PH.140.644_HW3

Kunyu An

Chapter 10

Q8

In Section 10.2.3, a formula for calculating PVE was given in Equation 10.8. We also saw that the PVE can be obtained using the `sdev` output of the `prcomp()` function.

On the `USArrests` data, calculate PVE in two ways:

(a) Using the sdev output of the prcomp() function, as was done in Section 10.2.3.

```
attach(USArrests)
set.seed(1)
pr.out = prcomp(USArrests, scale=TRUE)
pr.var = pr.out$sdev^2
pve_1 = pr.var / sum(pr.var)
pve_1
```

```
## [1] 0.62006039 0.24744129 0.08914080 0.04335752
```

(b) By applying Equation 10.8 directly. That is, use the prcomp() function to compute the principal component loadings. Then, use those loadings in Equation 10.8 to obtain the PVE.

```
USArrests_scaled = scale( USArrests )
denom = sum( apply( USArrests_scaled^2, 2, sum ) )

Phi = pr.out$rotation
USArrests_projected = USArrests_scaled %*% Phi

numer = apply( pr.out$x^2, 2, sum )
pve_2 = numer / denom
pve_2
```

```
##        PC1        PC2        PC3        PC4
## 0.62006039 0.24744129 0.08914080 0.04335752
```

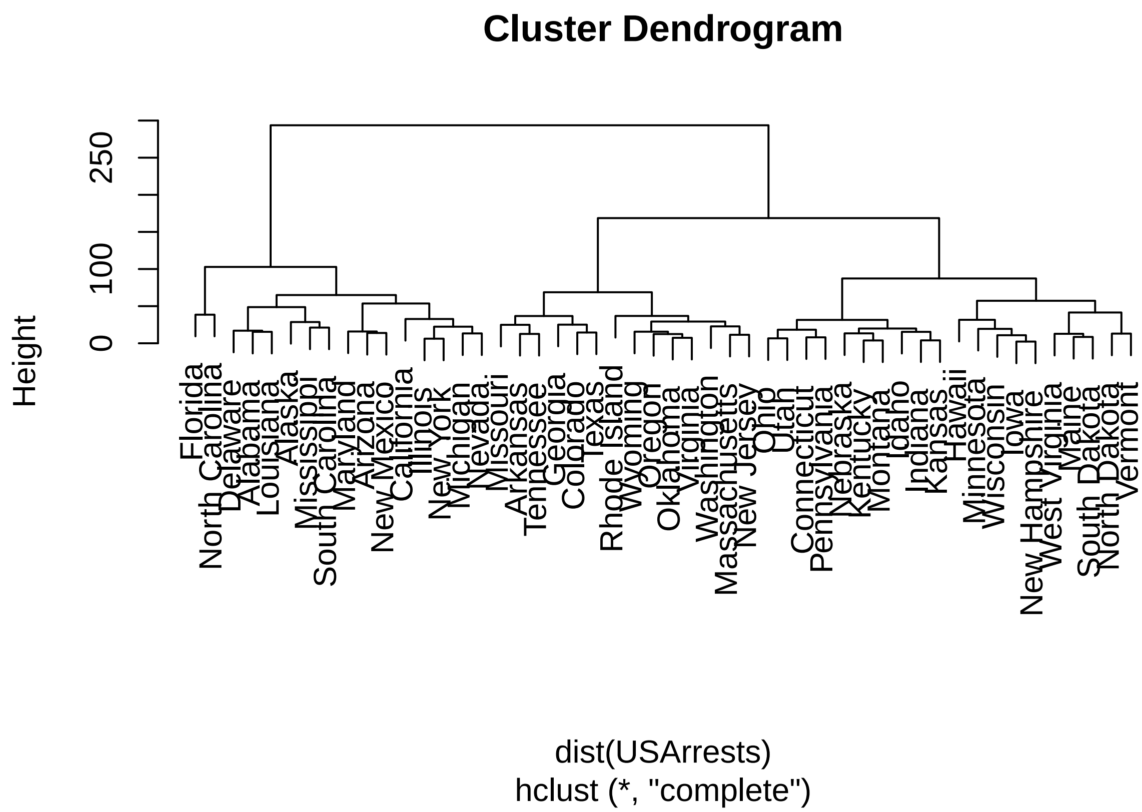
As shown above, the result from (a) and (b) are the same.

Q9

Consider the `USArrests` data. We will now perform hierarchical clustering on the states.

(a) Using hierarchical clustering with complete linkage and Euclidean distance, cluster the states.

```r
set.seed(1)
hclust.complete = hclust( dist(USArrests), method="complete" )
plot(hclust.complete)
```



(b) Cut the dendrogram at a height that results in three distinct clusters. Which states belong to which clusters?

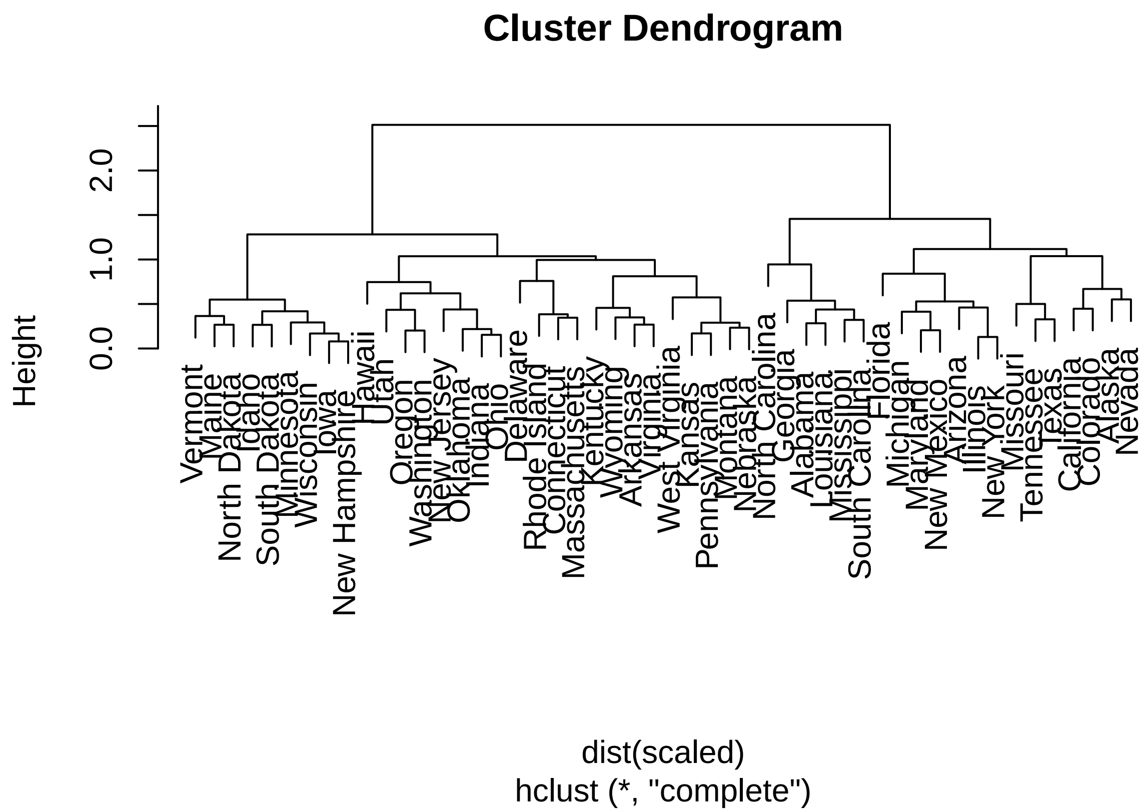
```r
ct = cutree( hclust.complete, 3 )
# Print which states go into each cluster
for( k in 1:3 ){
  print(k)
  print( rownames( USArrests )[ ct == k ] )
}
```

```
## [1] 1
##  [1] "Alabama"        "Alaska"         "Arizona"        "California"
##  [5] "Delaware"       "Florida"        "Illinois"       "Louisiana"
##  [9] "Maryland"       "Michigan"       "Mississippi"    "Nevada"
## [13] "New Mexico"     "New York"       "North Carolina" "South Carolina"
## [1] 2
##  [1] "Arkansas"      "Colorado"      "Georgia"       "Massachusetts"
##  [5] "Missouri"      "New Jersey"    "Oklahoma"      "Oregon"
##  [9] "Rhode Island"  "Tennessee"     "Texas"         "Virginia"
## [13] "Washington"    "Wyoming"
## [1] 3
```

```
##  [1] "Connecticut"   "Hawaii"        "Idaho"         "Indiana"
##  [5] "Iowa"          "Kansas"        "Kentucky"      "Maine"
##  [9] "Minnesota"     "Montana"       "Nebraska"      "New Hampshire"
## [13] "North Dakota"  "Ohio"          "Pennsylvania"  "South Dakota"
## [17] "Utah"          "Vermont"       "West Virginia" "Wisconsin"
```

(c) Hierarchically cluster the states using complete linkage and Euclidean distance, after scaling the variables to have standard deviation one.

```
scaled = scale(USArrests, center = FALSE)
hclust.complete.scale = hclust( dist(scaled), method="complete" )
plot(hclust.complete.scale)
```



(d) What effect does scaling the variables have on the hierarchical clustering obtained? In your opinion, should the variables be scaled before the inter-observation dissimilarities are computed? Provide a justification for your answer.

```
ct = cutree( hclust.complete.scale, k=3 )
# Print which states go into each cluster
for( k in 1:3 ){
  print(k)
  print( rownames( USArrests )[ ct == k ] )
}
```

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

```
## [1] "Alabama"        "Georgia"        "Louisiana"      "Mississippi"
## [5] "North Carolina" "South Carolina"
## [1] 2
##  [1] "Alaska"     "Arizona"    "California" "Colorado"   "Florida"
##  [6] "Illinois"   "Maryland"   "Michigan"   "Missouri"   "Nevada"
## [11] "New Mexico" "New York"   "Tennessee"  "Texas"
## [1] 3
##  [1] "Arkansas"      "Connecticut"   "Delaware"      "Hawaii"
##  [5] "Idaho"         "Indiana"       "Iowa"          "Kansas"
##  [9] "Kentucky"      "Maine"         "Massachusetts" "Minnesota"
## [13] "Montana"       "Nebraska"      "New Hampshire" "New Jersey"
## [17] "North Dakota"  "Ohio"          "Oklahoma"      "Oregon"
## [21] "Pennsylvania"  "Rhode Island"  "South Dakota"  "Utah"
## [25] "Vermont"       "Virginia"      "Washington"    "West Virginia"
## [29] "Wisconsin"     "Wyoming"
```

Scaling the variables will affect the clusters. We should scale the variables since units of measure are very different.

Q11

On the book website, www.StatLearning.com, there is a gene expression data set (`Ch10Ex11.csv`) that consists of 40 tissue samples with measurements on 1,000 genes. The first 20 samples are from healthy patients, while the second 20 are from a diseased group.

(a) Load in the data using `read.csv()`. You will need to select `header=F`.

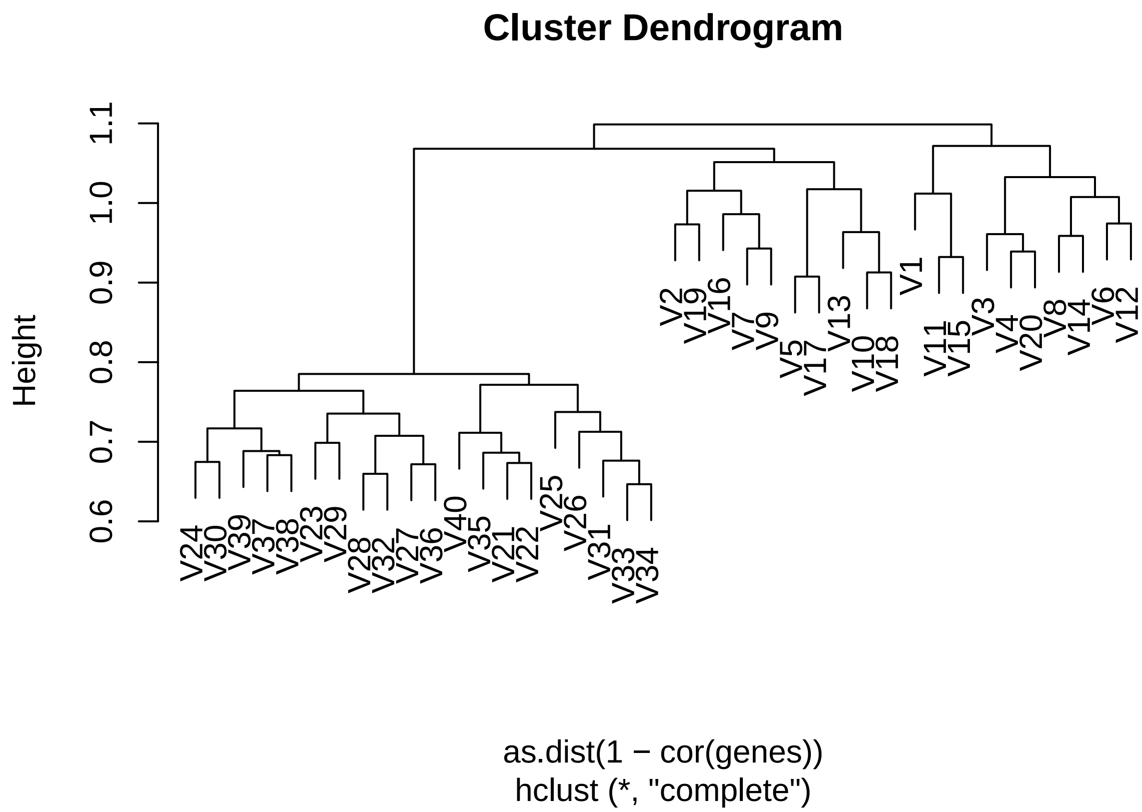
```
genes = read.csv("Ch10Ex11.csv", header=FALSE)
head(genes)
```

```
##            V1         V2         V3         V4         V5         V6
## 1 -0.96193340  0.4418028 -0.9750051  1.4175040  0.8188148  0.3162937
## 2 -0.29252570 -1.1392670  0.1958370 -1.2811210 -0.2514393  2.5119970
## 3  0.25878820 -0.9728448  0.5884858 -0.8002581 -1.8203980 -2.0589240
## 4 -1.15213200 -2.2131680 -0.8615249  0.6309253  0.9517719 -1.1657240
## 5  0.19578280  0.5933059  0.2829921  0.2471472  1.9786680 -0.8710180
## 6  0.03012394 -0.6910143 -0.4034258 -0.7298590 -0.3640986  1.1253490
##            V7          V8          V9        V10        V11        V12
## 1 -0.02496682 -0.06396600  0.03149702 -0.3503106 -0.7227299 -0.2819547
## 2 -0.92220620  0.05954277 -1.40964500 -0.6567122 -0.1157652  0.8259783
## 3 -0.06476437  1.59212400 -0.17311700 -0.1210874 -0.1875790 -1.5001630
## 4 -0.39155860  1.06361900 -0.35000900 -1.4890580 -0.2432189 -0.4330340
## 5 -0.98971500 -1.03225300 -1.10965400 -0.3851423  1.6509570 -1.7449090
## 6 -1.40404100 -0.80613040 -1.23792400  0.5776018 -0.2720642  2.1765620
##           V13         V14        V15        V16        V17        V18
## 1  1.33751500  0.70197980  1.0076160 -0.4653828  0.6385951  0.2867807
## 2  0.34644960 -0.56954860 -0.1315365  0.6902290 -0.9090382  1.3026420
## 3 -1.22873700  0.85598900  1.2498550 -0.8980815  0.8702058 -0.2252529
## 4 -0.03879128 -0.05789677 -1.3977620 -0.1561871 -2.7359820  0.7756169
## 5 -0.37888530 -0.67982610 -2.1315840 -0.2301718  0.4661243 -1.8004490
## 6  1.43640700 -1.02578100  0.2981582 -0.5559659  0.2046529 -1.1916480
##          V19         V20        V21        V22        V23        V24
## 1 -0.2270782 -0.22004520 -1.2425730 -0.1085056 -1.8642620 -0.5005122
```

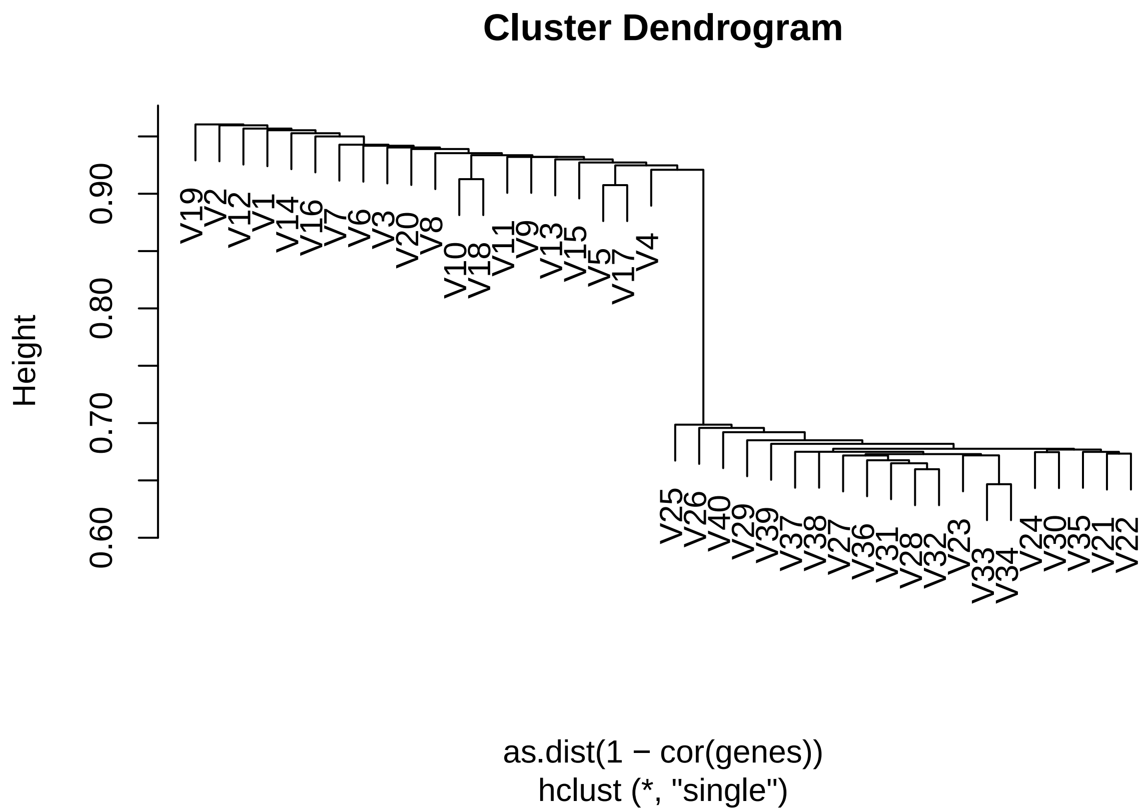
```
## 2 -1.6726950 -0.52550400  0.7979700 -0.6897930  0.8995305  0.4285812
## 3  0.4502892  0.55144040  0.1462943  0.1297400  1.3042290 -1.6619080
## 4  0.6141562  2.01919400  1.0811390 -1.0766180 -0.2434181  0.5134822
## 5  0.6262904 -0.09772305 -0.2997108 -0.5295591 -2.0235670 -0.5108402
## 6  0.2350916  0.67096470  0.1307988  1.0689940  1.2309870  1.1344690
##           V25         V26        V27        V28         V29         V30
## 1 -1.32500800  1.06341100 -0.2963712 -0.1216457  0.08516605  0.62417640
## 2 -0.67611410 -0.53409490 -1.7325070 -1.6034470 -1.08362000  0.03342185
## 3 -1.63037600 -0.07742528  1.3061820  0.7926002  1.55946500 -0.68851160
## 4 -0.51285780  2.55167600 -2.3143010 -1.2764700 -1.22927100  1.43439600
## 5  0.04600274  1.26803000 -0.7439868  0.2231319  0.85846280  0.27472610
## 6  0.55636800 -0.35876640  1.0798650 -0.2064905 -0.00616453  0.16425470
##          V31          V32         V33        V34        V35        V36
## 1 -0.5095915 -0.216725500 -0.05550597 -0.4844491 -0.5215811  1.9491350
## 2  1.7007080  0.007289556  0.09906234  0.5638533 -0.2572752 -0.5817805
## 3 -0.6154720  0.009999363  0.94581000 -0.3185212 -0.1178895  0.6213662
## 4 -0.2842774  0.198945600 -0.09183320  0.3496279 -0.2989097  1.5136960
## 5 -0.6929984 -0.845707200 -0.17749680 -0.1664908  1.4831550 -1.6879460
## 6  1.1567370  0.241774500  0.08863952  0.1829540  0.9426771 -0.2096004
##           V37        V38         V39        V40
## 1  1.32433500  0.4681471  1.06110000  1.6559700
## 2 -0.16988710 -0.5423036  0.31293890 -1.2843770
## 3 -0.07076396  0.4016818 -0.01622713 -0.5265532
## 4  0.67118470  0.0108553 -1.04368900  1.6252750
## 5 -0.14142960  0.2007785 -0.67594210  2.2206110
## 6  0.53626210 -1.1852260 -0.42274760  0.6243603
```

(b) Apply hierarchical clustering to the samples using correlation- based distance, and plot the dendrogram. Do the genes separate the samples into the two groups? Do your results depend on the type of linkage used?

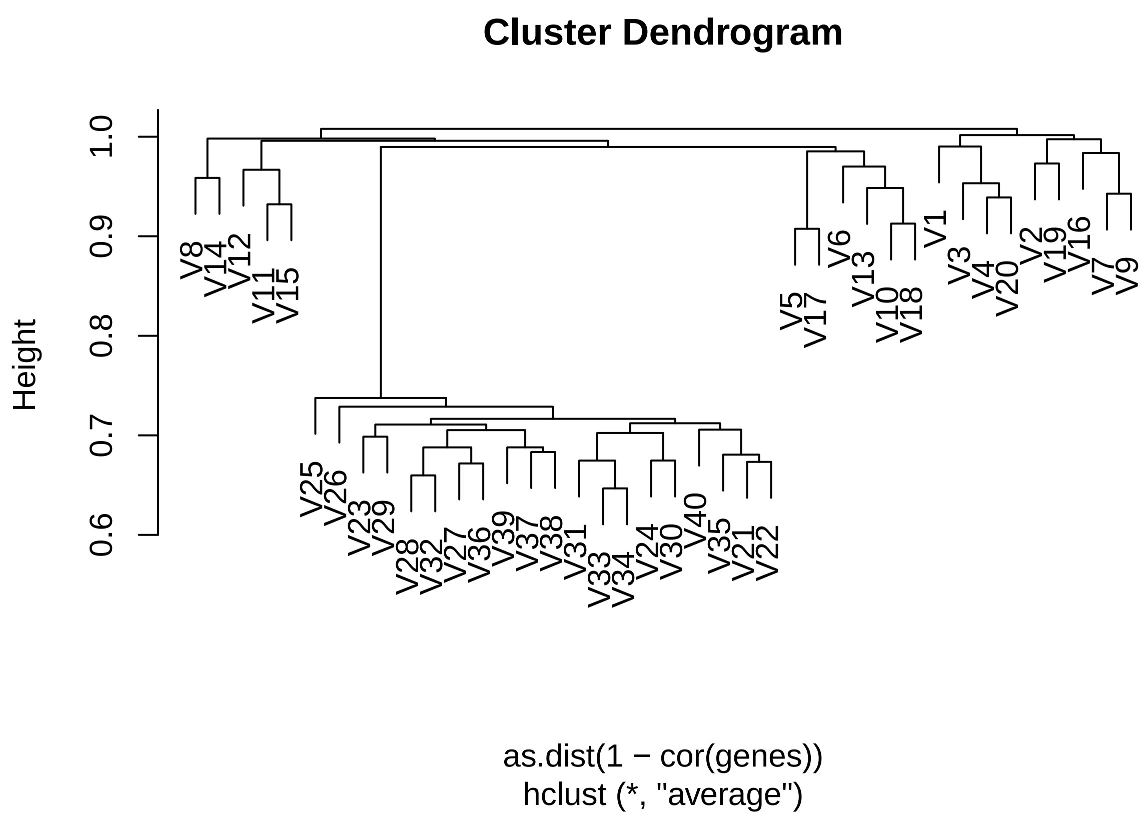
```
plot(hclust(as.dist(1 - cor(genes)), method = "complete"))
```




```
plot(hclust(as.dist(1 - cor(genes)), method = "single"))
```




```
plot(hclust(as.dist(1 - cor(genes)), method = "average"))
```



Based on the plots above, we can obtain two clusters for complete and single linkages or three clusters for average cluster.

(c) Your collaborator wants to know which genes differ the most across the two groups. Suggest a way to answer this question, and apply it here.

We can use PCA to determine which genes function best to illustrate the variance

```
pr.out <-  prcomp(t(genes))
summary(pr.out)
```

```
## Importance of components:
##                            PC1     PC2     PC3     PC4     PC5     PC6
## Standard deviation     11.9409 6.06818 5.93476 5.83115 5.75209 5.70031
## Proportion of Variance  0.1267 0.03271 0.03129 0.03021 0.02939 0.02887
## Cumulative Proportion   0.1267 0.15939 0.19068 0.22089 0.25029 0.27915
##                            PC7     PC8     PC9    PC10    PC11    PC12
## Standard deviation     5.63448 5.57726 5.54943 5.50625 5.48852 5.46025
## Proportion of Variance 0.02821 0.02764 0.02736 0.02694 0.02676 0.02649
## Cumulative Proportion  0.30736 0.33499 0.36236 0.38929 0.41605 0.44254
##                           PC13    PC14    PC15    PC16    PC17    PC18
## Standard deviation     5.40230 5.33441 5.27756 5.21594 5.20000 5.15140
## Proportion of Variance 0.02593 0.02528 0.02475 0.02417 0.02402 0.02358
## Cumulative Proportion  0.46847 0.49375 0.51850 0.54267 0.56669 0.59027
##                           PC19    PC20    PC21    PC22    PC23    PC24
## Standard deviation     5.11600 5.05591 5.03836 5.01868 4.95965 4.91393
## Proportion of Variance 0.02325 0.02271 0.02255 0.02238 0.02185 0.02145
```

```
## Cumulative Proportion  0.61352 0.63623 0.65878 0.68116 0.70301 0.72447
##                           PC25    PC26    PC27    PC28    PC29    PC30
## Standard deviation     4.86397 4.81796 4.80811 4.73485 4.70098 4.65564
## Proportion of Variance 0.02102 0.02062 0.02054 0.01992 0.01963 0.01926
## Cumulative Proportion  0.74548 0.76611 0.78665 0.80656 0.82620 0.84545
##                           PC31    PC32    PC33    PC34    PC35    PC36
## Standard deviation     4.61621 4.56733 4.53032 4.49528 4.36502 4.35858
## Proportion of Variance 0.01893 0.01853 0.01823 0.01795 0.01693 0.01688
## Cumulative Proportion  0.86439 0.88292 0.90115 0.91910 0.93603 0.95291
##                           PC37    PC38    PC39      PC40
## Standard deviation     4.26700 4.20277 4.13922 5.251e-15
## Proportion of Variance 0.01618 0.01569 0.01522 0.000e+00
## Cumulative Proportion  0.96909 0.98478 1.00000 1.000e+00
```

```r
total_load <-  apply(pr.out$rotation, 1, sum)
indices <-  order(abs(total_load), decreasing = TRUE)
indices[1:10]
```

```
##  [1] 865  68 911 428 624  11 524 803 980 822
```

```r
total_load[indices[1:10]]
```

```
##  [1]  0.7765416  0.7137785 -0.7099501 -0.6363706 -0.6195945  0.5885202
##  [7]  0.5583279  0.5535498 -0.5217130  0.4981997
```

Above shows the top 10 genes which are most distinct from others