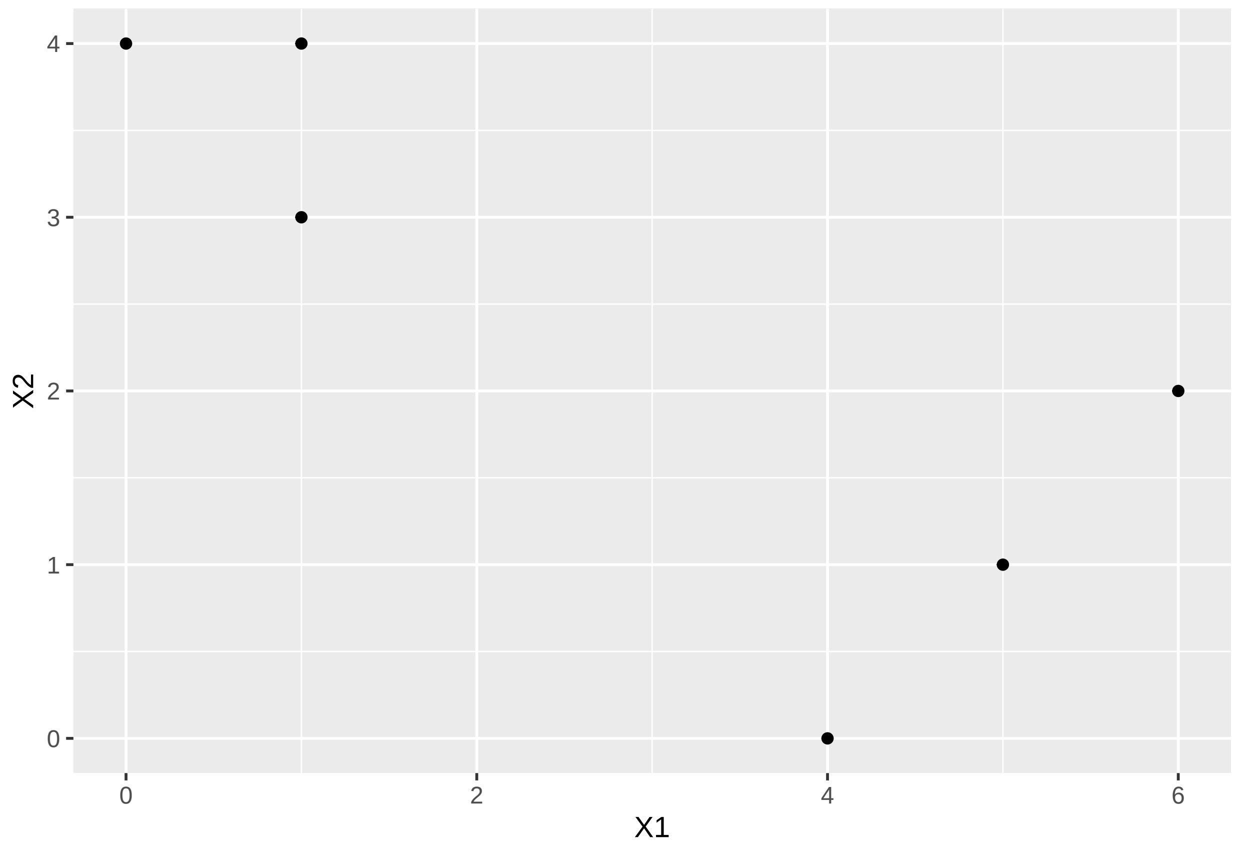
# Gill_Sarah_ML_PS4

Sarah Gill

2/28/2020

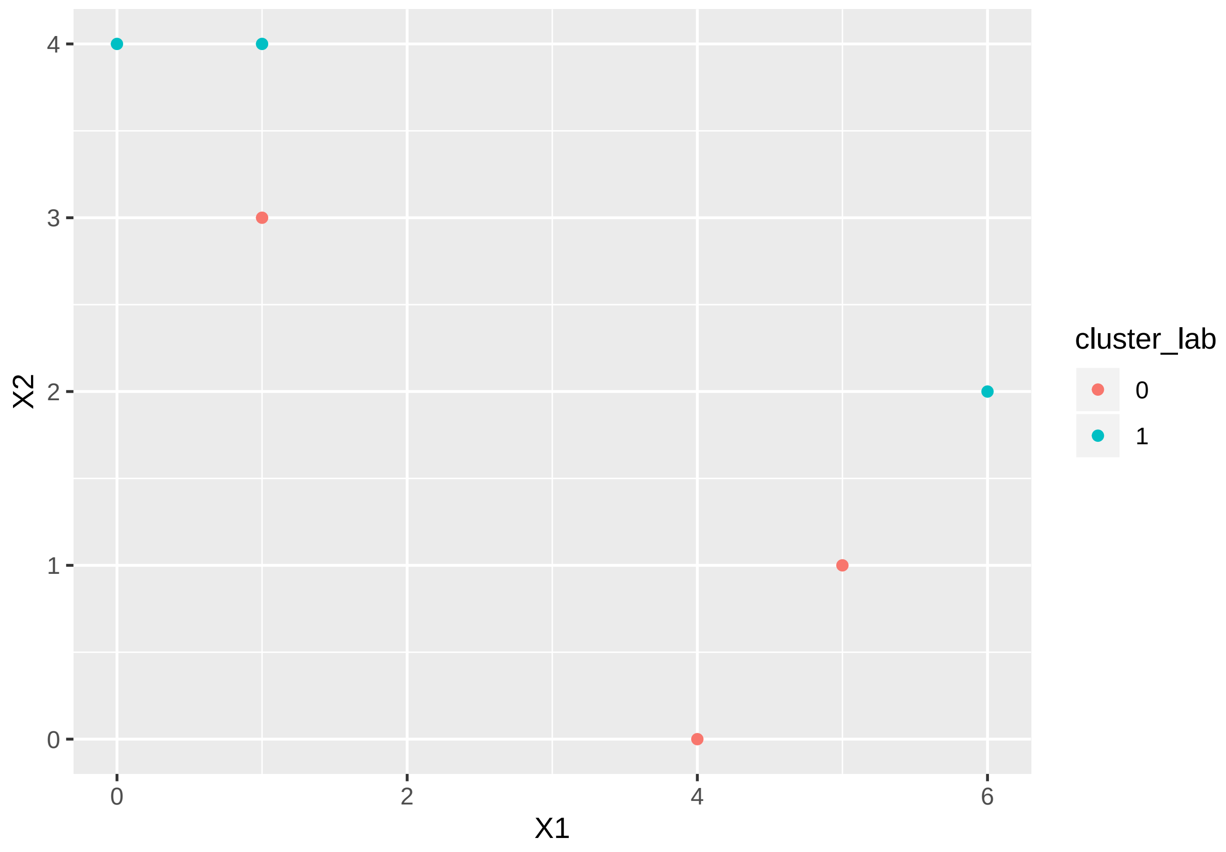
**Performing k-Means By Hand**

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

#1 Plot the observations
#plot(x)
df <- data.frame(x)

ggplot(df)+
  geom_point(aes(x=X1, y=X2))
```



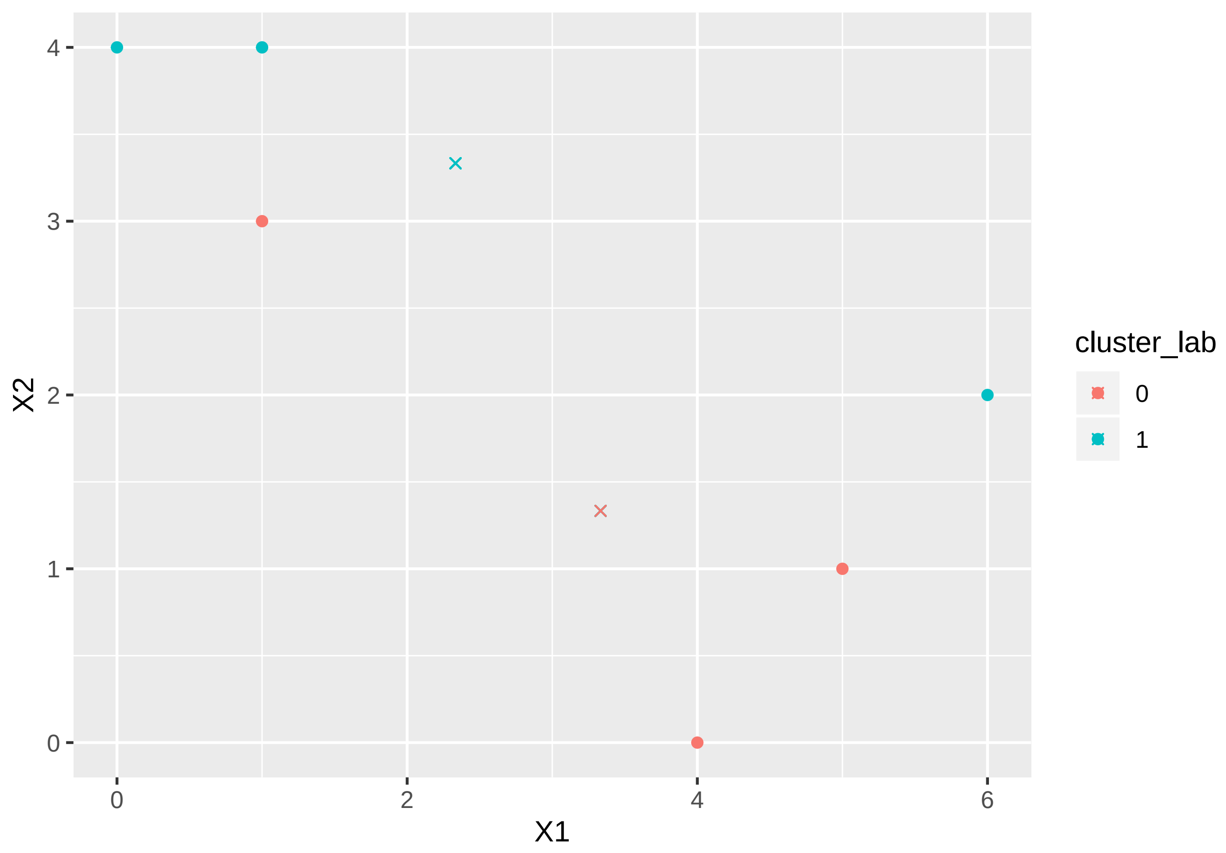
2. Randomly assign a cluster label to each observation. Report the cluster labels for each observation and plot the results with a different color for each cluster.

```r
set.seed(246810)
cluster_lab <- as.factor(sample(c(0,1), size = nrow(x), replace = TRUE))

df$cluster_lab = cluster_lab

ggplot(df)+
  geom_point(aes(x=X1, y=X2, color = cluster_lab))
```



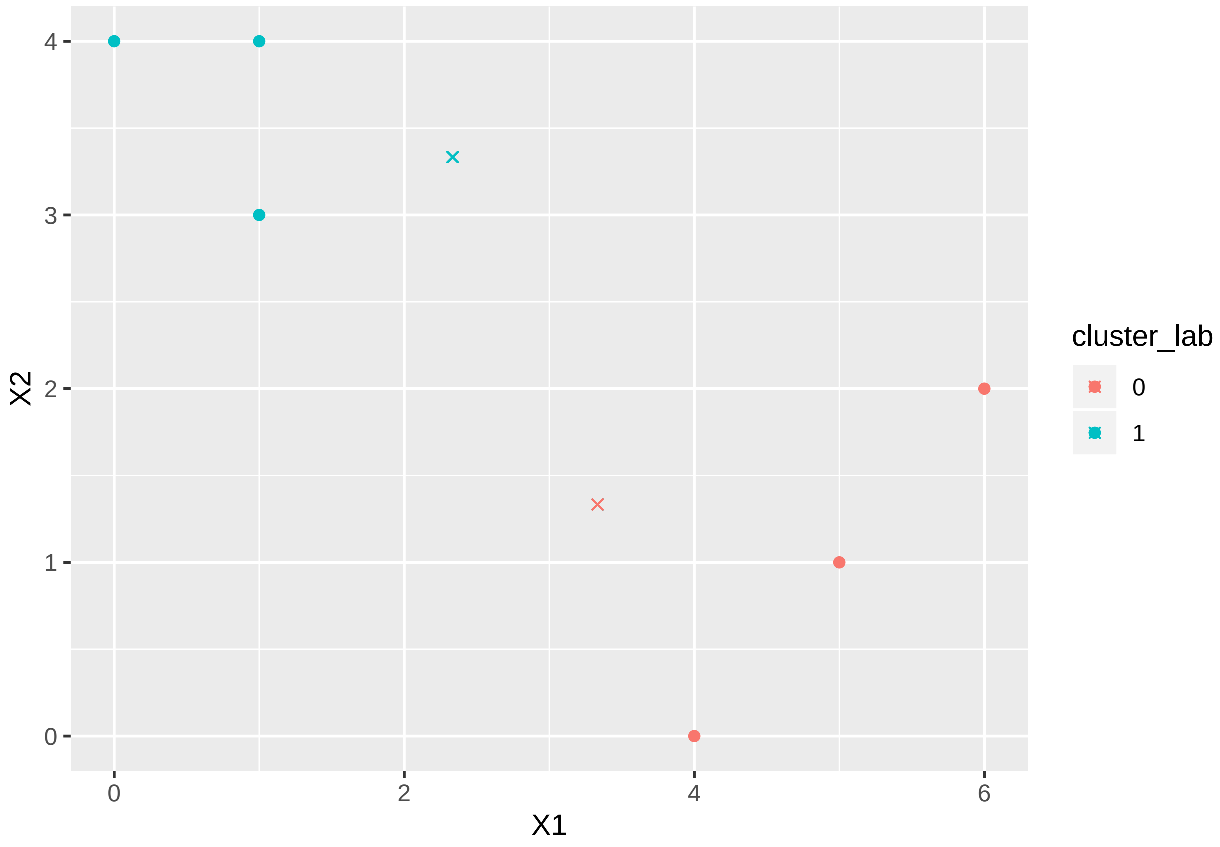
3.Compute the centroid for each cluster.

```r
centroid_0 <- c(mean(df$X1[df$cluster_lab==0]), mean(df$X2[df$cluster_lab==0]))
centroid_1 <- c(mean(df$X1[df$cluster_lab==1]), mean(df$X2[df$cluster_lab==1]))

ggplot(df)+
  geom_point(aes(x=X1, y=X2, color = cluster_lab))+
  geom_point(aes(x=centroid_0[1], y=centroid_0[2], color = cluster_lab), shape = 4) +
  geom_point(aes(x=centroid_1[1], y=centroid_1[2], color = cluster_lab[1]), shape = 4)
```

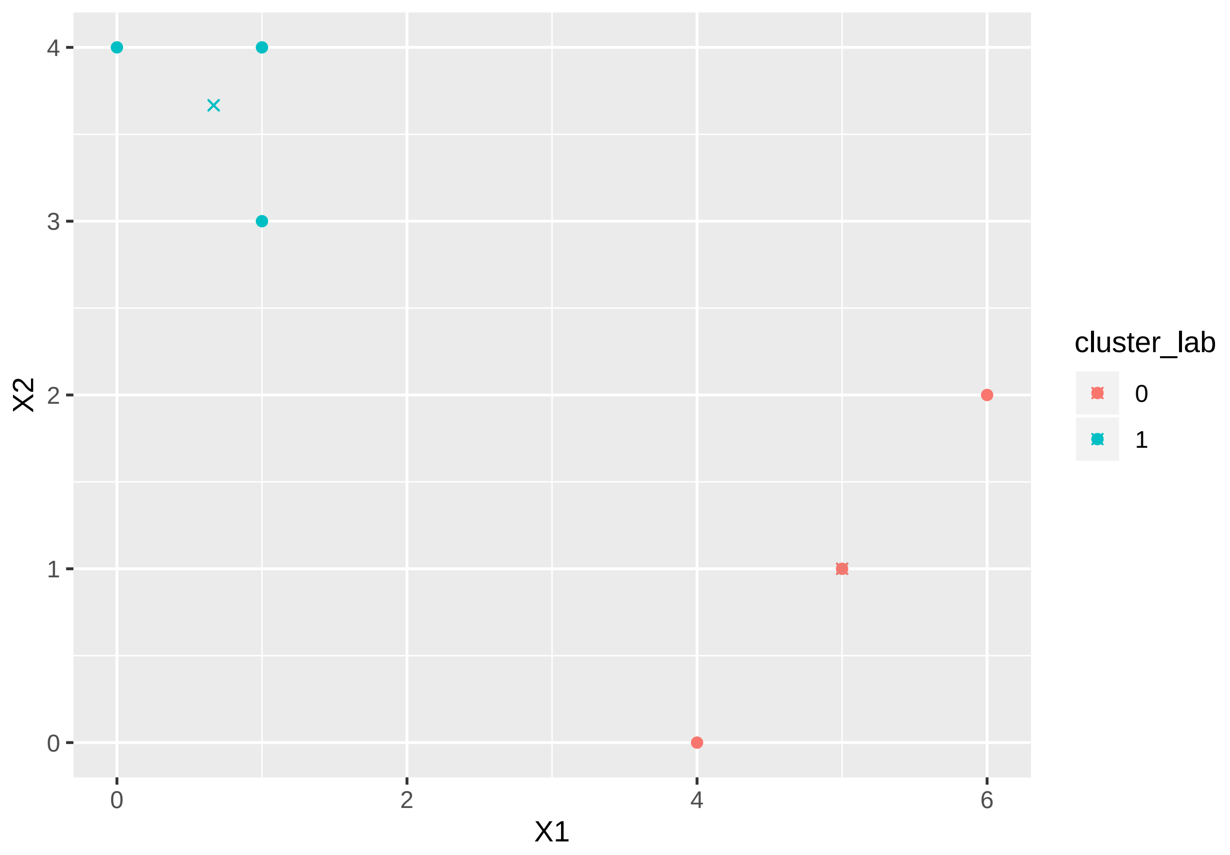


4. Assign each observation to the centroid to which it is closest, in terms of Euclidean distance. Report the cluster labels for each observation

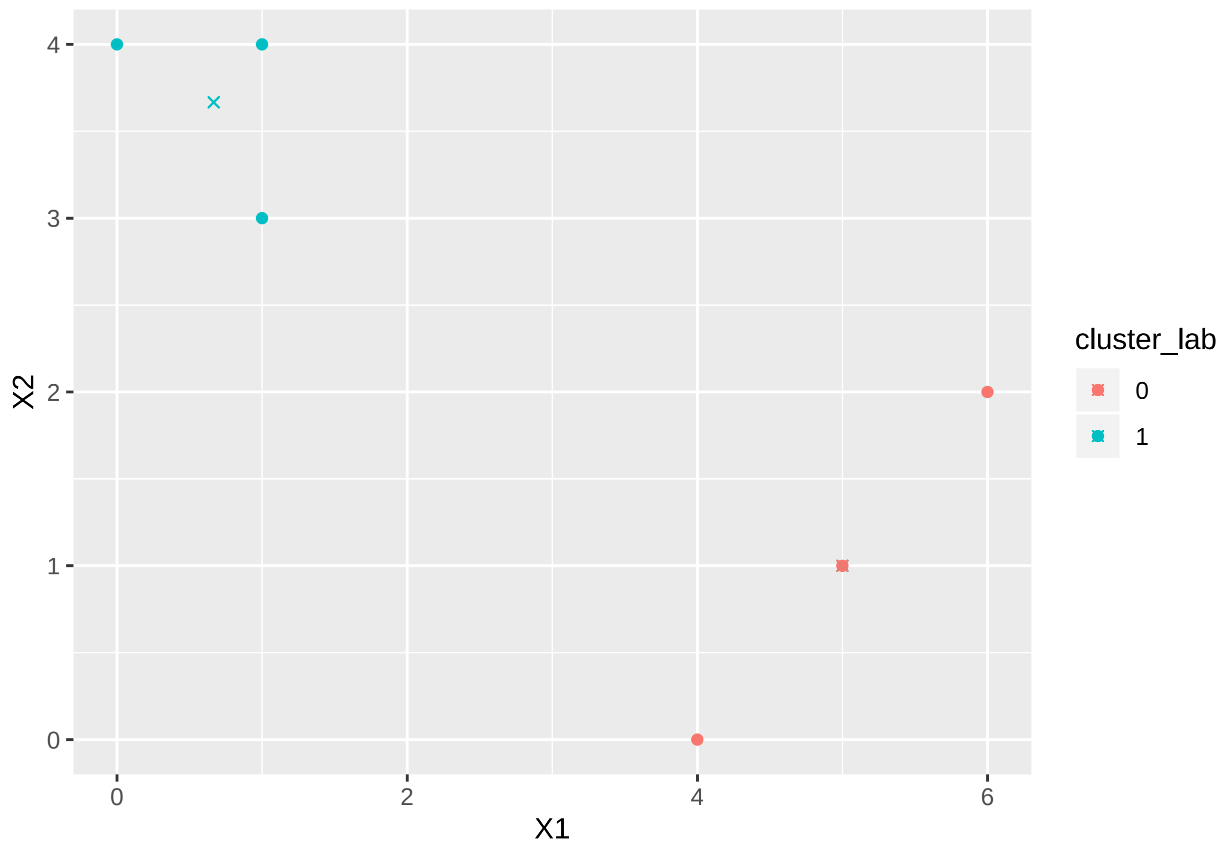
```
#Euclidean distance: d((x1,y1),(x2,y2) = sqrt((x1-x2)^2+(y1-y2)^2)

euclid_dist <- function (df, i, centroid_x){
  x1 <- df$X1[i]
  y1 <- df$X2[i]
  x2 <- centroid_x[1]
  y2 <- centroid_x[2]
  dist <- sqrt((x1-x2)^2+(y1-y2)^2)
  return(dist)
}



for (i in 1:6) {
if (euclid_dist(df, i ,centroid_0) < euclid_dist(df, i ,centroid_1)){
  df$cluster_lab[i] <- 0

} else {
  df$cluster_lab[i] <- 1
}
  }

ggplot(df)+
  geom_point(aes(x=X1, y=X2, color = cluster_lab))+
  geom_point(aes(x=centroid_0[1], y=centroid_0[2], color = cluster_lab), shape = 4) +
  geom_point(aes(x=centroid_1[1], y=centroid_1[2], color = cluster_lab[1]), shape = 4)
```



```
centroid_0 <- c(mean(df$X1[df$cluster_lab==0]), mean(df$X2[df$cluster_lab==0]))
centroid_1 <- c(mean(df$X1[df$cluster_lab==1]), mean(df$X2[df$cluster_lab==1]))

ggplot(df)+
  geom_point(aes(x=X1, y=X2, color = cluster_lab))+
  geom_point(aes(x=centroid_0[1], y=centroid_0[2], color = cluster_lab), shape = 4) +
  geom_point(aes(x=centroid_1[1], y=centroid_1[2], color = cluster_lab[1]), shape = 4)
```



```r
for (i in 1:6) {
if (euclid_dist(df, i ,centroid_0) < euclid_dist(df, i ,centroid_1)){
  df$cluster_lab[i] <- 0

} else {
  df$cluster_lab[i] <- 1
}
  }

ggplot(df)+
  geom_point(aes(x=X1, y=X2, color = cluster_lab))+
  geom_point(aes(x=centroid_0[1], y=centroid_0[2], color = cluster_lab), shape = 4) +
  geom_point(aes(x=centroid_1[1], y=centroid_1[2], color = cluster_lab[1]), shape = 4)
```



No change, wiht the randomization that happened this time around, it only took one itteration before cluster assignement and location of centroids stopped changeing

6. Plot See above

**Clustering State Legislative Professionalism**

1. Load the state legislative professionalism data

```r
load("Data and Codebook/legprof-components.v1.0.RData")
legprof <- x
#skim(legprof)
```

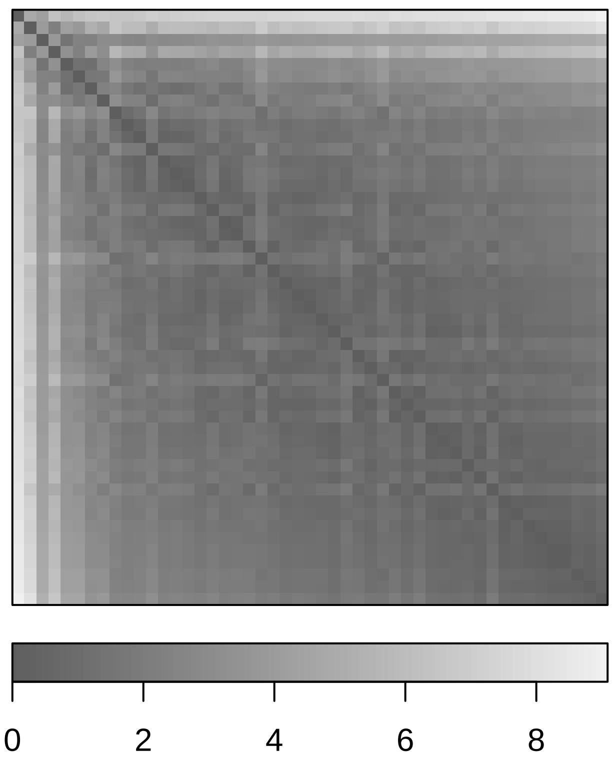
2. Munge the data

```r
lp_clean <- legprof %>%
  filter(sessid == "2009/10")  %>%
  select("expend", "salary_real", "t_slength", "slength") %>%
  na.omit() %>%
  scale()


state_name <- legprof %>%
  filter(sessid == "2009/10")  %>%
  na.omit() %>%
  select("fips", "stateabv", "state")

lp_for_reference <- legprof %>%
  filter(sessid == "2009/10")  %>%
  select("state", "stateabv", "expend", "salary_real", "t_slength", "slength") %>%
  na.omit()

#is.na(lp_clean) #check
#dist(lp_clean)
```

```
e. and anything else you think necessary to get this subset of data into workable form (hint: consider s
```

3. Diagnose clusterability

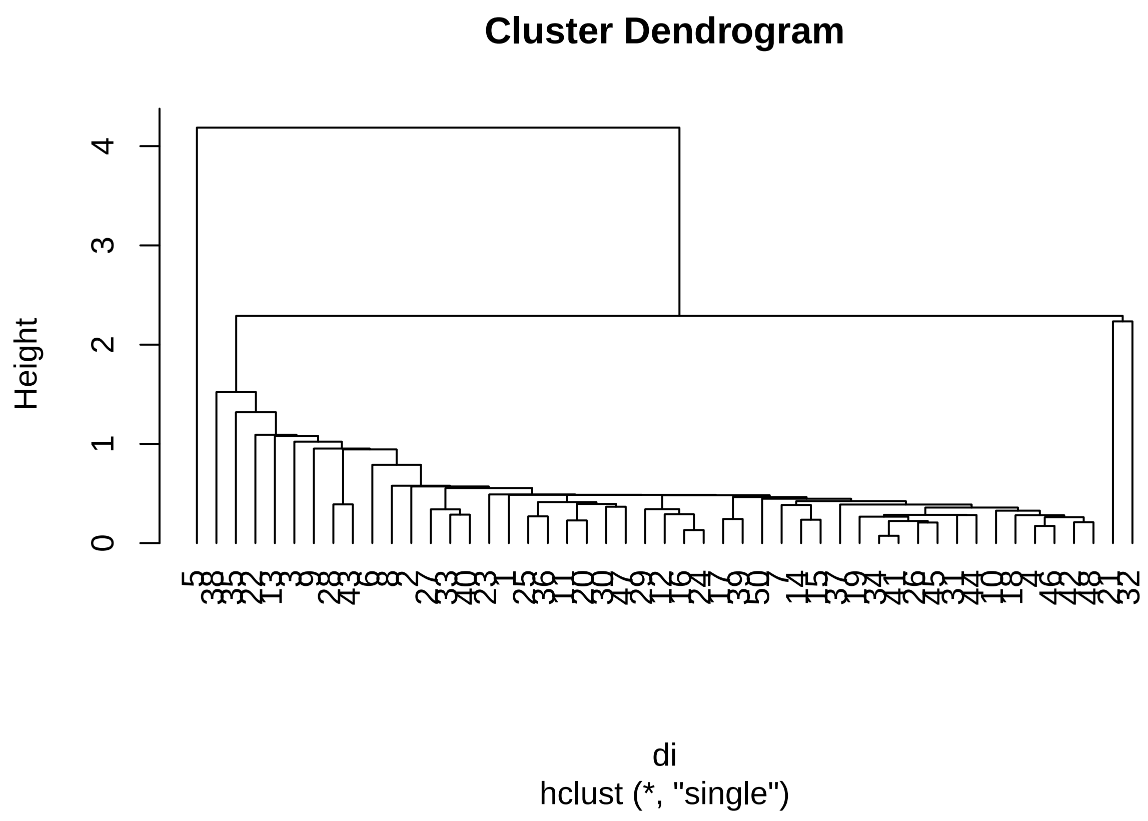
```r
di <- dist(lp_clean)
dissplot(di)
```



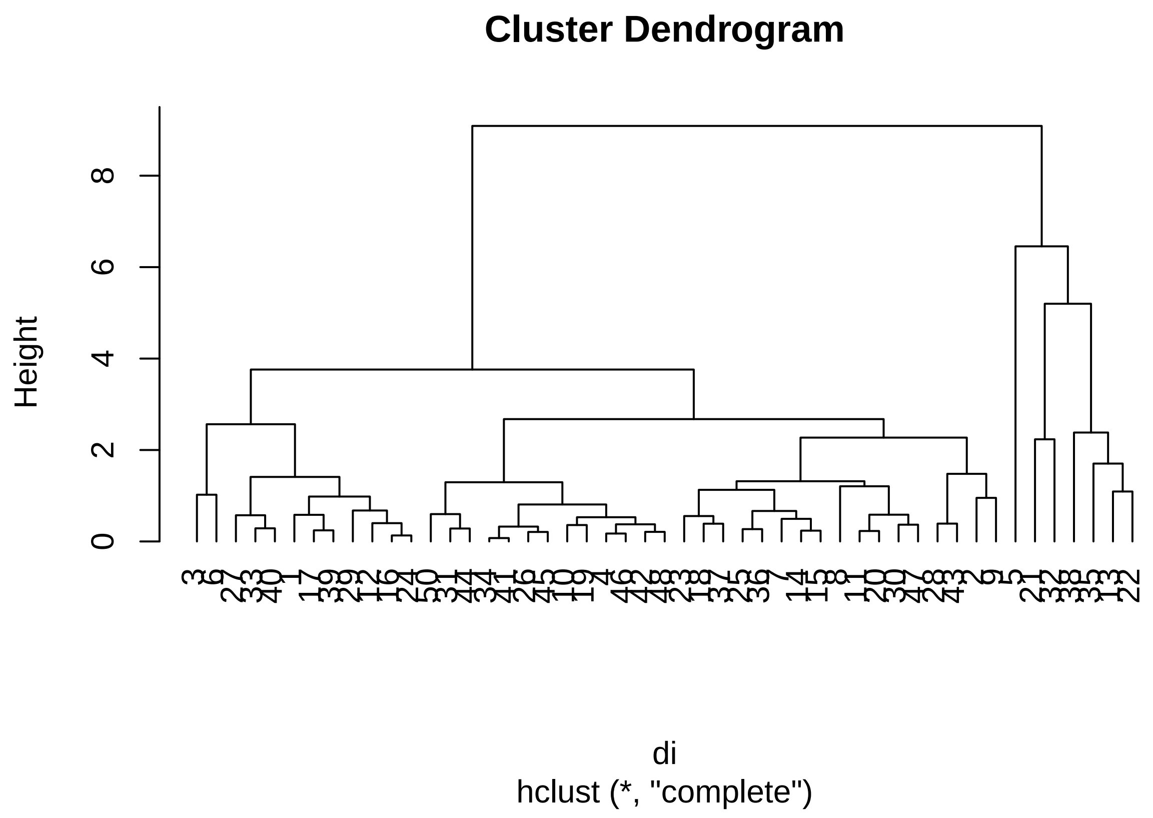
This plot does not support the presence of non-random structure in this data. The central diagonal is very narrow and there are few rectangles of particularly dark pigmentation. The picture is generally grey overall, without much visually apparent structure.

4. (5 points) Fit an **agglomerative hierarchical** clustering algorithm using any linkage method you prefer, to these data and present the results.

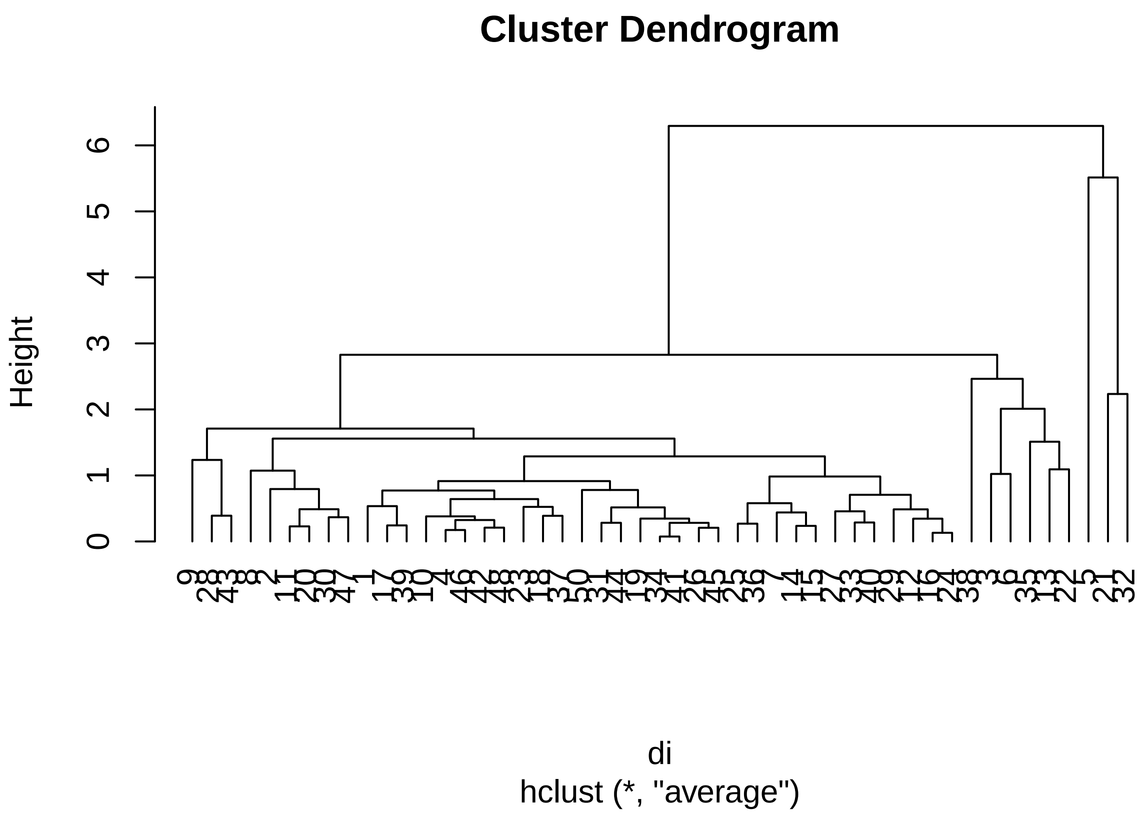
```r
hc_single <- hclust(di,
                    method = "single");plot(hc_single, hang = -1)
```



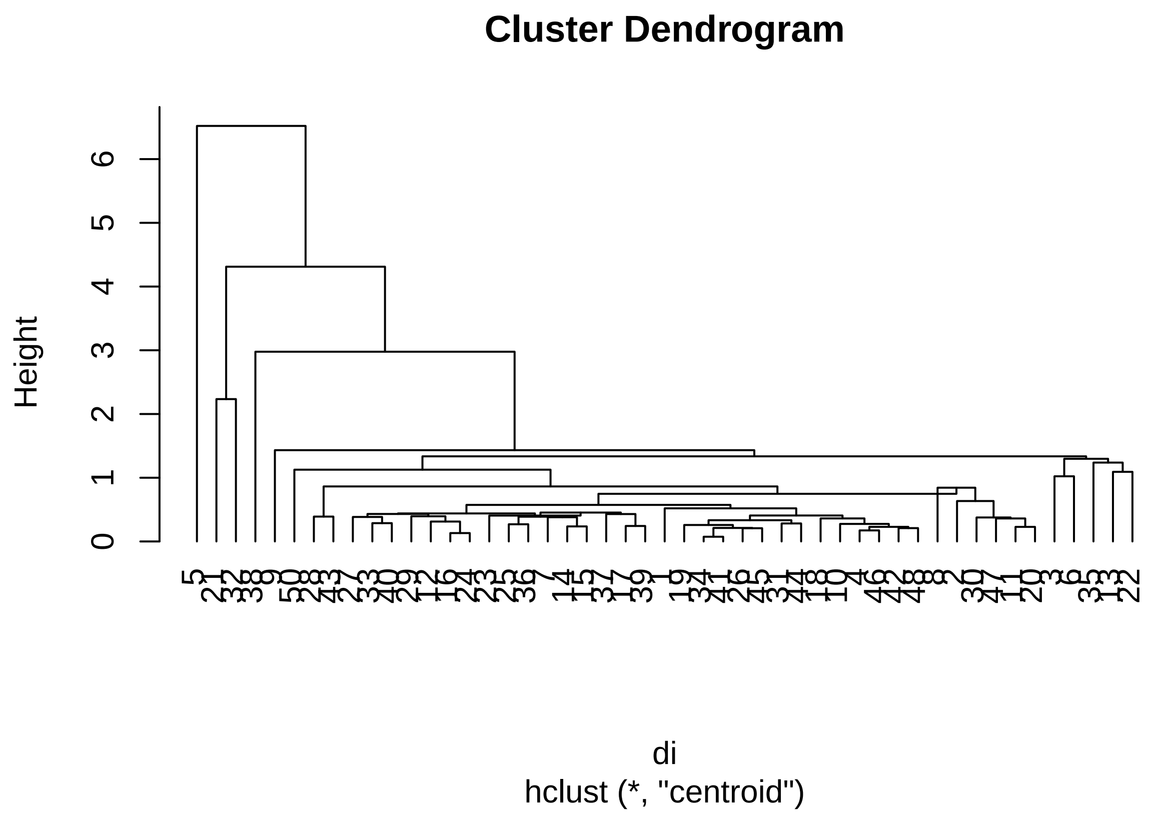
```r
hc_complete <- hclust(di,
                      method = "complete"); plot(hc_complete, hang = -1)
```



```
hc_average <- hclust(di,
                     method = "average"); plot(hc_average, hang = -1)
```



```r
hc_centroid <- hclust(di,
                      method = "centroid"); plot(hc_centroid, hang = -1)
```



At the k=2 level, all linkage methods except complete have one extremely small cluster and one large cluster (for instance at the point of thee being two clusters for linkage method = single, there is one state in a cluster and all other states in a cluster). This seems incorrect, however the test of clusterability did not support a high level of clusterability. Looking at linkage method = complete we see a lot of pair-wise clustering early on.

```r
hc <- cutree(hc_complete,
             k = 2)

hc_results <- data.frame(hc)
hc_results$state <- lp_for_reference$state

c1 <- hc_results%>%
  filter(hc_results$hc == 1)%>%
  select("state")

as.list(c1)
```

```
## $state
##  [1] "Alabama"       "Alaska"        "Arizona"       "Arkansas"
##  [5] "Colorado"      "Connecticut"   "Delaware"      "Florida"
##  [9] "Georgia"       "Hawaii"        "Idaho"         "Indiana"
## [13] "Iowa"          "Kansas"        "Kentucky"      "Louisiana"
## [17] "Maine"         "Maryland"      "Minnesota"     "Mississippi"
## [21] "Missouri"      "Montana"       "Nebraska"      "Nevada"
## [25] "New Hampshire" "New Jersey"    "New Mexico"    "North Carolina"
## [29] "North Dakota"  "Oklahoma"      "Oregon"        "Rhode Island"
## [33] "South Carolina" "South Dakota"   "Tennessee"      "Texas"
## [37] "Utah"           "Vermont"        "Virginia"       "Washington"
## [41] "West Virginia"  "Wyoming"
```

```
c2 <- hc_results%>%
  filter(hc_results$hc == 2)%>%
  select("state")

as.list(c2)
```

```
## $state
## [1] "California"    "Illinois"      "Massachusetts" "Michigan"
## [5] "New York"      "Ohio"          "Pennsylvania"
```

As mentioned above, if we use complete as our linkage method and assume that there are 2 clusters, then we see one large cluster and one small one (although still a reasonable size). The small cluster has mostly populous, high cost of living states, but not all of them have above average population density or cost of living. The larger cluster is too large to characterize without more information

5. (5 points) Fit a **k-means** algorithm to these data and present the results.

```
set.seed(246810)

kmeans <- kmeans(lp_clean[ ,2],
                 centers = 2, #k
                 nstart = 15) #start 15 times


kmeans_results <- data.frame(kmeans$cluster)
kmeans_results$state <- lp_for_reference$state

c1 <- kmeans_results%>%
  filter(kmeans$cluster == 1)%>%
  select("state")

as.list(c1)
```

```
## $state
##  [1] "California"    "Delaware"      "Hawaii"        "Illinois"
##  [5] "Maryland"      "Massachusetts" "Michigan"      "New Jersey"
##  [9] "New York"      "Ohio"          "Pennsylvania"  "Washington"
```

```
c2 <- kmeans_results%>%
  filter(kmeans$cluster == 2)%>%
  select("state")

as.list(c2)
```

```
## $state
##  [1] "Alabama"        "Alaska"         "Arizona"        "Arkansas"
##  [5] "Colorado"       "Connecticut"    "Florida"        "Georgia"
##  [9] "Idaho"          "Indiana"        "Iowa"           "Kansas"
## [13] "Kentucky"       "Louisiana"      "Maine"          "Minnesota"
## [17] "Mississippi"    "Missouri"       "Montana"        "Nebraska"
## [21] "Nevada"         "New Hampshire"  "New Mexico"     "North Carolina"
## [25] "North Dakota"   "Oklahoma"       "Oregon"         "Rhode Island"
## [29] "South Carolina" "South Dakota"   "Tennessee"      "Texas"
## [33] "Utah"           "Vermont"        "Virginia"       "West Virginia"
## [37] "Wyoming"
```

In one cluster we see many, highly populous, traditionally blue states that also have a high cost of liveing, such as California and New York. However, this cluster also has Ohio which does not immediately appear related to the others, but has been blue and purple in the past. In the other cluster we see some Southern states but also Southwestern and Eastern states too. Many are traditionally red states, but not all (for instance CT and CO are often blue states). Many of the states here have below average costs of living (but not all). The second cluster also has more observations in it. I do not know much, a priori, about legislative professionalism, so cannot easily judge the results given by this kmeans clustering

6. Fit a Gaussian mixture model via the EM algorithm*to these data and present the results.

```
library(mixtools) #journal of statistical software to accompony this
```

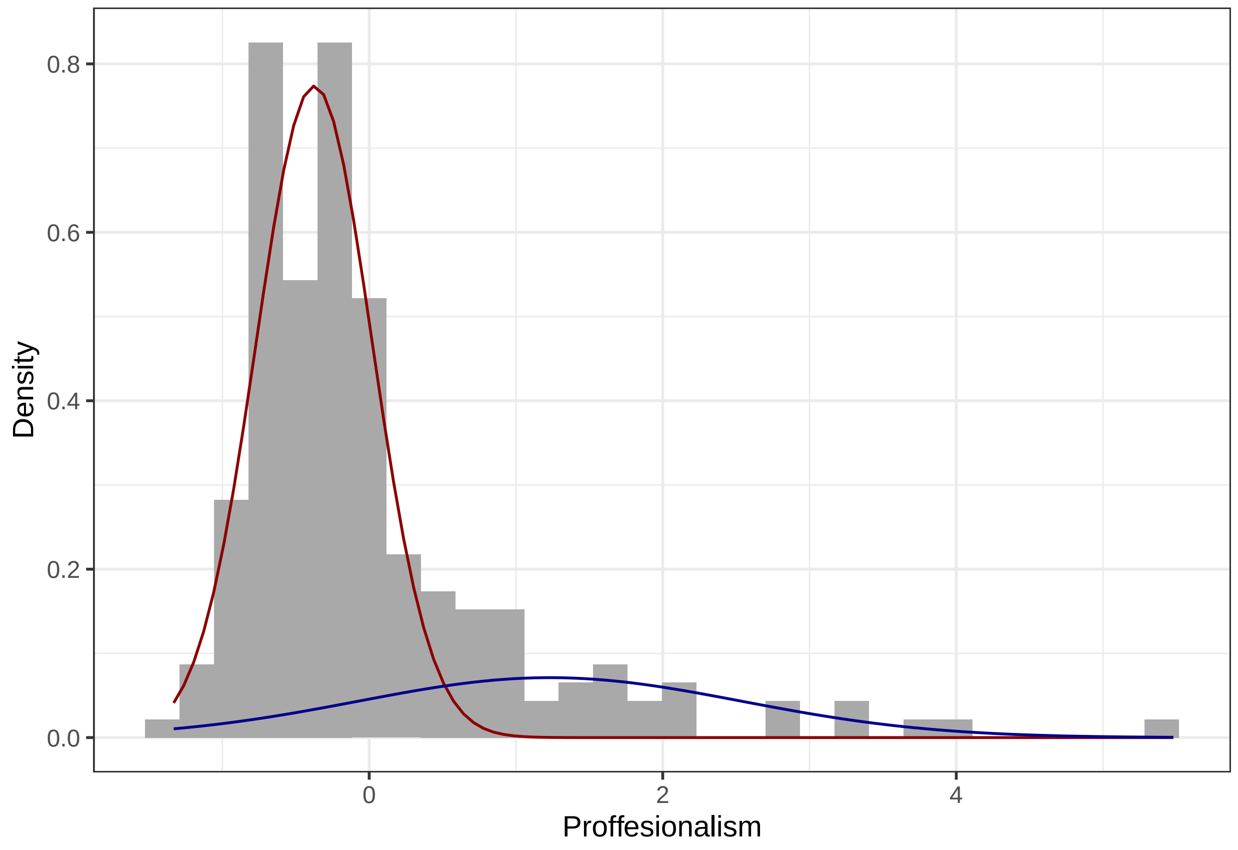
```
## mixtools package, version 1.2.0, Released 2020-02-05
## This package is based upon work supported by the National Science Foundation under Grant No. SES-0518
```

```
library(plotGMM)
set.seed(7355) # set seed for iterations to return to same place

gmm1 <- normalmixEM(lp_clean, k = 2)
```

```
## number of iterations= 40
```

```
ggplot(data.frame(x = gmm1$x)) +
  geom_histogram(aes(x, ..density..), fill = "darkgray") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[1], gmm1$sigma[1], lam = gmm1$lambda[1]),
                colour = "darkred") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[2], gmm1$sigma[2], lam = gmm1$lambda[2]),
                colour = "darkblue") +
  xlab("Proffesionalism") +
  ylab("Density") +
  theme_bw()
```

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

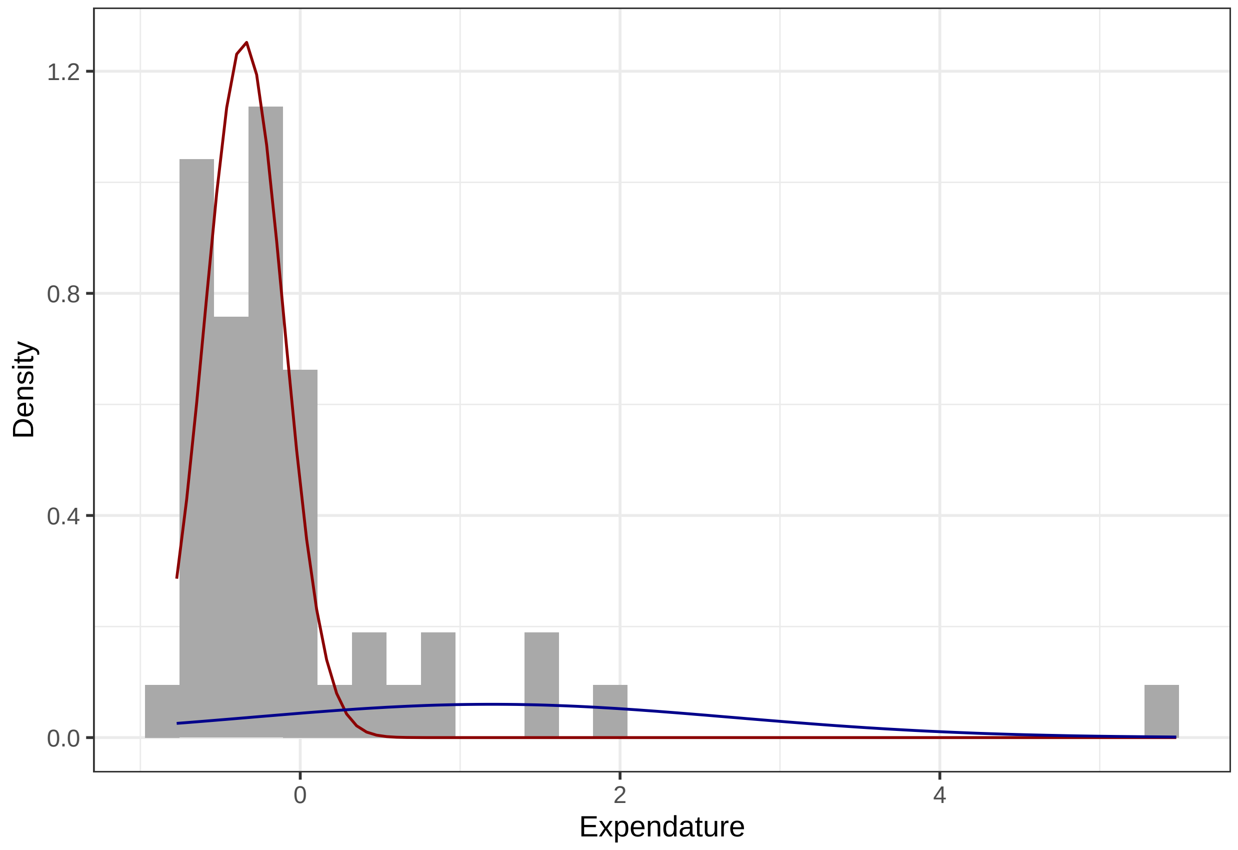


```
lp <- data.frame(lp_clean)
gmm_e <- normalmixEM(lp$expend, k = 2) # fit the GMM using EM and 2 cluters
```

```
## number of iterations= 15
```

```
ggplot(data.frame(x = gmm_e$x)) +
  geom_histogram(aes(x, ..density..), fill = "darkgray") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm_e$mu[1], gmm_e$sigma[1], lam = gmm_e$lambda[1]),
                colour = "darkred") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm_e$mu[2], gmm_e$sigma[2], lam = gmm_e$lambda[2]),
                colour = "darkblue") +
  xlab("Expendature") +
  ylab("Density") +
  theme_bw()
```

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

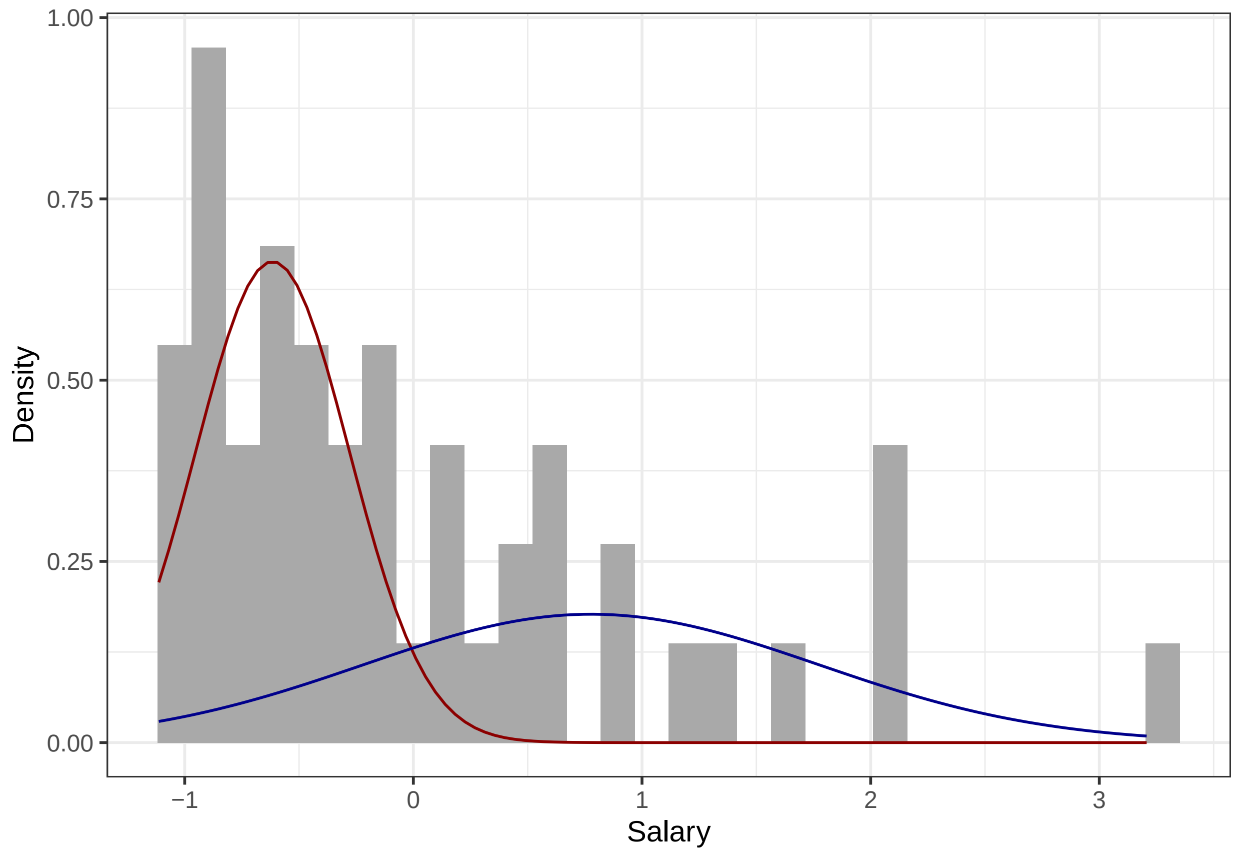


```
gmm_s <- normalmixEM(lp$salary_real, k = 2) # fit the GMM using EM and 2 cluters
```

```
## number of iterations= 56
```

```
ggplot(data.frame(x = gmm_s$x)) +
  geom_histogram(aes(x, ..density..), fill = "darkgray") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm_s$mu[1], gmm_s$sigma[1], lam = gmm_s$lambda[1]),
                colour = "darkred") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm_s$mu[2], gmm_s$sigma[2], lam = gmm_s$lambda[2]),
                colour = "darkblue") +
  xlab("Salary") +
  ylab("Density") +
  theme_bw()
```

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



```r
posterior_e <- data.frame(cbind(gmm_e$x, gmm_e$posterior))
rownames(posterior_e) <- lp_for_reference$state
#round(head(posterior_e, 10), 3)

posterior_e$component <- ifelse(posterior_e$comp.1 > 0.3, 1, 2) #maually, say prob > 30% -> cluster 1
#table(posterior_e$component)

posterior_s <- data.frame(cbind(gmm_s$x, gmm_s$posterior))
rownames(posterior_s) <- lp_for_reference$state
#round(head(posterior_s, 10), 3)

posterior_s$component <- ifelse(posterior_s$comp.1 > 0.3, 1, 2) #maually, say prob > 30% -> cluster 1
table(posterior_s$component)
```

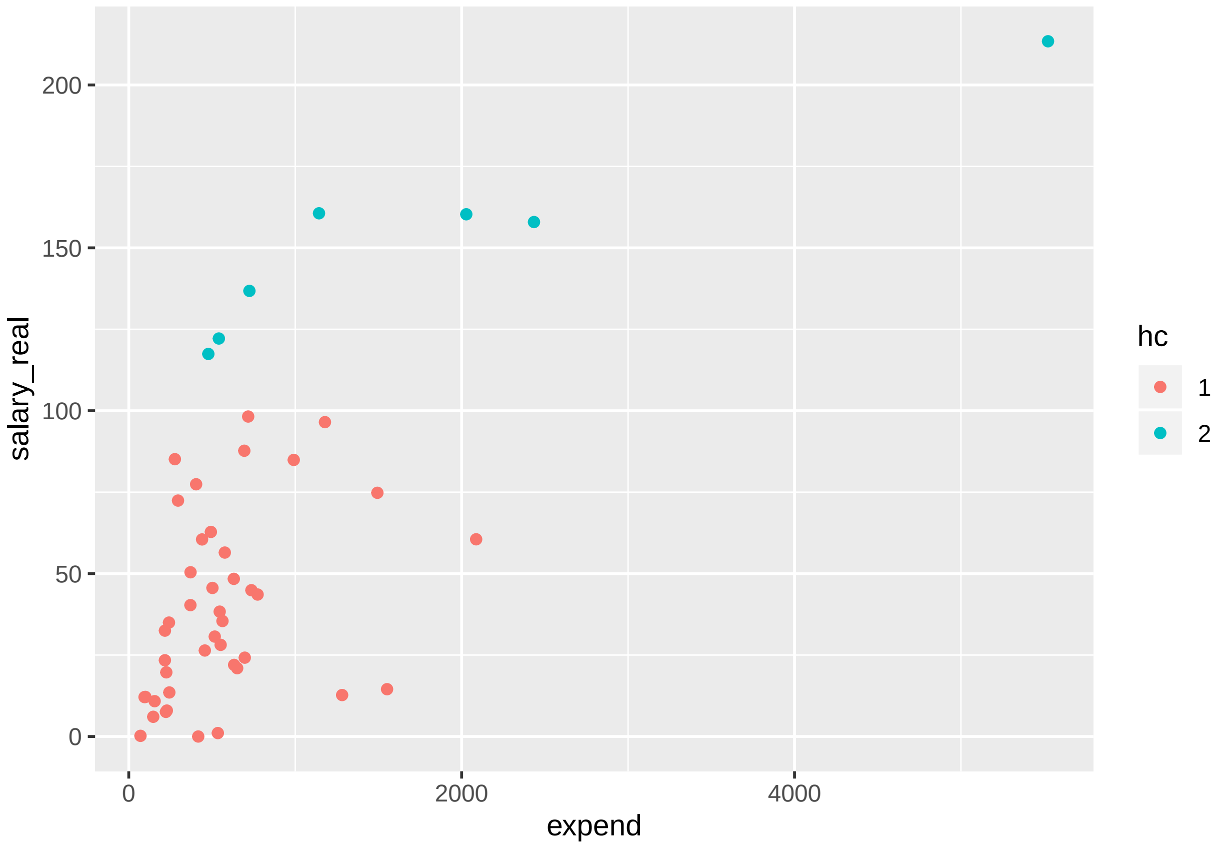
```
##
##  1  2
## 33 16
```

We see one high density cluster centered at negative normalized overal professionalism and a broad, low density cluster centered at below 2 normalized overal professionalism with tails expanding across the spread of the data. So we have one tight cluster and one broad one.

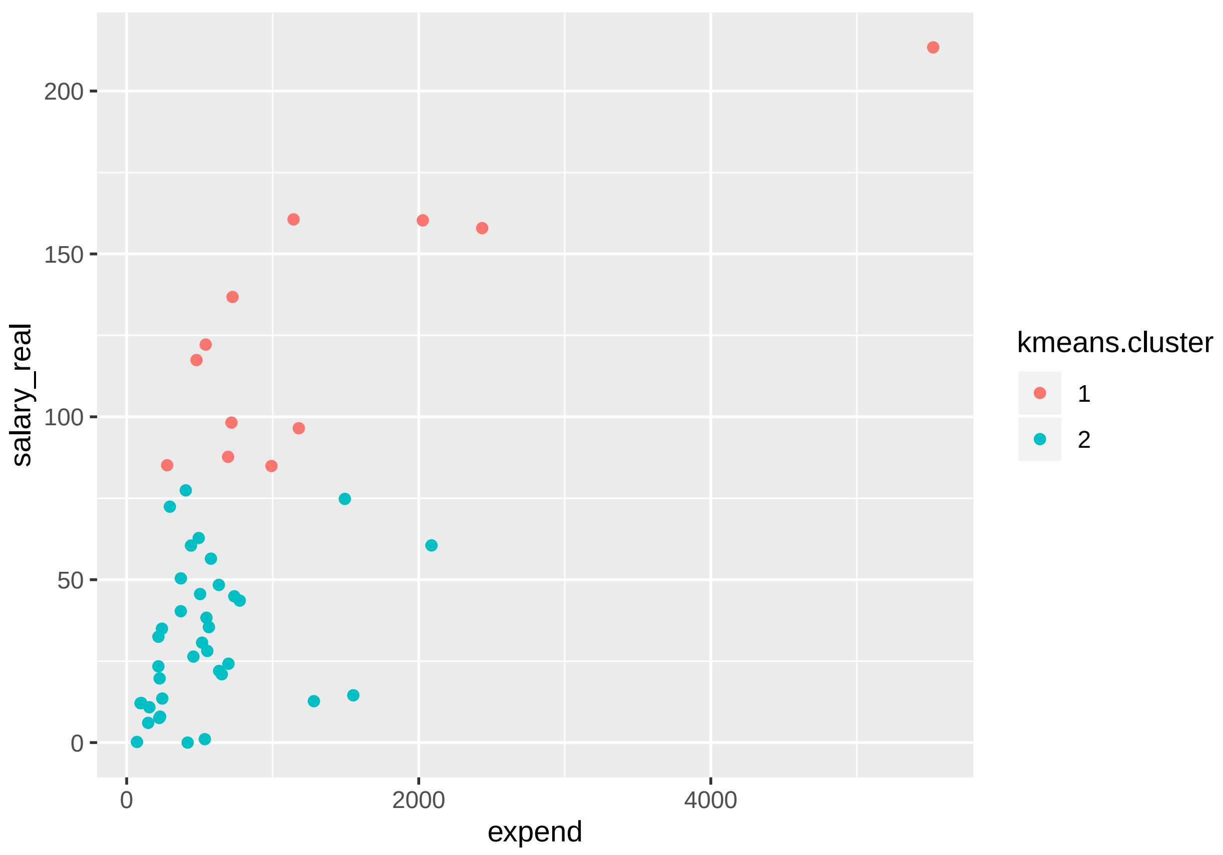
If we look specifically at different features, (for instance expenditures and salaries) we get different clustering. However, the structure of one dense and high frequency cluster and one broad and low frequency cluster are maintained

7. (15 points) Compare output of all in visually useful, simple ways

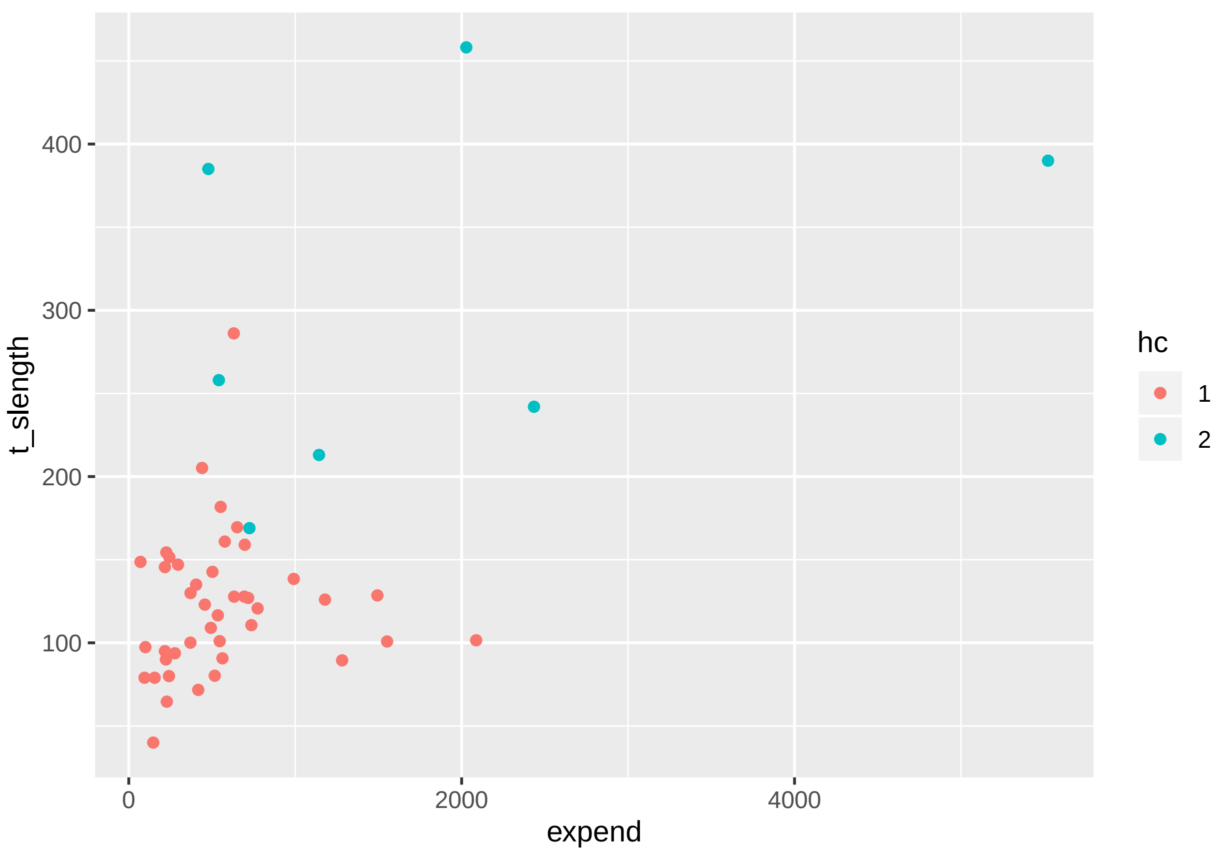
```r
#lp <- data.frame(lp_clean)
df <- merge(lp_for_reference, hc_results, on = state)
df <- merge(df, kmeans_results, on = state)
#df <- merge(df, gmm_results, on = state)

df$hc <- as.factor(df$hc)
df$kmeans.cluster <- as.factor(df$kmeans.cluster)



ggplot(data = df)+
  geom_point(aes(x = expend, y = salary_real, color = hc))
```



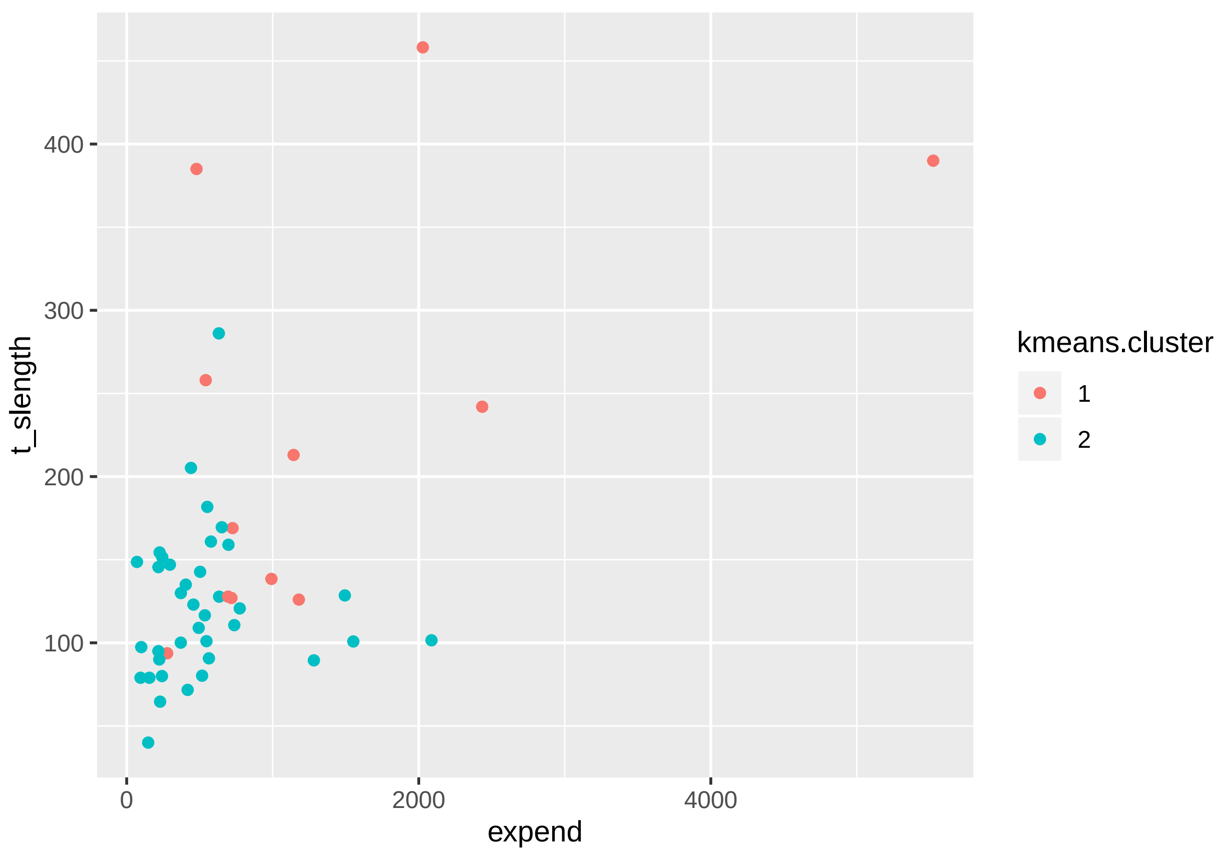
```r
ggplot(data = df)+
  geom_point(aes(x = expend, y = salary_real, color = kmeans.cluster))
```



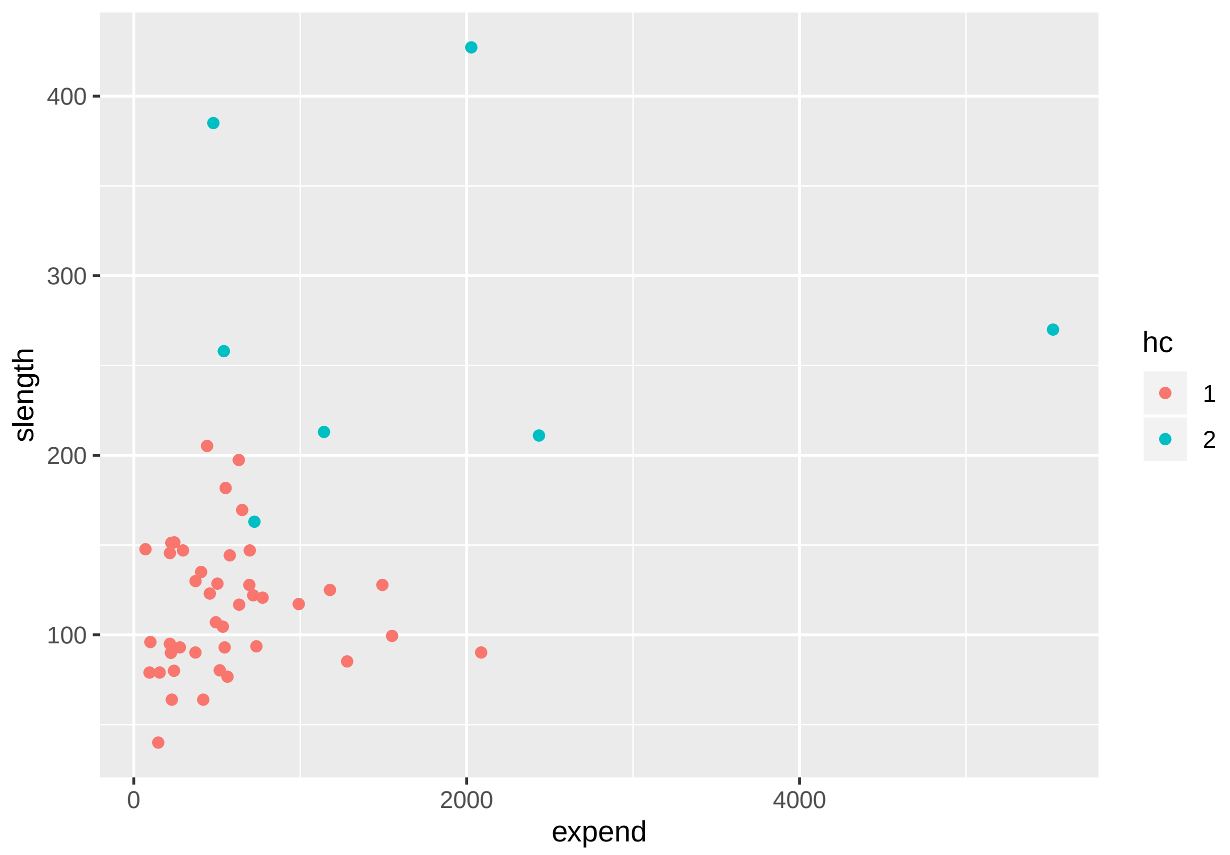
```
ggplot(data = df)+
  geom_point(aes(x = expend, y = t_slength, color = hc))
```



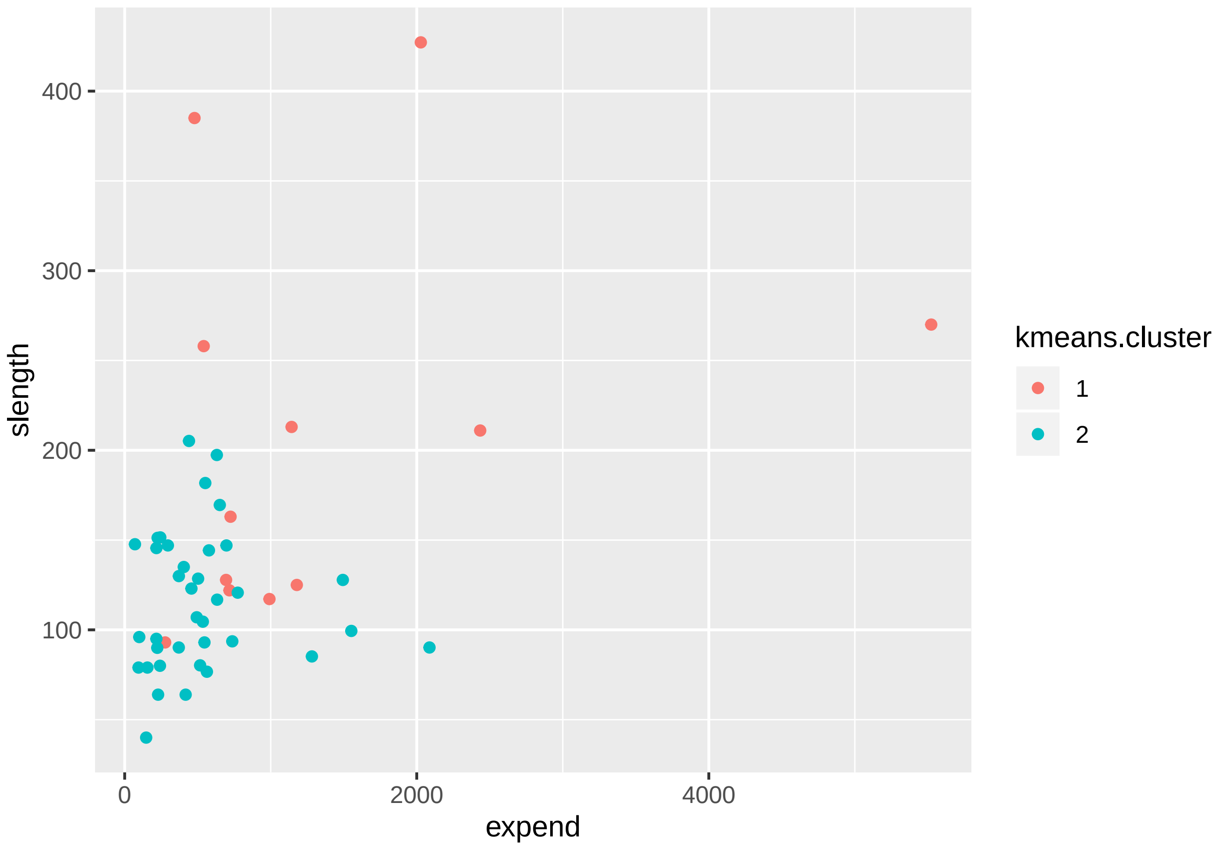
```
ggplot(data = df)+
  geom_point(aes(x = expend, y = t_slength, color = kmeans.cluster))
```



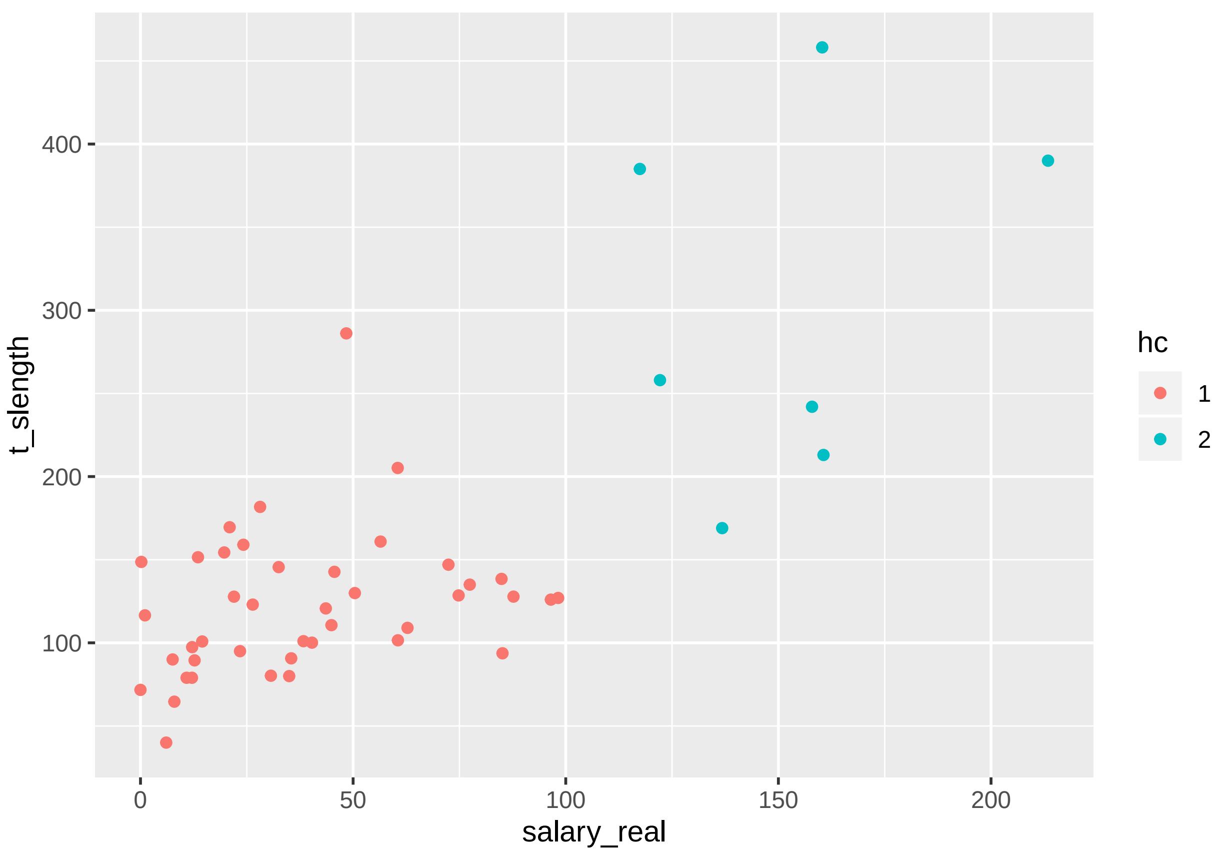
```r
ggplot(data = df)+
  geom_point(aes(x = expend, y = slength, color = hc))
```



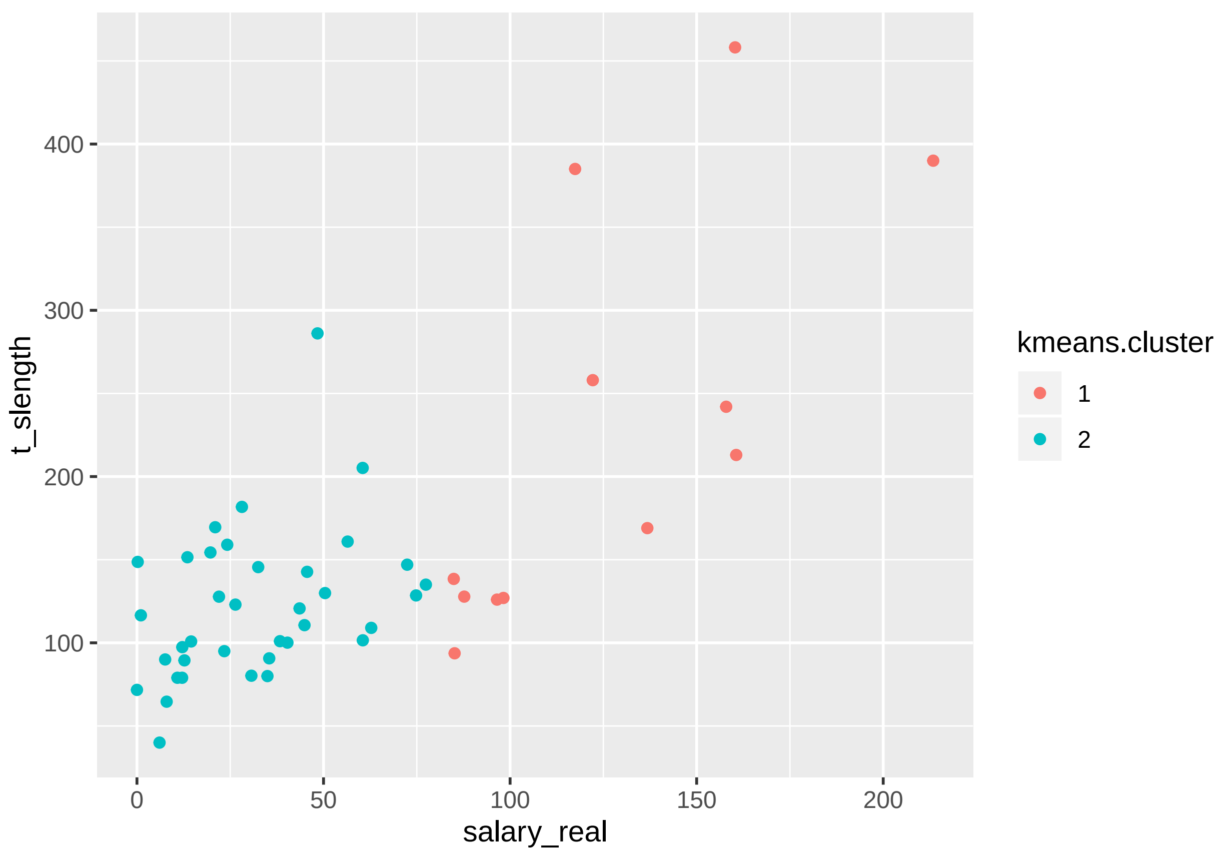
```r
ggplot(data = df)+
  geom_point(aes(x = expend, y = slength, color = kmeans.cluster))
```



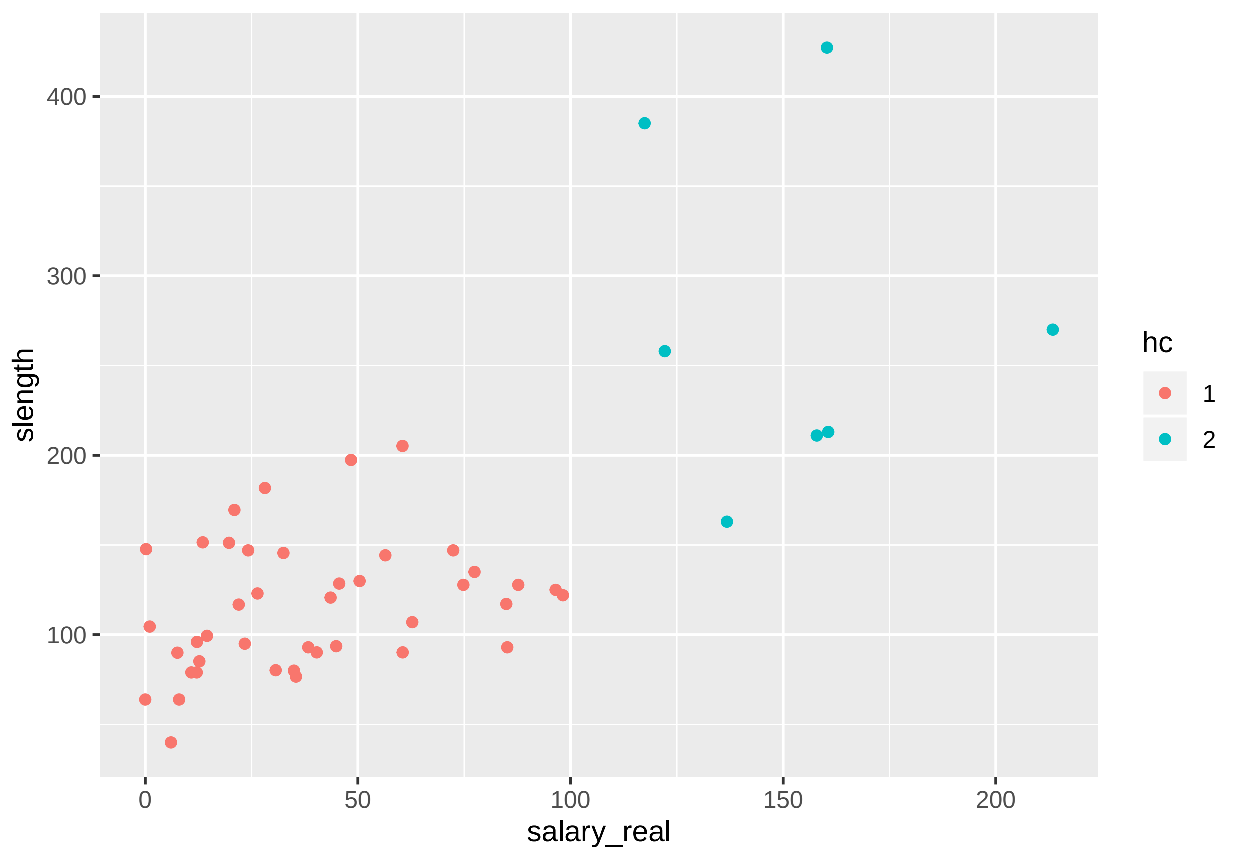
```r
ggplot(data = df)+
  geom_point(aes(x = salary_real, y = t_slength, color = hc))
```



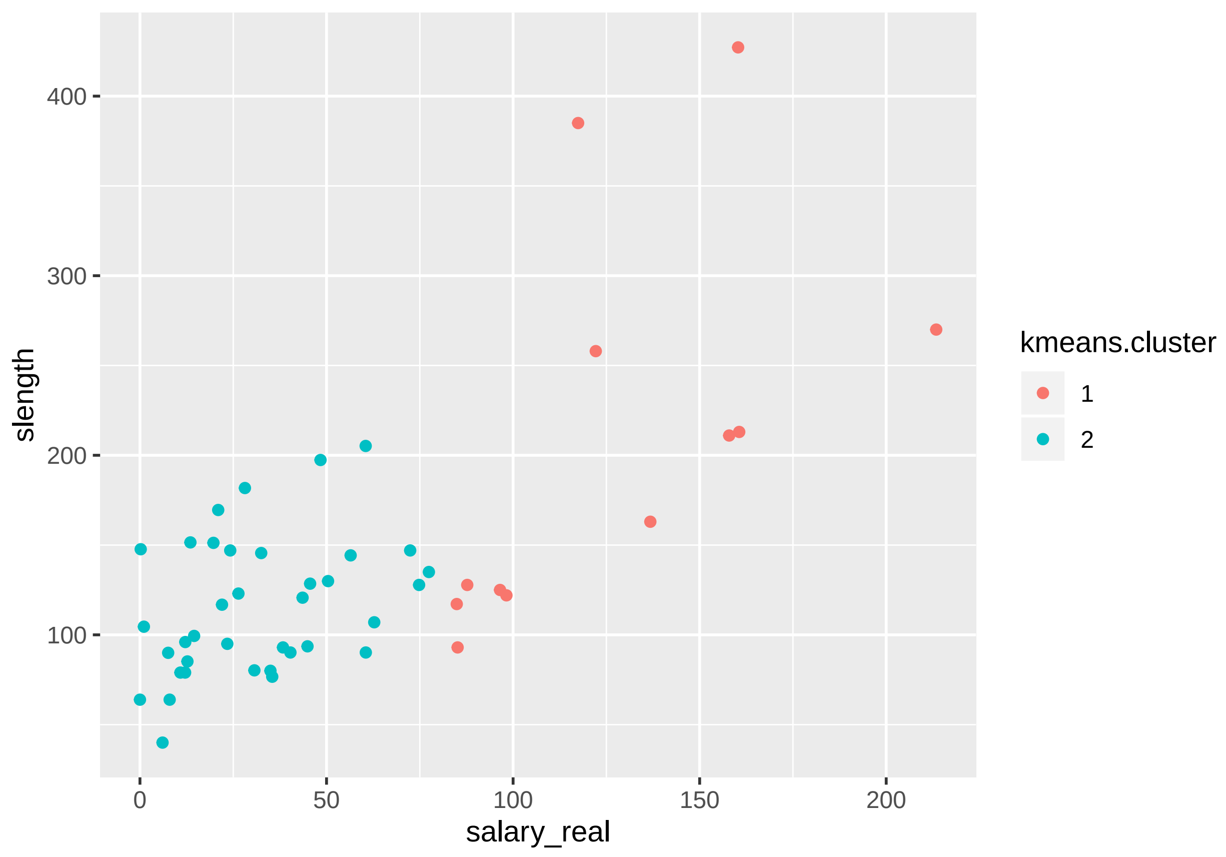
```r
ggplot(data = df)+
  geom_point(aes(x = salary_real, y = t_slength, color = kmeans.cluster))
```



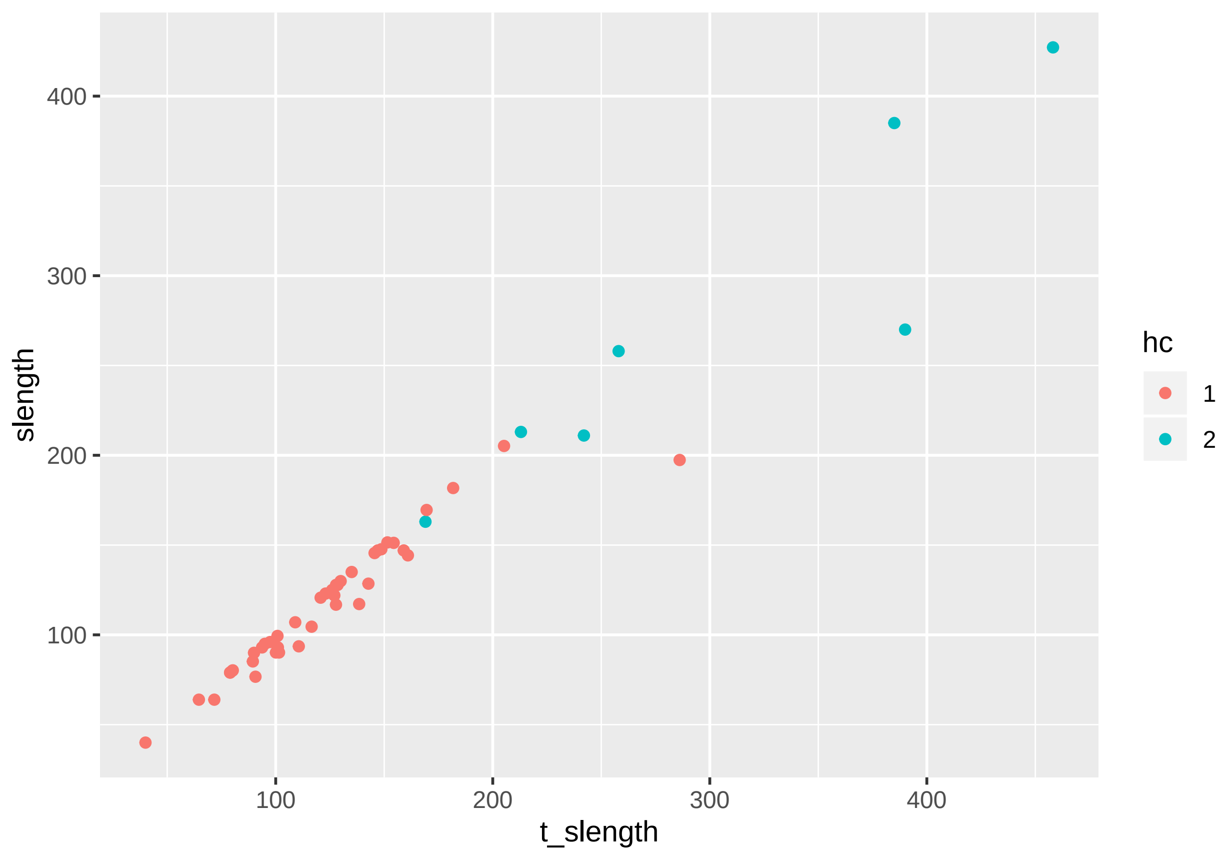
```r
ggplot(data = df)+
  geom_point(aes(x = salary_real, y = slength, color = hc))
```



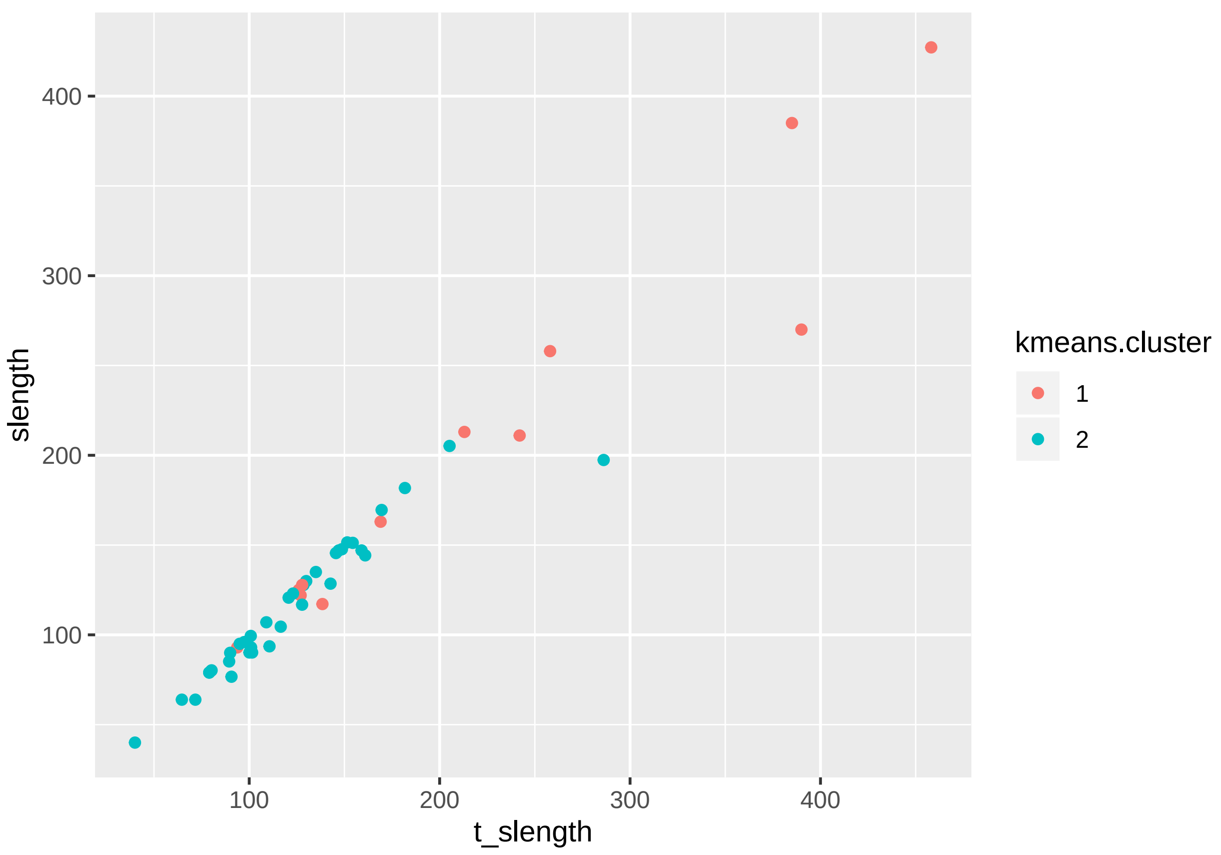
```r
ggplot(data = df)+
  geom_point(aes(x = salary_real, y = slength, color = kmeans.cluster))
```



```
ggplot(data = df)+
  geom_point(aes(x = t_slength, y = slength, color = hc))
```



```r
ggplot(data = df)+
  geom_point(aes(x = t_slength, y = slength, color = kmeans.cluster))
```



8. (5 points) Select a validation strategy

```r
library(clValid)
```

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

```r
cl <- clValid(lp_clean, nClust = 2:6,
              clMethods = c("hierarchical", "kmeans"),
              method = "complete",
              validation = c("stability", "internal"))
summary(cl)
```

```
##
## Clustering Methods:
##  hierarchical kmeans
##
## Cluster sizes:
##  2 3 4 5 6
##
## Validation Measures:
##                                  2       3       4       5       6
##
## hierarchical APN            0.0437  0.0661  0.2167  0.2427  0.2243
##              AD             1.8139  1.5295  1.3694  1.2615  1.0919
##              ADM           0.5576  0.3320  0.3996  0.5182  0.5128
##              FOM           0.8243  0.7322  0.6390  0.5671  0.5262
##              Connectivity   7.9071 10.5238 12.9583 25.7397 31.7056
##              Dunn          0.1673  0.2077  0.2872  0.1731  0.1094
##              Silhouette    0.6204  0.5884  0.5236  0.2391  0.3125
## kmeans       APN           0.0136  0.0801  0.3345  0.2506  0.1334
##              AD            1.5945  1.4966  1.4072  1.1812  0.9794
##              ADM           0.0920  0.2803  0.6002  0.4560  0.2697
##              FOM           0.7094  0.7009  0.6998  0.5805  0.4917
##              Connectivity   8.4460 10.8960 16.1885 28.7437 35.1774
##              Dunn          0.1735  0.2581  0.2562  0.1090  0.1130
##              Silhouette    0.6458  0.6131  0.4932  0.3042  0.3388
##
## Optimal Scores:
##
##              Score  Method       Clusters
## APN          0.0136 kmeans       2
## AD           0.9794 kmeans       6
## ADM          0.0920 kmeans       2
## FOM          0.4917 kmeans       6
## Connectivity 7.9071 hierarchical 2
## Dunn         0.2872 hierarchical 4
## Silhouette   0.6458 kmeans       2
```

9. (10 points) Discuss the validation output, e.g.,

- What can you take away from the fit?

Hierarchical and kmeans have similar fits

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

Mesuring by silhoette width, kmeans is the best fit, at a k=2

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

First of all, there are multiple, sometimes contradicting measures of fit, so the judgement call of which to use can influence the optimal choice. Beyond this questons of limitations on computing power and time could advantage some, less computatioally intensive, algorithms even if their fit is inferior