HW 7 Q4

Ben Howell

3/24/2022

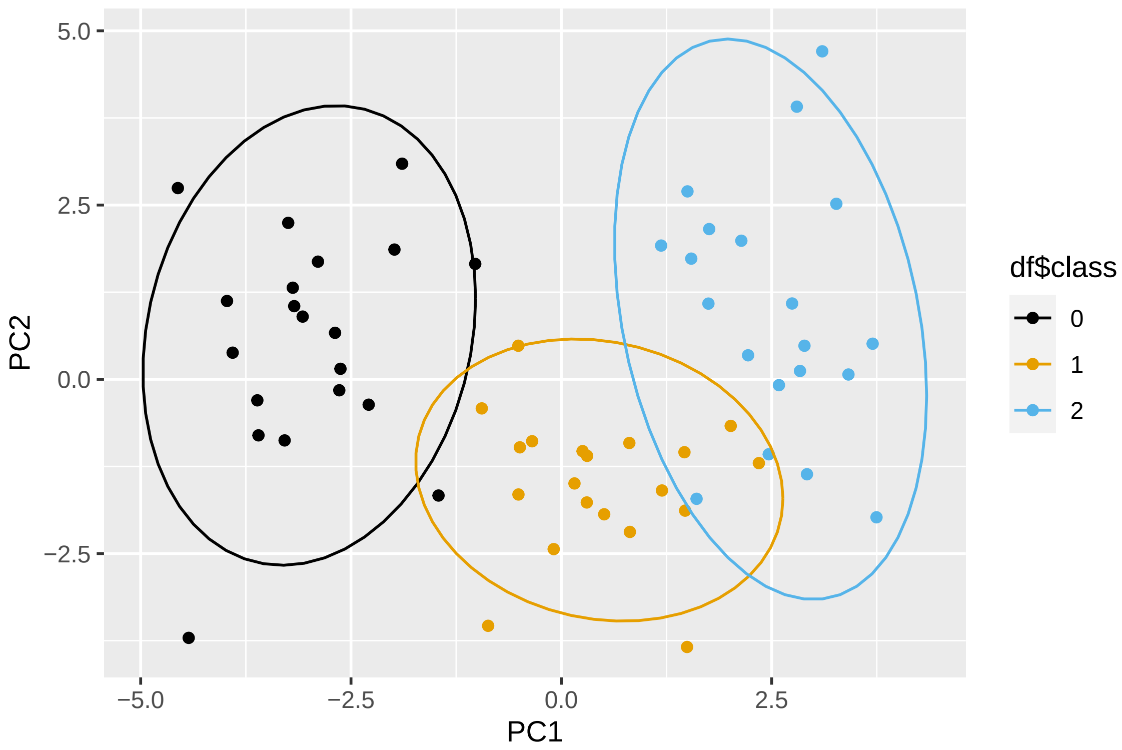
A)

```
set.seed(123)

require(tidyverse)
require(janitor)
require(MLmetrics)
require(LICORS)

df <- data.frame(replicate(50, rnorm(20, mean = rnorm(1, mean = 0), sd = 3))) %>%
    rbind(data.frame(replicate(50, rnorm(20, mean = rnorm(1, mean = 1), sd = 3)))) %>%
    rbind(data.frame(replicate(50, rnorm(20, mean = rnorm(1, mean = 2), sd = 3)))) %>%
    clean_names() %>%
    dplyr::mutate(class = ifelse(row_number() <= 20, "0",
                                 ifelse(row_number() > 20 & row_number() <= 40, "1", "2")))

res <- prcomp(df %>%
         dplyr::select(-c(class)),
       scale. = TRUE)

head(res$x %>%
       data.frame() %>%
       dplyr::select(PC1:PC10))
```

```
##         PC1        PC2         PC3         PC4        PC5        PC6
## 1 -3.599741 -0.8040346 -0.14826007  1.19002268  2.5687074 -1.9255866
## 2 -1.459633 -1.6674755 -1.01988783  1.99040039 -1.6554270 -0.6138785
## 3 -3.175022  1.0494639 -0.11562471  0.41592368 -0.1280092 -0.8203033
## 4 -3.074665  0.8994304  3.24555837  0.50422037  0.8655841  1.2432481
## 5 -2.625246  0.1506271  1.05989162  0.04765814  1.4004633 -0.4352534
## 6 -3.246501  2.2443454  0.08463199 -1.51525747 -1.5802781  2.0304992
##           PC7        PC8        PC9       PC10
## 1 -0.13123134  0.8183368  0.2714540 -1.4578003
## 2  3.30119495 -2.8426974 -2.5388748  1.4410693
## 3 -1.21414831 -0.0442793  0.7460123 -0.6239612
## 4 -1.82899598  2.5445949 -0.5808259 -0.5326450
## 5  1.33139235 -1.4813495  2.0233303  1.6077639
## 6 -0.08079525  1.2605349 -0.1320301 -1.8497137
```

B)

```
res$x %>%
  data.frame() %>%
  ggplot() +
  geom_point(aes(x = PC1, y = PC2, color = df$class)) +
  stat_ellipse(aes(x = PC1, y = PC2, color = df$class)) +
  ggthemes::scale_color_colorblind()
```



C)

```
km <- kmeans(df %>%
               dplyr::select(-c(class)),
             3)

print(table(df$class, km$cluster))
```

```
##
##       1  2  3
##   0   0 19  1
##   1   3  0 17
##   2  20  0  0
```

The K-Means clustering does a good job of differentiating between the classes that we created. We see that Clusters 0 and 2 got most of their observations assigned to their own cluster, but some of Cluster 1 got split, which will be interesting to keep an eye on. From looking at the plot in part B, that's not super surprising.

D)

```
km <- kmeans(df %>%
               dplyr::select(-c(class)),
             2)

print(table(df$class, km$cluster))
```

```
##
##       1  2
##   0   0 20
##   1  10 10
##   2  20  0
```

With `K = 2`, we see that our classes 0 and 2 are distinctly their own thing, while class 1 gets split 50/50 between those two classes. Again, this makes sense from looking at the way the PCA vectors came out.

E)

```
km <- kmeans(df %>%
               dplyr::select(-c(class)),
             4)

print(table(df$class, km$cluster))
```

```
##
##       1  2  3  4
##   0   0 20  0  0
##   1   2  0 14  4
##   2  15  0  0  5
```

`K = 4` starts to return some interesting results where we begin to see some separation and dispersion among the classes. Class 0 remains undefeated in being its own cluster, but the other two classes get spread more. It's interesting to see that classes 1 got split across 3 different clusters which kinda indicated how that data generated was more dispered itself.

F)

```r
km <- kmeans(
  res$x %>%
    data.frame() %>%
    dplyr::select(PC1, PC2),
  3
)

table(df$class, km$cluster)
```

```
##
##      1  2  3
##   0  1 19  0
##   1 20  0  0
##   2  4  0 16
```

It's interesting to see that this approach accuractely classified 92% of the observations, barely below the 93% of the first `K = 3` classifier that we ran in part C. Being able to get that close while using just two inputs for each variable, rather than 50 is quite the improvement and shows how the PCA vectors improved our clustering.

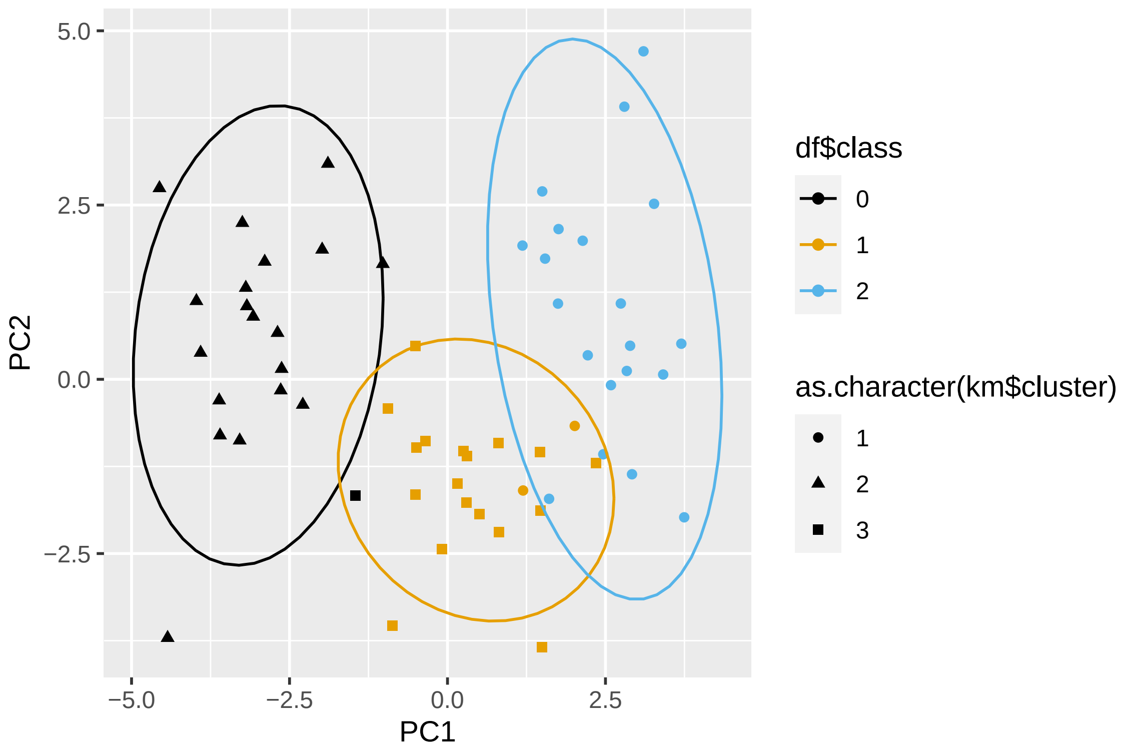
G)

```r
km <- scale(df %>%
                dplyr::select(-c(class))) %>%
  data.frame() %>%
  kmeans(
    centers = 3
  )

table(df$class, km$cluster)
```

```
##
##      1  2  3
##   0  0 19  1
##   1  2  0 18
##   2 20  0  0
```

Interestingly, once we scale the data, we see the most accurate of our `K = 3` classifiers, predicting 95% of the classes properly, which is good to see. Putting everything onto a consistent scale helped out the quality of our data and allowed the model to improve it's ability to classify our generated classes. On the diagnostic plot below, the observations that are mis-classified make sense as they are outliers among their generated class and almost fit within the elipse of another cluster.

```r
res$x %>%
  data.frame() %>%
  ggplot() +
  geom_point(aes(x = PC1, y = PC2, color = df$class, shape = as.character(km$cluster))) +
  stat_ellipse(aes(x = PC1, y = PC2, color = df$class)) +
  ggthemes::scale_color_colorblind()
```


5.0
2.5
0.0
−2.5
−5.0
−2.5
0.0
2.5
PC1
PC2
df$class
0
1
2
as.character(km$cluster)
1
2
3