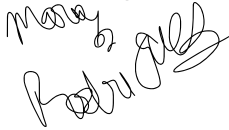


Final Project Report from Group 34

Maria Rodriguez (mar1); Anirudh Narain (anarain); Kayla Oliva (keoliva); Jinhee Lee (jinheel1)

Signatures



Unsupervised Learning

Q1a: Is there any interesting structure present in the data?

The data frame given to us contained 133 patients with 248 gene encodings. Given this high dimensionality it is extremely difficult to determine if there is any structure in the data. To solve this we explore several different dimension reduction methods to determine what sort of structure exists in the data.

The first method explored is Principal Component Analysis (PCA). PCA is a technique used to emphasize variation and highlight strong patterns in data. This also makes it easy to explore and visualize the data.

The two methods that apply PCA in R are `princomp()` and `prcomp()`. `princomp()` cannot be used on X because it uses the spectral decomposition approach. This would factorize X into a canonical form, where it is represented by its eigenvalues and eigenvectors. Only diagonalizable matrices can be factored in this way - X is not a diagonalizable matrix.

Thus we apply `prcomp()` to our data because it uses singular value decomposition. This examines the covariance and correlations between our observed patients and doesn't require X to be a diagonalizable matrix. But before this we explore it for a bit.

Load Data, Explore type, Shape and NAs

```
## [1] 133 248
```

```
## [1] 0
```

```
## [1] 0
```

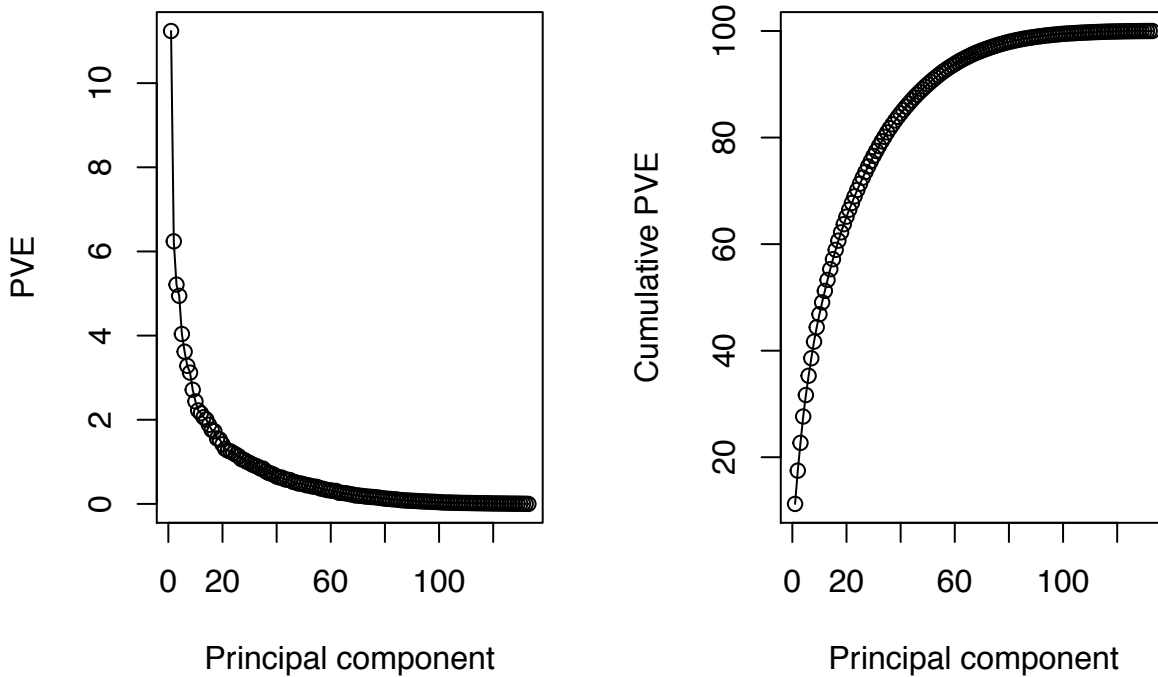
X1	X2	X3	X4	X5
186.2	-6.0	2497.9	57.6	34.5
36.0	0.2	203.8	142.3	-11.7
545.0	-3.0	273.7	172.1	-21.0
715.9	3.2	182.7	72.1	-8.3
96.0	9.6	458.4	178.8	-4.7
121.7	-2.3	1476.6	372.9	-7.2

PCA: With `prcomp()` we consider the case where X is scaled. From our summary output we see that they have many different ranges, necessitating scaling, lest the magnitude of certain variables dominate associations between variables in the sample. (Source)[<https://stats.stackexchange.com/questions/268264/normalizing-all-the-variables-vs-using-scale-true-option-in-prcomp-in-r>].

```
## [1] "sdev" "rotation" "center" "scale" "x"
```

To visualize the dimensions, we use the package `factoextra` to create a `ggplot2` visualization of our PCA objects.

The screeplot of PC object shows an elbow around 12 but it doesn't explain enough variance. To determine the amount of PC's to retain we need to see the increasing gains in variance explained by each new PC. It seems that the elbow of cumulative variance explained is passed around 60. So going down to 65 PC's seems reasonable, provided it covers 95% of variance. The proportion of variance explained by the principal component directions can quantify how much structure of the data is being captured and if we can capture 95% of the data from 65 PC's, that's much better than 100% of the data being captured from 248 variables. However, for certain classification methods we compared the outputs of using both 12 and 65 PC's.

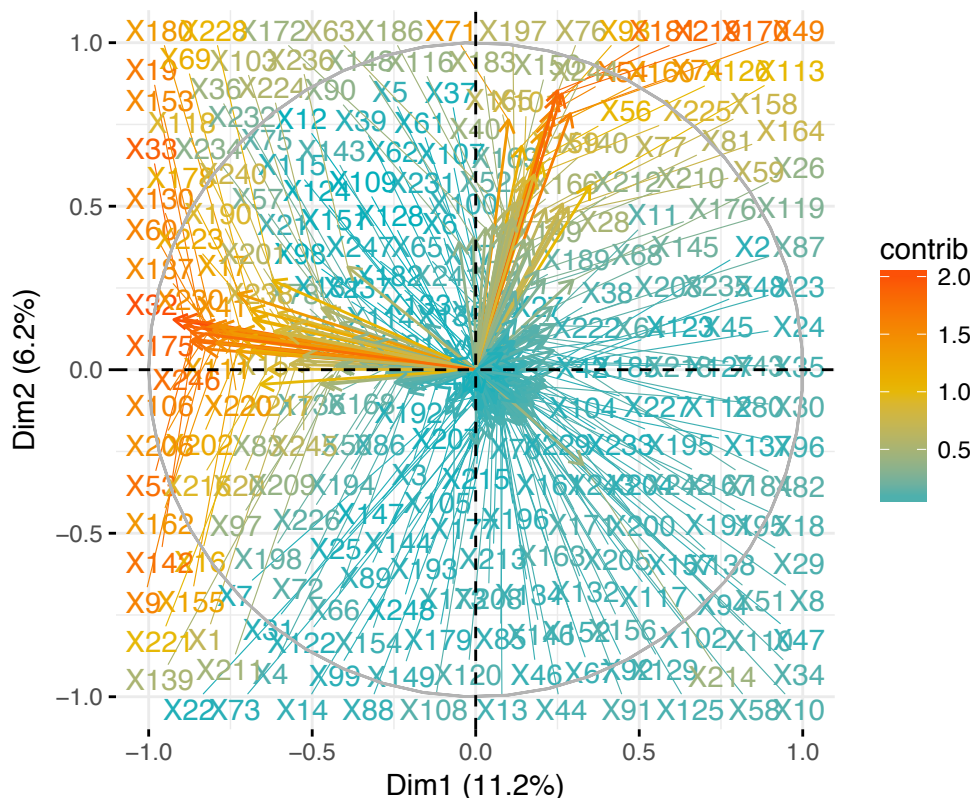


One can observe that there are two groups of variables that have similar meaning in the context of the data. The variables group on the right seem to have large positive loadings on dimension 2. On the group that is pointing left they seem to have high negative loadings on dimension 1.

There's also the variables clustered in the center lower right half. These are pointing away from the two extreme clusters, which means they have low positives loadings on dimension 1 and dimension 2.

Looking at the contributions of variables it seems that there's three different variable behaviors, those positive in dim2, negative on dim1, and all others.

Variables – PCA



For the patients themselves it seems there are two different groups, one highly positive on dimension 2 and one highly negative on dimension 1 and these groups have high contributions to the data. Most other patients lie in the center of the two dimensions.

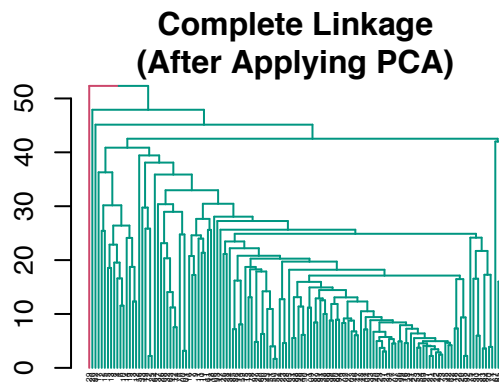
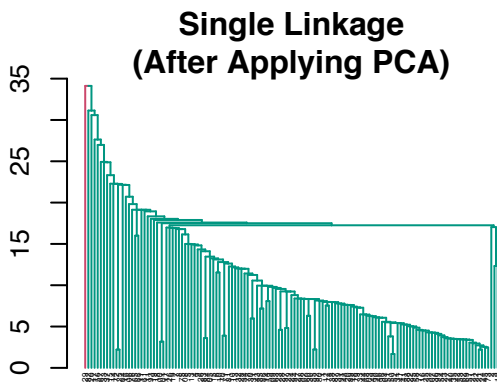
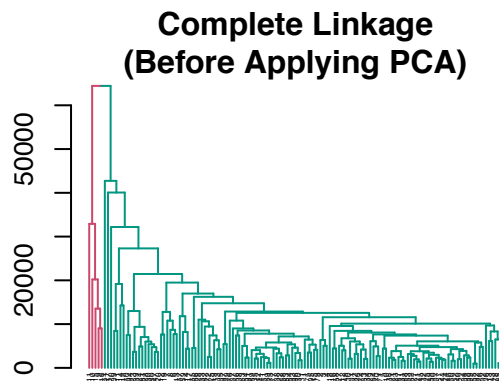
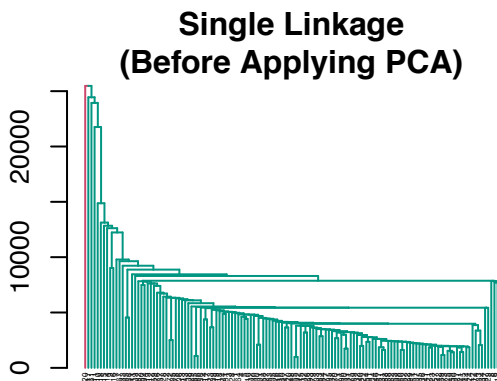
So settling on 65 PC's, we apply our clustering methods to see which works best on our data.

Clustering Methods Used

Hierarchical Clustering

One of the methods we used for clustering was hierarchical clustering. We tried four different types of linkage, and observed that the complete and average linkage had the lowest hamming clustering errors. The classification errors are presented below, including before and after applying normalized PCA to the training data. Although, single linkage suffers from chaining, complete linkage suffers from crowding, and average linkage strikes a balance so that clusters are simultaneously relatively compact and far away, the clustering results of complete and average linkage ended up doing better than single and centroid linkages' results. Since, average linkage results in more ideal clusters, we will choose the result of average linkage hierarchical clustering, to compare against the rest of the unsupervised learning models.

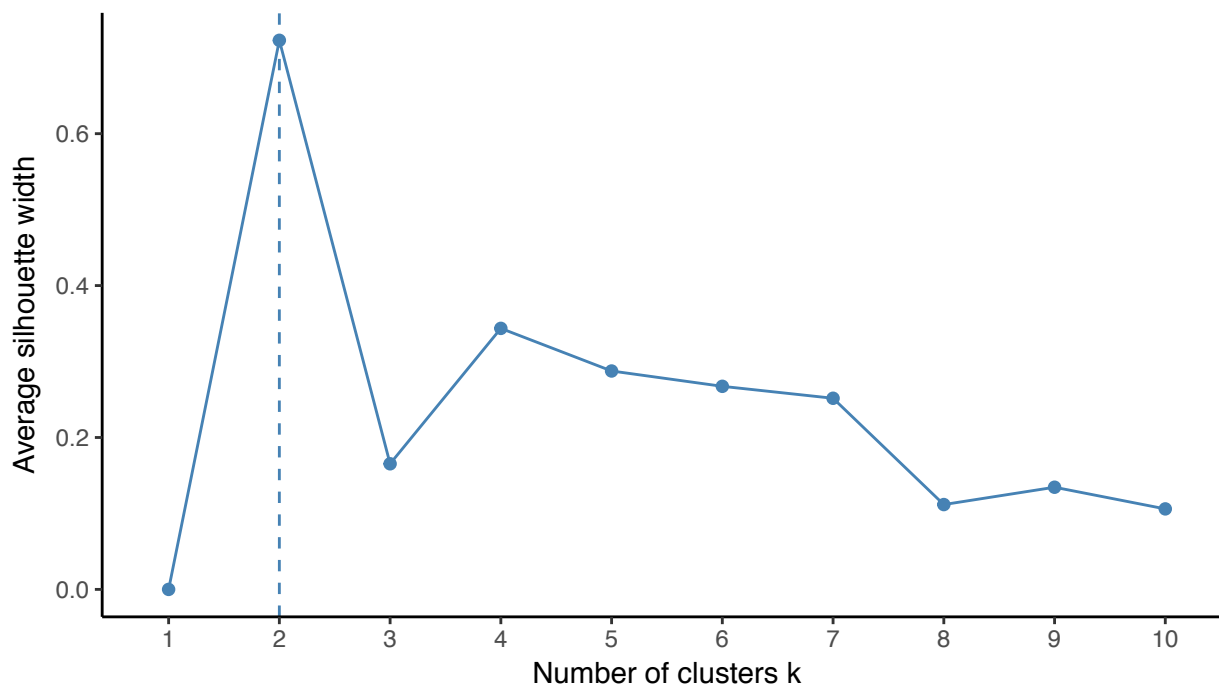
	Single	Complete	Average	Centroid
Before Applying PCA	0.1729	0.1278	0.1278	0.1579
After Applying PCA	0.1729	0.1729	0.1729	0.1729



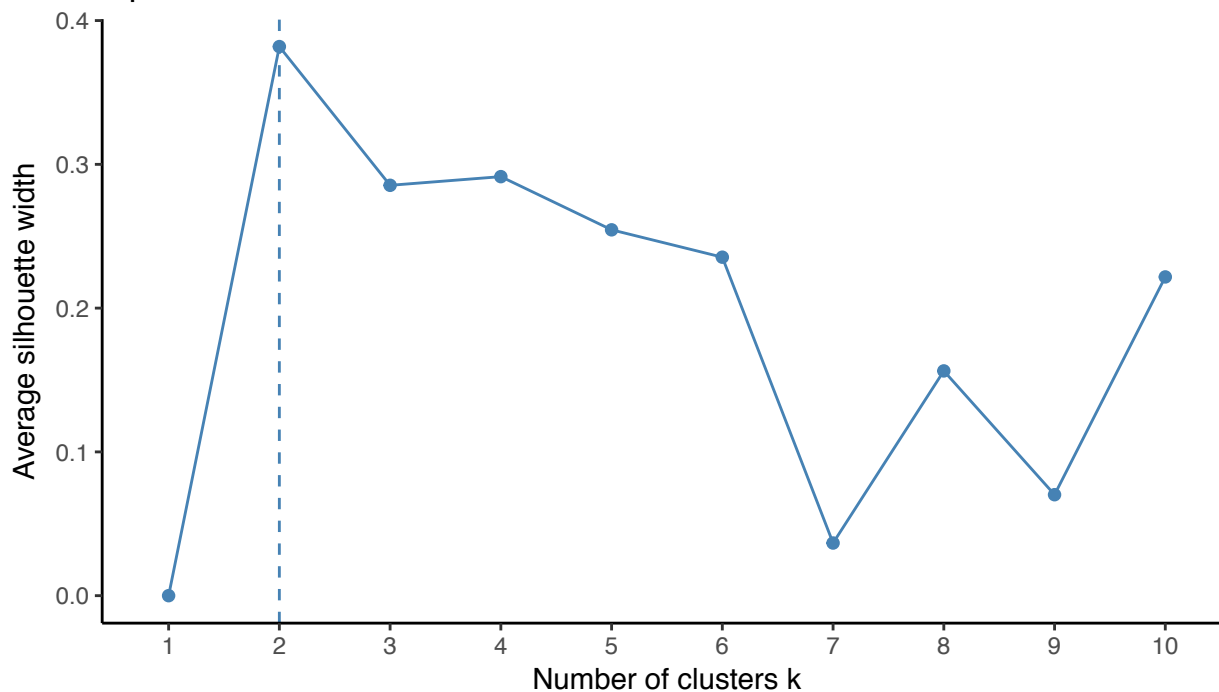
K-Means

An unsupervised learning method we apply is K-Means to determine cluster groupings in our data. But before we can determine the final cluster's it's helpful to determine the best K.

Optimal number of clusters



Optimal number of clusters

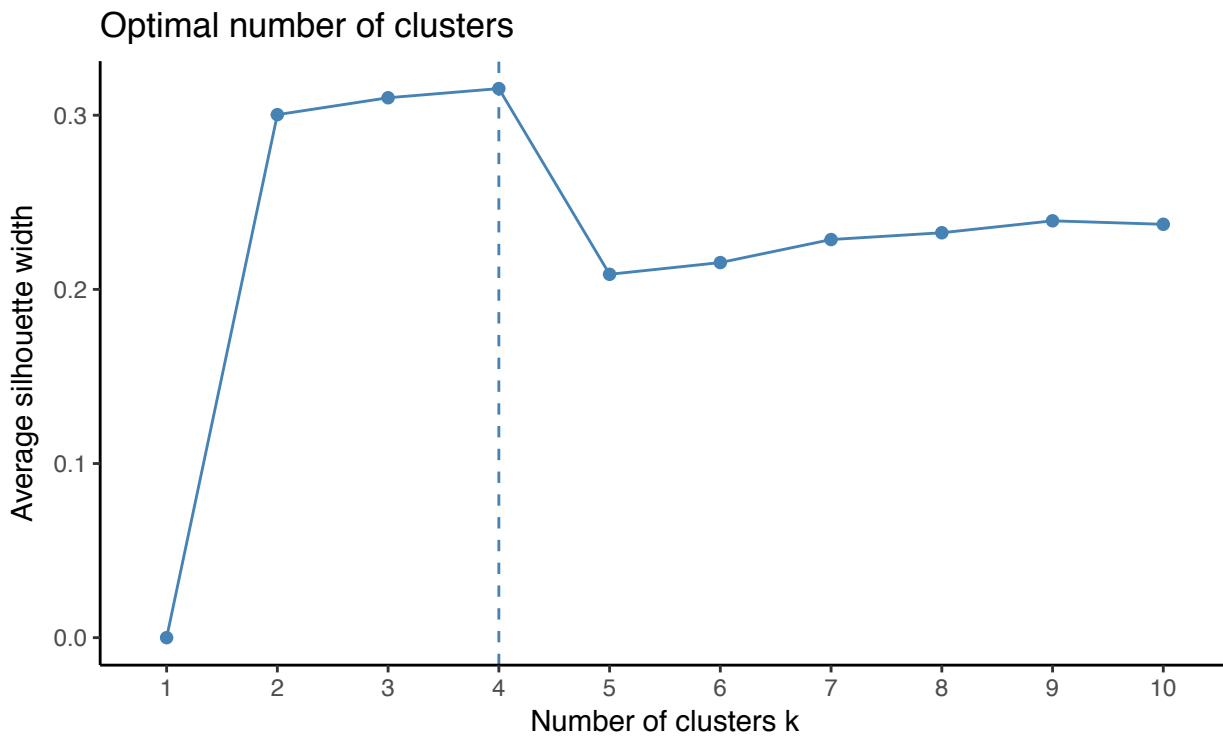
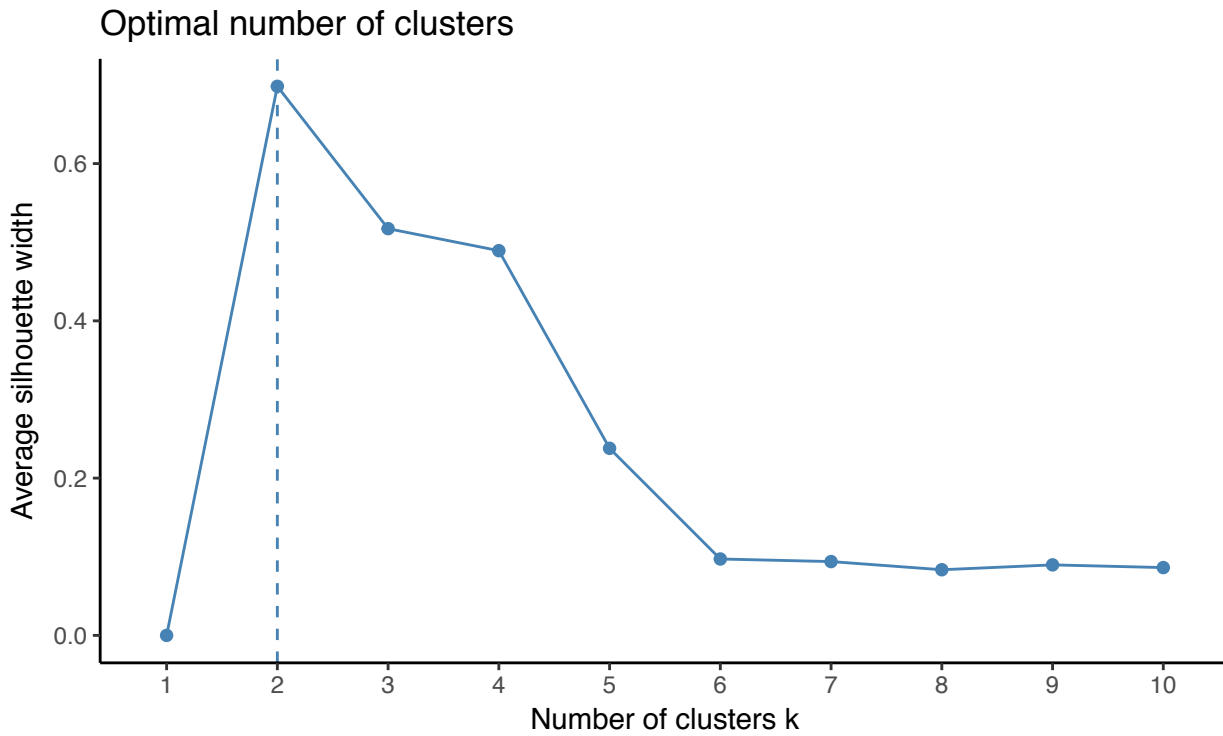


```
## [1] 0.1203008
```

```
## [1] 0.03759398
```

K-Medoids

Another method we used for clustering was K-medoids clustering. Below, the code calculates the optimal number of clusters to be 2, clusters the data using the `pam()` function, and visualizes the data. We used a custom function `best_cluster()` and calculated the error in the clustering to be about 10.5%. Although more computationally difficult, k-medoids clustering...



Error of Full Data: 10.5263 %

Error of PCA with k = 2: 10.5263 %

Error of PCA with k = 4: 13.5338 %

Supervised Learning

For this problem, we have the class labels associated with the training data available to us, and we attempt to find a supervised learning approach that yields the best results (i.e. has the lowest Hamming classification error). The approaches that we chose to investigate were linear discriminant analysis (LDA), logistic regression, k-nearest neighbors (k-NN), and decision trees. For many of the methods we used, we applied the method to both the original full data, as well as a reduced dimension matrix obtained by applied PCA. The

number of principle components used were 65 (as this were how many components were needed to achieve 95% proportion of variance explained [PVE]), and 12 (as this was the number of components after which we found an 'elbow point' when looking at change in PVE, even though 12 components only account for 51% of PVE).

Methods Used

LDA

Linear discriminant analysis is an effective way to determine a decision boundary between 2 classes. One assumption of LDA is that each class is normally distributed - this is something we cannot confirm for the original data, but for the principal components, they should be normalized because of the fact that we scaled and centered them. LDA was fit using the `lda()` function, and testing errors were computed using the following method: Separate the data into a test set (of 27 samples) and a training set (of 106 samples), compute the Hamming classification error after applying the fit, repeat 50 times (randomly selecting different indices to comprise the test set) and compute the mean test error. This testing method was applied to the original data, and the two forms of reduced data after applying PCA. The mean test errors obtained were: 9.1% (for full data), 8.3% (for 65 principal components), and 2.7% (for 12 principal components).

```
## [1] 0.07703704 0.07629630 0.01777778
```

Logistic Regression

Logistic regression allows us to fit a model that returns the probability of belonging to each class. An advantage that logistic regression has over LDA is that it assumes nothing about the normality of the data in each class. Logistic regression was fit using `glm()`, with `class = binomial`, to each of a) the full data, b) reduced dimension (PC = 65) data, and c) reduced dimension (PC = 12) data. Again, Hamming classification errors were computed by averaging the test error over 50 repetitions, like we did for LDA. The results obtained were: 38.8% (for full data), 9.8% (for 65 principal components), and 4.3% (for 12 principal components)

```
## [1] 0.38074074 0.10074074 0.03481481
```

kNN

```
##
## knn.pred  0  1
##           0 28  3
##           1  0  2
## [1] 0.09090909
```

Trees

```
##
## Call:
## randomForest(formula = as.factor(label) ~ ., data = data, mtry = 20, importance = T)
##           Type of random forest: classification
##           Number of trees: 500
## No. of variables tried at each split: 20
##
##           OOB estimate of  error rate: 1.5%
## Confusion matrix:
##      0  1 class.error
## 0 111  0 0.00000000
## 1  2 20 0.09090909
##
## Call:
## randomForest(formula = as.factor(label) ~ ., data = train.knn, mtry = 20, importance = T)
##           Type of random forest: classification
##           Number of trees: 500
## No. of variables tried at each split: 20
##
##           OOB estimate of  error rate: 2%
```

```
## Confusion matrix:
##      0  1 class.error
## 0 86  0    0.0000000
## 1  2 12    0.1428571

##
## rf.pred  0  1
##          0 25  1
##          1  0  7
```