Homework 5 Solutions

Alexander McLain

```r
library(printr)
library(nnet)
library(e1071)
library(tidyverse)
library(caret)
library(fastDummies)
```

1. **(40 points)** The data we'll use for this problem originated in a Kaggle competition. The goal was to predict success or failure of a grant application based on information about the grant and the associated investigators. It is very important to read about the variables which you can do here. Think carefully about how to include them in your model. You *may* use listwise deletion for some missing values (but be careful with this). You will train a neural network using the data `unimelb\_training.csv`, then you will predict for the `unimelb\_test.csv` which does not include the outcome (I do have the outcomes to this data). Here's what you'll hand in:

- Turn in (via R markdown or copy paste) the code you used to fit your neural network. This code is work 15 points.
- An estimate of your misclassification rate on the test data (worth 10 points)
- An estimate of the 95% CI for the misclassification rate (worth 5 points)
- A csv file that will have two rows – `Proposal.num`, and `Pred.Grant.Status` – for all observations in the test data. The Pred.Grant.Status variable should be 0, 1, or NA. There should be a row for each observation in the test data (even if they have missing data). Your score on this portion is worth 10 points, and it will be based on the number of predictions you get correct (total number not percentage).

Solution from Peter:

```r
# Split data into training, validation, and test sets
split_data <- function(
    data,
    train_perc = 0.5, val_perc = 0.25 ){
  # Get the number of rows in the data
  n_rows <- nrow(data)
  # Calculate the number of rows in each split
  train_rows <- round(train_perc * n_rows)
  val_rows <- round(val_perc * n_rows)
  test_rows <- n_rows - train_rows - val_rows
  # Randomly sample rows for each split
  train_idx <- sample(1:n_rows, size = train_rows, replace = FALSE)
  val_idx <- sample(setdiff(1:n_rows, train_idx), size = val_rows, replace = FALSE)
  test_idx <- setdiff(1:n_rows, c(train_idx, val_idx))
  # Create the training, validation, and test sets
  train_set <- data[train_idx, ]
  val_set <- data[val_idx, ]
  test_set <- data[test_idx, ]
  # Return the training, validation, and test sets
  return(
    list(
      train = train_set,
      val = val_set,
      test = test_set
    )
  ) }
```

Load in the data

```
raw_unimelb_training <- read_csv("unimelb_training.csv")
raw_unimelb_test <- read_csv("unimelb_test.csv")
dim(raw_unimelb_training)
get_rfcd_division <- function(RFCD_col){
  division <- ifelse(RFCD_col == 999999, NA, RFCD_col)
  division <- str_sub(division, start = 1, end = 2)
  division <- paste0(division,"0000")
  division <- factor(
    division,
    levels = c(
      '210000',
      '220000',
      '230000',
      '240000',
      '250000',
      '260000',
      '270000',
      '280000',
      '290000',
      '300000',
      '310000',
      '320000',
      '330000',
      '340000',
      '350000',
      '360000',
      '370000',
      '380000',
      '390000',
      '400000',
      '410000',
      '420000',
      '430000',
      '440000'
    ),
    labels = c(
      'Science',
      'Social Sciences, Humanities and Arts',
      'Mathematical Sciences',
      'Physical Sciences',
      'Chemical Sciences',
      'Earth Sciences',
      'Biological Sciences',
      'Information, Computing and Communication Sciences',
```

```r
      'Engineering and Technology',
      'Agricultural, Veterinary and Environmental Sciences',
      'Architecture, Urban Environment and Building',
      'Medical and Health Sciences',
      'Education',
      'Economics',
      'Commerce, Management, Tourism and Services',
      'Policy and Political Science',
      'Studies in Human Society',
      'Behavioural and Cognitive Sciences',
      'Law, Justice and Law Enforcement',
      'Journalism, Librarianship and Curatorial Studies',
      'The Arts',
      'Language and Culture',
      'History and Archaeology',
      'Philosophy and Religion'
    )
  )
  return(division)
}
get_seo_division <- function(seo_col){
  division <- ifelse(seo_col == 999999, NA, seo_col)
  division <- str_sub(division, start = 1, end = 2)
  division <- factor(
    division,
    levels = c(
      '61',
      '62',
      '63',
      '64',
      '65',
      '66',
      '67',
      '68',
      '69',
      '70',
      '71',
      '72',
      '73',
      '74',
      '75',
      '76',
      '77',
      '78'
    ),
    labels = c(
      'Defence',
      'Plant Production and Plant Primary Products',
      'Animal Production and Animal Primary Products',
      'Mineral Resources (Excl. Energy)',
      'Energy Resources',
      'Energy Supply',
      'Manufacturing',
      'Construction',
      'Transport',
      'Information and Communication Services',
      'Commercial Services and Tourism',
      'Economic Framework',
      'Health',
      'Education and Training',
      'Social Development and Community Services',
      'Environmental Policy Frameworks and other aspects',
      'Environmental Management',
      'Non-Oriented Research'
    )
  )
  return(division)
}
```

Pre-processing

```
raw_unimelb_training |>
  mutate_at(
    vars(
      Sponsor.Code,
      Grant.Category.Code,
      starts_with("Person.ID"),
      starts_with("Role"),
      starts_with("Country.of.Birth"),
      starts_with("Home.Language"),
      starts_with("Dept.No"),
      starts_with("Faculty.No"),
      starts_with("With.PHD"),
    ),
    as.factor
  ) |>
  mutate_at(
    vars(
      starts_with("RFCD.Code"),
    ),
    get_rfcd_division
  ) |>
  mutate_at(
    vars(
      starts_with("SEO.Code"),
    ),
    get_seo_division
  ) |>
  mutate(
    contract_band_code = factor(
      Contract.Value.Band...see.note.A,
      levels = LETTERS[1:17],
      ordered = T
    ),
    No..of.Years.in.Uni.at.Time.of.Grant.1 = factor(
      No..of.Years.in.Uni.at.Time.of.Grant.1,
      levels = c(
        "Less than 0",
        ">=0 to 5",
        ">5 to 10",
        ">10 to 15",
        "more than 15"
      ),
      ordered = T
    ),
    No..of.Years.in.Uni.at.Time.of.Grant.2 = factor(
      No..of.Years.in.Uni.at.Time.of.Grant.2,
      levels = c(
        "Less than 0",
        ">=0 to 5",
        ">5 to 10",
        ">10 to 15",
        "more than 15"
      ),
      ordered = T
    ),
  ) |>
  mutate_at(
    vars(
      Sponsor.Code,
      Grant.Category.Code,
      starts_with("Role"),
      starts_with("Country.of.Birth"),
      starts_with("Dept.No"),
      starts_with("Faculty.No"),
      starts_with("Person.ID")
    ),
    \(fac) fct_lump_min(fac, min = 30)
  ) |>
  dplyr::select(
    -Contract.Value.Band...see.note.A,
    -Start.date
  ) |>
  mutate_at(
    vars(contains("Percentage")),
    \(column){
      ordered(cut2(oneval = F,column, seq(0,90,10)))
    }
  ) |>
  mutate_at(
    vars(contains("Year.of.Birth")),
    \(column){
      ordered(cut2(oneval = F,column, c(-Inf,seq(1940,1980,10),Inf)))
    }
  ) |>
  mutate_at(
    vars(contains("Number.of.")),
    \(column){
      ordered(cut2(oneval = F,column, c(0:8,Inf)))
```

```r
    }
  ) |>
  mutate_at(
    vars(A..1, A.1, B.1, C.1, A..2, A.2, B.2, C.2),
    \(column){
      ordered(cut2(oneval = F,column, c(0:5,Inf)))
    }
  ) |>
  mutate_if(
    is.factor,
    fct_na_value_to_level
  ) -> unimelb_training
# lump factors
```

Model matrix

```r
y = unimelb_training$Grant.Status
model_dummies <- dummyVars( data = unimelb_training[,-1], ~0+.
)
x <- predict(model_dummies, unimelb_training[,-1])
dim(x)
```

Split data

```r
data_model <- data.frame(y,x)
set.seed(17)
data_splitted <- split_data(data_model)
data_train <- data_splitted$train
data_validation <- data_splitted$val
data_test <- data_splitted$test
rm(data_splitted)
```

Fit Layer 1

```r
set.seed(71)
layer1 <- c(seq(20,300,10))
auc_validation <- NULL
for (neurons in layer1) {
  model = neuralnet(
    y ~ .,
    data = data_train,
    hidden = neurons,
    linear.output = FALSE,
    lifesign = "minimal",
    err.fct = 'ce',
    learningrate = 0.1
  )
  # Predict in validation
  y_hat_validation <- predict(model, newdata = data_validation) |> as.vector()
  roc_validation <- roc(data_validation$y, y_hat_validation)
  auc_validation <- c(auc_validation,roc_validation$auc)
}
```

```
ggplot(aes(layer1, auc_validation), data = NULL) +
  geom_point() +
  geom_line()
data.frame(layer1, auc_validation) |>
  arrange(-auc_validation)
```

Chosen model

```
set.seed(717)
model = neuralnet(
  y ~ .,
  data = data_train,
  hidden = c(230),
  linear.output = FALSE,
  lifesign = "minimal",
  err.fct = 'ce',
  learningrate = 0.1
)
# Predict in validation
y_hat_validation <- predict(model, newdata = data_validation) |> as.vector()
roc_validation <- roc(data_validation$y, y_hat_validation)
roc_validation

# Finding the best cutoff

(best_cutoff <- coords(
  roc_validation,
  "best",
  best.method = "topleft"))

# Confusion in validation

table(
  predicted = 1*I(y_hat_validation > best_cutoff$threshold),
  reference = data_validation$y) |>
  confusionMatrix(positive = "1")
```

Solution from Guyin:

```
train<-read.csv("unimelb_training.csv",na.strings = "")
test<-read.csv("unimelb_test.csv",na.strings = "")
x<-replace(data[,41:55], is.na(data[,41:55]), 0)
data<-cbind(data[1:40],x)
##Add new variable "whether or not have a second project member",
"whether or not the first PI has successful grant", and
"whether or not the first PI has unsuccessful grant"
data1<-data|>
  mutate(person2=case_when(Person.ID.2!=0 | Role.2!=0 ~ "Yes",
                           Person.ID.2==0 & Role.2==0 ~ "No"),
         success.grant1=case_when(Number.of.Successful.Grant.1!=0  ~ "Yes",
                                  Number.of.Successful.Grant.1==0 ~ "No",
                                  Number.of.Successful.Grant.1==NA ~ NA),
         nsuccess.grant1=case_when(Number.of.Unsuccessful.Grant.1!=0  ~ "Yes",
                                    Number.of.Unsuccessful.Grant.1==0 ~ "No",
                                    umber.of.Unsuccessful.Grant.1==NA ~ NA),
         before = 41)

##Separate start_date
data1<-data1|>
  separate(Start.date, into = c("day", "month", "year"), sep = "/")
data1$day<-as.numeric(data1$day)
data1$month<-as.numeric(data1$month)
data1$year<-as.numeric(data1$year)

##Add new variable "season"
data1<-data1|>
  mutate(season=case_when(month>=1 & month <=3 ~ "Q1",
                          month>=4 & month <=6 ~ "Q2",
                          onth>=7 & month <=9 ~ "Q3",
                          month>=10 & month <=12 ~ "Q4"),
         before = 5)

##Remove ID
data1<-data1[,c(-29,-47)]
str(data1)

##Clean
data1 <- data1|>
  mutate_if(is.character, replace_na, replace = "unknown")

##Split data
y = as.matrix(data1[,1])
y = as.numeric(y)
x = model.matrix(~0 + Sponsor.Code+ Grant.Category.Code+
                   Contract.Value.Band...see.note.A+season+Role.1+
                   Country.of.Birth.1+Home.Language.1 +With.PHD.1+
                   No..of.Years.in.Uni.at.Time.of.Grant.1+
                   person2+success.grant1 +unsuccess.grant1 +
                   Role.2 +Country.of.Birth.2+Home.Language.2 +
                   With.PHD.2+No..of.Years.in.Uni.at.Time.of.Grant.2,
                 data = data1)

df <- data.frame(y,x)
train<-df[1:7599,]
test<-cbind(test$Proposal.num,df[7600:8708,])
set.seed(7687)
ids <- sample(1:dim(train)[1],size=dim(train)[1])
train_jumbled <- train[ids,]
train <- train_jumbled[1:5400,]
valid <- train_jumbled[5401:nrow(train_jumbled),]
y_train = as.matrix(train[,1])
y_valid = as.matrix(valid[,1])
x_train = as.matrix(train[,-1])


x_test = as.matrix(test[,-c(1:2)])
```

```r
Y_train = to_categorical(y_train,2)
Y_valid = to_categorical(y_valid,2)

###Fit model
model <- keras_model_sequential()
n <- nrow(x_train)
model %>%
  layer_dense(units = n/2, activation = 'relu',
              input_shape = c(ncol(x_train))) %>%
  layer_dropout(rate = 0.25) %>%
  layer_dense(units = n/10, activation = 'relu') %>%
  layer_dropout(rate = 0.2) %>%
  layer_dense(units = 2, activation = 'softmax')
summary(model)

model %>% compile(
  loss = 'categorical_crossentropy',
  metrics = c('accuracy'))
accuracy<-NULL

for(i in 1:30){
  set.seed(123)
  history <- model %>% fit(
    x_train, Y_train,
    epochs = 30, batch_size = 90,
    validation_split = 0.2)
  set.seed(123)
  results<-model %>% evaluate(x_valid, Y_valid)
  accuracy1<-unname(results[2])
  accuracy<-rbind(accuracy,accuracy1)
}

###Predict
y_pred<-model %>% predict (x_test) %>% k_argmax()
result<-as.numeric(y_pred[1:1109])
result<-cbind(test$`test$Proposal.num`,result)
colnames(result)<-c("Proposal.num","Pred.Grant.Status")
write.csv(result,"Guyin-Zhang.csv")

## 95% CI
mean<-mean(accuracy)
se<-sd(accuracy)/sqrt(30)
lower<-mean-se*1.96
upper<-mean+se*1.96

print(paste("Estimate value of accuracy is:", mean))
print(paste("95% CI of this estimate value is:", lower,upper))
```

2. In this problem, you are asked to do some K-means experiments on simulated data and deal with some practical issues, such as how to choose K and whether to do standardization or not before running the algorithm.

a. **(2 points)** Simulate 2-dimensional data from four clusters, each of which is specified by a Gaussian distribution with the same covariance matrix and different means. Using the code demonstrated in class to generate the data, though change the seed and make the variance of the means equal to 1 (the means are randomly generated).

Solution: To do this we'll use the code from class

```r
set.seed(1979)
library(clusterGeneration)
```

```
## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(EMCluster)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

##
## Attaching package: 'EMCluster'

## The following object is masked from 'package:dplyr':
##
##     recode
```

```r
M <- 1000 #number of subjects
num.vars <- 2 #number of outcomes
K <- 4 #number of clusters
prior_vec <- c(0.2,0.15,0.2,0.45)
#Variance of the K clusters
cls_var <-genPositiveDefMat(num.vars,covMethod=c("unifcorrmat"),rangeVar=c(0.2,5))$Sigma
round(cls_var,3)
```

2.514	1.542
1.542	1.414

```r
#Mean of the K clusters
clst_mean <- matrix(rnorm(K*num.vars,0,1),K,num.vars)
round(clst_mean,3)
```

0.608	-1.479
-0.631	-0.594
-0.285	-0.765

-0.342	-0.474

```
clust_mem <- sample(1:K,M,replace=TRUE,prob = prior_vec)
data1 <- NULL
for(i in 1:K){
  clus <- mvrnorm(n=table(clust_mem)[i],clst_mean[i,], cls_var)
  data1 <- rbind(data1,cbind(i,clus))
}
table(clust_mem)
```

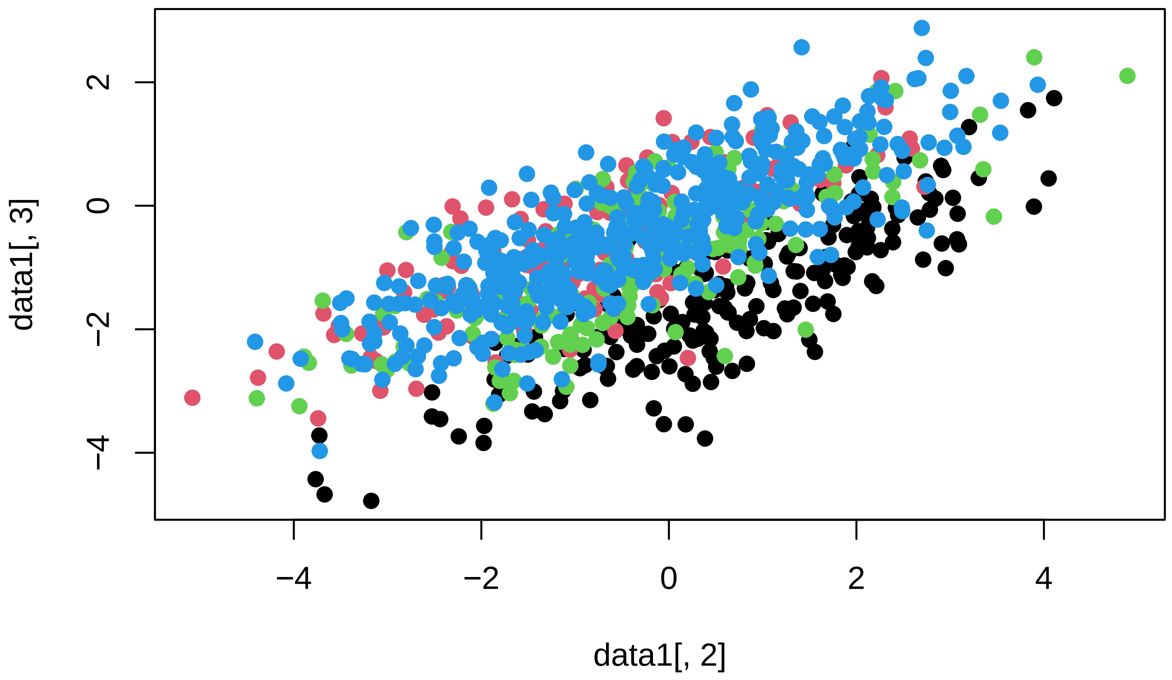
1	2	3	4
205	156	182	457

```
colnames(data1) <-c("Cluster",paste(rep("X",num.vars),1:num.vars,sep="_"))
head(data1)
```

Cluster	X_1	X_2
1	-2.2408046	-3.7351723
1	0.2221323	-2.1007186
1	2.0206140	-0.4134351
1	3.3056042	0.4496701
1	-3.1741425	-4.7788704
1	-1.4281855	-2.1294414

Now let's visualize a scatter plot of the clusters

```
plot(data1[,2],data1[,3],col=data1[,1],pch=19)
```



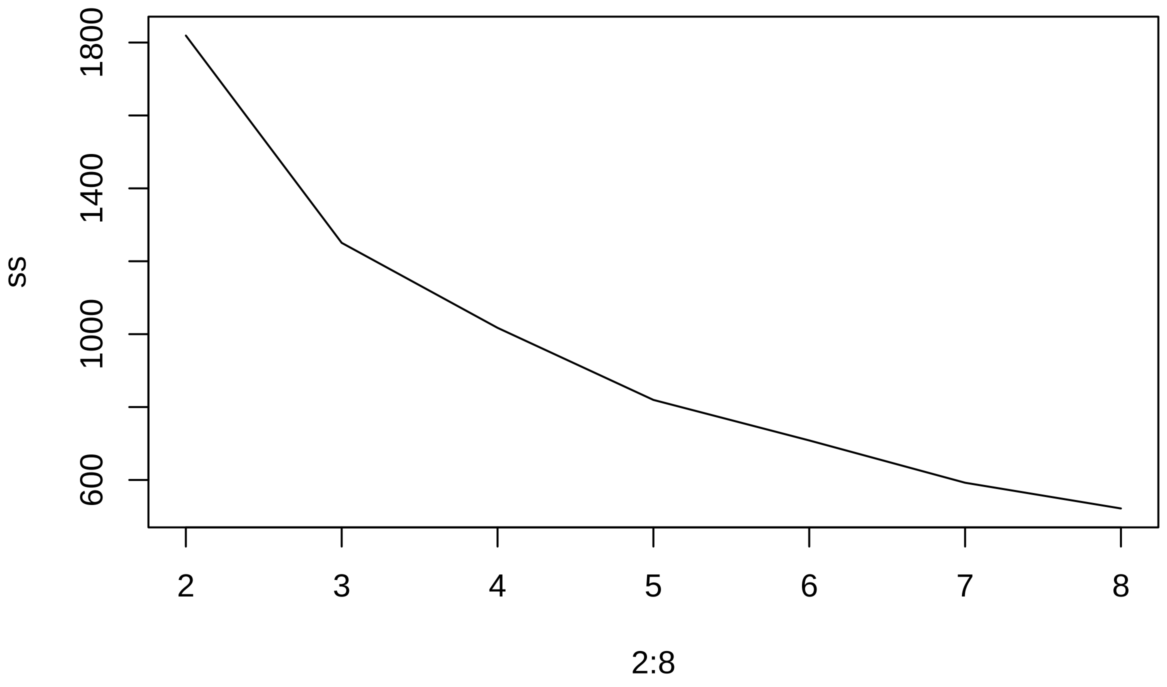
Clearly this is going to be a difficult clustering problem!

b. **(10 points)** Using Gaussian mixture model clustering, plot K (ranging from 2 to 6) against BIC value for two different covariance structures in the `mclust` package. Find how many points are in wrong clusters (you can also use the RAND index). Dont forget to record the sample covariance matrix of the sim- ulated data at this point.

c. **(5 points)** Using K-means clustering, plot K (ranging from 2 to 6) against within cluster sum of squares. Choose an appropriate K from the plot and argue why do you choose this particular K. Run the function K-means with chosen K and plot the clustering result. Write down how many points are in wrong clusters. Don't forget to record the sample covariance matrix of the simulated data at this point.

Solution: Now i'll look for the elbow in the number of clusters versus the intra-cluster sum of squares. This is a standard way of selecting the number of clusters.

```r
ss <- NULL
for(k in 2:8) {
  temp <- kmeans(data1[,2:3],centers=k)
  ss <- c(ss,temp$tot.withinss)
}
plot(2:8,ss,type="l")
```



The "biggest" elbow appears to be at $K = 3$ (though it's not much of one), we'll go with $K = 3$.

```r
est.class <- kmeans(data1[,2:3],centers=3)$cluster
temp <- RRand(data1[,1],est.class)
temp
```

```
##       Rand    adjRand     Eindex
##  0.5577598 -0.0006364  0.2145974
```

```r
table(est.class,data1[,1])
```

est.class/	1	2	3	4
1	38	52	61	133

est.class/	1	2	3	4
2	71	35	35	137
3	96	69	86	187

The Rand index gives us a value around 0.55, this means that approximately 55% of the pairs will be put in the same or different clusters if they are truely in the same or different true clusters, respectively. This value is **extremely** poor (just using a coin flip would give you a 50/50 chance). It appears that we have a tough clustering problem (or K-means just isn't workng very well).

d. **(5 points)** Now choose the number of clusters using the gap statistic. If the results differ from what you found in (a), run the function K-means with chosen K and plot the clustering result. Which method do you prefer?

Solution: Recall that the Gap estimate of k^* is the smallest k producing a gap within one standard deviation of the gap at $k+1$, i.e.

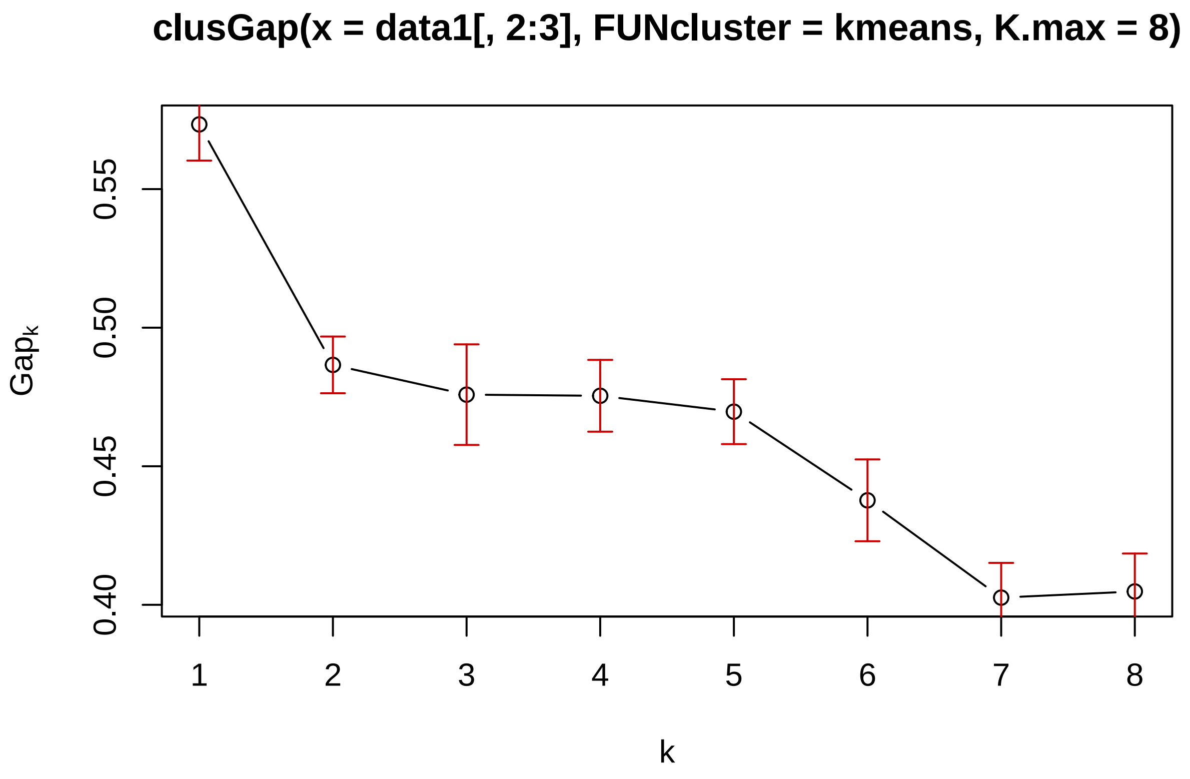
$$k^* = argmin_k\{k|G(k+1) - G(k) < s'_{k+1}\}$$

where $s_k\sqrt{1+1/B}$ and s_k is the standard deviation of B simulations of $\log W_k^*$.

```r
library(cluster)
test <- clusGap(data1[,2:3],FUNcluster = kmeans,K.max = 8)
#Calulate if formula above holds
test$Tab[2:8,3] - test$Tab[1:7,3]<test$Tab[2:8,4]
```

```
## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

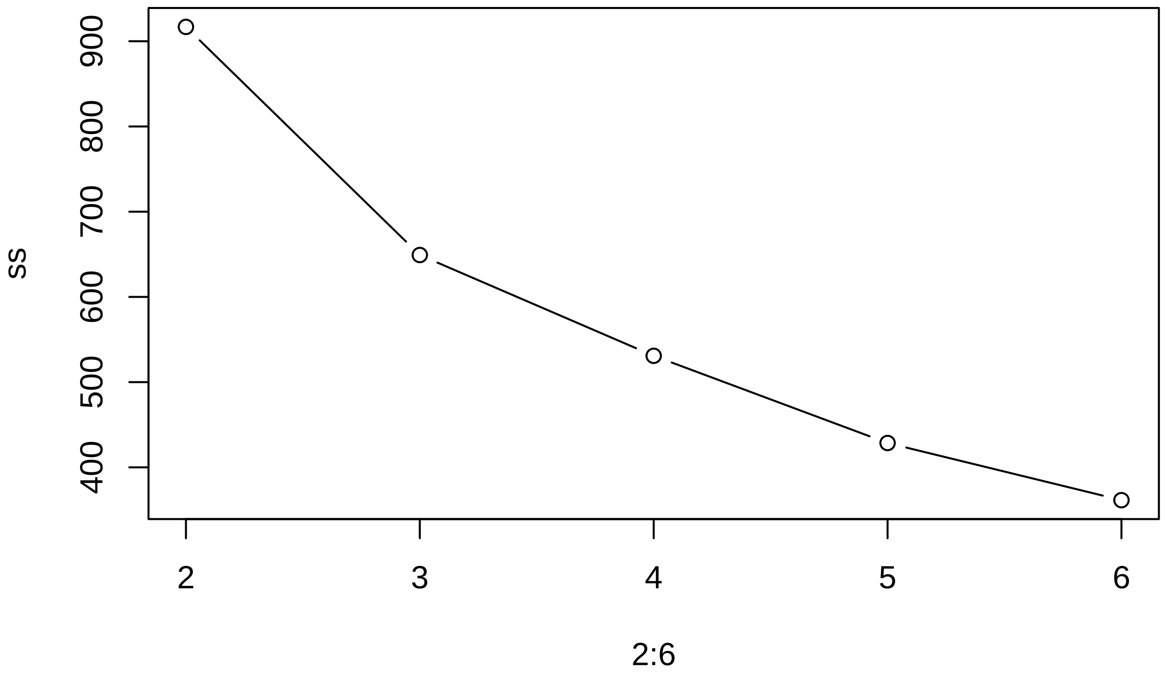
```r
#The smallest k where it's true is 1.  We can see this on the plot
plot(test)
```



So the Gap statistic doesn't find ANY clusters. This isn't that surprising based on the plot of the data above.

e. **(3 points)** Standardize the data. Now plot K against within cluster sum of squares on standardized data, choose an appropriate K from the plot. Is the K you choose the same as the one you chose in parts (b) and (c)? Run the function K-means with all chosen K's (if they differ) on the standardized data. Compare the clustering results with the one you obtained in parts (b) and (c).

```
data1[,2] <- scale(data1[,2])
data1[,3] <- scale(data1[,3])
ss <- NULL
for(k in 2:6) {
  temp <- kmeans(data1[,2:3],centers=k)
  ss <- c(ss,temp$tot.withinss)
}
plot(2:6,ss,type="b")
```

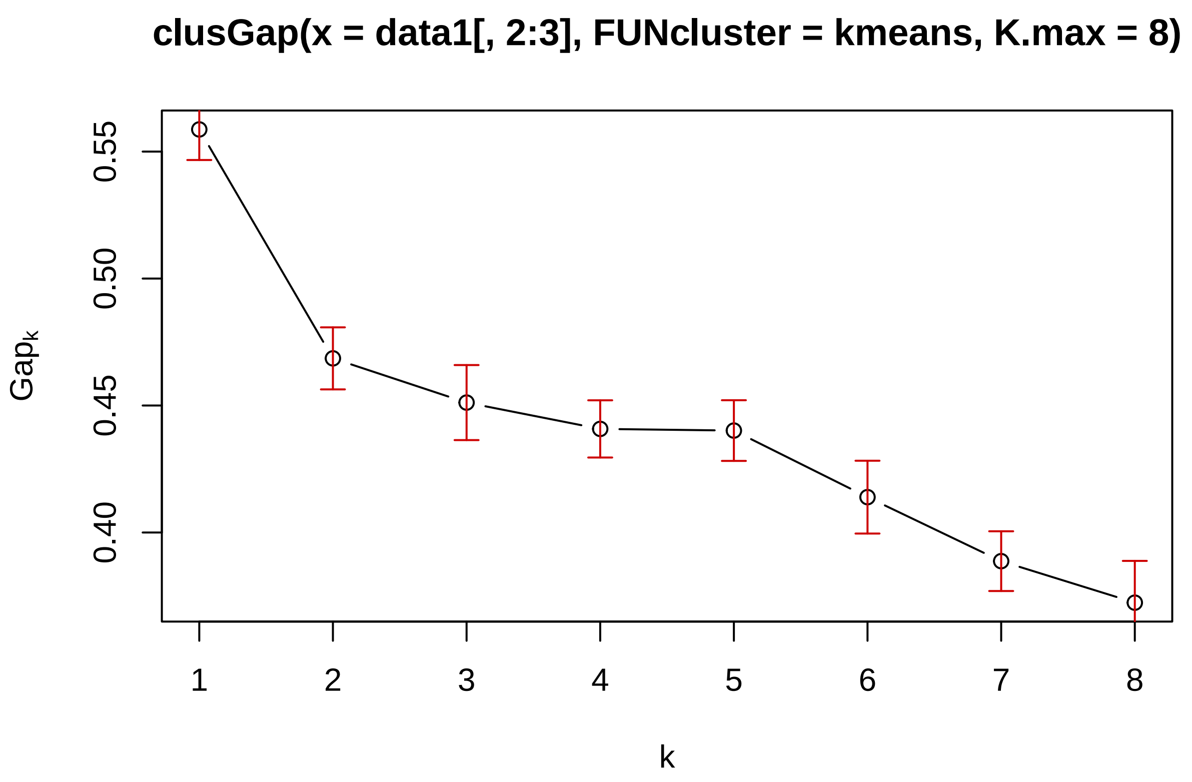


For this graph there till isn't an apparent elbow in the plot. Let's take a look at the gap statistic

```
library(cluster)
test <- clusGap(data1[,2:3],FUNcluster = kmeans,K.max = 8)
#Calulate if formula above holds
test$Tab[2:8,3] - test$Tab[1:7,3]<test$Tab[2:8,4]
```

```
## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
#The smallest k where it's true is 5.  We can see this on the plot
plot(test)
```



Again, it doesn't see any clusters. We'll use $K = 2$ just as an example.

```
est.class <- kmeans(data1[,2:3],centers=2)$cluster
temp <- RRand(data1[,1],est.class)
temp
```

```
##      Rand   adjRand    Eindex
##  0.499111 -0.001039  0.307920
```

```
table(est.class,data1[,1])
```

est.class/	1	2	3	4
1	96	69	79	229
2	109	87	103	228

Which still doesn't appear to do much better.

3. Consider a mixture model density in p-dimensional feature space,

$$g(x) = \sum_{k=1}^{K} \pi_k g_k(x)$$

where $g_k = N\left(\mu_k, \mathbf{L} \cdot \sigma^2\right)$ and $\pi_k \geq 0 \; \forall k$ with $\sum_k \pi_k = 1$. Here, $\{\mu_k, \pi_k\}$ for $k = 1, 2, \ldots, K$ and σ^2 are the unknown parameters.

Suppose we have data $x_1, x_2, \ldots, x_N \sim g(x)$ and we wish to fit the mixture model.

a. **(5 points)** Write down the log-likelihood of the data

Solution: the log-likelihood is given by

$$\begin{aligned}
\prod_{i=1}^{n} g\left(x_i\right) &= \prod_{i=1}^{n}\left(\sum_{k=1}^{K} \pi_k g_k\left(x_i\right)\right) \\
\log[p(X|\theta)] &= \log\left[\prod_{i=1}^{n}\left(\sum_{k=1}^{K} \pi_k g_k\left(x_i\right)\right)\right] = \sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} \pi_k g_k\left(x_i\right)\right)
\end{aligned}$$

b. **(5 points)** Derive an EM algorithm for computing the maximum likelihood estimates (see Section 8.1).

Solution: let $z_{ik} = 1$ is observation i is in cluster k and 0 otherwise. The complete data lot-likelihood (when the z_{ik} are known) is given by

$$\log[p(X|\theta)] = \sum_{i=1}^{n}\sum_{k=1}^{K}\left[z_{ik}\log\left(\pi_k\right) + z_{ik}\log\left(g_k\left(x_i\right)\right)\right]$$

The MLE's are giiven by

$$\hat{\pi}_k = \frac{\sum_{i=1}^{n} z_{ik}}{n},$$

$$\hat{\mu}_k = \frac{\sum_{i=1}^{n} z_{ik} x_i}{\sum_{i=1}^{n} z_{ik}}$$

and

$$\hat{\Sigma}_k = \frac{\sum_{i=1}^{n} z_{ik}(x_i - \hat{\mu}_k)'(x_i - \hat{\mu}_k)}{\sum_{i=1}^{n} z_{ik}}$$

Of course, the z_{ik} are not known. Thus, we look to find the expectation of the complete data log-likelihood given the previous parameter values and the data, which is

$$Q(\theta|\theta^{(m)}) = E\{\log[p(X|\theta)]|\mathcal{D}, \theta^{(m)}\} = \sum_{i=1}^{n}\sum_{k=1}^{K}\left[E\{z_{ik}|\mathcal{D}, \theta^{(m)}\}\log\left(\pi_k\right) + E\{z_{ik}|\mathcal{D}, \theta^{(m)}\}\log\left(g_k\left(x_i\right)\right)\right]$$

where

$$E\{z_{ik}|\mathcal{D}, \theta^{(m)}\} = \gamma_{ik}^{(m)} = \frac{\pi_k^{(m)} g_k\left(x_i|\theta^{(m)}\right)}{\sum_{\ell=1}^{K} \pi_\ell^{(m)} g_\ell\left(x_i|\theta^{(m)}\right)}$$

and $\theta^{(m)}$ are the current values of the parameters.

Thus, to estimate the parameters we start from some initial value of the parameters $\theta^{(0)}$, then we use that in our ***expectation step*** to calculate $\gamma_{ik}^{(m)}$ for all $i = 1, \ldots, n$ and $k = 1, ..., K$, then we calculate the parameters in the ***maximization step*** by putting $\gamma_{ik}^{(m)}$ in place of z_{ik} in the formulas for $\hat{\pi}_k$, $\hat{\mu}_k$, and Σ_k yielding $\pi_k^{(m)}$, $\mu_k^{(m)}$, and $\Sigma_k^{(m)}$. This is repeated for $m = 1, 2, \ldots$ until the parameters converge.

c. **(5 points)** Show that if σ has a known value in the mixture model and we take $\sigma \to 0$, then in a sense this EM algorithm coincides with K-means clustering.

Solution: to see the connection note that if $\gamma_{ik} = 0$ or 1 for all $i = 1, \dots, n$ and $k = 1, ..., K$, then the above EM algorithm is exactly same as K-means clustering. So, it suffices to show that as $\sigma \to 0$, $\gamma_{ik} \to 0$ or 1 for all $i = 1, \dots, n$ and $k = 1, ..., K$. Under the normality assumption (and assuming a constant σ for all clusters) after removing the constant terms we have

$$\gamma_{ik} = \frac{\pi_k \exp\left(-\frac{1}{2}\left(\frac{x_i-\mu_k}{\sigma}\right)^2\right)}{\sum_{j=1}^{K} \pi_j \exp\left(-\frac{1}{2}\left(\frac{x_i-\mu_j}{\sigma}\right)^2\right)}$$

Let's instead look at

$$\alpha_{ik\ell} = \frac{\pi_\ell \exp\left\{-\frac{1}{2}\left(\frac{x_i-\mu_\ell}{\sigma}\right)^2\right\}}{\pi_k \exp\left\{-\frac{1}{2}\left(\frac{x_i-\mu_k}{\sigma}\right)^2\right\}} = \pi_\ell/\pi_k \exp\left\{-\frac{1}{2\sigma^2}\left[(x_i-\mu_\ell)^2 - (x_i-\mu_k)^2\right]\right\}$$

Noting that $\gamma_{ik} = 1/(\sum_{\ell=1}^{K} \alpha_{ik\ell})$. Let $k^* = \text{argmin}_k\{(x_i-\mu_k)^2\}$ and note that $\alpha_{ikk} = 1$. Then, all we have to show is that $\alpha_{ik^*\ell} \to 0$ for all $\ell \neq k^*$, since this will imply:

- $\gamma_{ik^*} \to 1/\alpha_{ik^*k^*} = 1$ (i.e., the probability goes to 1 for the cluster that has the closest mean) and
- $\alpha_{i\ell k^*} = 1/\alpha_{ik^*\ell} \to \infty$ thus $\gamma_{i\ell} \to 0$ (i.e., the probability goes to 0 for all clusters that do not have closest mean).

So, let $k^* = \text{argmin}_k\{(x_i-\mu_k)^2\}$ and $\ell \neq k^*$. Then for $c > 0$ we have

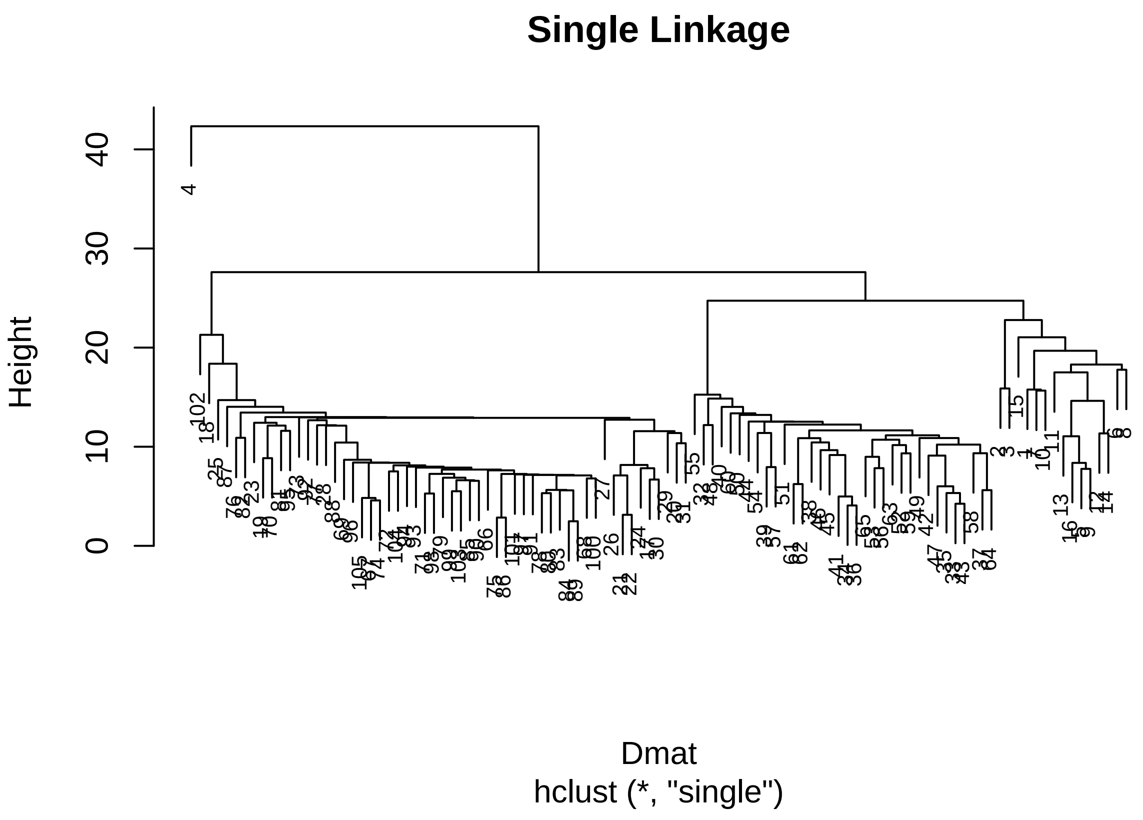
$$\begin{aligned}
\alpha_{ik^*\ell} = \exp\left\{-\frac{1}{2\sigma^2}\left[(x_i-\mu_\ell)^2 - (x_i-\mu_{k^*})^2\right]\right\} &= \exp\left\{-\frac{1}{2\sigma^2}\left[(x_i-\mu_{k^*}+c)^2 - (x_i-\mu_{k^*})^2\right]\right\} \\
&= \exp\left\{-\frac{1}{2\sigma^2}\left[(x_i-\mu_{k^*})^2 + c^2 + 2c(x_i-\mu_{k^*}) - (x_i-\mu_{k^*})^2\right]\right\} \\
&= \exp\left\{-\frac{d}{2\sigma^2}\right\} \to 0 \qquad (1)
\end{aligned}$$

as $\sigma \to 0$ where $d = c^2 + 2c(x_i - \mu_{k^*}) > 0$ as desired.

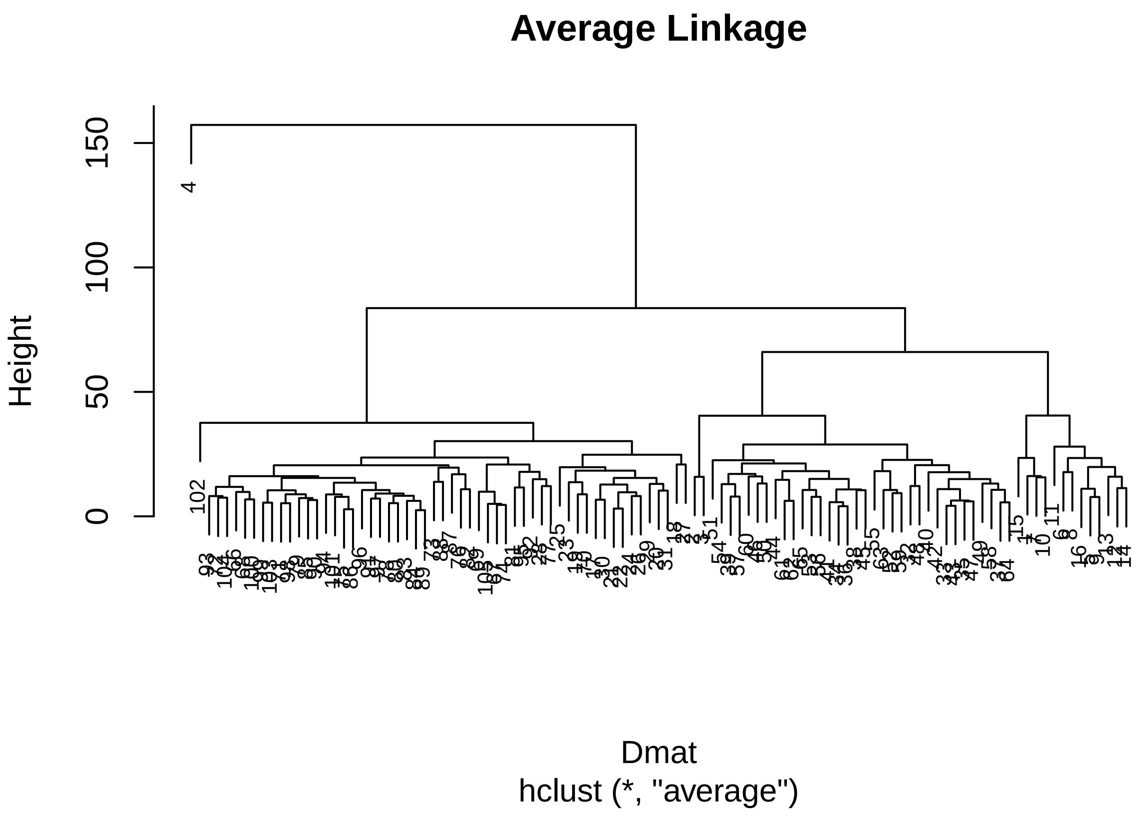
4. **(20 points)** (Question 12.3 from "Modern Multivariate Statistical Techniques"" by Izenman) Cluster the **primate.scapulae** data using single, average, and complete-linkage agglomerative clustering methods. Find the five-cluster solutions for al three methods, which allows comparison with the true primate classifications. Show that the orderings of the methods in terms of their misclassification rates (you can also use the RAND index).

Solution: below we do this clustering for all three methods:

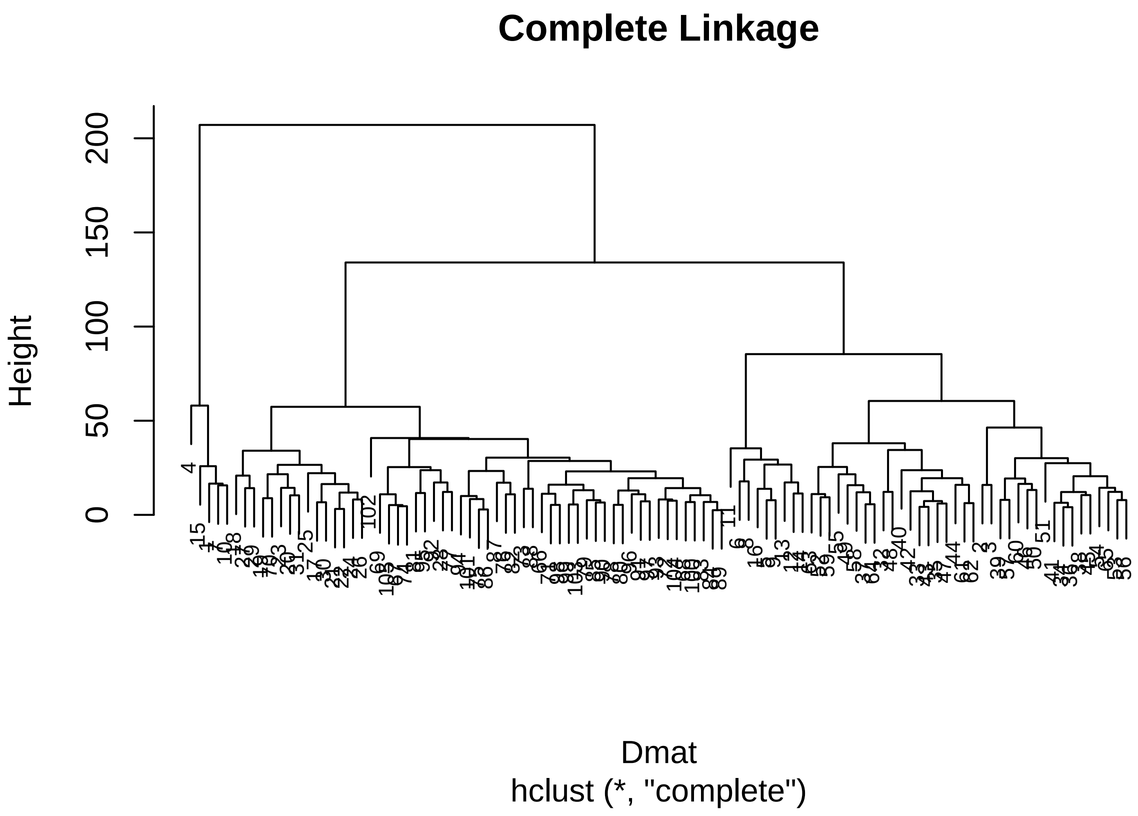
```r
primate = read.table("primate.scapulae.txt",sep = " ", header = T)
X = subset(primate, select = -c(class,gamma))
Y = primate$class
#Euclidean distance
Dmat = dist(X)
#Single linkage
sing.hclust = hclust(Dmat, method = 'single')
plot(sing.hclust,cex=.7,main="Single Linkage")
```



```r
#Average linkage
ave.hclust = hclust(Dmat,method="average")
plot(ave.hclust,cex=.7,main="Average Linkage")
```

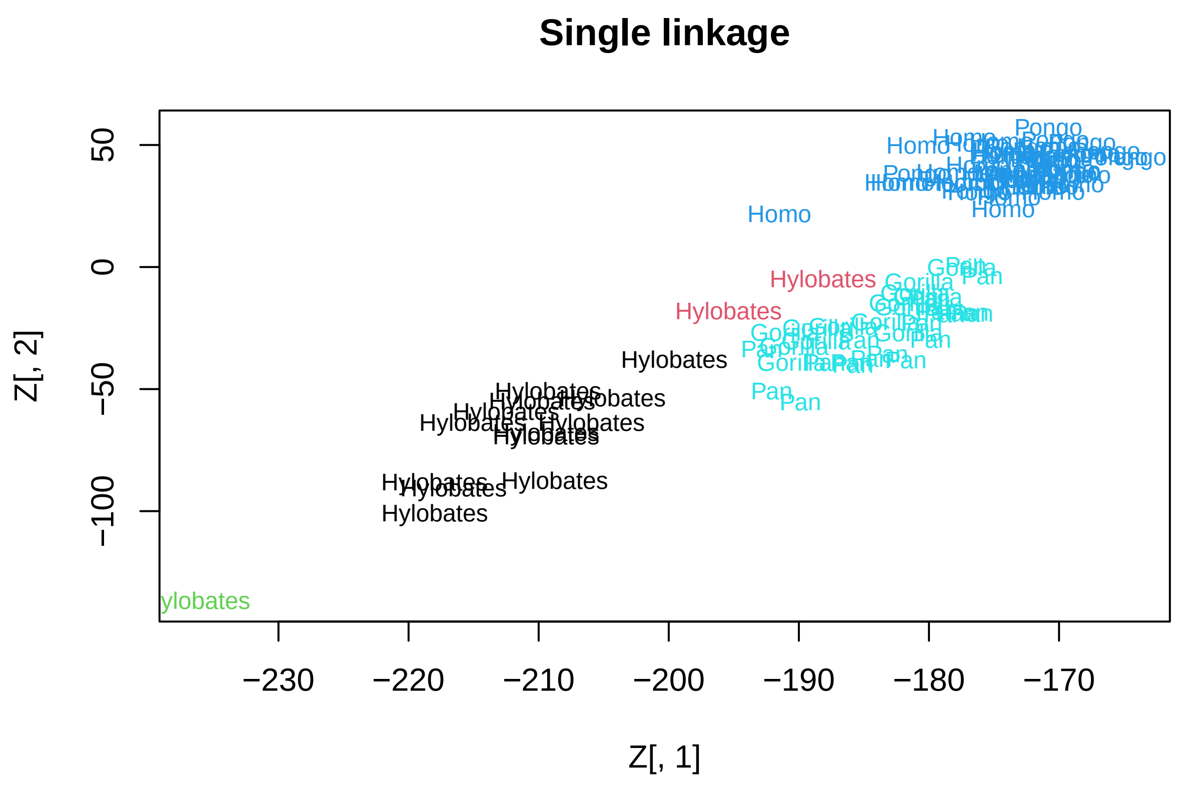


```
#Complete linkage
com.hclust = hclust(Dmat,method="complete")
plot(com.hclust,cex=.7,main="Complete Linkage")
```

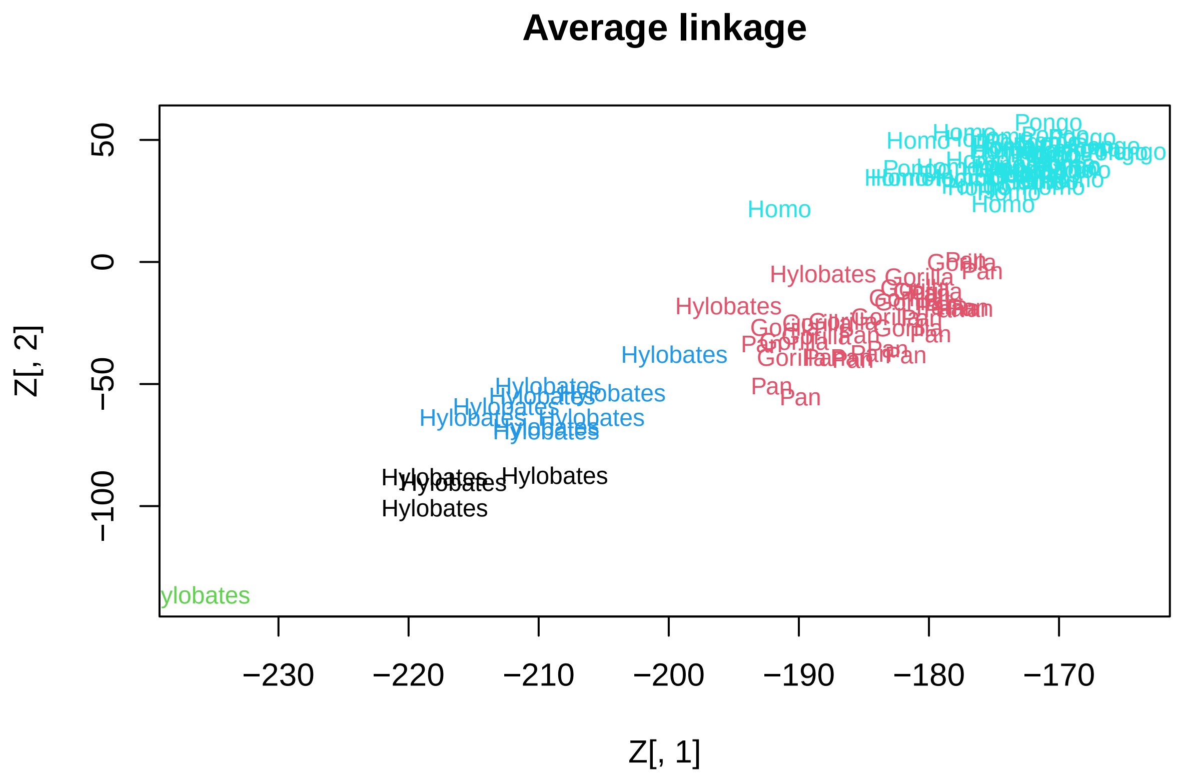


To find the five-cluster solutions, we need some cutting of the tree. Here we'll cut the tree, then display the results along the first two principal components.

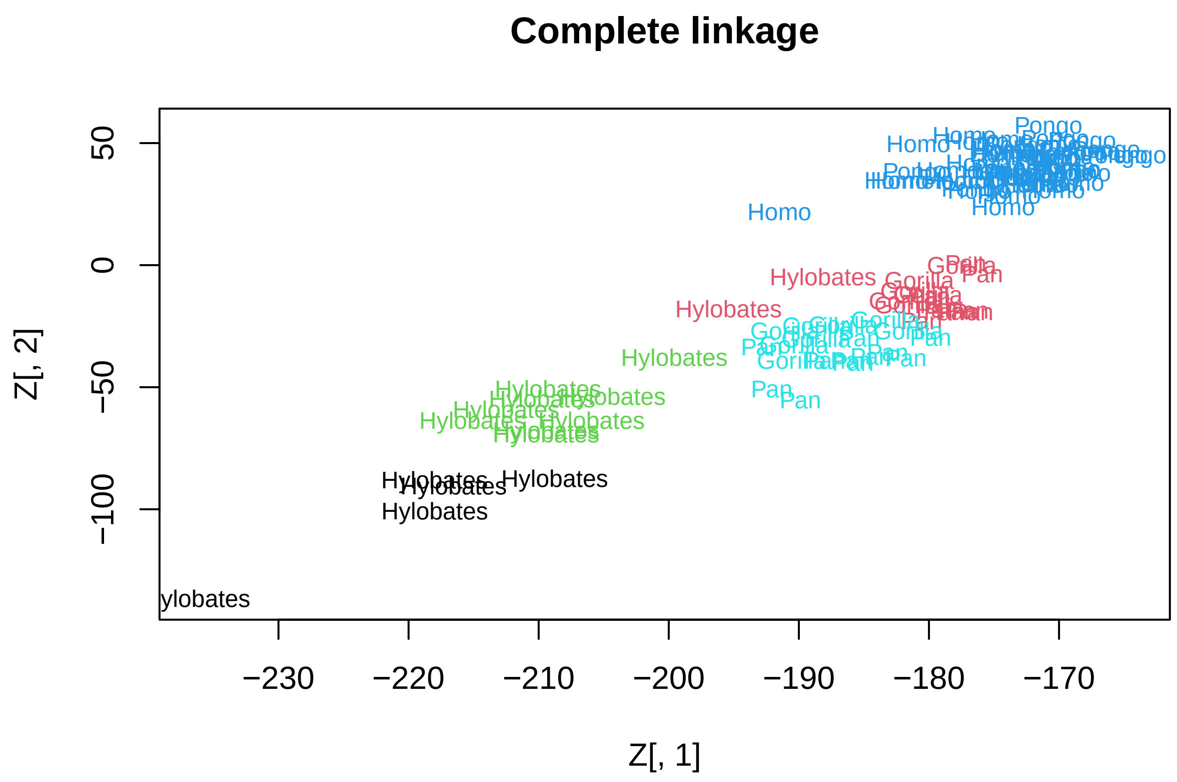
```
#SVD of data
sv = svd(X);
U = sv$u
V = sv$v
D = sv$d
Z = as.matrix(X)%*%V;
#For single linkage
sing.hclust_cut <- cutree(sing.hclust,k=5)
plot(Z[,1],Z[,2],col=sing.hclust_cut,type="n",main = 'Single linkage')
text(Z[,1],Z[,2],primate$class,cex=.75,col=sing.hclust_cut)
```



```r
#For average linkage
ave.hclust_cut <- cutree(ave.hclust,k=5)
plot(Z[,1],Z[,2],col=ave.hclust_cut,type="n",main = 'Average linkage')
text(Z[,1],Z[,2],primate$class,cex=.75,col=ave.hclust_cut)
```



```r
#For complete linkage
com.hclust_cut <- cutree(com.hclust,k=5)
plot(Z[,1],Z[,2],col=com.hclust_cut,type="n", main = 'Complete linkage')
text(Z[,1],Z[,2],primate$class,cex=.75,col=com.hclust_cut)
```



Now, we'll look at the linkage classifications against the truth

```r
library(dplyr)
#Recode five classes to 1 to 5 for classification
Y = dplyr::recode(primate$class, 'Hylobates'='5','Pongo'='4',
                  'Pan'='3','Gorilla'='2','Homo'='1')
#Single linkage
sing.RRand = RRand(sing.hclust_cut,as.integer(Y))
sing.RRand
```

```
##    Rand adjRand  Eindex
##  0.8313  0.6178  0.1813
```

```r
table(sing.hclust_cut,as.integer(Y))
```

sing.hclust_cut/	1	2	3	4	5
1	0	0	0	0	13
2	0	0	0	0	2
3	0	0	0	0	1
4	40	0	0	15	0
5	0	14	20	0	0

```r
#Average linkage
ave.RRand = RRand(ave.hclust_cut,as.integer(Y))
ave.RRand
```

```
##    Rand adjRand  Eindex
##  0.8123  0.5777  0.1840
```

```
table(ave.hclust_cut,as.integer(Y))
```

ave.hclust_cut/	1	2	3	4	5
1	0	0	0	0	4
2	0	14	20	0	2
3	0	0	0	0	1
4	0	0	0	0	9
5	40	0	0	15	0

```
#Complete linkage
com.RRand = RRand(com.hclust_cut,as.integer(Y))
com.RRand
```

```
##    Rand adjRand  Eindex
##  0.8183  0.5607  0.1568
```

```
table(com.hclust_cut,as.integer(Y))
```

com.hclust_cut/	1	2	3	4	5
1	0	0	0	0	5
2	0	6	9	0	2
3	0	0	0	0	9
4	40	0	0	15	0
5	0	8	11	0	0

For the Rand index (and the adjusted Rand) the single linkage works the best. Using "misclassification" (which is difficult here) the most observations correctly classified is 88 for single, 84 for average, and 81 for complete.