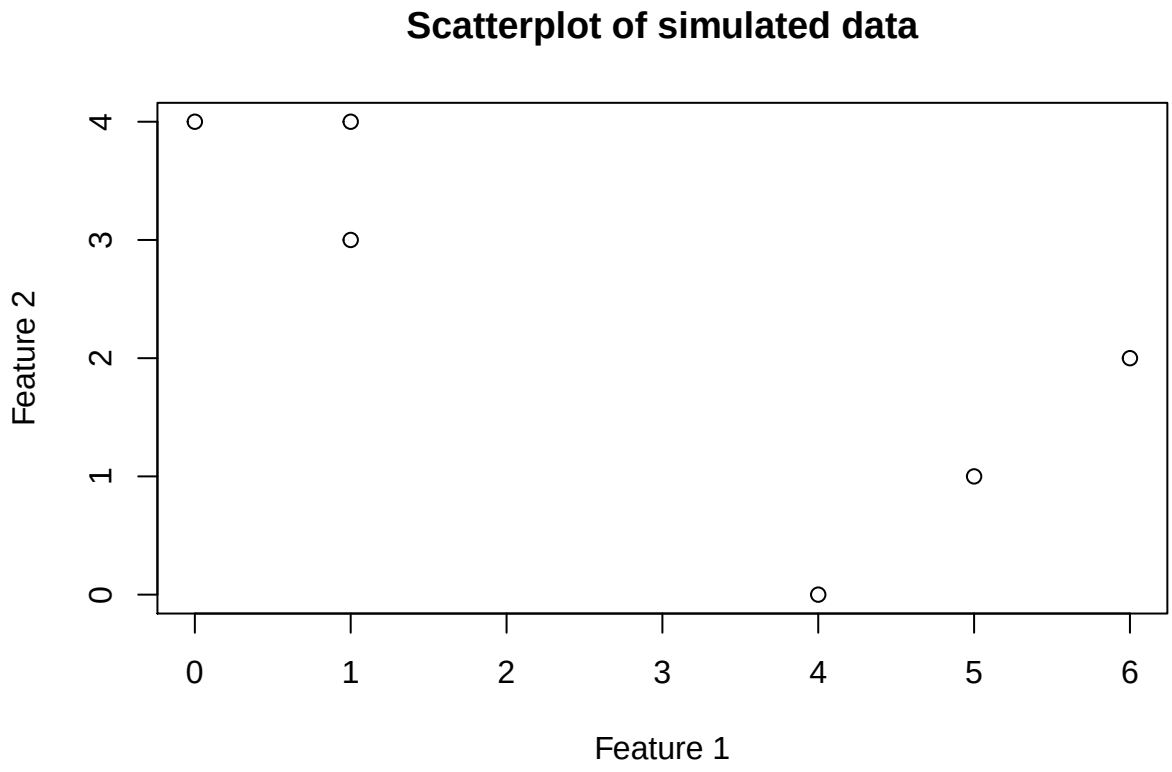
# Problem Set 4

*Audrey Glaser*

*3/2/2020*

**1. (5 points) Plot the observations.**

```
x <- cbind(c(1, 1, 0, 5, 6, 4), c(4, 3, 4, 1, 2, 0))

plot(x[,1], x[,2],
     main="Scatterplot of simulated data",
     xlab = "Feature 1",
     ylab = "Feature 2")
```

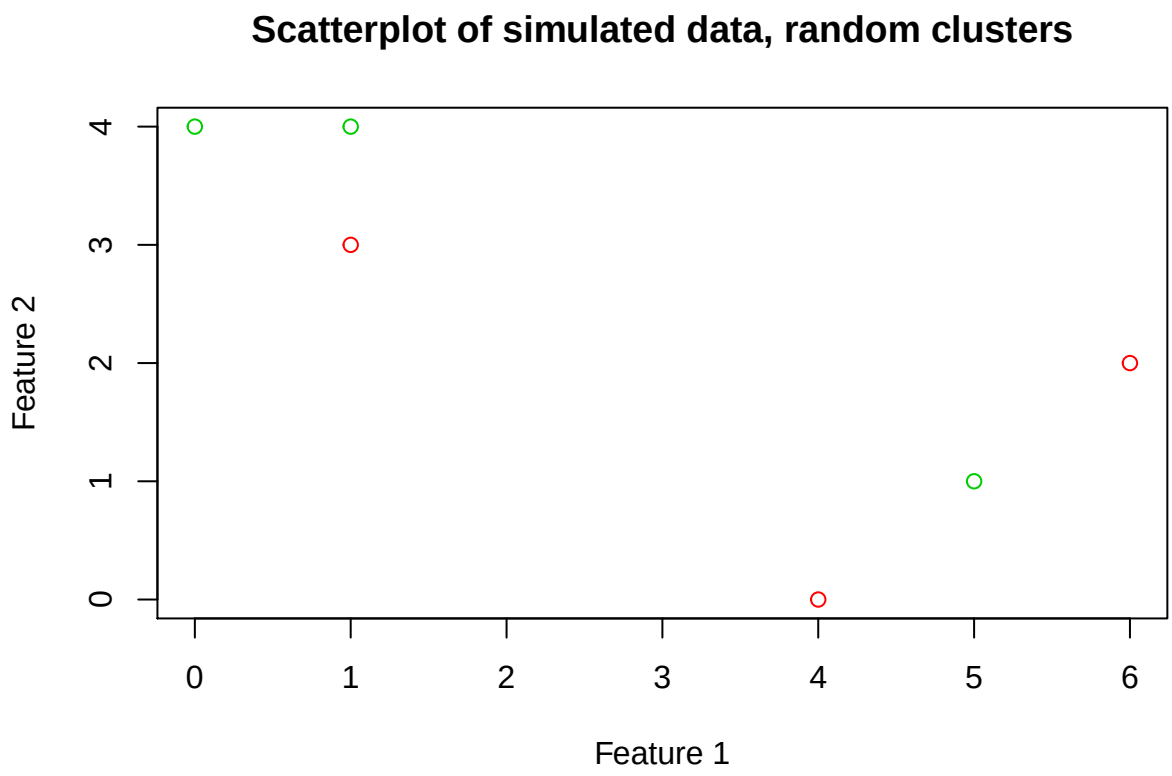



**2. (5 points) Randomly assign a cluster label to each observation. Report the cluster labels for each observation and plot the results with a different color for each cluster (remember to set your seed first).**

```
set.seed(100)
cluster_label <- sample(2, nrow(x), replace = T)
cluster_label
```

```
## [1] 2 1 2 2 1 1
```

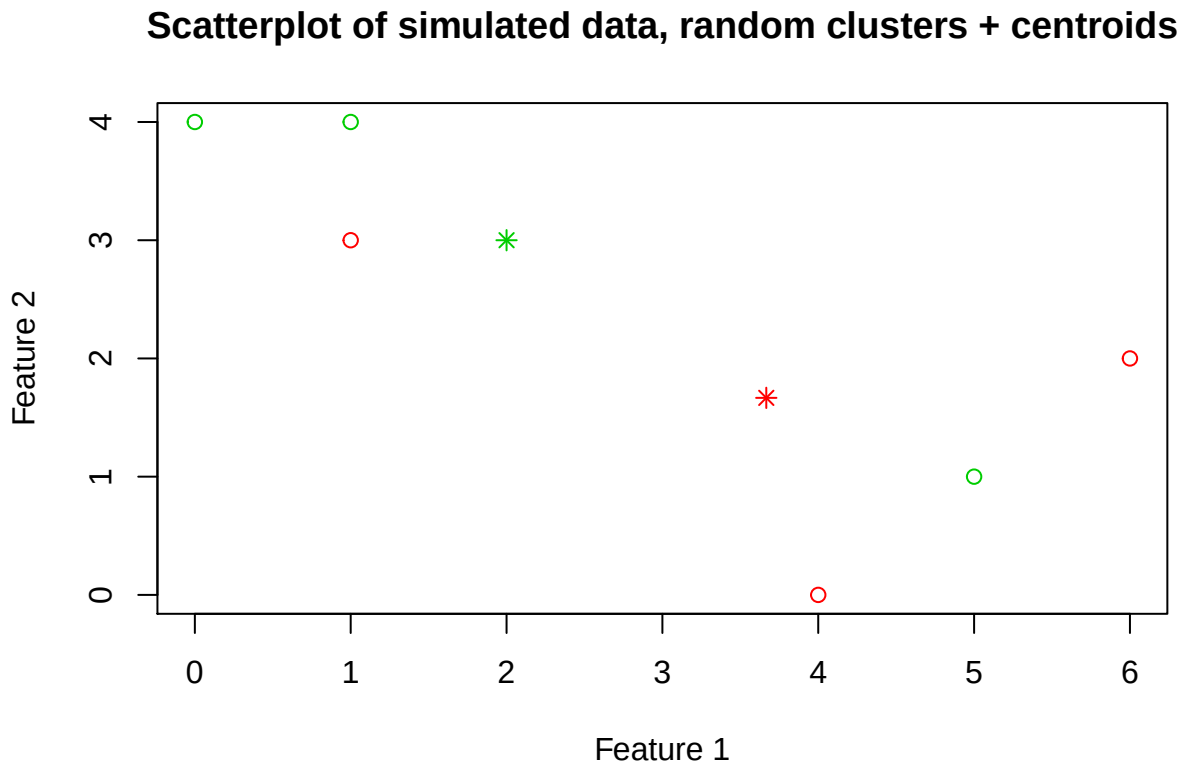
```
plot(x[,1], x[,2],
     main="Scatterplot of simulated data, random clusters",
     xlab = "Feature 1",
     ylab = "Feature 2",
     col=(cluster_label + 1))
```




**3. (10 points) Compute the centroid for each cluster.**

```
centroid1 <- c(mean(x[cluster_label == 1, 1]),
               mean(x[cluster_label == 1, 2]))
centroid2 <- c(mean(x[cluster_label == 2, 1]),
               mean(x[cluster_label == 2, 2]))

# different colors for each feature
plot(x[,1], x[,2],
     main="Scatterplot of simulated data, random clusters + centroids",
     xlab = "Feature 1",
     ylab = "Feature 2",
     col=(cluster_label + 1)) +
points(centroid1[1], centroid1[2],
       col = 2, pch = 8) +
```

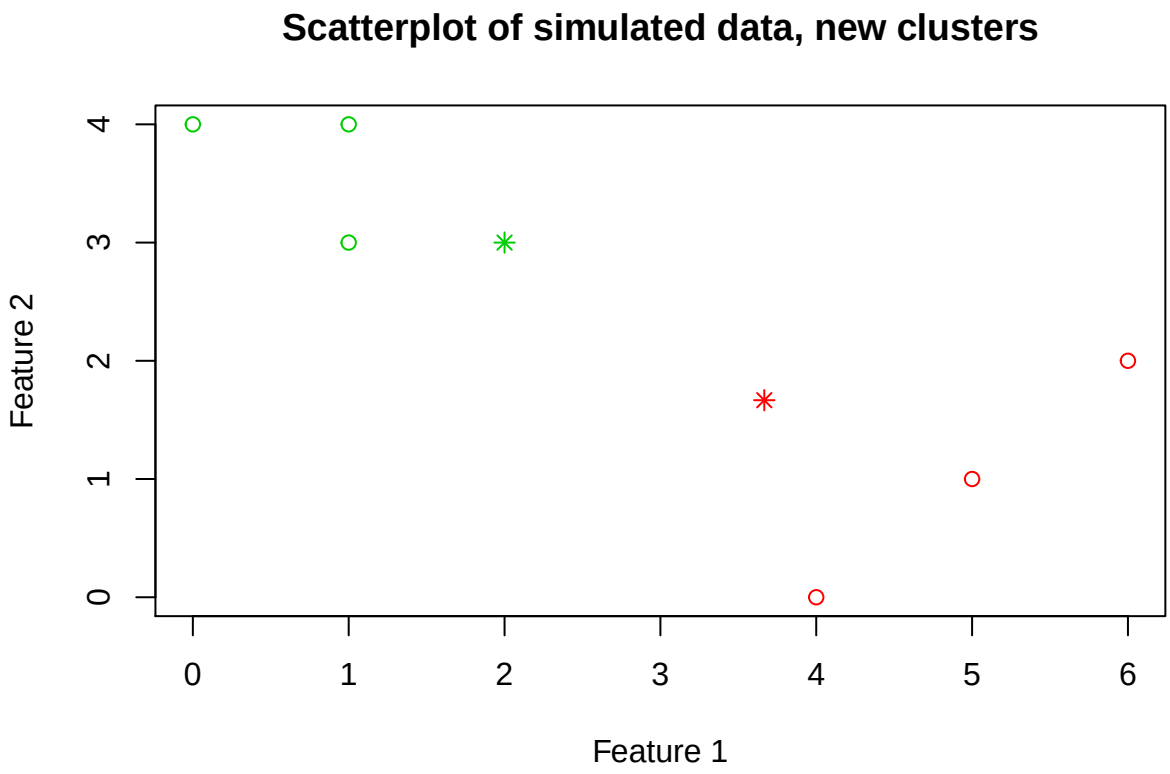
```
points(centroid2[1], centroid2[2],
       col = 3, pch = 8)
```




```
## integer(0)
```

**4. (10 points) Assign each observation to the centroid to which it is closest, in terms of Euclidean distance. Report the cluster labels for each observation.**

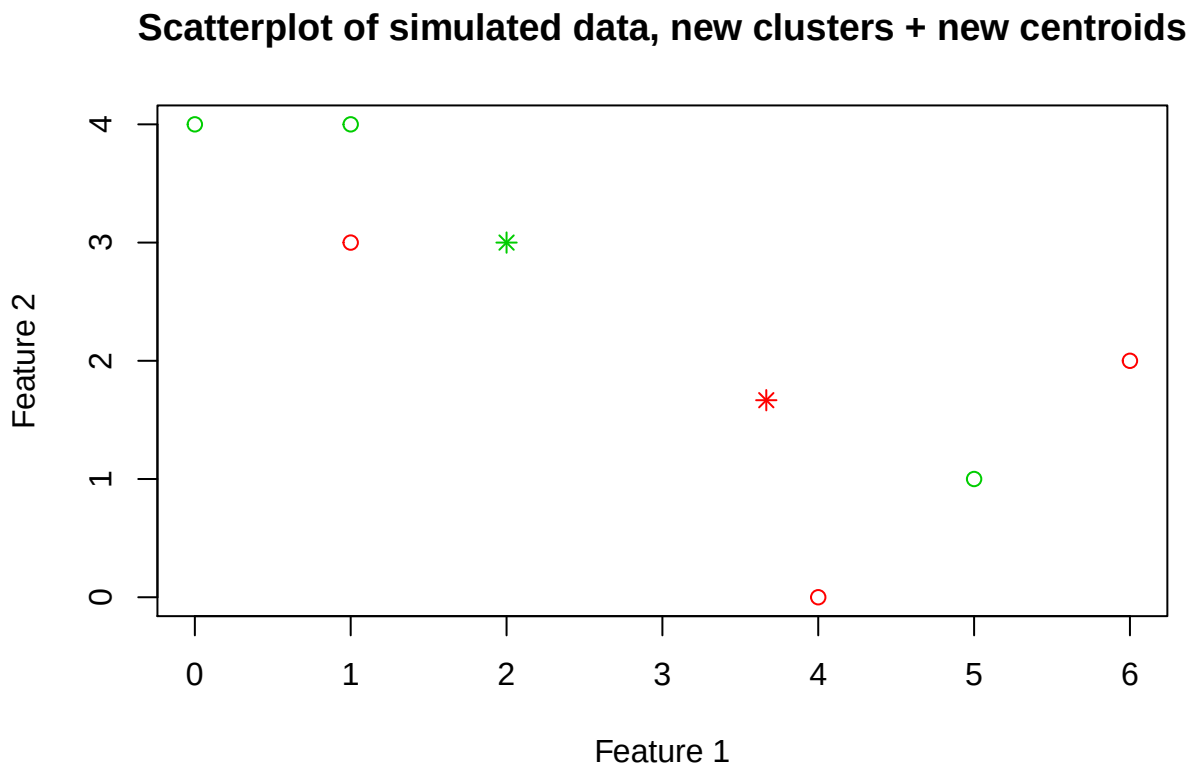
```
# Re-assign labels to observations
labels <- c(2, 2, 2, 1, 1, 1)

# Plot relabeled observations
plot(x[, 1], x[, 2],
     main="Scatterplot of simulated data, new clusters",
     xlab = "Feature 1",
     ylab = "Feature 2",
     col = (labels + 1)) +
points(centroid1[1], centroid1[2],
       col = 2, pch = 8) +
points(centroid2[1], centroid2[2],
       col = 3, pch = 8)
```




```
## integer(0)
```

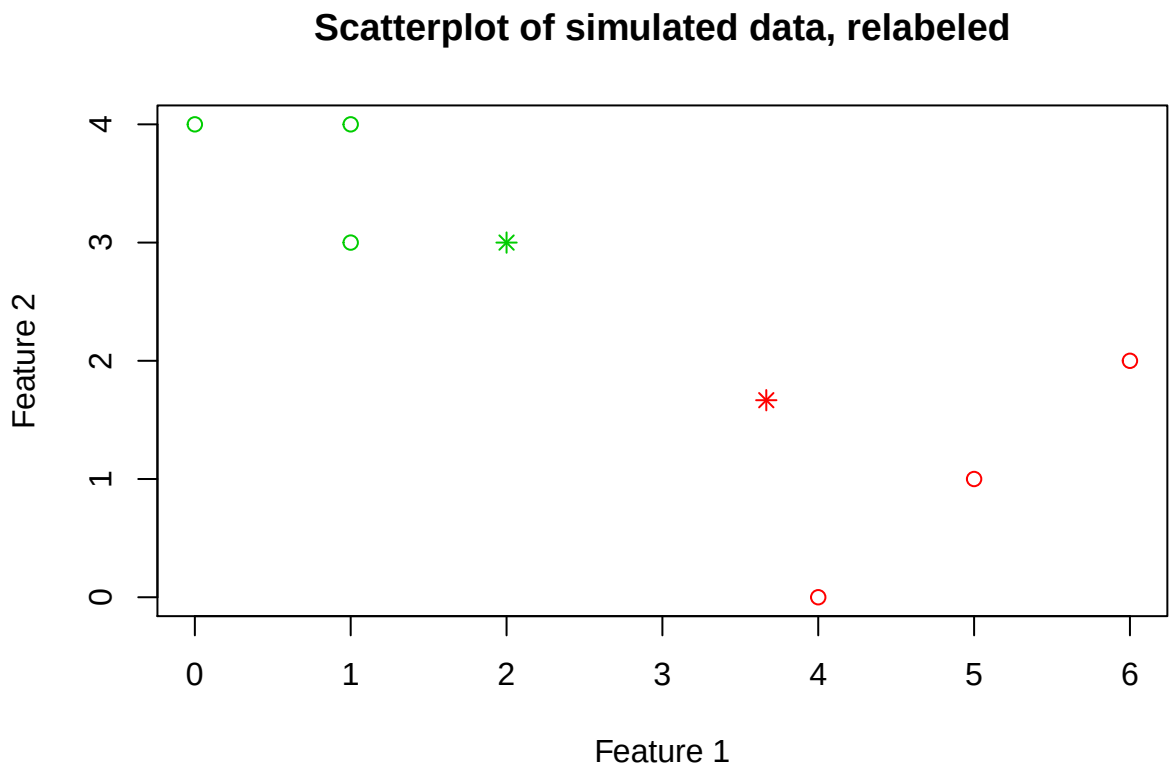
**5. (5 points) Repeat (3) and (4) until the answers/clusters stop changing.**

```
# Compute new centoids
centroid1 <- c(mean(x[cluster_label == 1, 1]),
               mean(x[cluster_label == 1, 2]))
centroid2 <- c(mean(x[cluster_label == 2, 1]),
               mean(x[cluster_label == 2, 2]))

# Plot new centroids
plot(x[,1], x[,2],
     main="Scatterplot of simulated data, new clusters + new centroids",
     xlab = "Feature 1",
     ylab = "Feature 2",
     col=(cluster_label + 1)) +
points(centroid1[1], centroid1[2],
       col = 2, pch = 8) +
points(centroid2[1], centroid2[2],
       col = 3, pch = 8)
```




```
## integer(0)
```

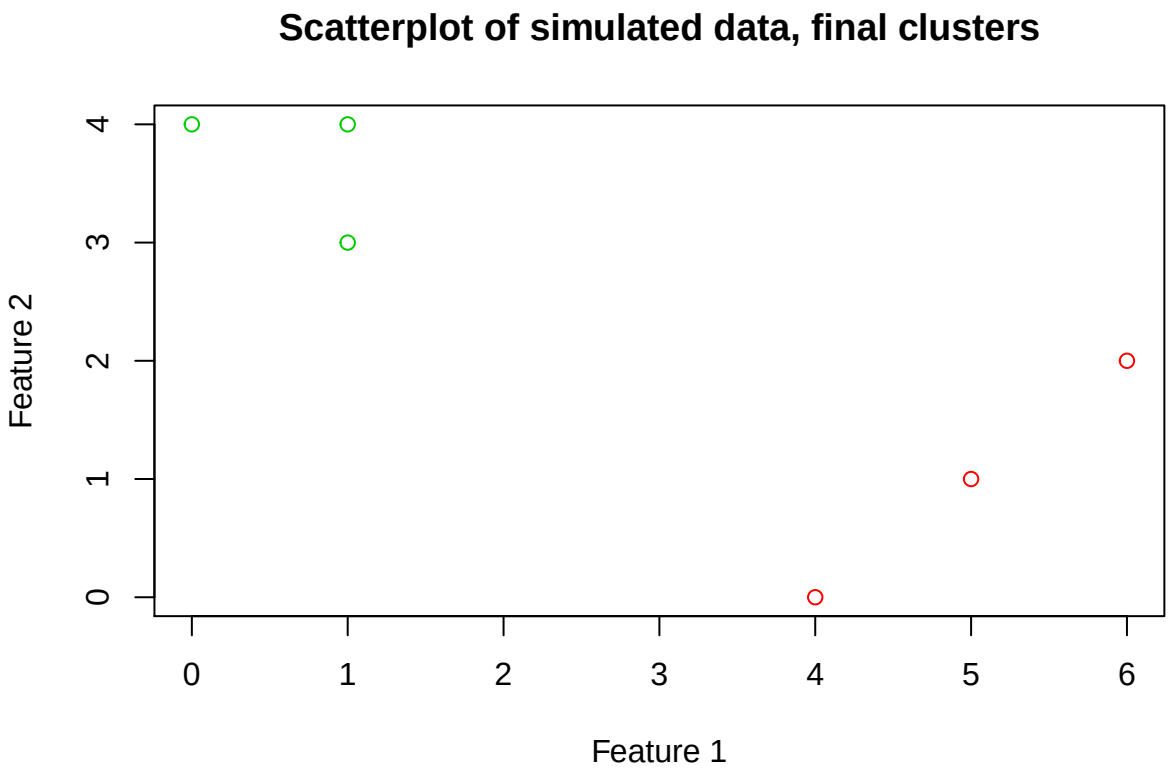
```
# Relabel observations based on new centroids
labels <- c(2, 2, 2, 1, 1, 1)

# Plot relabeled observations
plot(x[, 1], x[, 2],
     main="Scatterplot of simulated data, relabeled",
     xlab = "Feature 1",
     ylab = "Feature 2",
     col = (labels + 1)) +
points(centroid1[1], centroid1[2],
       col = 2, pch = 8) +
points(centroid2[1], centroid2[2],
       col = 3, pch = 8)
```




```
## integer(0)
```

**6. (10 points) Reproduce the original plot from (1), but this time color the observations according to the clusters labels you obtained by iterating the cluster centroid calculation and assignments.**

```
plot(x[, 1], x[, 2],
     main="Scatterplot of simulated data, final clusters",
     xlab = "Feature 1",
     ylab = "Feature 2",
     col = (labels + 1))
```




## Clustering State Legislative Professionalism

**1. Load the state legislative professionalism data. See the codebook (or above) for further reference.**

**2. (5 points) Munge the data:**

**a. Select only the continuous features that should capture a state legislature's level of "professionalism" (session length (total and regular), salary, and expenditures);**

**b. Restrict the data to only include the 2009/10 legislative session for consistency;**

**c. Omit all missing values;**

**d. Standardize the input features;**

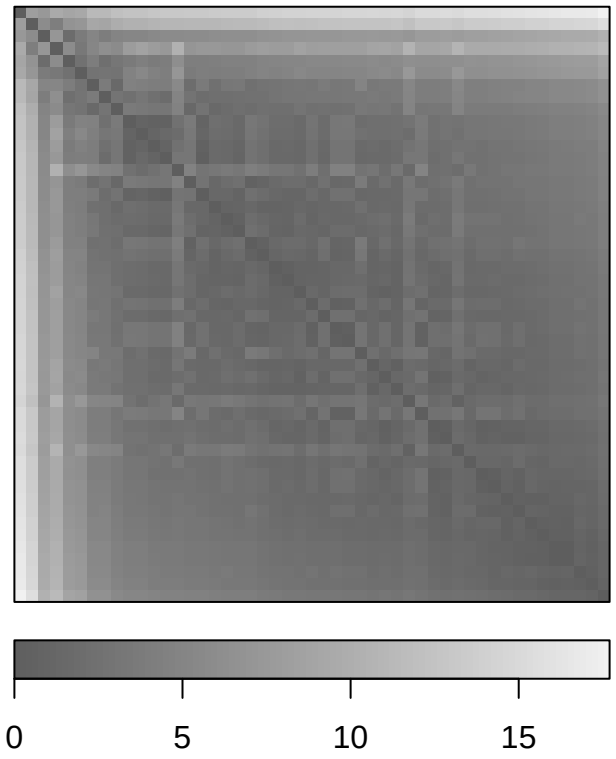
**e. Do anything else you think necessary to get this subset of data into workable form (hint: consider storing the state names as a separate object to be used in plotting later)**

```
legprof_sub <- legprof %>%
  filter(sessid == "2009/10") %>%
  select(state, t_slength, slength, salary_real, expend)
```

```
legprof_states <- scale(legprof_sub[, -c(1)]) %>% #standardizing, making them unit-less
  as_tibble() %>%
  na.omit()

states <- legprof_sub %>%
  na.omit() %>%
  select(state)
```

**3. (5 points) Diagnose clusterability in any way you'd prefer (e.g., sparse sampling, ODI, etc.); display the results and discuss the likelihood that natural, non-random structure exist in these data. Hint: We didn't cover how to do this R in class, but consider dissplot() from the seriation package, the factoextra package, and others for calculating, presenting, and exploring the clusterability of some feature space.**

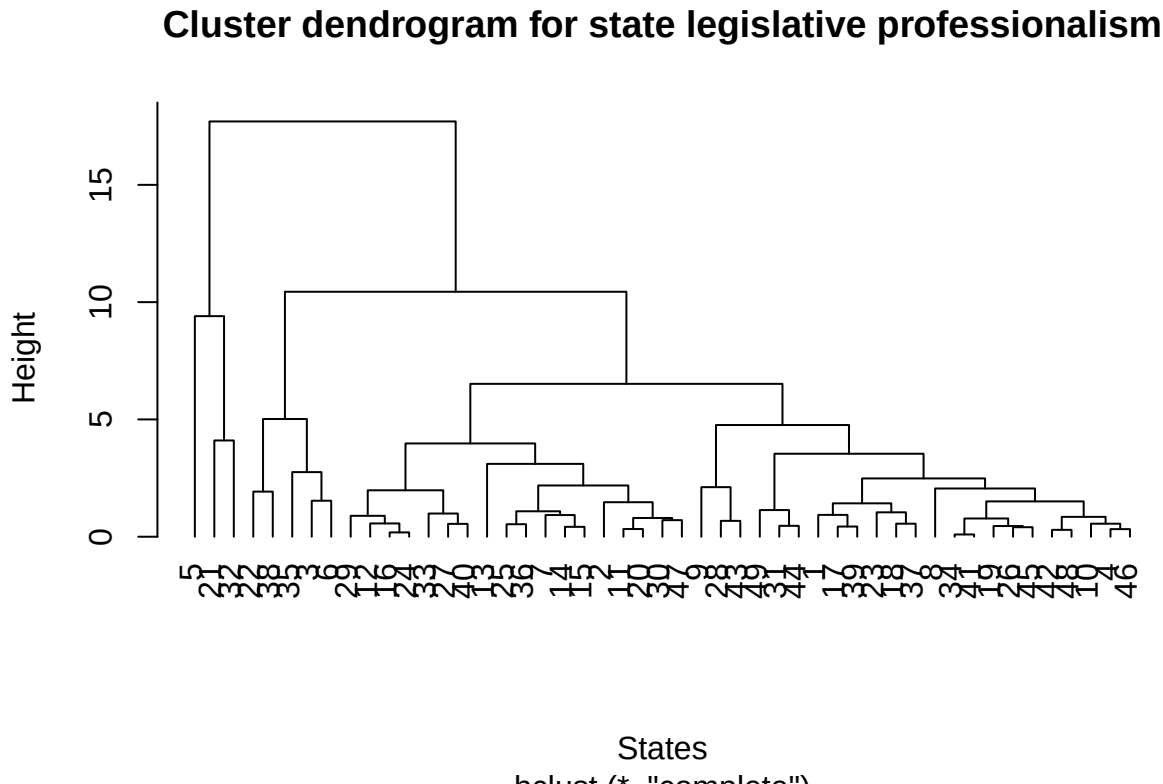
```
legprof_dist <- dist(legprof_states, method = "manhattan")
dissplot(legprof_dist)
```




The dissimilarity matrix heat map shows low contrast throughout, suggesting low likelihood of clusterability in the data. If natural, non-random structure existed in the data, we'd expect to see defined regions of contrast in the dissimilariy matrix heat map.

**4. (5 points) Fit an agglomerative hierarchical clustering algorithm using any linkage method you prefer, to these data and present the results. Give a quick, high level summary of the output and general patterns.**

```
hc_complete <- hclust(legprof_dist,
                      method = "complete");
plot(hc_complete, hang = -1,
     main = "Cluster dendrogram for state legislative professionalism",
     xlab = "States",
     sub = NULL)
```




This agglomerative hierarchical clustering method defines the cluster distance between two clusters to be the maximum distance between their individual components. Therefore, in this cluster diagram where each cluster represents a state, we can interpret the branch distances as indicative of similarity. For example, states 4 and 46 are separate by only one short branch, indicating similarity between the states. By contrast, the branch connecting states 3 and 46 is longer, and the states are interpreted to be relatively dissimilar.

**5. (5 points) Fit a k-means algorithm to these data and present the results. Give a quick, high level summary of the output and general patterns. Initialize the algorithm at k=2, and then check this assumption in the validation questions below.**

```r
set.seed(1234)

kmeans <- kmeans(legprof_states,
                 centers = 2,
                 nstart = 15)

kmeans_cluster <- as.data.frame(kmeans$cluster)
kmeans_state <- states

kmeans_df <- cbind(kmeans_cluster, kmeans_state)
```

When we fit a k-means clustering algorithm to the dataset with the parameter k set to 2, 43 observations, or states, end up in one cluster, and 6 stated end up in the other. The within-cluster sum of squared deviations accounts for nearly half of total error, so the clusters are not very compact. As for a substantial interpetation of the clustering, the smaller cluster includes California, Massachusetts, Michigan, New York, Ohio, and Pennsylvania, which vary in population size and wealth but are relatively politically diverse states. Thus, political heterogeneity might be a relevant factor to understanding levels of state legislative professionalism in the future, but it is not measured in our current dataset.

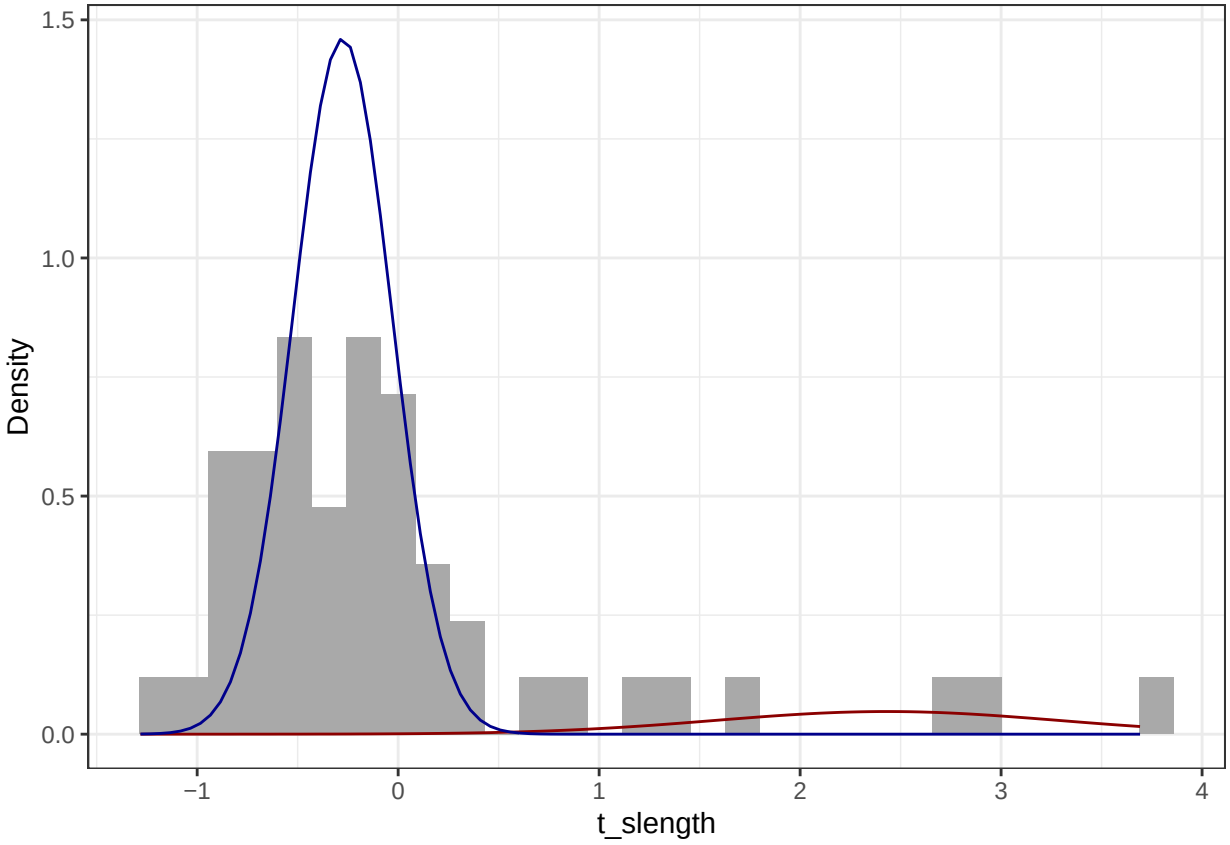
## 6. (5 points) Fit a Gaussian mixture model via the EM algorithm to these data and present the results. Give a quick, high level summary of the output and general patterns. Initialize the algorithm at k = 2, and then check this assumption in the validation questions below.

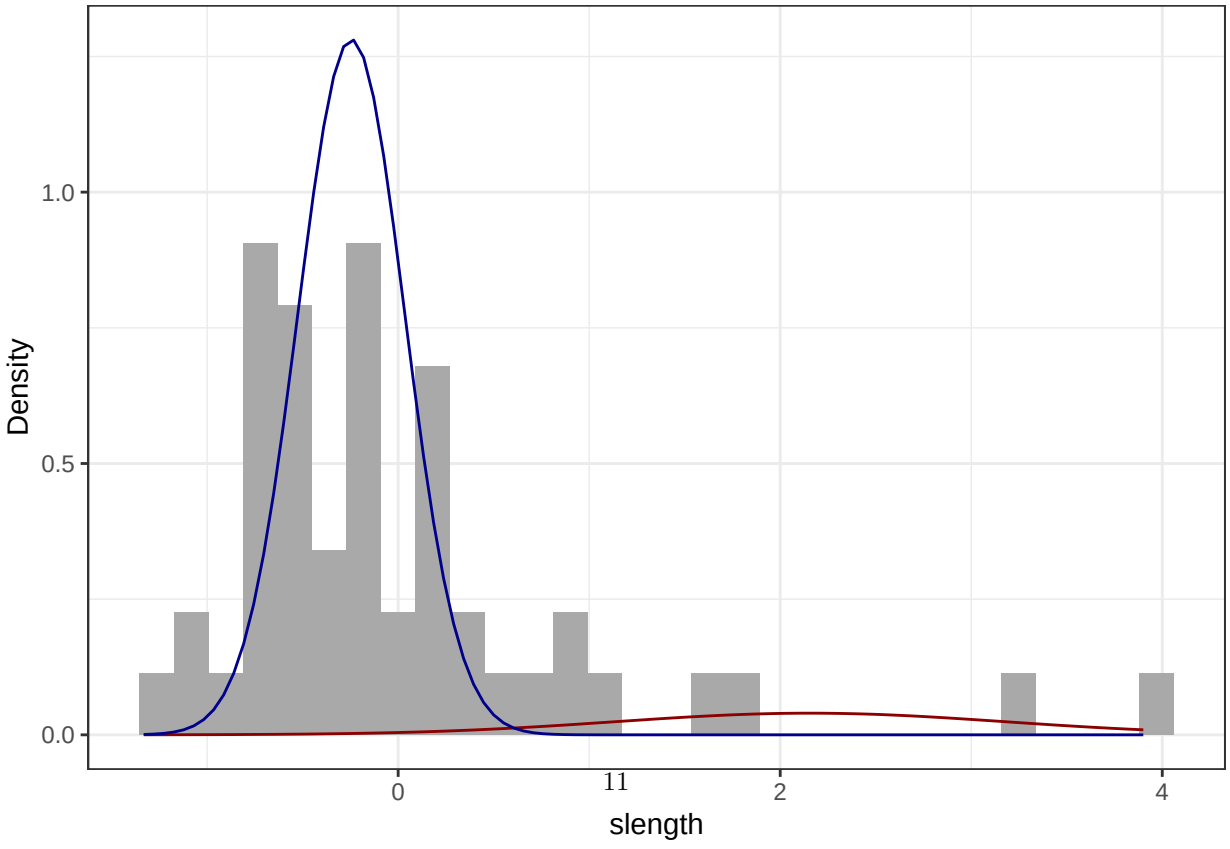
```r
set.seed(5678)

gmm <- mvnormalmixEM(legprof_states, k = 2)
```

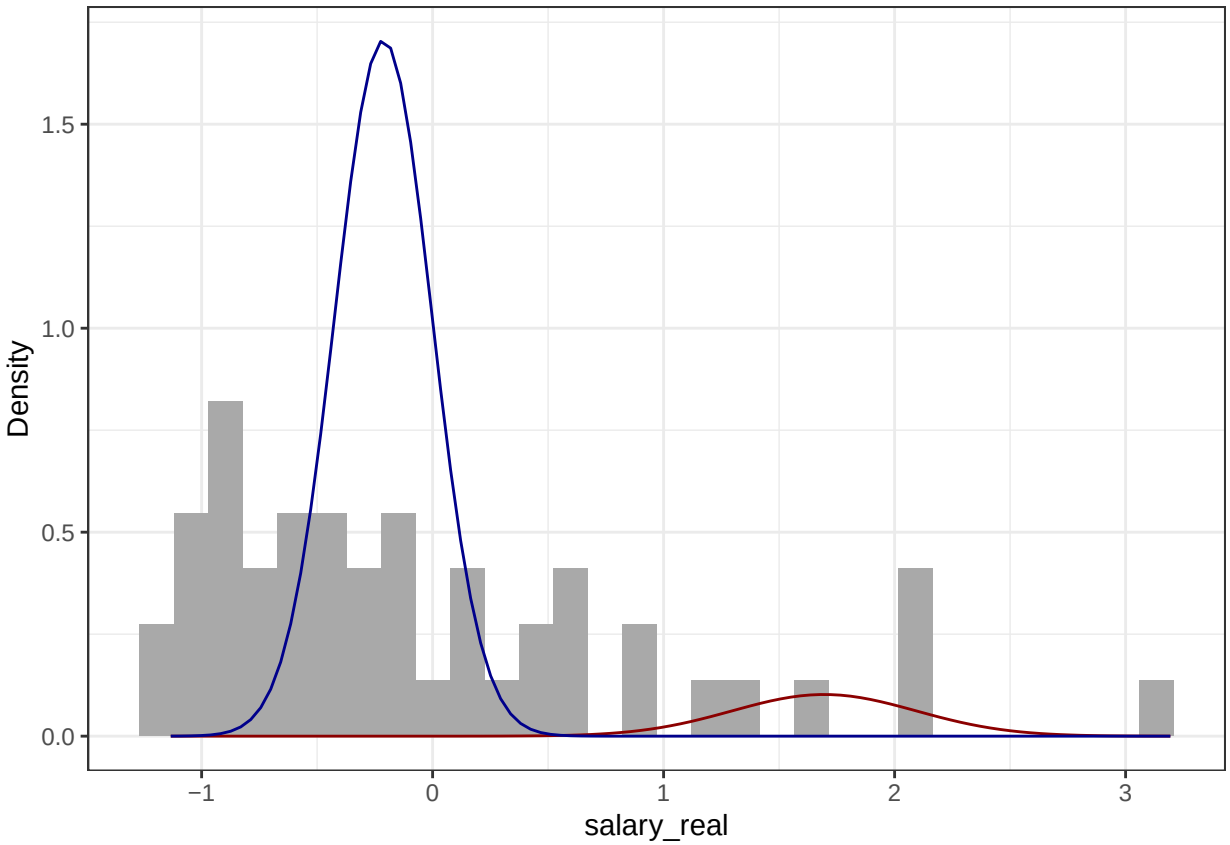
```
## number of iterations= 12
```

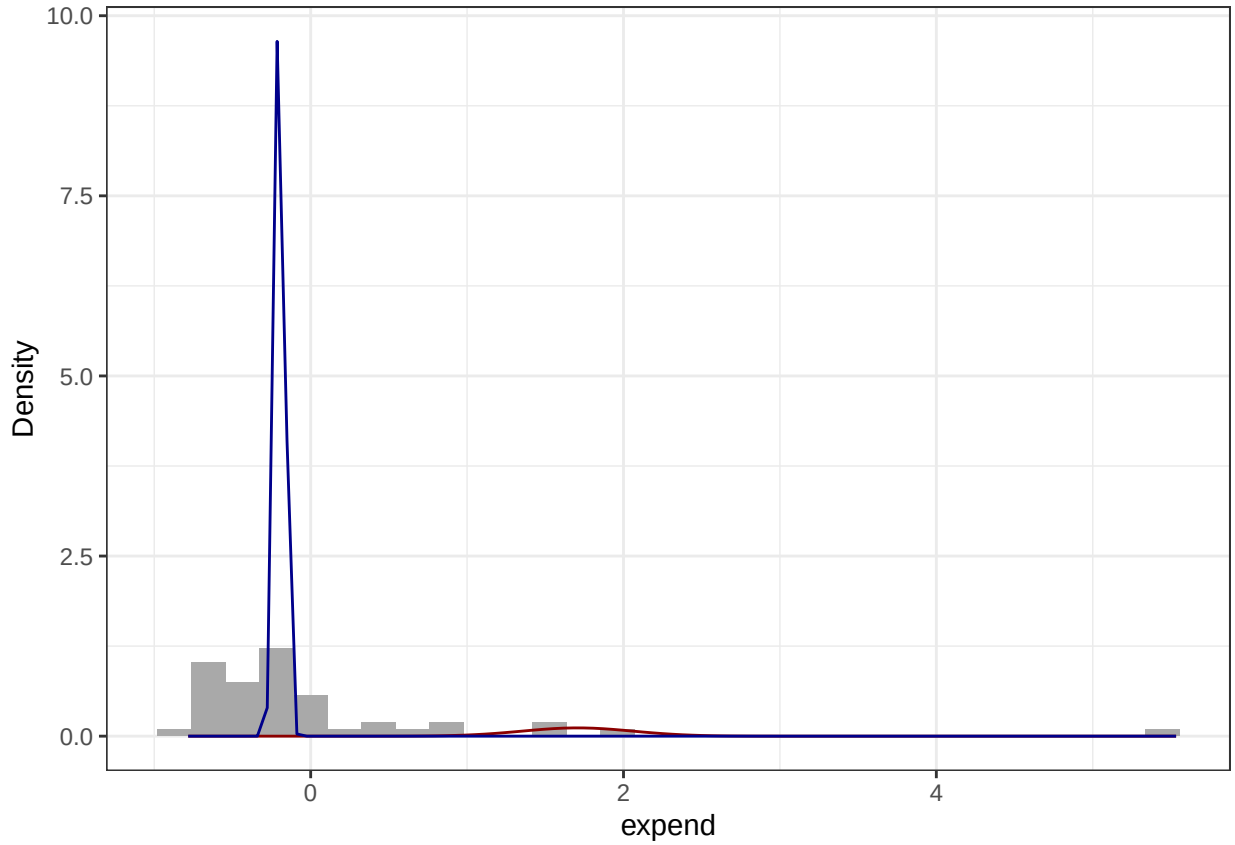
When we apply the Gaussian mixture model to our data and view the output, there are two lambda values or mixing coefficients, 0.102 and 0.898. The larger lambda will yield a higher curve than the smaller lambda, and the corresponding cluster has more observations than the other cluster. Additionally, we have 2 mu vectors of length 4, corresponding to the four input features. One of the clusters has lower mean values across all features, while the other has a higher mean for all features. Finally, the sigma values tell us that one of the clusters has less variance in its distributions of individual features, while the other has more variance.

**7. (15 points) Compare output of all in visually useful, simple ways (e.g., present the dendrogram, plot by state cluster assignment across two features like salary and expenditures, etc.). There should be several plots of comparison and output.**




Density
1.5
1.0
0.5
0.0
−1
0
1
2
3
salary_real




When the plotted curves fit the histogram of an individual feature well, it suggests that the feature is important in explaining the clustering model. In our multivariate clustering model, it appears that our observations primarily cluster together based on similarity in features 's_length' and 't_slength', with some variance, shown in the plots by the histogram bars falling outside the curves. The curve for the feature 'salary_real' fits somewhat well, suggesting 'salary-real' partially explains the clustering. However, the curve does not fit the data for the feature 'expend' well, suggeting this feature is weakly influencing the clustering.

**8. (5 points) Select a single validation strategy (e.g., compactness via min(WSS), average silhouette width, etc.), and calculate for all three algorithms. Display and compare your results for all three algorithms you fit (hierarchical, k-means, GMM). Hint: Here again, we didn't cover this in R in class, but think about using the clValid package, though there are many other packages and ways to validate cluster patterns across iterations.**

```
legprof_states_mat <- as.matrix(legprof_states)
clvalid <- clValid(legprof_states_mat, 2:20,
                           validation = "internal",
                           clMethods = c("model", "kmeans", "hierarchical"))
```

```
## Warning in clValid(legprof_states_mat, 2:20, validation = "internal",
## clMethods = c("model", : rownames for data not specified, using
## 1:nrow(data)
```

Looking at the validity measure of connectivity only, for which lower values indicate higher validity, we see that a hierarchical clustering algorithm is the optimal method for this dataset, because it results in lower connectivity measure values for any given number of clusters than k-means and GMM. Furthermore, out of the possible k values for the hierarchical clustering model, the optimal k value is 2.

## 9. (10 points) Discuss the validation output, e.g., "What can you take away from the fit?", "Which approach is optimal? And optimal at what value of k?", "What are reasons you could imagine selecting a technically "sub-optimal" clustering method, regardless of the validation statistics?"

We compared three clustering methods: GMM, kmeans, and hierarchical (HAC). Our chosen validity measure tells us that the hierarchical clustering algorithm is the optimal method. Furthermore, it tells us that this method is most optimal when k = 2, or there are two clusters. However, one might select a sub-optimal clustering method if that particular method is more useful for their particular problem. For example, GMM gives you more information such as likelihood, or probability, of belonging to one of the clusters than k-means. GMM might also be less computationally expensive than other models. Thus one might select GMM in certain cases, even if it is sub-optimal according to the validation statistics.