Lab Assignment #12

Nick Noel, Liz Villa, and Cadee Pinkerton

Due May 12, 2023

Instructions

The purpose of this lab is to introduce unsupervised learning methods.

```
library(tidyverse)
library(tidymodels) # only main part of tidymodels you might need
library(broom)
library(tidyclust)
library(mclust)
library(GGally)

madden17_QB <- readr::read_csv("madden17_QB.csv")
cereal3 <- readr::read_csv("cereal3.csv")
```

This lab assignment is worth a total of **20 points**.

Problem 1: Principal Components Analysis

Part a (Code: 1 pt)

Run the code in ISLR Lab 12.5.1.

```
states <- row.names(USArrests)
states
```

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

```
names(USArrests)
```

```
## [1] "Murder"   "Assault"  "UrbanPop" "Rape"
```

```
apply(USArrests, 2, mean)
```

```
##   Murder  Assault UrbanPop     Rape
##    7.788  170.760   65.540   21.232
```

```
apply(USArrests, 2, var)
```

```
##     Murder    Assault   UrbanPop       Rape
##   18.97047 6945.16571  209.51878   87.72916
```

```
pr.out <- prcomp(USArrests, scale = TRUE )
```

```
names(pr.out)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

```
pr.out$scale
```

```
##    Murder   Assault  UrbanPop      Rape
##  4.355510 83.337661 14.474763  9.366385
```

```
pr.out$center
```

```
##   Murder  Assault UrbanPop     Rape
##    7.788  170.760   65.540   21.232
```

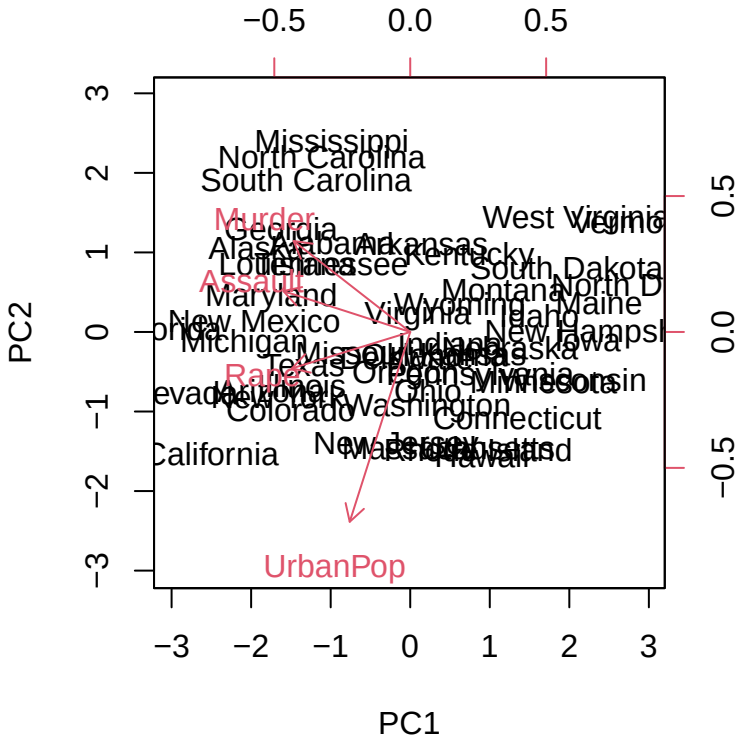
```
pr.out$rotation
```

```
##                 PC1        PC2        PC3         PC4
## Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
## Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
## UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
## Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```

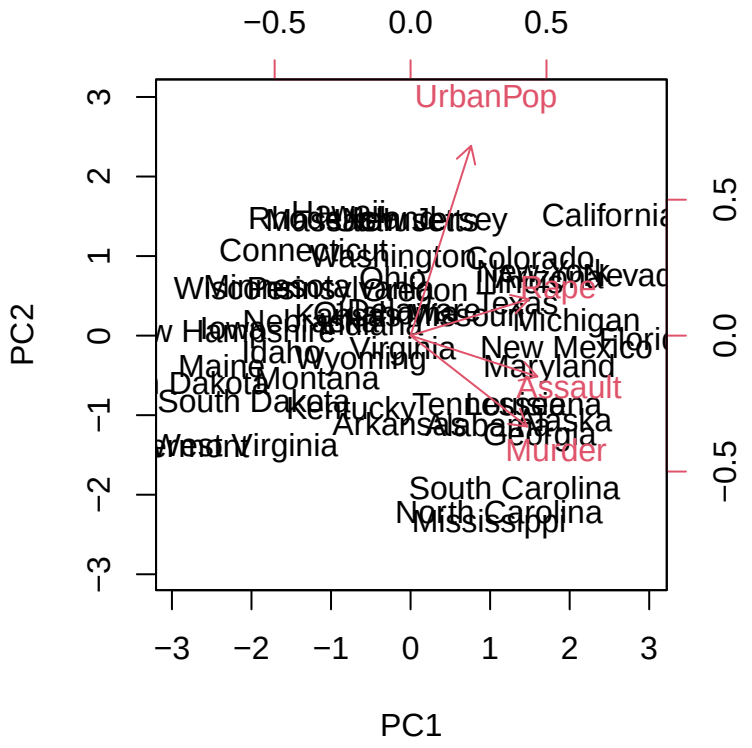
```
dim(pr.out$x)
```

```
## [1] 50  4
```

```
biplot(pr.out, scale = 0)
```




```
pr.out$rotation = -pr.out$rotation

pr.out$x = -pr.out$x

biplot(pr.out, scale = 0)
```




```
pr.out$sdev
```

```
## [1] 1.5748783 0.9948694 0.5971291 0.4164494
```

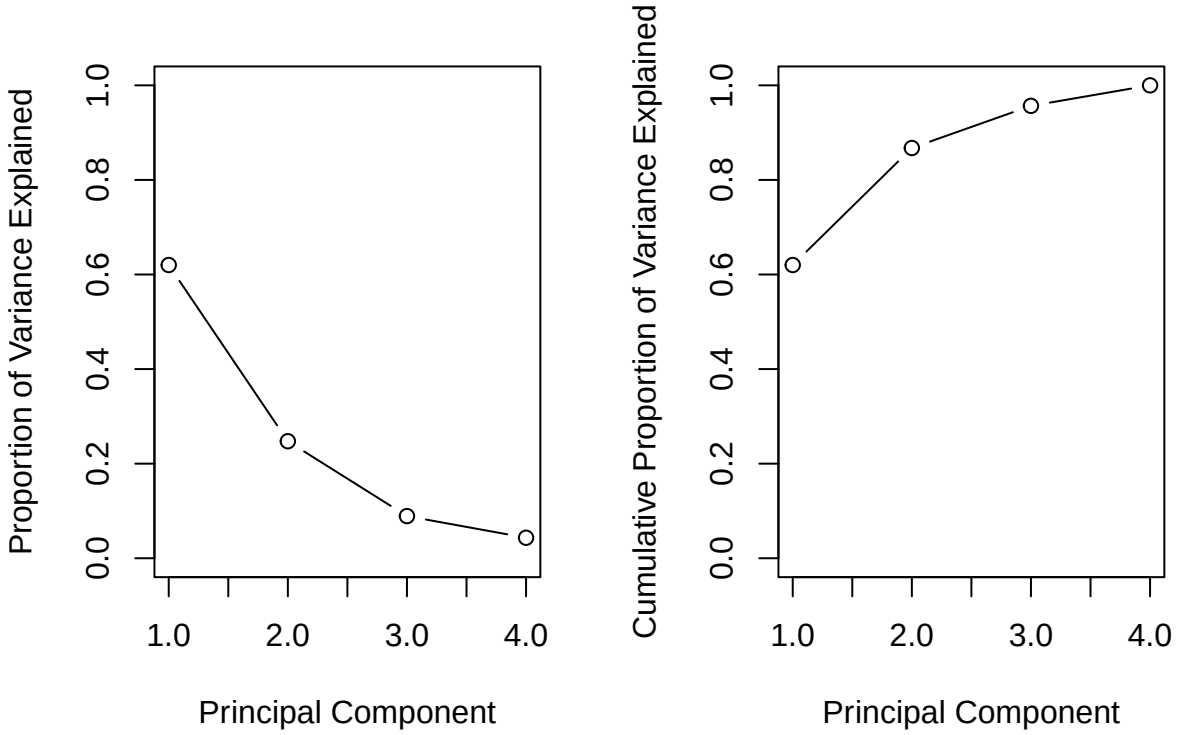
```
pr.var <- pr.out$sdev^2

pr.var
```

```
## [1] 2.4802416 0.9897652 0.3565632 0.1734301
```

```
pve <- pr.var / sum(pr.var)

pve
```

```
## [1] 0.62006039 0.24744129 0.08914080 0.04335752
```

```
par(mfrow = c(1,2))
plot(pve, xlab = "Principal Component",
     ylab = "Proportion of Variance Explained", ylim = c(0,1),
     type = "b")
plot(cumsum(pve), xlab = "Principal Component",
     ylab = "Cumulative Proportion of Variance Explained",
     ylim = c(0,1), type = "b")
```




```
a <- c(1,2,8,-3)

cumsum(a)
```

```
## [1]  1  3 11  8
```

Part b (Code: 0.5 pts)

The madden17_QB dataset contains the overall rating (`OVR`) and individual skill ratings for 112 quarterbacks in the Madden NFL 2017 video game. According to an article on fivethirtyeight.com, the overall rating for quarterbacks is a linear combination of the following skill ratings: `AWR`, `THP`, `SAC`, `MAC`, `DAC`, `PAC`, `SPD`, `AGI`, `RUN`, and `ACC`. The other 34 skill ratings are not relevant.

Subset the dataset to contain only the 10 skill ratings used to create the overall rating. Call the new dataset `madden`.

```
madden <- madden17_QB %>%
  select(AWR, THP, SAC, MAC, DAC, PAC, SPD, AGI, RUN, ACC)
```

Part c (Code: 1 pt)

Perform principal component analysis on the `madden` dataset. Remember to scale the data (either beforehand or using the argument `scale = TRUE` in `prcomp`). You can use either the "Base R" or tidyverse version.

```
apply(madden, 2, mean)
```

```
##       AWR       THP       SAC       MAC       DAC       PAC       SPD       AGI
## 67.35714 88.51786 82.49107 79.11607 74.17857 71.81250 74.81250 71.51786
```

```
##      RUN      ACC
## 77.99107 79.29464
```

```
apply(madden, 2, var)
```

```
##       AWR       THP       SAC       MAC       DAC       PAC       SPD       AGI
## 216.57400  23.22490  33.49542  42.17560  34.92278  90.94651  46.18975  99.04472
##       RUN       ACC
##  20.40532  40.46195
```

```
pr.out <- prcomp(madden, scale = TRUE )
```

```
names(pr.out)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

```
pr.out$scale
```

```
##       AWR       THP       SAC       MAC       DAC       PAC       SPD       AGI
## 14.716453  4.819222  5.787522  6.494274  5.909550  9.536588  6.796304  9.952122
##       RUN       ACC
##  4.517225  6.360971
```

```
pr.out$center
```

```
##      AWR      THP      SAC      MAC      DAC      PAC      SPD      AGI
## 67.35714 88.51786 82.49107 79.11607 74.17857 71.81250 74.81250 71.51786
##      RUN      ACC
## 77.99107 79.29464
```

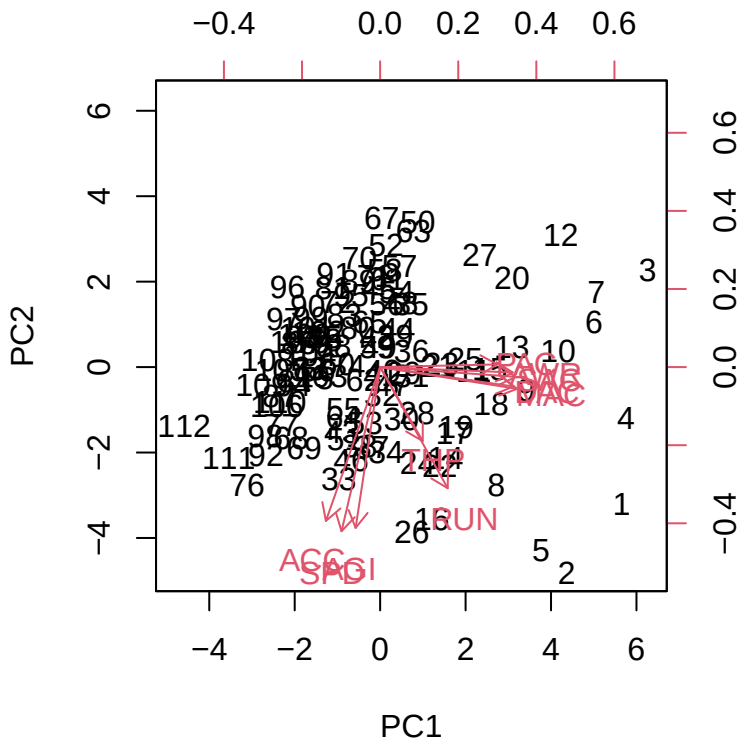
```
pr.out$rotation
```

```
##             PC1          PC2         PC3         PC4          PC5         PC6
## AWR -0.42441593  0.017256381 -0.05706060 -0.26224744  0.163830684 -0.40035753
## THP -0.13637091  0.236733925  0.90668505  0.08014228  0.008906555 -0.01796417
## SAC -0.43331673  0.031210028 -0.24065938 -0.11790315  0.130547005  0.17081050
## MAC -0.43818989  0.069546191 -0.12427166 -0.23248998 -0.017640419  0.31278125
## DAC -0.42751367  0.062114235  0.19824025 -0.31932461 -0.047648450  0.06051957
## PAC -0.37463870 -0.008543857 -0.06649819  0.79412673  0.420213108 -0.05465583
## SPD  0.12354895  0.525418497 -0.11803405 -0.03227210  0.239220411  0.10738145
## AGI  0.07850563  0.514770751 -0.14120154 -0.10763514  0.098833530 -0.65808845
## RUN -0.21633226  0.387737120 -0.14978995  0.33058143 -0.798819637  0.02818448
## ACC  0.17364074  0.492467008 -0.03574598 -0.04831553  0.267898787  0.51022286
##              PC7         PC8          PC9        PC10
## AWR  0.658221073 -0.21925306  0.251078155  0.12668836
## THP -0.090278423 -0.29638026 -0.018892161  0.01044276
## SAC -0.357301545 -0.46796848  0.095852941 -0.58084337
## MAC -0.261059404 -0.08495510 -0.251738167  0.70482134
## DAC -0.008303772  0.75664336 -0.053300261 -0.30125479
## PAC -0.013024952  0.19963995 -0.006461988  0.06985520
## SPD  0.325725277 -0.06778776 -0.693288811 -0.18329987
## AGI -0.471455624  0.09746628  0.105988675  0.10711014
## RUN  0.150307892 -0.03770208  0.079814905 -0.04298444
## ACC  0.107610376  0.09996693  0.602417536  0.08447235
```

```
dim(pr.out$x)
```

```
## [1] 112  10
```

```
pr.out$rotation = -pr.out$rotation

pr.out$x = -pr.out$x

biplot(pr.out, scale = 0)
```




```
pr.out$sdev
```

```
##  [1] 2.1109425 1.7064976 0.9464790 0.6739242 0.6599395 0.5361461 0.4510820
##  [8] 0.3807177 0.3587281 0.2858526
```

```
pr.var <- pr.out$sdev^2

pr.var
```

```
##  [1] 4.45607820 2.91213407 0.89582253 0.45417389 0.43552009 0.28745266
##  [7] 0.20347501 0.14494598 0.12868584 0.08171173
```

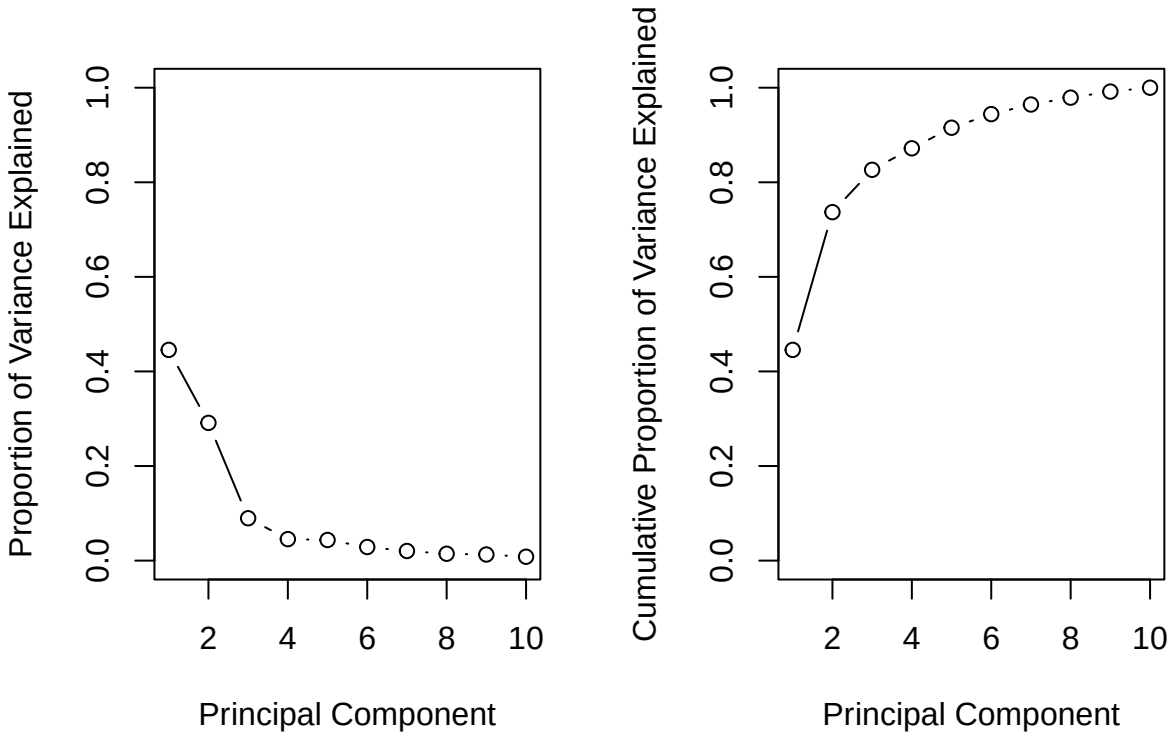
```
pve <- pr.var / sum(pr.var)

pve
```

```
##  [1] 0.445607820 0.291213407 0.089582253 0.045417389 0.043552009 0.028745266
##  [7] 0.020347501 0.014494598 0.012868584 0.008171173
```

Part d (Code: 2 pts; Explanation: 1 pt)

Find the proportion of variance explained by each component and the cumulative proportion of variance explained. Produce a scree plot showing either the proportion of variance explained or the cumulative proportion of variance explained. Suggest an appropriate number of principal components to use to visualize or interpret the data and justify your decision based on the scree plot.

```
par(mfrow = c(1,2))
plot(pve, xlab = "Principal Component",
     ylab = "Proportion of Variance Explained", ylim = c(0,1),
     type = "b")
plot(cumsum(pve), xlab = "Principal Component",
     ylab = "Cumulative Proportion of Variance Explained",
     ylim = c(0,1), type = "b")
```

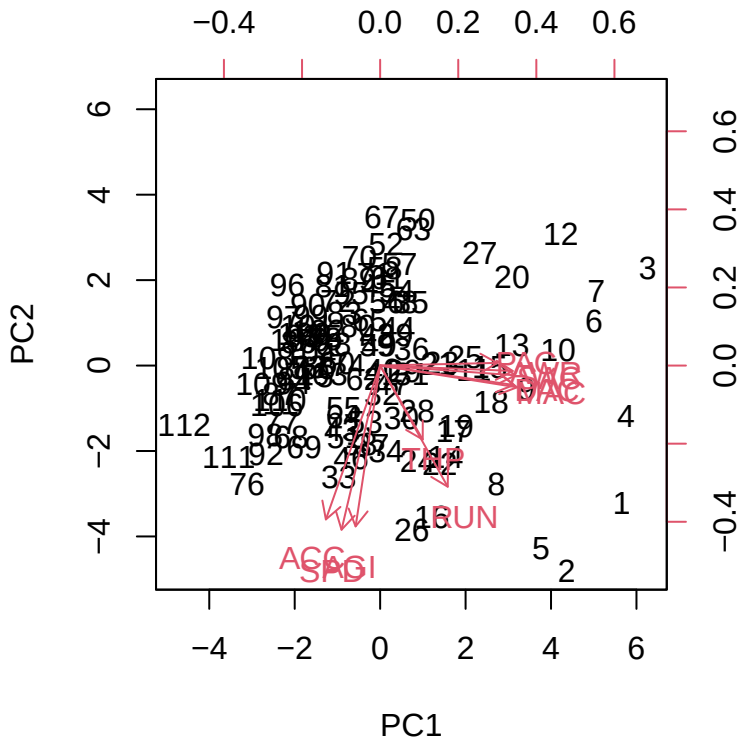


An appropriate amount of principal components to use would be 3 as after this our variance tends to level out.

Part e (Code: 0.5 pts; Explanation: 1 pt)

Produce a biplot showing the first two principal components. Which of the ten variables being investigated contribute mainly to PC1, which ones contribute mainly to PC2, and which contribute to both? Explain your reasoning based on the biplot and/or the loadings matrix `rotation`.

```
biplot(pr.out, scale = 0)
```



ACC, AGI, and SPD contribute to PC2. THP and RUN in between meaning they contribute to both, and The remaining variables: PAC, AWR, SAC, MAC, and DAC contribute mainly to PC1. (Note, I was not sure how to get the variable names from this plot but it would definitely be something nice to know)

Problem 2: Clustering

Part a (Code: 2 pts)

Run the code in ISLR Lab 12.5.3.

```
library(ISLR2)
nci.labs <- NCI60$labs
nci.data <- NCI60$data
```

```
dim(nci.data)
```

```
## [1]   64 6830
```

```
nci.labs[1:4]
```

```
## [1] "CNS"   "CNS"   "CNS"   "RENAL"
```

```
table(nci.labs)
```

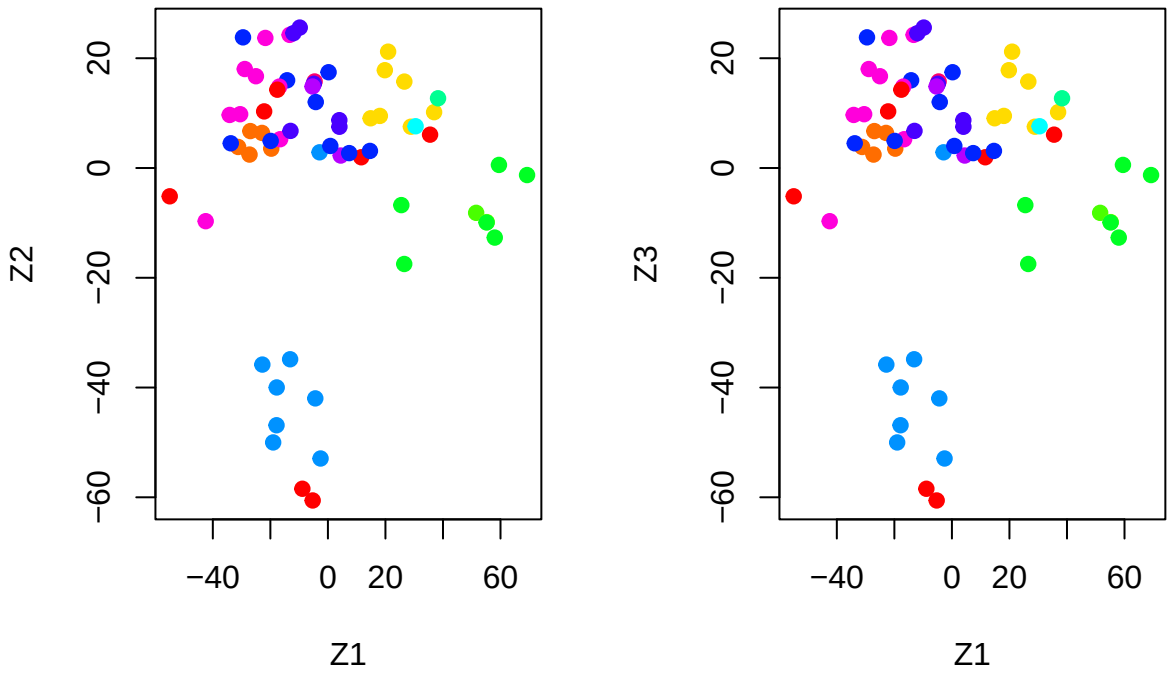
```
## nci.labs
##      BREAST         CNS       COLON K562A-repro K562B-repro    LEUKEMIA
##           7           5           7           1           1           6
```

```
## MCF7A-repro MCF7D-repro    MELANOMA       NSCLC     OVARIAN    PROSTATE
##           1           1           8           9           6           2
##       RENAL     UNKNOWN
##           9           1
```

```
pr.out <- prcomp(nci.data, scale = TRUE)
```

```
Cols <- function(vec){
  cols <- rainbow(length(unique(vec)))
  return(cols[as.numeric(as.factor(vec))])
}
```

```
par(mfrow = c(1, 2))
plot(pr.out$x[, 1:2], col = Cols(nci.labs), pch = 19, xlab = "Z1", ylab = "Z2")
plot(pr.out$x[, 1:2], col = Cols(nci.labs), pch = 19, xlab = "Z1", ylab = "Z3")
```

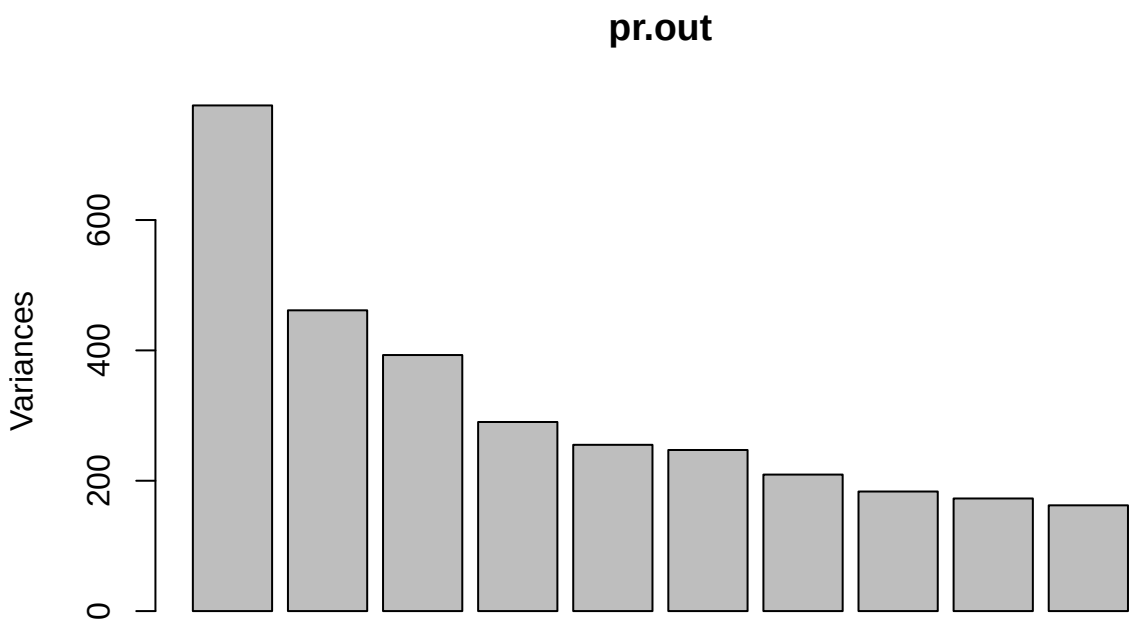



```
summary(pr.out)
```

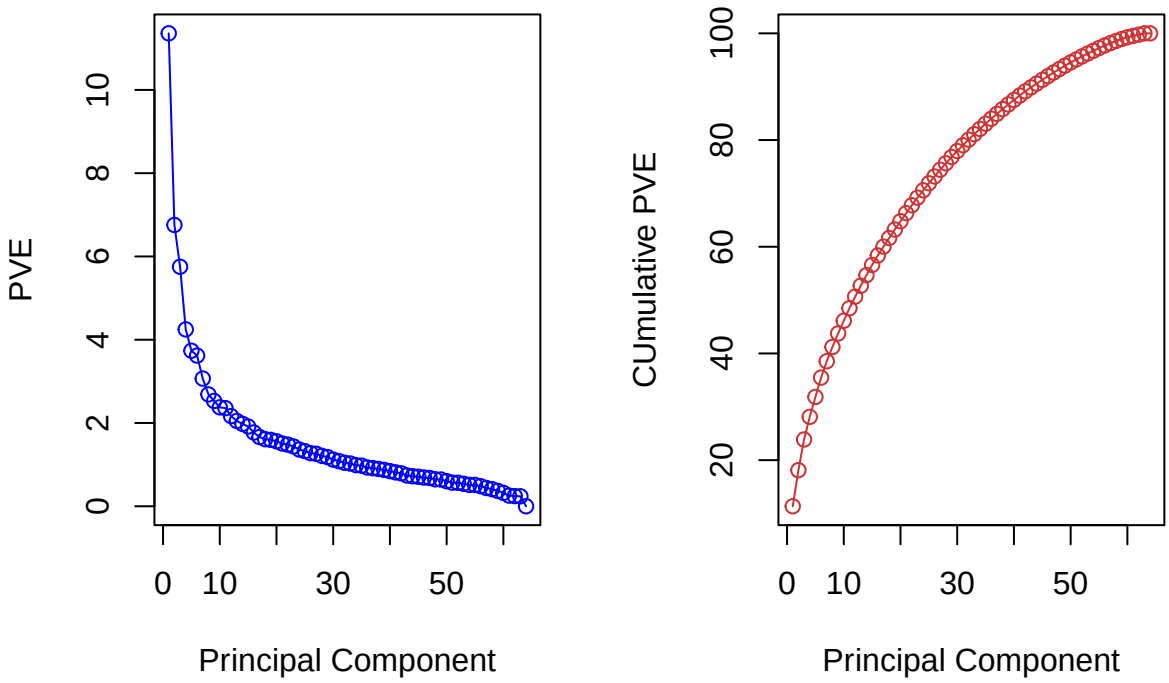
```
## Importance of components:
##                             PC1      PC2      PC3      PC4      PC5      PC6
## Standard deviation     27.8535 21.48136 19.82046 17.03256 15.97181 15.72108
## Proportion of Variance  0.1136  0.06756  0.05752  0.04248  0.03735  0.03619
## Cumulative Proportion   0.1136  0.18115  0.23867  0.28115  0.31850  0.35468
##                             PC7      PC8      PC9     PC10     PC11     PC12
## Standard deviation     14.47145 13.54427 13.14400 12.73860 12.68672 12.15769
## Proportion of Variance  0.03066  0.02686  0.02529  0.02376  0.02357  0.02164
## Cumulative Proportion   0.38534  0.41220  0.43750  0.46126  0.48482  0.50646
##                            PC13     PC14     PC15     PC16     PC17     PC18
```

```
## Standard deviation     11.83019 11.62554 11.43779 11.00051 10.65666 10.48880
## Proportion of Variance  0.02049  0.01979  0.01915  0.01772  0.01663  0.01611
## Cumulative Proportion   0.52695  0.54674  0.56590  0.58361  0.60024  0.61635
##                            PC19    PC20     PC21    PC22    PC23    PC24
## Standard deviation     10.43518 10.3219 10.14608 10.0544 9.90265 9.64766
## Proportion of Variance  0.01594  0.0156  0.01507  0.0148 0.01436 0.01363
## Cumulative Proportion   0.63229  0.6479  0.66296  0.6778 0.69212 0.70575
##                            PC25    PC26    PC27   PC28    PC29    PC30    PC31
## Standard deviation     9.50764 9.33253 9.27320 9.0900 8.98117 8.75003 8.59962
## Proportion of Variance 0.01324 0.01275 0.01259 0.0121 0.01181 0.01121 0.01083
## Cumulative Proportion  0.71899 0.73174 0.74433 0.7564 0.76824 0.77945 0.79027
##                            PC32    PC33    PC34    PC35    PC36    PC37    PC38
## Standard deviation     8.44738 8.37305 8.21579 8.15731 7.97465 7.90446 7.82127
## Proportion of Variance 0.01045 0.01026 0.00988 0.00974 0.00931 0.00915 0.00896
## Cumulative Proportion  0.80072 0.81099 0.82087 0.83061 0.83992 0.84907 0.85803
##                            PC39    PC40    PC41   PC42    PC43   PC44    PC45
## Standard deviation     7.72156 7.58603 7.45619 7.3444 7.10449 7.0131 6.95839
## Proportion of Variance 0.00873 0.00843 0.00814 0.0079 0.00739 0.0072 0.00709
## Cumulative Proportion  0.86676 0.87518 0.88332 0.8912 0.89861 0.9058 0.91290
##                          PC46    PC47    PC48    PC49    PC50    PC51    PC52
## Standard deviation     6.8663 6.80744 6.64763 6.61607 6.40793 6.21984 6.20326
## Proportion of Variance 0.0069 0.00678 0.00647 0.00641 0.00601 0.00566 0.00563
## Cumulative Proportion  0.9198 0.92659 0.93306 0.93947 0.94548 0.95114 0.95678
##                            PC53    PC54    PC55    PC56    PC57   PC58    PC59
## Standard deviation     6.06706 5.91805 5.91233 5.73539 5.47261 5.2921 5.02117
## Proportion of Variance 0.00539 0.00513 0.00512 0.00482 0.00438 0.0041 0.00369
## Cumulative Proportion  0.96216 0.96729 0.97241 0.97723 0.98161 0.9857 0.98940
##                            PC60    PC61    PC62    PC63      PC64
## Standard deviation     4.68398 4.17567 4.08212 4.04124 2.148e-14
## Proportion of Variance 0.00321 0.00255 0.00244 0.00239 0.000e+00
## Cumulative Proportion  0.99262 0.99517 0.99761 1.00000 1.000e+00
```

```
plot(pr.out)
```




```
pve <- 100 * pr.out$sdev^2 / sum(pr.out$sdev^2)
par(mfrow = c(1, 2))
plot(pve, type = "o", ylab = "PVE", xlab = "Principal Component", col = "blue")
plot(cumsum(pve), type = "o", ylab = "CUmulative PVE", xlab = "Principal Component", col = "brown3")
```



Part b (Code: 1 pt; Explanation: 1 pt)

The `cereal3` dataset on Canvas contains information about 88 cereals being sold at an Albertson's in Irvine. This data was collected by Dr. Wynne in 2019.

We want to cluster cereals based on their nutritional information. The chunk below creates a matrix of relevant variables (we use `model.matrix` to simultaneously convert categorical variables into dummy variables, should we have any).

```
cereal <- cereal3 %>%
  mutate(Complex.Carbs = Total.Carbohydrate - Dietary.Fiber - Sugar) %>%
  select(Cereal.Abb, Total.Fat, Sodium, Complex.Carbs, Dietary.Fiber,
         Sugar, Protein)

x.matrix <- model.matrix(~ Total.Fat + Sodium +
                           Complex.Carbs + Dietary.Fiber +
                           Sugar + Protein, data = cereal)[,-1]
```

Perform k-means clustering on the nutritional variables (i.e., the `x.matrix`) *without* scaling the variables. Use 4 clusters and `nstart = 20`.

```
kmeans_model <- k_means(num_clusters = 4) %>%
  set_args(nstart = 20)

kmeans_recipe_ck <- recipe(~ Total.Fat + Sodium +
                           Complex.Carbs + Dietary.Fiber +
                           Sugar + Protein, data = cereal)
```

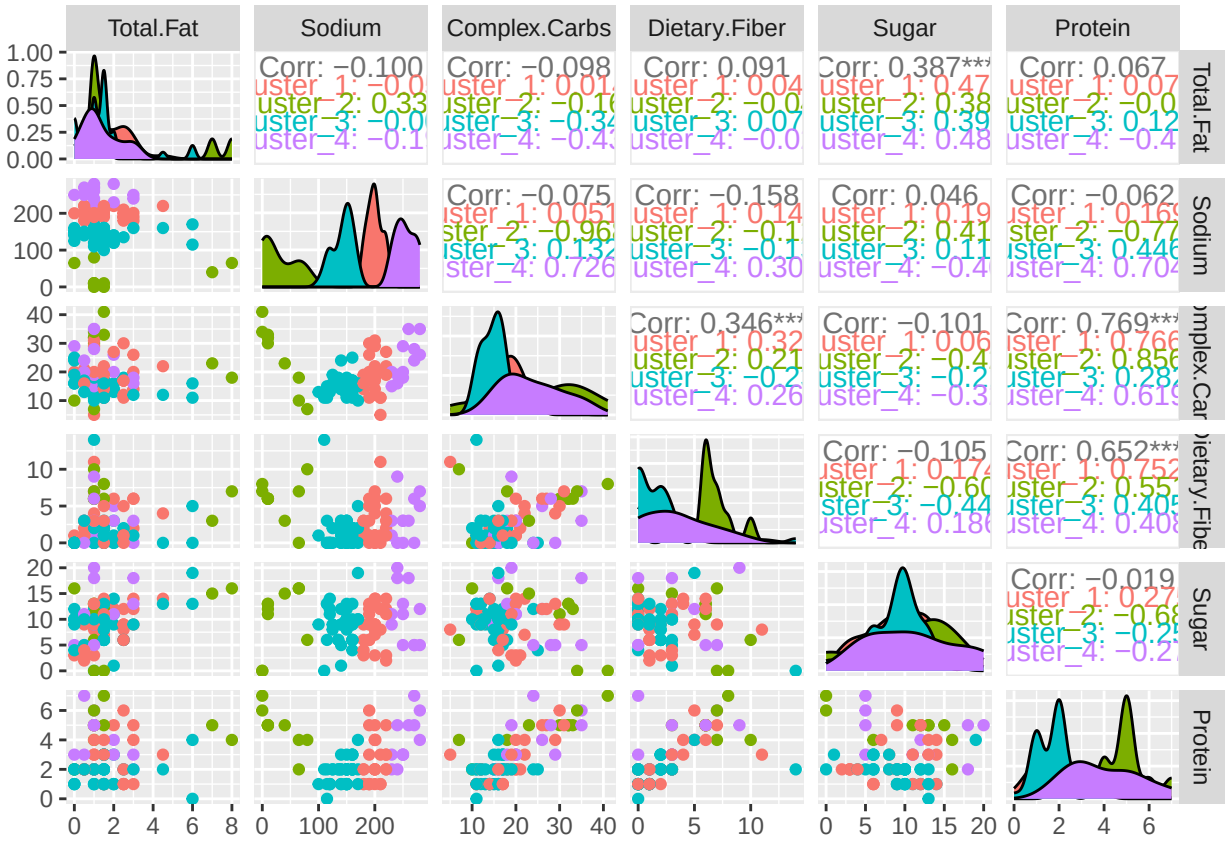
```
kmeans_wflow_ck <- workflow() %>%
  add_model(kmeans_model) %>%
  add_recipe(kmeans_recipe_ck)
```

```
set.seed(437)

kmeans_ck_fit <- fit(kmeans_wflow_ck, data = x.matrix)
```

```
assignments <- bind_cols(
  cereal,
  kmeans_ck_fit %>% extract_cluster_assignment())
```

Using the `pairs` function, produce a plot of the clusters, color-coded by the cluster number.

```
ggpairs(assignments, columns = c("Total.Fat", "Sodium",
                                 "Complex.Carbs", "Dietary.Fiber",
                                 "Sugar", "Protein"),
        aes(color = .cluster))
```



Looking at the cluster `centers` or the plot, which variable appears to be the most important for distinguishing between the clusters? Why? Is this what you expected?

Sodium appears to be the most important for distinguishing between the clusters since we can see our data has clear clusters that do not seem to overlap when grouped by this variable. I would not have expected such a clear distinction based on sodium, I figured all cereals would have roughly the same amount of all the nutritional variables possibly with the only distinction being the "healthy" cereals which would be lower in sugar, fats, and higher in fiber and complex carbs.

Part c (Code: 1 pt; Explanation: 2 pts)

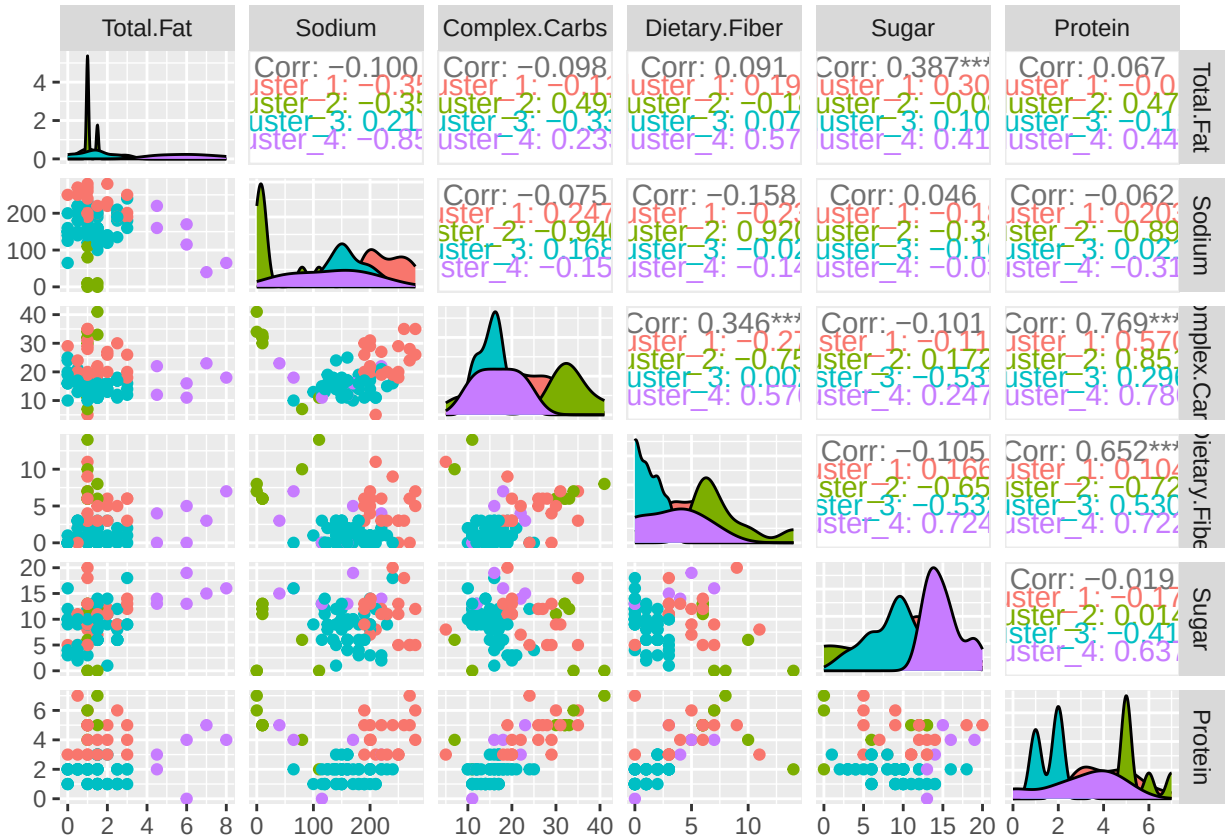
Scale the nutritional variables and re-run k-means clustering with 4 clusters and `nstart = 20`. Using the `pairs` function, produce a plot of the clusters (on the original scale), color-coded by the new cluster number.

```
kmeans_recipe_ck2 <- recipe(~ Total.Fat + Sodium +
                            Complex.Carbs + Dietary.Fiber +
                            Sugar + Protein, data = cereal) %>%
  step_normalize(all_numeric_predictors())

kmeans_wflow_ck2 <- workflow() %>%
   add_model(kmeans_model) %>%
  add_recipe(kmeans_recipe_ck2)
```

```
set.seed(437)

kmeans_ck_fit2 <- fit(kmeans_wflow_ck2, data = x.matrix)
```

```
assignments2 <- bind_cols(
  cereal,
  kmeans_ck_fit2 %>% extract_cluster_assignment())
```

```
ggpairs(assignments2, columns = c("Total.Fat", "Sodium",
                            "Complex.Carbs", "Dietary.Fiber",
                            "Sugar", "Protein"),
        aes(color = .cluster))
```



How does this plot compare to the plot you made in part (b)? What does this suggest about the importance

of scaling the variables before running k-means clustering?

The distinct clusters we saw for sodium in the previous plot now have more overlap, and it seems that the clusters for Total.Fat have become slightly more distinct. The clusters in all the other variables have also changed slightly. This shows that scaling variables when running k-means clustering is very important as without it, you could end up infering there are relationships that don't actually exist.

Looking at the cluster `centers` (remember, a mean of 0 is average after scaling) or the plot, try to assign a meaning to each cluster of cereals. For example, you should find that one of your clusters contains cereals that are high in fat.

It seems that cluster 1 has cereals that are very high in sodium and above average in complex carbs, sugar, and protein, probably contains cereals that are marketed or perceived to be "healthier" but actually aren't (like Honey Nut Cheerios). Cluster 2 has cereals that are lower in fat, sugar, and sodium and higher in complex carbs and fiber so these cereals are probably the healthier ones (I'll bet shredded wheat is in this cluster). Cluster 3 seems to have more sugar and sodium than average and with not a lot of fiber, protein, or complex carbs. Lastly, cluster 4 seems to be the "good stuff", it's far above average for fat content, sodium, and sugar.

```
assignments2 %>%
  filter(.cluster == "Cluster_4") %>%
  select(Cereal.Abb)
```

```
## # A tibble: 6 x 1
##   Cereal.Abb
##   <chr>
## 1 CheeriosOC
## 2 OatBranC
## 3 Donettes
## 4 KraveChoc
## 5 NutterButter
## 6 RMCherAlmPec
```

Oh, I didn't expect Cheerios (OC?) or Oat Bran (C?) to be in this but I KNOW Krave Chocolate^{TM} is THE "good stuff".

Part d (Code and Explanation: 1.5 pts)

Using the `augment` function from the broom package, augment the `cereal` or `cereal3` dataset with the information from the k-means clustering in part (b).

```
predictions <- augment(kmeans_ck_fit, new_data = cereal)
```

Obtain the size of each cluster. For one of the smaller clusters, filter the augmented dataset to look at only observations from that cluster. What cereals are in that cluster? Do they appear to have anything in common (think about the cereal names and anything you might know about them)?

```
predictions %>%
  group_by(.pred_cluster) %>%
  summarize(size = n())
```

```
## # A tibble: 4 x 2
##   .pred_cluster  size
##   <fct>         <int>
## 1 Cluster_1        27
## 2 Cluster_2        10
## 3 Cluster_3        39
## 4 Cluster_4        12
```

```
predictions %>%
  filter(.pred_cluster == "Cluster_2")
```

```
## # A tibble: 10 x 8
##    Cereal.Abb   Total.Fat Sodium Complex.Carbs Dietary.F~1 Sugar Protein .pred~2
##    <chr>            <dbl>  <dbl>         <dbl>       <dbl> <dbl>   <dbl> <fct>
##  1 AllBranOrig        1       80             7          10     6       4 Cluste~
##  2 OatBranC           8       65            18           7    16       4 Cluste~
##  3 FMWBlue            1       10            32           6    13       5 Cluste~
##  4 FMWLBOrig          1       10            30           6    11       5 Cluste~
##  5 FMWOrig            1.5     10            33           6    12       5 Cluste~
##  6 FMWStraw           1       10            32           6    13       5 Cluste~
##  7 GoldenCrisp        0       65            10           0    16       2 Cluste~
##  8 RMCherAlmPec       7       40            23           3    15       5 Cluste~
##  9 ShWheatBig         1        0            34           7     0       6 Cluste~
## 10 ShWheatSpoon       1.5      0            41           8     0       7 Cluste~
## # ... with abbreviated variable names 1: Dietary.Fiber, 2: .pred_cluster
```

Looking at the nutritional values, these seem to be all over the place, I'm seeing some of the "good stuff" cereals from the last k-means clustering but also some cereals I expected to be grouped with healthier ones such as shredded wheat. They seem to have some similarity in complex carbs, protein, and dietary fiber, but protein seems to be the closest one.

Part e (Code: 1 pt)

Use the scaled version of `x.matrix` to perform hierarchical clustering on the dataset. Use complete linkage (the default). Plot the dendogram using the arguments `labels = cereal$Cereal.Abb, cex = 0.7`. (You don't need to use the `dendextend` package to make things fancier, just get the plot out.)

```
ck_hc_complete <- hier_clust()

ck_hc <- workflow() %>%
  add_model(ck_hc_complete) %>%
  add_recipe(kmeans_recipe_ck2)
```

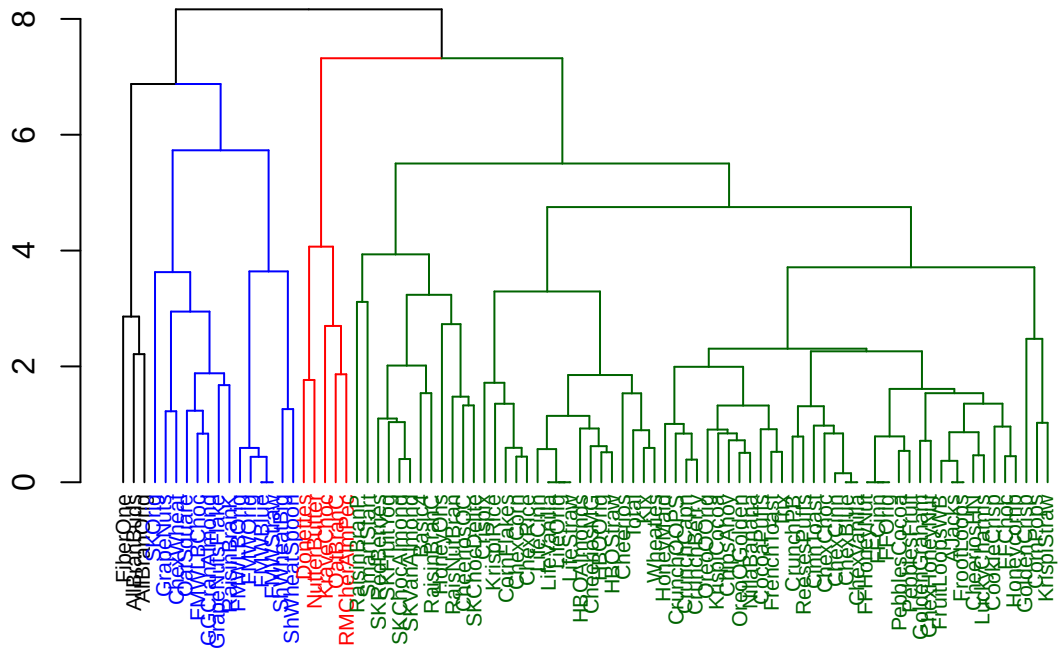
```
hc_complete_fit <- fit(ck_hc, data = x.matrix)
```

```
ck_dendrogram <- extract_fit_engine(hc_complete_fit) %>% as.dendrogram()
#plot(ck_dendrogram, labels = cereal$Cereal.Abb, cex = 0.7)
```

```
library(dendextend)
```

```
## Warning: package 'dendextend' was built under R version 4.1.3
```

```
ck_dendrogram_4clusters <- ck_dendrogram %>%
  set("labels", cereal$Cereal.Abb[labels(ck_dendrogram)]) %>%
  set("labels_cex", 0.7) %>%
  color_branches(k = 4, col = c("black","blue", "red", "darkgreen")) %>%
  color_labels(k = 4, col = c("black", "blue", "red", "darkgreen"))

plot(ck_dendrogram_4clusters)
```



Part f (Explanation: 1 pt)

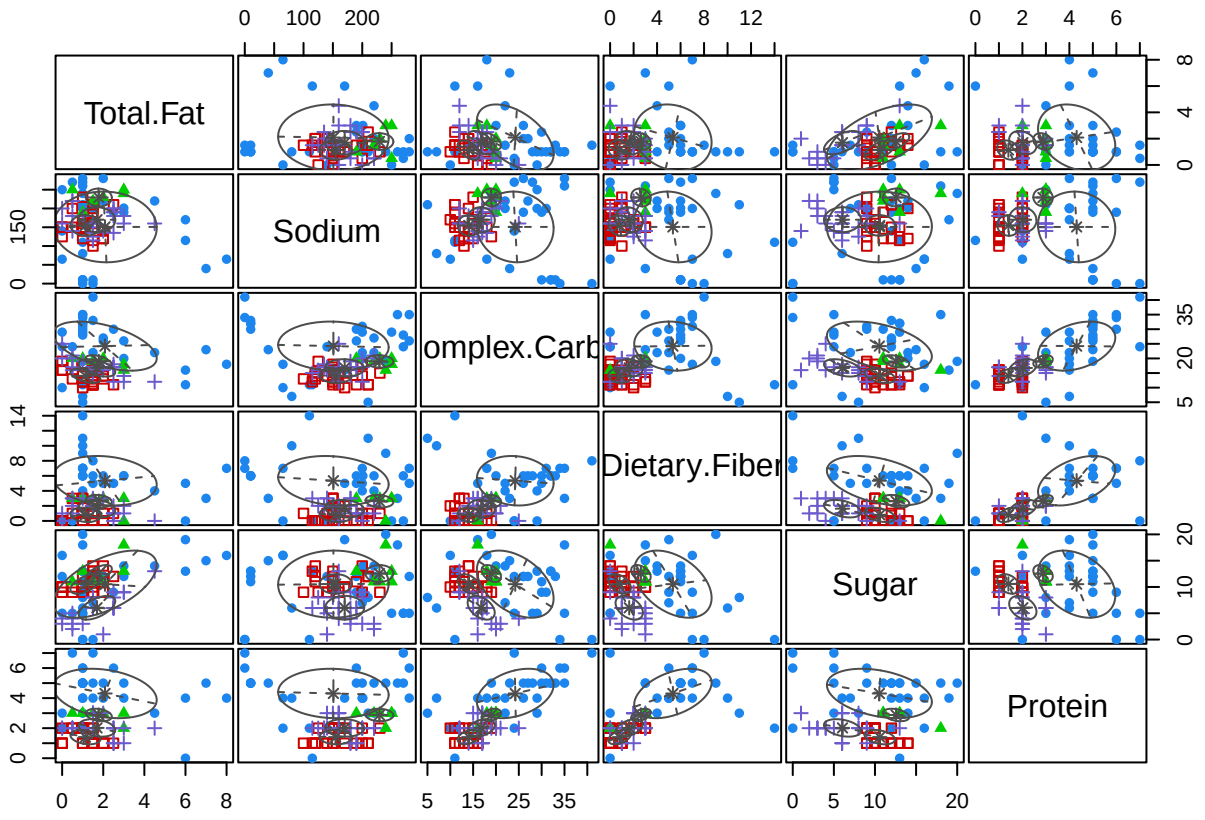
88 cereals is a bit too much to get a good look, so you may want to zoom in on the dendrogram to answer these questions.

- Which cereal or cereals are most similar to Cheerios? Total and Kix are the most similar to Cheerios.
- Which cereal or cereals are most similar to Honey Nut Cheerios (CheeriosHN)? Lucky Charms.

Part g (Code: 1.5 pts; Explanation: 1 pt)

Fit a Gaussian mixture model on `x.matrix` (scaled or unscaled, it doesn't matter, you should get basically the same results) using the `Mclust` function. Use 4 clusters (`G = 4`). Produce a "classification" plot of the resulting clusters.

```
ck_mclust <- mclust::Mclust(x.matrix, G = 4)
plot(ck_mclust, "classification")
```




```
tidy.mclust <- broom::tidy(ck_mclust)
print(tidy.mclust)
```

```
## # A tibble: 4 x 9
##   component  size proportion mean.Tota~1 mean.~2 mean.~3 mean.~4 mean.~5 mean.~6
##       <int> <int>      <dbl>       <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1         1    30     0.344         2.08    151.    24.2   5.35    10.5     4.31
## 2         2    28     0.314         1.13    155.    14.1   0.810   10.6     1.40
## 3         3     6     0.0661        1.78    229.    19.0   2.63    12.6     2.88
## 4         4    24     0.276         1.70    170.    16.8   1.63     6.05    2.02
## # ... with abbreviated variable names 1: mean.Total.Fat, 2: mean.Sodium,
## #   3: mean.Complex.Carbs, 4: mean.Dietary.Fiber, 5: mean.Sugar,
## #   6: mean.Protein
```

```
predictions %>%
  group_by(.pred_cluster) %>%
  summarize(mean.Total.Fat = mean(Total.Fat),
            mean.Sodium = mean(Sodium),
            mean.Complex.Carbs = mean(Complex.Carbs),
            mean.Sugar = mean(Sugar),
            mean.Protein = mean(Protein),
            counts = n())
```

```
## # A tibble: 4 x 7
##   .pred_cluster mean.Total.Fat mean.Sodium mean.Complex~1 mean.~2 mean.~3 counts
##   <fct>                  <dbl>       <dbl>          <dbl>   <dbl>   <dbl>  <int>
## 1 Cluster_1               1.76        200.           18.7    8.85    2.56     27
```

```
## 2 Cluster_2               2.3          29            26     10.2     4.8      10
## 3 Cluster_3               1.53        143.           15.1    9.18    1.79     39
## 4 Cluster_4               1.33        254.           23.8   11       4        12
## # ... with abbreviated variable names 1: mean.Complex.Carbs, 2: mean.Sugar,
## #   3: mean.Protein
```

```
table(ck_mclust$classification, predictions$.pred_cluster)
```

```
##
##     Cluster_1 Cluster_2 Cluster_3 Cluster_4
##   1         9        10         4         7
##   2         5         0        22         1
##   3         2         0         0         4
##   4        11         0        13         0
```

kclust 1 = mclust 4, kclust 2 = mclust 1, kCluster_3 = mclust 2, kclust 4 = mclust 3

Augment the dataset from part (d) with the information from the Gaussian mixture model. Produce a table showing the cluster assignments from the k-means vs. Gaussian mixture models.

```
(k_vs_m <- predictions %>% cbind(mclust = ck_mclust$classification) %>%
  mutate(kclust = case_when(.pred_cluster == "Cluster_1" ~ 4,
                            .pred_cluster == "Cluster_2" ~ 1,
                            .pred_cluster == "Cluster_3" ~ 2,
                            TRUE ~ 3)) %>%
  select(Cereal.Abb, kclust, mclust))
```

```
##        Cereal.Abb kclust mclust
## 1     AllBranBuds      4      1
## 2     AllBranOrig      1      1
## 3          AJacks      2      2
## 4          Basic4      3      1
## 5        ChexBlue      4      4
## 6      CrunchCapn      4      2
## 7        Cheerios      2      4
## 8      CheeriosOC      4      1
## 9       ChipsAhoy      2      2
## 10       ChexChoc      4      4
## 11         FFChoc      2      2
## 12       ChexCinn      4      4
## 13         FFCinn      2      2
## 14      CinnToast      4      4
## 15    KrispiCocoa      2      2
## 16   PebblesCocoa      2      2
## 17     CocoaPuffs      2      2
## 18    CookieCrisp      2      2
## 19       ChexCorn      4      4
## 20     CornFlakes      4      4
## 21       CornPops      2      1
## 22       OatBranC      1      1
## 23        Crispix      3      1
## 24    CrunchBerry      4      2
## 25       FiberOne      2      1
## 26    FrenchToast      2      2
## 27     FrootLoops      2      2
## 28         FFOrig      2      2
```

```
## 29        FMWBlue      1      1
## 30      FMWLBChoc      4      1
## 31      FMWLBOrig      1      1
## 32        FMWOrig      1      1
## 33       FMWStraw      1      1
## 34   PebblesFruit      2      2
## 35    GoldenCrisp      1      1
## 36   GoldenGraham      3      2
## 37    OreoOGolden      2      2
## 38      GrapeNuts      3      1
## 39 GrapeNutsFlake      4      1
## 40   GGCranAlmond      4      1
## 41        HBOOrig      2      4
## 42     HBOAlmonds      2      4
## 43       HBOStraw      2      4
## 44      HoneyMaid      4      2
## 45     CheeriosHN      2      2
## 46   ChexHoneyNut      4      2
## 47     FFHoneyNut      2      2
## 48       HoneyOhs      3      3
## 49      Honeycomb      2      2
## 50       Donettes      2      1
## 51    RaisinBranK      4      1
## 52            Kix      4      4
## 53      KraveChoc      2      1
## 54       LifeCinn      2      4
## 55       LifeOrig      2      4
## 56      LifeStraw      2      4
## 57    LifeVanilla      2      4
## 58    LuckyCharms      2      2
## 59     CheeriosMG      2      4
## 60    NillaBanana      2      2
## 61   NutterButter      2      4
## 62     CrunchOOPS      4      2
## 63      OreoOOrig      2      2
## 64       CrunchPB      4      4
## 65    RaisinBranP      3      1
## 66      OatSquare      4      1
## 67    RaisinBranC      4      1
## 68    RaisNutBran      4      1
## 69   RMCherAlmPec      1      1
## 70    ReesesPuffs      2      4
## 71       ChexRice      4      4
## 72     KrispiRice      2      4
## 73     ShWheatBig      1      1
## 74   ShWheatSpoon      1      1
## 75     SmartStart      3      1
## 76   SKChocAlmond      3      3
## 77   SKChocDelite      3      3
## 78     SKFruitYog      4      3
## 79         SKOrig      3      1
## 80   SKRedBerries      3      3
## 81    SKVanAlmond      4      3
## 82    KrispiStraw      2      2
```

```
## 83          Total      2      4
## 84           Trix      2      2
## 85    ChexVanilla      4      4
## 86      ChexWheat      3      1
## 87       Wheaties      4      4
## 88   FruitLoopsWB      2      2
```

Do the two models mostly agree on the clusters? If not, which types of cereals do they tend to agree about, and which do they not agree about?

```
sum(k_vs_m$kclust == k_vs_m$mclust)
```

```
## [1] 47
```

They agree on just over half of the cluster assignments when we compare clusters that are similar in size.

```
k_vs_m[which(k_vs_m$kclust == k_vs_m$mclust),]
```

```
##      Cereal.Abb kclust mclust
## 2   AllBranOrig      1      1
## 3        AJacks      2      2
## 5      ChexBlue      4      4
## 9     ChipsAhoy      2      2
## 10     ChexChoc      4      4
## 11       FFChoc      2      2
## 12     ChexCinn      4      4
## 13       FFCinn      2      2
## 14    CinnToast      4      4
## 15  KrispiCocoa      2      2
## 16 PebblesCocoa      2      2
## 17   CocoaPuffs      2      2
## 18  CookieCrisp      2      2
## 19     ChexCorn      4      4
## 20   CornFlakes      4      4
## 22     OatBranC      1      1
## 26  FrenchToast      2      2
## 27   FrootLoops      2      2
## 28       FFOrig      2      2
## 29      FMWBlue      1      1
## 31    FMWLBOrig      1      1
## 32      FMWOrig      1      1
## 33     FMWStraw      1      1
## 34 PebblesFruit      2      2
## 35  GoldenCrisp      1      1
## 37  OreoOGolden      2      2
## 45   CheeriosHN      2      2
## 47   FFHoneyNut      2      2
## 48     HoneyOhs      3      3
## 49    Honeycomb      2      2
## 52          Kix      4      4
## 58  LuckyCharms      2      2
## 60  NillaBanana      2      2
## 63    OreoOOrig      2      2
## 64     CrunchPB      4      4
## 69 RMCherAlmPec      1      1
## 71     ChexRice      4      4
```

```
## 73   ShWheatBig      1      1
## 74 ShWheatSpoon      1      1
## 76 SKChocAlmond      3      3
## 77 SKChocDelite      3      3
## 80 SKRedBerries      3      3
## 82  KrispiStraw      2      2
## 84         Trix      2      2
## 85  ChexVanilla      4      4
## 87     Wheaties      4      4
## 88 FruitLoopsWB      2      2
```

```
k_vs_m[which(k_vs_m$kclust != k_vs_m$mclust),]
```

```
##        Cereal.Abb kclust mclust
## 1     AllBranBuds      4      1
## 4          Basic4      3      1
## 6      CrunchCapn      4      2
## 7        Cheerios      2      4
## 8      CheeriosOC      4      1
## 21       CornPops      2      1
## 23        Crispix      3      1
## 24    CrunchBerry      4      2
## 25       FiberOne      2      1
## 30      FMWLBChoc      4      1
## 36   GoldenGraham      3      2
## 38      GrapeNuts      3      1
## 39 GrapeNutsFlake      4      1
## 40   GGCranAlmond      4      1
## 41        HBOOrig      2      4
## 42     HBOAlmonds      2      4
## 43       HBOStraw      2      4
## 44      HoneyMaid      4      2
## 46   ChexHoneyNut      4      2
## 50       Donettes      2      1
## 51    RaisinBranK      4      1
## 53      KraveChoc      2      1
## 54       LifeCinn      2      4
## 55       LifeOrig      2      4
## 56      LifeStraw      2      4
## 57    LifeVanilla      2      4
## 59     CheeriosMG      2      4
## 61   NutterButter      2      4
## 62     CrunchOOPS      4      2
## 65    RaisinBranP      3      1
## 66      OatSquare      4      1
## 67    RaisinBranC      4      1
## 68    RaisNutBran      4      1
## 70    ReesesPuffs      2      4
## 72     KrispiRice      2      4
## 75     SmartStart      3      1
## 78     SKFruitYog      4      3
## 79         SKOrig      3      1
## 81    SKVanAlmond      4      3
## 83          Total      2      4
## 86      ChexWheat      3      1
```

It's a bit hard to tell what they're agreeing on just from looking at what matches but it seems like they tend to agree on the moderately unhealthy cereals and tend not to agree on the very unhealthy and healthier cereals.