Lab #4 – Unsupervised Learning

- The only package we will be using today is *ISLR* which should be installed on your computer!

Useful commands

Command	Explanation
table()	Counts the number of observations for each category in the variable (do not use on continuous variables!)
prcomp(data, scale)	Runs PCA on the data. Scale indicates whether the function should scale the data.
rainbow(n)	Generate n different colors
scale(data)	Scale data to have mean 0 and sd 1
dist(data)	Calculates distances of every two observations in data
hclust(dist_data, method)	Runs hierarchical clustering with the determined method. dist_data must be of class *dist*
cutree(hc_model, k)	Divides observation into k clusters using the hierarchical model
kmeans(data, k)	Runs k-means clustering

Example code for classification:

```
library(ISLR)
full_dat <-NCI60
dat <- full_dat$data
labs <- full_dat$labs
# PCA
pca_model <- prcomp(dat, scale=TRUE)
# calculate proportion of variance explained
ve <- pca_model$sdev^2   # variance explained
pve <- ve/sum(ve)   # proportion variance explained
cpve <- cumsum(pve)  # cumulative proportion variance explained
# plot proportion of variance explained
plot(pve, type="o")
plot(cpve, type="o")
# generate colors for each label
cols <- rainbow(length(unique(labs)))
obs_cols <- cols[as.numeric(as.factor(labs))]
# plot observations on first two PCs
plot(x=pca_model$x[,1], y=pca_model$x[,2], col = obs_cols)

# scale data
scaled_dat <- scale(dat)
# convert to dist object
dist_dat <- dist(scaled_dat)
# HIERARCHICAL CLUSTERING
hc_model <- hclust(dist_dat, method = "complete")
```

```
# plot dendrogram
Plot(hc_model, labels=labs)
# cluster observations into 4 clusters
hc_clusters <- cutree(hc_model, 4)

set.seed(2)
# run k-means
km <- kmeans(scaled_dat, clusters=4)
# plot clusters on first two PCs
plot(x=pca_model$x[,1], y=pca_model$x[,2], col = km$cluster+1)
```

Exercise:

We will implement PCA and two clustering methods – hierarchical and k-means – on genomic data regarding cancer cells.

1. Load the *ISLR* libraray
2. Load the NCI60 data by running: *full_dat <- NCI60*
3. Learn your data:
   a. Run *?NCI60*
   b. What is the format of the data? Notice that *NCI60* is a list containing two elements – data and labels.
      Since we are in unsupervised learning, **we do not use the labels for the model**. The labels will be used to evaluate the outcome of the model.
   c. What is the size of the data? (i.e. the size of *full_dat$data*, see comment above!)
   d. Use the *str* function in order to take a glimpse at the data
   e. How many different labels are there? Use *table* to learn about the different labels you have (the labels are stored in *full_dat$labs*)

4. PCA

   a. Run PCA using the *prcomp* function. Remember that the data should be scaled before running PCA!
      * Hint: set *scale=TRUE* when running *prcomp*
   b. Explore the PCA object – What does it contain? What is the meaning of each value?
      * Hint: run *names(pca.model)*; run *?prcomp*
   c. Extract the standard deviation of each of the PCs. How much of the variance is explained by each of the PCs?
      * Hint: the sd of each of the PCs is stored in *pca.model$sdev*. To get the proportion of variance explained, just normalize by the sum of all the PCs' variance
   d. Calculate the cumulative proportion of variance explained by the PCs. How many PCs are needed to explain 50%/75%/100% of the variance in the data?

* Hint: use *cumsum* to get the cumulative sum of the proportion of variance explained

e. Plot the cumulative proportion of variance explained by the PCs as a function of the number of PCs (i.e. the cumulative sum on the y axis and the number of PCs on the x axis)

f. Let's see how the data looks in the PC space. Plot the PC scores each observation got on the first two PCs.

   * Hint: the scores are stored in *pca.model$x*, where each column represents a different PC. The columns are ordered, i.e. the first column is the first PC, the second column is the second PC, etc. Set the first column in *pca.model$x* to be the x axis and the second column to be the y axis

g. So far we have not used the labels. It would be interesting to see the distribution of the labels when looking at the first two PCs.

   i. Run the following code in order to generate a different color for each label type, and color the observations with the color of their label:

   ```
   cols <- rainbow(length(unique(labs)))
   obs_cols <- cols[as.numeric(as.factor(labs))]
   ```

   (*labs* contains the labels, change it according to the name you gave the labels, or simply replace it with *full_dat$labs*)

   ii. Color the observations in the plot you made in f by setting *col=obs_cols* in the plotting function. Can you see anything interesting?

5. Hierarchical Clustering:

Recall that in clustering methods we measure distances between observations and divide observations into cluster. Many distance measures can be used, depending on the data and on the question. We will use Euclidian distance.

a. Remember that Euclidian distance is sensitive to the scale of each vriable. Scale the data in order for all the variables to have the same scale.

   * Hint: run *scale(dat)*

b. Use the *dist* function in order to calculate the distances between every two observations.

c. Run hierarchical clustering using complete linkage (recall, complete linkage means that the distance between two clusters is the maximum distance of all pairs of observations from the clusters).

   * Hint: use the *hclust* function for hierarchical clustering and set *method="complete"*

d. Plot the dendrogram of the model

   * Hint: use the *plot* function on *hc_model* and set *labels=labs*, in order to identify the label of each of the observations.

e. Recall that hierarchical clustering offers many options of clustering, depending on the desired number of clusters. Choose a number and find how the model divides the observations into these clusters.

   * Hint: use the *cuttree(hc_model, number)*

f. How is this division obtained from the dendrogram?

6. K-Means

a. Run *kmeans* on the scaled data from 5a. and set the k it should use.

   * Hint: set *clusters=k* in the *kmeans* function

b. Explore the resulting kmeans object – What did the function return? What does each of its values?

   * Hint: run *?kmeans*

c. How much are the clusters diverse?

   i. Extract the within-cluster sum of squares

   ii. Extract the between-cluster sum of squares

   * Hint: both of these are contained in the output from the *kmeans* function

d. Extract the clusters of each observation

   * Hint: same exact syntax as e. just <u>do not</u> set *type=class*

e. Return to the plot from 4f. Color the observations using the k-means clusters.

   * Hint: set *col=km$cluster+1*