Lab 6

PSTAT131-231

Objectives

- Implement clustering methods in R
  - K-means
  - agglomerative hierarchical clustering
- Explore visualization techniques related to clustering
  - variable density plots
  - dendrograms

For this lab you'll work with a small subset of the CDC's Behavioral Risk Factor Surveillance System (BRFSS) data. The CDC collects this data annually as part of a public health monitoring effort. The data come from phone surveys in which respondents answer questions about their health, diet, and lifestyle.

You'll look for respondent clusters based on just 12 of a very large number of variables in the overall dataset:

- `sleep`: hours of sleep
- `children`: number of children
- `height`: height in centimeters
- `weight`: weight in grams
- `alcohol`: total number of alcoholic drinks per month
- `fruit_juice`: daily fruit juice intake
- `fruit`: daily fruit intake
- `beans`: daily bean intake
- `greens`: daily intake of dark greens
- `vegetables`: daily vegetable intake
- `maxvo2`: age-gender-specific maximum oxygen consumption
- `strength_exercise`: strength activity frequency per week

A subsample of the 2013 data comprising 3944 respondents will be used for the lab.

```
# import data
load('data/brfss.RData')
```

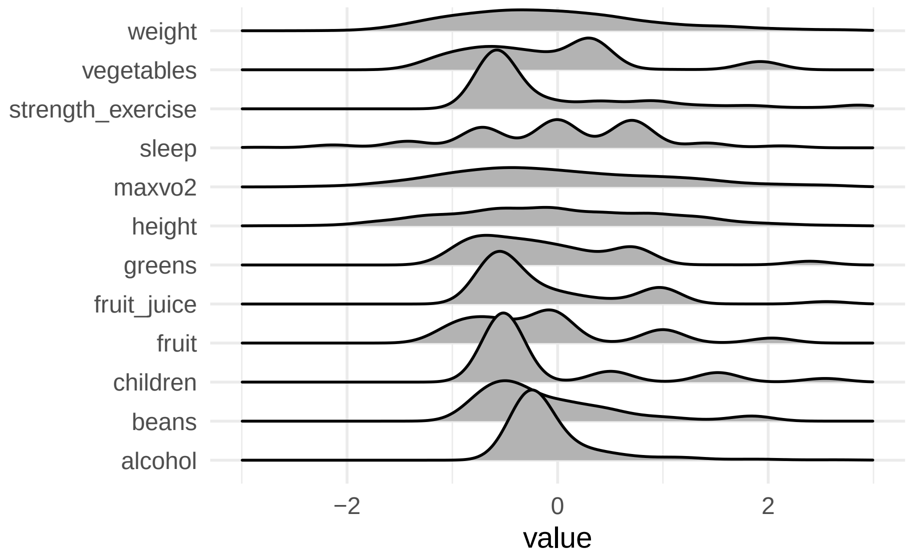
Visualizations

In general, it's a good idea to standardize (center and scale) data before clustering. However, it is also helpful to do this in order to visualize variable distributions.

```
# center and scale data
brfss_std <- brfss %>% scale() %>% as.data.frame()
```

The `ggridges` package includes a `geom_density_ridges` to compactly plot several densities, which can take on the appearance of staggered ridges (the documentation says, "as in the cover of the famous Joy Division album Unknown Pleasures).

```
# ridge plot
brfss_std %>%
  gather(key = 'variable', value = 'value', 1:12) %>%
  ggplot(aes(y = variable, x = value)) +
  geom_density_ridges(bandwidth = 0.2) +
  theme_minimal() +
  xlim(c(-3, 3)) +
  labs(y = '')
```

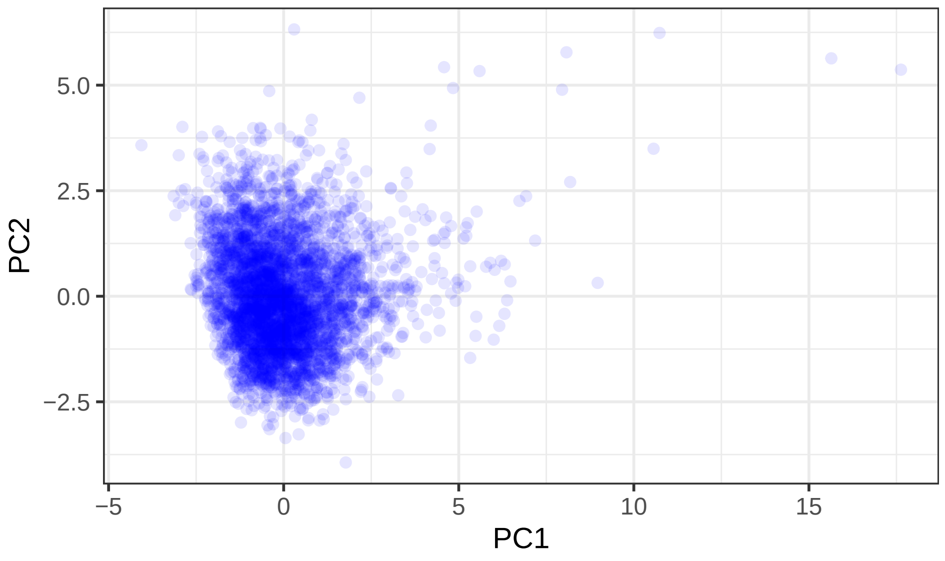


Take a moment to examine the plot and consider whether it suggests that there may be subgroups in the dataset.

Another visualization strategy is to look for clusters in a scatterplot of the first two principal components.

Your turn (1) Compute the first two principal components from the standardized data, and make a scatterplot showing the data projected onto these directions (*i.e.*, a scatterplot of the first two PCs). Store the loadings and a data frame or tibble of the first two PCs (you'll need them later).

```
# compute loadings for first two pcs
pc_svd <- svd(brfss_std)
pc_loadings <- pc_svd$v[, 1:2]

# compute principal components
brfss_pc <- data.frame(as.matrix(brfss_std) %*% pc_loadings)
colnames(brfss_pc) <- paste('PC', 1:2, sep = '')

# scatterplot
brfss_pc %>%
  ggplot(aes(x = PC1, y = PC2)) +
  geom_point(color = "blue", alpha = 0.1) +
  theme_bw()
```



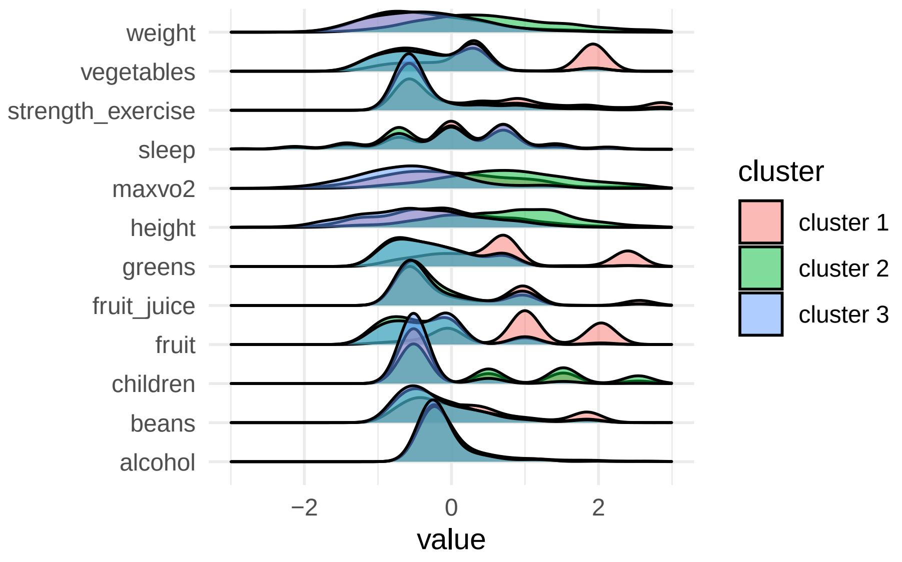
K-means clustering

To start with, compute K-means clusterings from the standardized data with $K = 3$ clusters.

```
# kmeans with k = 3
kmeans_out <- kmeans(brfss_std, centers = 3, nstart = 5)
clusters <- factor(kmeans_out$cluster,
                   labels = paste('cluster', 1:3))
centers <- kmeans_out$centers
```

To visualize the result, add the cluster labels to the dataset and map one of the aesthetic layers to the cluster labels. For example, the ridge plot can be modified as follows:

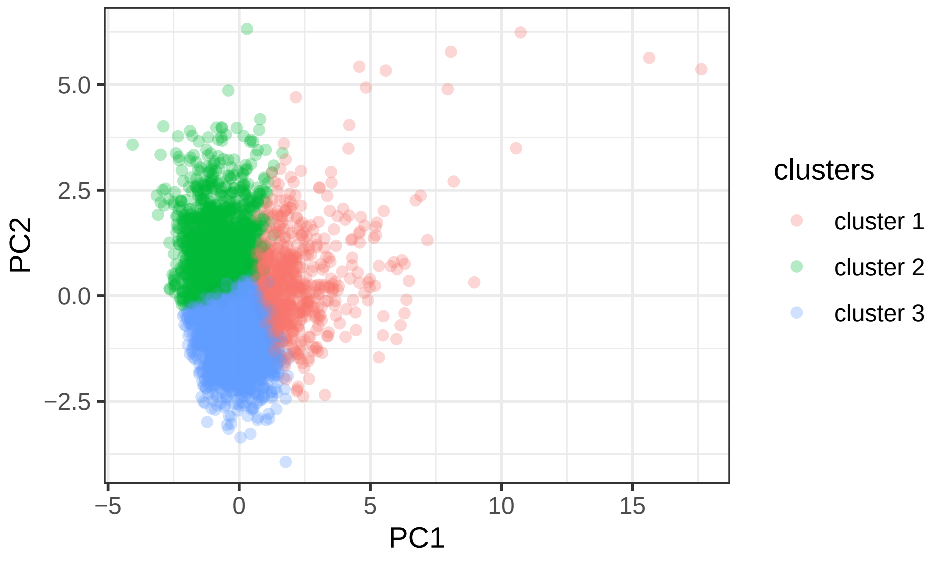
```
# ridge plot with clusters
brfss_std %>%
  mutate(cluster = clusters) %>%
  gather(key = 'variable', value = 'value', 1:12) %>%
  ggplot(aes(y = variable, x = value)) +
  geom_density_ridges(aes(fill = cluster),
                      bandwidth = 0.2,
                      alpha = 0.5) +
  theme_minimal() +
  xlim(c(-3, 3)) +
  labs(y = '')
```



Take a moment to consider whether it seems like the clusters have captured any meaningful distinctions between respondents based on the above plot. If so, how might you describe the distinctions?

Your turn (2) Now add the cluster labels to the scatterplot of the data along the first two principal components. Store your plot as `p` (so we can modify it in the next step).

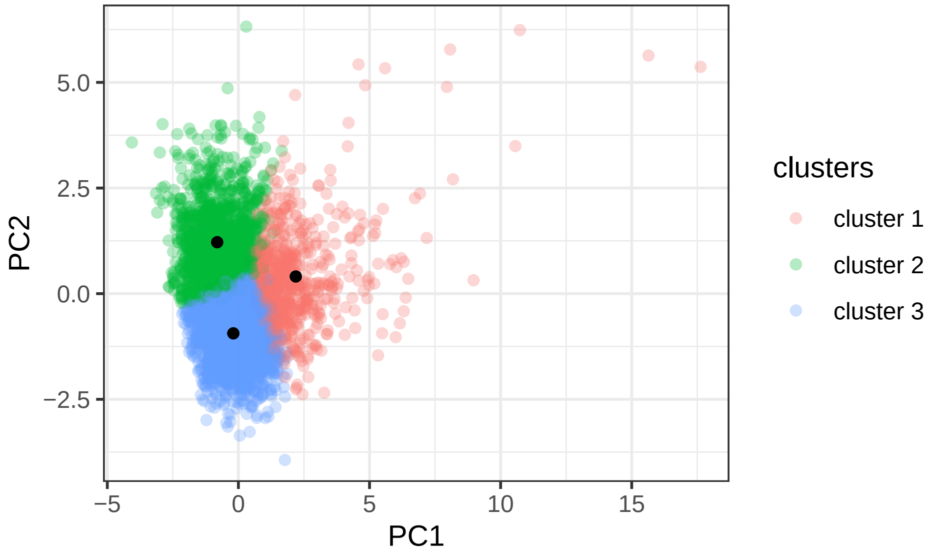
```r
# pc-based visualization of clusters
p <- brfss_pc %>%
  mutate(cluster = clusters) %>%
  ggplot(aes(y = PC2, x = PC1)) +
  geom_point(aes(col = clusters), alpha = 0.3) +
  theme_bw()
p
```



The center points can be projected onto the first two principal components using the loadings, so that they can be displayed on the scatterplot.

```r
# project centers onto PCs
kmeans_ctrs <- centers %*% pc_loadings %>%
  as.data.frame()
```

```
colnames(kmeans_ctrs) <- paste('PC', 1:2, sep = '')

# add to plot
p + geom_point(data = kmeans_ctrs)
```

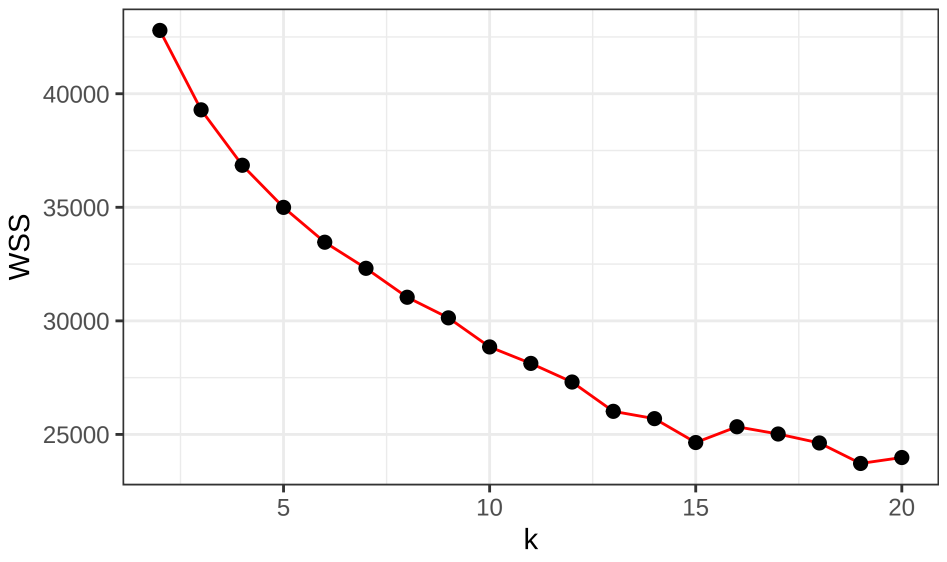


It was indicated in lecture that a simple heuristic for choosing K is to compute clusterings for a sequence of K and plot the within-cluster sum of squares against K. One way to do this using the `sapply` function is shown below:

```
# compute wss for a sequence of k
k_seq <- 2:20
set.seed(22021)
wss <- sapply(k_seq, function(k){
  kmeans(brfss_std,
         centers = k,
         nstart = 5,
         iter.max = 15)$tot.withinss
})
```

Your turn (3) Now use `wss` from the previous chunk to plot the within-cluster sum of squares against K. Is there an obvious choice of K based on the plot?

```
# plot of wss against k
data.frame(k = k_seq, WSS = wss) %>%
  ggplot(aes(x = k, y = WSS)) +
  geom_line(color = "red") +
  geom_point(size = 2) +
theme_bw()
```



Hierarchical clustering

Hierarchical clustering is implemented using `hclust`. Since `hclust` requires as input a matrix of distances, first compute the pairwise distances between points using `dist`:

```
# compute distances between points
d_mx <- dist(brfss_std, method = 'euclidean')

# compute hierarchical clustering
hclust_out <- hclust(d_mx, method = 'complete')
```

For the sake of having a comparison with the K-means clustering in the previous part, select three clusters from the sequence.

```
# cut at 3 clusters
clusters <- cutree(hclust_out, k = 3) %>%
  factor(labels = paste('cluster', 1:3))
```

But notice that with the usual choice of similarity measure and linkage, the method simply picks out two outliers, and leaves the rest of the data in one cluster.

```
# count number of data points per cluster
tibble(clusters) %>% count(clusters)
```

```
## # A tibble: 3 x 2
##   clusters      n
##   <fct>     <int>
## 1 cluster 1  3942
## 2 cluster 2     1
## 3 cluster 3     1
```

So change the linkage. Ward's minimum variance linkage performs better in this situation:

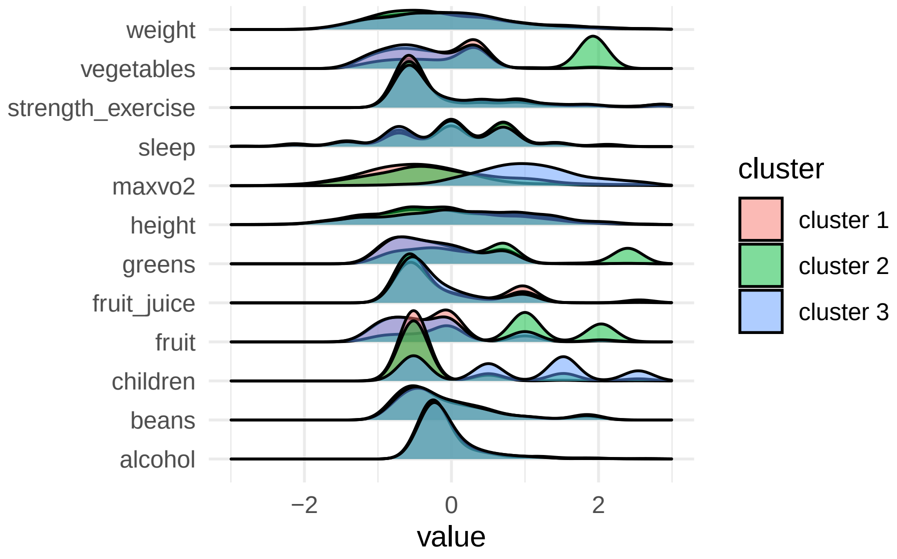
```
# change linkage
hclust_out <- hclust(d_mx, method = 'ward.D')

# cut at 3 clusters
clusters <- cutree(hclust_out, k = 3) %>%
  factor(labels = paste('cluster', 1:3))
```

```r
# count number of data points per cluster
tibble(clusters) %>% count(clusters)
```

```
## # A tibble: 3 x 2
##   clusters      n
##   <fct>     <int>
## 1 cluster 1  2174
## 2 cluster 2   742
## 3 cluster 3  1028
```

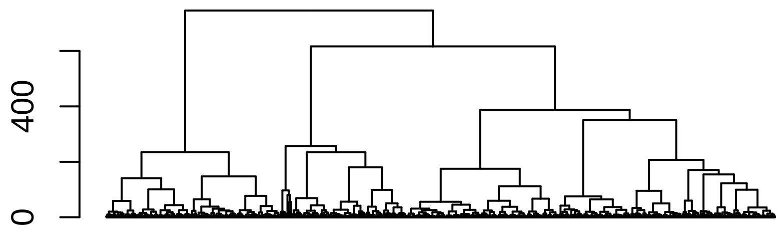
Your turn (4) Choose one of the visualizations from the previous parts and plot the clusters from hierarchical clustering.

```r
# plot clusters
brfss_std %>%
  mutate(cluster = clusters) %>%
  gather(key = 'variable', value = 'value', 1:12) %>%
  ggplot(aes(y = variable, x = value)) +
  geom_density_ridges(aes(fill = cluster),
                      bandwidth = 0.2,
                      alpha = 0.5) +
  theme_minimal() +
  xlim(c(-3, 3)) +
  labs(y = '')
```



Another visualization option is the *dendrogram*, a tree-based representation of the full sequence of clusterings. (This may take a moment or so to run.)

```r
# plot dendrogram
hclust_out %>%
  as.dendrogram() %>%
  set('labels', NULL) %>%
  plot()
```



Your turn (5) Choose one additional dendrogram visualization from this list of examples. If need be, you can downsample the data (`slice_sample()`) so that your plot doesn't take too long to construct.

```
library(ape)
colors = c("red", "blue", "green", "black")
clus3 = cutree(hclust_out, 3)
plot(as.phylo(hclust_out), type = "fan", tip.color = colors[clus3],
     label.offset = 1, cex = 0.7)
```