Problem Set 4

Eric Karsten

March 2, 2020

1 K Means By Hand

Do k-means by hand to understand it.

1.1 Plot the observations (5 points)

1.1.1 Solution

```
library(tidyverse)

set.seed(3)

df <- tibble(
  a = c(1, 1, 0, 5, 6, 4),
  b = c(4, 3, 4, 1, 2, 0))

k <- 2
```

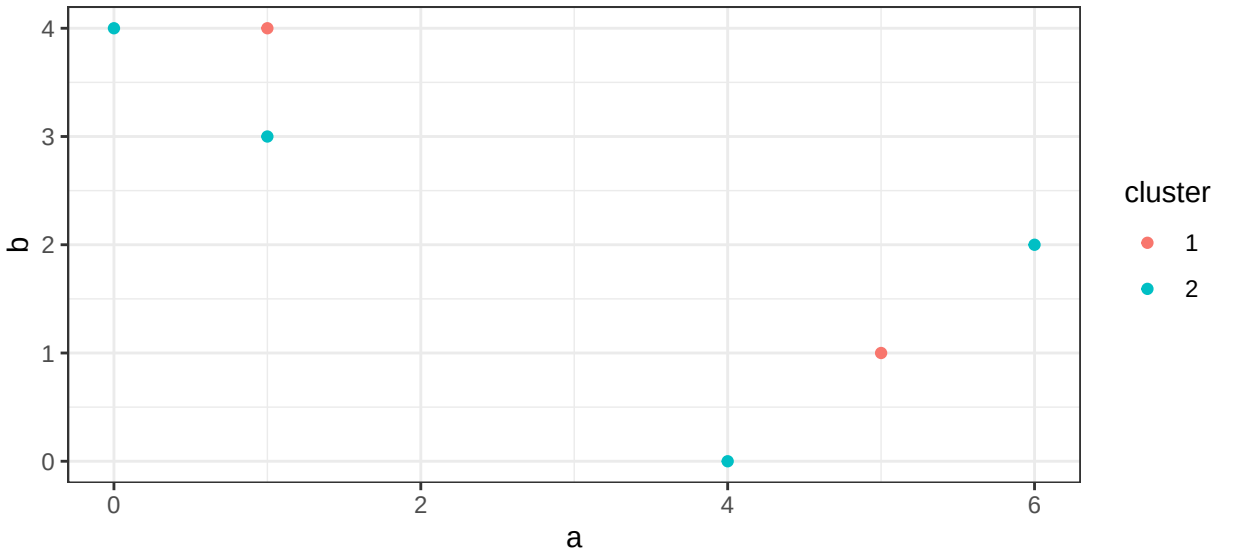
1.2 Random Clusters (5 points)

1.2.1 Problem Statement

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).

1.2.2 Solution

```
df <- df %>%
  mutate(cluster = as.factor(sample(1:k, length(a), replace = T)))

df %>%
  ggplot(aes(x = a, y = b, color = cluster)) +
  geom_point() +
  theme_bw()
```



1.3 Compute Centroid (10 Points)

1.3.1 Problem Statement

Compute the centroid for each cluster.

1.3.2 Solution

Below is the centroid for each of the above clusters.

```
df %>%
  group_by(cluster) %>%
  summarise_all(~mean(.)) %>%
  knitr::kable()
```

cluster	a	b
1	3.00	2.50
2	2.75	2.25

1.4 Reassignment (10 points)

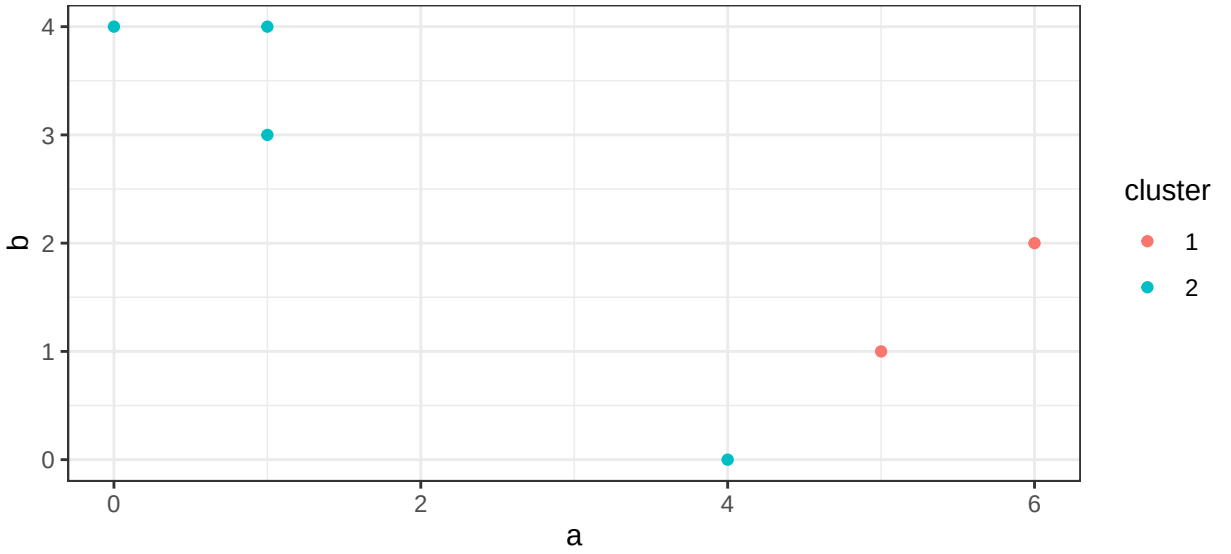
1.4.1 Problem Statement

Assign each observation to the centroid to which it is closest, in terms of Euclidean distance. Report the cluster labels for each observation.

1.4.2 Solution

```
new_cluster <- function(d) {
  c<- d %>%
  group_by(cluster) %>%
  summarise_all(~mean(.)) %>%
  pivot_wider(names_from = "cluster", values_from = c("a", "b"))

  e <- d %>%
```

```
    cbind(c) %>%
    mutate(dist_1 = (a - a_1)^2 + (b - b_1)^2,
           dist_2 = (a - a_2)^2 + (b - b_2)^2,
           cluster = as.factor(if_else(dist_1 < dist_2, 1, 2)))

  e %>% select(a, b, cluster) %>% return()

}

new_cluster(df) %>%
  ggplot(aes(x = a, y = b, color = cluster)) +
  geom_point() +
  theme_bw()
```




```
knitr::kable(new_cluster(df))
```

a	b	cluster
1	4	2
1	3	2
0	4	2
5	1	1
6	2	1
4	0	2

1.5 Iteration (5 points)

1.5.1 Problem Statement

Repeat 1.3 and 1.4 until the answers/clusters stop changing.

1.5.2 Solution

```
same = F
```

```
while(!same) {
  old_clusters <- df$cluster
  df <- new_cluster(df)
  same <- all(old_clusters == df$cluster)
}
```

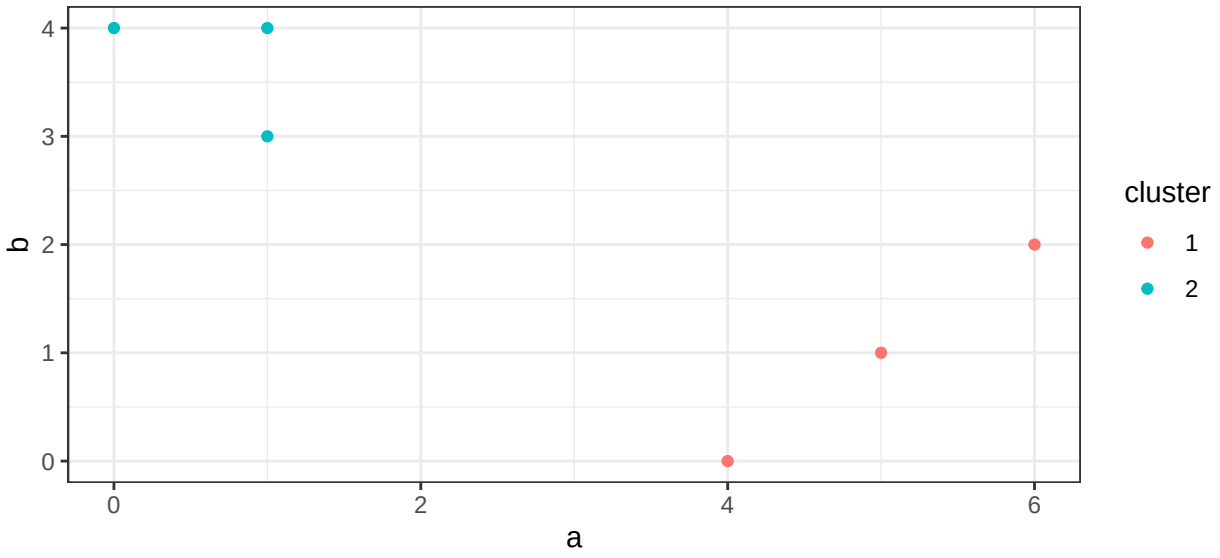
1.6 Results (10 points)

1.6.1 Problem Statement

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.

1.6.2 Solution

```
df %>%
  ggplot(aes(x = a, y = b, color = cluster)) +
  geom_point() +
  theme_bw()
```



2 Clustering State Legislative Professionalism

2.1 Data Setup

2.1.1 Problem Statement

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

2.1.2 Solution

```
load("Data and Codebook/legprof-components.v1.0.RData")

lpf <- as_tibble(x)
```

2.2 Data Cleaning (5 points)

2.2.1 Problem Statement

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. and 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)

2.2.2 Solution

```
# The features we want are t_slength, slength, expend
lpf_clean <-
  lpf %>%
  filter(sessid == "2009/10") %>%
  mutate(rowname = stateabv) %>%
  column_to_rownames() %>%
  select(t_slength, slength, salary_real, expend) %>%
  drop_na() %>%
  scale()
```

2.3 Diagnose Clusterability (5 points)

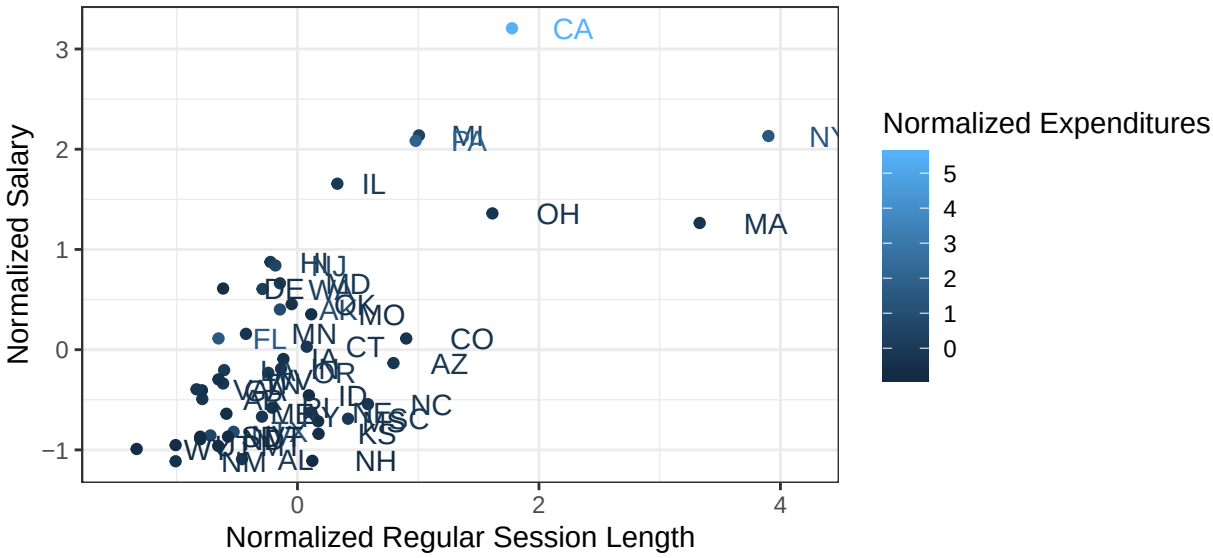
2.3.1 Problem Statement

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.

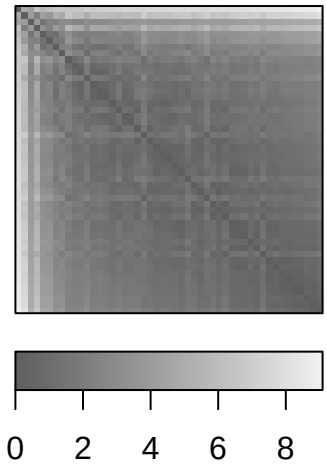
2.3.2 Solution

Based on a pricipal components analysis and a visaulization of the data, it seems like there is some meaningful and useful clustering that can be done along the lines of expenditures and session length.

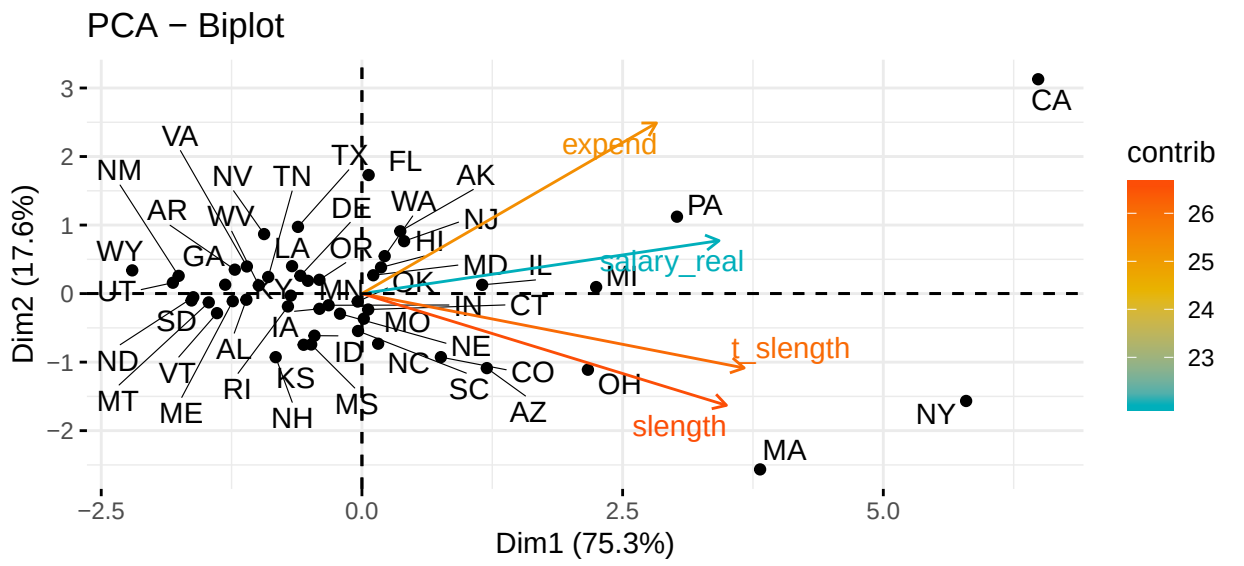
```
lpf_clean %>%
  as_tibble() %>%
  ggplot(aes(x = slength, y = salary_real, color = expend)) +
  geom_point() +
  geom_text(aes(label = rownames(lpf_clean)), hjust = -1) +
  labs(x = "Normalized Regular Session Length",
       y = "Normalized Salary",
       color = "Normalized Expenditures") +
  theme_bw() +
  coord_cartesian((xlim = c(-1.5, 4.2)))
```




```
seriation::dissplot(lpf_clean %>% dist())
```




```
my_pca <- FactoMineR::PCA(lpf_clean,  graph = FALSE)

factoextra::fviz_pca_biplot(my_pca, col.var="contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE,
             )
```



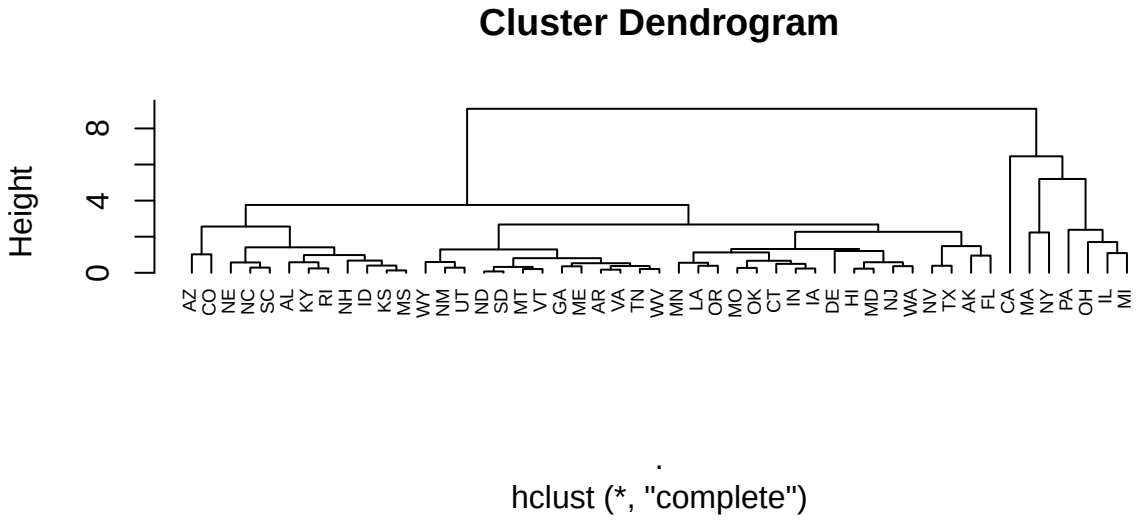
2.4 Hierarchical Clustering (5 points)

2.4.1 Problem Statement

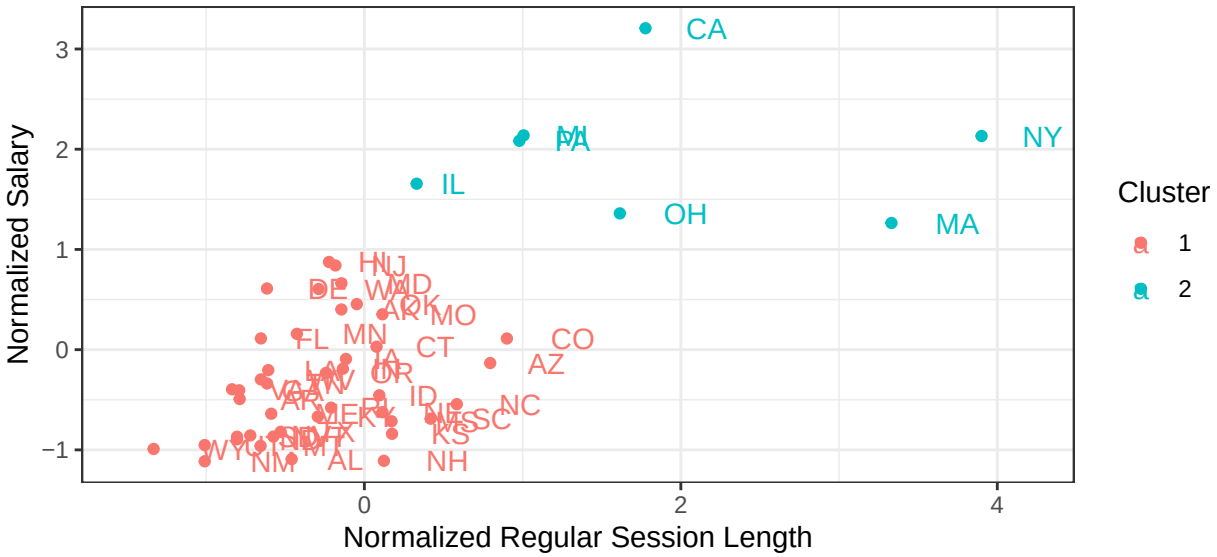
Fit an agglomerative hierarchical clustering algorithm using any linkage method you prefer, to these data and present the results. Give a quick, high level summary of the output and general patterns.

2.4.2 Solution

```
h_cluster <-
  lpf_clean %>%
  dist() %>%
  hclust(method = "complete" )

# Plot the dendogram
plot(h_cluster, cex = 0.6, hang = -1)
```



```r
# cut using k = 2
lpf_clean %>%
  as_tibble() %>%
  mutate(cluster = as.factor(cutree(h_cluster, k = 2))) %>%
  ggplot(aes(x = slength, y = salary_real, color = cluster)) +
  geom_point() +
  geom_text(aes(label = rownames(lpf_clean)), hjust = -1) +
  labs(x = "Normalized Regular Session Length",
       y = "Normalized Salary",
       color = "Cluster") +
  theme_bw() +
  coord_cartesian((xlim = c(-1.5, 4.2)))
```



We see that the first break picks out the states that were in the upper right of our plot when we did the exploration, that looked visually different from some of the rest of the states. This checks out with what we visually think is going on in the data if we just want $k = 2$.

2.5 k-Means (5 points)

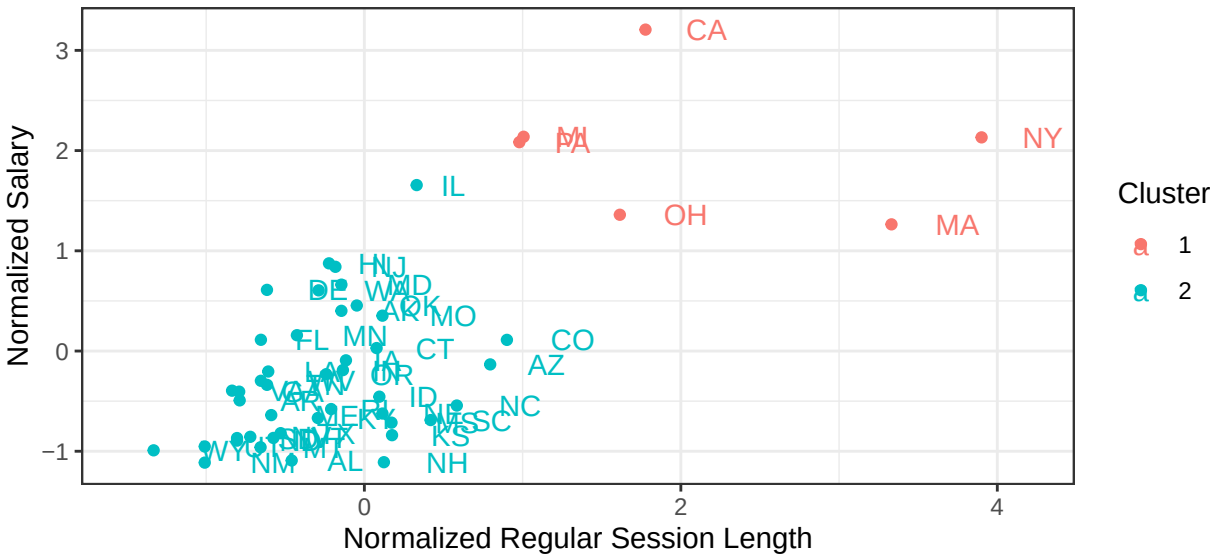
2.5.1 Problem Statement

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.

2.5.2 Solution

```r
k_cluster <- kmeans(lpf_clean, 2)

lpf_clean %>%
  as_tibble() %>%
  mutate(cluster = as.factor(k_cluster$cluster)) %>%
  ggplot(aes(x = slength, y = salary_real, color = cluster)) +
  geom_point() +
  geom_text(aes(label = rownames(lpf_clean)), hjust = -1) +
```

```
  labs(x = "Normalized Regular Session Length",
       y = "Normalized Salary",
       color = "Cluster") +
  theme_bw() +
  coord_cartesian((xlim = c(-1.5, 4.2)))
```



We see that kmeans is generally cutting along the same lines as the hirearchical clusters and returning (to a first pass) visually plausible results. We notice however that Illinois for example gets sorted differently than under hirearchical, which may be an interesting marginal case to explore.

2.6 Gaussian Mixture Model (5 points)

2.6.1 Problem Statement

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.

2.6.2 Solution

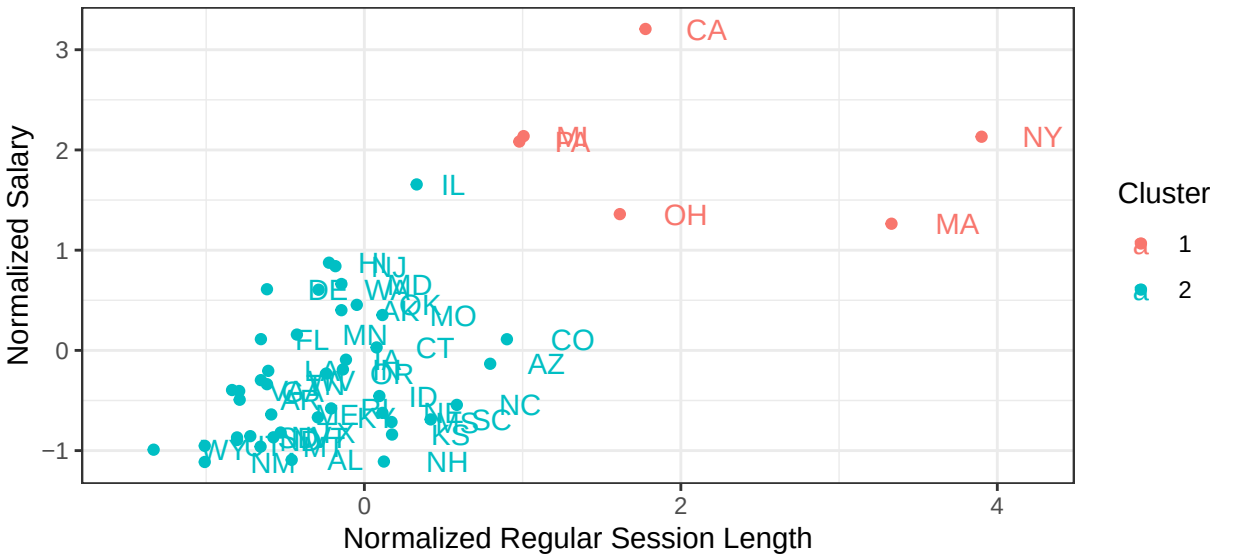
Essentially this model seems to be clustering similarly to the prior ones based on only looking at the two dimensions availab.e Futher tuning will be done later.

```
g_cluster <- mixtools::mvnormalmixEM(lpf_clean, k = 2)
```

```
## number of iterations= 14
```

```
# Assign to bins based on threshold of which is more likely:
lpf_clean %>%
  as_tibble() %>%
  mutate(cluster = as.factor(apply(g_cluster$posterior, 1,  which.max))) %>%
  ggplot(aes(x = slength, y = salary_real, color = cluster)) +
  geom_point() +
  geom_text(aes(label = rownames(lpf_clean)), hjust = -1) +
  labs(x = "Normalized Regular Session Length",
       y = "Normalized Salary",
       color = "Cluster") +
```

```
  theme_bw() +
  coord_cartesian((xlim = c(-1.5, 4.2)))
```



2.7 Compare (15 points)

2.7.1 Problem Statement

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.

2.7.2 Solution

We already have the dendrogram plotted above for the hirearchical cluster. Now, we will plot cluster assignments across all pairs of dimensions. It seems from the below plot that cluster assignment across various pairs of dimensions doesn't change very much between these different methods. Certainly not in particularly dramatic ways. We also see that the alrogrithms are all doing fairly plausible things in where they draw the lines between the clusters.

```
lpf_compare <-
  lpf_clean %>%
  as_tibble() %>%
  mutate(hirerarchical = factor(3 - cutree(h_cluster, k = 2) ),
         kmeans = as.factor(k_cluster$cluster),
         mixture = as.factor(apply(g_cluster$posterior, 1,  which.max)),
         state = rownames(lpf_clean)) %>%
  pivot_longer(c("hirerarchical", "kmeans", "mixture"), names_to = "cluster_strategy", values_to = "clus
  pivot_longer(c("t_slength", "slength", "salary_real", "expend"), names_to = "variable") %>%
  mutate(variable_x = variable, variable_y = variable, value_x = value, value_y = value)


x <- lpf_compare %>%
  select(state, cluster_strategy, variable_x, value_x) %>%
  unique()

y <- lpf_compare %>%
```

```
  select(state, cluster_strategy, variable_y, value_y) %>%
  unique()

c <- lpf_compare %>%
  select(state, cluster_strategy, cluster) %>%
  unique()

lpf_all <- expand.grid(list(
  "variable_x" = unique(lpf_compare$variable),
  "variable_y" = unique(lpf_compare$variable),
  "state" = unique(lpf_compare$state),
  "cluster_strategy" = unique(lpf_compare$cluster_strategy))) %>%
  as_tibble() %>%
  filter(variable_x != variable_y) %>%
  left_join(x) %>%
  left_join(y) %>%
  left_join(c)

lpf_all %>%
  ggplot(aes(x = value_x, y = value_y, color = cluster)) +
  facet_grid(variable_x + variable_y ~ cluster_strategy) +
  geom_point() +
  labs(x = "",
       y = "",
       color = "Cluster") +
  theme_bw()
```


hirerarchical
kmeans
mixture
salary_real
slength
t_slength
expend
Cluster
1
2
0
2
4

2.8 Validation (5 points)

2.8.1 Problem Statement

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.

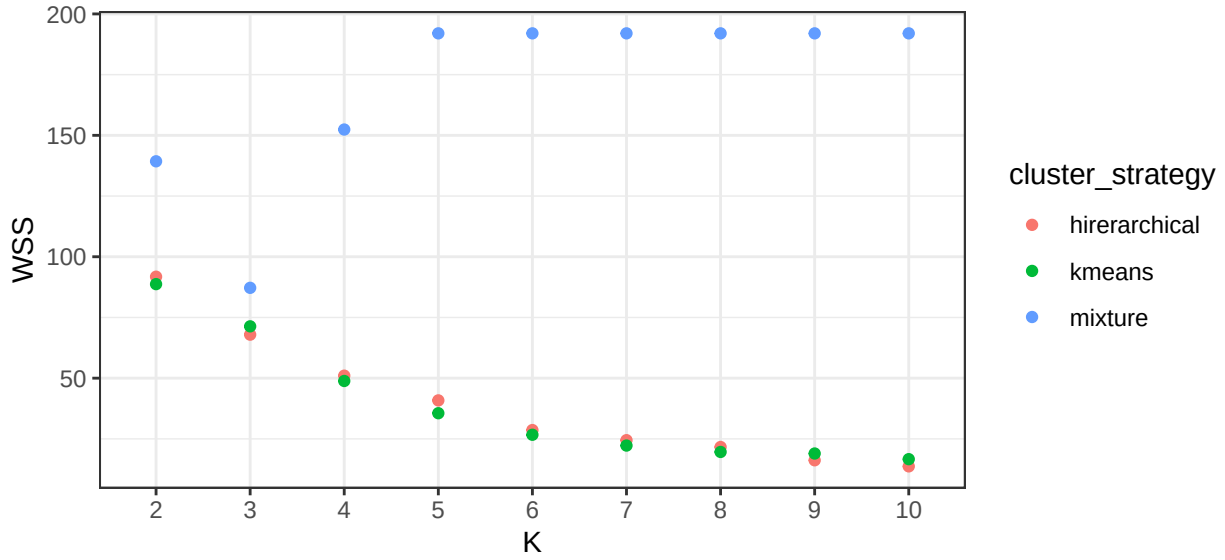
2.8.2 Solution

I will report the within cluster sum of squares for each of my above three clustering alrogrithms across a variety of values of k.

```
final_df <- tibble()

for (clusts in 2:10) {
  g_cluster <- NULL
  attempt <- 1
  while( is.null(g_cluster) && attempt <= 20 ) {
    attempt <- attempt + 1
    try(
      g_cluster <- mixtools::mvnormalmixEM(lpf_clean, k = clusts), silent = T
    )
    try(
      g_out <- as.factor(apply(g_cluster$posterior, 1,  which.max)), silent = T
    )
    if(attempt == 21) {g_out <- as.factor(NA)}
  }
  k_cluster <- kmeans(lpf_clean, clusts)


  lpf_c <-
    lpf_clean %>%
    as_tibble() %>%
    mutate(hirerarchical = factor(3 - cutree(h_cluster, k = clusts) ),
           kmeans = as.factor(k_cluster$cluster),
           mixture = g_out,
           state = rownames(lpf_clean)) %>%
    pivot_longer(c("hirerarchical", "kmeans", "mixture"), names_to = "cluster_strategy", values_to = "cl
    pivot_longer(c("t_slength", "slength", "salary_real", "expend"), names_to = "variable")

  d <- lpf_c %>%
    group_by(variable, cluster_strategy, cluster) %>%
    mutate(within_deviation = (value -mean(value))^2) %>%
    group_by(cluster_strategy) %>%
    summarise(WSS = sum(within_deviation), K = clusts)

  final_df <- bind_rows(final_df, d)
}
```

```
## number of iterations= 18
## number of iterations= 85
## number of iterations= 27
## Need new starting values due to singularity...
```

```r
final_df %>%
  ggplot(aes(x = as.factor(K), y = WSS, color = cluster_strategy)) +
  geom_point() +
  labs(x = "K") +
  theme_bw()
```



2.9 Discuss Validation (10 points)

2.9.1 Problem Statement

Discuss the validation output, e.g.,

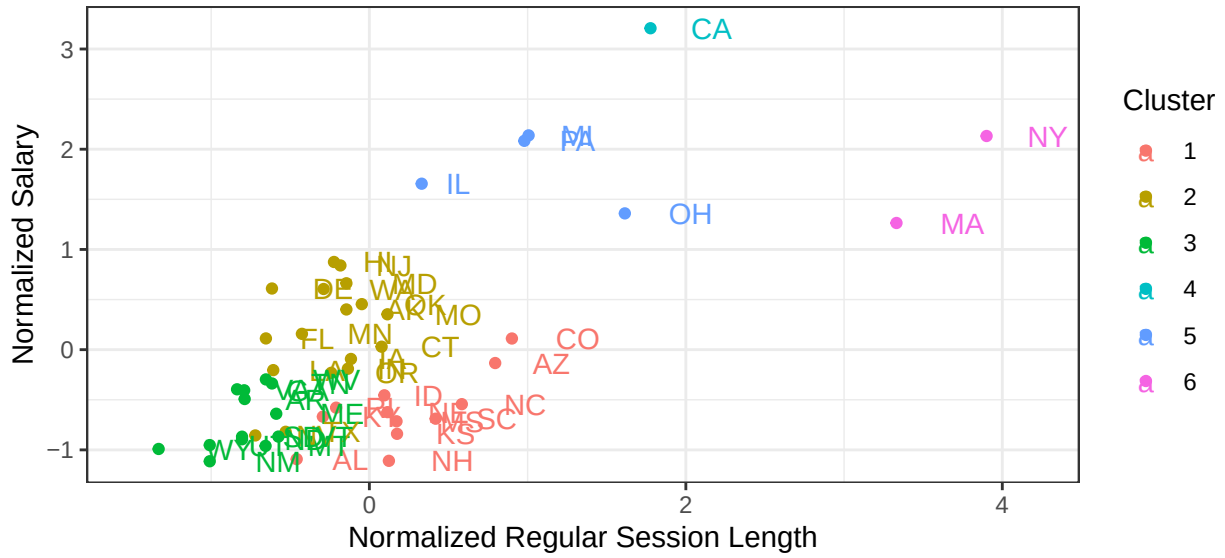
- What can you take away from the fit?
- Which approach is optimal? And optimal at what value of k?
- What are reasons you could imagine selecting a technically "sub-optimal" clustering method, regardless of the validation statistics?

2.9.2 Solution

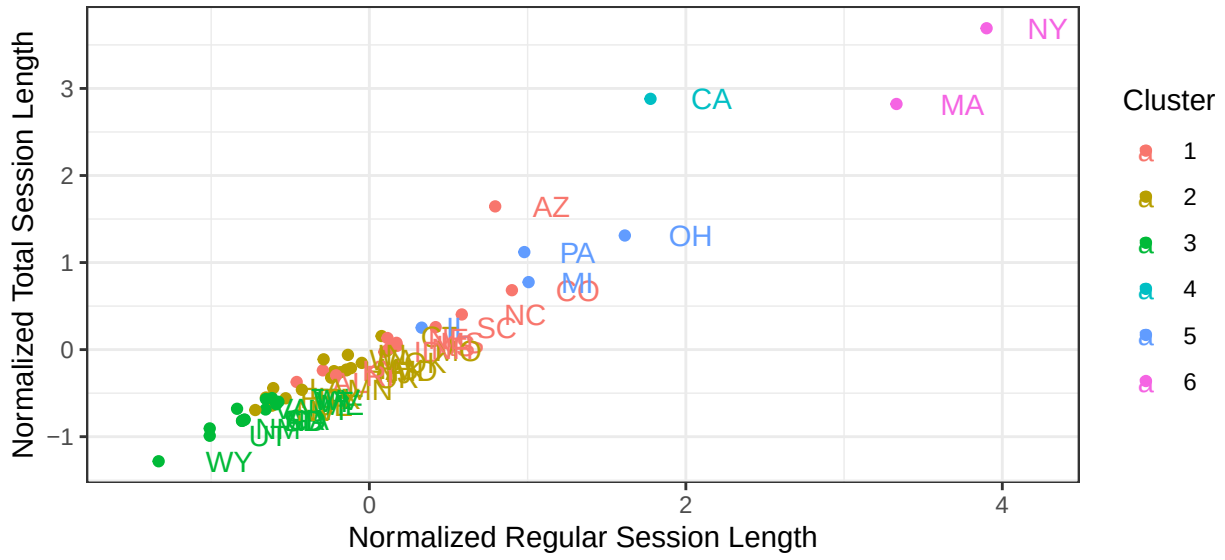
Based on the above, we learn a few things, first that the mixture model isn't generally very good as k increases, but for $k = 2$ it performs admiarbly. This likely has something to do with the fact that for these higher dimensions we really just don't have enough data to sensibly fit that many 4 dimensional gaussians. It seems that the hirerarchical clustering performs more reliably than the k-means clustering, so I will choose this as my optimal approach although the two are both quite good. In terms of tuning the optimal value of k, I think there is somewhat of a kink in the curve at $ K = 6$, so I will use this as my optimal parameter with the hirerarchical clustering model below. I report a variety of visualizations of this model's clusters.

There are a few reasons someone might wish to use a model that is "sub-optimal." A first and obvious one is computational speed. The Gaussian Mixture Model takes longer than the others to fit, and in some settings with large amounts of data, or when a model needs to be run quickly in a production environment, these computational speed issues mean we are willing to trade off a perfect clustering for a faster but still good one. Another reason might be consistency. If one method is really good on average but has a high standard deviation of its WSS depending on the random seed, we might choose not to use that one becuase we want a model that we know is essentially correct after fitting it instead of one where we have to do a bunch of random starts in order to hone in on what is likely the best clustering from that model.

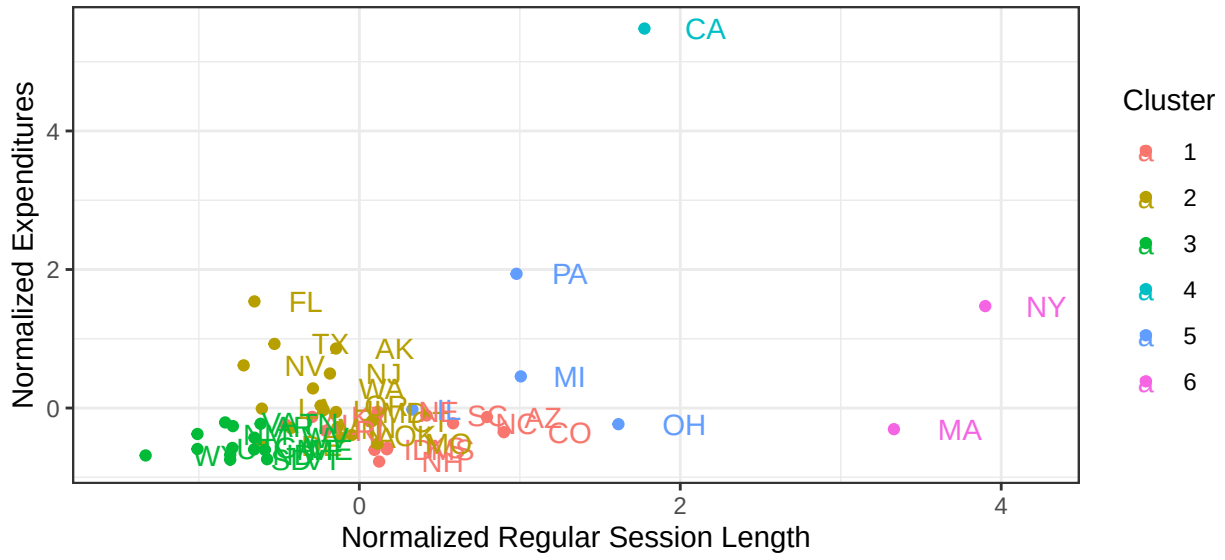
```
# cut using k = 6
lpf_clean %>%
  as_tibble() %>%
  mutate(cluster = as.factor(cutree(h_cluster, k = 6))) %>%
  ggplot(aes(x = slength, y = salary_real, color = cluster)) +
  geom_point() +
  geom_text(aes(label = rownames(lpf_clean)), hjust = -1) +
  labs(x = "Normalized Regular Session Length",
       y = "Normalized Salary",
       color = "Cluster") +
  theme_bw() +
  coord_cartesian((xlim = c(-1.5, 4.2)))
```




```
# cut using k = 6
lpf_clean %>%
  as_tibble() %>%
  mutate(cluster = as.factor(cutree(h_cluster, k = 6))) %>%
  ggplot(aes(x = slength, y = t_slength, color = cluster)) +
  geom_point() +
  geom_text(aes(label = rownames(lpf_clean)), hjust = -1) +
  labs(x = "Normalized Regular Session Length",
       y = "Normalized Total Session Length",
       color = "Cluster") +
  theme_bw() +
  coord_cartesian((xlim = c(-1.5, 4.2)))
```




```
# cut using k = 6
lpf_clean %>%
  as_tibble() %>%
  mutate(cluster = as.factor(cutree(h_cluster, k = 6))) %>%
  ggplot(aes(x = slength, y = expend, color = cluster)) +
  geom_point() +
  geom_text(aes(label = rownames(lpf_clean)), hjust = -1) +
  labs(x = "Normalized Regular Session Length",
       y = "Normalized Expenditures",
       color = "Cluster") +
  theme_bw() +
  coord_cartesian((xlim = c(-1.5, 4.2)))
```




```
# cut using k = 6
lpf_clean %>%
  as_tibble() %>%
  mutate(cluster = as.factor(cutree(h_cluster, k = 6))) %>%
```

```
ggplot(aes(x = expend, y = salary_real, color = cluster)) +
geom_point() +
geom_text(aes(label = rownames(lpf_clean)), hjust = -1) +
labs(x = "Normalized Expenditures",
     y = "Normalized Salary",
     color = "Cluster") +
theme_bw() +
coord_cartesian((xlim = c(-1.5, 4.2)))
```