Assignment 3 - CSC/DSC 265/465 - Spring 2020 - Due April 29

Q1: Suppose a logistic regression model is fit with response $Y =$ `Minor injury during the past year` and one predictor $X =$ `age`. The sample size is $n = 100$ and the range of X in years is $[3.2, 18.1]$. The data was fit using the `R glm()` function, and produced the following output:

```
> fit = glm(y ~ x, family='binomial')
> summary(fit)

Call:
glm(formula = y ~ x, family = "binomial")

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.0094  -0.6021  -0.4758  -0.3770   2.3251

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.89379    1.09305  -3.562 0.000368 ***
x            0.19859    0.09147   2.171 0.029927 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 84.542  on 99  degrees of freedom
Residual deviance: 79.500  on 98  degrees of freedom
AIC: 83.5

Number of Fisher Scoring iterations: 5
```

What is the estimated odds ratio for probability of `Minor injury during the past year` between 15 year old and 5 year old subjects? Give an approximate 95% confidence interval.

SOLUTION: In general, for logistic regressiom with binary response y and a single predict variable x, we have

$$\begin{aligned} P(y=1) &= \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}}, \text{ and} \\ Odds(y=1) &= \frac{P(y=1)}{1-P(y=1)} = e^{\beta_0+\beta_1 x}. \end{aligned}$$

Given two predictor observations x', x'' the odds ratio between them is therefore

$$OR(y=1; x', x'') = \frac{e^{\beta_0+\beta_1 x'}}{e^{\beta_0+\beta_1 x''}} = e^{\beta_1(x'-x'')}$$

For this problem we have $x' - x'' = 15 - 5$. We therefore need to estimate

$$OR = e^{10\beta_1}.$$

From the given output we have estimate $\hat{\beta}_1 = 0.19859$, and standard error $S_{\hat{\beta}_1} = 0.09147$. We then an approximate 95% confidence interval for β_1

$$CI_{\hat{\beta}_1} = [\hat{\beta}_1 - 2S_{\hat{\beta}_1}, \hat{\beta}_1 + 2S_{\hat{\beta}_1}] = [0.01565, 0.38153].$$

So, by substituting the estimate of β_1 into the expression for OR, we obtain estimate

$$\hat{OR} = e^{10\hat{\beta}_1} = e^{10\times 0.19859} \approx 7.286.$$

To obtain a confidence interval for OR we may substitute the upper and lower confidence bounds for β_1 into the expression for OR. This gives

$$CI_{\hat{OR}} = [e^{10\times 0.01565}, e^{10\times 0.38153}] \approx [1.169, 45.390].$$

Q2: For this question, use the `NCI60` data set from the `ISLR` package. From the `help` page:

```
NCI microarray data. The data contains expression levels on 6830 genes
from 64 cancer cell lines. Cancer type is also recorded.
```

The object `NCI60$data` is a 64×6830 matrix. Each row represents a cancer cell line, and contains 6830 gene expression measurements. Assume that the measurements are already standardized. The object `NCI60$labs` is a vector of length 64 containing text identifying the type of cancer (`OVARIAN`, `MELANOMIA`, and so on).

(a) Use the `hclust()` function (from library `stats`) to create a hierarchical clustering of the cancer cell lines. Use the option `method = 'average'` to specify the average distance agglomeration method. Plot the dendogram using labels from the `NCI60$labs` object.

(b) From the dendogram, it can be seen that some cancer types cluster more definititively than others. The tendency of a cancer type to cluster can be quantified by comparing the maximum cophenetic distance between samples of this type, and the minimum cophenetic distance between a sample of this type and a sample not of this type. Determine these quantities for the `MELANOMA`, `RENAL` and `COLON` cancer types, and comment briefly on what you find.

SOLUTION:

(a) The plot can be produced by the following code. See Figure 1.

```
library(ISLR)
library(class)

x = NCI60$data
y = NCI60$labs

par(mfrow=c(1,1))
hfit = hclust(dist(x),method='average')
plot(hfit,labels=y)
```

(b) Suppose we have n observations, and an $n \times n$ distance matrix D, such that the i,jth entry D_{ij} is the distance between observations i and j. Let A be a subset of indices $\{1, \ldots, n\}$ representing a possible cluster. In our example, D is the cophenetic distance matrix. The *maximum within-cluster distance* for A, say d_{max}, is the maximum distance between two observations in A:

$$d_{max} = \max_{i,j \in A} D_{ij}.$$

The *minimum without-cluster distance* for A, say d_{min}, is the minimum distance between an observation in A and an observation not in A:

$$d_{min} = \min_{i \in A, j \notin A} D_{ij}.$$

To evaluate d_{min}, d_{max} for (true) clusters `MELANOMA`, `RENAL` and `COLON` the following code may be used:

```
> # Get cophenetic distance matrix
>
> coph.dist = as.matrix(cophenetic(hfit))
>
> # Create labels
>
> my.labs = c("MELANOMA", "RENAL", "COLON")
>
> # Create a loop through the labels
>
> coph.tab = NULL
> for (my.lab in my.labs) {
+
+   # Get the maximum entry in coph.dist between
+   # two observations with label == my.lab
+
+   max.coph = max(coph.dist[y==my.lab,y==my.lab])
+
+   # Get the minimum entry in coph.dist between
+   # an observations with label == my.lab and an
+   # observaton with label != my.lab
+
+   min.coph = min(coph.dist[y==my.lab,y!=my.lab])
+
+   # store the values in table coph.tab
+
+   coph.tab = rbind(coph.tab, c(max.coph, min.coph))
+ }
>
> # Format and display coph.tab
>
> rownames(coph.tab) = my.labs
> colnames(coph.tab) = c('MAX within','MIN without')
> coph.tab
         MAX within MIN without
MELANOMA   90.80837    71.27390
RENAL      90.31513    77.36448
COLON      78.03069    84.81252
>
```

The values of d_{min}, d_{max} `MELANOMA`, `RENAL` and `COLON` are shown in the output. If a cluster is consistent, we should have $d_{min} \geq d_{max}$. If this condition holds, then any observation in A will be closer to any other observation in A then to any other observation not in A. However, this is only true for `COLON`. Looking at the dendogram (Figure 1) a definitive cluster of `COLON` can be observed towards the right side of the plot. In addition, there is no other `COLON` observation oustide this cluster (there are 7 in total). However, this is not the case for `MELANOMA` or `RENAL`.

Q3: For this question, use the `mammals` dataset from the `MASS` library. This data frame contains average body and brain weights for 62 species of land mammals (in kilograms and grams respectively). The names of the mammals can be accessed using `row.names(mammals)`.

(a) First, log-transform the data. Then for each $K = 1, \ldots, 10$ calculate a K-means cluster solution based on the two log-transformed features. Use option `nstart=100`. For each solution calculate $R^2 = 1 - SS_{within}/SS_{total}$, and plot these values against K. Identify the smallest value of K, say K^*, for which $R^2 \geq 0.8$.

(b) Draws a scatterplot of the features, and superimpose the centers output with the clustering solution for K^*. Distinguish the observations by using separate colors for the clusters identified by the solution. These clusters can be identified as follows:

```
fit = kmeans(x,centers=nc,nstart=100)
fit$cluster
```

Create separate lists of the species names for each cluster. Does the clustering make sense? Comment briefly.

SOLUTION:

(a) The following code may be used to create the plot. See Figure 2. Note that

$$R^2 = 1 - SS_{within}/SS_{total} = SS_{between}/SS_{total}.$$

From Figure 2, the smallest value of K for which $R^2 \geq 0.8$ is $K^* = 3$.

```
library(MASS)
library(class)

# Transform data

x = mammals
x = log(x)

# Calculate a K-means clustering for K = 1,...,10
# Calculate R2 for each

r2 = rep(0,10)
for (i in 1:10) {
  fit = kmeans(x,centers=i,nstart=100)
  r2[i] = fit$betweenss/fit$totss
}

# Create plot

par(mfrow=c(1,1))
plot(r2,type='b',ylim=c(0,1),xlab="K",ylab="R-sq")
abline(h=0.8,col='gray')
```

(b) The following code may be used to create the plot. See Figure 3. The clusters clearly divide the species on the basis of size. Cluster 1 consists mostly of rodents, while cluster 3 contains larger apes, equine mammals, and so on.

```
> # Recreate fit for K* = 3 clusters
>
> fit = kmeans(x,centers=3,nstart=100)
>
> # Create a single data set by combing the feature data and the
> # fitted cluster centroids
>
> xfit = rbind(x,fit$centers)
>
> # get cluster IDs, define distinct colors aand plotting characters.
>
> colv=c(fit$cluster+1,rep(1,nc))
> pchv=c(rep(3,ns),rep(19,nc))
>
> # Create plot
>
> par(mfrow=c(1,1))
> plot(xfit,col=colv,pch=pchv)
>
> # list species by cluster
>
> split(row.names(mammals),fit$cluster)
$`1`
 [1] "Ground squirrel"           "Lesser short-tailed shrew" "Star-nosed mole"
 [4] "Big brown bat"             "Galago"                    "Golden hamster"
 [7] "Mouse"                     "Little brown bat"          "Rat"
[10] "E. American mole"          "Mole rat"                  "Musk shrew"
[13] "Tree shrew"

$`2`
 [1] "Arctic fox"                "Owl monkey"                "Mountain beaver"
 [4] "Guinea pig"                "Verbet"                    "Chinchilla"
 [7] "Arctic ground squirrel"    "African giant pouched rat" "Nine-banded armadillo"
[10] "Tree hyrax"                "N.A. opossum"              "European hedgehog"
[13] "Cat"                       "Genet"                     "Rock hyrax-a"
[16] "Water opossum"             "Yellow-bellied marmot"     "Slow loris"
[19] "Rabbit"                    "Desert hedgehog"           "Rock hyrax-b"
[22] "Raccoon"                   "Echidna"                   "Tenrec"
[25] "Phalanger"                 "Red fox"

$`3`
 [1] "Cow"              "Grey wolf"        "Goat"             "Roe deer"         "Asian elephant"
 [6] "Donkey"           "Horse"            "Patas monkey"     "Giraffe"          "Gorilla"
[11] "Grey seal"        "Human"            "African elephant" "Rhesus monkey"    "Kangaroo"
[16] "Okapi"            "Sheep"            "Jaguar"           "Chimpanzee"       "Baboon"
[21] "Giant armadillo"  "Pig"              "Brazilian tapir"

>
```

Q4: For this question, use the `Khan` data set from the `ISLR` package. From the `help` page:

```
The data consists of a number of tissue samples corresponding to four distinct types
of small round blue cell tumors. For each tissue sample, 2308 gene expression
measurements are available.

...

Format

The format is a list containing four components: xtrain, xtest, ytrain, and ytest.
xtrain contains the 2308 gene expression values for 63 subjects and ytrain records
the corresponding tumor type. ytrain and ytest contain the corresponding testing
sample information for a further 20 subjects.
```

Consolidate the training and test data into a single dataset:

```
> library(ISLR)
> library(class)
>
> x = rbind(Khan$xtrain,Khan$xtest)
> y = c(Khan$ytrain,Khan$ytest)
> dim(x)
[1]   83 2308
> length(y)
[1] 83
```

There is now a single data set. The object `x` is an 83×2308 table, with 2308 gene expression measurements for each of 83 tissue samples. Then `y` is a vector of length 83 containing the tumor type, labeled 1 to 4.

(a) Use the `prcomp()` function (from library `stats`) to create a matrix of principal components, using the gene expressions as a feature set. Use centering, but not scaling.

(b) Build a KNN classifier for tumor type based on (a) the entire set of gene expressions; and (b) the first 10 principal components. Use the method of **Q4** of Assignment 2, using `k.list = seq(1,50,1)`. For each analysis plot CE against K (the neighborhood size). Report only CE. Which classifier is preferable (give several reasons for your answer)?

SOLUTION:

(a) The following code may be used to create the plot.

```
# Assemble data

x = rbind(Khan$xtrain,Khan$xtest)
y = c(Khan$ytrain,Khan$ytest)
dim(x)
length(y)

# Calculate the PCA

prc<-prcomp(x, scale.=F)
```

(b) The following code creates the required plots and evaluations. See Figure 4.

```
>
>  # Use the following function to calculate CE
>
> lrfp = function(m) { (m[2,2]/(m[1,2]+m[2,2]))/(m[2,1]/(m[1,1]+m[2,1]))  }
> lrfn = function(m) { (m[1,2]/(m[1,2]+m[2,2]))/(m[1,1]/(m[1,1]+m[2,1]))  }
> f0 = function(m) {c(1-sum(diag(m))/sum(m),lrfp(m),lrfn(m))}
>
> # Use the following KNN function
>
> knn.function = function(k.list, xtrain, gr) {
+
+     pr.tab = matrix(NA,length(k.list),3)
+     for (i in 1:length(k.list)) {
+         knn.fit = knn.cv(xtrain,gr,k=k.list[i],use.all=T)
+         cm = table(knn.fit,gr)
+         pr.tab[i,] = f0(cm)
+     }
+     return(pr.tab)
+ }
>
> par(mfrow=c(2,1))
>
> # First, use entire data set
>
> k.list = seq(1,50,1)
> knn.fit = knn.function(k.list,x,y)
>
> # Plot CE vs K, then identify optimal K
>
> plot(k.list, knn.fit[,1],xlab="K",ylab="CE")
> title("Complete Dataset")
> min(knn.fit[,1])
[1] 0.08433735
> which.min(knn.fit[,1])
[1] 3
>
> # Next, use first 10 principal components
>
> k.list = seq(1,50,1)
> knn.fit = knn.function(k.list,prc$x[,c(1:10)],y)
>
> # Plot CE vs K, then identify optimal K
>
> plot(k.list, knn.fit[,1],xlab="K",ylab="CE")
> title("First 10 principal components")
> min(knn.fit[,1])
[1] 0.06024096
> which.min(knn.fit[,1])
[1] 3
```

>

The plots are shown in Figure 4. From the output above, for the full data model we have $CE = 0.084$ for $K = 3$, and by using the first 10 principal components we have $CE = 0.060$. The CE for the 10 PC model is smaller than for the full data. In addition, the size of the input data set is reduced considerably, from 2308 feature to 10.

Q5: [**For Graduate Students**] Explain why scaling makes a difference for K-means clustering but not linear discriminant analysis.

SOLUTION: For LDA the class j discriminating function for $x \sim N(\mu_j, \Sigma_j)$, $x \in R^q$ is

$$h_j(x) = x^T \Sigma^{-1} \mu_j - \frac{1}{2} \mu_j^T \Sigma^{-1} \mu_j + \log(\pi_j)$$

Suppose the data is transformed to $y = Ax + b$, where A is a $q \times q$ matrix and b is a vector of length q. Then the new mean vector will be

$$\mu_j^* = A\mu_j + b$$

and the new covariance matrix will be

$$\Sigma^* = A\Sigma^* A^T$$

The new discriminant function will be

$$\begin{aligned}
h_j^*(x) &= y^T [\Sigma^*] \mu_j^* - \frac{1}{2} (\mu_j^*)^T [\Sigma^*]^{-1} \mu_j^* + \log(\pi_j) \\
&= (Ax + b)^T [A^T]^{-1} \Sigma^{-1} A^{-1} (A\mu_j + b) - \frac{1}{2} (A\mu_j + b)^T [A^T]^{-1} \Sigma^{-1} A^{-1} (A\mu_j + b) + \log(\pi_j) \\
&= x^T \Sigma^{-1} \mu_j + x^T \Sigma^{-1} A^{-1} b + b^T [A^T]^{-1} \Sigma^{-1} \mu_j^T + b^T [A^T]^{-1} \Sigma A^{-1} b \\
&\quad - \frac{1}{2} \left\{ \mu_j^T \Sigma^{-1} \mu_j + 2 b^T [A^T]^{-1} \Sigma^{-1} \mu_j^T + b^T [A^T]^{-1} \Sigma A^{-1} b \right\} + \log(\pi_j) \\
&= x^T \Sigma^{-1} \mu_j - \frac{1}{2} \mu_j^T \Sigma^{-1} \mu_j + \log(\pi_j) + C \\
&= h_j(x) + C,
\end{aligned}$$

where C does not depend on class label k. Therefore the transformation $y = Ax + b$ does not change the classification.

On the other hand, The KNN algorithm depends on a distance matrix between observations. Suppose one feature is multiplied by a factor α, which is allowed to increase indefinitely. Then the classification will eventually be dominated by that feature. Applying this transformation to a different feature may result in a distinct classification. The classifier is therefore not location-scale invariant.

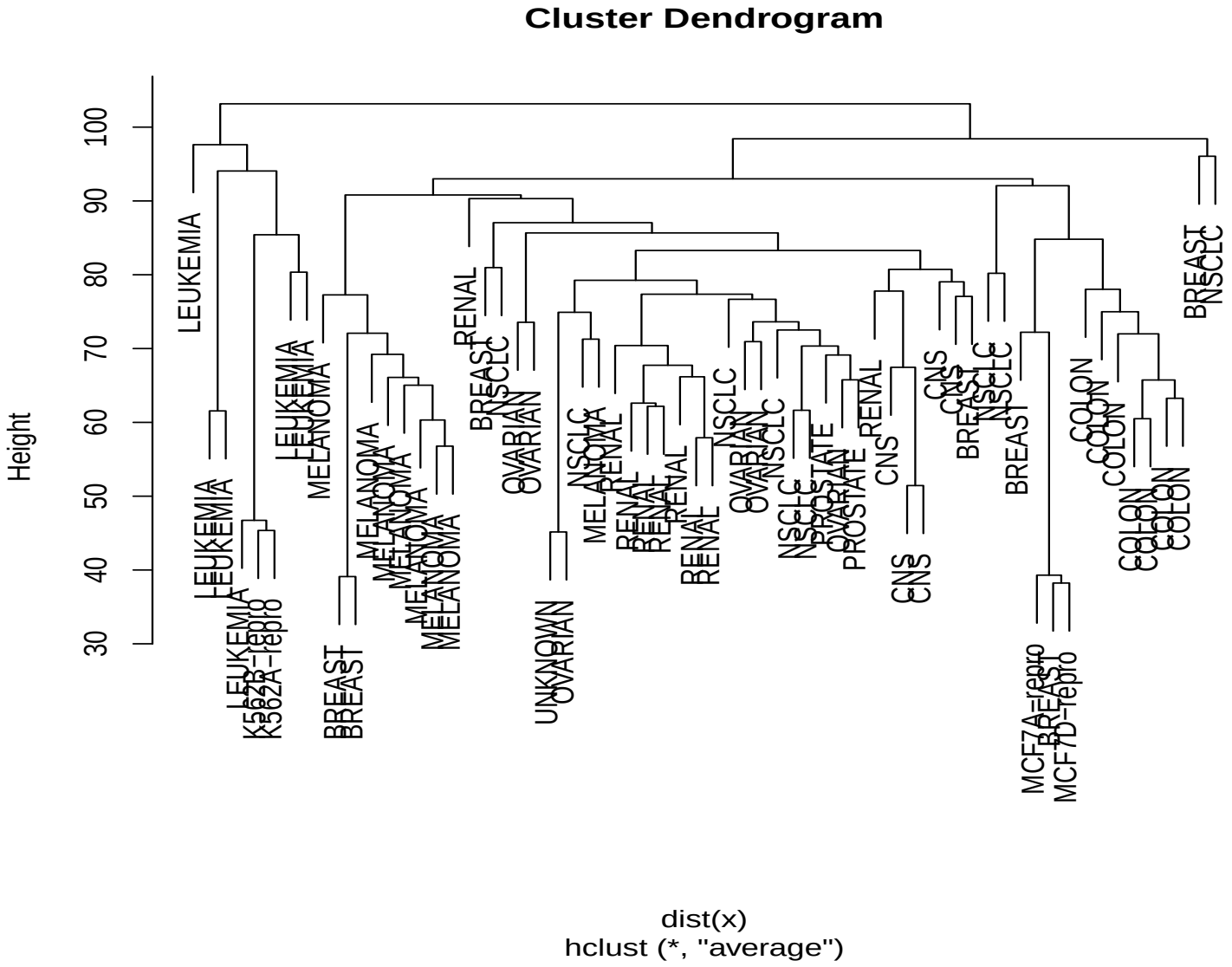

Figure 1: Figure for Q2 (a).

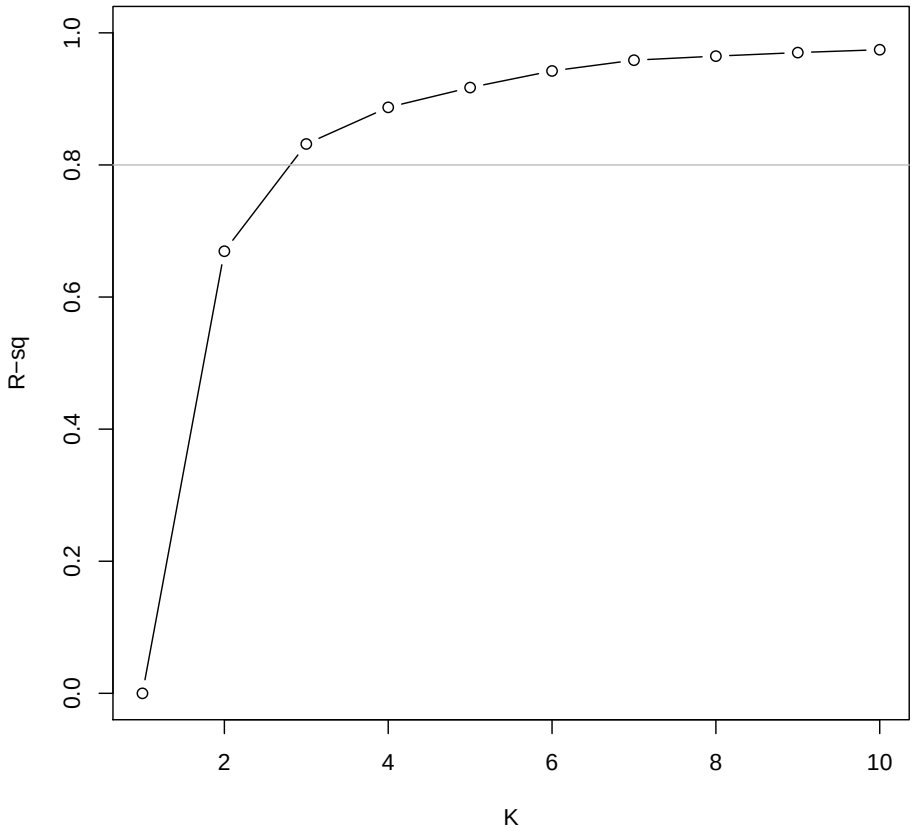

Figure 2: Figure for Q3 (a).

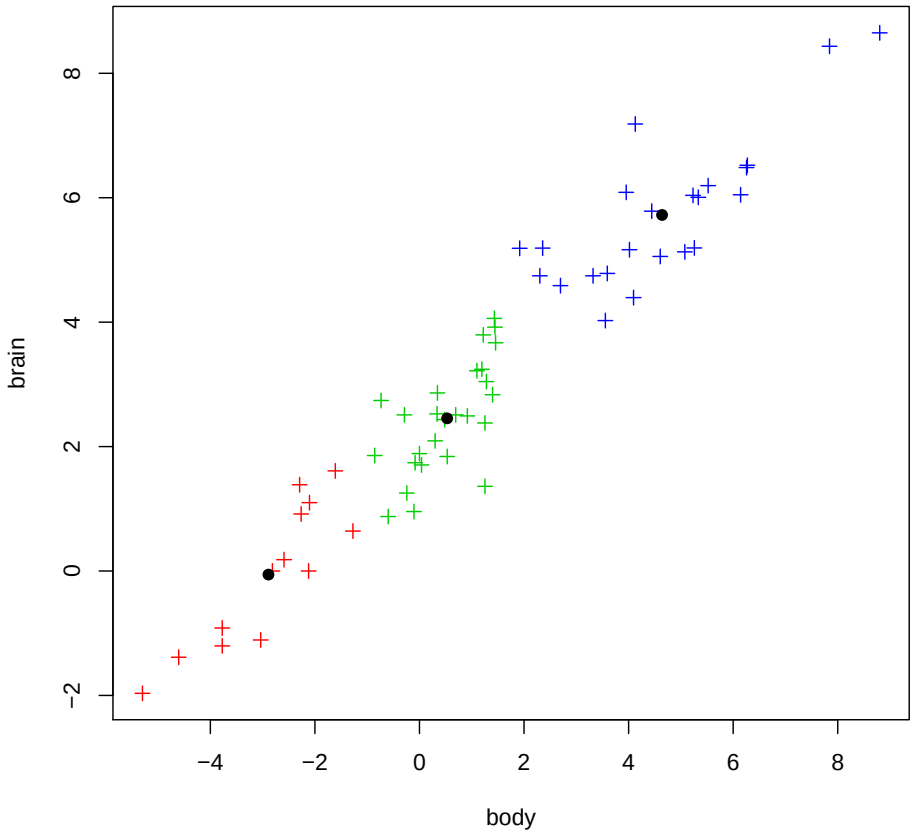

Figure 3: Figure for Q3 (b).

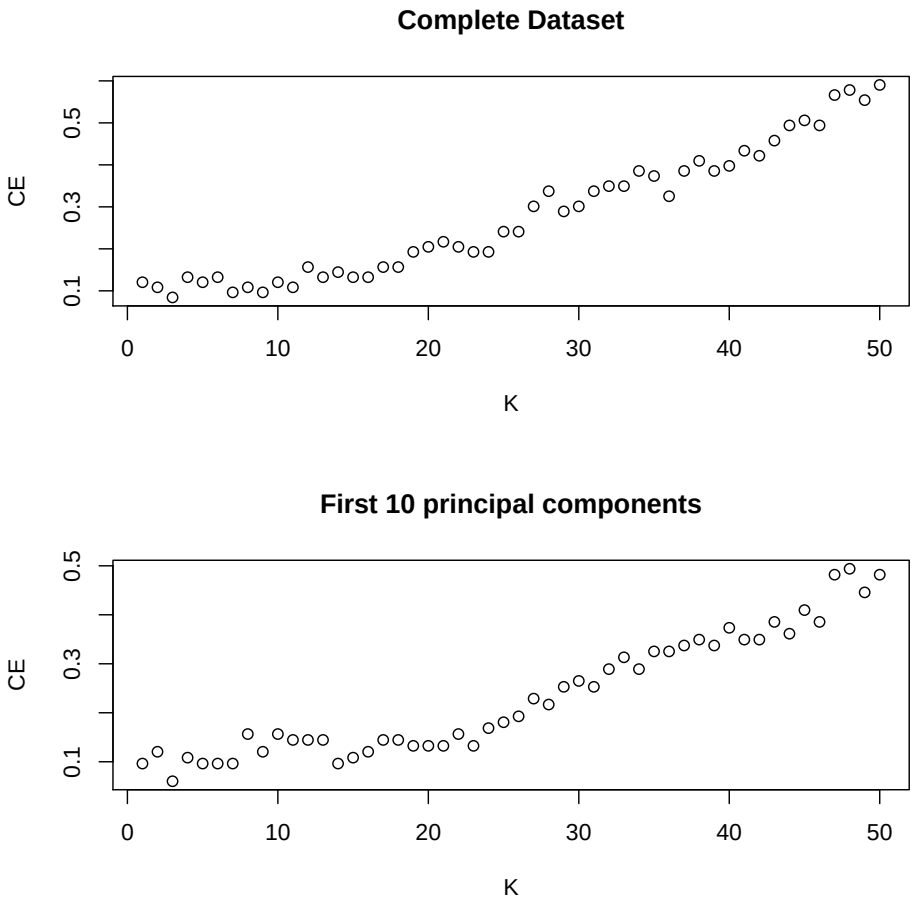

Figure 4: Figure for Q4 (b).