Lab 3

Michał Ciach, Ewa Szczurek, Krzysztof Gogolewski, Senbai Kang

Clustering

The Dreadful Mouse

The Mouse is a type of a distribution that is designed to look cute, sound cute, and completely destroy the k-means algorithm. It's a mixture of three Normal distributions on a 2-dimentional space. The first distribution is centered at the origin. This is the head of the mouse. The other two are centered at points (-1, 1) and (1, 1). These are the ears. The ears have a common standard deviation, which is smaller than the standard deviation of the head. A simuated point belongs to head with a given probability. Formally, let M be an observation from the mouse, and $f_M(x, y)$ it's 2-dimentional density. Let H be a random point from the head, and E_1 and E_2 from the ears. Then, we have

$$H \sim \mathcal{N}((0, 0), \sigma_H^2 I),$$
$$E_1 \sim \mathcal{N}((-1, 1), \sigma_E^2 I),$$
$$E_2 \sim \mathcal{N}((1, 1), \sigma_E^2 I).$$

The density of a random point from the Mouse is given by

$$f_M(x, y) = p f_H(x, y) + \frac{1-p}{2} f_{E1}(x, y) + \frac{1-p}{2} f_{E2}(x, y),$$

where p is the probability that a given point belongs to the head.

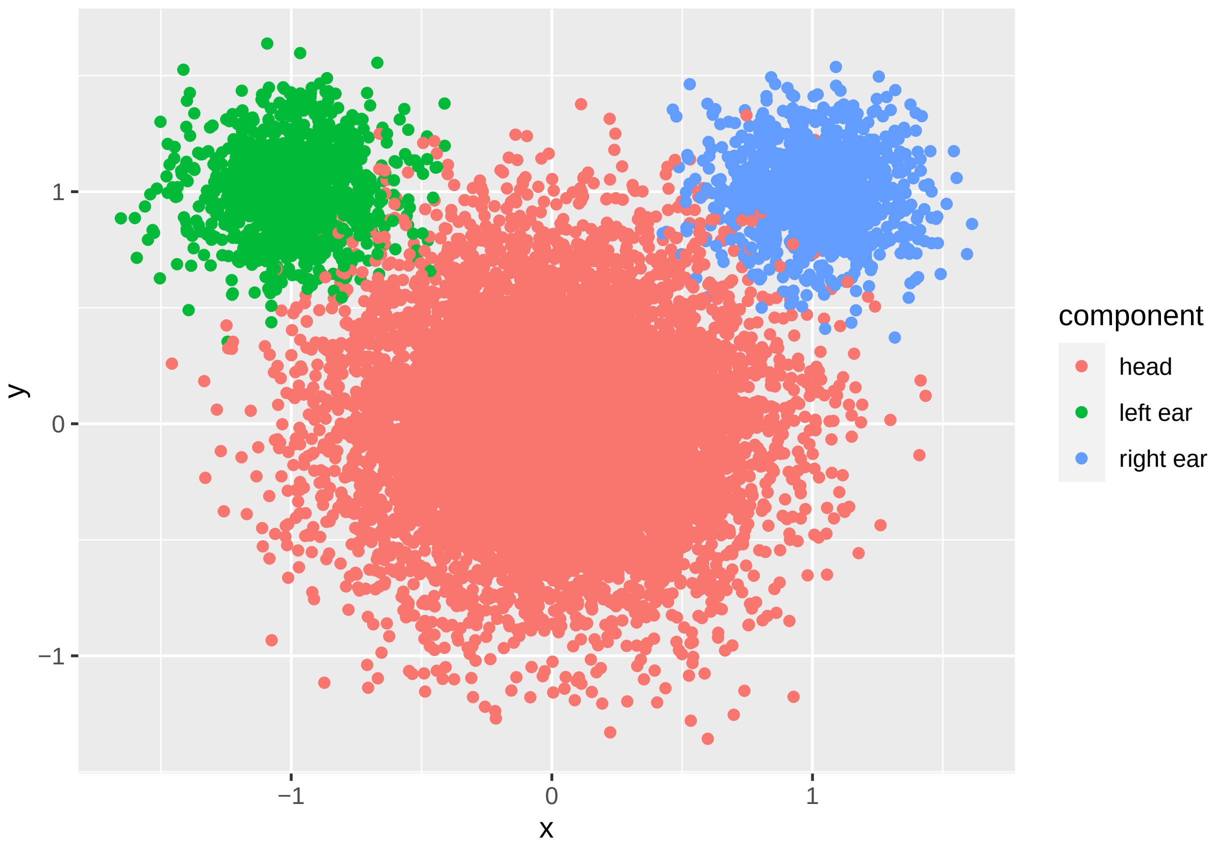
Your tasks:

- Write a function `mouse(N, sds, prop)` to simulate observations from the Mouse distribution. In the function call, `N` is the number of observations, `sds` is a vector of standard deviations (length 2), and `prop` is the probability of the head. The function should return a data frame with coordinates and the component where they come from (head/ear1/ear2).
- Simulate 10 000 observations with chosen parameters and plot them on a 2-dimentional scatter plot.

```
mouse.data <- mouse(10000, c(0.4, 0.2), 0.8)
head(mouse.data)
```

```
##           x           y component
## 1 0.2566234 -0.59686447      head
## 2 0.4235462 -0.21667008      head
## 3 0.5681747 -0.06542812      head
## 4 0.1401701  0.56765871      head
## 5 1.4786279  0.89244834 right ear
## 6 0.7122386 -0.11748176      head
```

```
ggplot(data=mouse.data) + geom_point(aes(x=x, y=y, col=component))
```



- Cluster the mouse using the k-means algorithm (using the `kmeans` function). Do the clusters agree with your expectations and why not? Why does the k-means algorithm work so poorly in this example?

The mouse is an artificial dataset, but it illustrates a very common problem. The k-means algorihtm always tends to return equally sized clusters, but such clusters are very rare in reality. This sometimes leads to very wrong conclusions.

The EM algorithm for Gaussian Mixtures

The EM algorithm is a basis for a clustering method which can handle difficult distributions like The Mouse. We will apply EM to cluster The Mouse and the Iris dataset. First, install the EMCluster package and load it.

```
install.packages("EMCluster")
```

```
library(EMCluster)
```

```
## Loading required package: MASS

## Loading required package: Matrix
```

The main function is `init.EM()`. It accepts the data in a matrix format and the number of desired clusters. However, for educational purposes, we will use a more basic function: `emcluster()`. Instead of the number of clusters, it accepts a list of initial parameters: mixing proporions, centers of clusters, and a list of covariance matrices.

Your tasks:

Estimate the initial parameters from the results of the k-means clustering.

- Estimate the proportions of observations belonging to different clusters from k-means clustering of The Mouse.
- Use the function `mean()` to estimate the centers of the clusters. Return your results in a 3x2 matrix.
- Use the function `cov(m)` to estimate the covariance matrix from data in matrix `m`. Estimate covariance matrices in groups of clusters. Next, take the lower traingular part of the matrix using the function `lower.tri(m, diag=TRUE)` and write it as a vector. Stack those vectors row by row into a 3x3 matrix. You can use a `for` loop in this task. Ask me if you have no idea how to do this point.
  - The function `lower.tri` returns a boolean matrix. Writing `m[lower.tri(m, diag=TRUE)` will yield a vector.

After we get our estimates, we can run the EM clustering by this command:

```
mouse.EM <- emcluster(mouse.data[,1:2], pi=cluster.props, Mu=cluster.means, LTSigma=cluster.covs)
```

A simpler way to do just the above would be typing `init.EM(mouse.data[,1:2], 3)`. However, it would take considerably longer.

We can now inspect the results. For this specific fun we had. The proportions:

```
mouse.EM$pi
```

```
## [1] 0.10132919 0.79880962 0.09986119
```

Close enough to $(0.1, 0.1, 0.8)$, the true values. The cluster centers:

```
mouse.EM$Mu
```

```
##              [,1]         [,2]
## [1,] -0.998994263  1.003360057
## [2,]  0.002495695 -0.006865098
## [3,]  1.002854438  1.001960219
```

Also close to $(1, 1), (-1, 1), (0, 0)$, the true values. The covariances:

```
mouse.EM$LTSigma
```

```
##            [,1]          [,2]       [,3]
## [1,] 0.03954465  0.0019594326 0.03947446
## [2,] 0.16251758 -0.0021164036 0.15849885
## [3,] 0.04354087 -0.0008848759 0.03603515
```

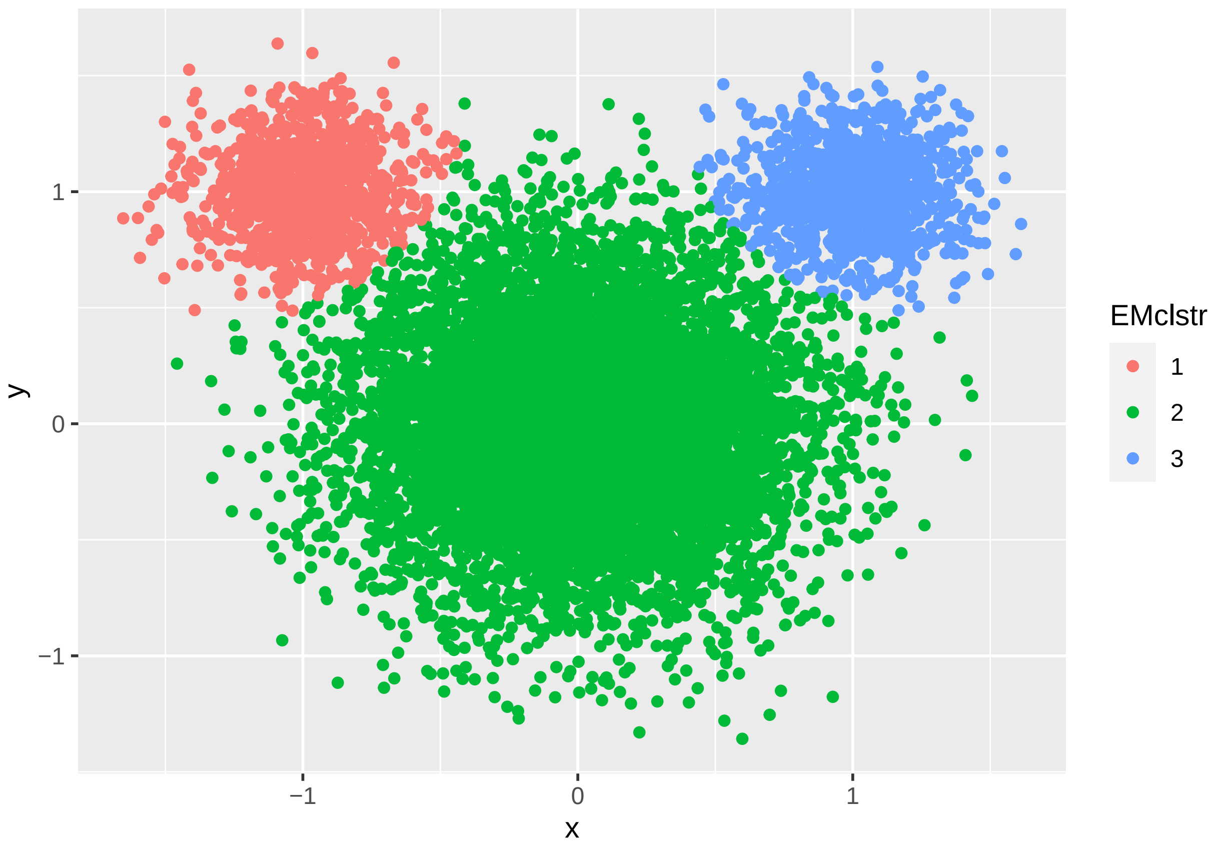
That's also close to true variances: 0.16 variance for the head and 0.04 for the ears at x and y coordinates, and zero covariances.

We can now assign the observations to clusters, by picking the cluster with the highest likelihood for each observation. Ask me if you're not sure what that means. The function `assign.class` will the assignment for us. It accepts the data matrix and the result of EM clustering.

```
mouse.data$EMclstr <- assign.class(mouse.data[,1:2], emobj=mouse.EM)$class
```

Finally, we can plot the results of the EM clustering and compare it with K-means. Before plotting, convert the EMcluster column to a factor.

```
mouse.data$EMclstr <- as.factor(mouse.data$EMclstr)
ggplot(data=mouse.data) + geom_point(aes(x=x, y=y, col=EMclstr))
```



Much better!