Problem Set 3

William Parker

Contents


```
library(tidyverse)
```

```
## -- Attaching packages ----------------------------------------------------------------------- tidy
```

```
## v ggplot2 3.2.1           v purrr   0.3.2
## v tibble  2.1.3           v dplyr   0.8.3
## v tidyr   0.8.99.9000     v stringr 1.4.0
## v readr   1.3.1           v forcats 0.4.0
```

```
## -- Conflicts -------------------------------------------------------------------------- tidyverse_
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(mixtools)
```

```
## mixtools package, version 1.1.0, Released 2017-03-10
## This package is based upon work supported by the National Science Foundation under Grant No. SES-0518
```

1. Load the state legislative professionalism data from the folder

```
load("State Leg Prof Data & Codebook/legprof-components.v1.0.RData")

set.seed(1324)
```

2. Munge the data

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

clean_df <- x %>%
  filter(sessid == "2009/10") %>%
  na.omit() %>%
  select(t_slength, slength, salary_real, expend) %>%
  mutate_all(scale)


head(clean_df)
```

```
##    t_slength    slength salary_real     expend
## 1 -0.3716599 -0.4594723  -1.0920009 -0.2399910
## 2 -0.2294089 -0.1452309   0.4011333  0.8591198
## 3  1.6453067  0.7951955  -0.1335656 -0.1299408
## 4 -0.8036462 -0.7881756  -0.4923902 -0.2612061
## 5  2.8807257  1.7767099   3.2069914  5.4785453
## 6  0.6827338  0.9008887   0.1113595 -0.3485530
```

```
head(state_names)
```

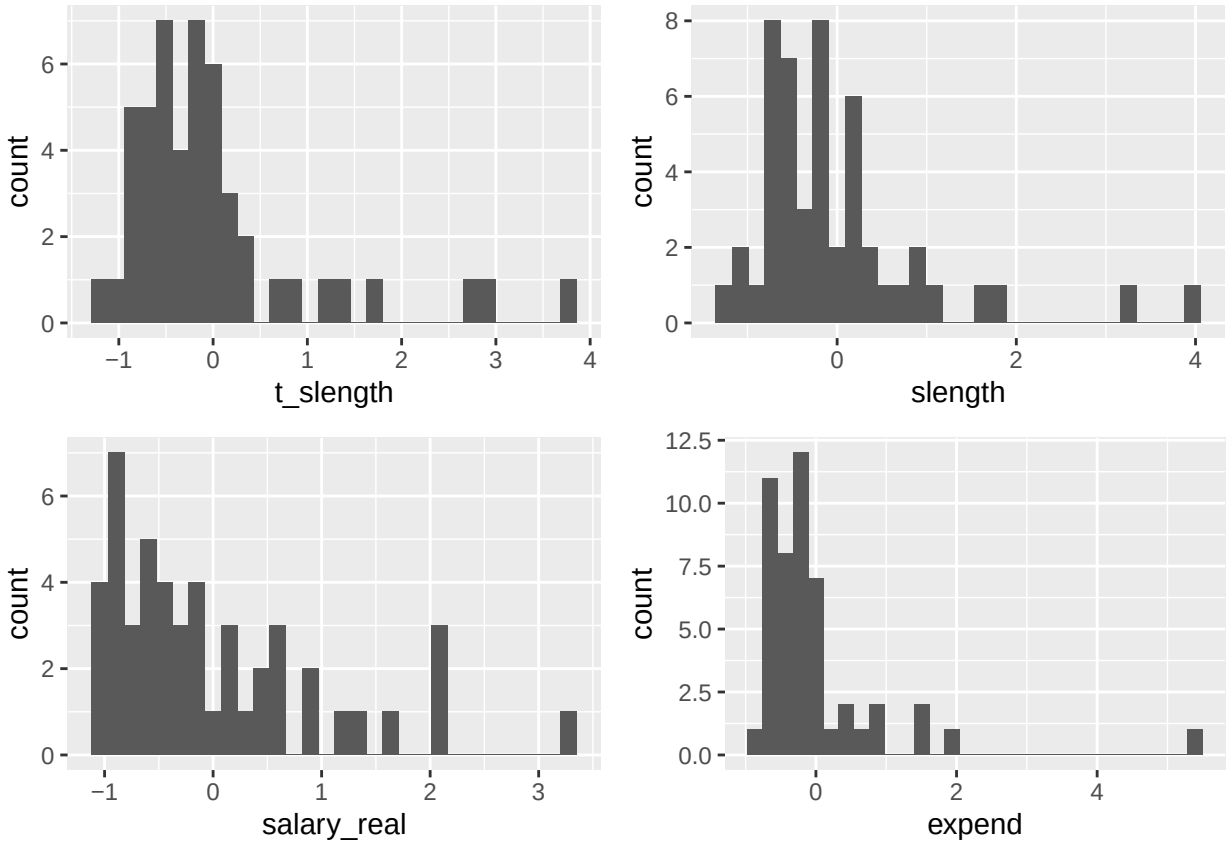
```
##   stateabv
## 1       AL
## 2       AK
## 3       AZ
## 4       AR
## 5       CA
## 6       CO
```

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

```
make_hist_list <- function(clean_df){
  var_names <- names(clean_df)

  plots <- vector('list', length(var_names))

  for (i in seq(1: length(var_names))) {
```

```
    x <- var_names[[i]]

    plots[[i]] <- clean_df %>%
      ggplot(aes_string(x = x)) +
      geom_histogram()
  }

  return(plots)
}

hist_list <- make_hist_list(clean_df)

cowplot::plot_grid(plotlist = hist_list)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

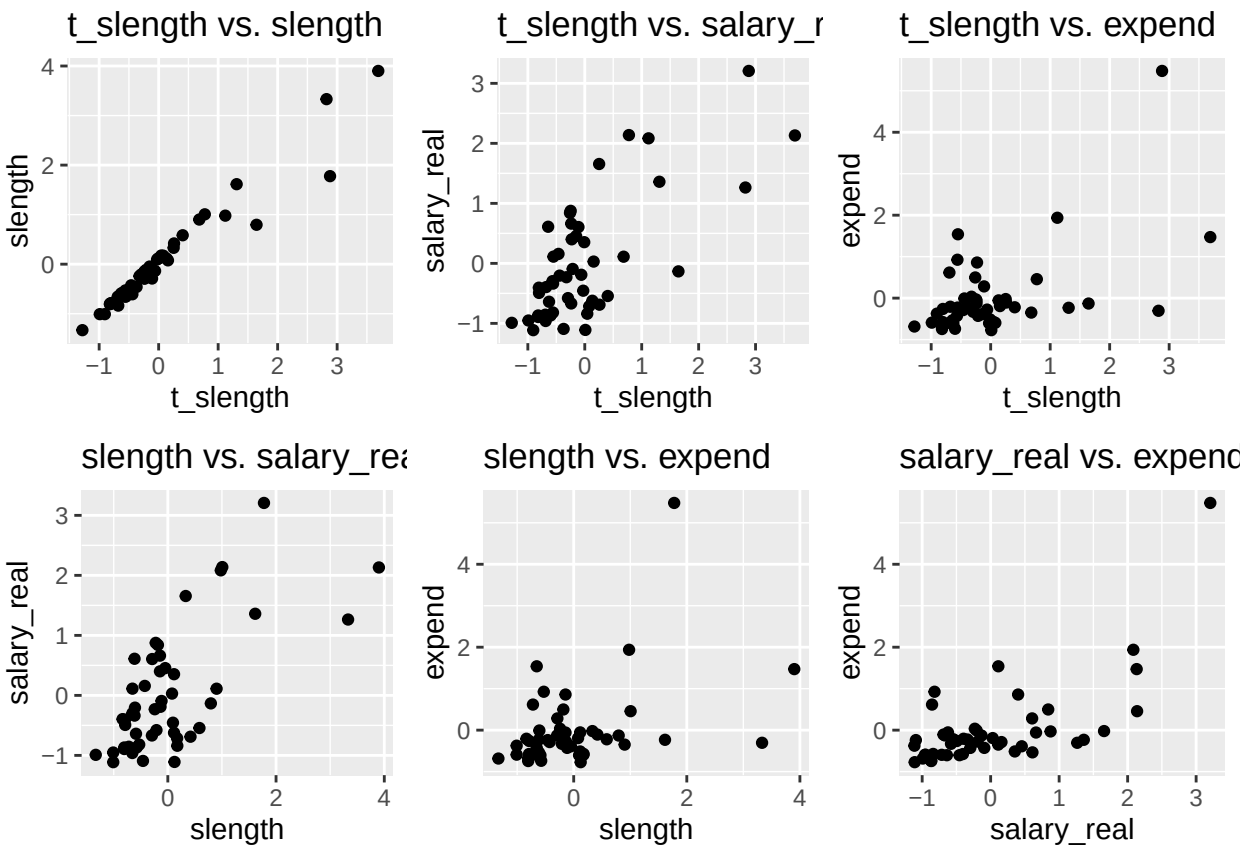


Looking at the histogram each normalized variable individually, the non-normaility of expenditures jumps out. Multiple outliers with very high expenditures and a bunch of states clustered below the mean.

All 4 variables have significant positive skew with long right tail. Each variable also appears bounded in the negative direction (>-2 on the standardized scale). This makes sense because there is presumably a minimum amount of time, salary, and expenditures required to actually be a state legislature at all.

```
make_plot_list <- function(clean_df){
  var_names <- combn(names(clean_df), m =2)
```

```r
  plots <- vector('list', dim(var_names)[2])


  for (i in seq(1:dim(var_names)[2])) {
    x <- var_names[1,i]
    y <- var_names[2,i]

    title <- paste(x, "vs.", y)

    plots[[i]] <- clean_df %>%
      ggplot(aes_string(x = x, y = y)) +
      geom_point() +
      labs(title = title)
  }

  return(plots)
}

plot_list <- make_plot_list(clean_df)

cowplot::plot_grid(plotlist = plot_list)
```



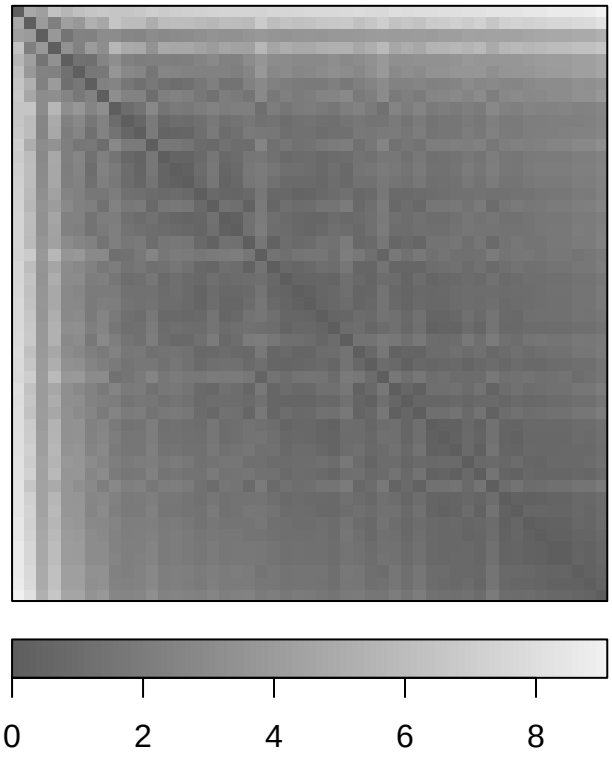
Total session length and length of regular sessions and very correlated, not suprisingly. There aren't as many clear linear relationships between the other variables, nor obvious clusters in any of the scatter plots aside from a group of states with values slightly below the mean in all variables.

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.

I decide to use ODI

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

```
## Registered S3 method overwritten by 'seriation':
##   method         from
##   reorder.hclust gclus
```



ODI suggests maybe two clusters in the data. I would guess the very large states may be clustered together in the upper left hand corner of the plot.

However compared to the iris ODI, the presence of clusterability is much less clear

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.

```
k_2_cluster <- kmeans(clean_df, centers = 2)

tibble(state_names = state_names$stateabv, cluster = k_2_cluster$cluster) %>%
  arrange(cluster)
```

```
## # A tibble: 49 x 2
##    state_names cluster
##    <I<chr>>      <int>
##  1 CA                1
##  2 MA                1
##  3 MI                1
##  4 NY                1
##  5 OH                1
##  6 PA                1
##  7 AL                2
##  8 AK                2
##  9 AZ                2
## 10 AR                2
## # ... with 39 more rows
```

Looks like cluster one is small and contains 6 large states (CA, MA, MI, NY, OH, PA). This is consistent with the ODI.

```
library(ggrepel)
make_plot_list_w_clusters <- function(clean_df, state_names, clusters, factor = TRUE){


  var_names <- combn(names(clean_df), m =2)

  combo_number <- dim(var_names)[2]
  plots <- vector('list', dim(var_names)[2])

  if (factor ==TRUE){
      clean_df$cluster <- factor(clusters)
  }

  if (factor != TRUE){
    clean_df$cluster <- clusters
  }

  clean_df$state_name <- state_names$stateabv

  for (i in seq(1:combo_number)) {
    x <- var_names[1,i]
    y <- var_names[2,i]

    title <- paste(x, "vs.", y)

    plots[[i]] <- clean_df %>%
      ggplot(aes_string(x = x, y = y)) +
```

```
      geom_label(aes(label = state_name, color = cluster), size = 2) +
      labs(title = title) +
      theme(legend.position = "none")
  }

  return(cowplot::plot_grid(plotlist = plots))
}

kmeans_scatter <- make_plot_list_w_clusters(clean_df, state_names, k_2_cluster$cluster)

kmeans_scatter
```

Internal Validity checks to fiind ideal K

```
library(clValid)
```

```
## Loading required package: cluster
```

```
wss <- function(k){
  kmeans(clean_df, k)$tot.withinss
}

dunn_extract <- function(k){
  dunn(Data = data.matrix(clean_df), clusters = kmeans(clean_df, centers = k)$cluster)
}
```

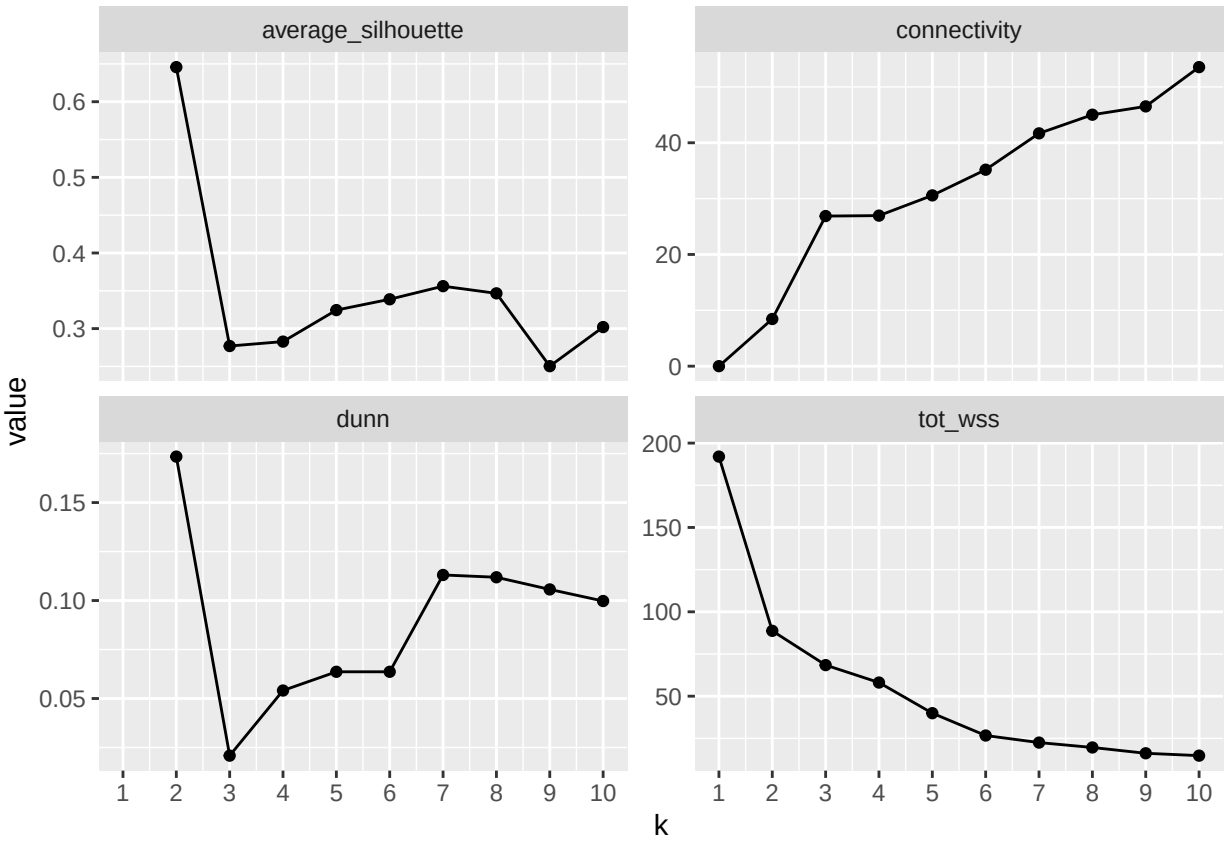
```
connect_extract <- function(k){
  connectivity(Data = data.matrix(clean_df), clusters = kmeans(clean_df, centers = k)$cluster)
}

avg_silhouette <- function(k){
  if (k ==1) {
    return(NaN)
  }
  mean(cluster::silhouette(kmeans(clean_df, centers = k)$cluster, dist = dist(clean_df))[,3])
}


k_values <- 1:10

wss_values <- map_dbl(k_values, wss)
dunn_values <- map_dbl(k_values, dunn_extract)
```

```
## Warning in min(interClust, na.rm = TRUE): no non-missing arguments to min;
## returning Inf
```

```
connect_values <- map_dbl(k_values, connect_extract)
sil_values <- map_dbl(k_values, avg_silhouette)


find_ideal_k <- tibble(k = k_values,
                       tot_wss = wss_values,
                       dunn = dunn_values,
                       connectivity = connect_values,
                       average_silhouette = sil_values) %>%
  mutate_all(function(x) ifelse(is.infinite(x), NaN, x)) %>%
  pivot_longer(cols = c("tot_wss", "dunn", "connectivity", "average_silhouette"),
               names_to = "validity_measure",
               values_to = "value")

kmeans_valid <- find_ideal_k %>%
  ggplot(aes(x = k, y = value)) +
  scale_x_continuous(breaks = k_values) +
  geom_point() + geom_line() + facet_wrap(~validity_measure, scales = "free_y")

kmeans_valid
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```

```
## Warning: Removed 1 rows containing missing values (geom_path).
```



For K-means, look like Dunn index is maximized, the average silhouette is maximized, and connectivity minimized at 2- clear that k =2 is ideal. The total within sum of squares drops dramatically from 1 to 2 clusters so it kind of looks like the elbow is present at 2.

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.

I first tried to fit a multivariate gaussian mixture model to all 4 variables

```
mvnormalmixEM(data.matrix(clean_df),
                    k = 2)
```

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

```
## $x
##          t_slength     slength salary_real       expend
##  [1,] -0.371659901 -0.45947229 -1.09200089 -0.239991004
##  [2,] -0.229408926 -0.14523091  0.40113330  0.859119849
##  [3,]  1.645306675  0.79519547 -0.13356561 -0.129940807
##  [4,] -0.803646214 -0.78817557 -0.49239021 -0.261206056
##  [5,]  2.880725686  1.77670986  3.20699144  5.478545259
##  [6,]  0.682733830  0.90088869  0.11135948 -0.348553011
##  [7,]  0.155953294  0.07723846  0.02971780 -0.191902436
##  [8,] -0.643316513 -0.61557933  0.61016386 -0.535852545
##  [9,] -0.550306302 -0.65382896  0.11213913  1.539500636
## [10,] -0.806381805 -0.79128419 -0.40535082 -0.576530958
## [11,] -0.247368734 -0.22362235  0.87503597 -0.030968419
## [12,] -0.026736988  0.09467375 -0.45539720 -0.604391173
## [13,]  0.252174862  0.33052374  1.65585716 -0.022426587
## [14,] -0.060515652 -0.13563473 -0.19004288 -0.277057831
## [15,] -0.212519663 -0.11644243 -0.09274477 -0.428020271
## [16,]  0.043793814  0.17482220 -0.83953996 -0.574076126
## [17,] -0.238210429 -0.29390430 -0.66821257 -0.127784042
## [18,] -0.441952865 -0.60746990 -0.20424551 -0.008731385
## [19,] -0.627973378 -0.58854782 -0.63936779 -0.604554127
## [20,] -0.237853581 -0.14523091  0.66244097 -0.057590394
## [21,]  2.821493958  3.33129222  1.26402392 -0.305301736
## [22,]  0.775506248  1.00631164  2.13811467  0.456899531
## [23,] -0.461458846 -0.42635872  0.15793203 -0.287914432
## [24,]  0.078048154  0.17144321 -0.71457282 -0.594888550
## [25,] -0.009490831  0.11427161  0.35281484 -0.514269618
## [26,] -0.687442853 -0.65612661 -0.96022656 -0.597593000
## [27,]  0.133235911  0.11427161 -0.62341582 -0.054489563
## [28,] -0.693865590 -0.72100229 -0.85573358  0.616190643
## [29,]  0.010371952  0.12332714 -1.10918394 -0.772769645
## [30,] -0.259262629 -0.18307507  0.84033067  0.498016486
## [31,] -0.904982206 -1.00888793 -1.11326602 -0.374895524
## [32,]  3.691294567  3.90071117  2.13199154  1.471428802
## [33,]  0.403940989  0.58407939 -0.54377433 -0.220619983
## [34,] -0.818275700 -0.80479995 -0.89381810 -0.675224373
## [35,]  1.310731529  1.61452076  1.35982443 -0.233023694
## [36,] -0.152217572 -0.04791749  0.45425467 -0.389504090
## [37,] -0.322300309 -0.24119288 -0.23104580  0.033991598
## [38,]  1.120429207  0.97928013  2.08360490  1.937703837
## [39,] -0.294944314 -0.21010659 -0.57896192 -0.329505886
## [40,]  0.258478612  0.41878162 -0.68872917 -0.106730457
```

```
## [41,] -0.818275700 -0.80479995 -0.86834092 -0.744838524
## [42,] -0.556610007 -0.61557933 -0.33730246 -0.227228690
## [43,] -0.558750912 -0.52907846 -0.81935590  0.925725780
## [44,] -0.989428844 -1.00888788 -0.95212512 -0.591232328
## [45,] -0.599190173 -0.57503206 -0.86728615 -0.739004684
## [46,] -0.679355001 -0.83615650 -0.39561825 -0.207959325
## [47,] -0.111183625 -0.28917375  0.60553675  0.282936858
## [48,] -0.567195612 -0.65382896 -0.29684902 -0.428715471
## [49,] -1.282137610 -1.33191452 -0.99080347 -0.684772532
##
## $lambda
## [1] 0.122743 0.877257
##
## $mu
## $mu[[1]]
## [1] 2.094531 2.095873 2.029265 1.463509
##
## $mu[[2]]
## [1] -0.2930600 -0.2932479 -0.2839282 -0.2047695
##
##
## $sigma
## $sigma[[1]]
##           [,1]       [,2]       [,3]       [,4]
## [1,] 1.1750992  1.0805604  0.1461815  0.6872095
## [2,] 1.0805604  1.2666146 -0.1622966 -0.3348949
## [3,] 0.1461815 -0.1622966  0.4071369  1.1884016
## [4,] 0.6872095 -0.3348949  1.1884016  3.8917291
##
## $sigma[[2]]
##            [,1]        [,2]      [,3]        [,4]
## [1,] 0.25252966 0.232776099 0.1014131 0.020065982
## [2,] 0.23277610 0.238827852 0.1052663 0.009251876
## [3,] 0.10141308 0.105266268 0.4029072 0.105171262
## [4,] 0.02006598 0.009251876 0.1051713 0.230522323
##
##
## $loglik
## [1] -106.8507
##
## $posterior
##             comp.1       comp.2
##  [1,] 2.976066e-46 1.000000e+00
##  [2,] 1.751136e-28 1.000000e+00
##  [3,] 1.332100e-62 1.000000e+00
##  [4,] 7.804244e-32 1.000000e+00
##  [5,] 1.000000e+00 3.696926e-39
##  [6,] 8.042665e-13 1.000000e+00
##  [7,] 7.573037e-16 1.000000e+00
##  [8,] 2.267901e-07 9.999998e-01
##  [9,] 2.238557e-47 1.000000e+00
## [10,] 1.870072e-24 1.000000e+00
## [11,] 1.642515e-05 9.999836e-01
## [12,] 3.098434e-24 1.000000e+00
```

```
## [13,] 2.478091e-02 9.752191e-01
## [14,] 8.014669e-20 1.000000e+00
## [15,] 2.313619e-18 1.000000e+00
## [16,] 1.679347e-35 1.000000e+00
## [17,] 1.333543e-34 1.000000e+00
## [18,] 1.637195e-22 1.000000e+00
## [19,] 3.953443e-30 1.000000e+00
## [20,] 3.151997e-08 1.000000e+00
## [21,] 1.000000e+00 1.028229e-12
## [22,] 9.915131e-01 8.486866e-03
## [23,] 1.098076e-14 1.000000e+00
## [24,] 3.135861e-30 1.000000e+00
## [25,] 2.211631e-09 1.000000e+00
## [26,] 3.496583e-40 1.000000e+00
## [27,] 1.904226e-33 1.000000e+00
## [28,] 8.424611e-66 1.000000e+00
## [29,] 5.005083e-40 1.000000e+00
## [30,] 9.552556e-11 1.000000e+00
## [31,] 1.300700e-46 1.000000e+00
## [32,] 1.000000e+00 1.951422e-17
## [33,] 1.617195e-32 1.000000e+00
## [34,] 1.109368e-36 1.000000e+00
## [35,] 9.981523e-01 1.847715e-03
## [36,] 6.516100e-09 1.000000e+00
## [37,] 1.568994e-29 1.000000e+00
## [38,] 9.999434e-01 5.659044e-05
## [39,] 8.414057e-33 1.000000e+00
## [40,] 2.168462e-40 1.000000e+00
## [41,] 1.016641e-34 1.000000e+00
## [42,] 3.617685e-25 1.000000e+00
## [43,] 4.884149e-79 1.000000e+00
## [44,] 6.581056e-40 1.000000e+00
## [45,] 3.778037e-34 1.000000e+00
## [46,] 1.034035e-25 1.000000e+00
## [47,] 1.375476e-09 1.000000e+00
## [48,] 7.122661e-22 1.000000e+00
## [49,] 1.047212e-39 1.000000e+00
##
## $all.loglik
##  [1] -382.5541 -119.4617 -108.8866 -107.2968 -106.8796 -106.8515 -106.8508
##  [8] -106.8507 -106.8507 -106.8507 -106.8507 -106.8507 -106.8507
##
## $restarts
## [1] 0
##
## $ft
## [1] "mvnormalmixEM"
##
## attr(,"class")
## [1] "mixEM"
```

I suspect that this error had something to do with how highly correlated `t_slength` and `slength` were in the data. so I dropped `t_slength` from the feature space and tried again

```
gmm_k2 <- mvnormalmixEM(data.matrix(clean_df %>% select(-t_slength)),
                        k = 2)
```

```
## number of iterations= 35
```

GMM fitted means

```
tibble( variable = names(clean_df %>% select(-t_slength)),
  mean_1 = gmm_k2$mu[[1]],
       mean_2 = gmm_k2$mu[[2]])
```

```
## # A tibble: 3 x 3
##   variable    mean_1 mean_2
##   <chr>        <dbl>  <dbl>
## 1 slength      0.964 -0.241
## 2 salary_real  0.878 -0.220
## 3 expend       1.24  -0.310
```

Inspecting the fitted means of the distribution, looks like GMM is identifying a distribution of states with "big" numbers on session length, salary and expenditures and another distribution with low numbers. Similar result to K-means

GMM covariance matrix

```
gmm_k2$sigma[[1]]
```

```
##           [,1]     [,2]      [,3]
## [1,] 2.5745757 1.527811 0.2309351
## [2,] 1.5278106 1.732329 1.3122551
## [3,] 0.2309351 1.312255 2.6024190
```

Inspecting the estimated variance-covariance matrix $\hat{\Sigma}_1$ for the first component, we see wide variance in each of the three parameters. This matches with the large state group being spread out in feature space

```
gmm_k2$sigma[[2]]
```

```
##            [,1]      [,2]       [,3]
## [1,] 0.29035275 0.1878141 0.06396421
## [2,] 0.18781414 0.5503432 0.14867523
## [3,] 0.06396421 0.1486752 0.09331835
```

Inspecting the estimated variance-covariance matrix $\hat{\Sigma}_2$ for the second component, we see much lower variance (especially in session length and expenditures). This makes sense as graphically this second group of states was much closer togehter in the feature space

```
gmm_k2$posterior
```

```
##            comp.1       comp.2
##  [1,] 0.020718647 9.792814e-01
##  [2,] 0.995648512 4.351488e-03
##  [3,] 0.082902668 9.170973e-01
##  [4,] 0.010170266 9.898297e-01
##  [5,] 1.000000000 9.736351e-99
##  [6,] 0.125556914 8.744431e-01
##  [7,] 0.009043478 9.909565e-01
```

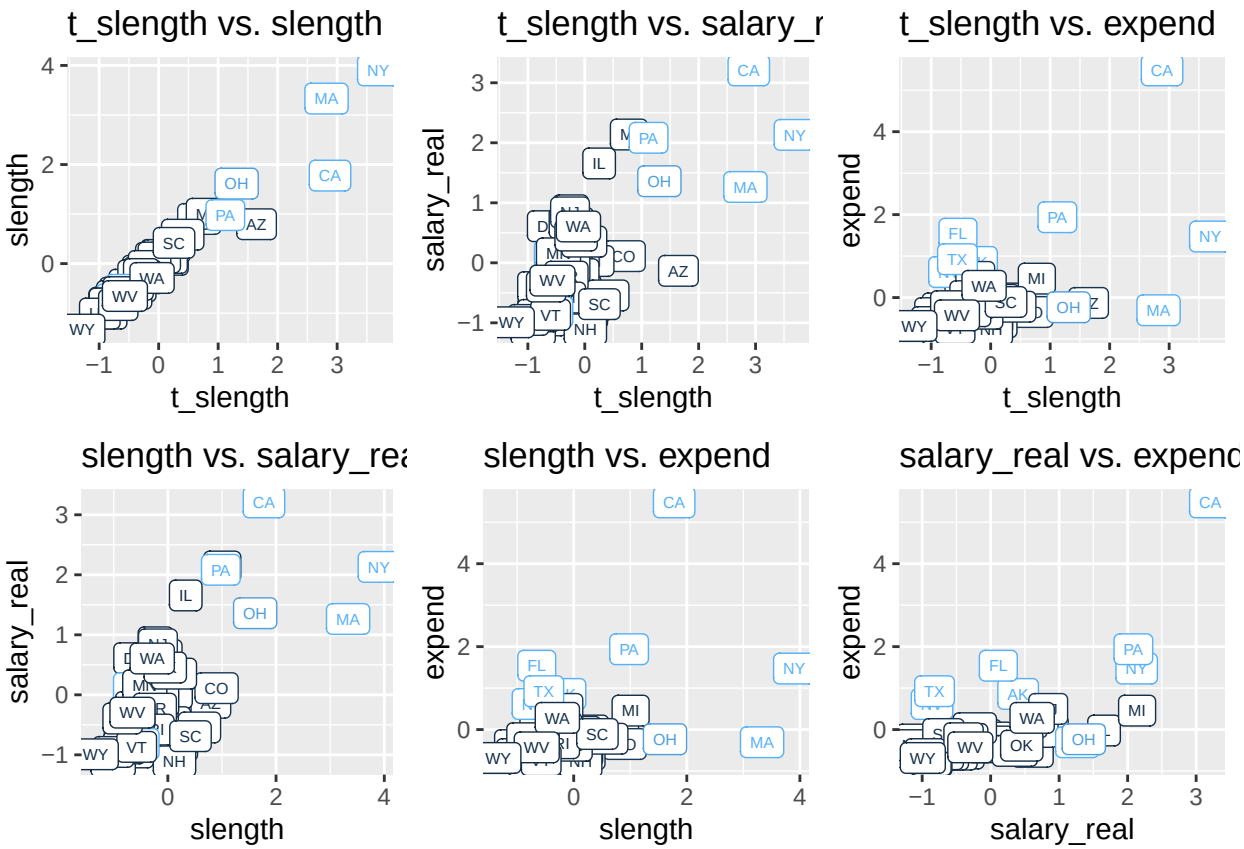
```
##  [8,] 0.004194890 9.958051e-01
##  [9,] 1.000000000 2.976798e-12
## [10,] 0.007218092 9.927819e-01
## [11,] 0.002453455 9.975465e-01
## [12,] 0.019284356 9.807156e-01
## [13,] 0.002657491 9.973425e-01
## [14,] 0.007228396 9.927716e-01
## [15,] 0.007195236 9.928048e-01
## [16,] 0.019647521 9.803525e-01
## [17,] 0.020163723 9.798363e-01
## [18,] 0.020062960 9.799370e-01
## [19,] 0.008219748 9.917803e-01
## [20,] 0.003610400 9.963896e-01
## [21,] 0.999999998 2.207111e-09
## [22,] 0.041915523 9.580845e-01
## [23,] 0.003852089 9.961479e-01
## [24,] 0.021144903 9.788551e-01
## [25,] 0.014525556 9.854744e-01
## [26,] 0.009006434 9.909936e-01
## [27,] 0.037882936 9.621171e-01
## [28,] 0.999146194 8.538060e-04
## [29,] 0.033209357 9.667906e-01
## [30,] 0.112732096 8.872679e-01
## [31,] 0.021030700 9.789693e-01
## [32,] 1.000000000 1.262384e-14
## [33,] 0.047628765 9.523712e-01
## [34,] 0.010639416 9.893606e-01
## [35,] 0.879266244 1.207338e-01
## [36,] 0.005743442 9.942566e-01
## [37,] 0.026147851 9.738521e-01
## [38,] 1.000000000 3.695672e-11
## [39,] 0.008878312 9.911217e-01
## [40,] 0.052010031 9.479900e-01
## [41,] 0.013037043 9.869630e-01
## [42,] 0.008107621 9.918924e-01
## [43,] 0.999998678 1.322276e-06
## [44,] 0.011828676 9.881713e-01
## [45,] 0.012688353 9.873116e-01
## [46,] 0.011469117 9.885309e-01
## [47,] 0.025891464 9.741085e-01
## [48,] 0.005520624 9.944794e-01
## [49,] 0.022513018 9.774870e-01
```

```
responsibility_1 <- gmm_k2$posterior[,1]
```

Looking at the posterior responsibilites for each state,the results are essentially equivalent to a hard partition. No state actually represents a "mixture" of the two components.

This becomes apparent when plotting:

```
gmm_plots <- make_plot_list_w_clusters(clean_df, state_names, responsibility_1, factor = FALSE)

gmm_plots
```



Here responsibility is mapped as a continuous color aesthetic, but because all the values are so close ot 0 or 1 it effectively becomes a hard partition.

comparision to kmeans

```
compare_groups <-
  tibble(state_name = state_names$stateabv,
         gmm_clusters = round(responsibility_1),
         kmeans_clusters = k_2_cluster$cluster
         ) %>% cbind(clean_df)

compare_groups %>%
  group_by(gmm_clusters) %>%
  count(kmeans_clusters)
```

```
## # A tibble: 4 x 3
## # Groups:   gmm_clusters [2]
##   gmm_clusters kmeans_clusters     n
##          <dbl>           <int> <int>
## 1            0               1     1
## 2            0               2    39
## 3            1               1     5
## 4            1               2     4
```

```
compare_groups %>%
  filter(gmm_clusters ==0 & kmeans_clusters ==1)
```

```
##   state_name gmm_clusters kmeans_clusters t_slength  slength salary_real
## 1         MI            0               1 0.7755062 1.006312    2.138115
##      expend
## 1 0.4568995
```

Looks like the result of the kmeans and GMM are the same except for Michigan. Comparing the plots, this probably has to do with removing `t_slength` as a feature from the GMM, which Michigan had a high value for.

For internal validity I selected connectivity and the average silhouette to compare kmeans to GMM.

```
valid_GMM <- function(k){

  data <- data.matrix(clean_df %>% select(-t_slength))

  fit <- mvnormalmixEM(data, k = k)

  gmm_clust <- round(fit$posterior[,1])


  conn_k <- connectivity(Data = data, clusters = gmm_clust)

  avg_sil <- ifelse(k ==1, NA, mean(cluster::silhouette(gmm_clust, dist = dist(data))[,3]))

 return(c(conn_k, avg_sil))
}

valid_GMM_results <- valid_GMM(2)
```

```
## number of iterations= 28
```

```
valid_GMM_results
```

```
## [1] 17.2551587  0.4768958
```

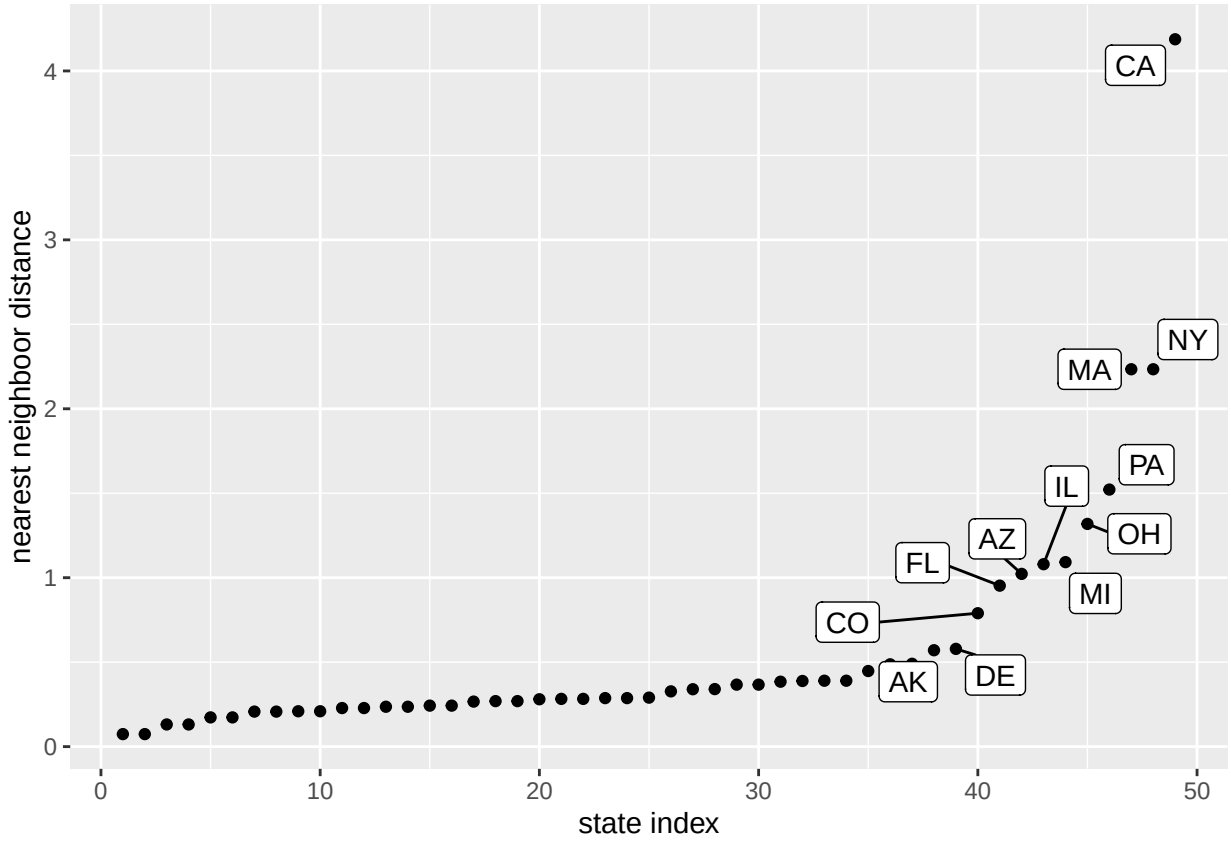
I got a connectivity value of 17.3 and an average silhouette of 0.5. These values suggest my GMM has slighlty worse internal validity than K-means.

7. Fit one additional partitioning technique of your choice

DBSCAN

I choose to fit **DBSCAN**. Here the key paramater is epislon, the neighborhood size. I read that you can guess epislon by plotting the nearest neighbor distances and finding the inflection point

```
dist_plot <- dist(clean_df)

nn_dist <- apply(data.matrix(dist_plot), 1, FUN = function(x) {min(x[x > 0])})

states_w_nn <- tibble(nn_dist = as.numeric(nn_dist),
       state = state_names$stateabv) %>%
  arrange(nn_dist) %>%
  mutate( n = row_number())

ggplot(data = states_w_nn, aes(x = n, y = nn_dist)) +
  geom_point() +
  geom_label_repel(data = states_w_nn %>% filter(nn_dist > 0.5), aes(label = state)) +
  labs(x = "state index", y = "nearest neighboor distance")
```



From this plot looks like epsilon 0.5 or epsilon of 1 are reasonable choices

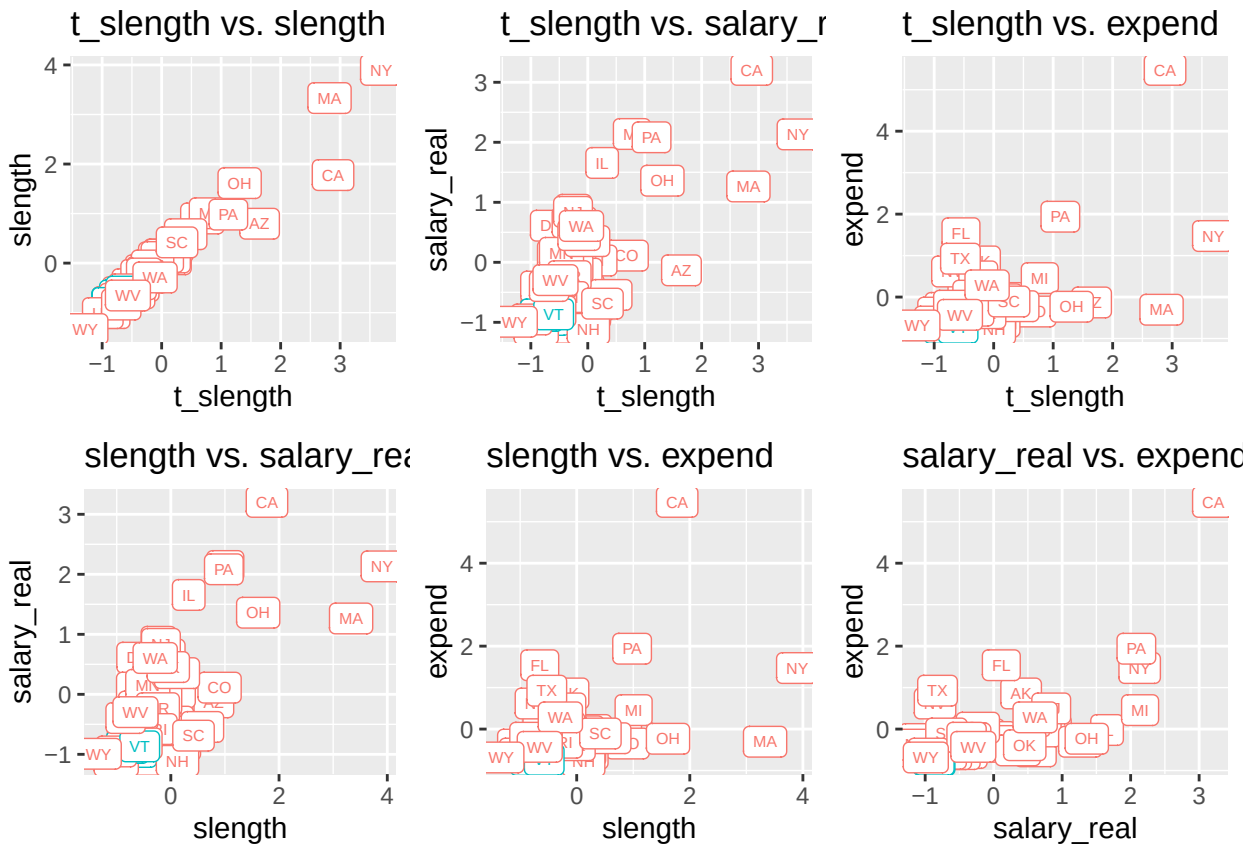
```
eps_list <- seq(0.25, 1.25, 0.25)
db_fits <- vector(mode = "list", length = length(eps_list))
db_plots <- vector(mode = "list", length = length(eps_list))
i <- 1
for (e in eps_list) {
```

```
  db_fits[[i]] <- dbscan::dbscan(data.matrix(clean_df), eps = e, minPts = 3)

  print(db_fits[[i]])
  db_plots[[i]] <- make_plot_list_w_clusters(clean_df, state_names, db_fits[[i]]$cluster)
  i <- i +1
}
```

```
## DBSCAN clustering for 49 objects.
## Parameters: eps = 0.25, minPts = 3
## The clustering contains 1 cluster(s) and 45 noise points.
##
##  0  1
## 45  4
##
## Available fields: cluster, eps, minPts
## DBSCAN clustering for 49 objects.
## Parameters: eps = 0.5, minPts = 3
## The clustering contains 2 cluster(s) and 14 noise points.
##
##  0  1  2
## 14 32  3
##
## Available fields: cluster, eps, minPts
## DBSCAN clustering for 49 objects.
## Parameters: eps = 0.75, minPts = 3
## The clustering contains 1 cluster(s) and 12 noise points.
##
##  0  1
## 12 37
##
## Available fields: cluster, eps, minPts
## DBSCAN clustering for 49 objects.
## Parameters: eps = 1, minPts = 3
## The clustering contains 1 cluster(s) and 8 noise points.
##
##  0  1
##  8 41
##
## Available fields: cluster, eps, minPts
## DBSCAN clustering for 49 objects.
## Parameters: eps = 1.25, minPts = 3
## The clustering contains 1 cluster(s) and 5 noise points.
##
##  0  1
##  5 44
##
## Available fields: cluster, eps, minPts
```

epsilon = 0.25

```
db_plots[[1]]
```



epsilon = 0.5

```
db_plots[[2]]
```

epsilon = 0.75

```
db_plots[[3]]
```

epsilon = 1

```
db_plots[[4]]
```

epsilon = 1.25

```
db_plots[[5]]
```

I choose epsilon = 1, under this parametrization DBCSAN has identified one main cluster of states and then a group of "noise" states that is visually appealing. Given the high variance in the corresponding component of the GMM, I think it makes sense to consider the outlier states as "noise" rather than a distinct cluster.

Therefore when validtating the DBSCAN, if could be argued that should just apply the measurees of validity to the one cluster and throw out the noise points. However to compare apples to apples, I'll treat the noise as a cluster.

```
valid_DBscan <- function(fit){

  data <- clean_df

  data$db_clust <- fit$cluster

  # data <- data %>%
  #   filter(db_clust ==1)

  clusters <- data$db_clust

  dist_matrix <- dist(data)

  data <- data.matrix(data)

  conn_k <- connectivity(Data = data, clusters = clusters)

  avg_sil <-mean(cluster::silhouette(clusters, dist = dist_matrix)[,3])
```

```
 return(c(conn_k, avg_sil))
}

valid_DBscan(db_fits[[4]])
```

```
## [1] 8.3059524 0.6148916
```

Even if we treat the noise points as a cluster, DBSCAN has lower connectivity and higher average silhouette than either kmeans or GMM, implying this has the method has the best internal validity.

9. Select a single validation strategy (e.g., compactness via min(WSS), average silhouette width, etc.), and calculate for all three algorithms.

As documented thoughout above, I used connectivity and average silhouette width to compare approaches. These measures both indicated that DBSCAN outperformed kmeans (which both beat GMM).

10. Discuss the validation output.

a. What can you take away from the fit?

Thinking about the results of all 3 approaches as a whole, I believe the results suggest that most state legislatures operate in similar fashion (expenditures, session length, salary) but there are 5-8 "outlier" states with large deviations in one or more variables. the outlier states aren't really that similar across the variables.

b. Which approach is optimal? And optimal at what value of k?

The DBSCAN apporach was optimmal, as it was well deisgned for this scenario where we had one big cluster of tightly grouped states in the feature space and then several noise outlier states (that were scattered about). The internal validation measures I chose confirm this.

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

I think DBSCAN would be the right approach here in a cluster vs. outlier scenario even if the "noise" cluster made the clustering less compact overall by the internal validation statistics. In fact, I think one could argue that only the DBSCAN identified clusters should be evaluated, as the algorithim sets noise points aside as outside the main data generating process.

Conversely, in a scenario where noise/outliers don't make sense (or I really wanted to definitively cluster all the data points) I would not choose DBSCAN even if the internal validation results were better.