Model 3 by Christine Pallon

2022-12-14

First clustering method: KMeans

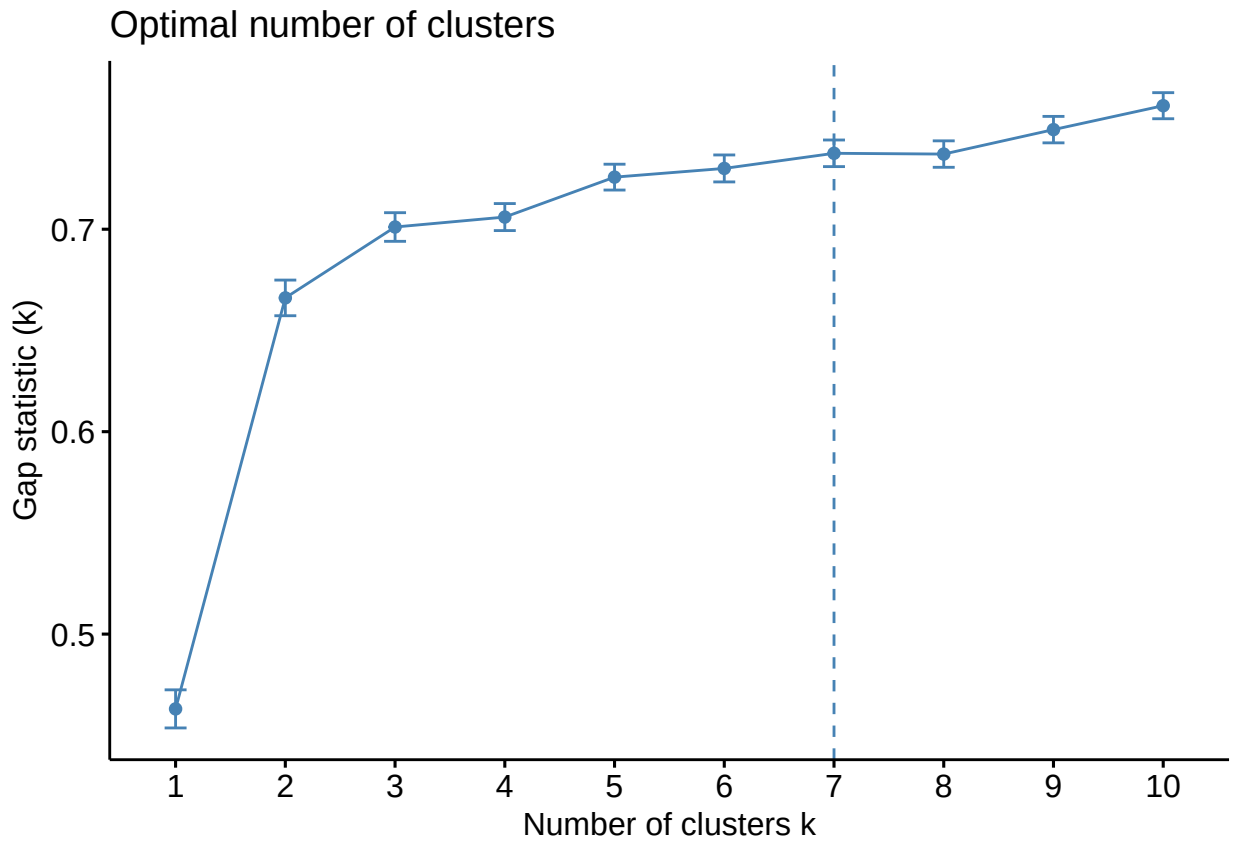
Preparing the data for Kmeans clustering

For this model, we will be comparing three clustering methods – **KMeans, hierarchical, and model-based** clustering - with the **radiomics** dataset. Let's start with KMeans. First, we load and prepare the dataset, omit any nulls, and then scale the dataset.

```
radiomics <-read.csv("radiomics_completedata.csv")
radiomics <- radiomics %>% select(-Institution, -Failure, -Failure.binary)
radiomics <- na.omit(radiomics)

radiomics <- scale(radiomics)
```

Determining optimal cluster numbers

Before we start with our clusters, let's figure out what our optimal number of clusters (or k) would be. We could use the elbow, silhouette, or gap statistic methods to determine this. Let's use the gap statistic method and plot the results:



NOTE - *I get a slightly different k value every time I run the above code and it changes every time I knit to PDF. Sometimes it is 6, sometimes it is 7, 8, or 9. For the sake of this demonstration, I will be going with 6, since that is what I got when I ran the code to test, but the number you see in the PDF output may be different.*

We'll use 6 clusters for our KMeans clustering since that seems to be our optimal number based on the gap statistic, but we'll also take a look at what 2 and 4 clusters look like as well just to compare.

Creating and plotting clusters

We use the **kmeans()** function to create our clusters with 2, 4, and 6 centers respectively.

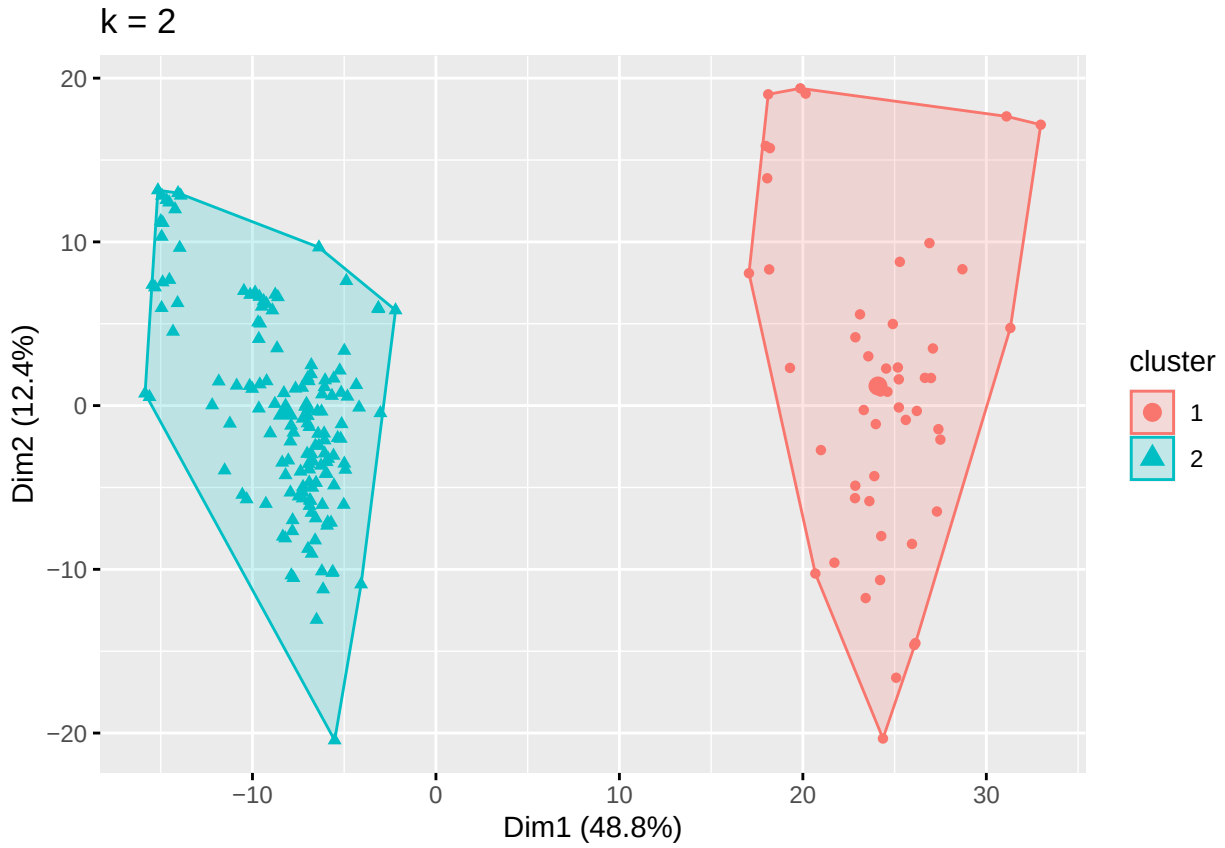
```
two_clusters <- kmeans(radiomics, centers = 2, nstart = 10)
four_clusters <- kmeans(radiomics, centers = 4, nstart = 10)
six_clusters <- kmeans(radiomics, centers = 6, nstart = 10)
```

Then, we make our three plots:

```
p1 <- fviz_cluster(two_clusters, geom = "point", data = radiomics) + ggtitle("k = 2")
p2 <- fviz_cluster(four_clusters, geom = "point",  data = radiomics) + ggtitle("k = 4")
p3 <- fviz_cluster(six_clusters, geom = "point",  data = radiomics) + ggtitle("k = 6")
```

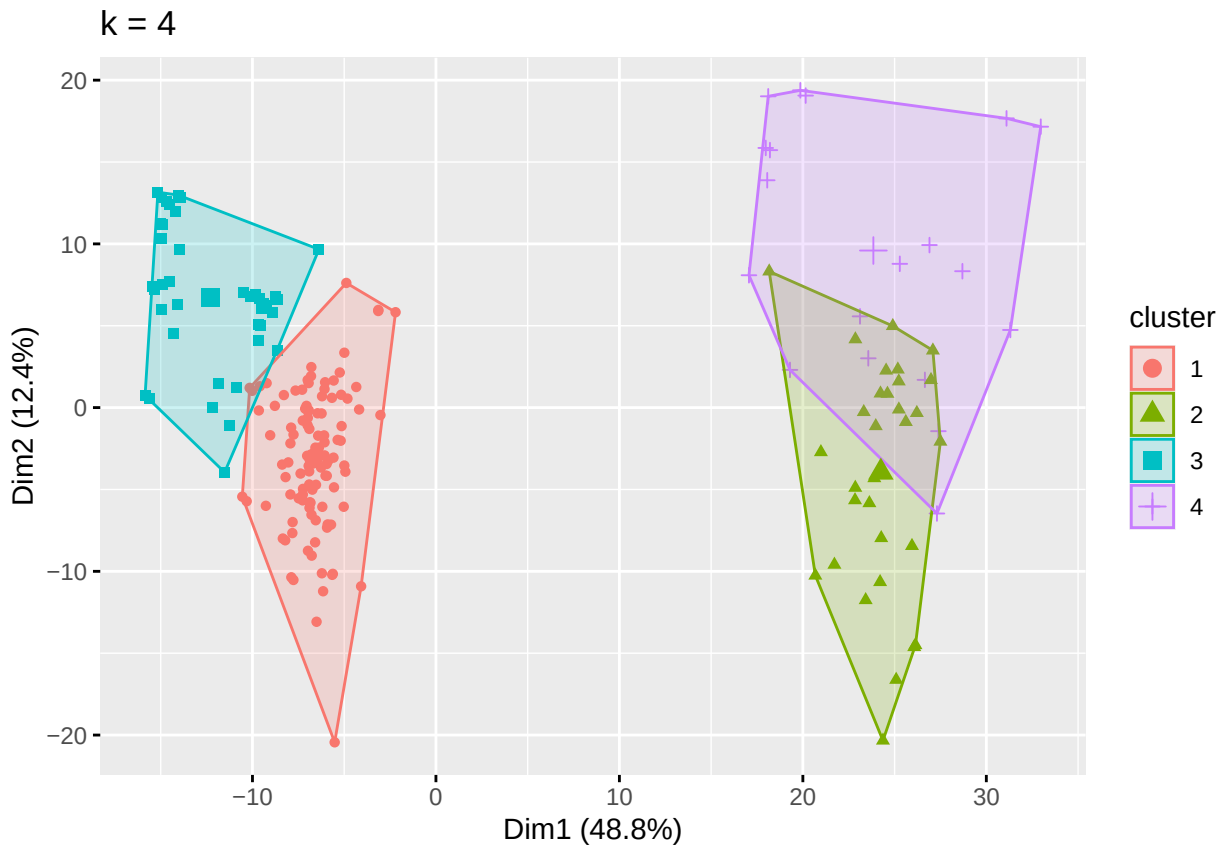
K = 2

```
plot(p1)
```



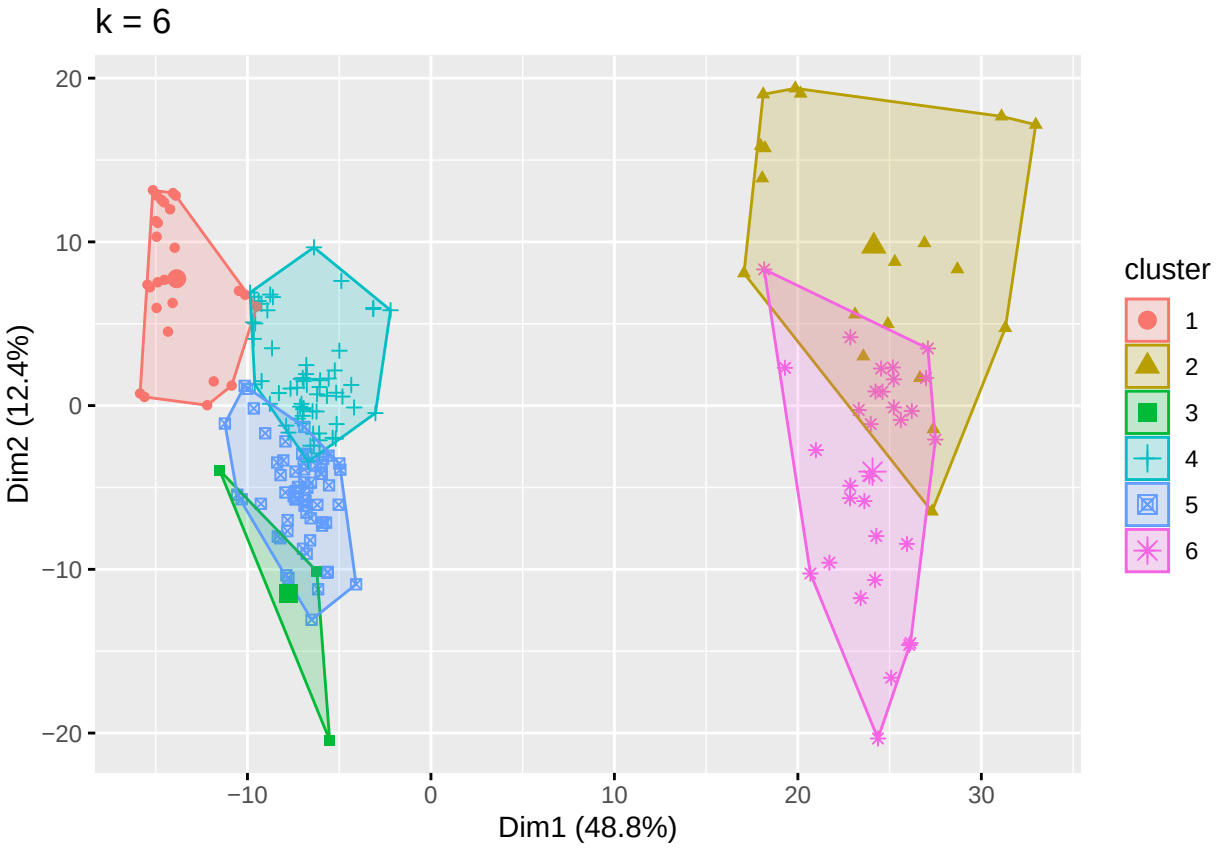
K = 4

```
plot(p2)
```



K = 6

```
plot(p3)
```



Comparing the sizes of the clusters

We can access the size of each cluster and see how many observations are in each one:

```
two_clusters$size
```

```
## [1]  50 147
```

```
four_clusters$size
```

```
## [1] 107  31  40  19
```

```
six_clusters$size
```

```
## [1] 26 19  3 59 59 31
```

Storing the centers in a data frame

Finally, we can also store the centers of our k = 6 clustering method into a data frame so that we can compare the z-scores of various features between the clusters.

```
k6_centers <- as.data.frame(six_clusters$centers)
cluster <- c(1:6)
center_k6 <- data.frame(cluster, k6_centers)
head(center_k6[, c(1, 2, 3)])
```

```
##   cluster Entropy_cooc.W.ADC GLNU_align.H.PET
## 1       1         0.17469288      0.122117770
## 2       2         0.42357307      0.009553357
## 3       3         0.20817813      0.064665940
## 4       4         0.06185452      0.096160211
## 5       5        -0.19048629     -0.086304449
## 6       6        -0.18145687     -0.133292373
```

Second clustering method: Hierarchical

Now let's move onto a hierarchical method of clustering! Like we did before, let's make sure our dataset is ready to go for clustering.

```
radiomics <-read.csv("radiomics_completedata.csv")
radiomics <- radiomics %>% select(-Institution, -Failure, -Failure.binary)
radiomics <- na.omit(radiomics)

radiomics <- scale(radiomics)
```

Determining method

Before we jump into clustering, we need to determine which method we want to use. To do this, we'll use the below function:

```
# The methods we're going to assess
m <- c( "average", "single", "complete", "ward")
names(m) <- c( "average", "single", "complete", "ward")


# Our function to compute the coefficients
ac <- function(x) {
  agnes(radiomics, method = x)$ac
}

#Accessing our coefficients

purrr::map_dbl(m, ac)
```

```
##   average    single  complete      ward
## 0.7654759 0.7145135 0.8514456 0.9659881
```

It looks like **Ward's method of minimum variance** is the one we want to go with, since its value is closest to 1 out of the four methods we evaluated.

Using Ward's method to determine our clusters

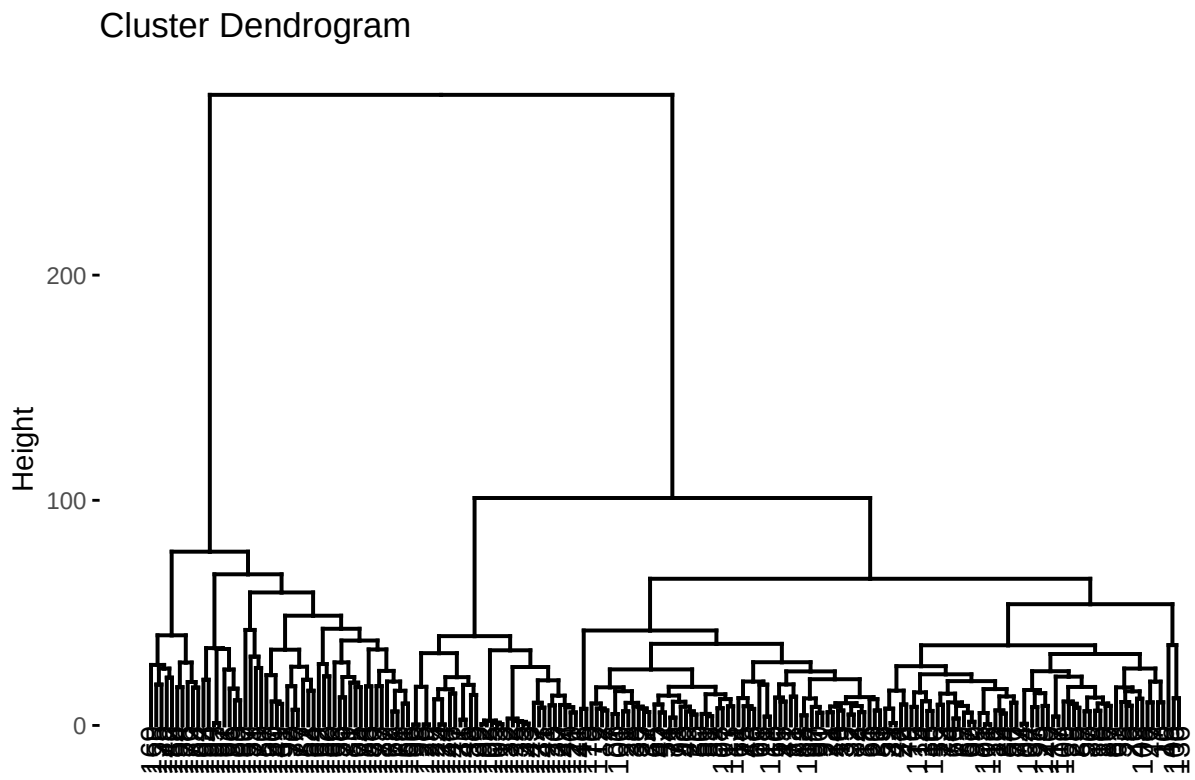
We now use a hierachical clustering technique using Ward's method. We also need to use the **dist()** function on our dataframe, otherwise we get an error when trying to create our cluster object.

```
set.seed(123)
ward_cluster <- hclust(dist(radiomics), method = "ward.D2")
```

Plotting our dendrogram

Now, we make a dendrogram:

```
dend_plot <- fviz_dend(ward_cluster)
dend_plot
```

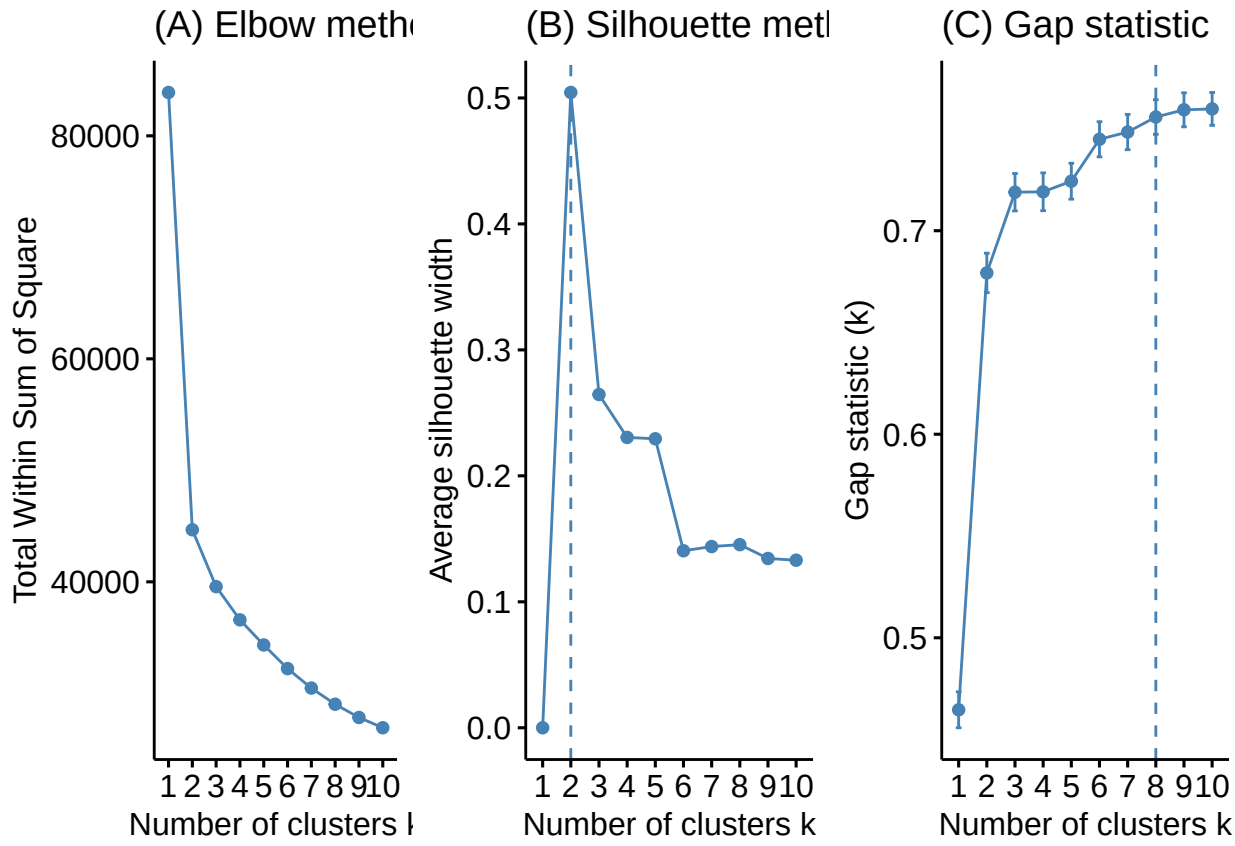


Determine number of groups and cutting tree into groups

We'll take a look at the elbow, silhouette, and gap statistic method to determine the number of groups we want to use.

```
p1 <- fviz_nbclust(radiomics, FUN = hcut, method = "wss",
                   k.max = 10) +
  ggtitle("(A) Elbow method")
```

```
p2 <- fviz_nbclust(radiomics, FUN = hcut, method = "silhouette",
                   k.max = 10) +
  ggtitle("(B) Silhouette method")
p3 <- fviz_nbclust(radiomics, FUN = hcut, method = "gap_stat",
                   k.max = 10) +
  ggtitle("(C) Gap statistic")

# Display plots side by side
gridExtra::grid.arrange(p1, p2, p3, nrow = 1)
```



Going off of our gap statistic, it looks like k = 8. We run the below code chunk to get our 8 groups and see the number of groups within each cluster:

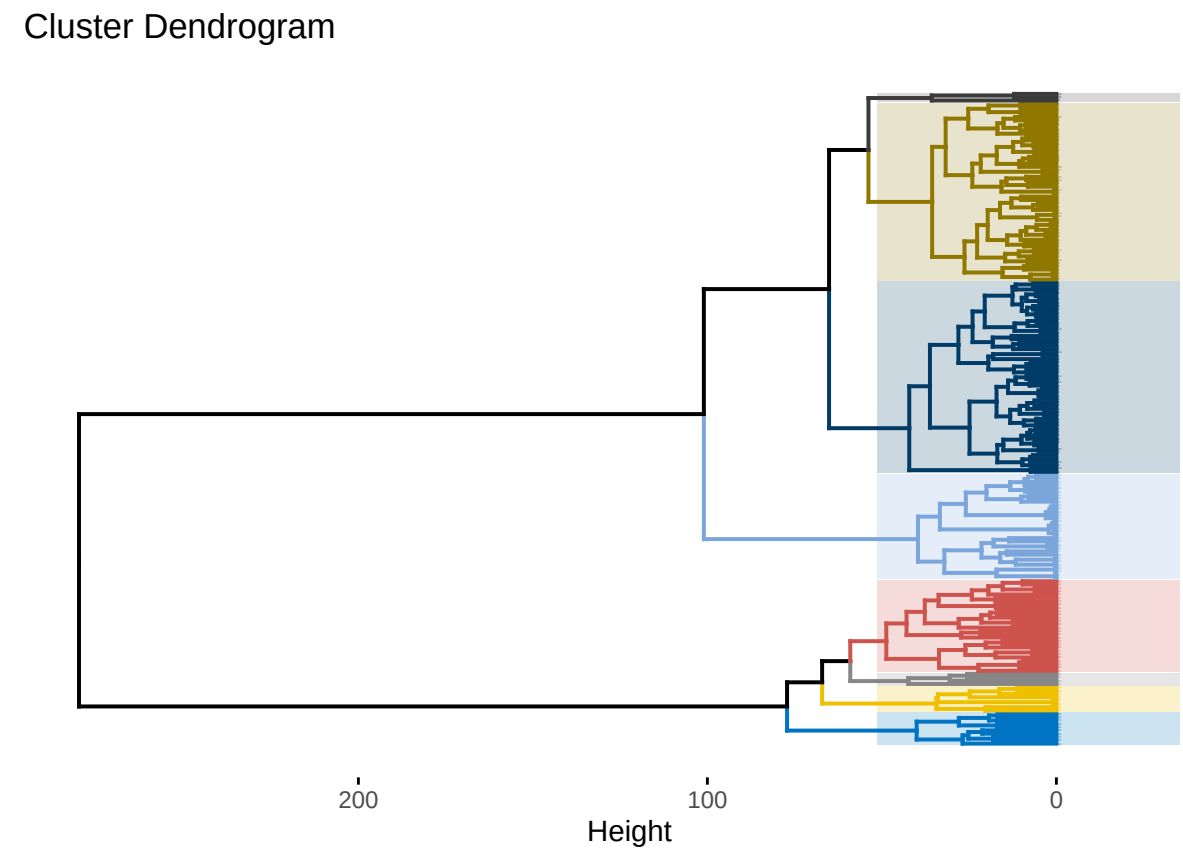
```
cluster_groups <- cutree(ward_cluster, k = 8)
table(cluster_groups)
```

```
## cluster_groups
##  1  2  3  4  5  6  7  8
## 58 54  3 32 10 28  4  8
```

Plotting our full dendrogram

Now, we run the below code chunk to plot our full dendrogram with k = 8.

```
plot <- fviz_dend(
  ward_cluster,
  k = 8,
  horiz = TRUE,
  rect = TRUE,
  rect_fill = TRUE,
  rect_border = "jco",
  k_colors = "jco",
  cex = 0.1,
)

plot(plot)
```



The shaded colors indicate which of the 8 groups and observation belongs to.

Appending clusters to the original dataframe

Before we finish with this method, let's get these clusters appended to our original dataframe so we can easily access which observation belongs to which cluster.

```
radiomics_with_clusters <- cbind(radiomics, cluster_id = cluster_groups)
head(radiomics_with_clusters[, c(1, 2, 429)])
```

```
##   Entropy_cooc.W.ADC GLNU_align.H.PET cluster_id
```

```
## 1         0.55290547      -0.57063689          1
## 2        -0.06486729      -0.78903636          1
## 3         0.45990825      -0.06024275          2
## 4         1.14318298       2.67468822          1
## 5         0.34499368      -0.06740573          2
## 6         0.84917904       0.07354603          2
```

Great! Now that our clusters are stored in a new dataframe, we could do further analysis on both the clusters and the individual observations within each cluster.

Third clustering method: Model-based

Now it's time to use our third and final clustering method, a model-based method. This method differs from the previous two. KMeans and hierarchical methods are heuristic methods and generate the clusters directly based on the data. Model-based methods, on the other hand, assign each observation a probability of belonging to a each cluster, which is called a soft assignment.

Once again, we make sure our data is prepared and scaled.

```
radiomics <-read.csv("radiomics_completedata.csv")
radiomics <- radiomics %>% select(-Institution, -Failure, -Failure.binary)
radiomics <- na.omit(radiomics)

radiomics <- as.data.frame(scale(radiomics))
```

Using Mclust()

Now we'll apply a Gaussian Mixture Modeling (GMM) technique to our data with the function **Mclust()**. Since our dataset has so many features, we'll take a look at just the first four features.

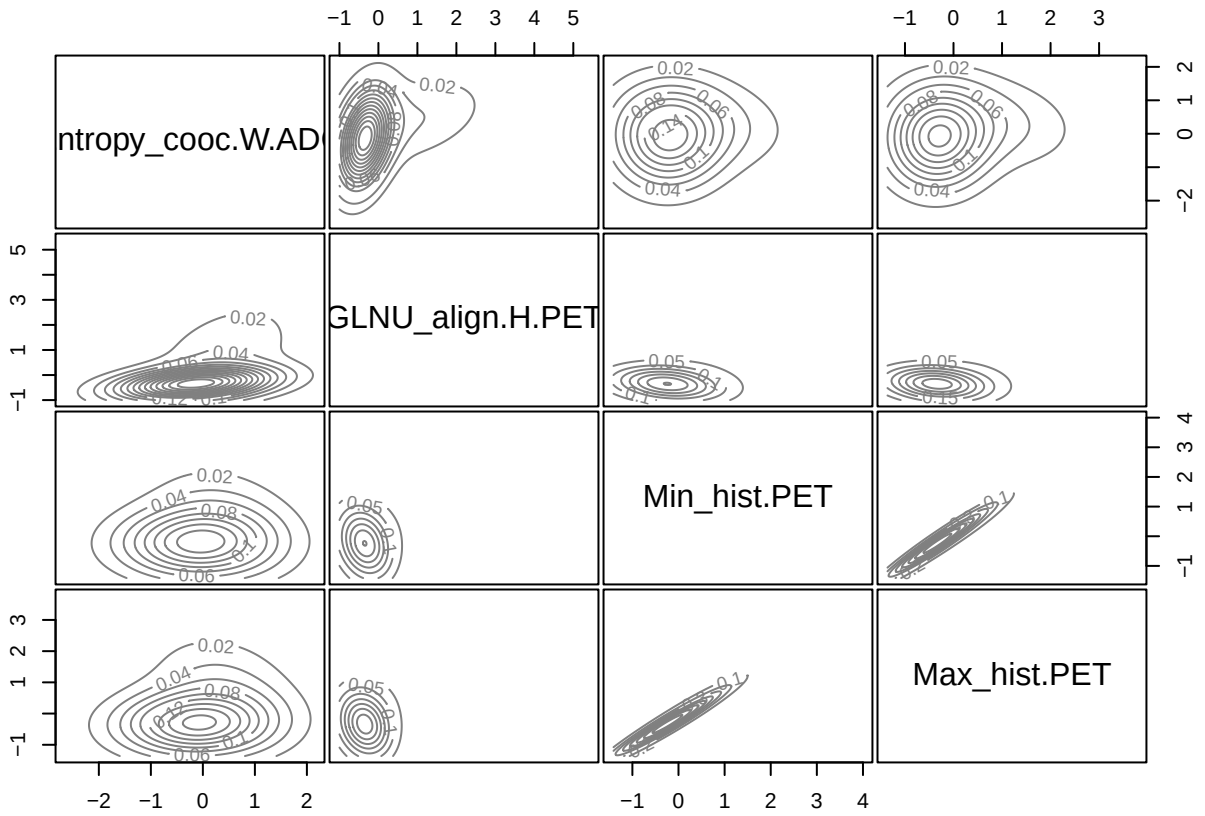
We're leaving G as **NULL** in order to force **Mclust()** to use the Bayesian Information Criterion (BIC) to determine the optimal number of components itself.

```
radiomics_mc <- Mclust(radiomics[, 1:4], G = NULL)
```

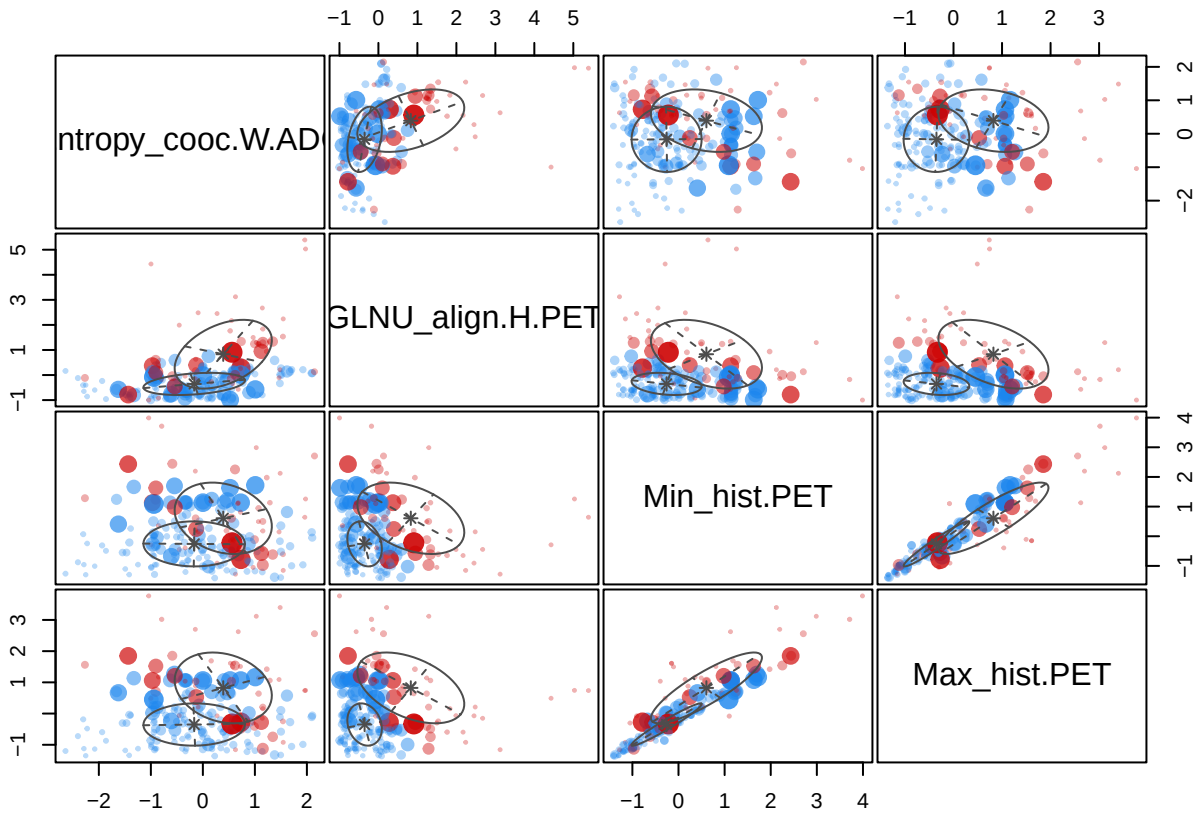
Plotting the density and uncertainty

Then, we can plot the results to see the density and uncertainty of our clusters.

```
plot(radiomics_mc, what = "density")
```




```
plot(radiomics_mc, what = "uncertainty")
```



Extracting model details

Now, we'll take a closer look at our **radiomics_mc** model to extract the model name and the value for G. In this case, the model name is **VVV** and the value for G (AKA our number of clusters) is **2**.

```
radiomics_mc$modelName
```

```
## [1] "VVV"
```

```
radiomics_mc$G
```

```
## [1] 2
```

We can also extract the uncertainty values and put them in descending order so we can see which six observations in the dataset had the highest degree of uncertainty:

```
sort(radiomics_mc$uncertainty, decreasing = TRUE) %>% head()
```

```
##       152       149        82       143       133       185
## 0.4417627 0.3756285 0.3573175 0.3229306 0.3149667 0.3084062
```

We can also use **summary()** to see the number of observations in our clusters:

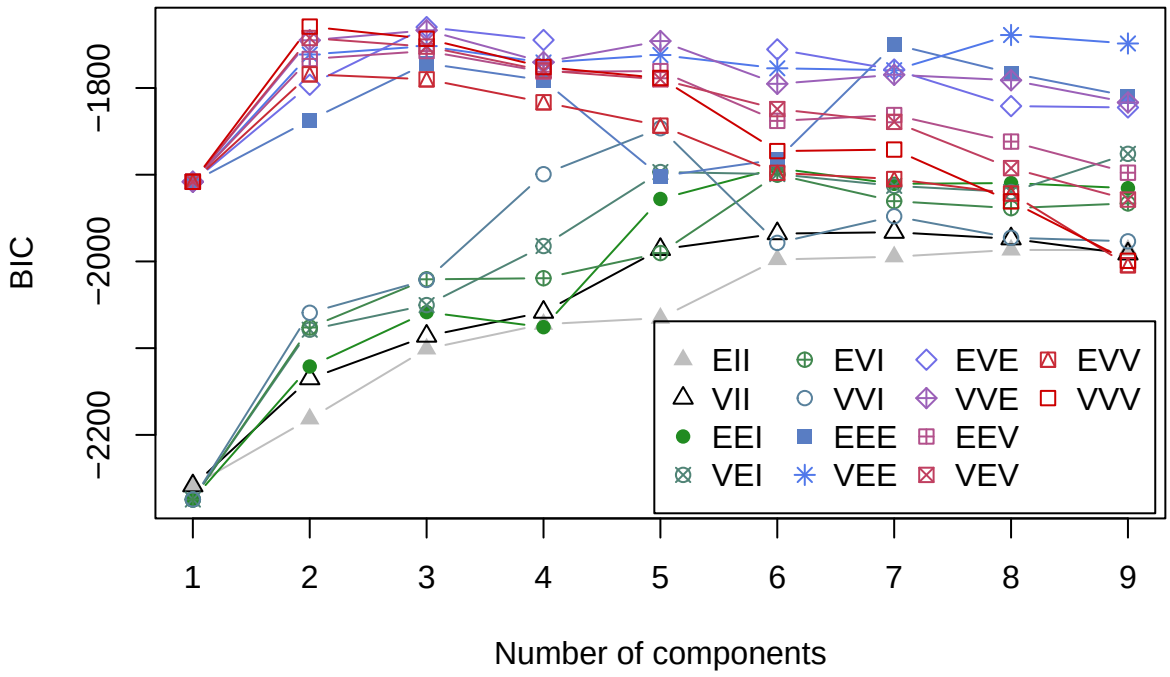
```
print(summary(radiomics_mc))
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VVV (ellipsoidal, varying volume, shape, and orientation) model with 2
## components:
##
##  log-likelihood   n df       BIC       ICL
##       -788.0107 197 29 -1729.234 -1749.257
##
## Clustering table:
##   1   2
## 142  55
```

Plotting the performance of all 14 covariance models

If we want to see the performance of all 14 covariance models across 1-9 components, we can run the following code chunk.

```
legend_args <- list(x = "bottomright", ncol = 4)
plot(radiomics_mc, what = 'BIC', legendArgs = legend_args)
```



Getting the probabilities

We can extract the probabilities into a matrix and rename the column names to correspond to our two clusters.

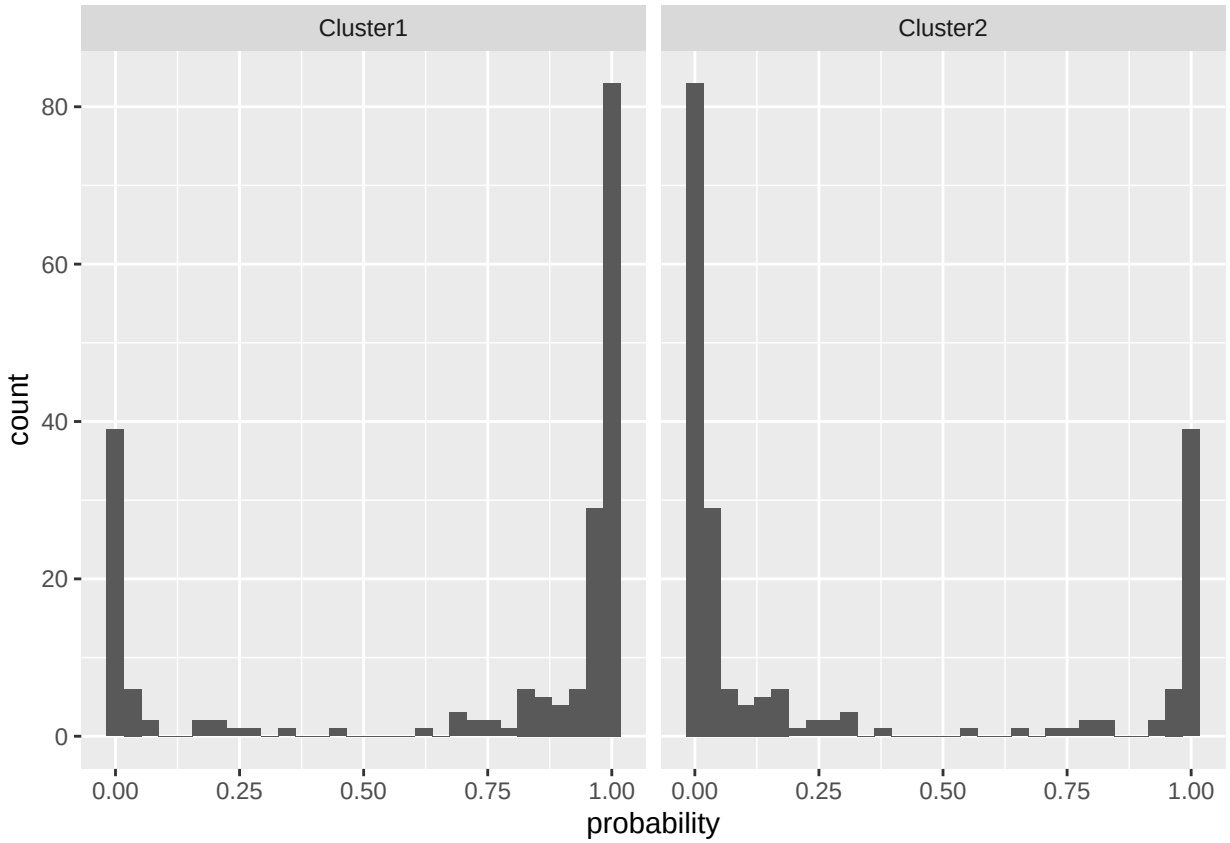
```
probabilities <- radiomics_mc$z
colnames(probabilities) <- paste0('Cluster', 1:2)
head(probabilities)
```

```
##       Cluster1    Cluster2
## 1 9.932578e-01 0.006742238
## 2 9.836850e-01 0.016315014
## 3 9.955950e-01 0.004405021
## 4 5.643926e-10 0.999999999
## 5 9.950843e-01 0.004915657
## 6 9.936693e-01 0.006330701
```

Plotting the probabilities

With our new probabilities matrix, we will now generate a plot of the probabilities using **ggplot()**.

```
probabilities <- probabilities %>%
  as.data.frame() %>%
  mutate(id = row_number()) %>%
  tidyr::gather(cluster, probability, -id)

ggplot(probabilities, aes(probability)) +
  geom_histogram() +
  facet_wrap(~ cluster, nrow = 1)
```

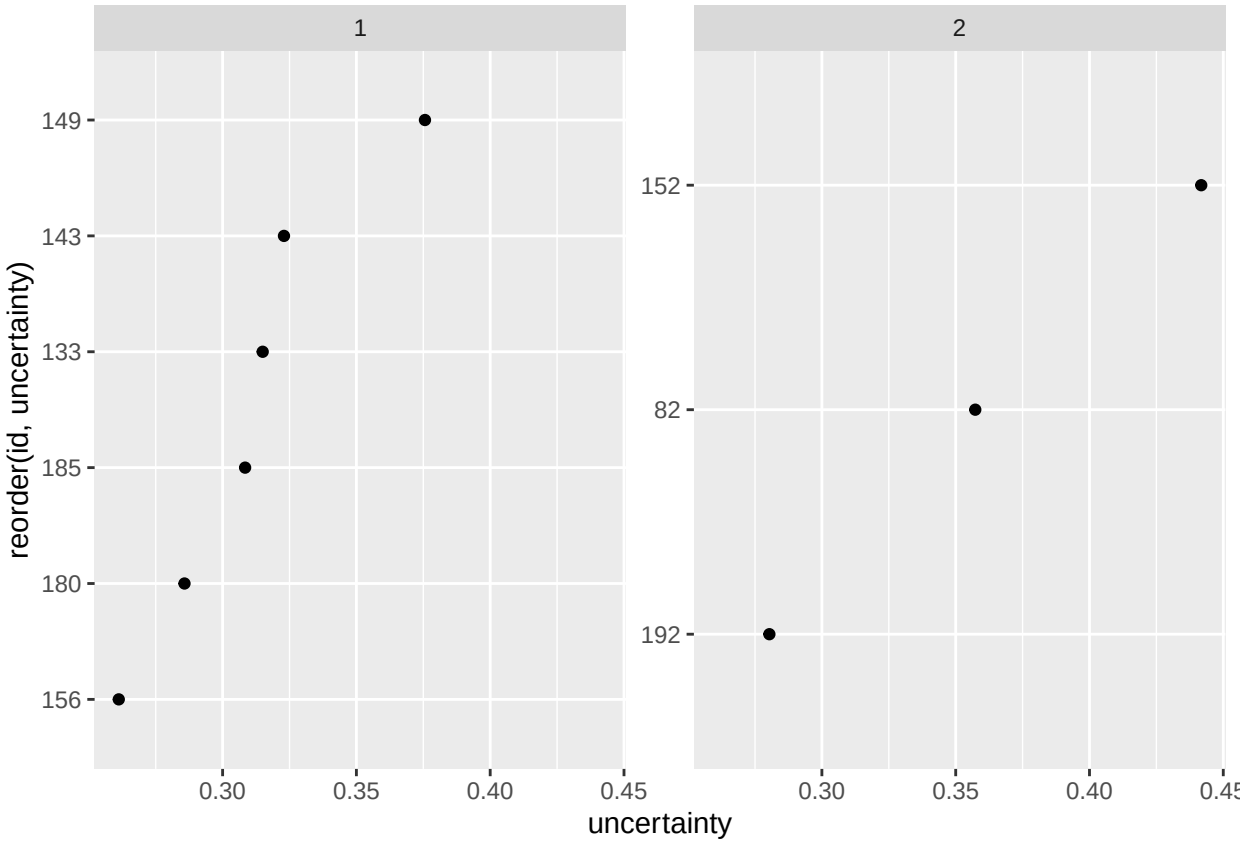


The plot on the left shows us the probability of observations belonging to Cluster 1, and the plot on the right shows us the probability of observations belonging to Cluster 2. As we can see, most observations fall either close to 0 or close to 1, with some observations falling more in the middle, which indicates a high degree of uncertainty.

Plotting the uncertainty

Speaking of uncertainty, we can also visualize our clusters by plotting the uncertainty values of all observations that have an uncertainty greater than **0.25**.

```
uncertainty <- data.frame(
  id = 1:nrow(radiomics),
  cluster = radiomics_mc$classification,
  uncertainty = radiomics_mc$uncertainty
)

uncertainty %>%
  group_by(cluster) %>%
  filter(uncertainty > 0.25) %>%
  ggplot(aes(uncertainty, reorder(id, uncertainty))) +
  geom_point() +
  facet_wrap(~ cluster, scales = 'free_y', nrow = 1)
```



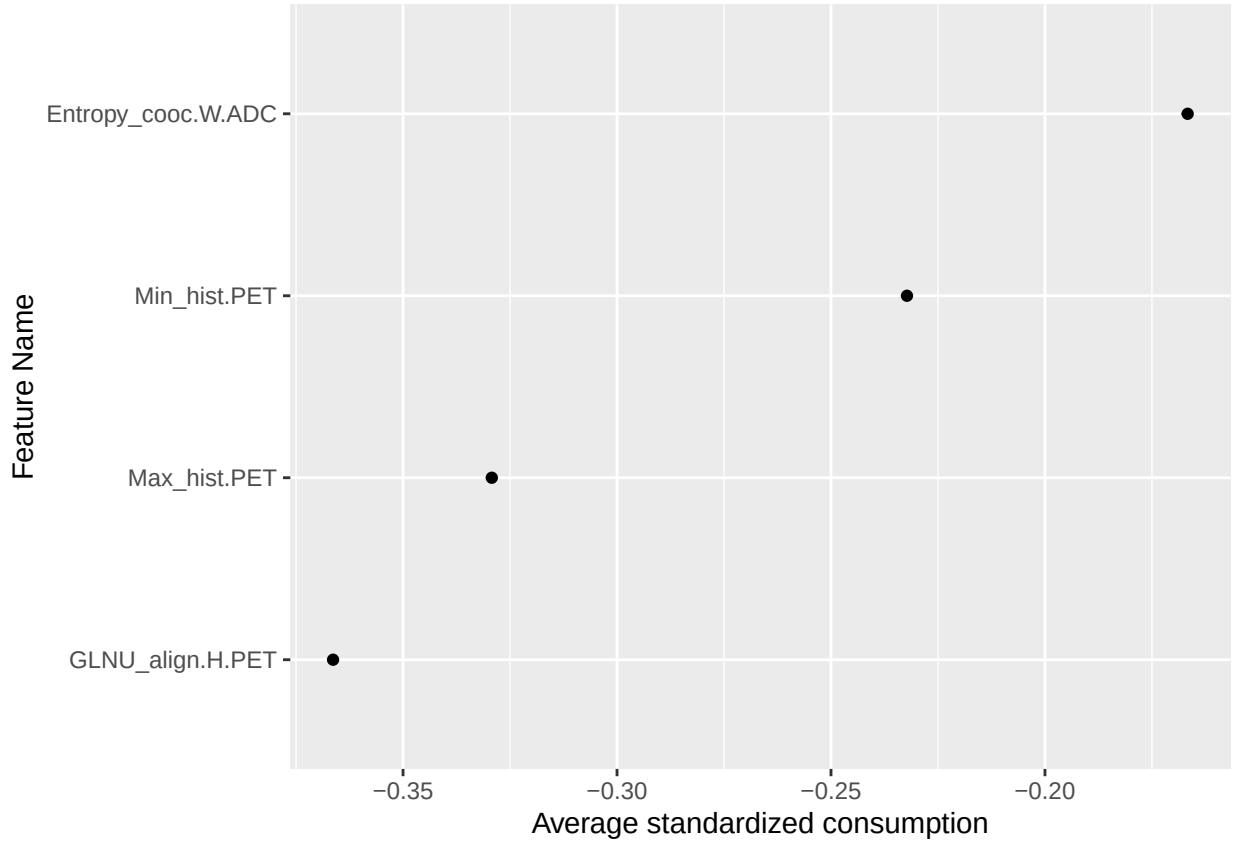
As we can see above, Cluster 1 has more observations with an uncertainty greater than 0.25, with 6 compared to 3. However, Cluster 2 contains the observation with the highest uncertainty value, which is observation 152.

Plotting the Average Standardized Consumption of each cluster

Finally, we can filter our data frame by each cluster and make two plots that demonstrate the average standardized consumption of each feature in both clusters so we can compare.

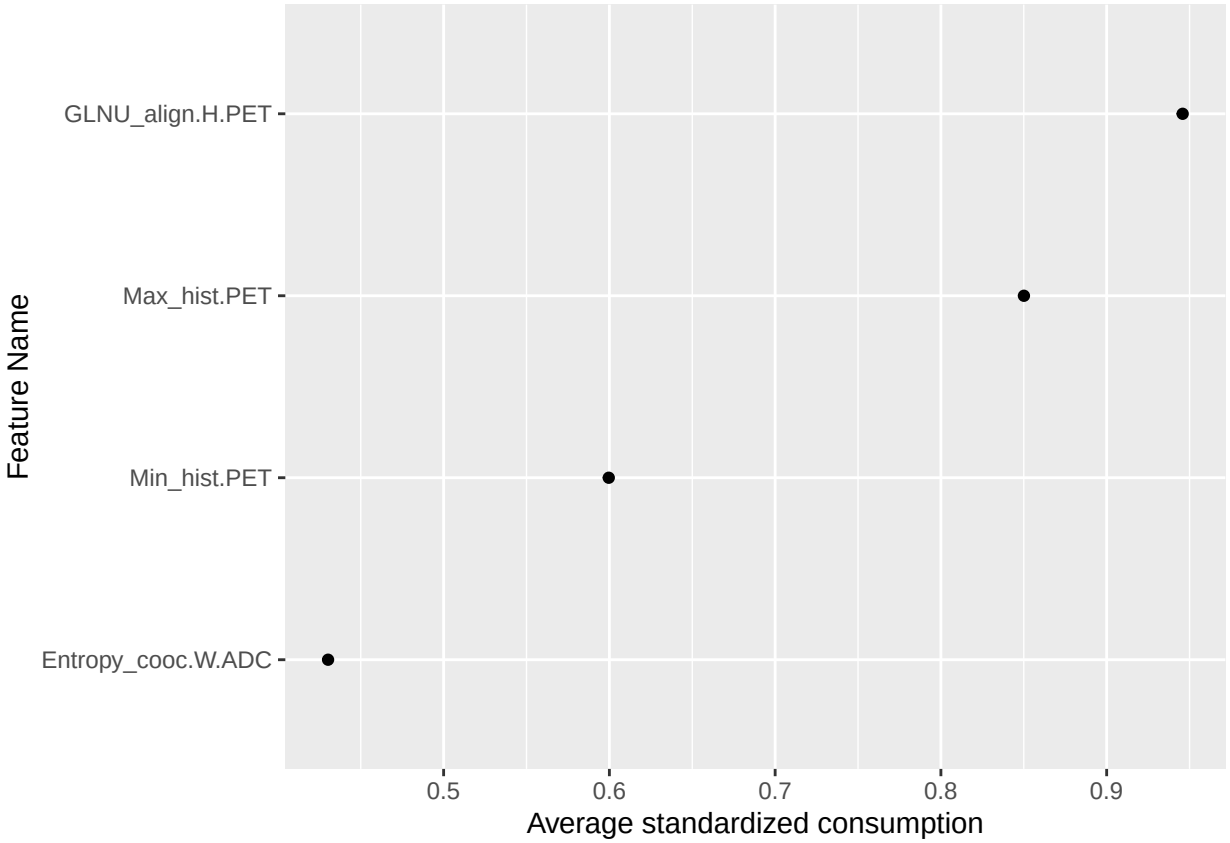
Cluster 1

```
cluster_1 <- radiomics[, 1:4] %>%
  mutate(cluster = radiomics_mc$classification) %>%
  filter(cluster == 1) %>%
  select(-cluster)

cluster_1 %>%
  tidyr::gather(x, std_count) %>%
  group_by(x) %>%
  summarize(avg = mean(std_count)) %>%
  ggplot(aes(avg, reorder(x, avg))) +
  geom_point() +
  labs(x = "Average standardized consumption", y = "Feature Name")
```



Cluster 2

```
cluster_2 <- radiomics[, 1:4] %>%
  mutate(cluster = radiomics_mc$classification) %>%
  filter(cluster == 2) %>%
  select(-cluster)

cluster_2 %>%
  tidyr::gather(x, std_count) %>%
  group_by(x) %>%
  summarize(avg = mean(std_count)) %>%
  ggplot(aes(avg, reorder(x, avg))) +
  geom_point() +
  labs(x = "Average standardized consumption", y = "Feature Name")
```


GLNU_align.H.PET
Max_hist.PET
Min_hist.PET
Entropy_cooc.W.ADC
Feature Name
0.5
0.6
0.7
0.8
0.9
Average standardized consumption