting time (scaled)
duration of eruption (scaled)

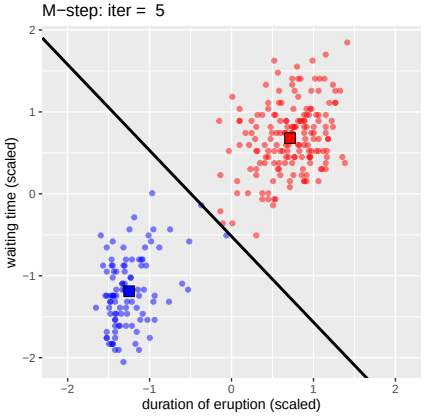

E−step: iter = 3
waiting time (scaled)
duration of eruption (scaled)

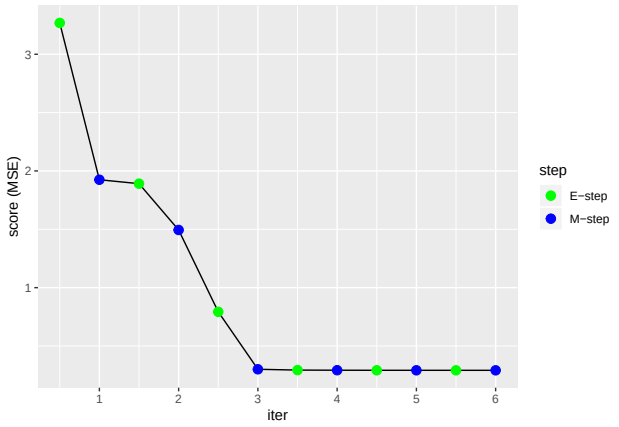

M−step: iter = 3
waiting time (scaled)
duration of eruption (scaled)


E−step: iter = 4
waiting time (scaled)
duration of eruption (scaled)


M−step: iter = 4
waiting time (scaled)
duration of eruption (scaled)

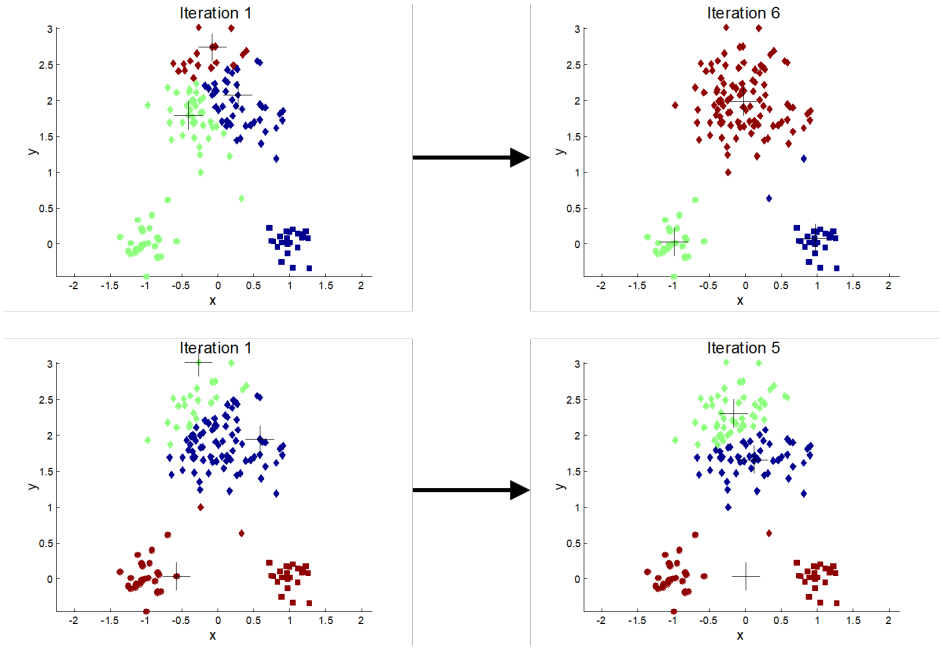

E−step: iter = 5
waiting time (scaled)
duration of eruption (scaled)


M−step: iter = 5
waiting time (scaled)
duration of eruption (scaled)



3.3 Initialization

- K-means may only find *local* solutions
- Important to run with several initializations
- There are some strategies to help
  - initialize with hierarchical clustering
  - sequentially choose prototypes that are farthest away from existing centroids



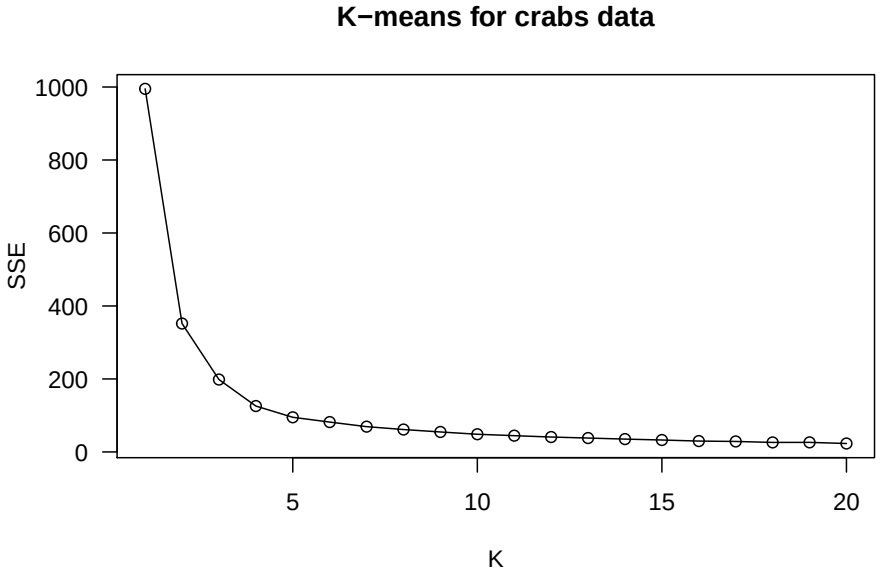
3.4 Choosing K

- Run K-means for several values of K and examine the SSE (sum of squared error); also known as *within cluster scatter*.
  - Or mean squared error (MSE = SSE/n)

```r
X = scale(crabsX)     # scale crabs data (mean=0, sd=1)

#-- Run kmeans for multiple K
Kmax = 20                                   # maximum K
SSE = numeric(Kmax)                         # initiate SSE vector
for(k in 1:Kmax){
  km = kmeans(X, centers=k, nstart=25)      # use 25 starts
  SSE[k] = km$tot.withinss                  # get SSE
}

#-- Plot results
plot(1:Kmax, SSE, type='o', las=1, xlab="K")
title("K-means for crabs data")
```



- Same warnings as with hierarchical clustering

Methods for choosing K

1. Look for *elbow* in plot of SSE vs. K
2. Use one of many statistical type tests (e.g. [Hamerly & Elkan, 2003])
3. Instead of K means, use the more flexible Gaussian Mixture Models!
    - Then use AI/BIC and other statistical measures to select K

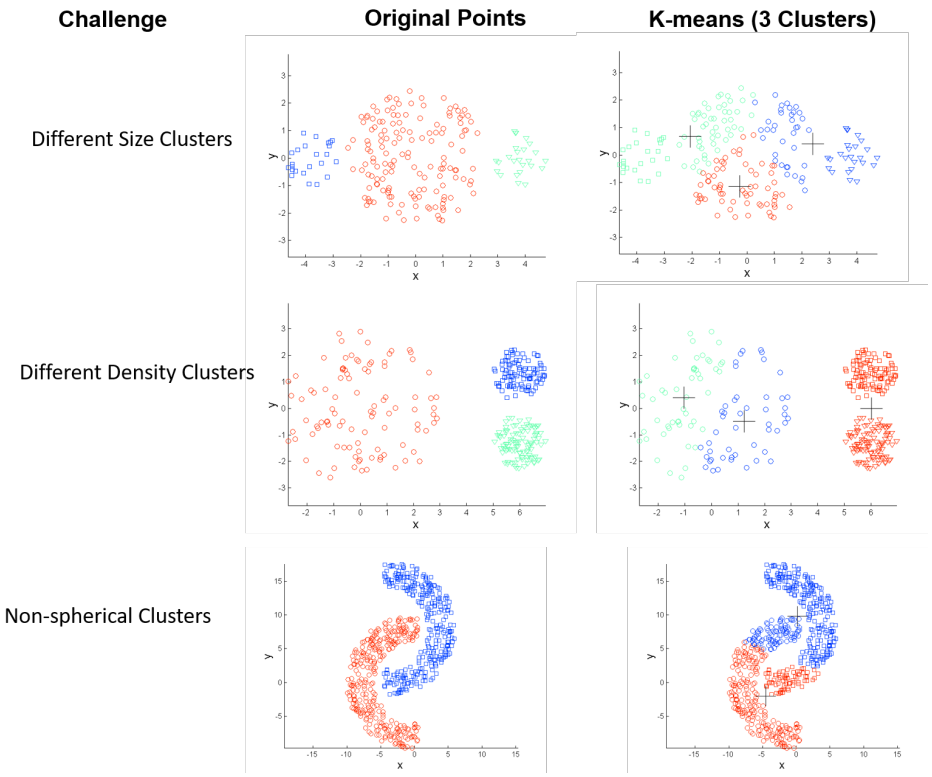
3.4.1 Evaluating K-means using known labels

```r
#-- Evaluate the classification ability of K-means
km = kmeans(X, centers=5, nstart=25)  # use K=5
table(true=crabsY, est=km$cluster)
     est
```

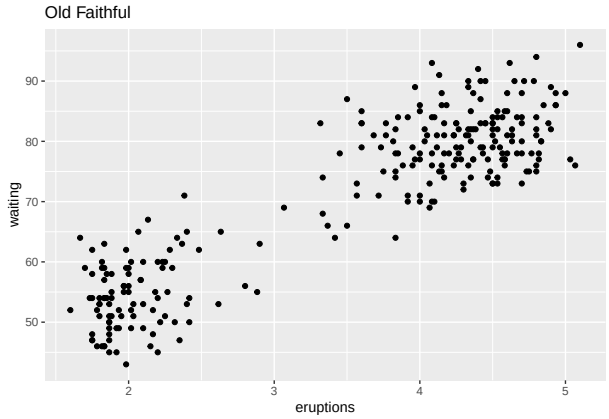
```
true    1  2  3  4  5
  B:F   6 16 12 15  1
  B:M  15 13  7 10  5
  O:F  15  5  2 13 15
  O:M  15 15  4  7  9
```

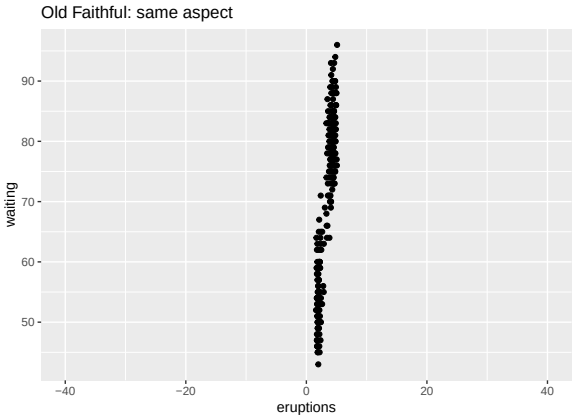
3.5 K means finds balanced, spherical clusters

- K means tends to find *balanced* clusters
  - clusters are around the same size (number of observations)
  - there is no mechanism to permit small/large clusters; everything is based on SSE
- K means tends to find *spherical* clusters
  - because Euclidean distance is used, clusters will be more spherical; larger K is needed to fit
  - **Importance of scaling to treat each variable equivalently** using Euclidean distance
- Important to scale appropriately. E.g., standardize all columns to have equal variance.
  - In R, the function `scale(X)` will transform all columns to have mean 0 and variance of 1.

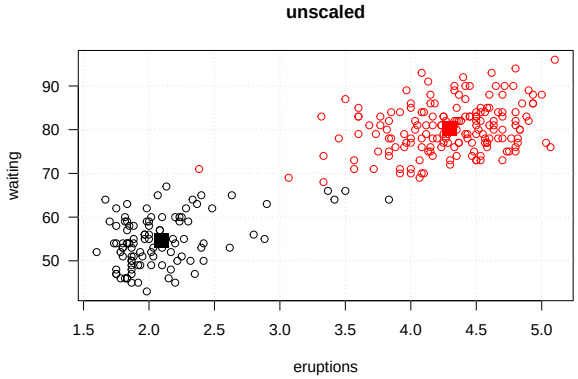


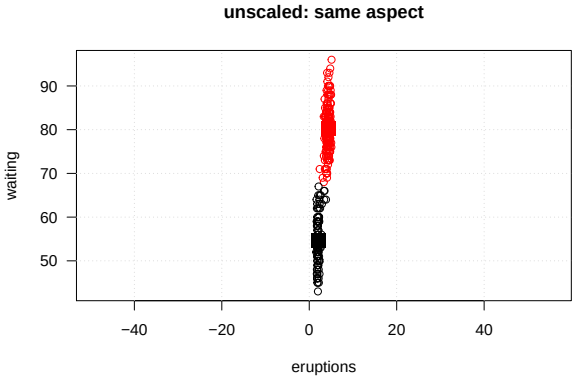
3.5.1 Old Faithful Example



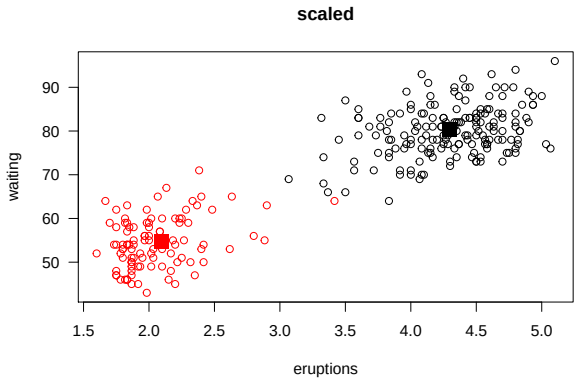


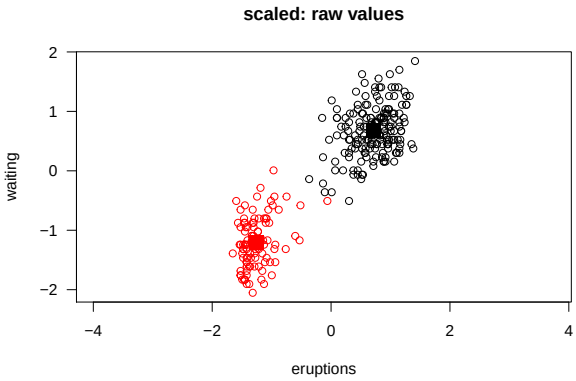
Unscaled Solution





Scaled Solution (mean=0, sd=1)





- Notice when we **do not** scale the data, the waiting time dominates the distance calculation
  - Thus, the *eruption* variable has practically no impact on the resulting clustering (i.e., clustering is only based on *waiting time*)
- By scaling the data (so each feature has the same variance), the resulting clustering can better adapt to the natural structure

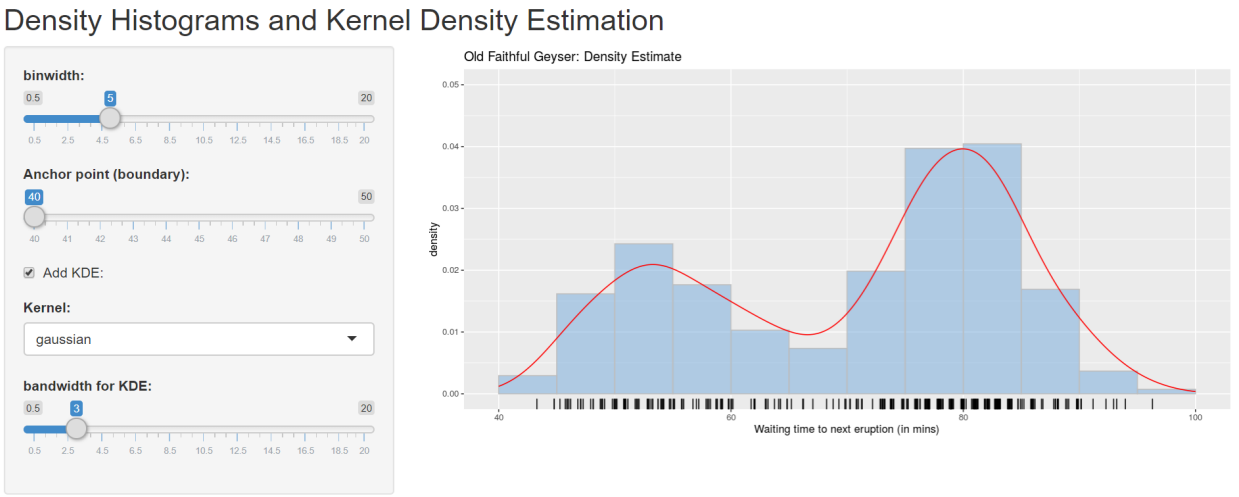
- Gaussian Mixture Models provide a natural way to handle input features that have different scales
  - I.e., use mixture models instead of k-means

4 Mixture Models

4.1 Example: Old Faithful

The old faithful geyser in Yellowstone National Park is one of the most regular geysers in the park. The waiting time between eruptions is between 35 and 120 mins.

- We used kernel density estimation (KDE) to estimate the density of waiting times in a previous section.
  - See the Shiny App



- We used KDE because we couldn't think of any parametric distribution that could capture the bi-modal structure of the waiting time distribution.
- While KDE can be a great approach, another idea is to use a mixture of parametric distributions.
  - These can also capture complex densities (like multiple modes), but also provide a nice intuitive interpretation (i.e., data comes from several sub-populations) which form the basis for clustering

4.2 Finite Mixture Models

Mixture models combine several parametric models to produce a more complex, yet easy to interpret distribution.

- natural representation when data come from different clusters/groups

$$\begin{aligned} f(x) &= \sum_{k=1}^{K} \pi_k \, f_k(x) \\ &= \pi_1 \, f_1(x) + \pi_2 \, f_2(x) + \ldots + \pi_K \, f_K(x) \end{aligned}$$

- $0 \leq \pi_k \leq 1, \sum_{k=1}^{K} \pi_k = 1$
- $f_k(x)$ is a parametric pdf/pmf

- Think of π_k as the probability that an observation comes from group k
- Think of $f_k(x)$ as the density of group k

- Usually written $f_k(x) = f(x; \theta_k)$
  - i.e., all component densities are of the same family, but have different parameter values

4.3 Gaussian Mixture Model (GMM): Univariate

Consider a two-component ($K = 2$) mixture of univariate Gaussian distributions (GMM):

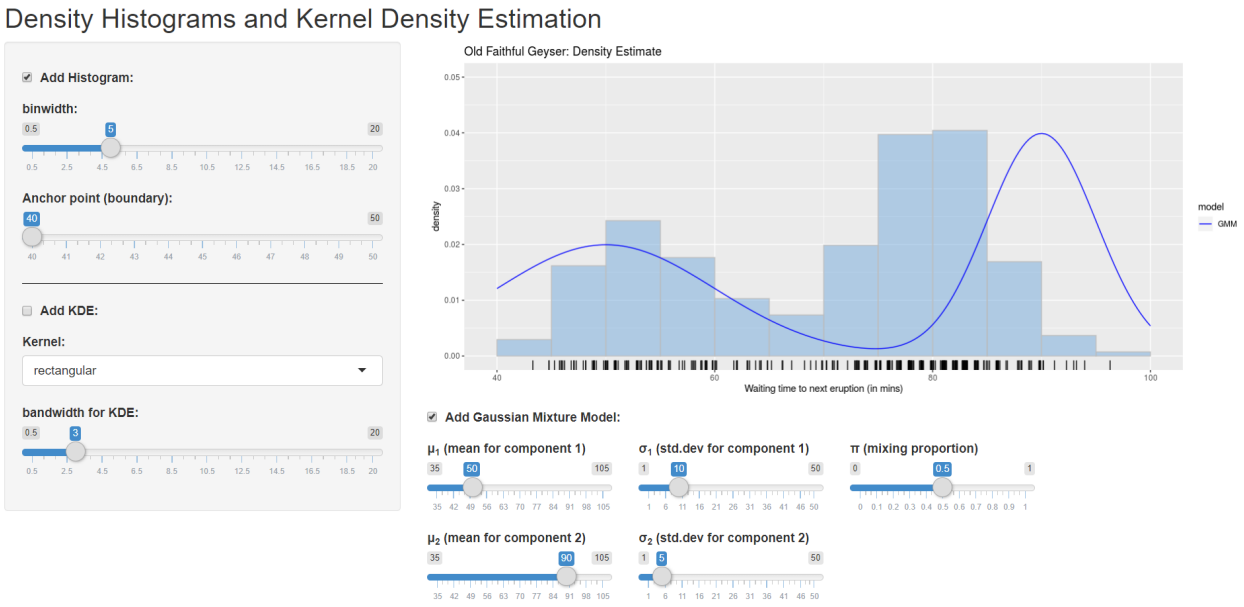
$$\begin{aligned} f(x;\theta) &= \pi f_1(x;\theta_1) + (1-\pi) f_2(x;\theta_2) \\ &= \pi \mathcal{N}(x;\mu_1,\sigma_1) + (1-\pi)\mathcal{N}(x;\mu_2,\sigma_2) \end{aligned}$$

- $\mathcal{N}(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$ is the pdf for a univariate Gaussian distribution
- $0 \leq \pi \leq 1$
- $\theta = (\pi, \mu_1, \sigma_1, \mu_2, \sigma_2)$
  - There are 5 parameters to estimate

4.3.1 Example: Old Faithful

Your Turn #2

Manually select parameters for the Gaussian Mixture Model. Use the new and updated Shiny app to help you decide.



4.4 Simulation (Generative Model)

- Data can be simulated from mixture models in a straightforward manner

Algorithm: Simulate Data from Gaussian Mixture Model (GMM)

To simulate n observations from a K component Gaussian mixture model (with known parameters):

$$\begin{aligned} f(x;\theta) &= \sum_{k=1}^{K} \pi_k f_k(x;\theta_k) \\ &= \sum_{k=1}^{K} \pi_k f_k(x;\theta_k) \mathcal{N}(x;\mu_k,\sigma_k) \end{aligned}$$

For i in $1, 2, \ldots, n$

1. Draw a label from a categorical distribution:

$$g_i \sim Cat(\pi_1, \ldots, \pi_K)$$

2. Draw the value from a Gaussian, conditional on the group label/parameters

$$x_i \mid g_i{=}k \sim N(\mu_k, \sigma_k)$$

To implement this more efficiently in R, I would do the following:

1. Draw all n labels from a multinomial distribution:

$$g \sim MN(n, \pi_1, \ldots, \pi_K)$$

2. For k in $1, \ldots, K$
   - Draw n_k observations from $N(\mu_k, \sigma_k)$
   - where $n_k = \sum_{i=1}^{n} \mathbb{1}(g_i = k)$

- See the companion Clustering R code for an example.

Your Turn #3

1. What is the expected value of n_k, $E[n_k]$?
2. What is the expected value of X ($E[X]$ is the overall mean)?

4.5 EM Algorithm

The details are found in the assigned readings ESL 8.5.1 and Gaussian Mixture Models: 11.1-11.3, so we will review only the basics.

Notation:

- Let $g_i \in \{1, 2, \ldots, K\}$ be the (unknown) group/component identifier.
  - π_k is prior probability that any observation is from component k
- The observed data is $D = \{X_1, X_2, \ldots, X_n\}$
- The parameters $\theta = (\pi_1, \ldots, \pi_k, \theta_1, \ldots, \theta_k)$
  - note: because we have the constraint that $\sum_k \pi_k = 1$, we only need to estimate $K - 1$ of the π's

- The **responsibilities** are the posterior probability that event i came from component k:

$$\begin{aligned} r_{ik} &= \Pr(g_i = k|D, \theta) \\ &= \frac{P(D|g_i = k, \theta_k)\pi_k}{\sum_{j=1}^{K} P(D|g_i = j, \theta_j)\pi_j} \end{aligned}$$

- The responsibilities are weights: $\sum_{k=1}^{K} r_{ik} = 1 \; \forall i$, which represent the probability that events come from the components, conditional on the parameters θ.
- *EM* stands for *Expectation-Maximization*
  - *E-step*: calculate the responsibilities
  - *M-step*: estimate parameters using new responsibilities as weights
  - Iterate until convergence

Algorithm: EM Algorithm

Initialize

1. Set θ to something reasonable.

Repeat until convergence:

2. E-step: update r_{ik}, using θ, for $i = 1, 2, \ldots, n$ and $k = 1, \ldots, K$.
3. M-step: update θ using r_{ik} (maximizing the *expected* log-likelihood)

For Gaussian components:

- Estimate like usual (MLE), except with *weighted* observations:

$$\begin{aligned} \hat{n}_k &= \sum_{i=1}^{n} r_{ik} \\ \hat{\pi}_k &= \hat{n}_k / n \\ \hat{\mu}_k &= \frac{1}{\hat{n}_k} \sum_{i=1}^{n} r_{ik} x_i \\ \hat{\sigma}_k^2 &= \frac{1}{\hat{n}_k} \sum_{i=1}^{n} r_{ik} (x_i - \hat{\mu}_k)^2 \end{aligned}$$

- The above is specific to univariate Gaussian mixture models.
- The assigned reading gives the mathematical details about what exactly EM is doing (e.g., what the *expected likelihood* is and why it is *maximized*)