STA 546 - Homework 2

Abbas Rizvi

March 9, 2016

Problem 1

Principal component analysis (PCA) is an exploratory technique that is useful when examining patterns in high-dimensional datasets. PCA uses an orthogonal transformation to convert a set of observations into mutually uncorrelated, ordered variables known as principal components (PC). PCA transforms the data such that the first principal component (`PC1`) captures the largest variance possible. The successive prinicpal components have the highest variance possible under the condition that the component is orthogonal to the preceding components. Generally, if the sample variances of the first few PCs are large, and the remaining set of PCs are small, the small PCs can be viewed as having low variance relative to other variables, and can be omitted form the analysis.

PCA was implemented on the `SwissBankNotes` dataset. The `SwissBankNotes` dataset was loaded into the `R` global environment.

```
load("SwissBankNotes.rdata")
```

`SwissDataNotes` consists of six variables of measurements conducted on old Swiss 1,000-franc bank notes, with the first 100 rows consisting of genuine notes and the second 100 are counterfeit. The measurement variables can be seen here:

```
colnames(SwissBankNotes)
```

```
## [1] "length"       "height.left"  "height.right" "inner.lower"
## [5] "inner.upper"  "diagonal"
```

```
SwissBankNotes <- cbind(SwissBankNotes,
                        as.factor(c(rep("genuine",100), rep("counterfeit", 100))))
colnames(SwissBankNotes)[7] <- "class"
```

The dataset was stratified into three different groups: (1) the 100 genuine notes only, (2) the 100 counterfeit notes only, and (3) the complete dataset. `stats::prcomp` was used to implement PCA across groups and return them as a `prcomp` object. Typically, the data is centered for PCA, such that we let x_i be a column in `SwissBankNotes` and we subtract by the `colMeans`, $x_i^* = x_i - \bar{x}$, where x_i^* is the centered variable. Colloquially, this means that the column-wise empirical mean is equal to zero. This is necessary for PCA to ensure that PC1 characterizes the maximum variance.

We conduct PCA using singular value decomposition of the centered covariance matrix (`prcomp(x, scale = FALSE)`) and the correlation matrix (`prcomp(x, scale = TRUE)`). The decision to implement scaling is subjective and may be useful to explore both options depending on the data at hand. For this exercise we will only used scaled data. PCA is quite sensitive to scaling of variables. PCA is less arbitrary when scaled by normalization (via using a correlation matrix or normalizing by mean-squared error (MSE)).

We investigated the proportion of variance captured by PCA by summary statistics or visually through screeplots and/or biplots. Explained proportion of variance can also be obtained dividing the eigenvalues of the correlation matrix by the sum of all the eigenvalues of the correlation matrix.

`stats::screeplot` plots the variances against ordered PCs from an object of class `prcomp`. The `biplot` is an enhanced scatterplot that uses both points and vectors to represent a structure. The biplots shown in this exercise will only be `PC1` vs. `PC2`.

```
genuine <- prcomp(SwissBankNotes[SwissBankNotes$class == 'genuine',-7], scale=TRUE)
counterfeit <- prcomp(SwissBankNotes[SwissBankNotes$class == 'counterfeit',-7], scale=TRUE)
complete <- prcomp(SwissBankNotes[,-7], scale=TRUE)

summary(genuine)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6
## Standard deviation     1.4845 1.3026 0.9827 0.76348 0.57156 0.47340
## Proportion of Variance 0.3673 0.2828 0.1610 0.09715 0.05445 0.03735
## Cumulative Proportion  0.3673 0.6501 0.8111 0.90820 0.96265 1.00000
```

```
summary(counterfeit)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6
## Standard deviation     1.3915 1.3285 0.9941 0.8823 0.56755 0.45840
## Proportion of Variance 0.3227 0.2941 0.1647 0.1297 0.05368 0.03502
## Cumulative Proportion  0.3227 0.6169 0.7816 0.9113 0.96498 1.00000
```

```
summary(complete)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6
## Standard deviation     1.7163 1.1305 0.9322 0.67065 0.51834 0.43460
## Proportion of Variance 0.4909 0.2130 0.1448 0.07496 0.04478 0.03148
## Cumulative Proportion  0.4909 0.7039 0.8488 0.92374 0.96852 1.00000
```

Looking at the summary statistics, we can see that all three groups show that `PC1` has the greatest proportion of variance, followed by `PC2`, and so on. This finding supports our convenient property that maximal amount of variance is being captured in the preceding PC.

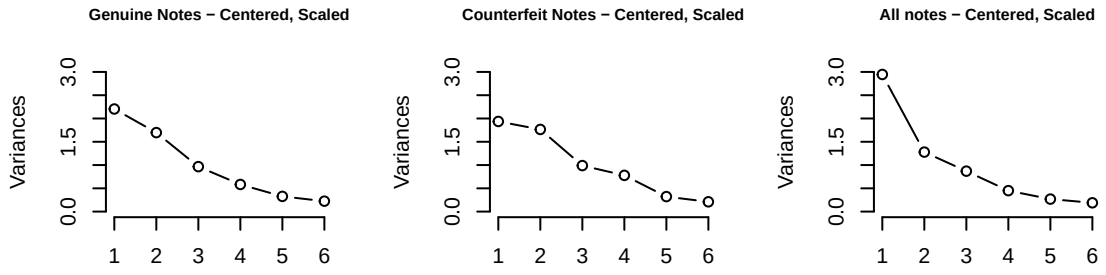

Figure 1: Screeplots of SwissBankNote Groups

When intepreting a screeplot, if two data points have a noticeable separation (decrease in value along y-axis) in the succeeding data point along the x-axis, we can call this an "elbow". An ideal "elbow" can be seen in the PCA of the complete datset (Figure 1, right panel), where a visible degree of separation between the `PC1` and `PC2` is present. The genuine (Figure 1, left panel) and counterfeit (Figure 1, middle panel) groups alone, do not have profound elbows. This could indicate that we may want to investigate the unscaled data.

We use biplots to explore samples and variables of a data matrix graphically. The axes of biplots are a pair of PCs. The biplot uses points to represent the observations on the PC. Observations with similar variance patterns will cluster in the same spatial area and have similar scores on the PC. Differing clusters along these

principal components can be intrepreted as having differing variance patterns. Biplots also use vectors to illustrate the spatial relationship that the variables have with the points. In other words, the vectors point away from the origin in some direction and length and are labeled by a variable that it is most represented by.

The helper function `ggbiplot` was written to plot ggbiplot, which uses the package `ggplot2` as it's source for visualization.

```
require(devtools)
require(ggbiplot)
gbiplots <- function(x, components, groups, legend.position){
        g <- ggbiplot(x, obs.scale = 1, var.scale = 1, ellipse = TRUE, circle=FALSE,
                      choice = components, groups=groups)
        g <- g + scale_color_discrete(name = '')
        g <- g + theme(legend.direction = 'horizontal', legend.position = legend.position)
        print(g)
}
```

```
gbiplots(genuine, components=c(1,2), groups=FALSE, legend.position = "none")
```

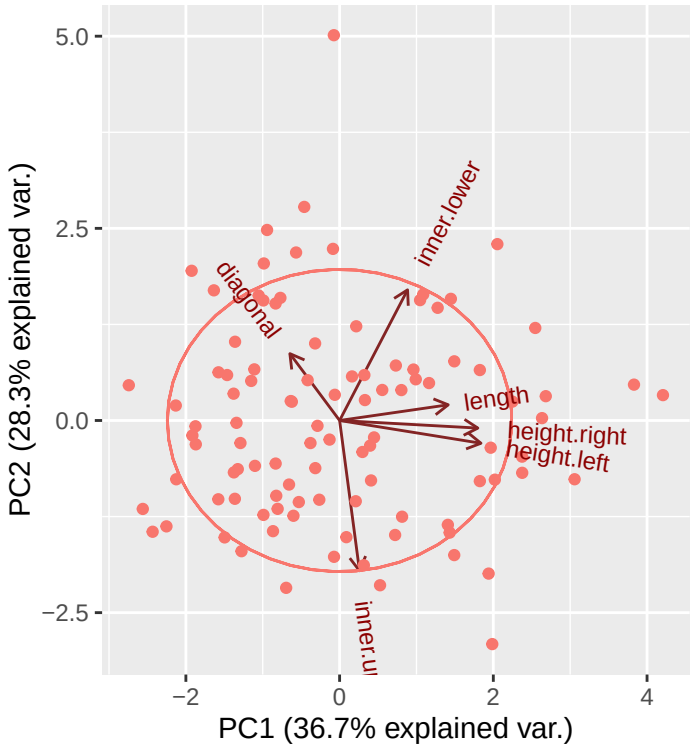

Figure 2: Genuine SwissBankNotes Biplot

```
gbiplots(counterfeit, components=c(1,2), groups=FALSE, legend.position = "none")
```

```
gbiplots(complete, components=c(1,2), groups=SwissBankNotes$class, legend.position="top")
```

The biplots of the three different groups (genuine alone, counterfeit alone, or both groups of bank notes) can be see in Figures 2-4, respectively.

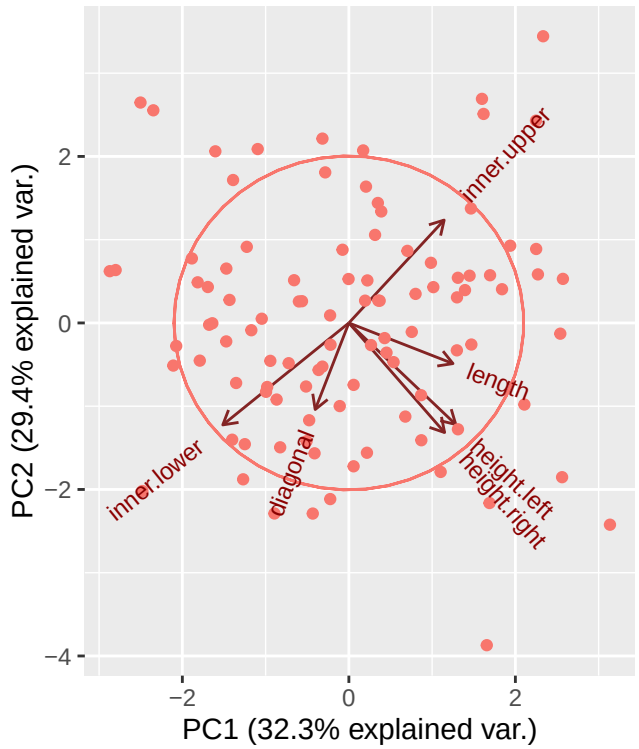

Figure 3: Counterfeit SwissBankNotes Biplot

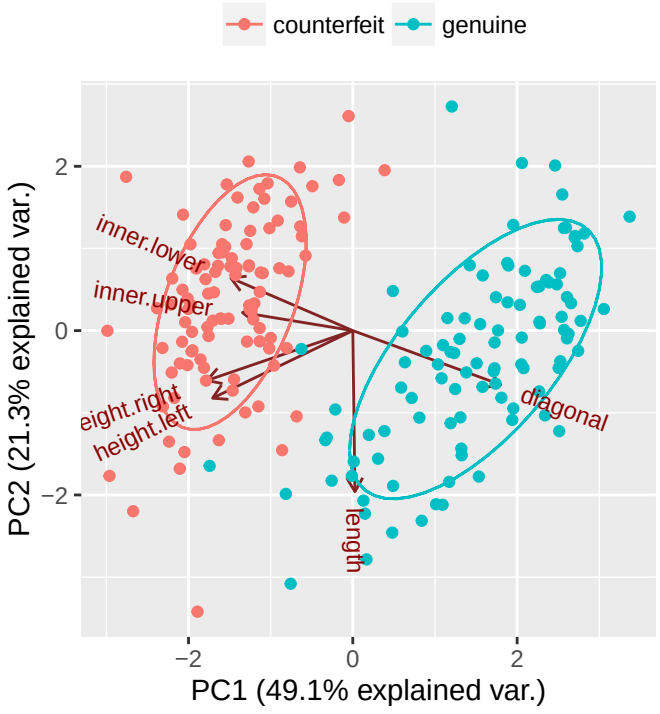

Figure 4: Complete SwissBankNotes Biplot

The first 2 PCs plotted against each other for Genuine bank notes can be seen in Figure 2. The relationship that Figure 2 summarizes is the explained variance *within* the genuine bank notes. The center of the biplot is at the origin. We can see that the vectors of `height.right`, `height.left` and `length` are mostly explained by the variation in `PC1`. While `inner.lower`, `diaganol`, and `inner.height` are mostly being explained by `PC2`.

Figure 3 shows the *within* counterfeit bank notes variance. We can see that all of the variables are well separated and pointing in different directions, with `height.left` and `height.right` have similar patterns of variance. The inference drawn from Figure 2 and 3 cannot be overanalyzed, and more analyses are needed to really understand the structure of these different patterns.

Figure 4 shows a biplot of the first two PCs of both sets of bank notes. The red group of points are counterfeit and the green group of points are genuine bank notes. The most interesting variable in Figure 5 is the `diaganol` measurement, here, we can see that the variation from the counterfeit cluster to the genuine cluster is mostly explained by `PC1`. It could be possible that this variable is the best differentiating variable between the genuine and counterfeit bank notes, but further analyses are needed.

Problem 2

2.1 Generate three classes of simulated data

Simulated data was generated from 3 random normal distributions with varying means and standard deviations. The dataset had 20 observations in each of three classes (total 60 observations) and 50 variables. The function `SimDat` was written in order to automate the generation of a random data set with the specific dimensions using `stats::rnorm`.

```
set.seed(333)
SimDat <- function(mean, sd, group){
        n <- 20 #20 observations
        m <- 50 #50 variables
        x <- matrix(rnorm(n*m,mean=mean,sd=sd),n,m)
        dimnames(x) <- list(rownames(x, do.NULL=FALSE,
                                     prefix=paste0("Obs-",deparse(substitute(group)),sep="-")),
                            colnames(x, do.NULL=FALSE, prefix="Var"))
        return(x)
}
```

Using `SimDat` we can input the mean, standard deviation, and the name that the user would like to use to differentiate the different sets of simulated data (e.g. class 1, class2, etc.)

```
rmat <- as.matrix(rbind(SimDat(15,13,class1), SimDat(0,2,class2), SimDat(30,15,class3)))
dim(rmat)
```

```
## [1] 60 50
```

`Class 1` has a mean of 15 and a standard deviation of 13. `Class 2` has a mean of 0 and a standard deviation of 2. `Class 3` has a mean of 30 and a standard deviation of 15. The 3 classes were bound together by rows so that the total dataset has 50 variables and 60 observations.

2.2 PCA on simulated data

PCA was performed on scaled and centered versions of the randomly generated data for all 60 observations.

```
groups = c(rep("class 1", 20), rep("class 2", 20), rep("class 3", 20))
rmat.pc <- prcomp(rmat, scale=TRUE)
gbiplots(rmat.pc, components=c(1,2), groups=groups,
         legend.position="top")
```

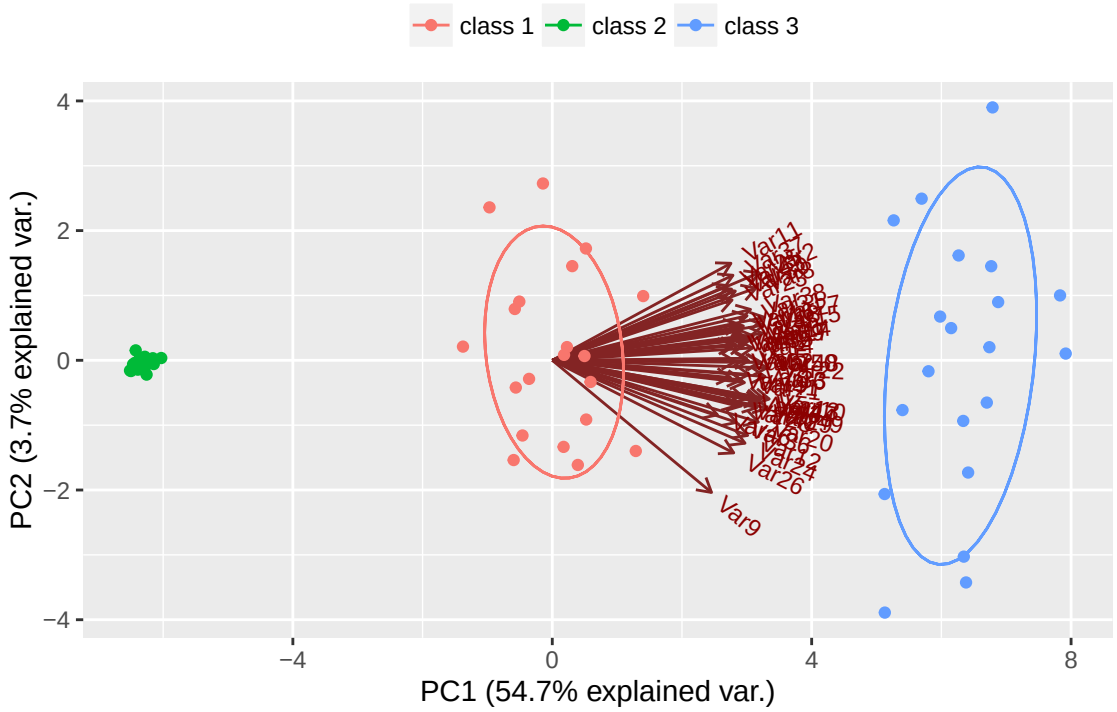

Figure 5: Biplot of 3 classes groups mean shifted normally distributed random variables

The first two principal component score vectors were plotted against one another using our helper function `gbiplot` from Problem 1. The `gibplot` plots the first 2 PC score vectors and uses a different color to differentiate between the different "classes" of observations (Figure 5). Figure 5 shows a good separation between the clusters of the randomly generated classes (class 1 red cluster, class 2 green cluster, and class 3 blue cluster). `Class 3` has the most dispersion in the first two PCs (Figure 5).

2.3 K-means clustering was conducted on the observations with a K = 3

K-means clustering algorithm is a popular iterative clustering method. The algorithm is as followed (adopted from Elements of Statistical Learning, 2e):

1. For a given cluster assignment, C, the total cluster variance is minimized with respect to $m_1, ..., m_k$ yielding the means of the currently assigned clusters.
2. Given a current set of means $m_1, ..., m_k$ is minimized by assigning each observation to the closest (current) cluster mean. That is,

$$C(i) = argmin_{1 \leq k \leq K} ||x_i - m_k||^2$$

3. Steps 1 and 2 are iterated until the assignments do not change.

Step 3 essentially means that steps 1 and 2 reduce the values such that convergence is assured. This can present some problems. The convergence may actually occurs at a local minimia along the landscape and not the desired global minima.

K-means was applied to the 60 observations with $\mathbf{K} = 3$.

```
km.k3<- kmeans(rmat[,-51], centers=3)
table(km.k3$cluster, groups)
```

```
##    groups
##     class 1 class 2 class 3
##   1      20      20       0
##   2       0       0      11
##   3       0       0       9
```

The **K**-means algorithm was able to distinguish each of the classes into separate clusters (class 3 in cluster 1, class 1 in cluster 2, and class 2 in cluster 3). Although the cluster number does not match the class, the fact that it was able to correctly identify these different classes into distinct clusters is a valid result.

2.4 K-means clustering was performed with K = 2 and K = 4

Just as in section 2.3, **K**-means clustering was applied to the dataset but this time with $\mathbf{K} = 2$ and $\mathbf{K} = 3$

```
km.k2<- kmeans(rmat[,-51], centers=2)
table(km.k2$cluster, groups)
```

```
##    groups
##     class 1 class 2 class 3
##   1      19       0      20
##   2       1      20       0
```

The table for $\mathbf{K} = 2$, shows that 19 observations of `class 1` were in cluster 1 and one observation was in cluster 2. `class 2` correctly clustered together, however, again, with 1 observation `class 1`. And lastly, all of `class 3` clustered together in cluster 1 along with the 19 observations from `class 1`. Intuitively, this makes sense, considering that mean and standard deviation of the random normally distributed variables in their (`class 1` and `class`) respective data sets are more similar than `class 2`. So it can be expected that the distance between `class 1` and `class 3` is less than the distance between either class comparatively to `class 2`.

```
km.k4 <- kmeans(rmat[,-51], centers=4)
table(km.k4$cluster, groups)
```

```
##    groups
##     class 1 class 2 class 3
##   1      20       0       0
##   2       0       9       0
##   3       0      11       0
##   4       0       0      20
```

The results for $\mathbf{K} = 4$ show that `class 1` and `class 2` clustered in their own clusters. `class 3` was split into cluster 3 and 4. This may be due to the large mean and variance of `class 3`. The **K**-means algorithm may have classifieds the distance due to the large variance.

2.5 K-means clustering was performed with K = 3 on first two prinicpal component score vectors

```
km.pc <- kmeans(as.matrix(rmat.pc$x[,1:2]), centers=3)
table(km.pc$cluster, groups)
```

```
##    groups
##     class 1 class 2 class 3
##   1      20       0       0
##   2       0       1      20
##   3       0      19       0
```

The results reveal that when **K** = 3 was applied to the first two PC score vectors, the classes all clustered together into their own respective clusters. This result appears to be more accurate than just conducting the **K**-means on the raw data. The first PC capture most of the variance, this amount of variance must be sufficient for the **K**-means algorithm to capture the distances from each class into their own clusters.

2.6 K-means clustering with K = 3 and variables normalized to standard deviation of 1

The `stats::scale` function was applied on the data such that each variable had a standard deviation of 1. Subsequently **K**-means clustering with **K** = 3 was performed on the scaled data.

```
rmat.sc <- scale(rmat, center = FALSE, scale = apply(rmat, 2, sd))
km.sc <- kmeans(rmat.sc, centers=3)
table(km.sc$cluster, groups)
```

```
##    groups
##     class 1 class 2 class 3
##   1       0       0      20
##   2      20       1       0
##   3       0      19       0
```

The results for 2.6 agree with the results from 2.2. In Figure 5, we can see that from left to right, the classification of clusters goes from `class 2` to `class 1` to `class 3`. Again the results from 2.6 show us that `class 1` is cluster 2, `class 2` is cluster 1, and `class 3` is cluster 3. All of the classes were clustered into unique clusters with each of their respective data points in `PC1` and `PC2`.

Problem 3

A gene expression data set was adopted from ISLR Ch10 Question 11. The dataset consists of 40 tissue samples with measurements on 1,000 genes. The first 20 samples are healthy patients and the second 20 are from a diseased group.

3.1 Load the data in using `read.csv(data=Ch10Ex11.csv, header=F)`

```r
dat <- read.csv("Ch10Ex11.csv", header=F)
colnames(dat)[1:20] <- paste0("healthy-", 1:20)
colnames(dat)[21:40] <- paste0("diseased-", 1:20)
```

3.2 Apply hierarchical clustering to the samples using correlation-based distance and plot the dendrogram.

In comparison to the exercises that were conducted in problem 2 – **K**-means clustering algorithms depend on the choice of **K** (cluster centers) to be searched for and an assigned starting point. Comparatively, hierarchical clustering methods do not require these pre-determined conditions. Hierarchical clustering requires that the user decides upon a measure of dissimilarity between groups of observations, based upon the pairwise dissimilarities between the two groups' observations.

The dissimilarity measure we chose was 1 - correlation of the raw data. `stats::hclust` uses an agglomeration method (bottom-up) approach for clustering. Ideally, we would like to see that the algorithm "naturally" represents different groups into different clusters. We explored different types of agglomerative methods and compared them (Figure 6). The different methods (linkage) that were chosen were: 1) complete, 2) average, and 3) single.

```r
dissimilarity <- 1-cor(dat)
dat.dist <- as.dist(dissimilarity)
par(mfrow=c(3,1))
plot(hclust(dat.dist, method="complete"),
     main="Dissimilarity = 1 - Correlation (Complete Linkage)", xlab="")
plot(hclust(dat.dist, method="average"),
     main="Dissimilarity = 1 - Correlation (Average Linkage)", xlab="")
plot(hclust(dat.dist, method="single"),
     main="Dissimilarity = 1 - Correlation (Single Linkage)", xlab="")
```

Looking at Figure 6 – it seems as though using `complete` linkage gave the best separation between the two groups. `average` linkage seems to not do as well of job distinguishing healthy from diseased individuals. However, `single` linkage does separate the two groups into different clusters, but it does not appear to be as clean as `complete` linkage.

3.3

A collaborator want to know which genes differ the most across the two groups. We can do this by doing a simple t-test or a permuted multiple hypothesis testing t-test.

```r
myttest <- function(x) t.test(x[1:20], x[21:40])$p.value
p.val <- apply(dat, 1, myttest)
sorted_p <- order(p.val)
dat.order <- dat[sorted_p, ]
DEgenes <- dat.order[1:100,]
DEgenes[1:5,1:5]
```

```
##      healthy-1   healthy-2   healthy-3  healthy-4  healthy-5
## 502 -1.2634790  0.15974770 -1.86663300 -0.8964610 -0.1393987
## 589 -0.3011929 -0.03294286 -0.82764930  0.9614763 -0.3857303
## 600  1.2134940 -0.80502150 -0.05612577 -1.4323110 -0.5125825
## 590  0.4913611 -0.61444950 -0.92325740 -0.4421924  1.3471190
## 565 -0.2453890 -0.39357150 -0.69497020  0.7439966  0.7338776
```

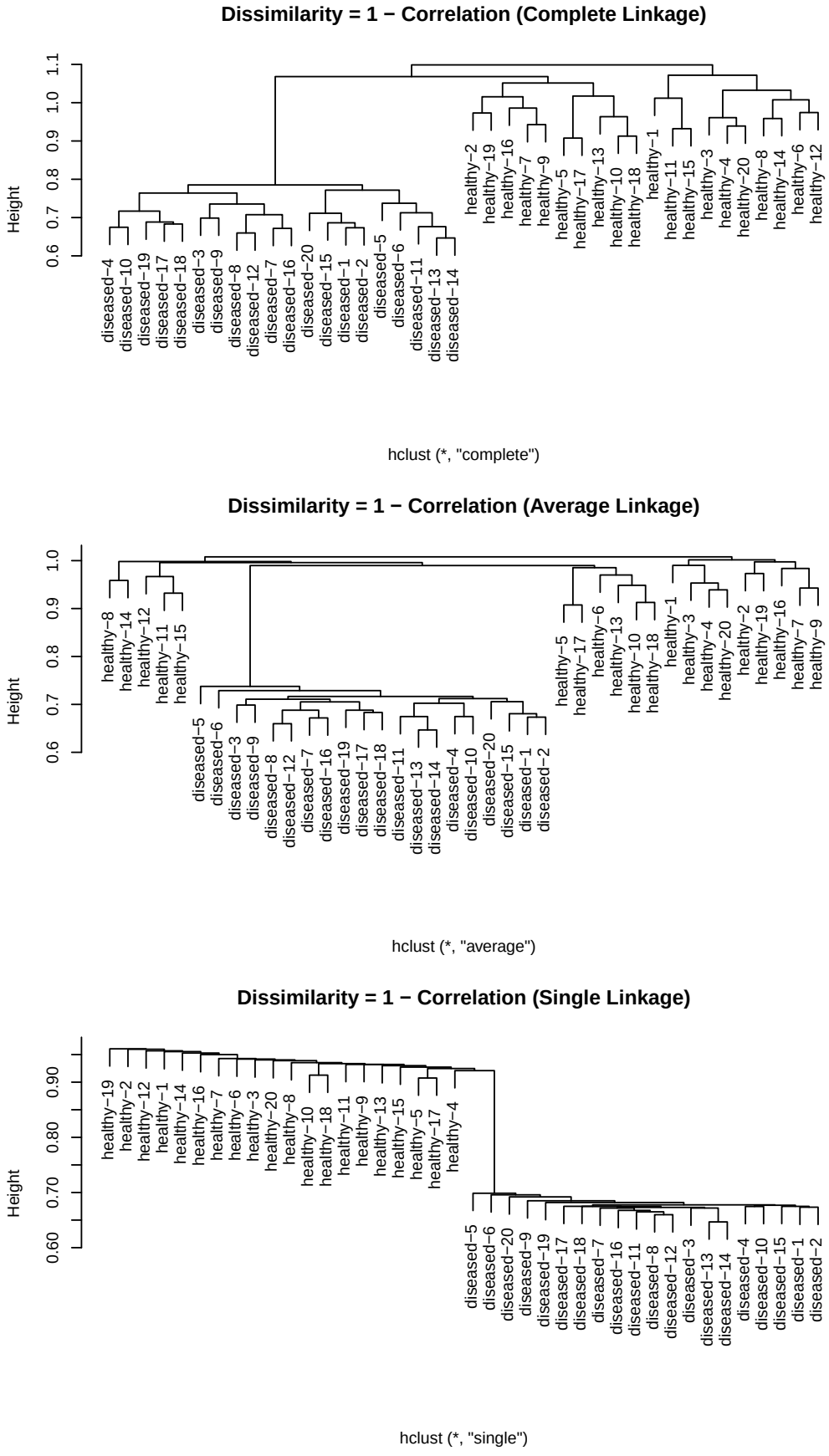

Figure 6: Hiearchical Clustering With Different Linkage Methods

`DEgenes` is ordered by lowest p-values from the t-test. Here we show just the top 5 genes (indicated by row index). We would send this list to the investigator he/she can investigate the results and gain some biological relevance insight.

Lastly, we decided to do a permuted t-test for multiple hypothesis testing on this data set.

```
require(multtest)
labels <- factor(c(rep("healthy", 20), rep("diseased", 20)))
mt.test <- mt.maxT(dat, classlabel=labels, B=10000)
```

```
## b=100    b=200   b=300   b=400   b=500   b=600   b=700   b=800   b=900   b=1000
## b=1100   b=1200  b=1300  b=1400  b=1500  b=1600  b=1700  b=1800  b=1900  b=2000
## b=2100   b=2200  b=2300  b=2400  b=2500  b=2600  b=2700  b=2800  b=2900  b=3000
## b=3100   b=3200  b=3300  b=3400  b=3500  b=3600  b=3700  b=3800  b=3900  b=4000
## b=4100   b=4200  b=4300  b=4400  b=4500  b=4600  b=4700  b=4800  b=4900  b=5000
## b=5100   b=5200  b=5300  b=5400  b=5500  b=5600  b=5700  b=5800  b=5900  b=6000
## b=6100   b=6200  b=6300  b=6400  b=6500  b=6600  b=6700  b=6800  b=6900  b=7000
## b=7100   b=7200  b=7300  b=7400  b=7500  b=7600  b=7700  b=7800  b=7900  b=8000
## b=8100   b=8200  b=8300  b=8400  b=8500  b=8600  b=8700  b=8800  b=8900  b=9000
## b=9100   b=9200  b=9300  b=9400  b=9500  b=9600  b=9700  b=9800  b=9900  b=10000
```

```
sorted_mt <- dat[mt.test$index,]
sorted_mt[1:5,1:5]
```

```
##      healthy-1   healthy-2   healthy-3  healthy-4  healthy-5
## 502 -1.2634790  0.15974770 -1.86663300 -0.8964610 -0.1393987
## 589 -0.3011929 -0.03294286 -0.82764930  0.9614763 -0.3857303
## 600  1.2134940 -0.80502150 -0.05612577 -1.4323110 -0.5125825
## 590  0.4913611 -0.61444950 -0.92325740 -0.4421924  1.3471190
## 565 -0.2453890 -0.39357150 -0.69497020  0.7439966  0.7338776
```

The multiple hypothesis testing computes an adjusted p-value, which controls family-wise type I error (FWER). These results reveal regardless of the controlling the FWER or not, the same genes differ the most across the two groups.