Unsupervised Learning: PSet 3

Felipe Alamos

10/25/2019

```
#Setups
library(dplyr)
library(skimr)
library(seriation)
library(ggplot2)
library(dbscan)
library(mixtools)
library(plotGMM)
library(gridExtra)
library(clValid)
```

1.

Load the state legislative professionalism data from the relevant subdirectory in this repo. See the codebook for reference in the same subdirectory and combine with our discussion of these data and the concept of state legislative professionalism from class for relevant background information.

```
data <- get(load("StateLegProfData&Codebook/legprof-components.v1.0.RData"))
head(data)
```

```
##   fips stateabv   state  sessid t_slength slength salary_real   expend
## 1    1       AL Alabama  1973/4     46.00    36.0    1.768022 125.0973
## 2    1       AL Alabama  1975/6    110.00    74.0    2.933038 203.8466
## 3    1       AL Alabama  1977/8     83.00    60.0    2.082810 184.0115
## 4    1       AL Alabama 1979/80     65.00    60.0    1.694951 175.9863
## 5    1       AL Alabama  1981/2    218.68   149.1    3.472914 204.1236
## 6    1       AL Alabama  1983/4    188.86   149.1   11.705946 206.6745
##   year       mds1        mds2
## 1 1974 -1.7061814  0.38482016
## 2 1976 -1.2128817 -0.08150655
## 3 1978 -1.4149657  0.12789978
## 4 1980 -1.5431539  0.27268842
## 5 1982 -0.5013642 -1.00387760
## 6 1984 -0.5840194 -0.74132360
```

2.

Munge the data: * select only the continuous features that should capture a state legislature's level of "professionalism" (session length (total and regular), salary, and expenditures); * restrict the data to only include the 2009/10 legislative session for consistency; * omit all missing values; * standardize the input features; * 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)

```
data_filtered <- data %>%
  filter(sessid == "2009/10") %>%
```

```
  na.omit() %>%
  select(t_slength, slength, salary_real, expend)

#Standardizing
data_scaled <- as.data.frame(scale(data_filtered))

states <- data %>%
  filter(sessid == "2009/10") %>%
  na.omit() %>%
  select(stateabv)
```

3.

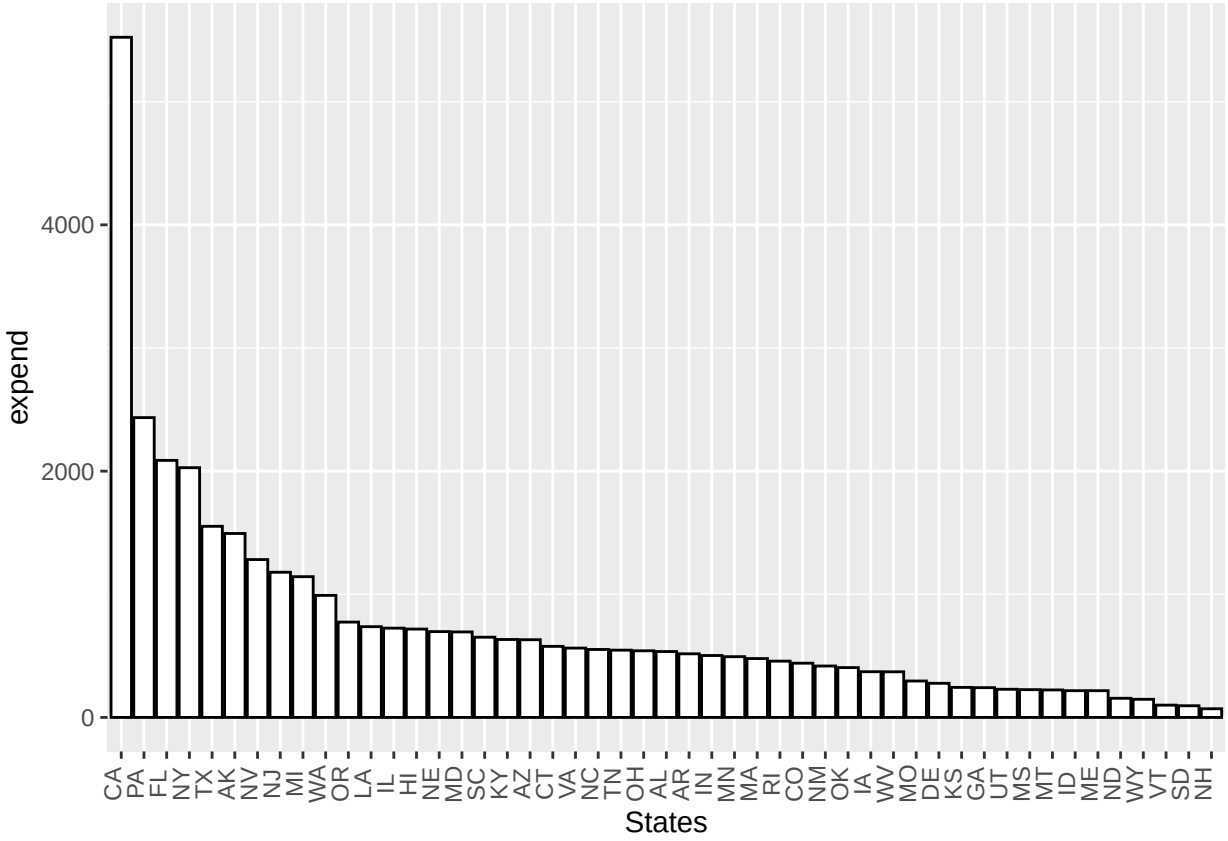
Perform quick EDA visually or numerically and discuss the patterns you see.

```
#skim(data_filtered), some problem with Knit happening
summary(data_filtered)
```

```
##    t_slength         slength       salary_real         expend
##  Min.   : 40.00   Min.   : 40.0   Min.   :  0.00   Min.   :  70.43
##  1st Qu.: 97.42   1st Qu.: 93.0   1st Qu.: 19.69   1st Qu.: 277.08
##  Median :127.77   Median :123.0   Median : 40.33   Median : 535.14
##  Mean   :147.80   Mean   :138.5   Mean   : 54.99   Mean   : 744.47
##  3rd Qu.:159.00   3rd Qu.:151.2   3rd Qu.: 77.43   3rd Qu.: 724.91
##  Max.   :458.15   Max.   :427.1   Max.   :213.41   Max.   :5523.10
```

- Most of the data is from 2010
- All numerical variables are skewed to the left, i.e, they have significant outliers on their high end. This is particularly exacerbated for expenditure, where its max value is more than 100 times the median value. Lets have a closer look at expend:

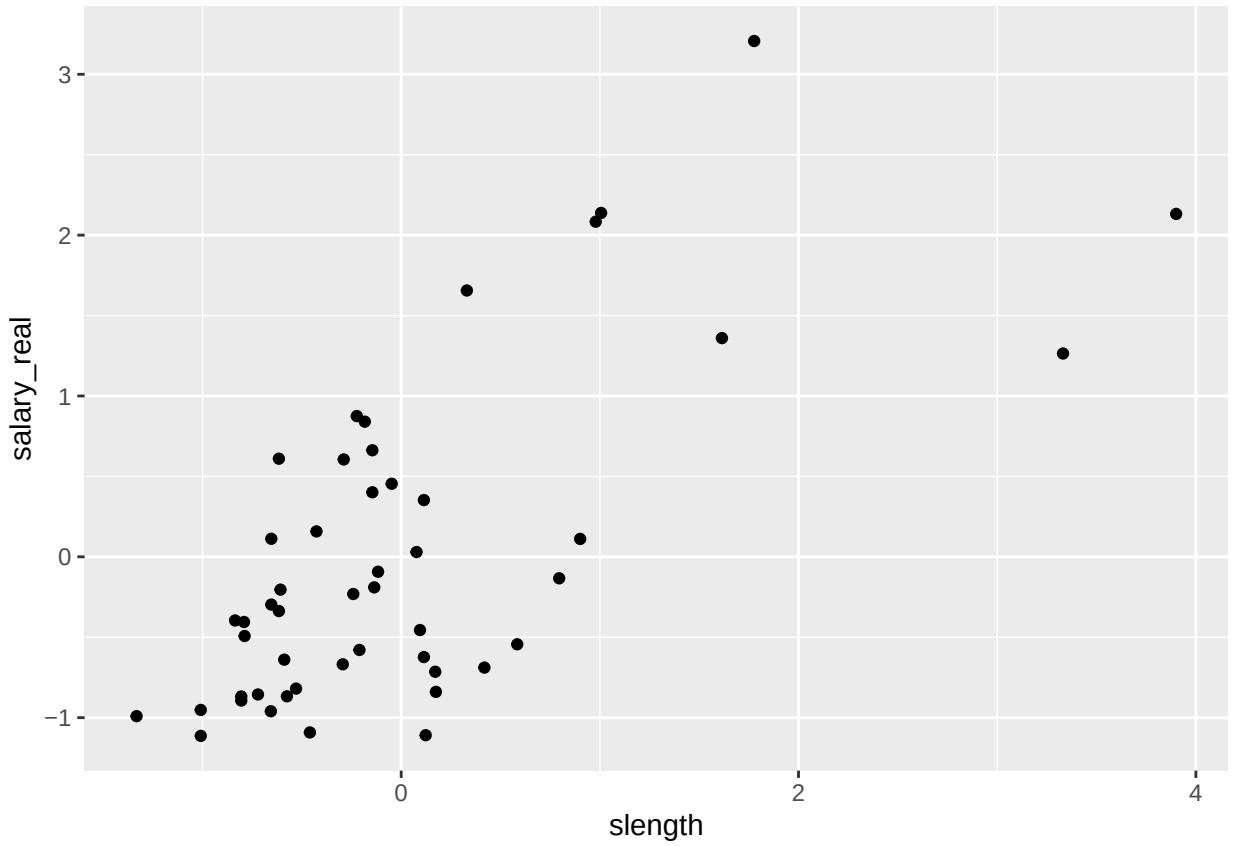
```
data_filtered_with_state<-data_filtered
data_filtered_with_state$state <- states$stateabv

ggplot(data_filtered_with_state, aes(x=reorder(as.factor(state),-expend), expend)) +
  geom_bar(stat="identity",color="black", fill="white")+
   theme(axis.text.x = element_text(size=9, angle = 90, vjust=-0.001))+
   labs(x = "States")
```



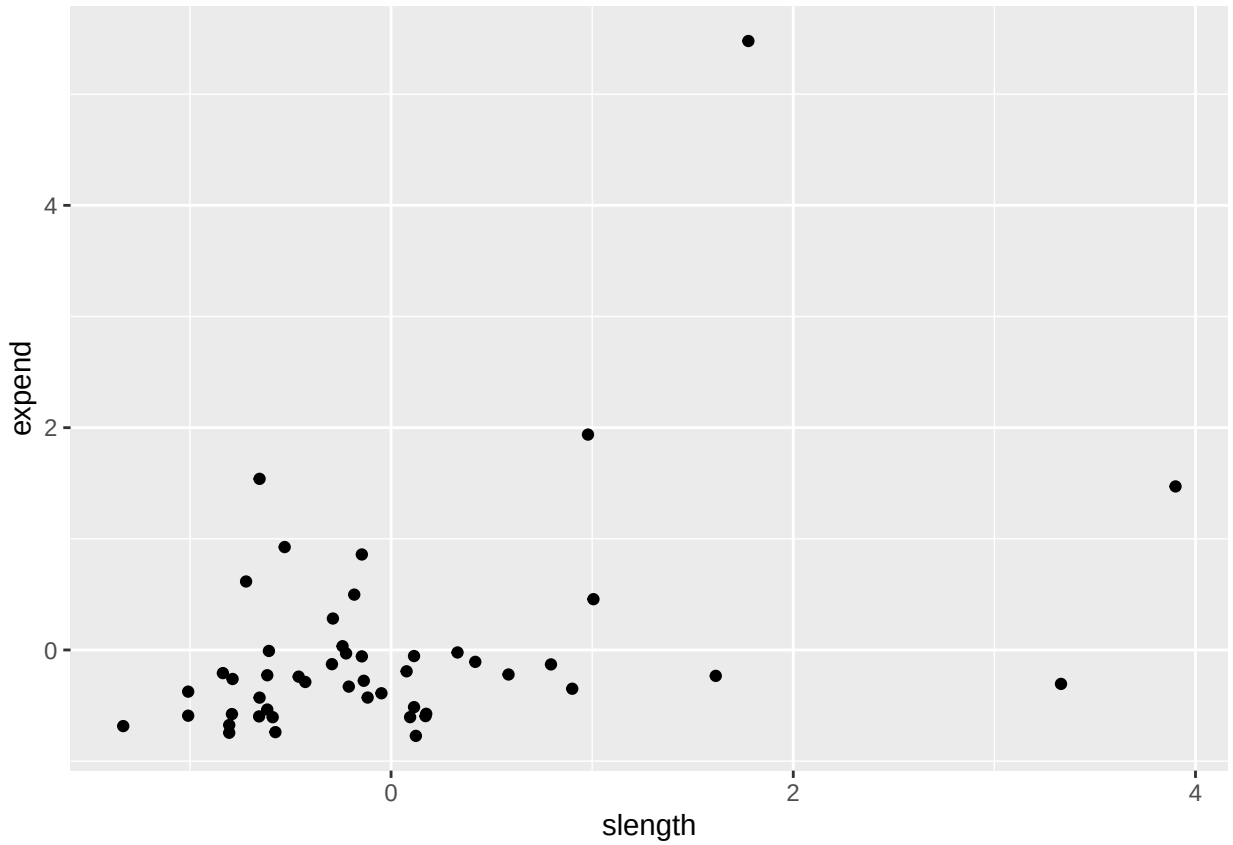
- We observe that the state with the extreme `expenditure` value is CA.
- We know try to understand correlation between the variables

session length (total and regular), salary, and expenditures

```
#stateabv, year, t_slength, slength, salary_real, expend
ggplot(data_scaled, aes(x=slength, y=salary_real)) +
  geom_point()
```




```
ggplot(data_scaled, aes(x=slength, y=expend)) +
  geom_point()
```

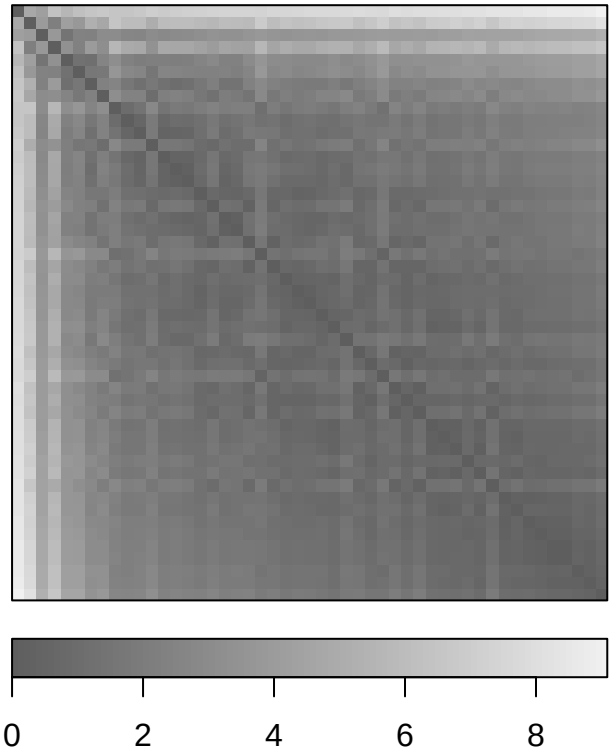


- We observe that there seems to be a positive correlation between length of regular session and salary. We might state the same for expenditure, but the correlation is much weaker.

4

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.

```
df_dist<- dist(data_scaled)
dissplot(df_dist)
```



- From a quick look at this ODI, it seems that there is one big cluster, and then a group of points that are not necessarily clustered together.
- It does look that some outliers exist - elements that are very far away in distance from most others (this might be the ones grouping in the small cluster).

5

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.

```
set.seed(123)
kmeans <- kmeans(data_scaled,
                 centers = 2,
                 nstart = 15)

str(kmeans)
```

```
## List of 9
##  $ cluster     : Named int [1:49] 1 1 1 1 2 1 1 1 1 1 ...
##   ..- attr(*, "names")= chr [1:49] "1" "2" "3" "4" ...
##  $ centers     : num [1:2, 1:4] -0.293 2.1 -0.293 2.101 -0.283 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "1" "2"
##   .. ..$ : chr [1:4] "t_slength" "slength" "salary_real" "expend"
##  $ totss       : num 192
##  $ withinss    : num [1:2] 48.4 40.4
##  $ tot.withinss: num 88.7
##  $ betweenss   : num 103
```

```
##  $ size        : int [1:2] 43 6
##  $ iter        : int 1
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

```
kmeans$size
```

```
## [1] 43  6
```

- We observe two clusters, one with 43 elements and the other with 6.
- We present the centers of the two clusters:

```
kmeans$centers
```

```
##    t_slength    slength salary_real     expend
## 1 -0.2930275 -0.2932285  -0.2833616 -0.2047966
## 2  2.1000302  2.1014710   2.0307585  1.4677087
```

-And the list indicating to which cluster was each state associated

```
kmeans$cluster
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
##  1  1  1  1  2  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  1  1  1
## 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 50
##  1  1  1  1  1  1  2  1  1  2  1  1  2  1  1  1  1  1  1  1  1  1  1  1
```

```
#We create a df where to save the clusters
data_scaled_with_clusters <- data_scaled
data_scaled_with_clusters$KmeansCluster <- as.factor(kmeans$cluster)
```

6.

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.

```
#We use the multi-variate method so as to consider all columns of data
set.seed(123)
gmm1 <- mvnormalmixEM(data_scaled, k = 2)
```

```
## number of iterations= 25
```

For the two cluster, we observe means on each of the variables:

```
gmm1$mu
```

```
## [[1]]
## [1] -0.2763465 -0.2450724 -0.1943875 -0.1921007
##
## [[2]]
## [1] 2.431846 2.156634 1.710608 1.690484
```

. . . standard deviations (notice that here we present the co variance matrix - in the diagonal we can find standard deviations)

```
gmm1$sigma
```

```
## [[1]]
##            [,1]       [,2]       [,3]       [,4]
```

```
## [1,] 0.24528126 0.27954563 0.2096713 0.03105197
## [2,] 0.27954563 0.32491503 0.2375517 0.02479181
## [3,] 0.20967129 0.23755167 0.5806866 0.13660923
## [4,] 0.03105197 0.02479181 0.1366092 0.23481667
##
## [[2]]
##           [,1]       [,2]      [,3]       [,4]
## [1,] 0.8556104  1.0197243 0.3979561  0.3509037
## [2,] 1.0197243  1.5611386 0.3426861 -0.3956420
## [3,] 0.3979561  0.3426861 1.2312537  1.9833537
## [4,] 0.3509037 -0.3956420 1.9833537  4.3511252
```

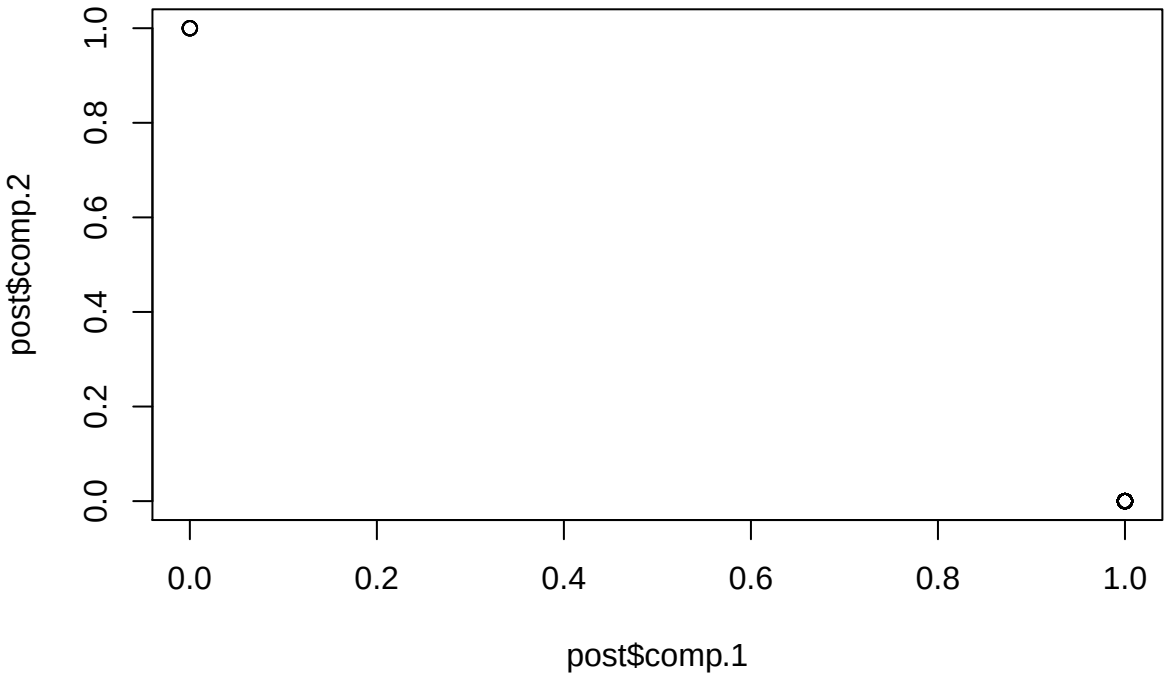
... and heights:

```
gmm1$lambda
```

```
## [1] 0.897959 0.102041
```

... and the probabilities of each point being part of each of the two clusters:

```
post <- as.data.frame(gmm1$posterior)

# quick viz distribution of mixture proportions
plot(post$comp.1, post$comp.2)
```



- We can observe that states are represented either by one distribution or the other, not a mixture of them

```
# get counts for each component
post$component <- ifelse(post$comp.1 < 0.2, 2, 1)
table(post$component)
```

```
##
##  1  2
## 44  5
```

- The first cluster has 44 states, the second one 5.

```
# We add the clusters obtained by gmm to our dataframe
data_scaled_with_clusters$ClusterGMM <- post$component
```

7.

Fit one additional partitioning technique of your choice (e.g., PAM, CLARA, fuzzy C-means, DBSCAN, etc.), and present and discuss results. Here again initialize at k=2.

We will be using DBSCAN:

```
#To identify the ideal epsilon, we can follow the techinque explained here: http://www.sthda.com/englis

dbscan <- dbscan(data_scaled,
                 eps= 0.5)
dbscan$cluster #0 indicates noise point
```

```
##  [1] 0 0 0 1 0 0 2 0 0 1 0 2 0 2 2 2 0 1 1 0 0 0 1 2 0 1 0 0 0 0 1 0 0 1 0
## [36] 0 1 0 2 0 1 1 0 1 1 1 0 1 1
```

```
n_non_clustered <- sum(dbscan$cluster == 0)
n_clustered_in_1<- sum(dbscan$cluster == 1)
n_clustered_in_2<- sum(dbscan$cluster == 2)
```

-We observe that the 49 data points have been separated in two clusters. One with 16 states, the other with 7 states, and 26 were not clustered. -Its interesting to notice that this algorithm - because of its totally different approach to clustering compared to kmeans - allows for the detection of outliers or samples that are not significantly coherent members of any cluster. In addition, the algorithm gives us the chance to modify this through the parameter epsilon (which determines the distance that must exist between the points to be part of the same cluster)

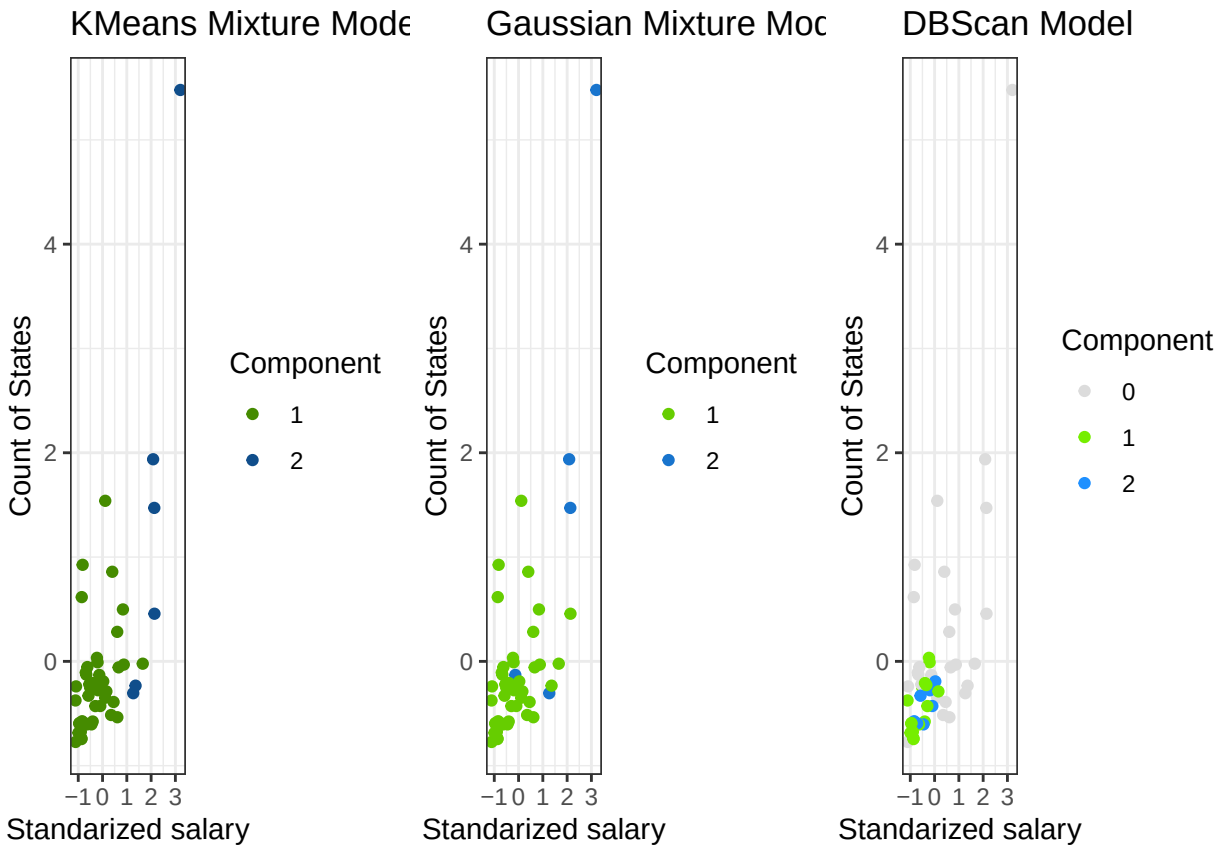
```
data_scaled_with_clusters$ClusterDBscan <- dbscan$cluster
```

8.

Compare output of all in a visually useful, simple way (e.g., plotting by state cluster assignment across two features like salary and expenditures).

```
# View the salary by clusters
kmeans_salary_plot <- ggplot(data_scaled_with_clusters, aes(x = salary_real, y=expend)) +
  geom_point(aes(colour = factor(KmeansCluster))) +
  labs(x = "Standarized salary",
       y = "Count of States",
       title = "KMeans Mixture Model") +
  scale_color_manual(values=c("chartreuse4", "dodgerblue4"),
                     name="Component") +
  theme_bw()

gmm_salary_plot <- ggplot(data_scaled_with_clusters, aes(x = salary_real, y=expend)) +
  geom_point(aes(colour = factor(ClusterGMM))) +
  labs(x = "Standarized salary",
       y = "Count of States",
       title = "Gaussian Mixture Model") +
  scale_color_manual(values=c("chartreuse3", "dodgerblue3"),
```

```
                    name="Component") +
  theme_bw()

dbscan_salary_plot <- ggplot(data_scaled_with_clusters, aes(x = salary_real, y=expend)) +
  geom_point(aes(colour = factor(ClusterDBscan))) +
  labs(x = "Standarized salary",
       y = "Count of States",
       title = "DBScan Model") +
  scale_color_manual(values=c("gainsboro", "chartreuse2", "dodgerblue1"),
                     name="Component") +
  theme_bw()

# view side by side
grid.arrange(kmeans_salary_plot, gmm_salary_plot, dbscan_salary_plot, ncol = 3)
```



- We can observe that KMeans and Gaussian Models fairly identify the same clusters. DBScan is more different and does not cluster many points, though some of them seem to be grouped in the same way as in the previous two methods.

9.

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 (k-means, GMM, X).

```r
#Possible values for clValid: "hierarchical", "kmeans", "diana", "fanny", "som", "model", "sota", "pam"
#I could not find a way to validate for gmm and dbscan, so doing it for hierarchical and pam instead

internal <- clValid(data_scaled, 2:10,
                    clMethods = c("kmeans", "pam", "hierarchical"),
                    validation = "internal")
summary(internal)
```

```
##
## Clustering Methods:
##  kmeans pam hierarchical
##
## Cluster sizes:
##  2 3 4 5 6 7 8 9 10
##
## Validation Measures:
##                                  2       3       4       5       6       7       8       9      10
##
## kmeans       Connectivity   8.4460 10.8960 16.1885 28.7437 30.7437 37.5266 39.4552 40.8694 45.6623
##              Dunn           0.1735  0.2581  0.2562  0.1090  0.1090  0.1108  0.1260  0.1324  0.1386
##              Silhouette     0.6458  0.6131  0.4932  0.3042  0.2858  0.2750  0.3131  0.3307  0.3288
## pam          Connectivity   7.9071 21.2952 25.4798 26.3464 35.7008 42.7266 49.2762 51.2762 53.8071
##              Dunn           0.1673  0.0324  0.0332  0.0670  0.0731  0.0991  0.1019  0.1019  0.1130
##              Silhouette     0.6204  0.2530  0.2673  0.2841  0.2993  0.2682  0.2564  0.2385  0.2415
## hierarchical Connectivity   6.0869  6.9536 16.1885 18.6774 20.6774 21.7607 27.5476 35.5813 37.5147
##              Dunn           0.3637  0.4371  0.2562  0.2836  0.2836  0.2836  0.2960  0.1568  0.1568
##              Silhouette     0.6994  0.6711  0.4932  0.4440  0.4284  0.3525  0.2553  0.2652  0.2630
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 6.0869 hierarchical 2
## Dunn         0.4371 hierarchical 3
## Silhouette   0.6994 hierarchical 2
```

```r
# par(mfrow = c(2, 2))
# plot(kmeans_internal, legend = FALSE,
#      type = "l",
#      main = " ")
```

Although the clValid method gives us scores for Connectivity, Dunn and Silhoutte, we will focus in a single validation strategy: Connectivity, due to its intuitive interpretation (measures proximity between individual observations in the same cluster, and hence a low connectivity score indicates good clustering).

We can observe that the hierarchical method is the one with best connectivity score, particularly when using 2 clusters (for 2 clusters, hierarchical has connectivity equal to 6.1, whereas kmeans and pam have 8.4 and 7.9 respectively). We can also notice that, for hierarchical, connectivity does not increase too much when moving from 2 to 3 clusters, but it does for kmeans and very significantly for pam.

10.

Discuss the validation output.

- What can you take away from the fit?

First of all, interesting enough, we can realize how a pairwise technique (hierarchical) performs better than partitioning techniques. Nevertheless, the differences in performance are not that significant, which seem to validate both approaches Second, its interesting to compare kmeans and pams, especially because they are fundamentally very similar in their algorithmic approach (only difference is in how to define the center of the clusters). Surprisingly, their performance on connectivity is very different, especially for k>2, with kmeans usually having better performance. This suggest that the idea of defining clusters based on fictitious points that are the result of averaging the points of a cluster, is better than using a point of the cluster to represent all of the set.

- Which approach is optimal? And optimal at what value of k?

When considering connectivity, the best approach is the hierarchical model, with k=2. If we do not want to consider hierarchical in the analysis, the best approach is pam with k=2.

- What are reasons you could imagine selecting a technically "sub-optimal" partitioning method, regardless of the validation statistics?

There might be domain specific restrictions that propose one method over other, or that makes some approach less reasonable. So, for example, it might be totally possible not to be willing to demand that all samples have to be in a cluster, and hence a gmm approach would be preferred, maybe relaxing restrictions of validations statistics. It could also happen that grouping around a fictitious average of points (like kmeans does), is not appropriate for certain context, and hence we could choose pam. This is particularly useful in context where we want to inform the public which is this member representing the cluster. It could also be the case that features of the samples are discrete values, and hence using an "average" representation points makes less sense.