# Lab_05R

36-600

Fall 2022

This lab will be a relatively short one, given that your EDA Project report is due at the same time this lab is due (next Tuesday at 8:35 AM).

To answer the questions below, it will again help you to refer to Sections 10.3 and 10.5 of ISLR; it might also help you to refer to your previous lab work (and, as always, to Google).

## Question 1

Like we did last time, let's create a fake dataset.

```
set.seed(505)
(df <- data.frame(x=runif(3),y=runif(3),z=runif(3)))
```

```
##            x          y         z
## 1 0.13110363 0.09073694 0.1741910
## 2 0.50395610 0.02070792 0.1276750
## 3 0.09991132 0.78558310 0.3049695
```

Compute and show the pairwise distance matrix for the *scaled* data. (This is what you would pass into `hclust()`, so there is a reason for doing this.) You should see a lower-triangular matrix as output, with three values between 1 and 4.) Note the smallest value.

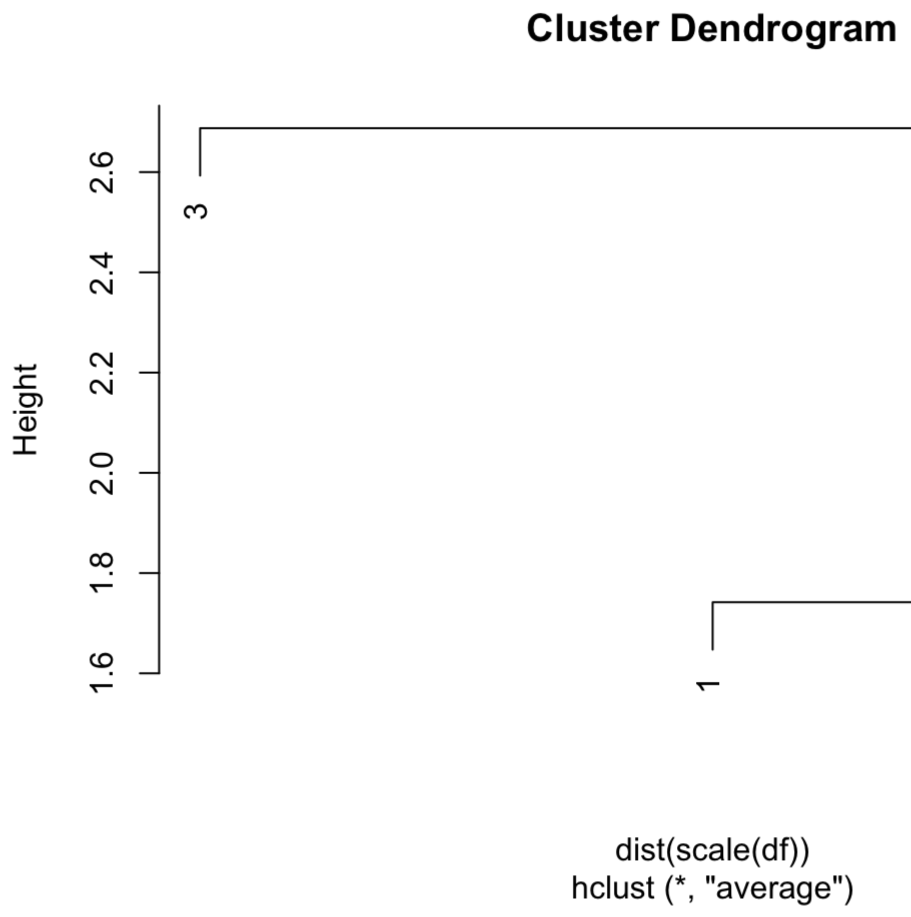
```
dist(scale(df))
```

```
##          1        2
## 2 1.741874
## 3 2.178001 3.197215
```

## Question 2

Now run hierarchical clustering on `df` with average linkage, and plot the dendrogram. What you *should* see is that the height of the first merge is (visually, roughly) is the same as that smallest distance value you saw above. You can confirm this by looking at the `height` element of the list output by `hclust`. This makes sense: the smallest dissimilarity between our data points is exactly the smallest observed Euclidean distance between the points.

```
hc.out <- hclust(dist(scale(df)),method="average")
plot(hc.out)
```



```
hc.out$height
```

```
## [1] 1.741874 2.687608
```

## Dataset

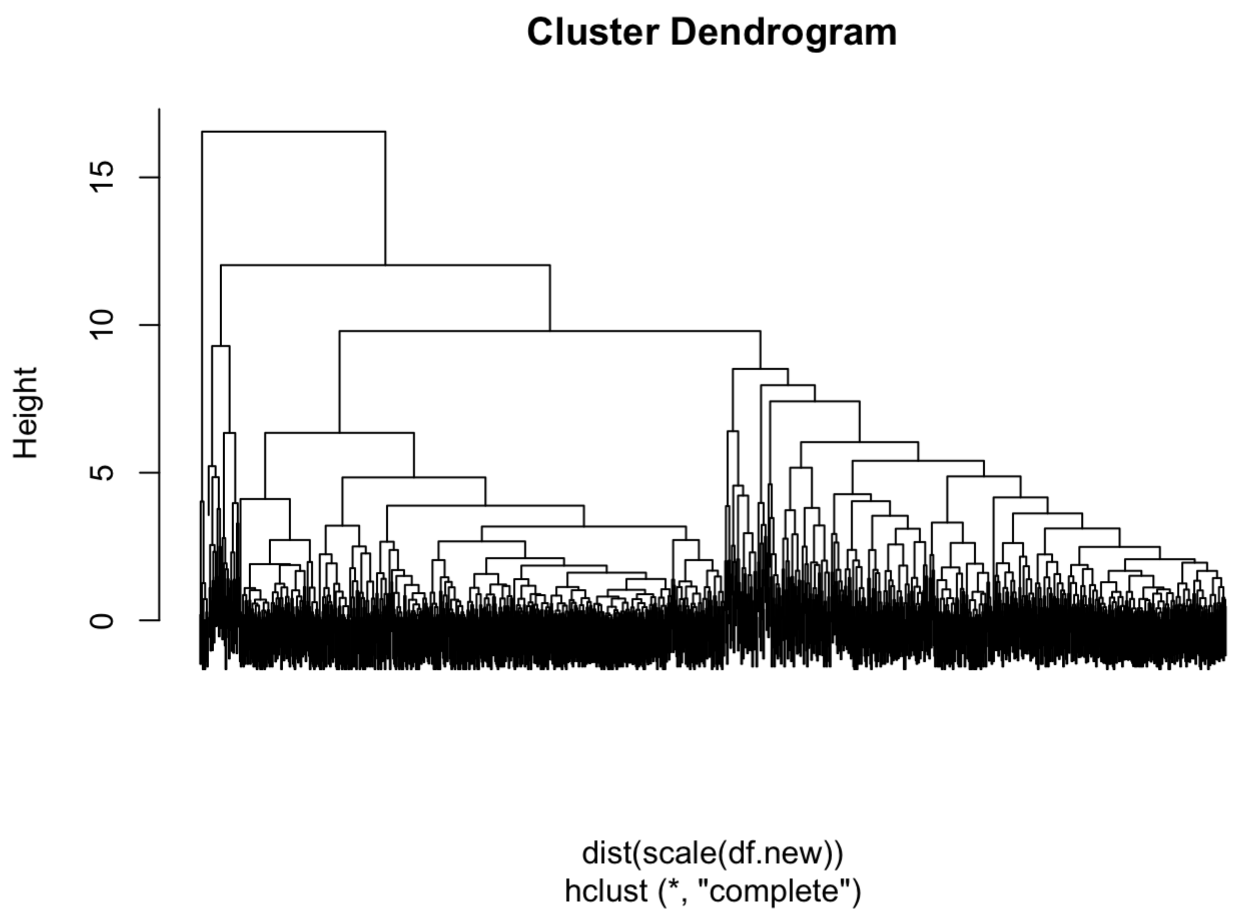
Let’s import the same stellar dataset we used in the previous lab.

```
file.path <- "https://raw.githubusercontent.com/pefreeman/36-290/master/EXAMPLE_DATASET
S/DRACO/draco_photometry.Rdata"
load(url(file.path))
df <- data.frame(ra,dec,velocity.los,log.g,mag.g,mag.r,mag.i)
rm(file.path,ra,dec,velocity.los,log.g,temperature,mag.u,mag.g,mag.r,mag.i,mag.z,metalli
city,signal.noise)
suppressWarnings(library(tidyverse))
df %>%
  filter(.,ra<264 & dec>56 & velocity.los>-350 & velocity.los< -250) %>%
  mutate(.,col.gr=mag.g-mag.r,col.ri=mag.r-mag.i) %>%
  select(.,-mag.g,-mag.r,-mag.i,-velocity.los) -> df.new
```

## Question 3

Use the `hclust()` function to build a hierarchical clustering tree for `df.new`, and use the basic `plot()` function to display the dendrogram. Try both complete and average linkage: which one makes for the best-looking output? (This should not be confused with: which one gives the best clustering result? Note: there is no "right" answer here; best-looking is in the eye of the statistical consultant.) Despite talking up the dendrogram in class, is this actually useful output here? Why or why not? If your client asked for a dendrogram, what step might you want to consider taking before providing one? (Note: when calling `plot()`, consider passing the argument `labels=FALSE` to remove the row numbers at the base of dendrogram.)

```
hc.out <- hclust(dist(scale(df.new)),method="complete") # for example
plot(hc.out,labels=FALSE)
```



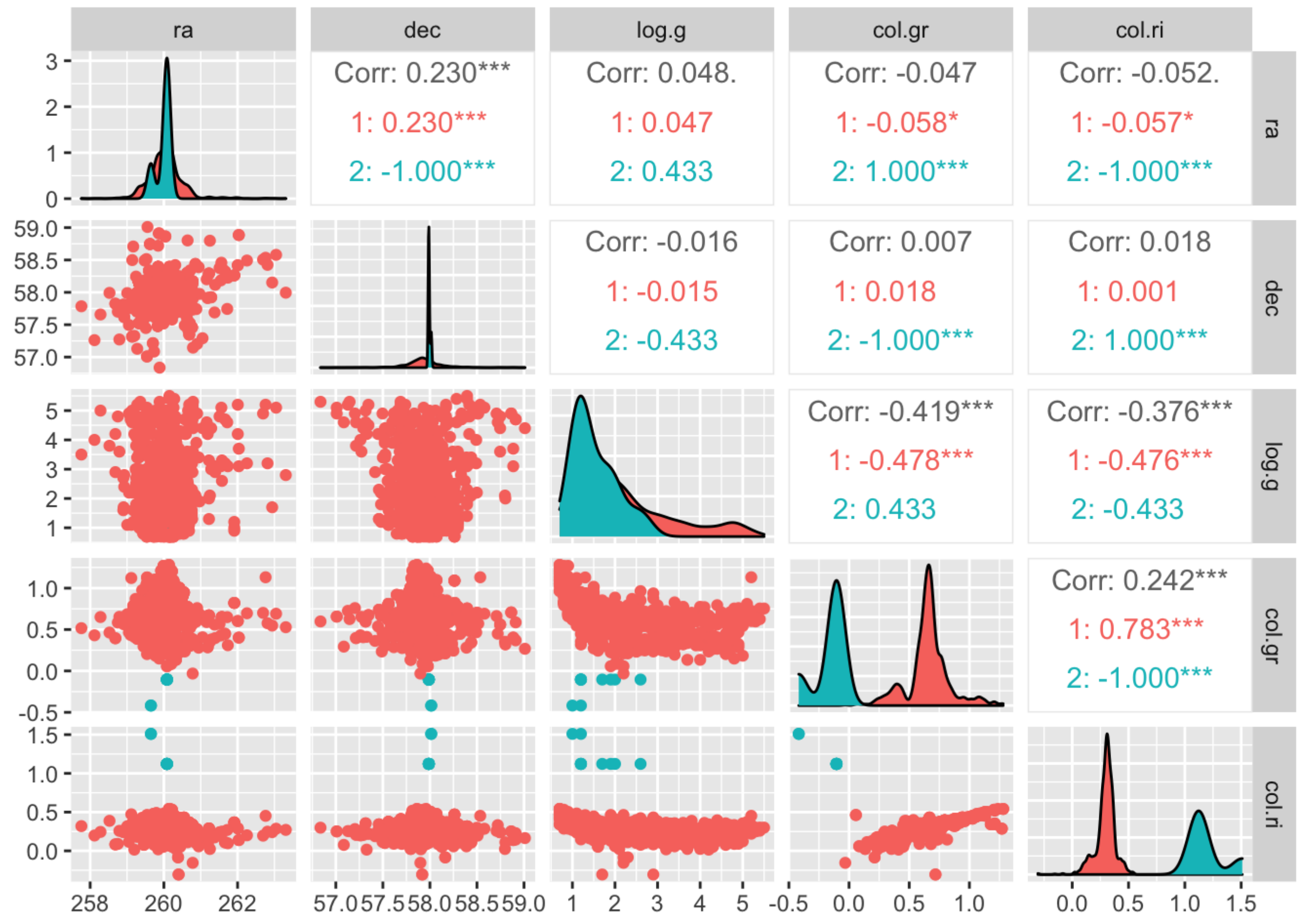
```
I found the complete-linkage dendrogram to offer better-looking output than average link
age. The dendrogram is a bit "busy" (to say the least): it would look better if there we
re far fewer observations. So, before giving this dendrogram to a client, I would consid
er taking (a) constructing dendrogram using only a subset of the observations, or (b) if
possible, use the full dendrogram structure, but only show the branching for a subset of
the observations. (I say "if possible" because I haven't tried to do it.)
```

## Question 4

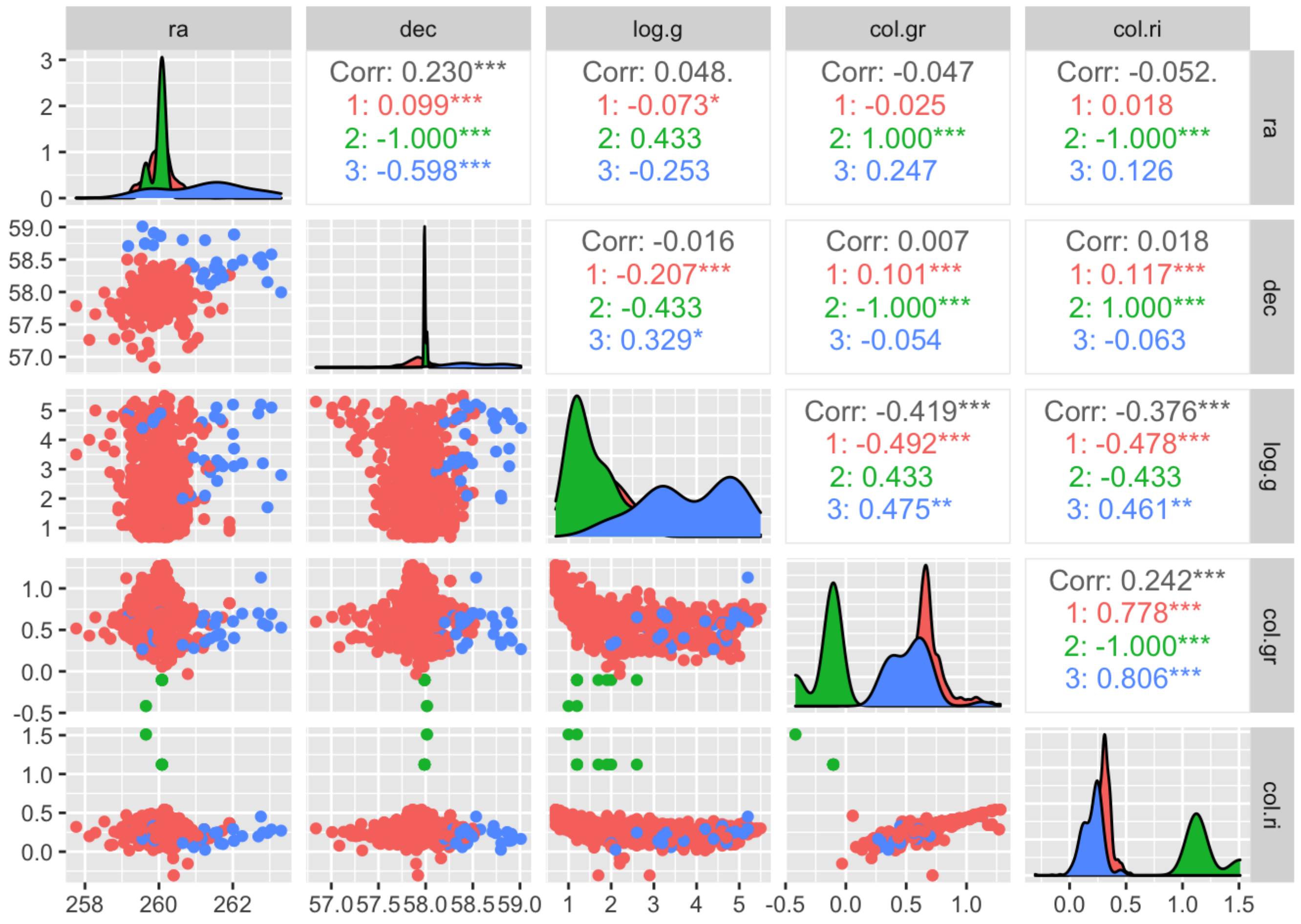
Use the `cutree()` function to map each observation to a cluster, then use `ggpairs()` to display the clusters in a similar manner as above for K-means. Assume the same number of clusters as you did for K-means. Does the output look the same or different from K-means? Is this what you expected? Why or why not? (Hint:

if `cluster` is the output from `cutree()`, then `color=factor(cluster)` will properly color each of the points.) Visualizing the output of hierarchical clustering in this manner (rather than using a dendrogram) is better when the sample size is large.

```
library(GGally)
cluster2 <- cutree(hc.out,2)
ggpairs(df.new,progress=FALSE,mapping=aes(color=factor(cluster2)))
```



```
cluster3 <- cutree(hc.out,3)
ggpairs(df.new,progress=FALSE,mapping=aes(color=factor(cluster3)))
```



```
The clusters look quite different, in the sense that what were the two dominant clusters
above pretty much merge
into one cluster here, leaving two small rump clusters. Expectation: because of the limi
tation imposed by the clustering being hierarchical (clusters within clusters), I would
have expected the output to differ.
```

## Question 5

Implement a GMM-based analysis using the `ClusterR` package, analogous to what is in the notes. Assume *two* clusters. Your final goal is to figure out the proportions of the observations that can be confidently placed in either Cluster 1 or Cluster 2 (cluster probabilities >0.95). The placement of the rest of the observations can be considered ambiguous. As a reminder, one of the outputs from `predict_GMM` is `cluster_proba` . Here, that will be a 1218 x 2 matrix, where the probabilities on each row sum to 1. So, determine how many values in the first column of `cluster_proba` are either <0.05 (the datum is to be associated with *Cluster 2* with high probability) or >0.95, then divide by the number of rows in `cluster_proba` . (Note that I found ≈ 48.5% of the data can be confidently placed in one of the two clusters.)

(Note: you will have to install `ClusterR` before completing this question.)

```
library(ClusterR)
```

```
## Loading required package: gtools
```

```
gmm.out <- GMM(df.new,gaussian_comps=2)
pred    <- predict_GMM(df.new,gmm.out$centroids,gmm.out$covariance_matrices,gmm.out$weig
hts)
p       <- pred$cluster_proba[,1]
sum(p<0.05|p>0.95)/length(p)
```

```
## [1] 0.7791461
```