Stats 102B - Introducing EM

Miles Chen, PhD

Department of Statistics

Week 8 Friday

UCLA

Section 1

EM Algorithm for Gaussian Mixtures

Gaussian Mixtures

We generate some data from a mixture of three multivariate Gaussian distributions.

The data generating process:

- Randomly select a value representing the cluster: 1, 2, 3 with probability vector $\alpha = [0.5, 0.3, 0.2]$
- Generate a random value from the selected multivariate Gaussian distribution
- Component 1: $\mathcal{N}_2\left(\boldsymbol{\mu} = [0, 0], \boldsymbol{\Sigma} = \left[\begin{smallmatrix} .9 & 0 \\ 0 & .9 \end{smallmatrix}\right]\right)$
- Component 2: $\mathcal{N}_2\left(\boldsymbol{\mu} = [4, 4], \boldsymbol{\Sigma} = \left[\begin{smallmatrix} 1 & .9 \\ .9 & 1 \end{smallmatrix}\right]\right)$
- Component 3: $\mathcal{N}_2\left(\boldsymbol{\mu} = [-4, -4], \boldsymbol{\Sigma} = \left[\begin{smallmatrix} 2 & -1 \\ -1 & 2 \end{smallmatrix}\right]\right)$

Mixture Models

The mixture model can be represented with:

$$p(\mathbf{x}|\mathbf{\Theta}) = \sum_{k=1}^{K} \alpha_k p_k(\mathbf{x}|z_k, \theta_k)$$

In our example, we have a mixture of 2-dimensional Gaussian distributions.

- $\mathbf{x}$ is the point in 2-dimensional space
- $\mathbf{\Theta}$ collects all of the parameters from the different mixtures (both $alpha_k$ and all θ_k)
- α_k are the mixture weights representing the probability that $\mathbf{x}$ came from component k, where $\sum_{k=1}^{K} \alpha_k = 1$
- $p_k()$ are the probability density functions of component k with parameters θ_k. For our example this is the multivariate Gaussian PDF, and θ_k represents the mean vector $\boldsymbol{\mu}_k$ and variance matrix $\mathbf{\Sigma}_k$ of component k.
- z_k is a vector of K indicator variables where one value is 1 and the rest are 0s. This represents the class membership of $\mathbf{x}$

Generating the data: Specify the component properties

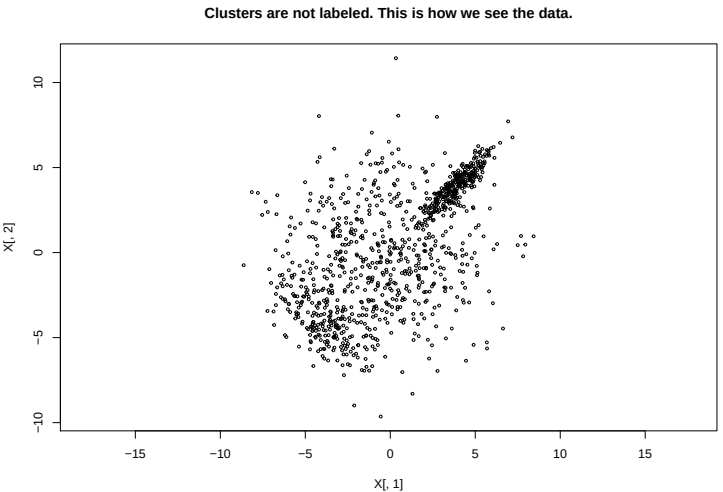
```r
# Specifying properties of the individual components
alpha <- c(.5, .3, .2) # mixture proportions
mean_1  <- c(0, 0)                # center of cluster 1
sigma_1 <- matrix(c(9, 0, 0, 9), 2) # sigma of cluster 1
mean_2  <- c(4, 4)
sigma_2 <- matrix(c(1, .9, .9, 1), 2)
mean_3  <- c(-4, -4)
sigma_3 <- matrix(c(2, -1, -1, 2), 2)
means  <- list(mean_1, mean_2, mean_3)
sigmas <- list(sigma_1, sigma_2, sigma_3)
```

Generating the mixture

```
# generating the mixtures
set.seed(1)
library(mvtnorm)
X <- matrix(0, ncol = 3, nrow = 1000)
for(i in 1:nrow(X)){
  k = sample(1:3,1, prob = alpha)
  X[i,] <- c(rmvnorm(1, means[[k]], sigmas[[k]]), k)
}
```

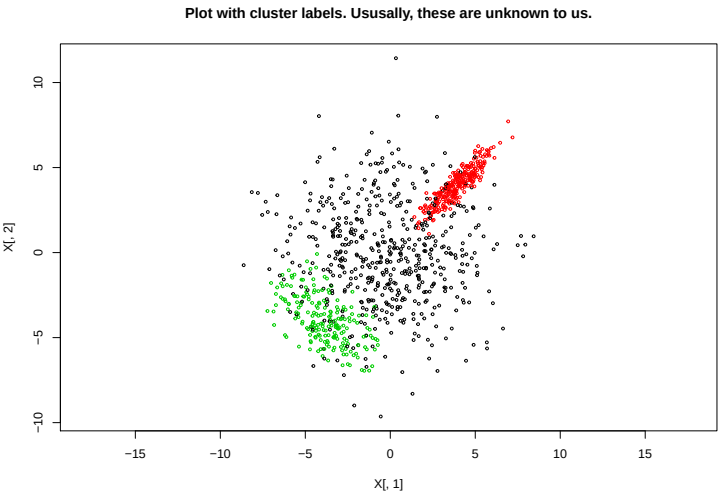
The resulting mixture

```
plot(X[,1], X[,2], cex = 0.5, asp = 1, main = "Clusters are not labeled. This
```

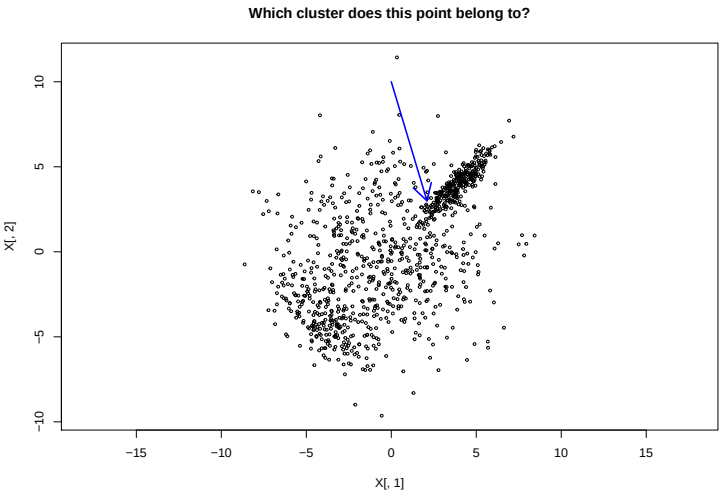


The mixture consists of three components

```
plot(X[,1], X[,2], col = X[,3], cex = 0.5, asp = 1, main = "Plot with cluster
```



Some points could be "assigned" to more than one cluster



Solution: Use a probabilistic assignment

Rather than strictly assigning a point to one cluster like in K-means clustering, we use a Bayes classifier to get a probabilistic assignment - what we might call a *membership weight*. Each point now has a vector of probabilities (that add to 1).

Once we have the membership weights, we calculate the parameters of each cluster distribution. We use a weighted mean and weighted variance-covariance matrix using the membership weights of the points in the clusters. This maximize the likelihood of the values "assigned" to the cluster.

We iterate back and forth between calculating membership weights (E-step) and recalculating the parameters of the cluster distributions (M-step).

It could be useful to think of the EM algorithm as a blend between k-means clustering and a Bayes classifier.

The E-step: Membership Weights

The membership weights are calculated using a Bayes classifier:

Given the x-values of the point (stored in $\mathbf{x}_i$) and all of the parameter values (the means and sigma matrices of all clusters stored in $\mathbf{\Theta}$), the weight/probability that point i belongs to cluster k is:

$$w_{ik} = P(z_{ik} = 1 | \mathbf{x}_i, \mathbf{\Theta}) = \frac{\text{likelihood} \cdot \text{prior}}{\text{marginal}} = \frac{p_k(\mathbf{x}_i | z_k, \theta_k) \cdot \alpha_k}{\sum_{m=1}^{K} p_m(\mathbf{x}_i | z_m, \theta_m) \cdot \alpha_m}$$

Note that the denominator (the marginal) is equal to the sum of the numerators across all possible clusters.

E-step: Membership Weights - $alpha_k$

$$w_{ik} = P(z_{ik} = 1|\mathbf{x}_i, \mathbf{\Theta}) = \frac{\text{likelihood } \cdot \text{ prior}}{\text{marginal}} = \frac{p_k(\mathbf{x}_i|z_k, \theta_k) \cdot \alpha_k}{\sum_{m=1}^{K} p_m(\mathbf{x}_i|z_m, \theta_m) \cdot \alpha_m}$$

α_k is the prior probability (from the previous iteration) that a point belongs to cluster k. It is equal to the proportion of "points that are assigned" to cluster k divided by the total number of points.

$$\alpha_k = \frac{N_k}{N}$$

The "number of points assigned" to cluster k, however, is not an integer. Rather it is the sum of the membership weights. That is:

$$N_k = \sum_{i=1}^{N} w_{ik}$$

E-step: Membership Weights - p_k

$$w_{ik} = P(z_{ik} = 1|\mathbf{x}_i, \mathbf{\Theta}) = \frac{\text{likelihood } \cdot \text{ prior}}{\text{marginal}} = \frac{p_k(\mathbf{x}_i|z_k, \theta_k) \cdot \alpha_k}{\sum_{m=1}^{K} p_m(\mathbf{x}_i|z_m, \theta_m) \cdot \alpha_m}$$

The likelihood is the probability density function evaluated for the values in $\mathbf{x}_i$ given the current estimates of the cluster's mean and sigma matrix. The PDF is the multivariate Normal distribution.

$$p_k(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_k|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)\right\}$$

Using the mvtnorm package in R, we can easily evaluate the above:

```
likelihood_k = dmvnorm(X_new, mean = xbar_k, sigma = var_k);
```

M-step - Calculating the estimates of $\boldsymbol{\mu}_k$

Once you have calculated the membership weights of each point to each cluster, we recalculate the "centroid" of each cluster according to the probabilistic weights of the points that have been "assigned" to each cluster:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N} w_{ik} \mathbf{x}_i$$

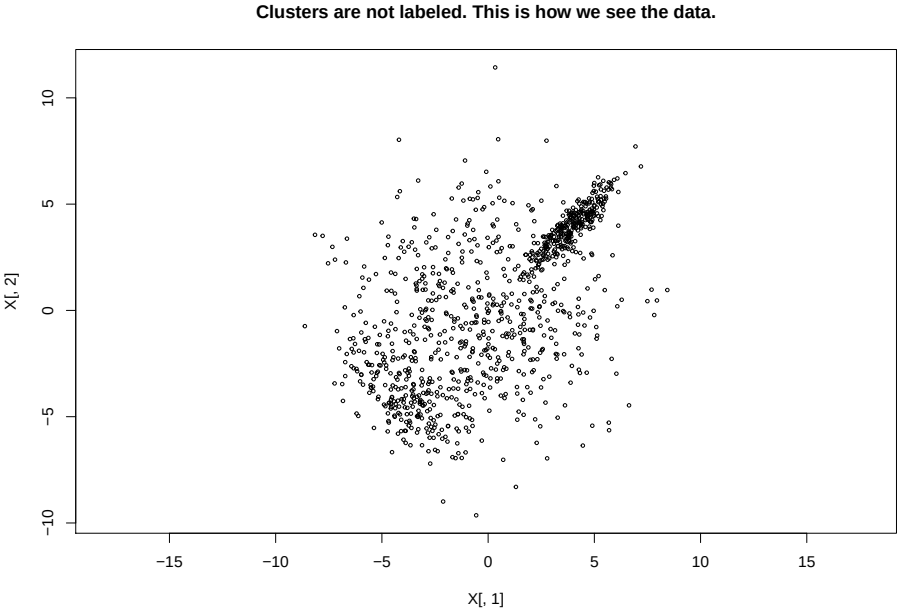
This is just a weighted mean of all the points.

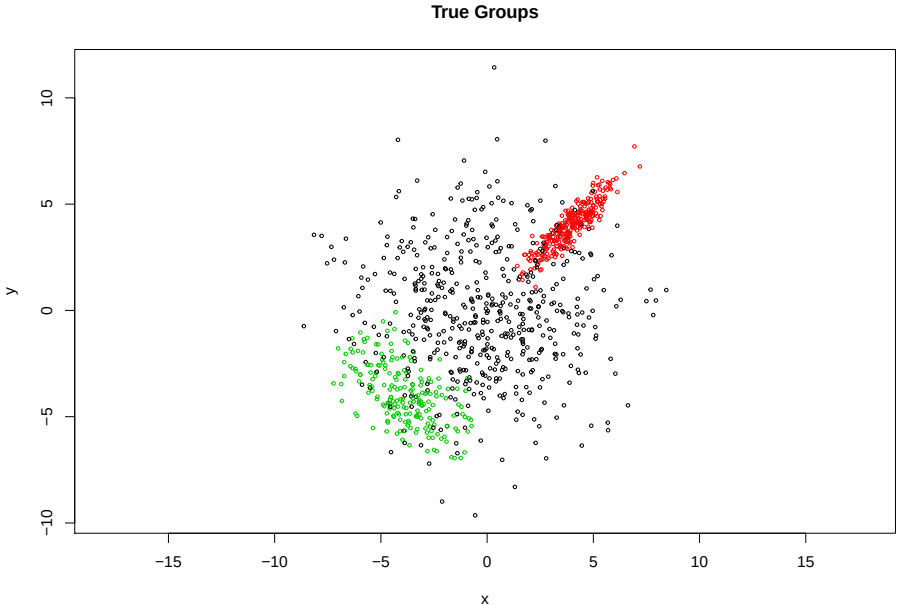
Points that are highly likely to be in cluster k, will have a membership weight w_{ik} close to 1 and will contribute more to the calculation of the "mean". While points that are unlikely to be in cluster k, will have w_{ik} close to 0 and will not contribute much to the value of the "mean".

M-step - Calculating the estimates of $\boldsymbol{\Sigma}_k$

We use the same idea to calculate the $\boldsymbol{\Sigma}$ matrices of each cluster, using the new $\boldsymbol{\mu}_k$ values we just calculated in the previous step:

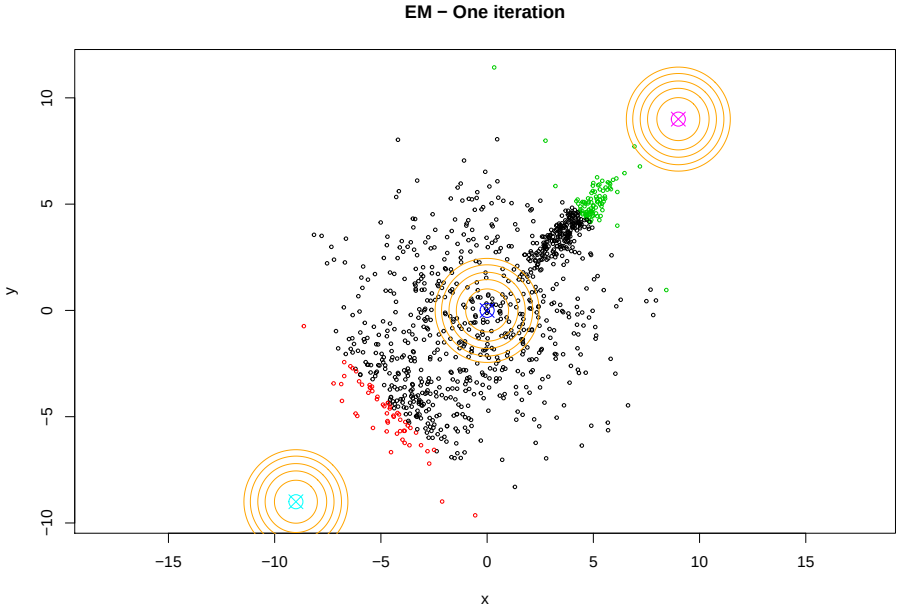
$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{i=1}^{N} w_{ik} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T$$





```r
# use these initial arbitrary values
N <- dim(dat)[1]  # number of data points
alpha <- c(0.33,0.33,0.34)  # arbitrary starting mixing parameters
mu <- matrix(  # starting means
    c(0,0,
      -9,-9,
      9,9),
    nrow = 3, byrow=TRUE
)
sig1 <- matrix(c(1,0,0,1), nrow=2)  # three arbitrary covariance matrices
sig2 <- matrix(c(1,0,0,1), nrow=2)
sig3 <- matrix(c(1,0,0,1), nrow=2)
```

I've intentionally hidden the rest of my code because you will code up the EM algorithm in your HW assignment.

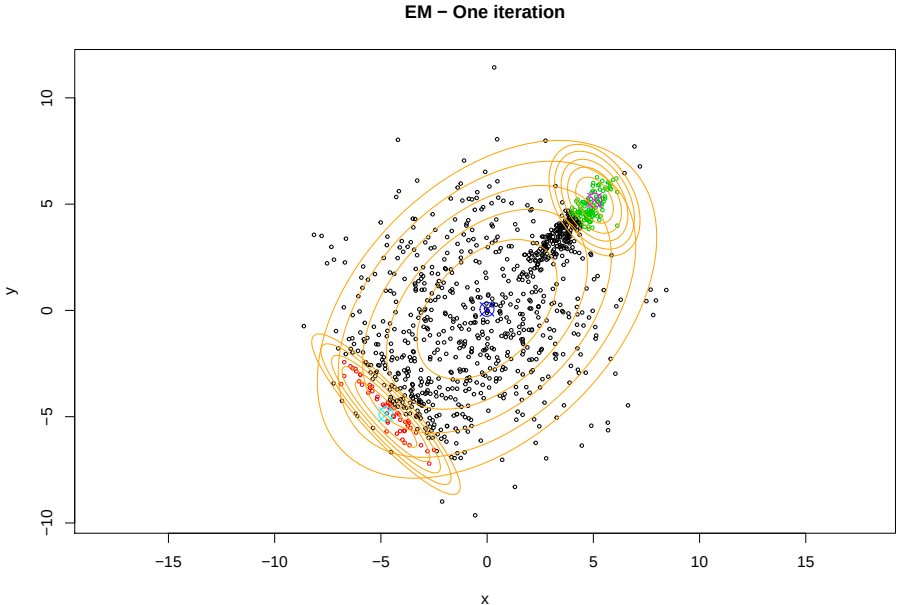


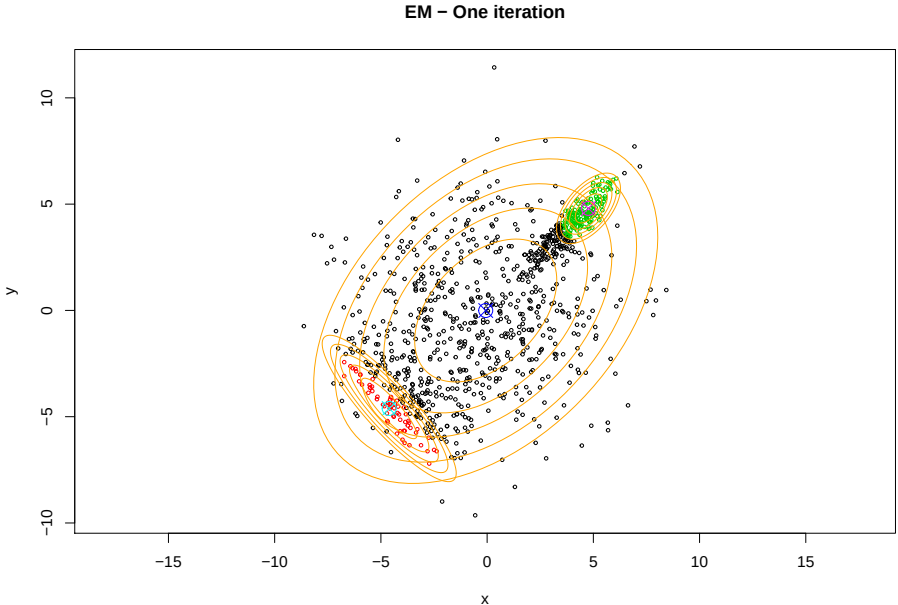
Before we continue

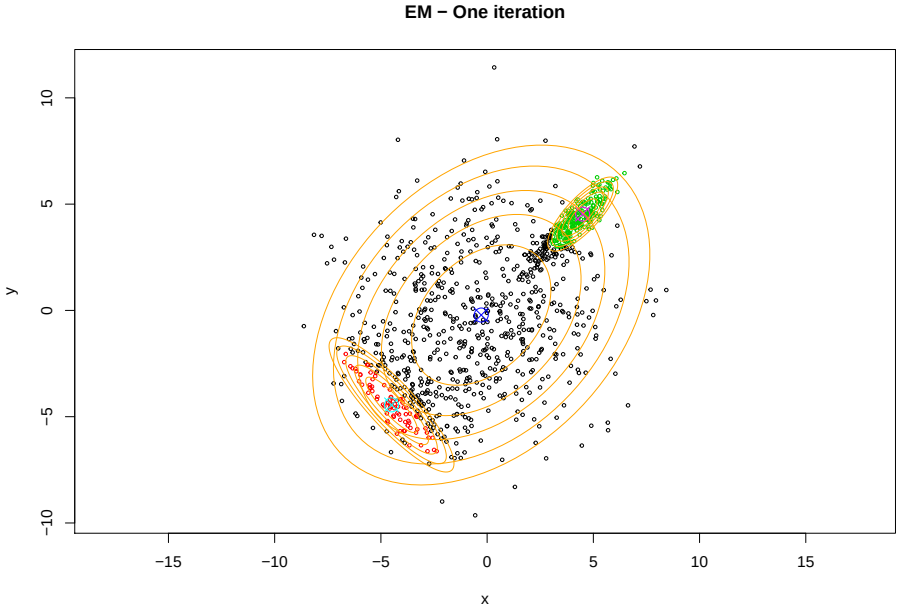
A few notes about the previous graph:

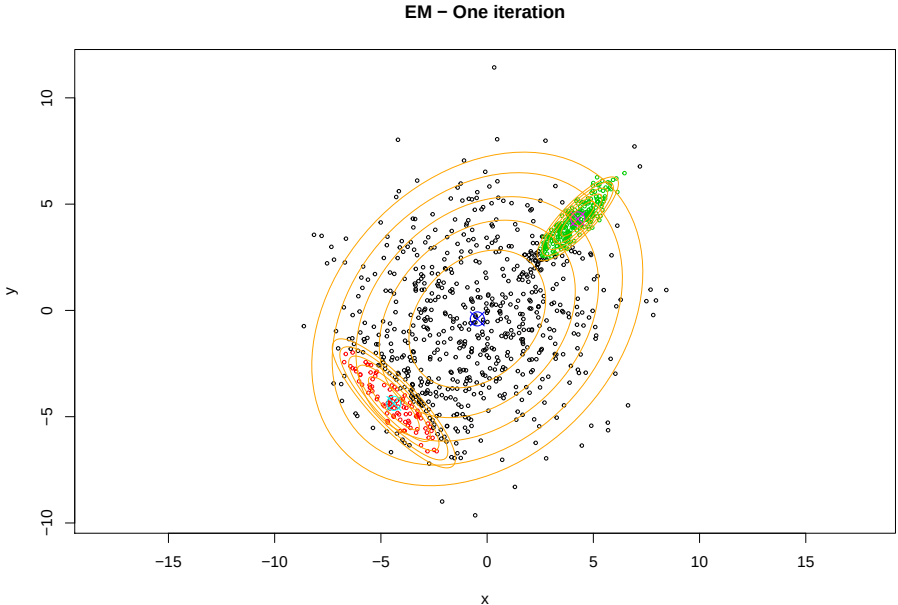
The circles/ellipses show the contour lines of the Multivariate Normal PDF. The initial parameter estimates have variance matrices of $\left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$. These are clearly bad estimates of the variances.

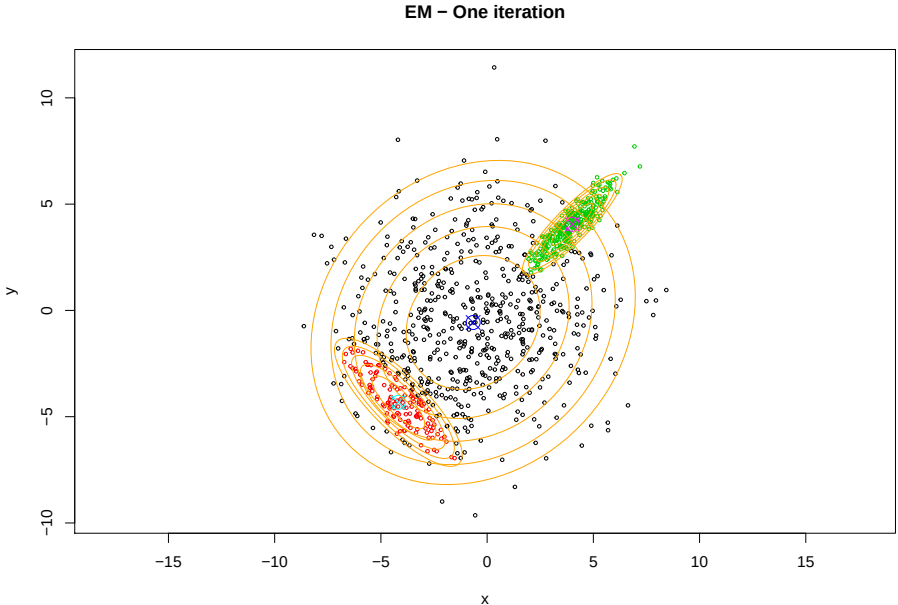
Also, the coloring of points is misleading. The graph colors the points according to the cluster which has the highest probability. However, each point has a probabilistic assignment. The points that are on the border between the red and black regions are more like "51% red/49% black" or "49% red/51% black", but they are represented with a single color in the graph.

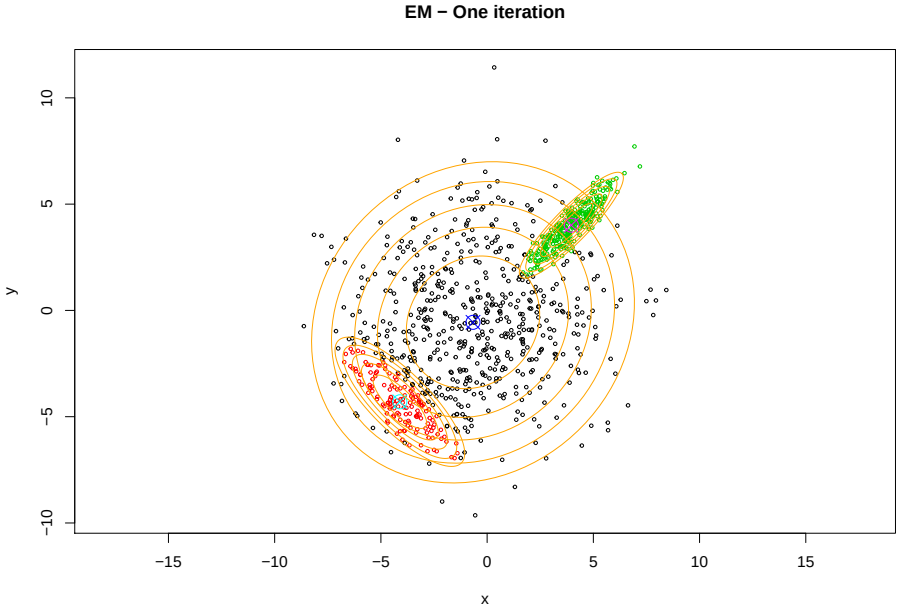


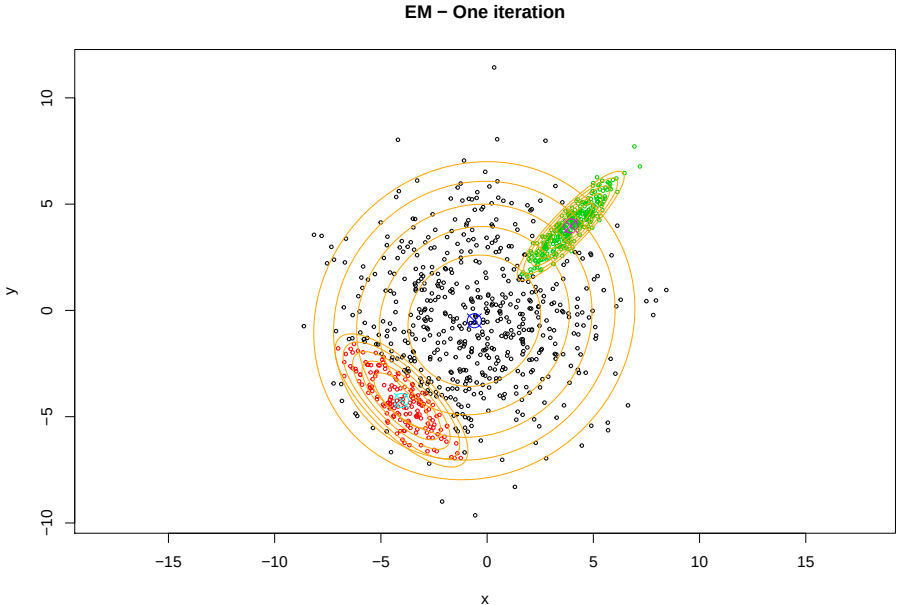


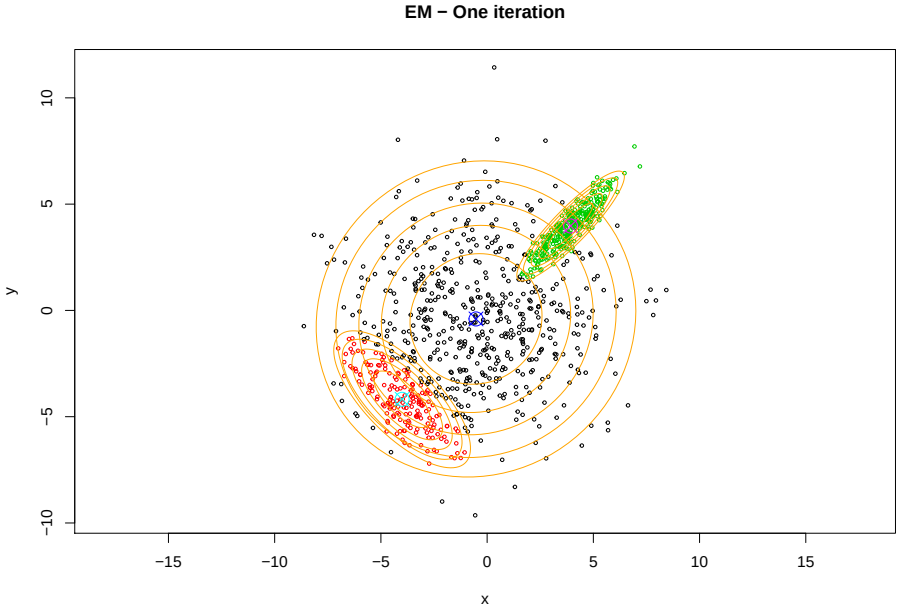


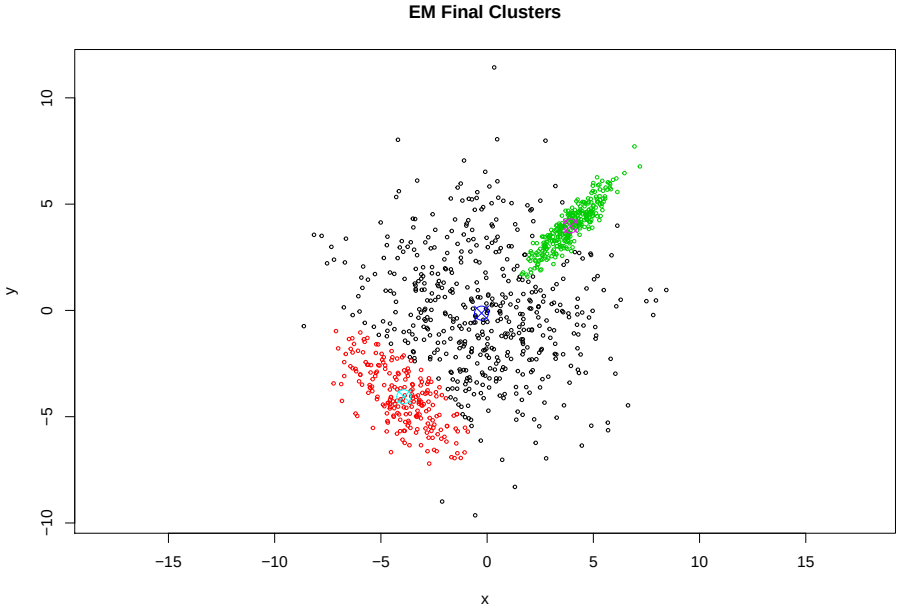


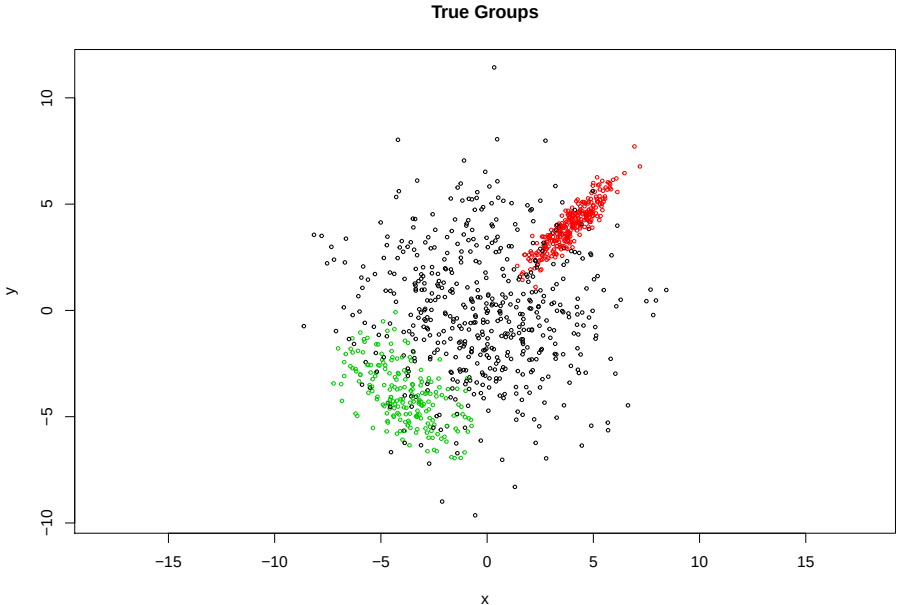






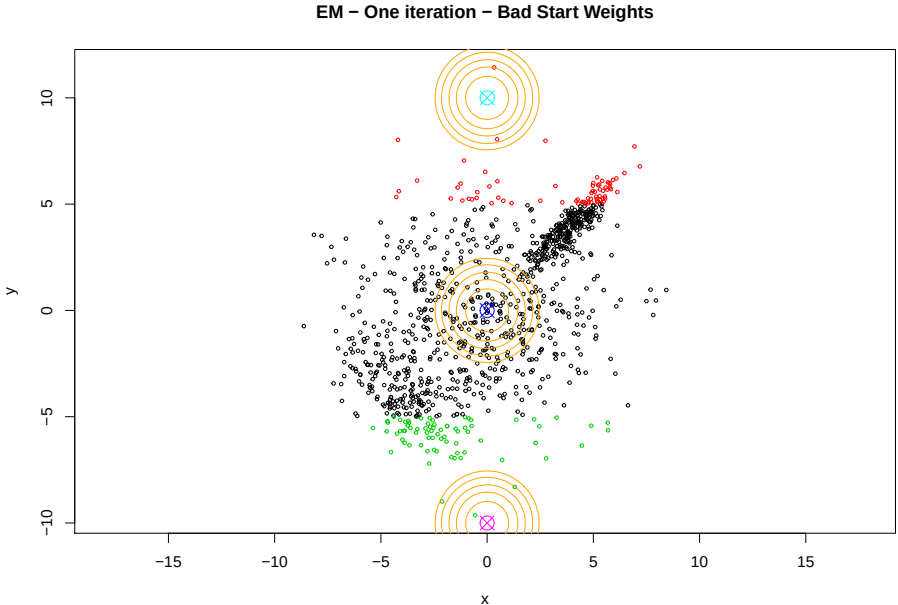


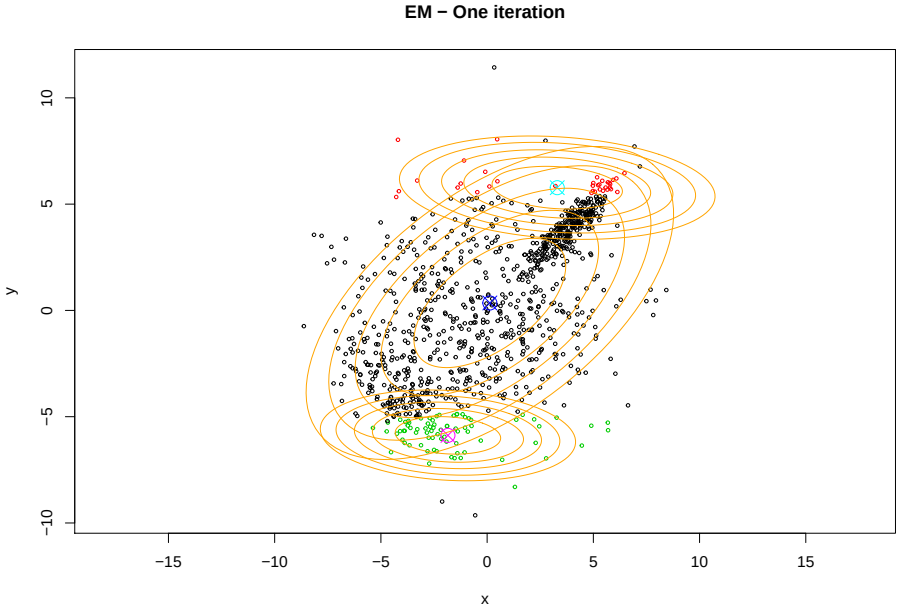


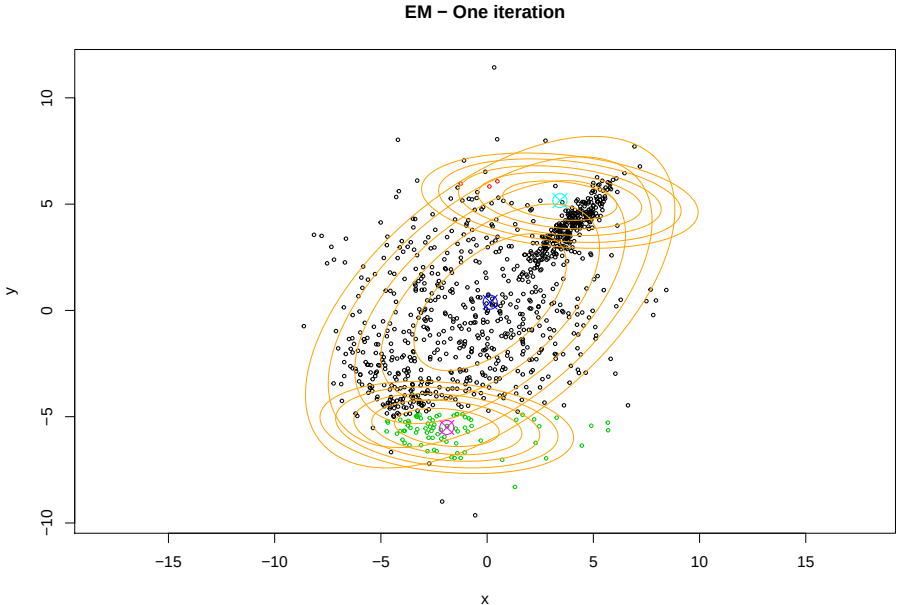


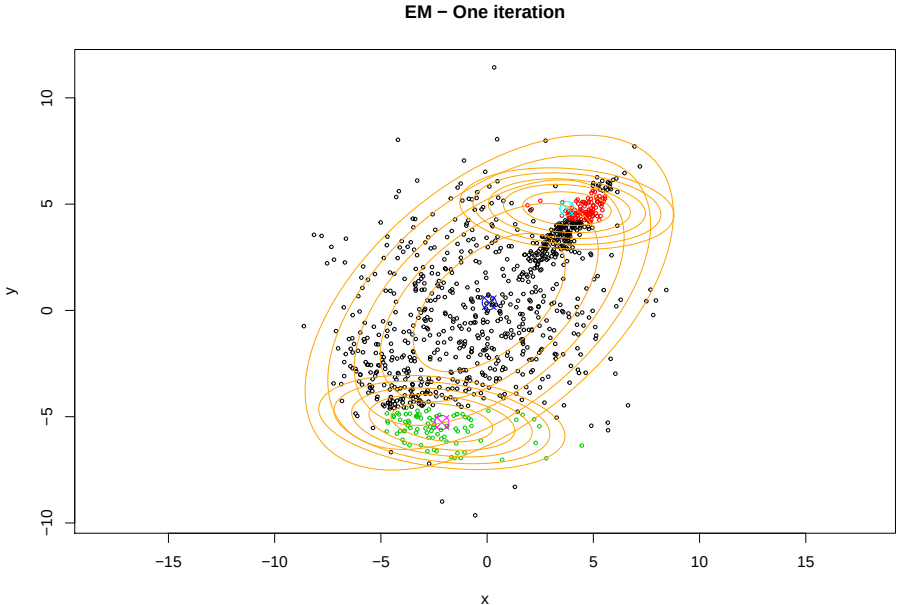
Different Starting Locations

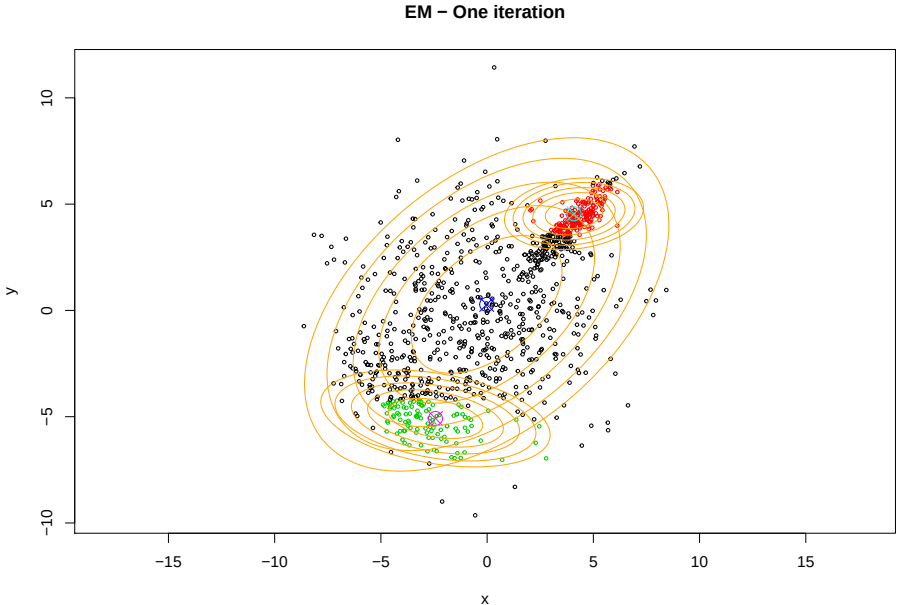
```
# use these initial arbitrary values
N <- dim(dat)[1]  # number of data points
alpha <- c(0.33,0.33,0.34)  # arbitrary starting mixing parameters
mu <- matrix(  # starting means
    c(0,0,
      0,10,
      0,-10),
    nrow = 3, byrow=TRUE
)
sig1 <- matrix(c(1,0,0,1), nrow=2)  # three arbitrary covariance matrices
sig2 <- matrix(c(1,0,0,1), nrow=2)
sig3 <- matrix(c(1,0,0,1), nrow=2)
```

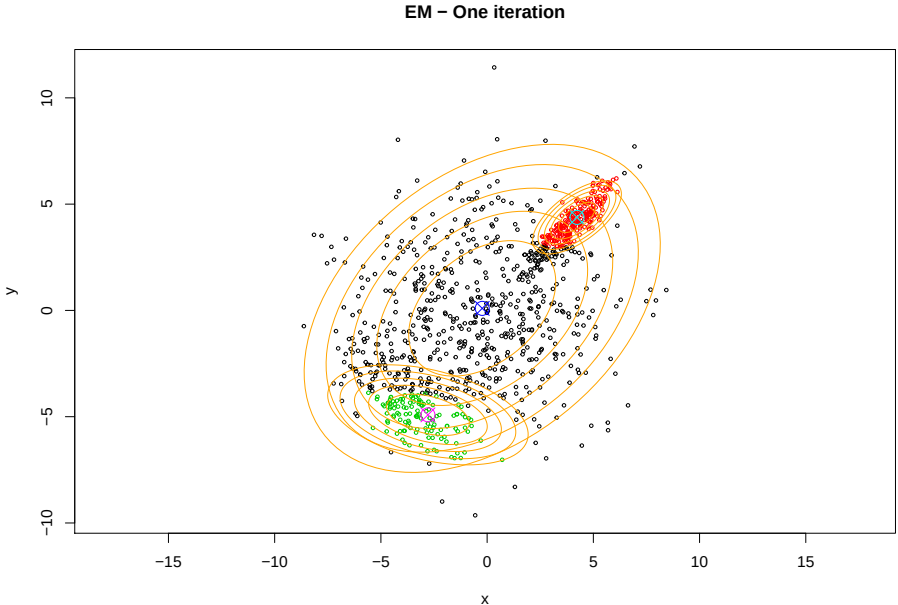


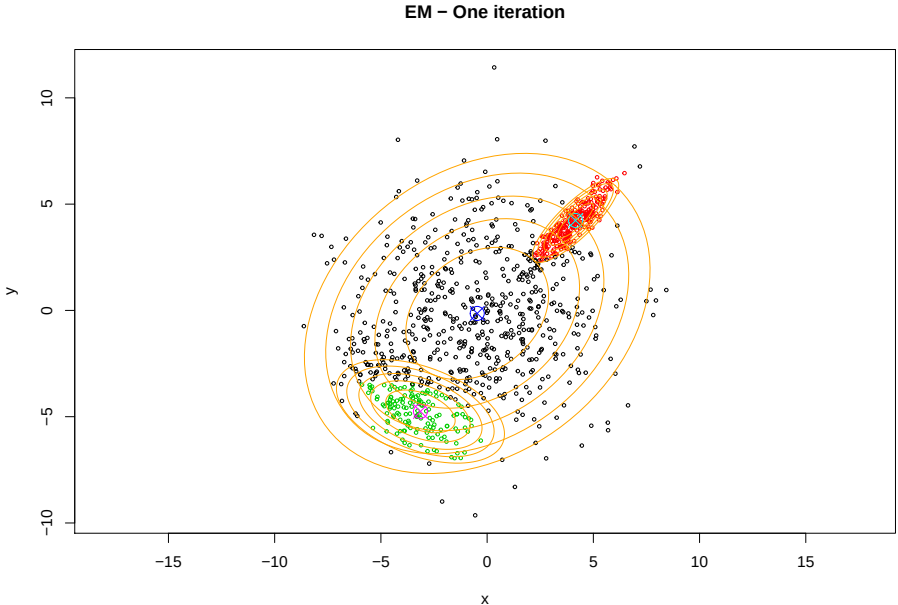


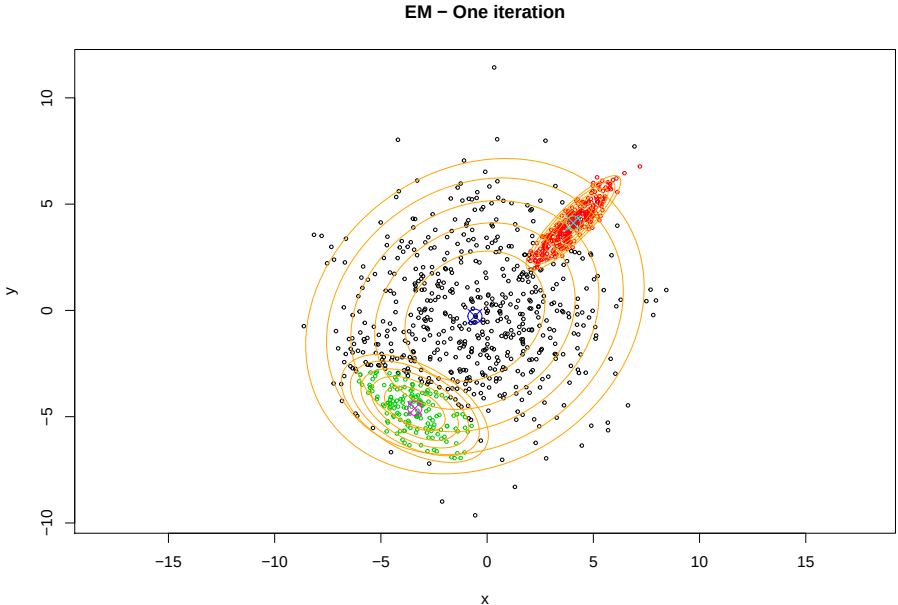


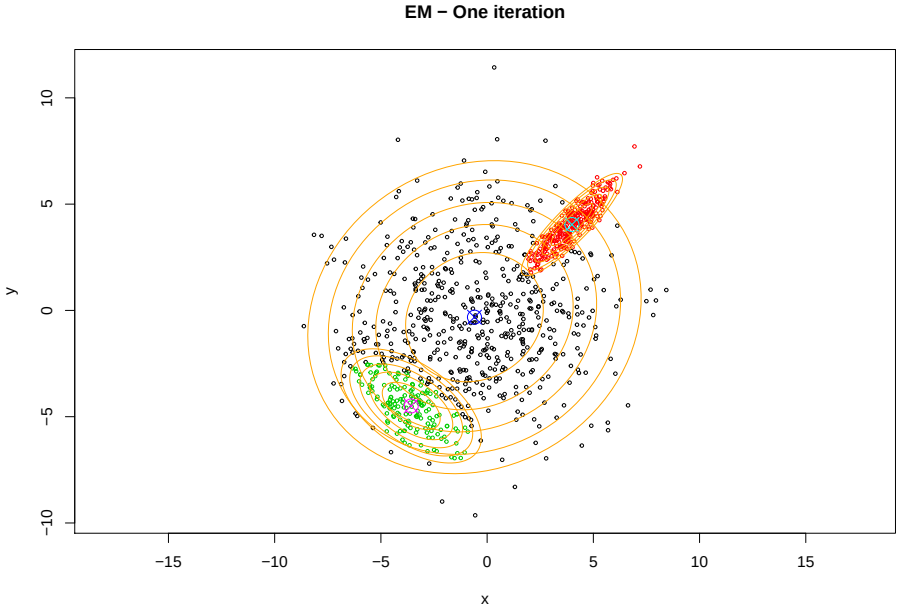


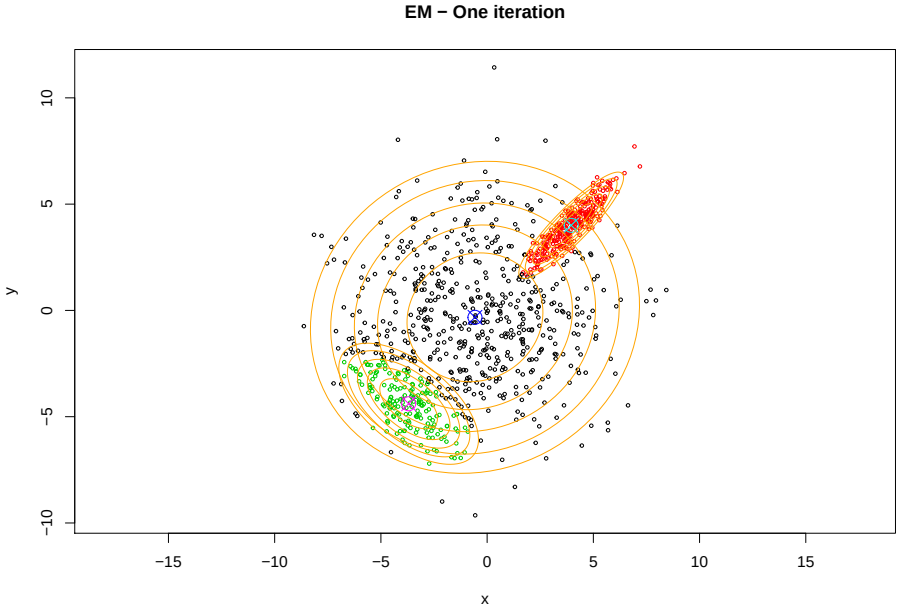


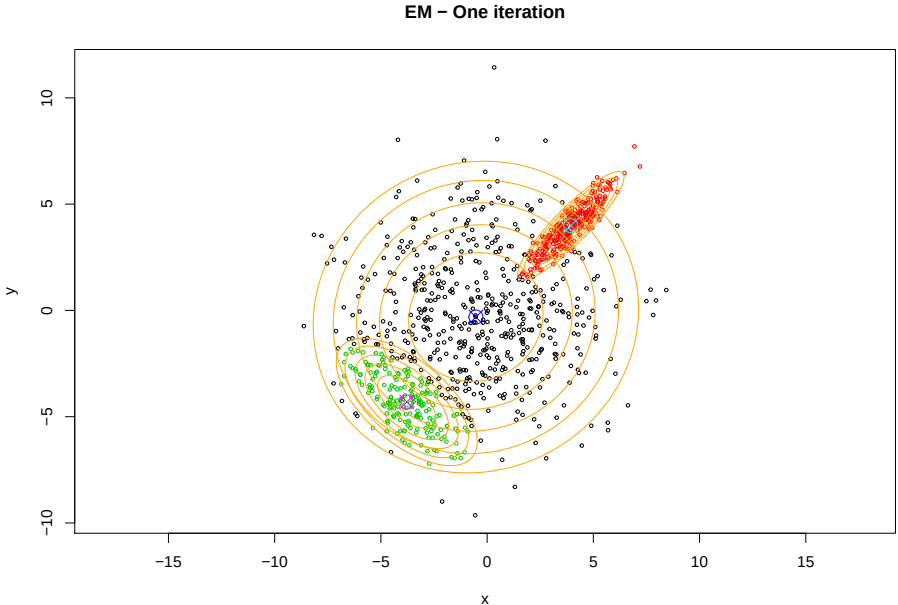


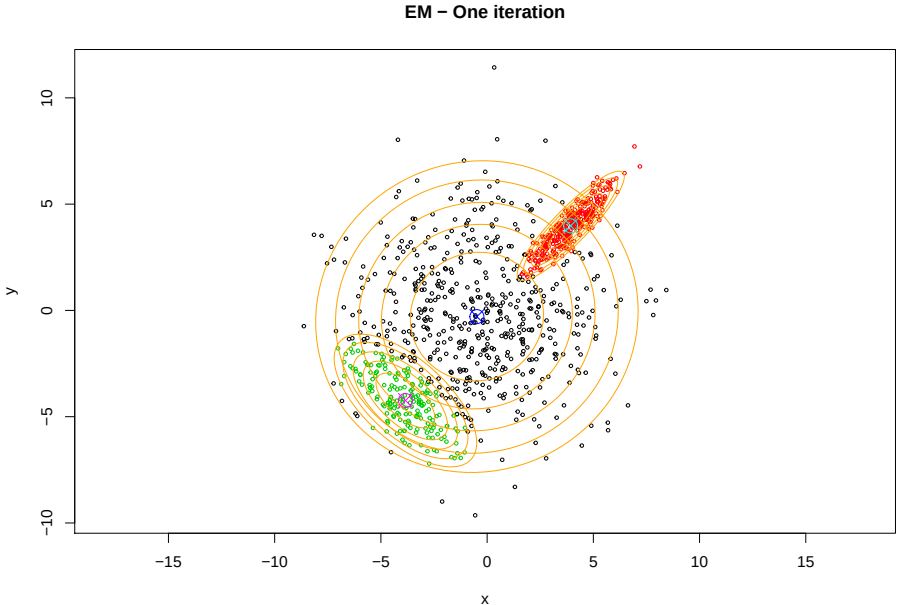


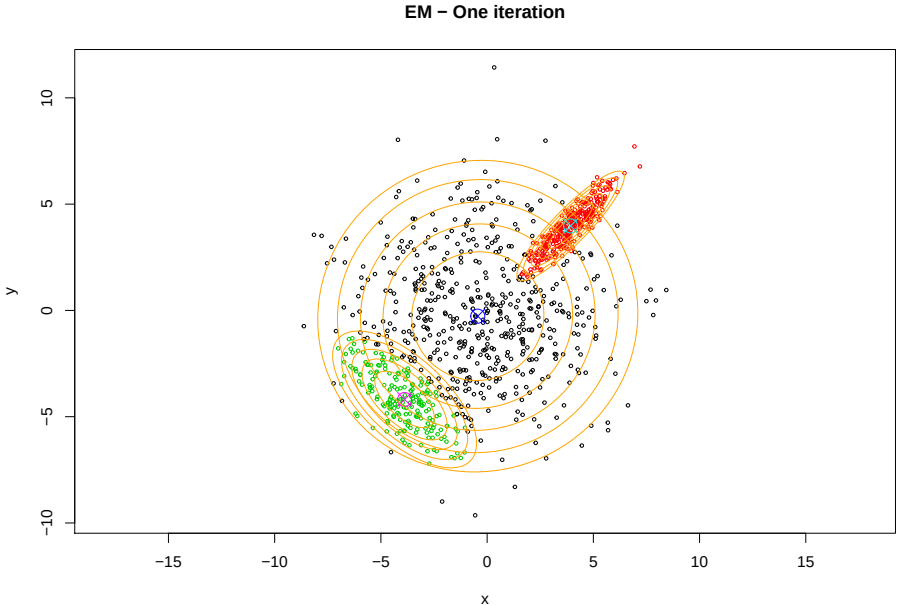


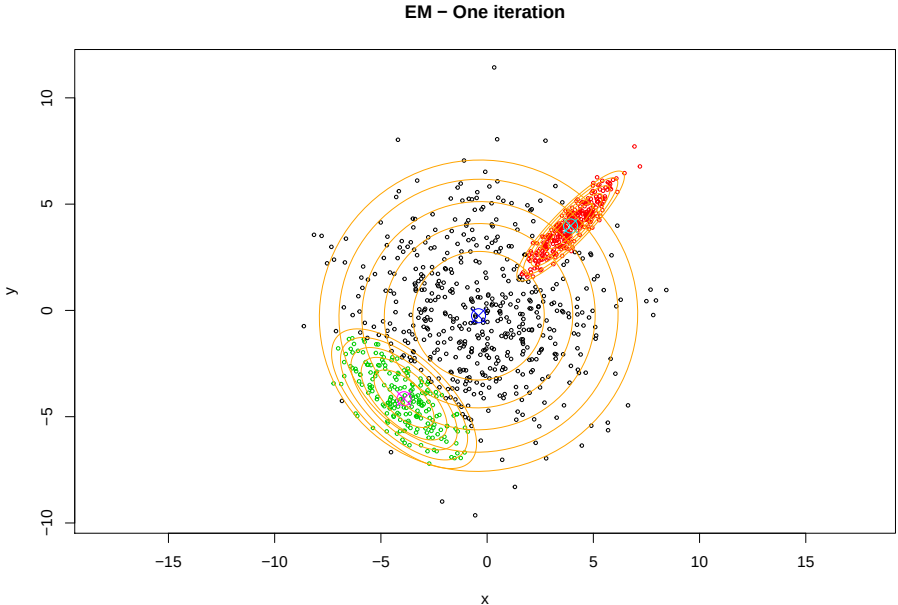


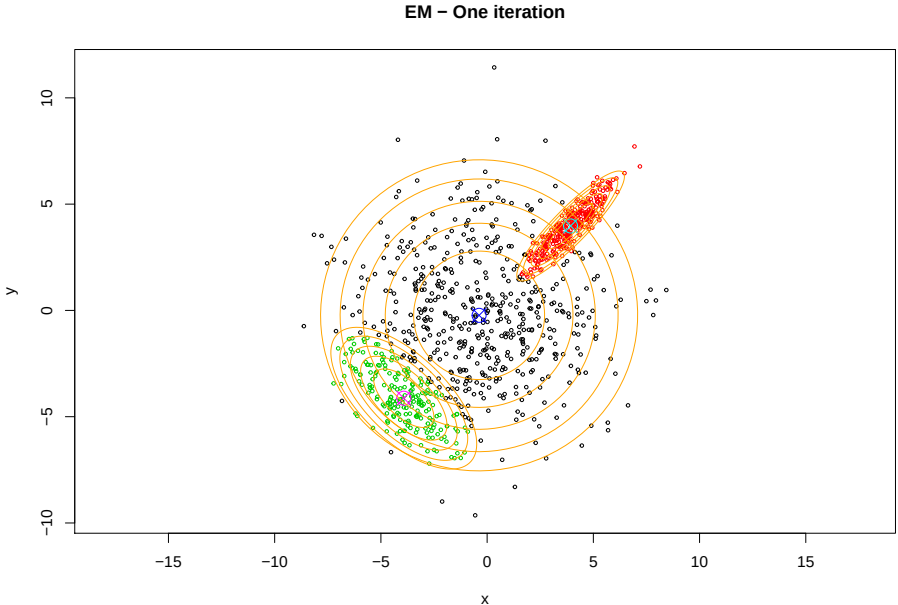


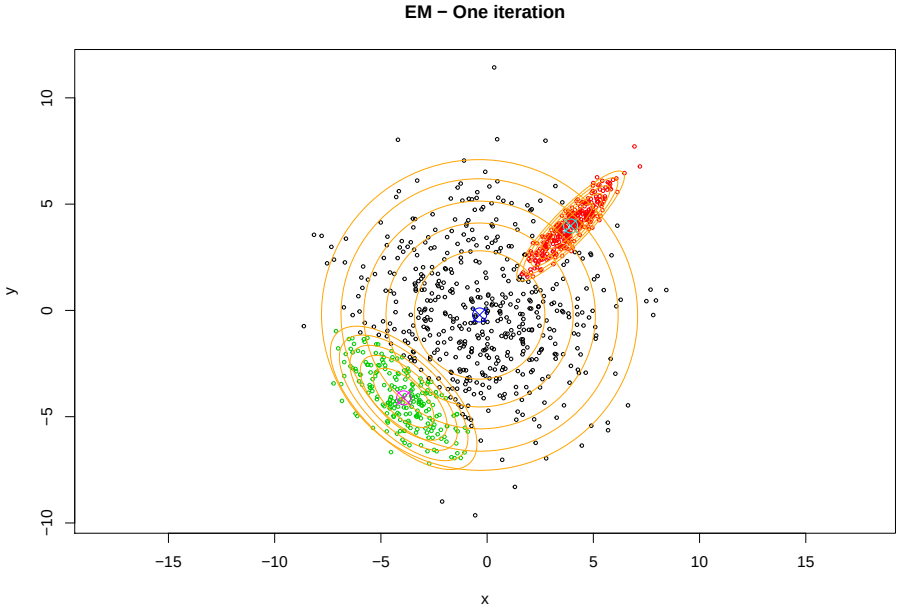












Fast forward to the end