R Notebook

Octavio del Ser

10/11/2020

Task1 Source code

Implement hierarchial agglomerative clustering with single, complete, average and centroid linkage. Package dependencies (for vis)

```
install.packages('factoextra')
```

```
Installing package into '/home/caramel/R/x86_64-pc-linux-gnu-library/3.6'
(as 'lib' is unspecified)
```

```
library(factoextra)
```

```
Loading required package: ggplot2
```

```
Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

Iorder

this is the function R package uses for making sure branches don't intersect for the merge matrix

```
iorder <- function(m)
  {
  N <- nrow(m) + 1
  iorder <- rep(0, N)
  iorder[1] <- m[N - 1, 1]
  iorder[2] <- m[N - 1, 2]
  loc <- 2
  for (i in seq(N - 2, 1))
    {
    for (j in seq(1, loc))
      {
      if (iorder[j] == i)
        {
        iorder[j] <- m[i, 1]
        if (j == loc)
          {
          loc <- loc + 1
          iorder[loc] <- m[i, 2]
        } else
          {
          loc <- loc + 1
          for (k in seq(loc, j + 2)) iorder[k] <- iorder[k - 1]
          iorder[j + 1] <- m[i, 2]
        }
      }
    }
  }
  -iorder
}
```

Specified metrics

```
l2_norm <- function(x, y) {
  return(sqrt(sum((x - y)^2)))
}

single_l <- function(x, y) {
  return(min(dist_m(x, y)))
}
complete_l <- function(x, y) {
  return(max(dist_m(x, y)))
}
average_l <- function(x, y) {
  return((1 / (length(x) + length(y))) * sum(dist_m(x, y)))
}
centroid_l <- function(x, y) {
  return(l2_norm(mean(x), mean(y)))
}
dist_m <- function(x, y) {
  d <- list()
  min_i <- 0
  for (v in 1:ncol(x)) {
    for (v2 in 1:ncol(y)) {
      min_i <- min_i + 1
      d[[min_i]] <- l2_norm(x[, v], y[, v2])
    }
  }
  return(unlist(d))
}
calc_similarity <- function(cluster, measure, n_attr) {
  sim <- matrix(nrow = length(cluster), ncol = length(cluster))
  for (min_i in 1:length(cluster)) {
    for (j in 1:length(cluster)) {
      if (j != min_i) {
        sim[min_i, j] <- measure(matrix(cluster[[min_i]], nrow = n_attr), matrix(cluster
[[j]], nrow = n_attr))
      }else {
        sim[min_i, j] <- Inf
      }
    }
  }
  return(sim)
}
```

Merging clusters (n -> n-1)

```
merge_cluster <- function(cluster, measure, n_attr) {
  d <- calc_similarity(cluster, measure, n_attr)
  min_rc <- which(d == min(d), arr.ind = TRUE)[1,]
  c1 <- cluster[[min_rc[[1]]]]
  c2 <- cluster[[min_rc[[2]]]]
  cluster[[ min_rc[[1]]]] <- c(c2, c1)
  return(list(cluster[-min_rc[[2]]], unlist(min_rc), min(d)))
}
```

Question 1

Hcluster implementation using previously defined functions (this code was checked against hcluster() (R) it runs around O(n) times slower (real world) due to non-vectorised operations etc...

```
my_hclust <- function(x, measure = c("single", "complete", "average", "median"))
  {
  y <- t(x)
  clusters <- lapply(seq_len(ncol(y)), function(i) y[, i])  # Pick a clustering function:
  method_fn <- switch(match.arg(measure),
                      single = single_l,
                      complete = complete_l,
                      average = average_l,
                      median = centroid_l)
  cluster_ids <- 1:length(clusters)
  cluster_ids <- cluster_ids * -1
  combination_matrix <- matrix(ncol = 2, nrow = length(clusters) - 1)
  heights <- list()
  order <- c(1, 1)
  i <- 0
  n_attr <- length(clusters[[1]])
  while (length(clusters) > 1) {
    i <- i + 1
    results <- merge_cluster(clusters, method_fn, n_attr)
    ids_combined <- results[[2]]
    heights[[i]] <- results[[3]]
    ids_c_values <- c(cluster_ids[ids_combined[[1]]], cluster_ids[ids_combined[[2]]])
    if (max(ids_c_values) < 0) {
      order <- c(order, rev(ids_c_values))
      combination_matrix[i, 1] <- max(ids_c_values)
      combination_matrix[i, 2] <- min(ids_c_values)
    }else {
      order <- c(rev(ids_c_values), order)
      combination_matrix[i, 1] <- min(ids_c_values)
      combination_matrix[i, 2] <- max(ids_c_values)
    }
    cluster_ids[ids_combined[[1]]] <- i
    cluster_ids <- cluster_ids[-ids_combined[[2]]]
    clusters <- results[[1]]
  }
  order <- abs(order[order < 0])

  mode(combination_matrix) <- 'integer'
  #return similar structure to hclust for plotting
  return(structure(list(merge = combination_matrix, height = unlist(heights), order = iorder
(combination_matrix),
                        labels = rownames(x), method = measure,
                        call = match.call(), dist.method = "euclidean"),
                   class = "hclust"))
}
```

Question 2

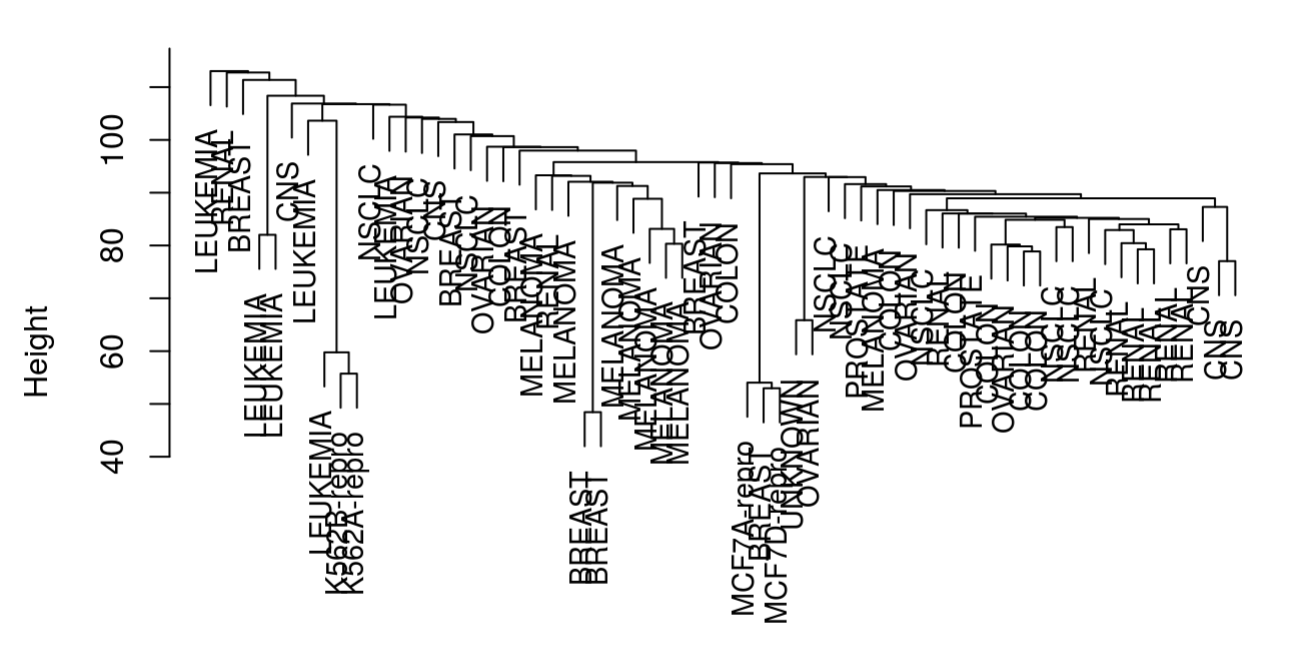
Load data

```
dname <- 'nci.data.txt'
lname <- 'label.txt'
x <- t(as.matrix(read.table(dname)))
x <- scale(x)
lab <- as.vector(as.matrix(read.table(lname)))
```

Run each type of linkage and plot I'm unsure why single average and complete does not perform as expected, I assume I made a mistake somewhere in the computation.

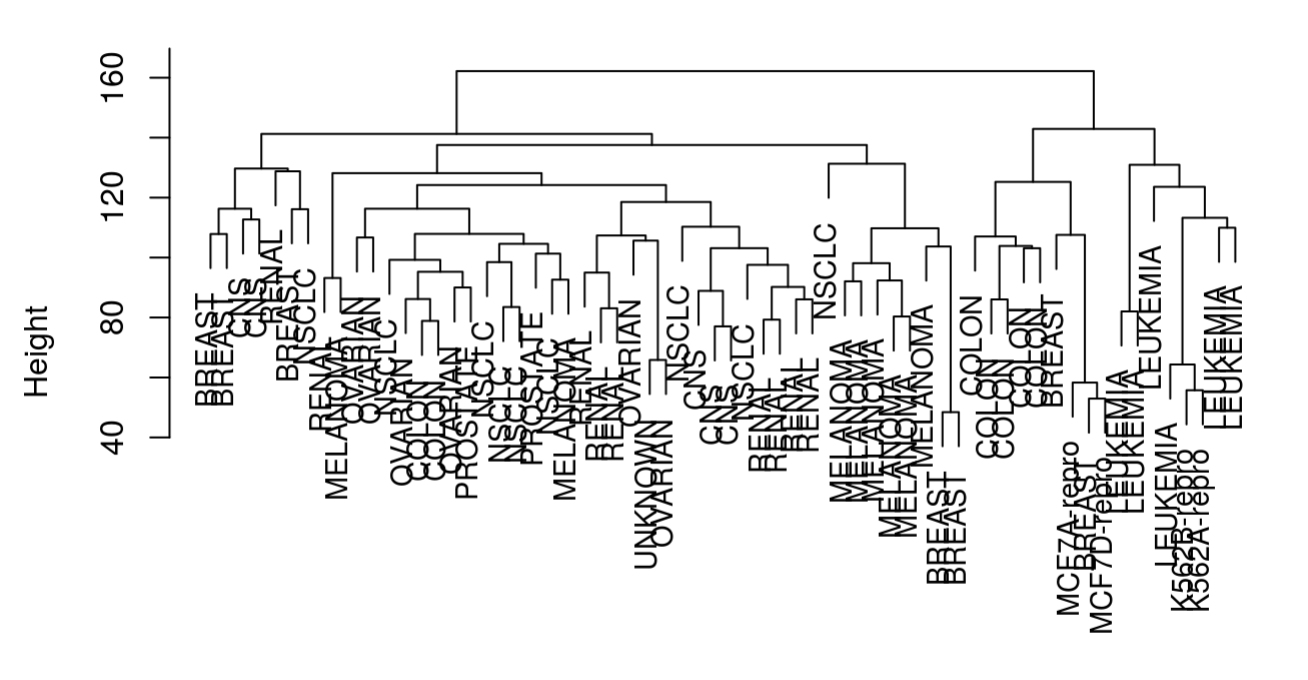
```
for (link in c('single', 'complete', 'average', 'median')) {
  #par(mfrow = c(1, 2))
  res_h <- my_hclust(x, measure = link)
  res_h$labels <- lab
  plot(res_h, main = paste0(link, " Linkage"), xlab = "", sub = "", cex = .9)
  #u <- length(unique(lab))
  #ku <- cutree(res_h, 3)
  clusters = cutree(res_h, k = 5)
  print(table(clusters, lab))

  #k <- my_hclust(ku)
  #k$labels <- lab
  #plot(k, main = paste0(link, " Linkage, cut: ", u), xlab = "", sub = "", cex = .9)
}
```

single Linkage



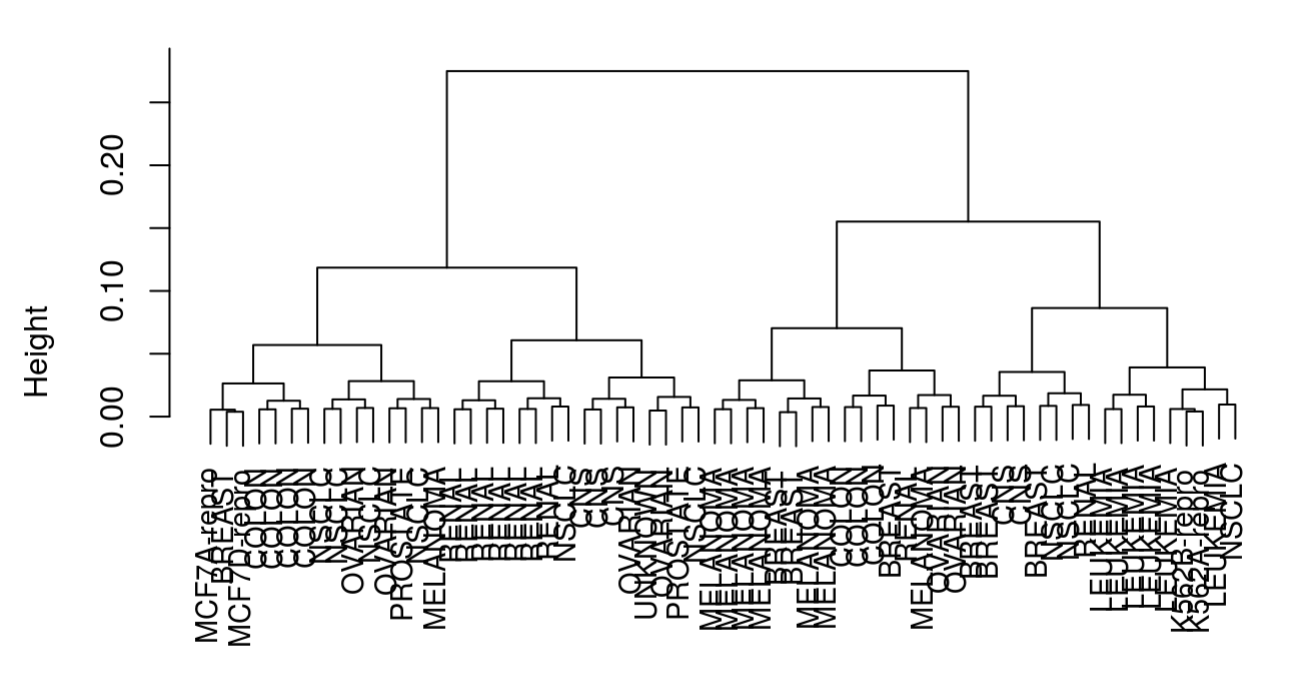

```
       lab
clusters BREAST CNS COLON K562A-repro K562B-repro LEUKEMIA MCF7A-repro
       1      6   5     7           1           1        3           1
       2      1   0     0           0           0        0           0
       3      0   0     0           0           0        0           0
       4      0   0     0           0           0        2           0
       5      0   0     0           0           0        1           0
       lab
clusters MCF7D-repro MELANOMA NSCLC OVARIAN PROSTATE RENAL UNKNOWN
       1           1        8     9       6        2     8       1
       2           0        0     0       0        0     0       0
       3           0        0     0       0        0     1       0
       4           0        0     0       0        0     0       0
       5           0        0     0       0        0     0       0
```

complete Linkage



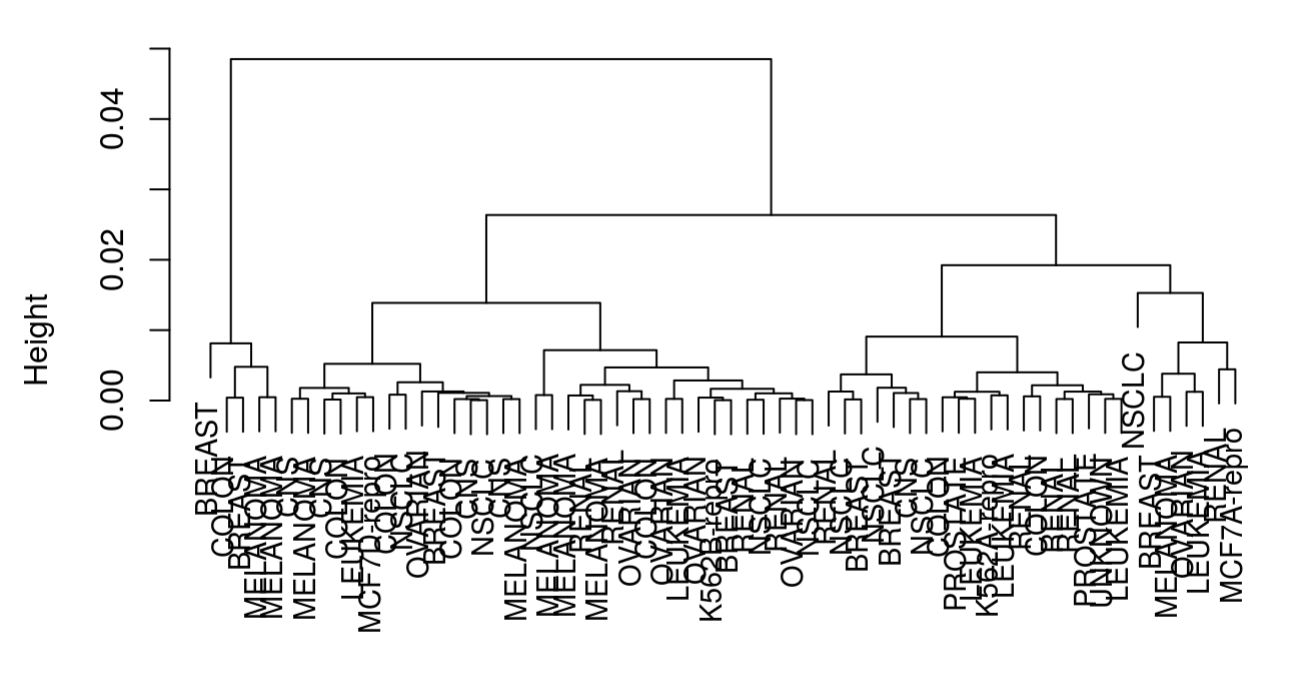

```
       lab
clusters BREAST CNS COLON K562A-repro K562B-repro LEUKEMIA MCF7A-repro
       1      0   3     2           0           0        0           0
       2      3   2     0           0           0        0           0
       3      0   0     0           1           1        6           0
       4      2   0     5           0           0        0           1
       5      2   0     0           0           0        0           0
       lab
clusters MCF7D-repro MELANOMA NSCLC OVARIAN PROSTATE RENAL UNKNOWN
       1           0        2     7       6        2     8       1
       2           0        0     1       0        0     1       0
       3           0        0     0       0        0     0       0
       4           1        0     0       0        0     0       0
       5           0        6     1       0        0     0       0
```

average Linkage




```
       lab
clusters BREAST CNS COLON K562A-repro K562B-repro LEUKEMIA MCF7A-repro
       1      0   3     0           0           0        0           0
       2      3   0     3           0           0        0           0
       3      3   2     0           0           0        0           0
       4      1   0     4           0           0        0           1
       5      0   0     0           1           1        6           0
       lab
clusters MCF7D-repro MELANOMA NSCLC OVARIAN PROSTATE RENAL UNKNOWN
       1           0        0     2       2        1     7       1
       2           0        7     0       2        0     1       0
       3           0        0     2       0        0     1       0
       4           1        1     4       2        1     0       0
       5           0        0     1       0        0     0       0
```

median Linkage




```
       lab
clusters BREAST CNS COLON K562A-repro K562B-repro LEUKEMIA MCF7A-repro
       1      2   4     4           0           1        2           0
       2      2   1     2           1           0        3           0
       3      1   0     0           0           0        1           1
       4      2   0     1           0           0        0           0
       5      0   0     0           0           0        0           0
       lab
clusters MCF7D-repro MELANOMA NSCLC OVARIAN PROSTATE RENAL UNKNOWN
       1           1        5     5       5        0     4       0
       2           0        0     3       0        2     4       1
       3           0        1     0       1        0     1       0
       4           0        2     0       0        0     0       0
       5           0        0     1       0        0     0       0
```

Question 3

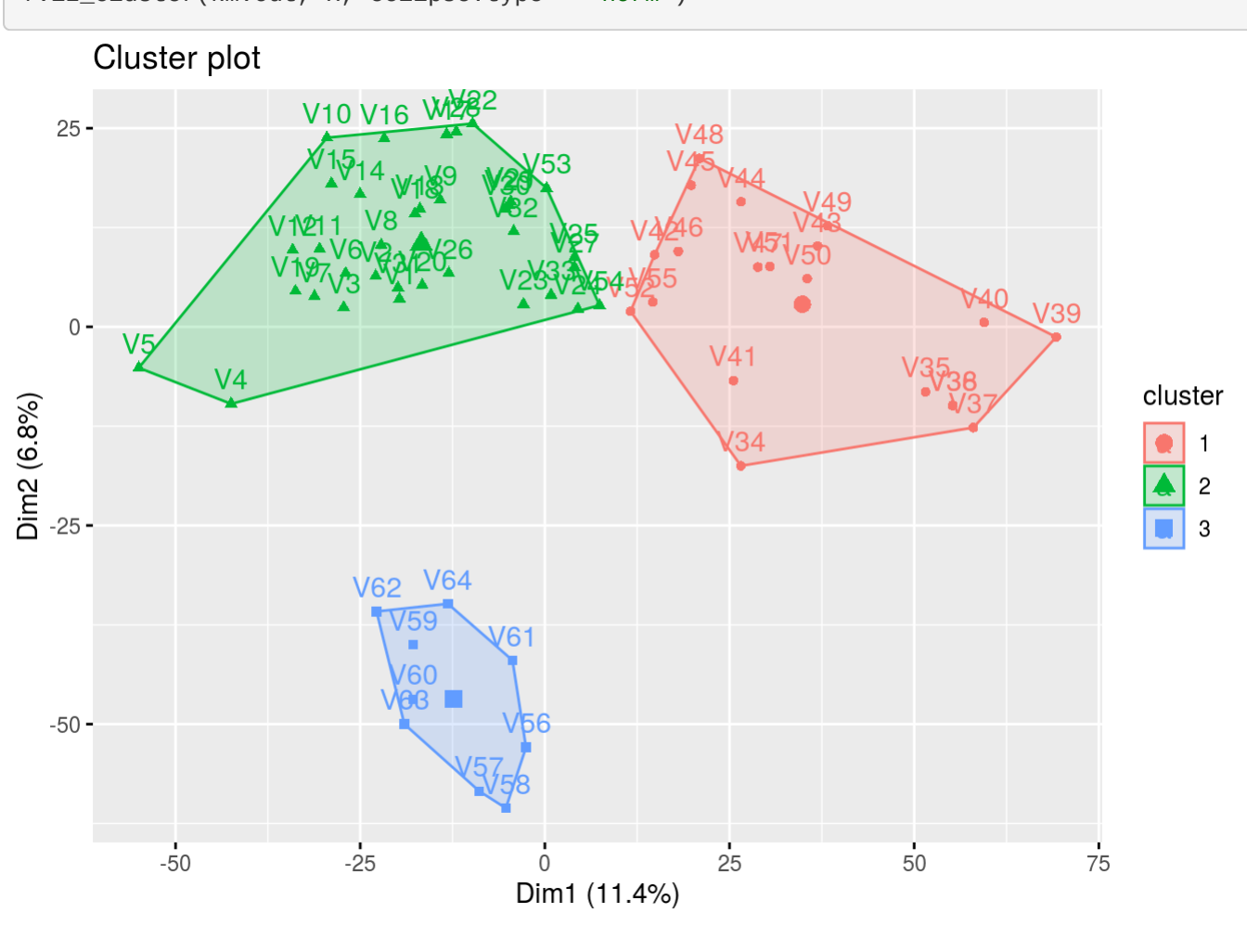
Even when performing these with hclust method, I don't seem to have gotten very good results (single and complete) matched the tree in hclust but average and centroid didn't. Average linkage seems to have produced the best results (actually separating similar labels)

Question 4

I applied kmeans with k from 1->5 for reference labels are printed first.

```
library(factoextra)
library(ggpubr)

km.out <- kmeans(x, centers = 3, nstart = 5)
fviz_cluster(km.out, x, eclipse.type = 'norm')
```

Cluster plot

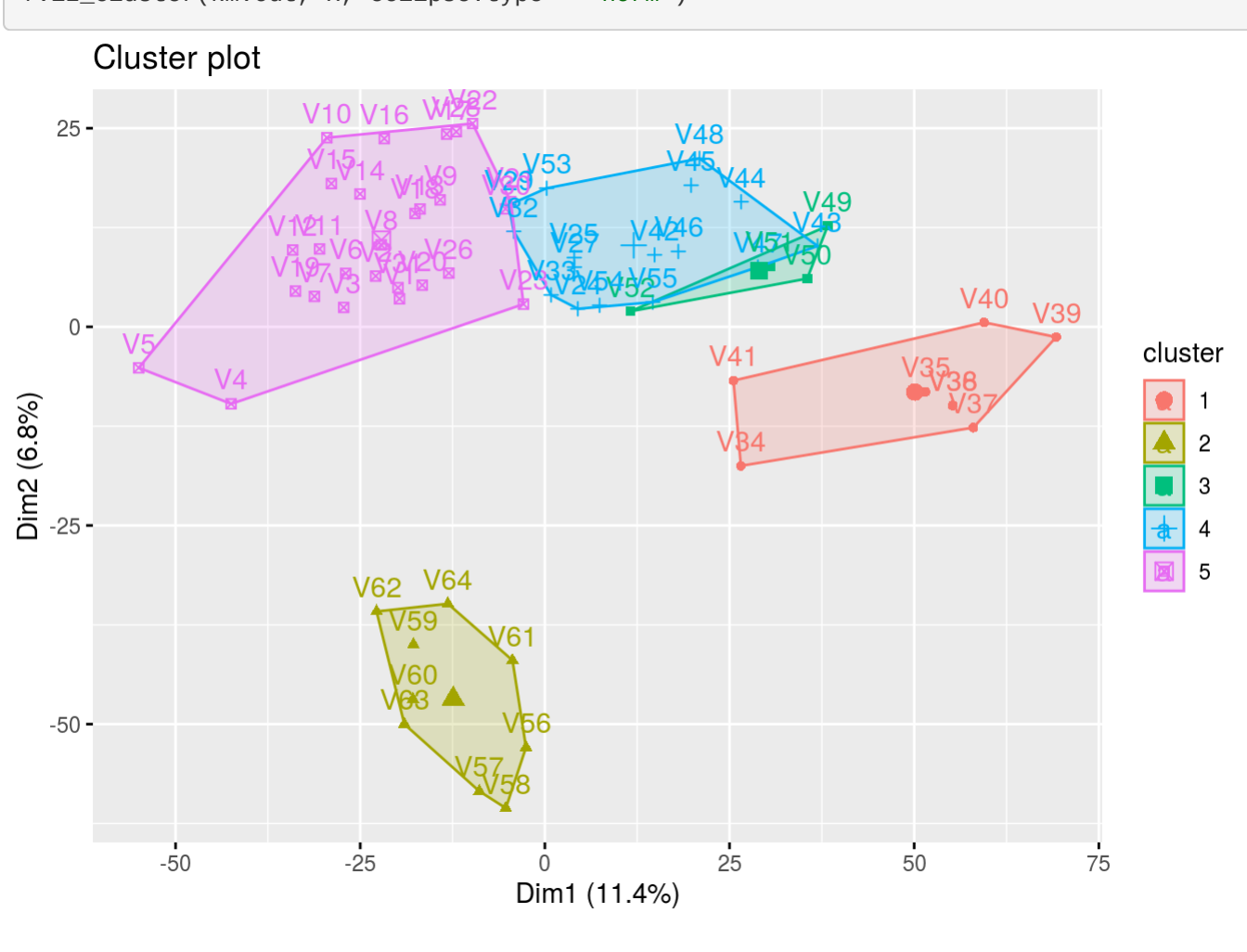



```
print(table(km.out$cluster, lab))
```

```
   lab
    BREAST CNS COLON K562A-repro K562B-repro LEUKEMIA MCF7A-repro MCF7D-repro
  1      2   0     7           1           1        6           1           1
  2      3   5     0           0           0        0           0           0
  3      2   0     0           0           0        0           0           0
   lab
    MELANOMA NSCLC OVARIAN PROSTATE RENAL UNKNOWN
  1        0     1       0        0     0       0
  2        1     8       6        2     9       1
  3        7     0       0        0     0       0
```

```
km.out <- kmeans(x, centers = 5, nstart = 5)
fviz_cluster(km.out, x, eclipse.type = 'norm')
```

Cluster plot

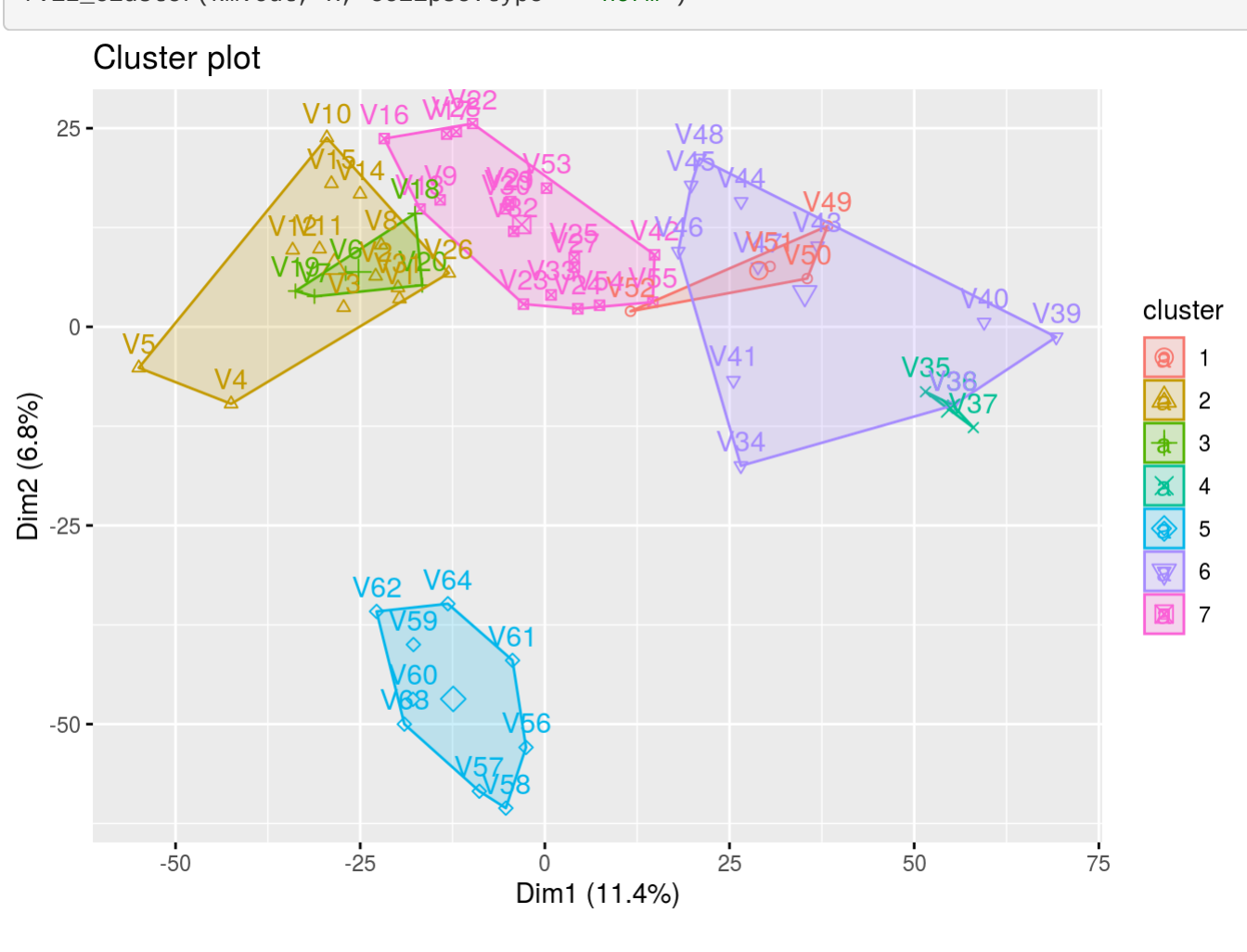



```
print(table(km.out$cluster, lab))
```

```
   lab
    BREAST CNS COLON K562A-repro K562B-repro LEUKEMIA MCF7A-repro MCF7D-repro
  1      0   0     0           1           1        6           0           0
  2      2   0     0           0           0        0           0           0
  3      2   0     0           0           0        0           1           1
  4      0   0     7           0           0        0           0           0
  5      3   5     0           0           0        0           0           0
   lab
    MELANOMA NSCLC OVARIAN PROSTATE RENAL UNKNOWN
  1        0     0       0        0     0       0
  2        7     0       0        0     0       0
  3        0     0       0        0     0       0
  4        0     5       3        1     0       0
  5        1     4       3        1     9       1
```

```
km.out <- kmeans(x, centers = 7, nstart = 5)
fviz_cluster(km.out, x, eclipse.type = 'norm')
```

Cluster plot

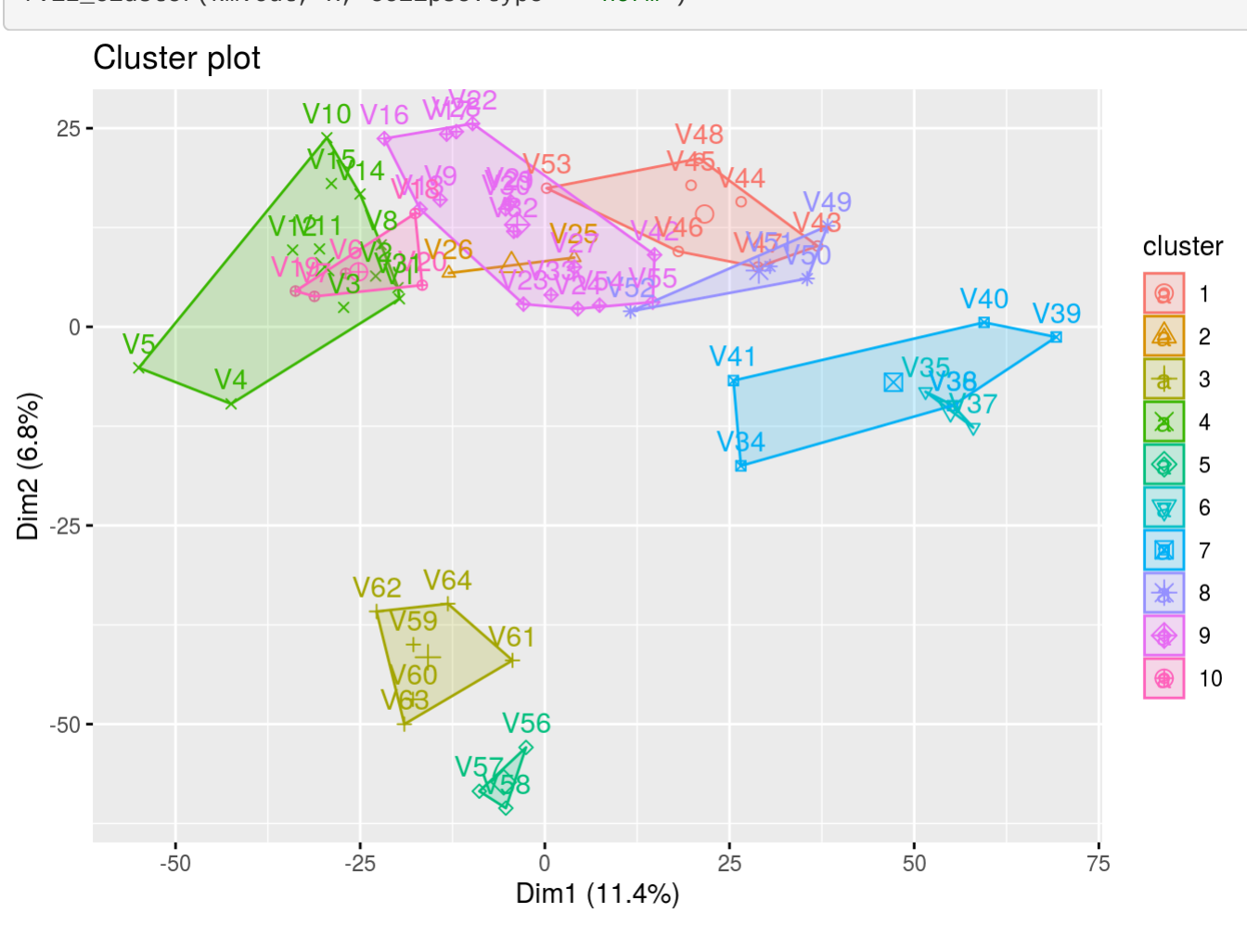



```
print(table(km.out$cluster, lab))
```

```
   lab
    BREAST CNS COLON K562A-repro K562B-repro LEUKEMIA MCF7A-repro MCF7D-repro
  1      2   0     0           0           0        0           1           1
  2      2   3     0           0           0        0           0           0
  3      1   2     0           0           0        0           0           0
  4      0   0     0           1           1        1           0           0
  5      2   0     0           0           0        0           0           0
  6      0   0     6           0           0        5           0           0
  7      0   0     1           0           0        0           0           0
   lab
    MELANOMA NSCLC OVARIAN PROSTATE RENAL UNKNOWN
  1        0     0       0        0     0       0
  2        0     2       1        0     5       0
  3        0     1       0        0     1       0
  4        0     0       0        0     0       0
  5        7     0       0        0     0       0
  6        0     0       0        0     0       0
  7        1     6       5        2     3       1
```

```
km.out <- kmeans(x, centers = 10, nstart = 5)
fviz_cluster(km.out, x, eclipse.type = 'norm')
```

Cluster plot




```
print(table(km.out$cluster, lab))
```

```
    lab
     BREAST CNS COLON K562A-repro K562B-repro LEUKEMIA MCF7A-repro MCF7D-repro
  1       0   0     6           0           0        0           0           0
  2       0   0     0           0           0        0           0           0
  3       0   0     0           0           0        0           0           0
  4       2   3     0           0           0        0           0           0
  5       2   0     0           0           0        0           0           0
  6       0   0     0           1           1        1           0           0
  7       0   0     0           0           0        5           0           0
  8       2   0     0           0           0        0           1           1
  9       0   0     1           0           0        0           0           0
  10      1   2     0           0           0        0           0           0
    lab
     MELANOMA NSCLC OVARIAN PROSTATE RENAL UNKNOWN
  1         0     1       0        0     0       0
  2         0     0       2        0     0       0
  3         6     0       0        0     0       0
  4         0     2       0        0     5       0
  5         1     0       0        0     0       0
  6         0     0       0        0     0       0
  7         0     0       0        0     0       0
  8         0     0       0        0     0       0
  9         1     5       4        2     3       1
  10        0     1       0        0     1       0
```

```
#
#
#}

# Visualize kmeans clustering
# use repel = TRUE to avoid overplotting
```

Question 5

Both Kmeans and hclust have generated clusters. For k means we require advancd knowledge of k in order to get good separation, whereas with hcluster we can cut the tree where we feel like good separation has been achieved. We also have more types of measures to represent clusters with hcluster such as mean, min, max etc..

Question 6

This is the bias variance tradeoff. To choose k we can increase it until lowest error is met on a test set.