HW 1 PH 240C

Daniel Lee

September 24, 2016

```
library(Biobase)
library(genefilter)
library(gplots)
library(hopach)
library(RColorBrewer)
```

Question 1

Examine the different objects in the R dataset `examiningDoxorubicinInDetail.RData`. Store the ex pression measures and the sample- and gene-level annotation metadata related to `doxorubicinNCI60Scaled` and `doxorubicin07Numbers` in objects of class `ExpressionSet` (Bioconductor R package `Biobase`).

```
load("C:\\Users\\Daniel\\Desktop\\Fall 2016\\PH 240C\\examiningDoxorubicinInDetail.RData")
```

```
### Creating an Expression Set for doxorubicinNCI60Scaled ###

exprs_scaled <- as.matrix(doxorubicinNCI60Scaled)
dim(exprs_scaled)
```

```
## [1] 12625    22
```

```
#phenodata for doxorubicinNCI60Scaled
samples_scaled <- data.frame(c(rep("Sensitive", 10), rep("Resistant", 12)))
rownames(samples_scaled) <- colnames(exprs_scaled)
colnames(samples_scaled) <- "Status"

#Expression Set for doxorubicinNCI60Scaled created
ExprSet.NCI60.Scaled <- ExpressionSet(assayData = exprs_scaled,
                                      phenoData = as(samples_scaled,
                                                     "AnnotatedDataFrame"))

### Creating an Expression Set for doxorubicin07Numbers ###
exprs_no <- as.matrix(doxorubicin07Numbers)
dim(exprs_no)
```

```
## [1] 8958  144
```

```
#phenodata for doxorubicin07Numbers
samples_no <- data.frame(doxorubicin07Info)
dim(samples_no)
```

```
## [1] 144   2
```

```
#Expression Set for doxorubicin07Numbers created
ExprSet.Doxo07.Numbers <- ExpressionSet(assayData = exprs_no,
                            phenoData = as(samples_no, "AnnotatedDataFrame"))
```

Question 2

Reconcile the training data in `doxorubicinNCI60Scaled` and `doxorubicin07Numbers`, i.e., match genes and samples and compare the expression measures and the sensitivity status assigned to the cell lines in the two

To answer this question, I first see if the training data microarray expression measures for the first gene `36460_at` in ExprSet.NCI60.Scaled has any matches to the microarray expression measures for any of the genes in the ExprSet.Doxo07.Numbers. I do this using correlation.

```
#Use correlation to see if the first gene 36460_at in
#doxorubicinNCI60Scaled for all the cell lines is the same as the
#microarray expression of the gene 36460_at in doxorubicin07Numbers for
#all the cell lines in the training data
temp <- cor(exprs(ExprSet.NCI60.Scaled)[1, ],
            t(exprs(ExprSet.Doxo07.Numbers)[, 1:22]))

max(temp)
```

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

```
sum(temp == max(temp))
```

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

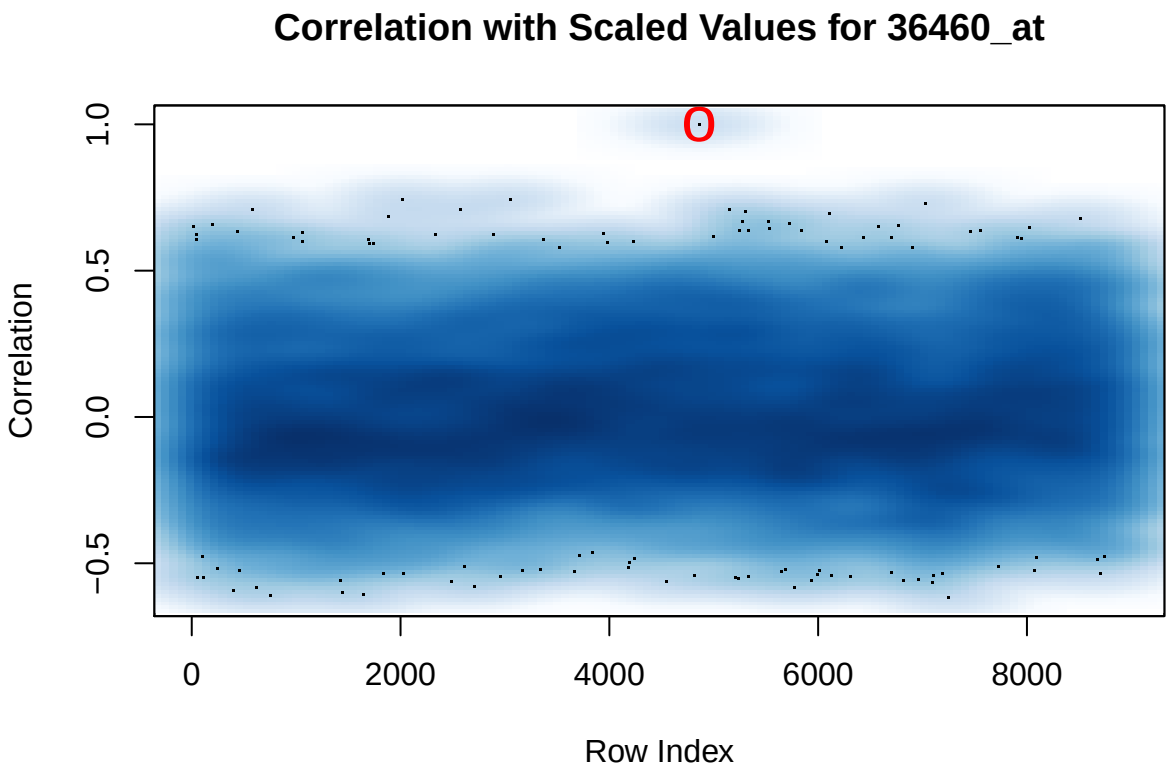
I notice that there is one gene with the exact same microarray expression measures as the first gene `36460_at`.

Now, I create a scatter plot to see if there are any other genes in `doxorubicin07Numbers` with relatively close gene expressions as gene `36460_at`.

```
rownames(exprs(ExprSet.Doxo07.Numbers))[which.max(temp)]
```

```
## [1] "36460_at"
```

```
smoothScatter(1:length(temp), temp,
              xlab = "Row Index",
              ylab = "Correlation",
              main = paste("Correlation with Scaled Values for",
                                                  rownames(exprs(ExprSet.NCI60.Scaled))[1]))
points(which.max(temp), max(temp), pch = "o", col = "red", cex = 2)
```



I see that the rest of the genes have pretty different microarray expressions.

Now I examine microarray expressions for each gene in training data of the `doxorubicinNCI60Scaled` expression set and see if they have exact matches in `doxorubicin07Numbers`.

```
#Find and store all the row indices that has maximum correlations for
#genes in the training data in doxorubicinNCI60Scaled to those in
#doxorubicin07Numbers
tempIndices <- apply(exprs(ExprSet.Doxo07.Numbers)[, 1:22], 1,
                     function(x) {
                       which.max(cor(x, t(exprs(ExprSet.NCI60.Scaled))))
                       }
                     )

#Give the output of the maximum correlations for the gene expressions
#between genes in doxorubicinNCI60Scaled to those in
#doxorubicin07Numbers
tempCors <- apply(cbind(exprs(ExprSet.Doxo07.Numbers)[, 1:22],
                        exprs(ExprSet.NCI60.Scaled)[tempIndices, ]), 1,
                  function(x) {
                    cor(x[1:22], x[23:44])
                    }
                  )
min(tempCors)
```

```
## [1] 0.9999924
```

I notice that the minimum of the tempCors is 0.9999924, which is essentially 1. This slightly lower value is due to rounding error.

This suggests that the cell lines in the training set of `doxorubicin07Numbers` is the same cell lines in `doxorubicinNCI60Scaled`.

Few outputs are shown below to confirm this.

```
exprs(ExprSet.Doxo07.Numbers)[1:4, 1:5]
```

```
##          Training1 Training2 Training3 Training4 Training5
## 35753_at      1.18      1.12      3.46      0.65      3.07
## 36138_at      1.75      4.02      0.43      0.31      0.76
## 41765_at      0.13      0.35      1.13      1.14      0.84
## 35298_at      0.19      0.42      0.52      0.52      1.32
```

```
exprs(ExprSet.NCI60.Scaled)[rownames(exprs(ExprSet.Doxo07.Numbers))[1:4], 1:5]
```

```
##          SF-539 SNB-75 MDA-MB-435 NCI-H23  M14
## 35753_at   1.18   1.12       3.46    0.65 3.07
## 36138_at   1.75   4.02       0.43    0.31 0.76
## 41765_at   0.13   0.35       1.13    1.14 0.84
## 35298_at   0.19   0.42       0.52    0.52 1.32
```

Now, I examine the sensitivity status assigned to the cell lines in the two datasets.

```
pData(ExprSet.Doxo07.Numbers)$status[1:22]
```

```
##  [1] Resistant Resistant Resistant Resistant Resistant Resistant Resistant
##  [8] Resistant Resistant Resistant Sensitive Sensitive Sensitive Sensitive
## [15] Sensitive Sensitive Sensitive Sensitive Sensitive Sensitive Sensitive
## [22] Sensitive
## Levels: Resistant Sensitive
```

```
pData(ExprSet.NCI60.Scaled)[,1]
```

```
##  [1] Sensitive Sensitive Sensitive Sensitive Sensitive Sensitive Sensitive
##  [8] Sensitive Sensitive Sensitive Resistant Resistant Resistant Resistant
## [15] Resistant Resistant Resistant Resistant Resistant Resistant Resistant
## [22] Resistant
## Levels: Resistant Sensitive
```

```
pData(ExprSet.Doxo07.Numbers)$status[1:22] == pData(ExprSet.NCI60.Scaled)[,1]
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

I notice that all the labels are switched. That is, the cell lines that are labeled as resistant in `doxorubicin07Numbers` is labeled as sensitive in `doxorubicinNCI60Scaled` and vice versa.

Question 3

Consider now the test data in `doxorubicin07Numbers`. Is there anything unusual with the samples and their assigned sensitivity statuses? Hint: Consider pairwise distances and dimensionality reduction and clustering methods.

To answer this question, I first do a PCA.

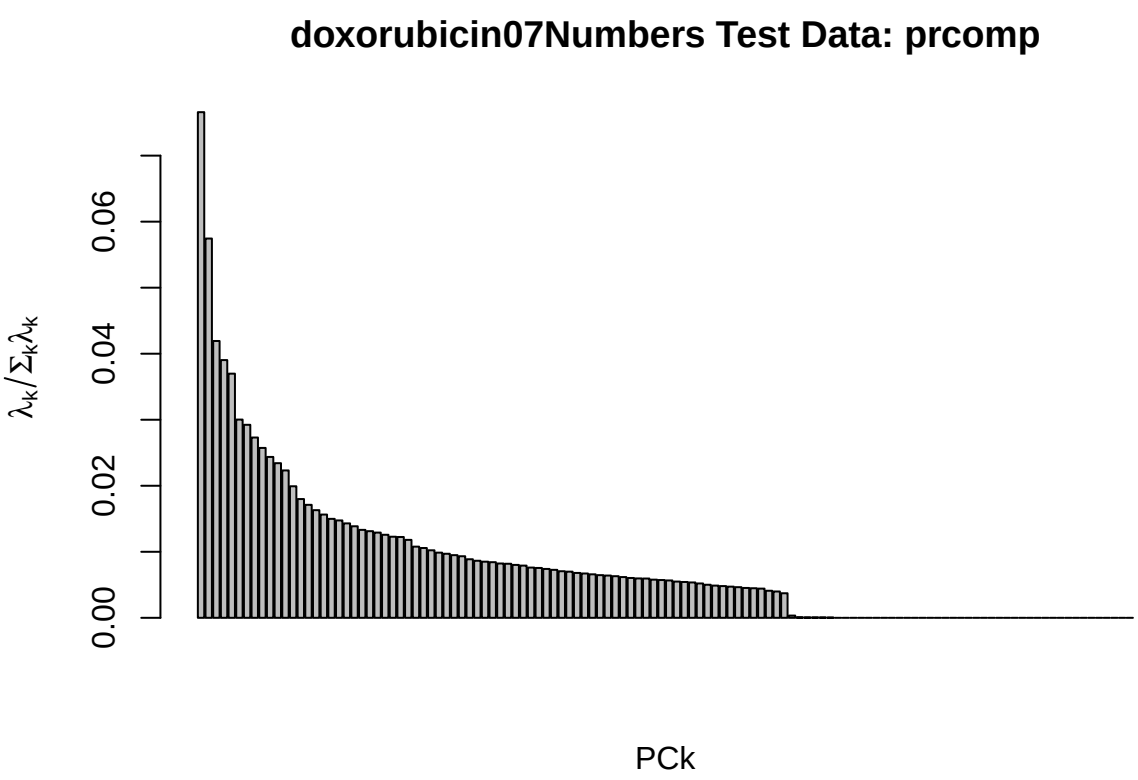
```
#PCA

#log-transform the data first
log_doxo_test_numbers <- log2(exprs(ExprSet.Doxo07.Numbers)[ , 23:144] + 1)

#Store the resistant/sensitive labels to object Y
Y <- pData(ExprSet.Doxo07.Numbers)$status

#Designate the colors for resistant and sensitive labels
colG <- c("red", "blue")[factor(Y)]

#Run principal component analysis
res <- prcomp(t(log_doxo_test_numbers),retx=TRUE)

#Print the summary of PCA
summary(res)
```

```
## Importance of components:
##                             PC1      PC2      PC3      PC4      PC5
## Standard deviation     18.37639 15.91505 13.59703 13.11960 12.76950
## Proportion of Variance  0.07658  0.05744  0.04193  0.03903  0.03698
## Cumulative Proportion   0.07658  0.13402  0.17595  0.21498  0.25196
##                             PC6      PC7      PC8      PC9     PC10
## Standard deviation     11.50571 11.35318 10.97292 10.65134 10.36493
## Proportion of Variance  0.03002  0.02923  0.02731  0.02573  0.02436
## Cumulative Proportion   0.28198  0.31121  0.33852  0.36425  0.38861
##                            PC11    PC12    PC13    PC14    PC15   PC16
## Standard deviation     10.16219 9.92074 9.37257 8.90584 8.68633 8.4789
## Proportion of Variance  0.02342 0.02232 0.01992 0.01799 0.01711 0.0163
## Cumulative Proportion   0.41203 0.43435 0.45427 0.47226 0.48937 0.5057
##                           PC17    PC18    PC19   PC20    PC21    PC22
## Standard deviation     8.30349 8.12917 8.06481 7.9412 7.81907 7.66330
## Proportion of Variance 0.01564 0.01499 0.01475 0.0143 0.01386 0.01332
## Cumulative Proportion  0.52131 0.53629 0.55104 0.5653 0.57921 0.59253
##                           PC23    PC24    PC25    PC26    PC27    PC28
## Standard deviation     7.60606 7.54427 7.44676 7.35959 7.34676 7.21541
## Proportion of Variance 0.01312 0.01291 0.01258 0.01228 0.01224 0.01181
## Cumulative Proportion  0.60565 0.61855 0.63113 0.64341 0.65565 0.66746
##                           PC29    PC30    PC31    PC32   PC33    PC34
## Standard deviation     6.89371 6.82780 6.71561 6.59722 6.5407 6.46928
## Proportion of Variance 0.01078 0.01057 0.01023 0.00987 0.0097 0.00949
## Cumulative Proportion  0.67824 0.68881 0.69904 0.70891 0.7186 0.72810
##                           PC35    PC36    PC37   PC38    PC39    PC40
## Standard deviation     6.41228 6.25374 6.17368 6.1224 6.09526 6.02275
## Proportion of Variance 0.00932 0.00887 0.00864 0.0085 0.00843 0.00823
## Cumulative Proportion  0.73742 0.74629 0.75494 0.7634 0.77186 0.78009
```

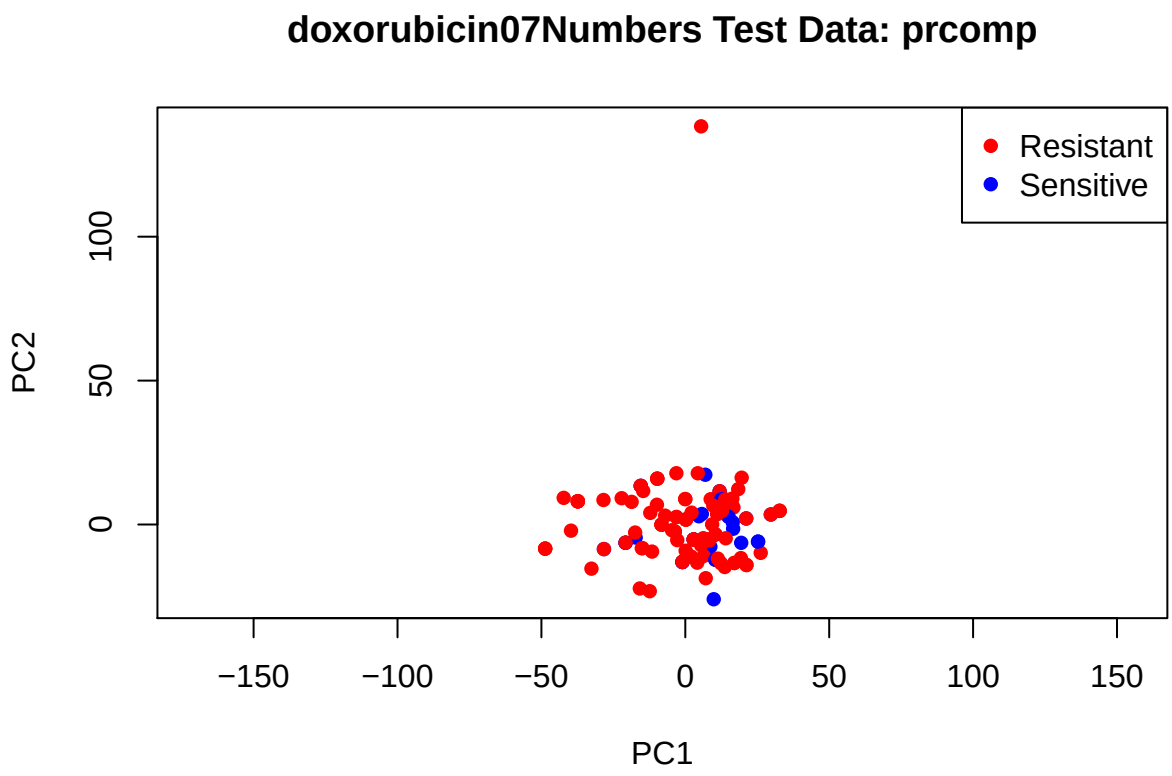
```
##                            PC41    PC42    PC43    PC44    PC45   PC46
## Standard deviation     6.00754 5.94462 5.90518 5.79297 5.77081 5.7127
## Proportion of Variance 0.00818 0.00801 0.00791 0.00761 0.00755 0.0074
## Cumulative Proportion  0.78827 0.79629 0.80419 0.81180 0.81936 0.8268
##                            PC47    PC48    PC49   PC50    PC51    PC52
## Standard deviation     5.65975 5.57715 5.55183 5.4743 5.43976 5.38991
## Proportion of Variance 0.00726 0.00705 0.00699 0.0068 0.00671 0.00659
## Cumulative Proportion  0.83402 0.84108 0.84807 0.8549 0.86157 0.86816
##                            PC53   PC54   PC55    PC56    PC57    PC58
## Standard deviation     5.33858 5.3130 5.2720 5.21680 5.15762 5.12952
## Proportion of Variance 0.00646 0.0064 0.0063 0.00617 0.00603 0.00597
## Cumulative Proportion  0.87462 0.8810 0.8873 0.89350 0.89953 0.90550
##                            PC59    PC60    PC61    PC62    PC63    PC64
## Standard deviation     5.12136 5.04734 5.02323 4.99523 4.91927 4.89290
## Proportion of Variance 0.00595 0.00578 0.00572 0.00566 0.00549 0.00543
## Cumulative Proportion  0.91145 0.91722 0.92295 0.92861 0.93409 0.93952
##                            PC65   PC66    PC67    PC68    PC69    PC70
## Standard deviation     4.85867 4.7885 4.69891 4.63812 4.60376 4.56325
## Proportion of Variance 0.00535 0.0052 0.00501 0.00488 0.00481 0.00472
## Cumulative Proportion  0.94488 0.9501 0.95508 0.95996 0.96477 0.96949
##                            PC71    PC72    PC73    PC74   PC75    PC76
## Standard deviation     4.52433 4.46539 4.44477 4.41630 4.2508 4.19175
## Proportion of Variance 0.00464 0.00452 0.00448 0.00442 0.0041 0.00398
## Cumulative Proportion  0.97413 0.97865 0.98313 0.98756 0.9917 0.99564
##                            PC77    PC78    PC79    PC80    PC81    PC82
## Standard deviation     4.04945 1.22552 0.59069 0.53523 0.51771 0.49914
## Proportion of Variance 0.00372 0.00034 0.00008 0.00006 0.00006 0.00006
## Cumulative Proportion  0.99936 0.99970 0.99978 0.99984 0.99990 0.99996
##                            PC83      PC84      PC85      PC86      PC87
## Standard deviation     0.41788 3.011e-14 2.467e-14 1.337e-14 1.128e-14
## Proportion of Variance 0.00004 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.00000 1.000e+00 1.000e+00 1.000e+00 1.000e+00
##                             PC88      PC89     PC90      PC91     PC92
## Standard deviation     1.048e-14 1.015e-14 9.86e-15 9.244e-15 8.77e-15
## Proportion of Variance 0.000e+00 0.000e+00 0.00e+00 0.000e+00 0.00e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.00e+00 1.000e+00 1.00e+00
##                            PC93      PC94      PC95      PC96      PC97
## Standard deviation     8.21e-15 8.161e-15 7.985e-15 7.279e-15 6.909e-15
## Proportion of Variance 0.00e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.00e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
##                            PC98      PC99     PC100     PC101    PC102
## Standard deviation     6.45e-15 5.483e-15 4.979e-15 4.609e-15 4.56e-15
## Proportion of Variance 0.00e+00 0.000e+00 0.000e+00 0.000e+00 0.00e+00
## Cumulative Proportion  1.00e+00 1.000e+00 1.000e+00 1.000e+00 1.00e+00
##                           PC103     PC104     PC105     PC106     PC107
## Standard deviation     4.238e-15 4.076e-15 3.688e-15 1.885e-15 1.755e-15
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
##                           PC108     PC109     PC110     PC111     PC112
## Standard deviation     1.461e-15 1.461e-15 1.461e-15 1.461e-15 1.461e-15
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
##                           PC113     PC114     PC115     PC116     PC117
## Standard deviation     1.461e-15 1.461e-15 1.461e-15 1.461e-15 1.461e-15
```

```
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
##                            PC118     PC119     PC120     PC121     PC122
## Standard deviation     1.461e-15 1.461e-15 1.461e-15 1.145e-15 5.157e-16
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
```

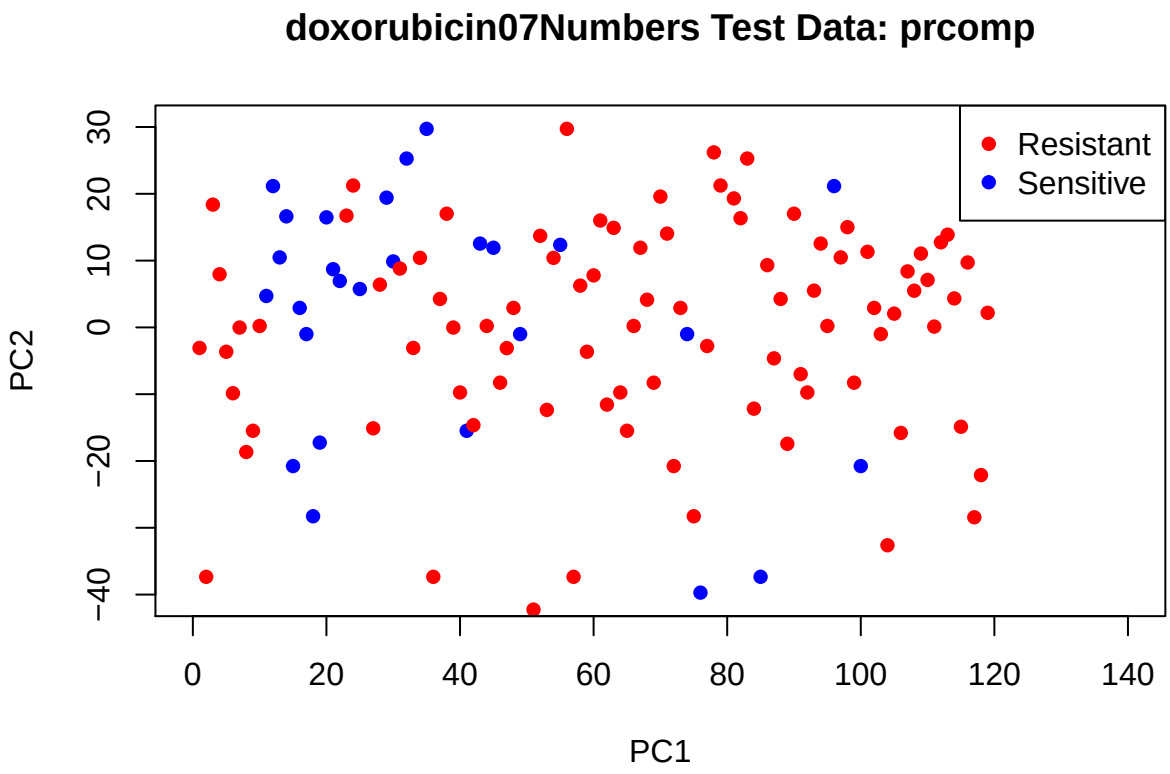
```
#Barplot of proportion of variance explained by each principal component
barplot(res$sdev^2/sum(res$sdev^2), xlab="PCk",
        ylab=expression(lambda[k]/Sigma[k]*lambda[k]),
        main="doxorubicin07Numbers Test Data: prcomp")
```




```
#Scatterplot of first two principal components
plot(res$x[,1:2], pch=16, col = colG, asp = 1,
     main="doxorubicin07Numbers Test Data: prcomp")
legend("topright", c("Resistant", "Sensitive"),
       pch=16, col=c("red", "blue"))
```



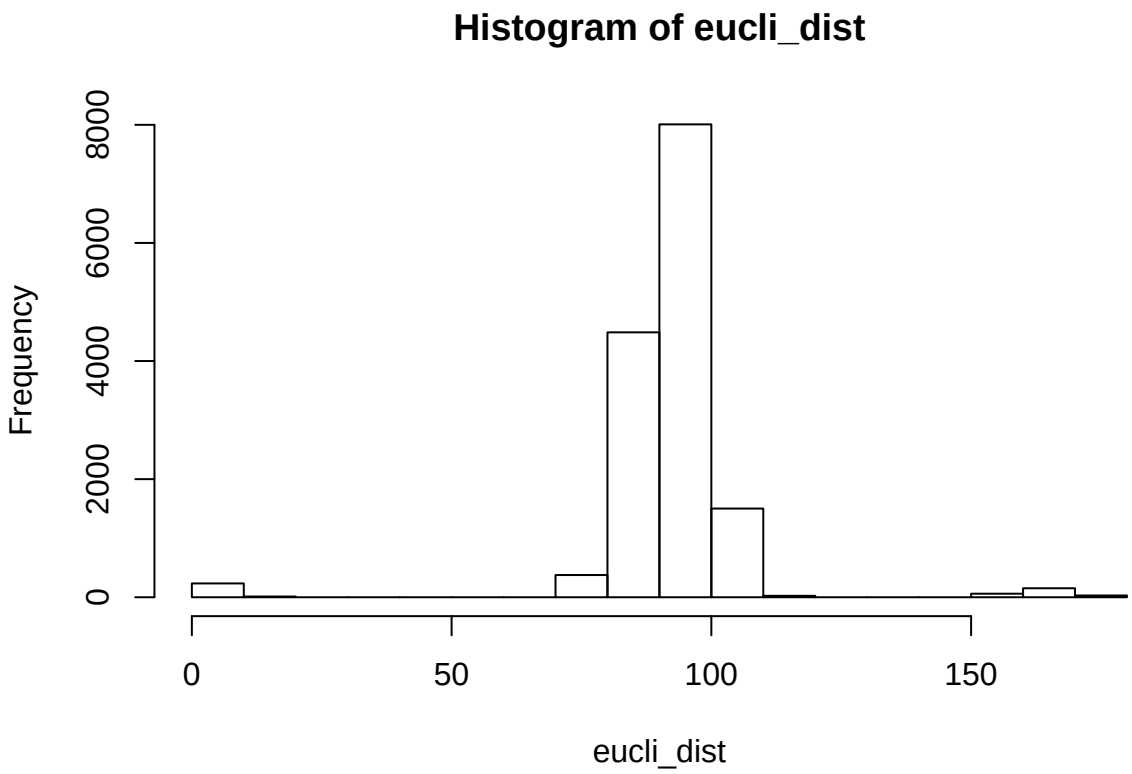

```
#Scatterplot of first two principal components with the outlier removed
plot(res$x[,1:2][which(res$x[,1] != max(res$x[,1]))], pch=16,
     col = colG, asp = 1,
     main="doxorubicin07Numbers Test Data: prcomp",
     xlim = c(0, 140), ylim = c(-40, 30),
     xlab = "PC1", ylab = "PC2")
legend("topright", c("Resistant", "Sensitive"),
       pch=16, col=c("red", "blue"))
```



PCA analysis of the test data for `doxorubicin07Numbers` indicate that there doesn't seem to be much distinction between the microarray expressions between resistant and sensitive cell lines. There is a lot of overlap in the gene expressions. This can also suggest that the assigned sensitivity statuses can be potentially switched.

Next, I examine the Euclidean pairwise distances among the cell lines.

```
#pairwise distance
eucli_dist <- as.matrix(dist(t(log_doxo_test_numbers)))
hist(eucli_dist)
```




```
min(eucli_dist)
```

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

```
sum(eucli_dist == 0)
```

```
## [1] 234
```

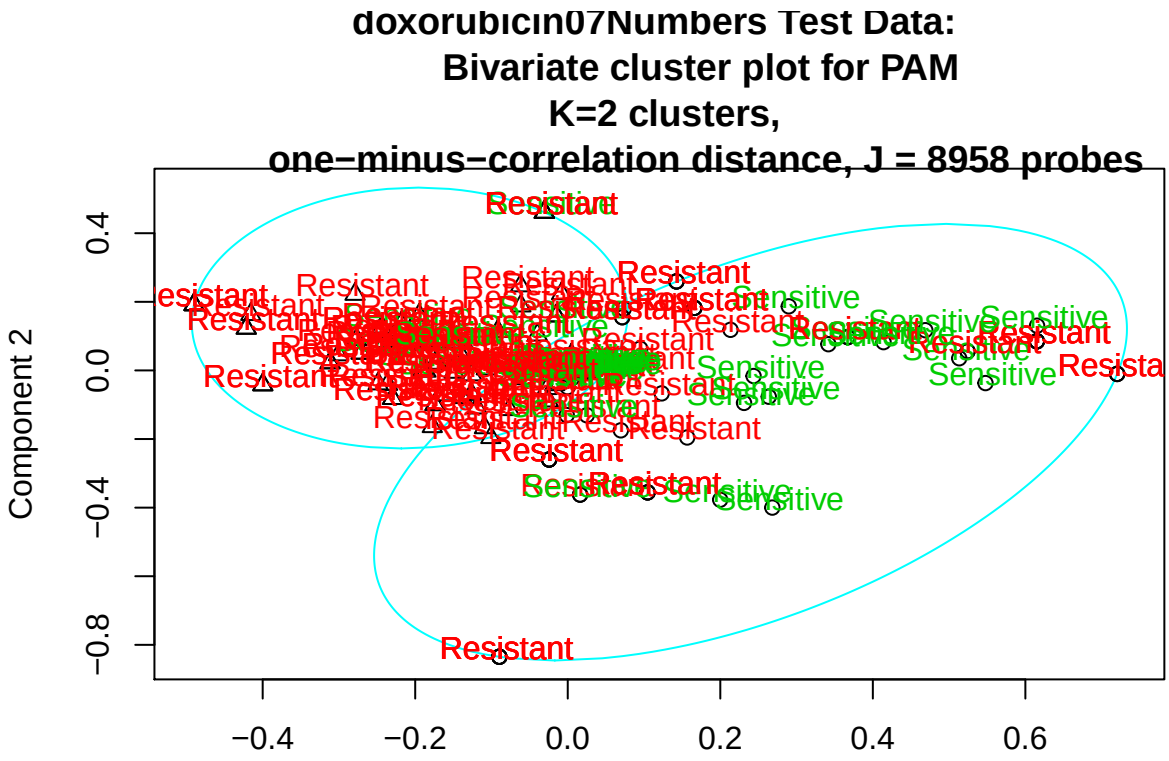
```
#there are more than 122 zeros, which suggests duplicates.
```

The `eucli_dist` is expected to contain 122 zeros since the Euclidean distance for a cell line to itself would be zero. However, the fact that there are 234 zeros indicate that some cell lines have exactly the same microarray expressions. This suggests that data for some cell lines have been included more than once in the dataset.

Next, I examing clustering with PAM.

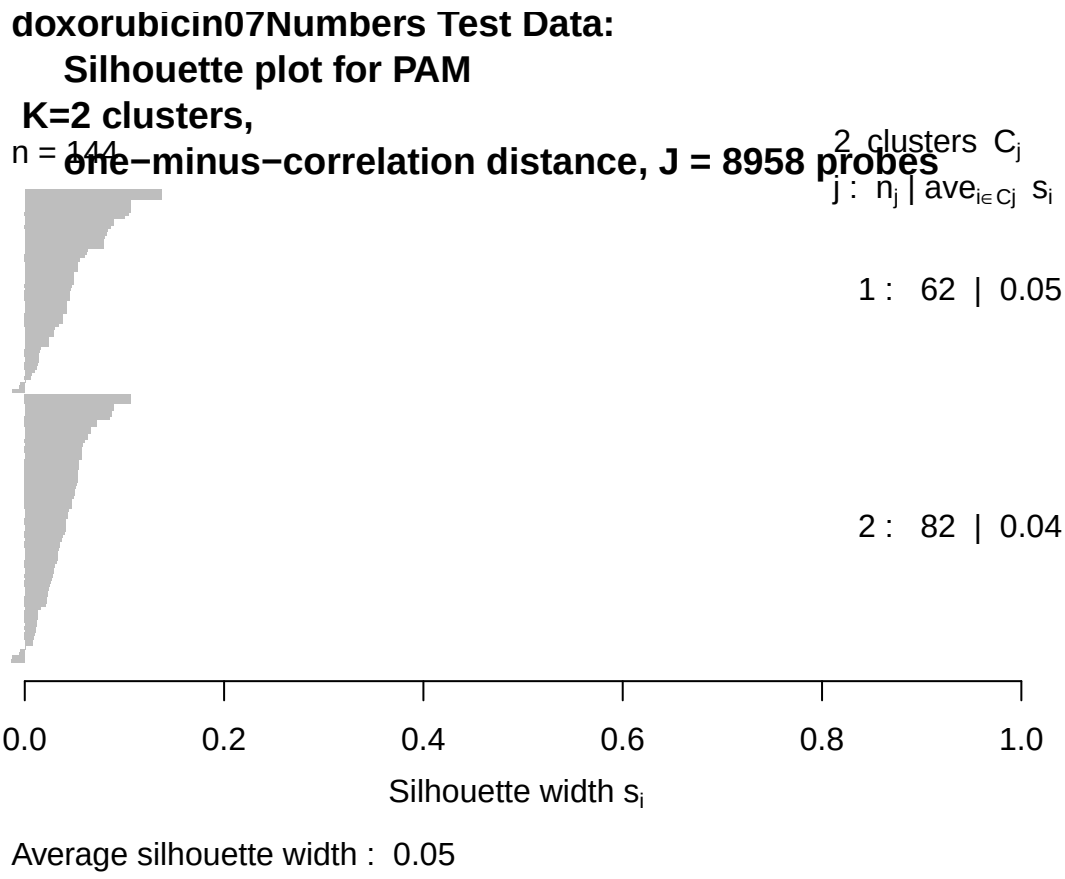
```
#Clustering using PAM
# One-minus-correlation distance matrix
r <- cor(exprs(ExprSet.Doxo07.Numbers))
d <- 1-r
dimnames(d) <- list(as.vector(Y),as.vector(Y))

# PAM, K=2
pam2 <- pam(as.dist(d), k=2, diss=TRUE)
```

```
# PAM, K=3
pam3 <- pam(as.dist(d), k=3, diss=TRUE)

# Graphical summaries
clusplot(d, pam2$clustering, diss=TRUE, labels=3,
         col.p=1, col.txt=rank(unique(Y))[factor(Y)]+1,
         main="doxorubicin07Numbers Test Data:
         Bivariate cluster plot for PAM \n K=2 clusters,
         one-minus-correlation distance, J = 8958 probes")
```

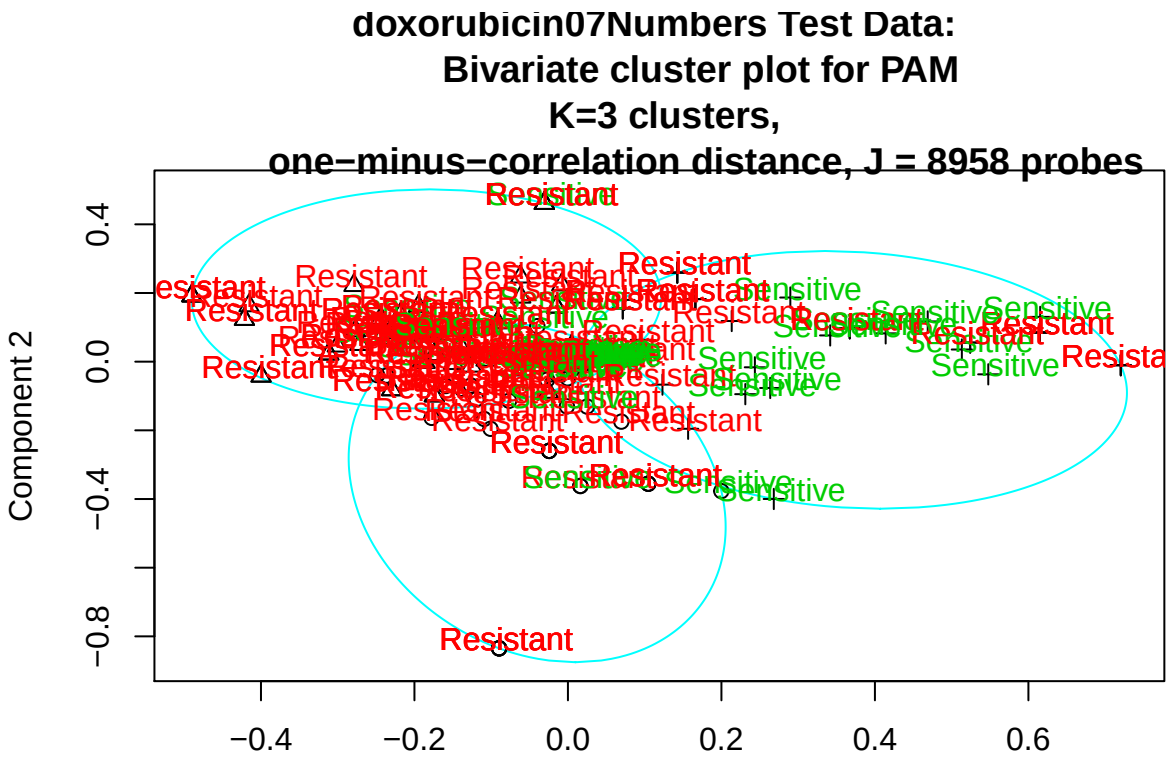


These two components explain 8.28 % of the point variability.

```
plot(pam2,which.plots=2,
     main="doxorubicin07Numbers Test Data:
     Silhouette plot for PAM \n K=2 clusters,
     one-minus-correlation distance, J = 8958 probes")
```

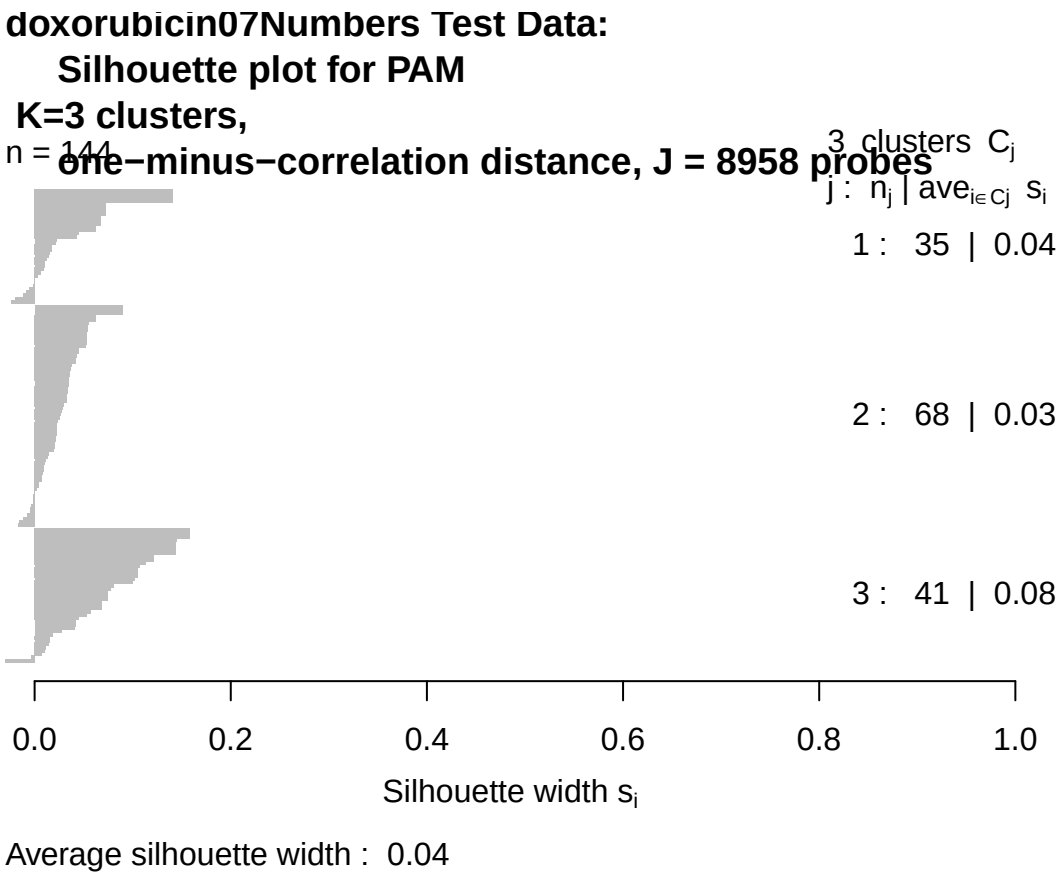



```
clusplot(d, pam3$clustering, diss=TRUE, labels=3,
         col.p=1, col.txt=rank(unique(Y))[factor(Y)]+1,
         main="doxorubicin07Numbers Test Data:
         Bivariate cluster plot for PAM \n K=3 clusters,
         one-minus-correlation distance, J = 8958 probes")
```



These two components explain 8.28 % of the point variability.

```
plot(pam3,which.plots=2,
     main="doxorubicin07Numbers Test Data:
     Silhouette plot for PAM \n K=3 clusters,
     one-minus-correlation distance, J = 8958 probes")
```




```
## ----pam2-----------------------------------------------------------------
table(pam2$clustering, Y)
```

```
##    Y
##     Resistant Sensitive
##   1        43        19
##   2        66        16
```

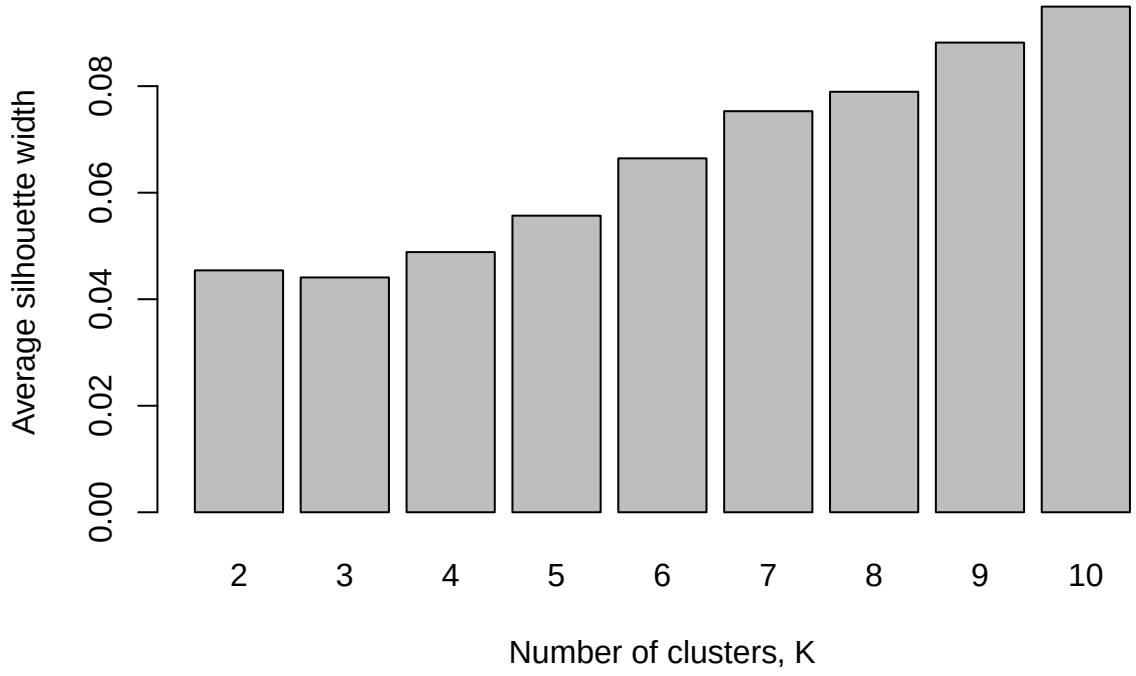
```
## ----pam3-----------------------------------------------------------------
table(pam3$clustering, Y)
```

```
##    Y
##     Resistant Sensitive
##   1        28         7
##   2        55        13
##   3        26        15
```

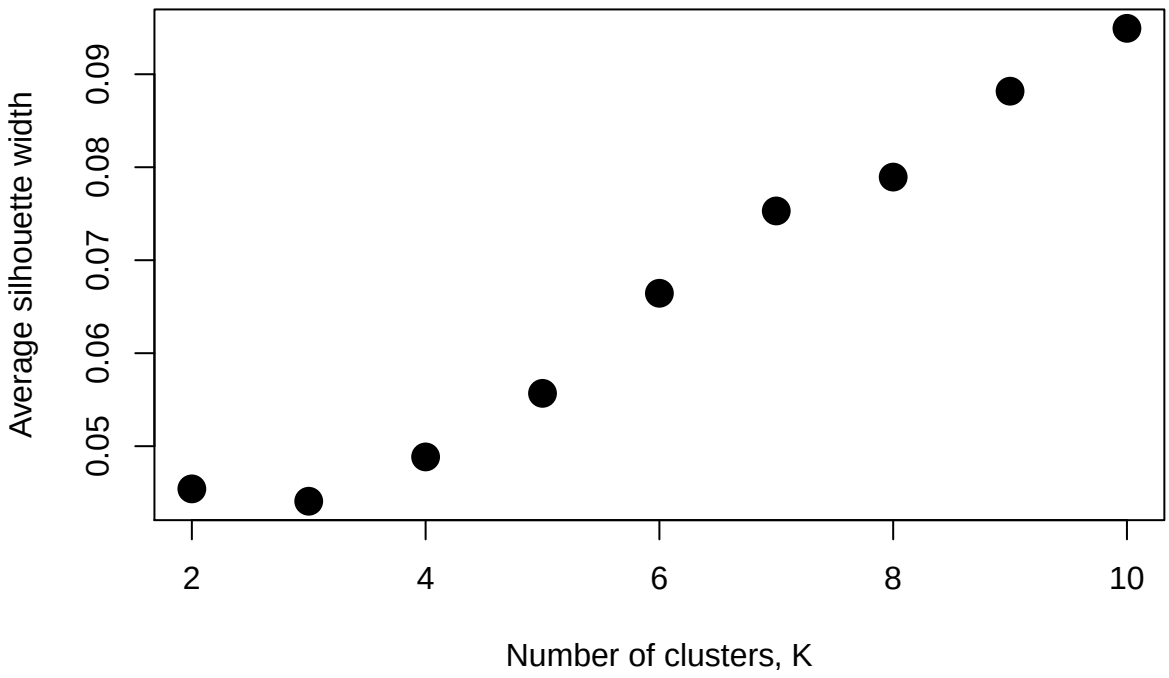
PAM clustering with K = 2 and K = 3 reveal that the clustering is not effective. The silhouette widths of the clusters for both K = 2 and K = 3 clusters are close to zero, indicating that the clustering is not effective.

```
## ----pamSil---------------------------------------------------------------
# Average silhouette widths for PAM with K = 2, ...,10 clusters
K <- 2:10
avgSil <- rep(NA, length(K))
```

```r
names(avgSil) <- K
for(k in K)
  avgSil[k-1] <- pam(as.dist(d), k=k, diss=TRUE)$silinfo$avg.width

# Graphical summaries
barplot(avgSil, names.arg=K, xlab="Number of clusters, K", ylab="Average silhouette width")
```




```r
plot(K, avgSil, pch=16, cex=2, xlab="Number of clusters, K", ylab="Average silhouette width")
```




```
## ----avgSil----------------------------------------------------------------
round(avgSil,3)
```

```
##     2     3     4     5     6     7     8     9    10
## 0.045 0.044 0.049 0.056 0.066 0.075 0.079 0.088 0.095
```

```
K[which.max(avgSil)]
```

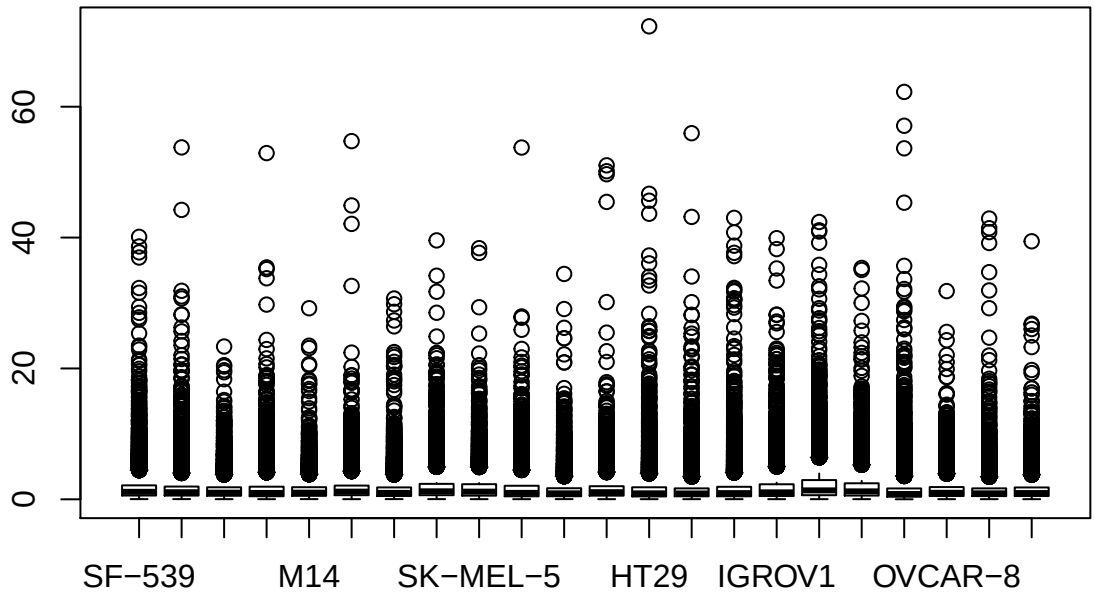
```
## [1] 10
```

The fact that the number of clusters with the highest average silhouette width value is K = 10 clusters indicate that there is something wrong with the data. I would expect the fit to be best for K = 2 clusters since there are two labels "resistant" and "sensitive" cell lines. The inconclusive results from PAM suggests that the sensitivity labels for the different cell lines could have been incorrectly assigned.

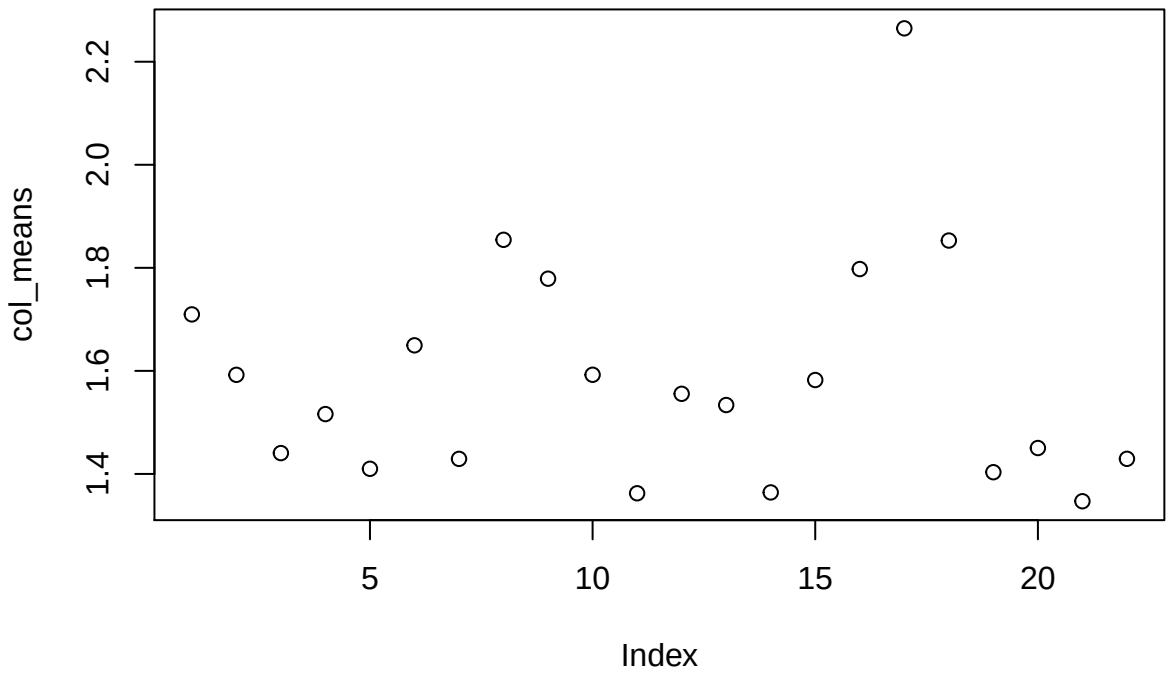
Question 4

a) Between-sample Normalization

```
boxplot(exprs(ExprSet.NCI60.Scaled))
```



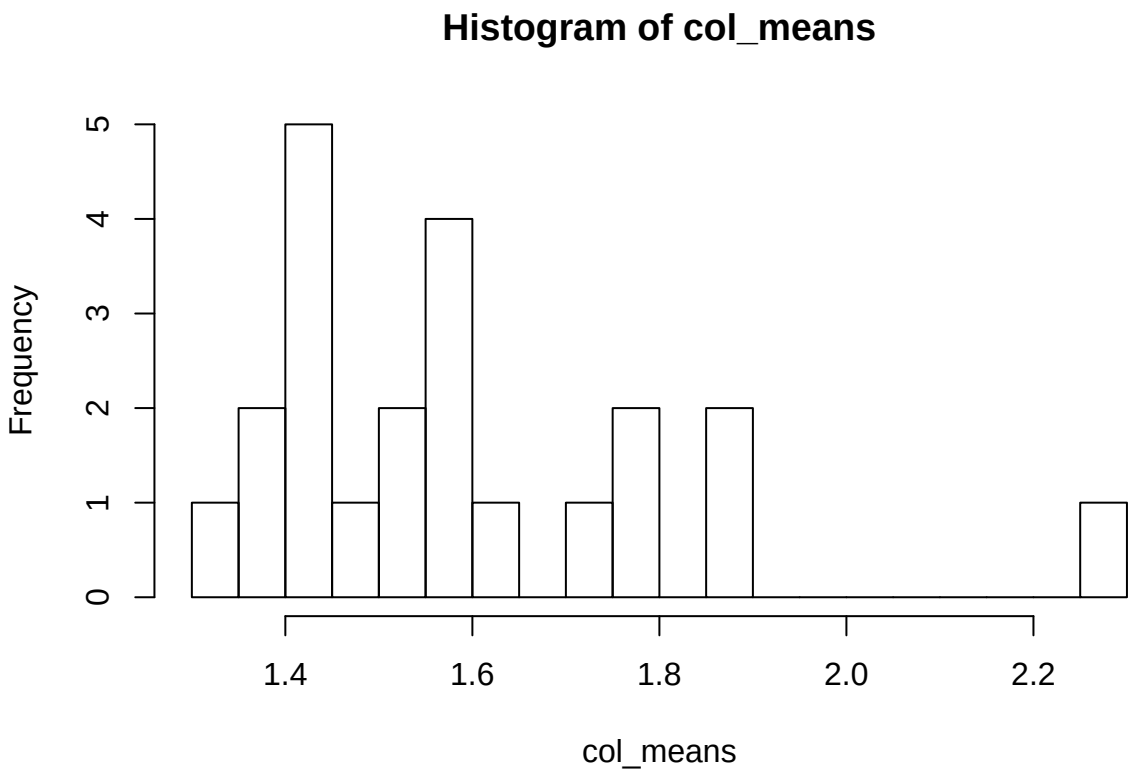

```
col_means <- apply(exprs(ExprSet.NCI60.Scaled), 2, mean)
plot(col_means)
```



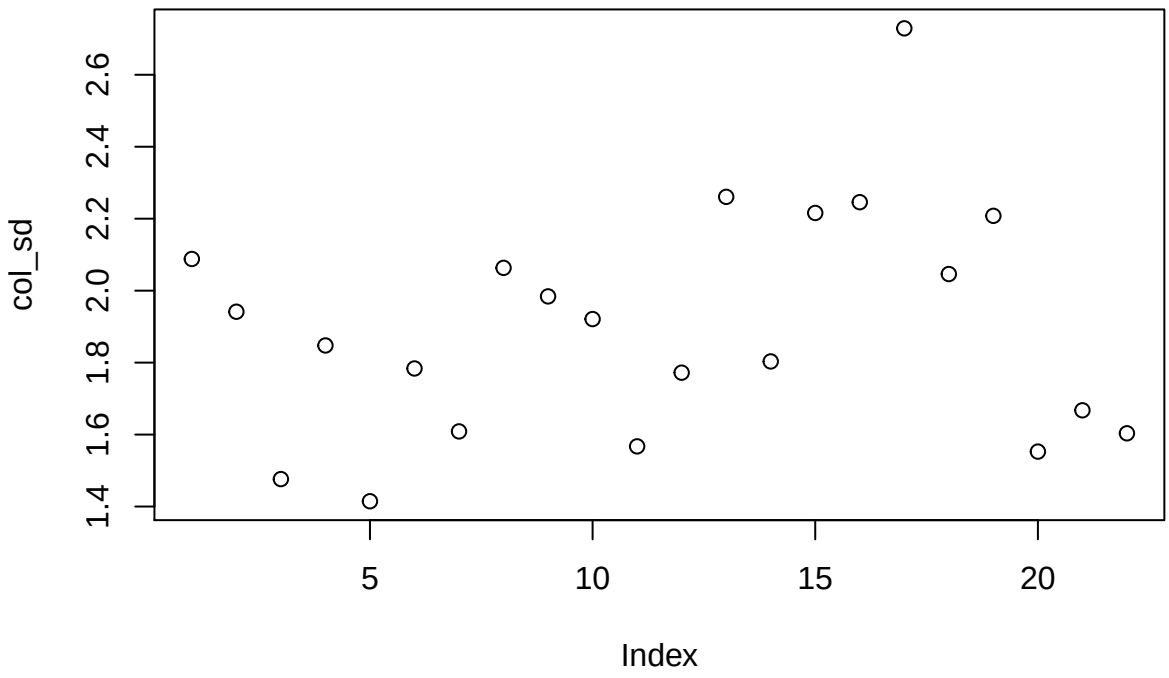

```
summary(col_means)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.347   1.429   1.545   1.587   1.695   2.265
```

```
hist(col_means, breaks = 22)
```




```
col_sd <- apply(exprs(ExprSet.NCI60.Scaled), 2, sd)
plot(col_sd)
```



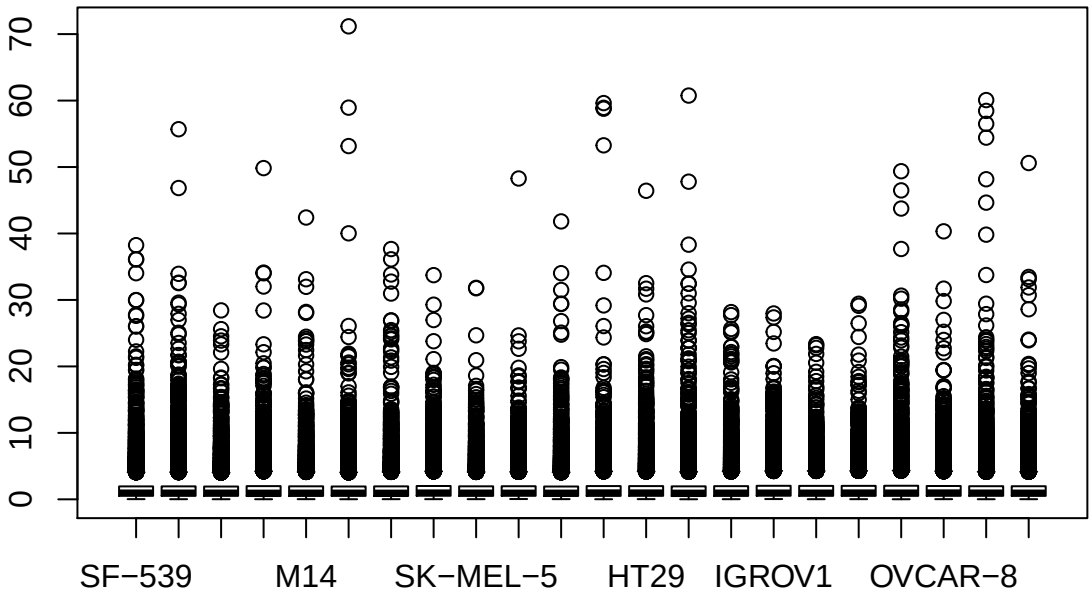
The boxplots among the cell lines in `doxorubicinNCI60Scaled` indicate that generally, the gene expressions are right-skewed for all cell lines. There is a wide range of the gene expression means among the cell lines. There is also a wide range of variation of gene expressions among the cells. This suggests that the data needs to be normalized.

I perform two normalizations. One is the loess procedure using `affy` package. The other is the full-quantile normalization using the `limma` package.

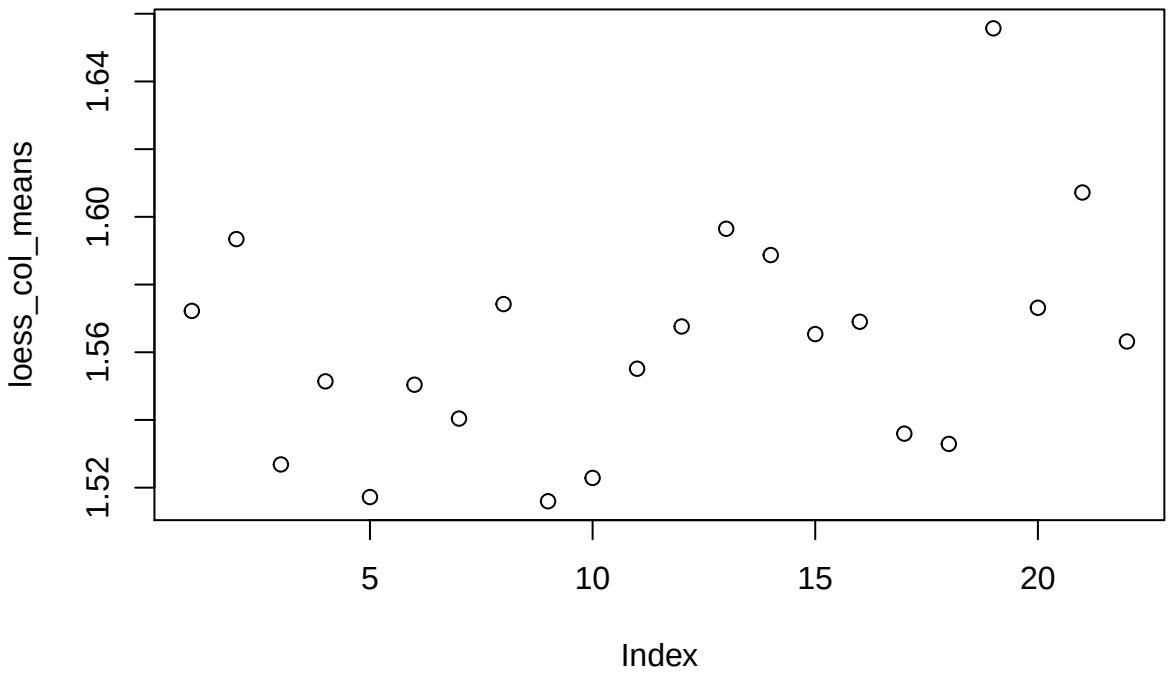
```
#loess procedure using affy package
library(affy)
```

```
exprSet.loess <- normalize.loess(exprs(ExprSet.NCI60.Scaled))
```

```
boxplot(exprSet.loess)
```



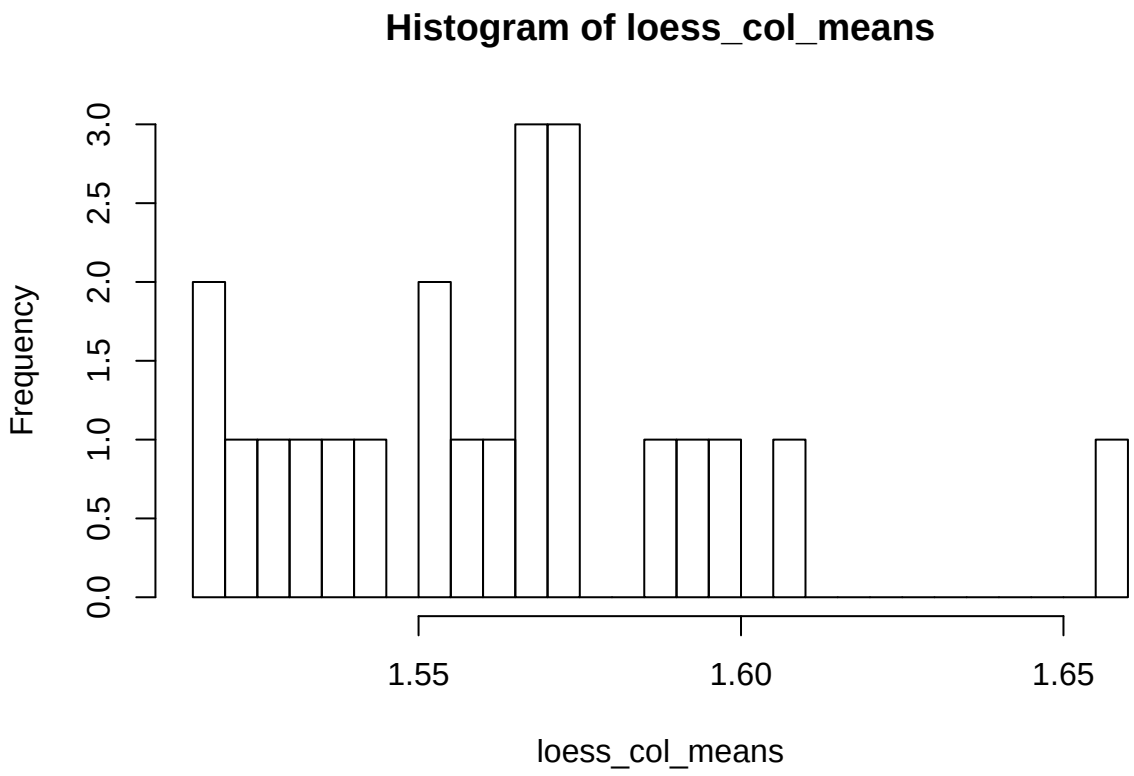

```
loess_col_means <- apply(exprSet.loess, 2, mean)
plot(loess_col_means)
```



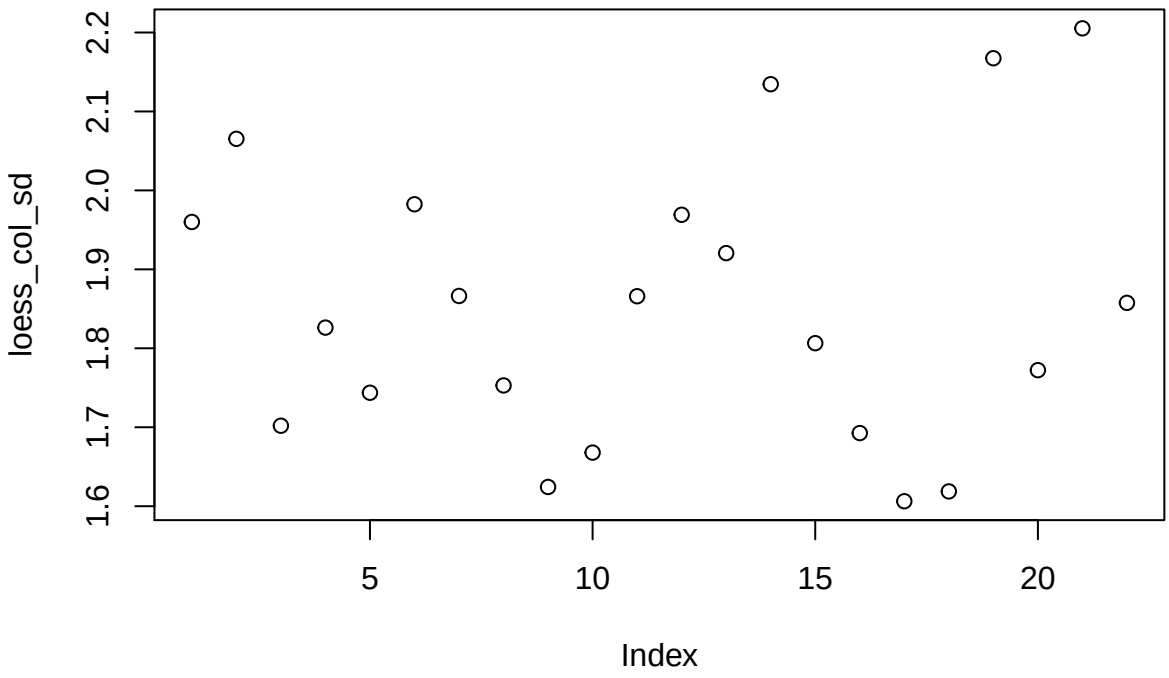

```
summary(loess_col_means)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.516   1.537   1.564   1.563   1.574   1.656
```

```
hist(loess_col_means, breaks = 22)
```



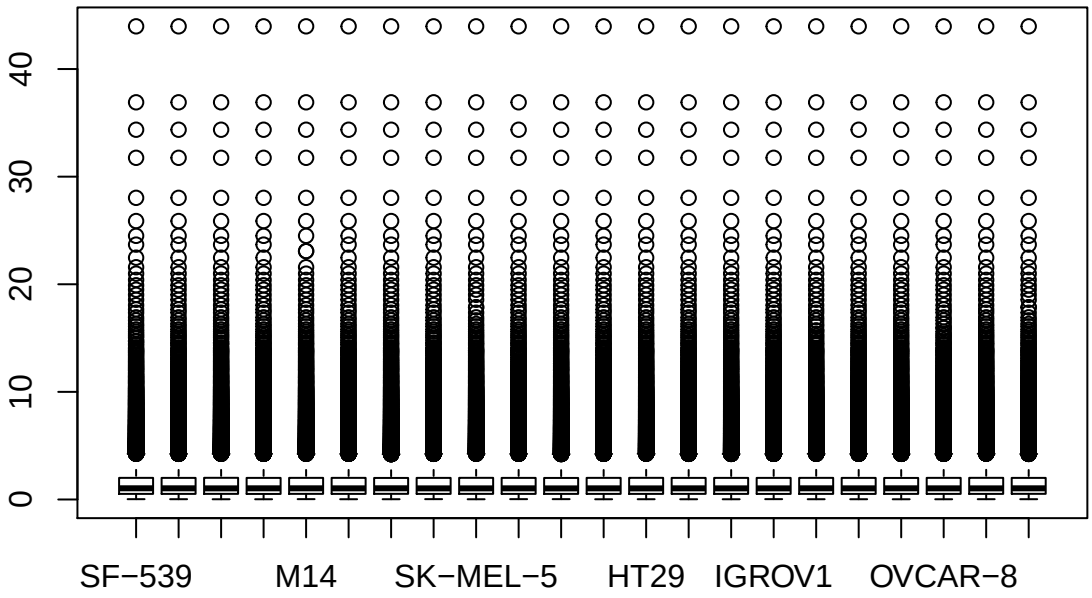

```
loess_col_sd <- apply(exprSet.loess, 2, sd)
plot(loess_col_sd)
```



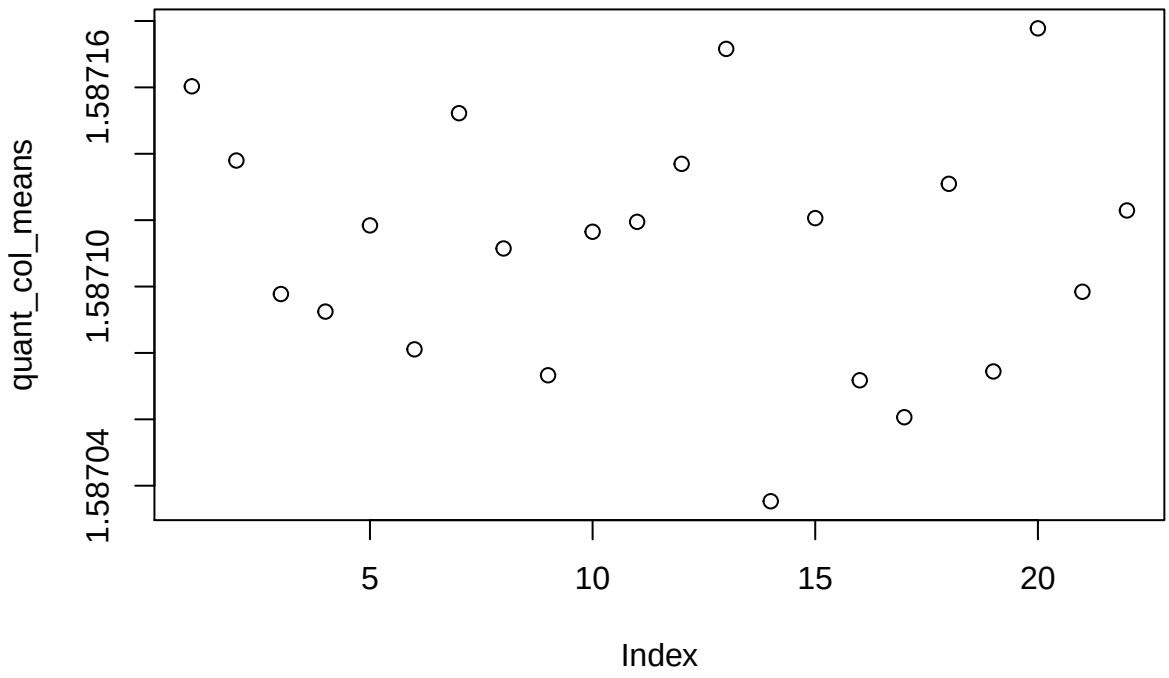

```
#full-quantile normalization using limma package
library(limma)
```

```
##
## Attaching package: 'limma'

## The following object is masked from 'package:BiocGenerics':
##
##     plotMA
```

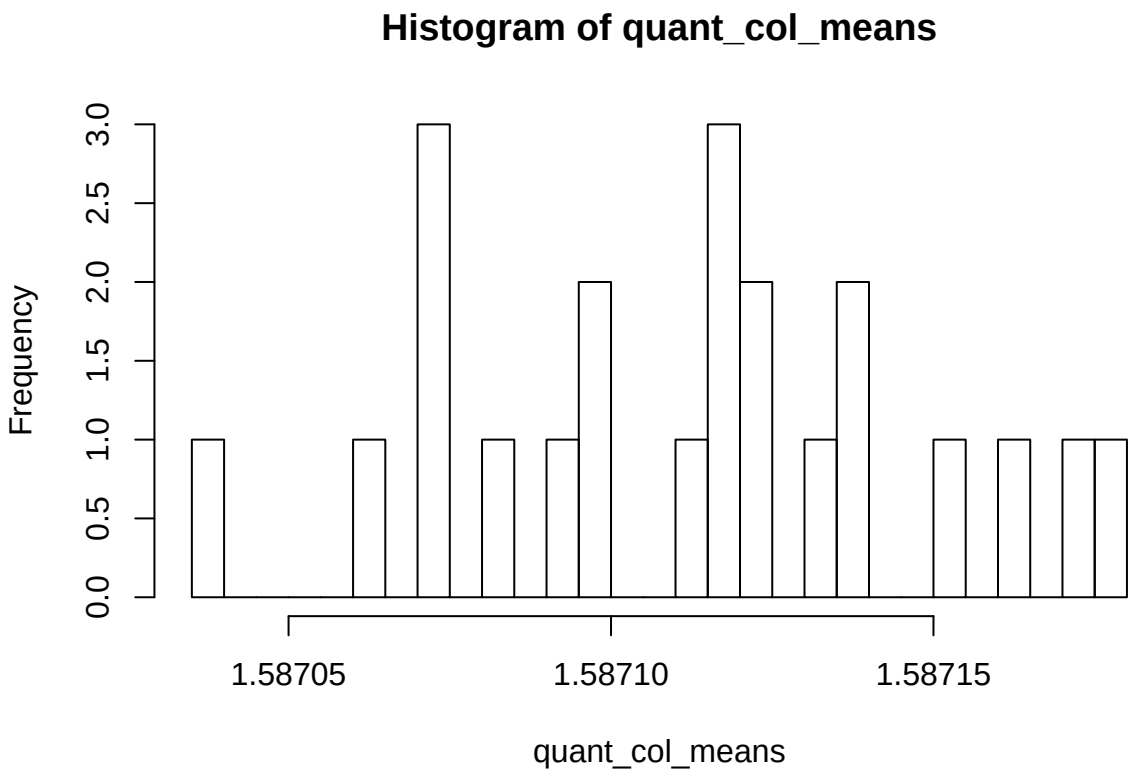
```
exprSet.quantile = normalizeQuantiles(exprs(ExprSet.NCI60.Scaled))
boxplot(exprSet.quantile)
```




```
quant_col_means <- apply(exprSet.quantile, 2, mean)
plot(quant_col_means)
```



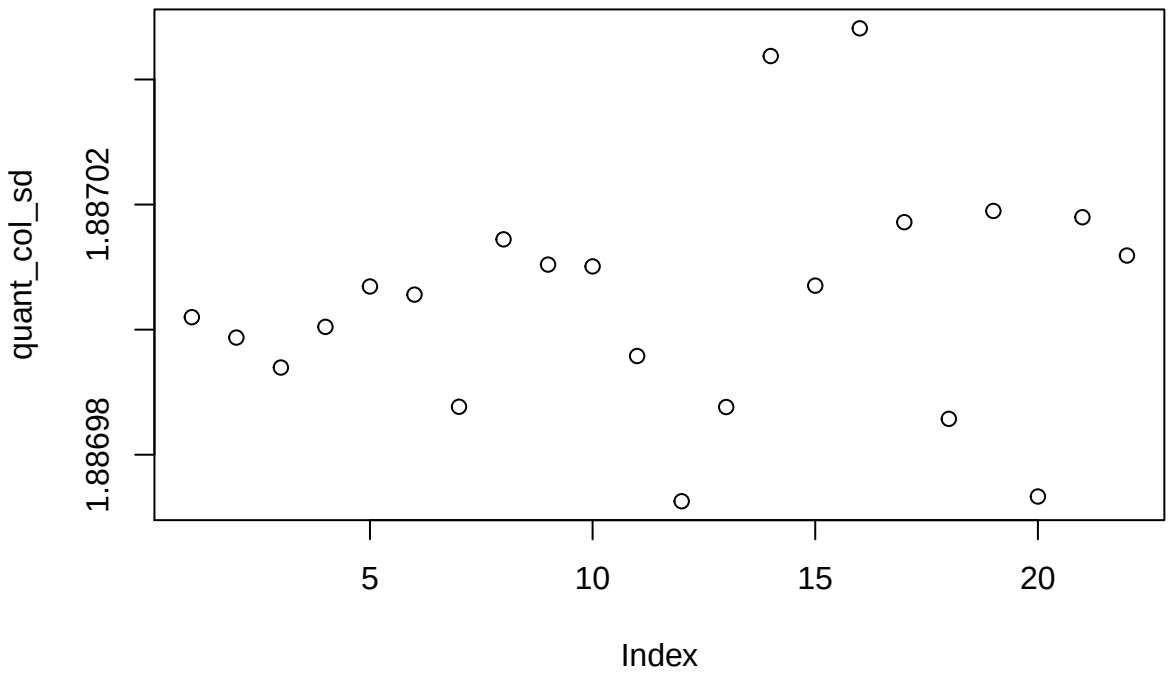

```
hist(quant_col_means, breaks = 22)
```



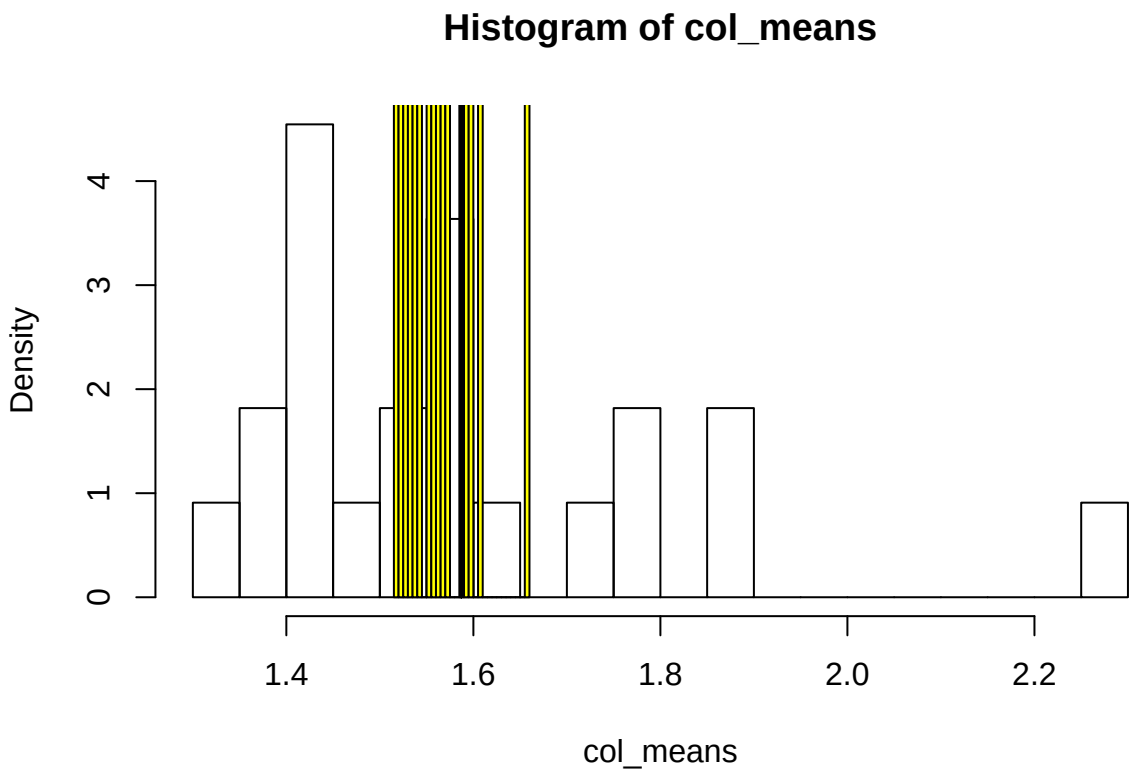

```
summary(quant_col_means)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.587   1.587   1.587   1.587   1.587   1.587
```

```
quant_col_sd <- apply(exprSet.quantile, 2, sd)
plot(quant_col_sd)
```




```
hist(col_means, breaks = 22, freq = FALSE)
hist(loess_col_means, breaks = 22, add = TRUE, col = "yellow", freq = FALSE)
hist(quant_col_means, breaks = 22, add = TRUE, col = "red", freq = FALSE)
```



After normalization, the data has a smaller mean and sd range. The full-quantile normalization has a smaller mean range than the loess normalization.

b Cluster analysis

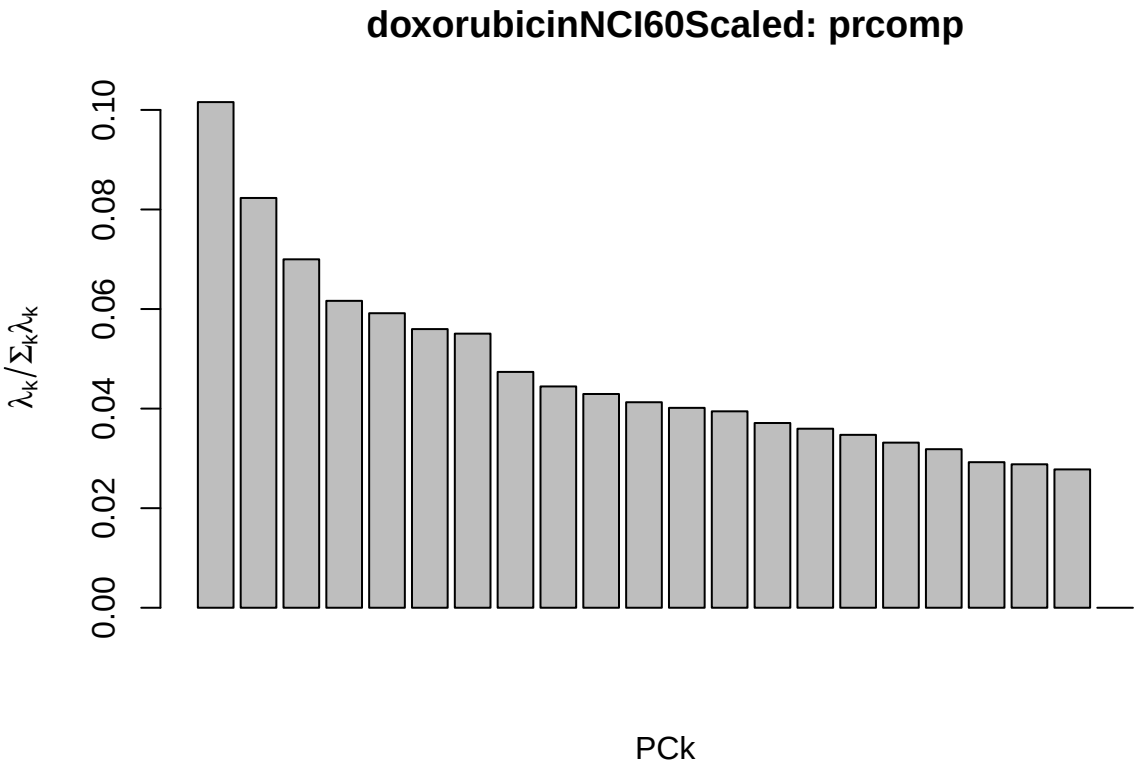
The following analysis is done using the loess normalized data. First, I perform the PCA.

```
#PCA
Y <- pData(ExprSet.NCI60.Scaled)[,1]

colG <- c("red", "blue")[factor(Y)]

res <- prcomp(t(exprSet.loess),retx=TRUE)

summary(res)
```

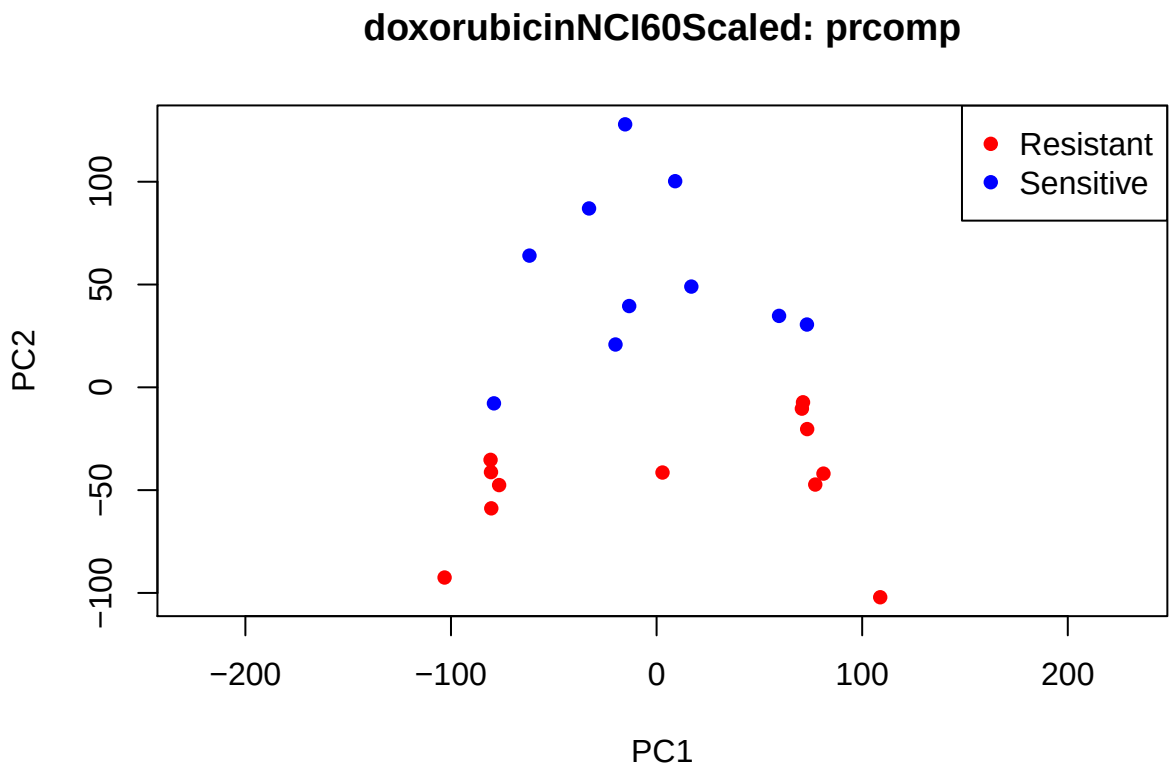
```
## Importance of components:
##                             PC1      PC2      PC3      PC4      PC5
## Standard deviation     67.9690 61.18745 56.41983 52.95246 51.87434
## Proportion of Variance  0.1016  0.08231  0.06998  0.06165  0.05916
## Cumulative Proportion   0.1016  0.18388  0.25386  0.31551  0.37467
##                             PC6      PC7      PC8      PC9     PC10
## Standard deviation     50.45806 50.04203 46.41910 44.96435 44.19328
## Proportion of Variance  0.05597  0.05506  0.04737  0.04445  0.04294
## Cumulative Proportion   0.43064  0.48570  0.53307  0.57752  0.62046
```

```
##                             PC11     PC12     PC13     PC14     PC15
## Standard deviation     43.33267 42.73520 42.36169 41.08481 40.44480
## Proportion of Variance  0.04128  0.04015  0.03945  0.03711  0.03596
## Cumulative Proportion   0.66174  0.70189  0.74134  0.77845  0.81442
##                             PC16     PC17     PC18     PC19     PC20
## Standard deviation     39.74440 38.83911 38.06139 36.46928 36.20274
## Proportion of Variance  0.03473  0.03316  0.03185  0.02924  0.02881
## Cumulative Proportion   0.84914  0.88231  0.91416  0.94340  0.97221
##                             PC21      PC22
## Standard deviation     35.55183 1.587e-13
## Proportion of Variance  0.02779 0.000e+00
## Cumulative Proportion   1.00000 1.000e+00
```

```
barplot(res$sdev^2/sum(res$sdev^2), xlab="PCk",
        ylab=expression(lambda[k]/Sigma[k]*lambda[k]),
        main="doxorubicinNCI60Scaled: prcomp")
```




```
plot(res$x[,1:2], pch=16, col = colG, asp = 1,
     main="doxorubicinNCI60Scaled: prcomp")
legend("topright", c("Resistant", "Sensitive"),
       pch=16, col=c("red", "blue"))
```



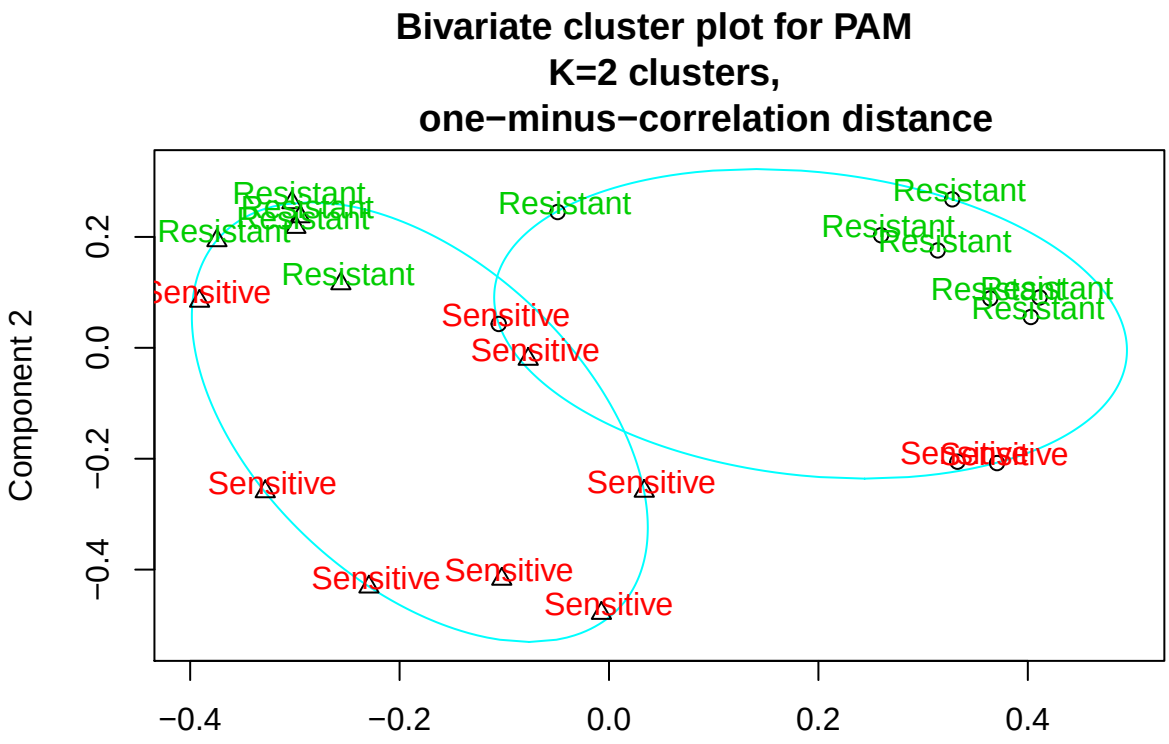

```
invisible(dev.off())
```

PCA shows pretty distinct separation between the resistant and the sensitive cell lines.

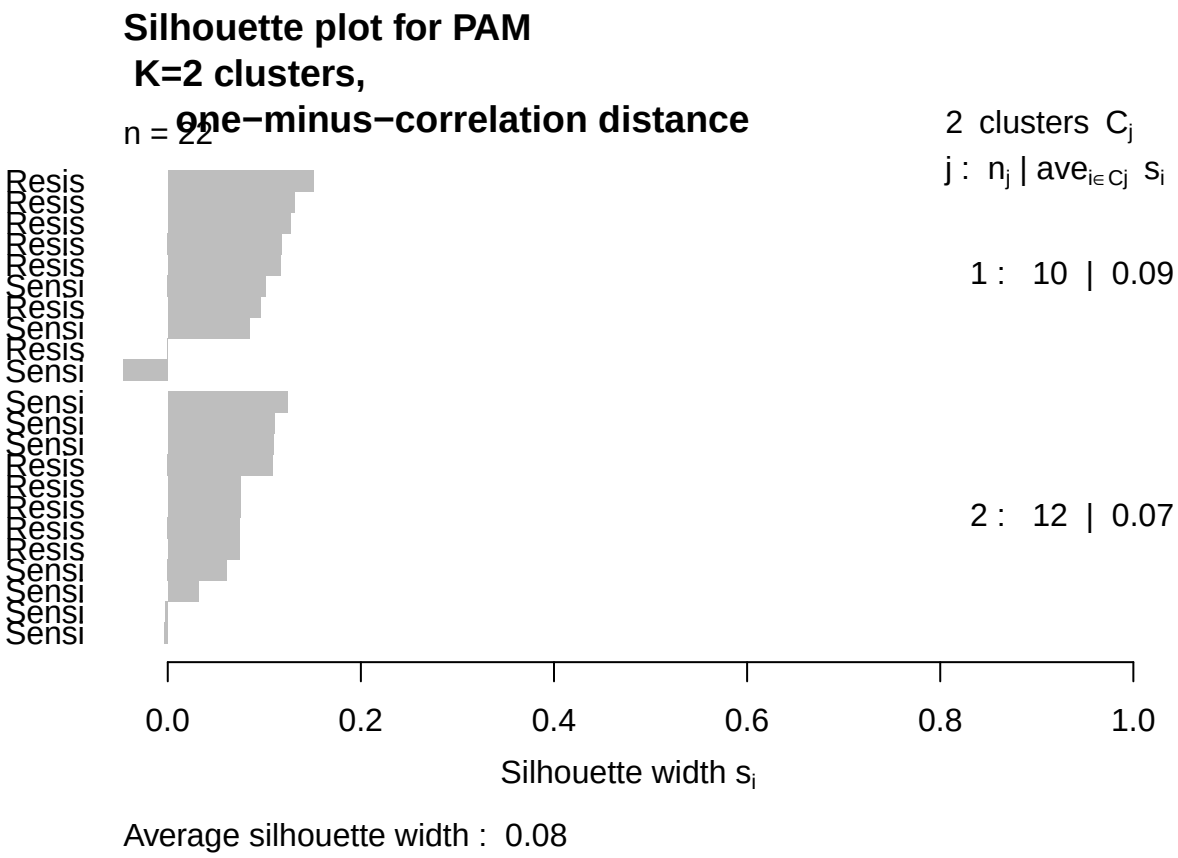
Partitioning Clustering

```
# One-minus-correlation distance matrix
r <- cor(exprSet.loess)
d <- 1-r

Y <- pData(ExprSet.NCI60.Scaled)[,1]

colG <- c("red", "blue")[factor(Y)]

dimnames(d) <- list(as.vector(Y),as.vector(Y))

# PAM, K=2
pam2 <- pam(as.dist(d), k=2, diss=TRUE)

# PAM, K=3
pam3 <- pam(as.dist(d), k=3, diss=TRUE)

# Graphical summaries
```

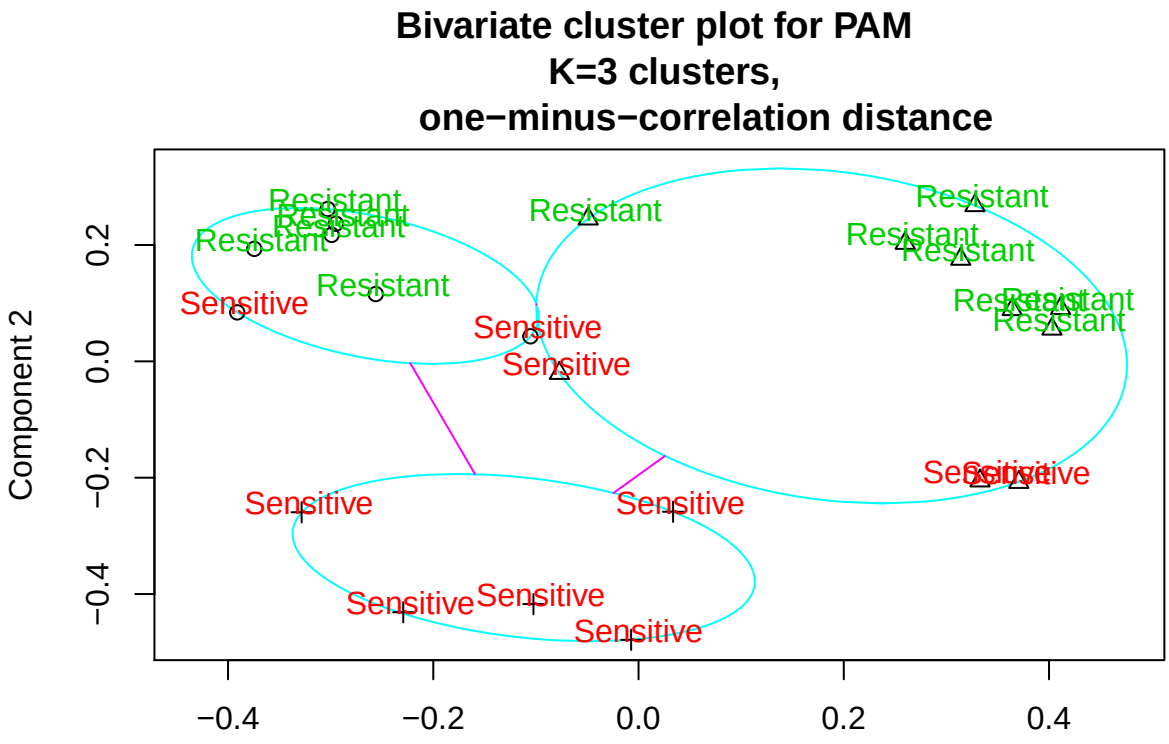
```
clusplot(d, pam2$clustering, diss=TRUE, labels=3, col.p=1,
         col.txt=rank(unique(Y))[factor(Y)]+1,
         main="Bivariate cluster plot for PAM \n K=2 clusters,
         one-minus-correlation distance")
```



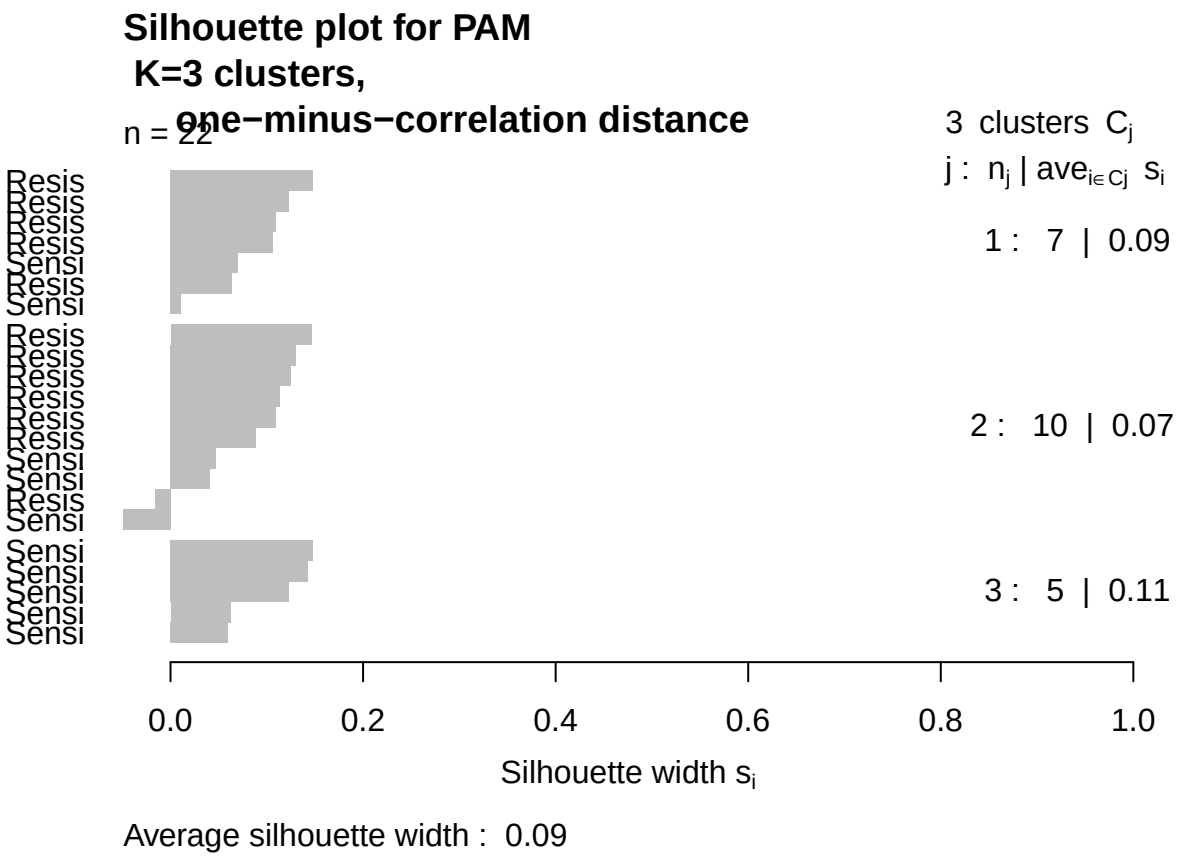

```
plot(pam2,which.plots=2,main="Silhouette plot for PAM \n K=2 clusters,
     one-minus-correlation distance")
```




```
clusplot(d, pam3$clustering, diss=TRUE, labels=3, col.p=1,
         col.txt=rank(unique(Y))[factor(Y)]+1,
         main="Bivariate cluster plot for PAM \n K=3 clusters,
         one-minus-correlation distance")
```




```
plot(pam3,which.plots=2,main="Silhouette plot for PAM \n K=3 clusters,
     one-minus-correlation distance")
```




```
## ----pam2----------------------------------------------------------------
table(pam2$clustering, Y)
```

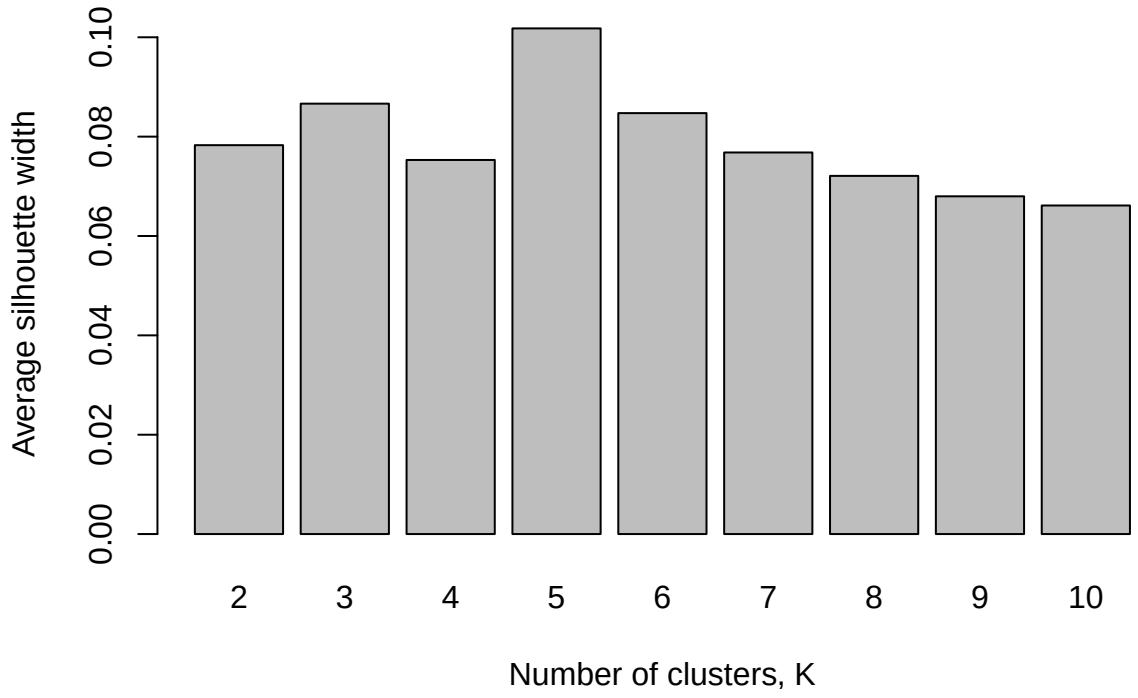
```
##    Y
##     Resistant Sensitive
##   1         7         3
##   2         5         7
```

```
## ----pam3----------------------------------------------------------------
table(pam3$clustering, Y)
```

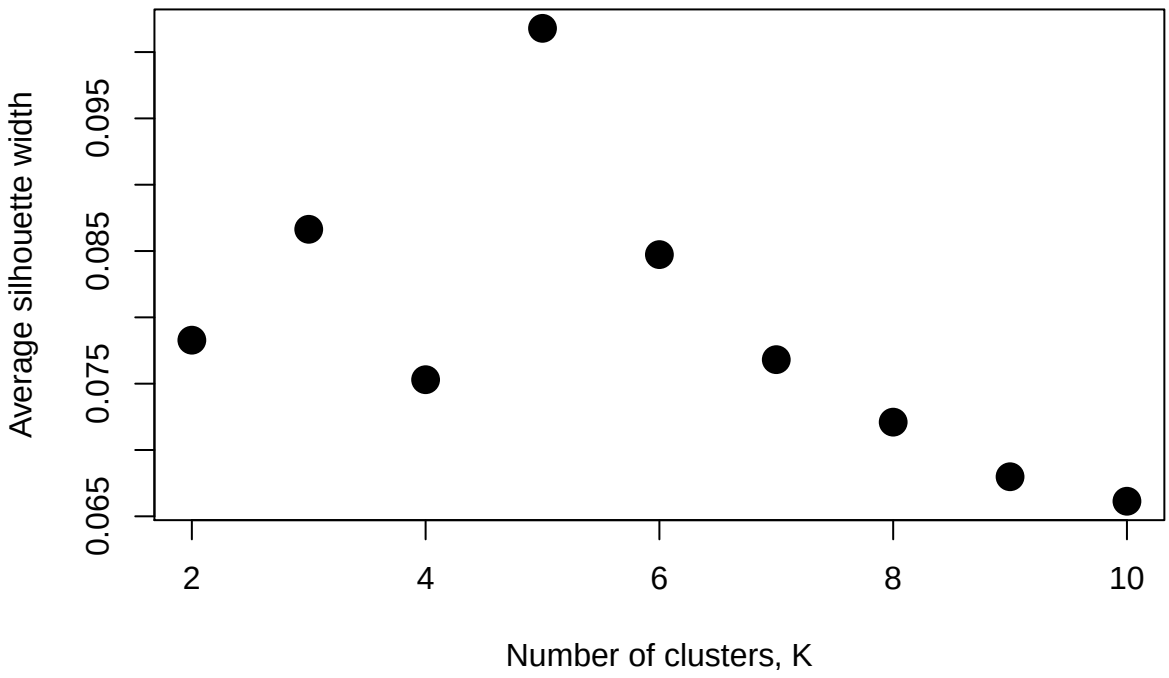
```
##    Y
##     Resistant Sensitive
##   1         5         2
##   2         7         3
##   3         0         5
```

```
## ----pamSil--------------------------------------------------------------
# Average silhouette widths for PAM with K = 2, ...,10 clusters
K <- 2:10
avgSil <- rep(NA, length(K))
names(avgSil) <- K
for(k in K)
  avgSil[k-1] <- pam(as.dist(d), k=k, diss=TRUE)$silinfo$avg.width
```

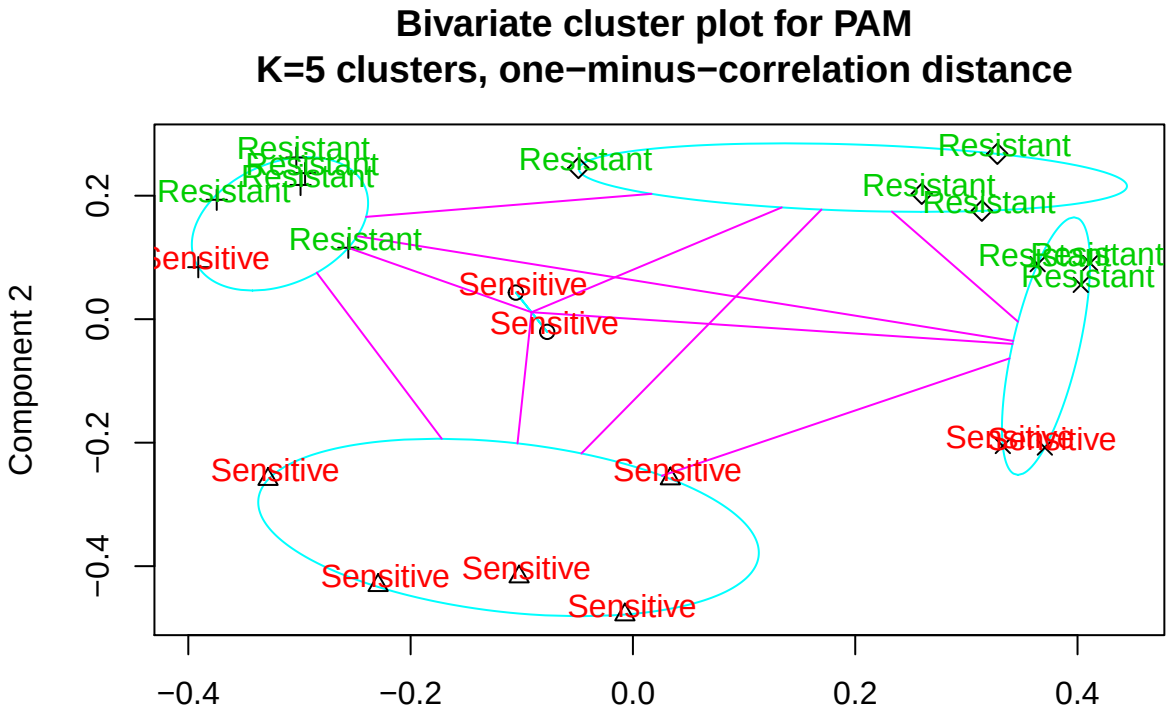
```r
# Graphical summaries
barplot(avgSil, names.arg=K, xlab="Number of clusters, K",
        ylab="Average silhouette width")
```



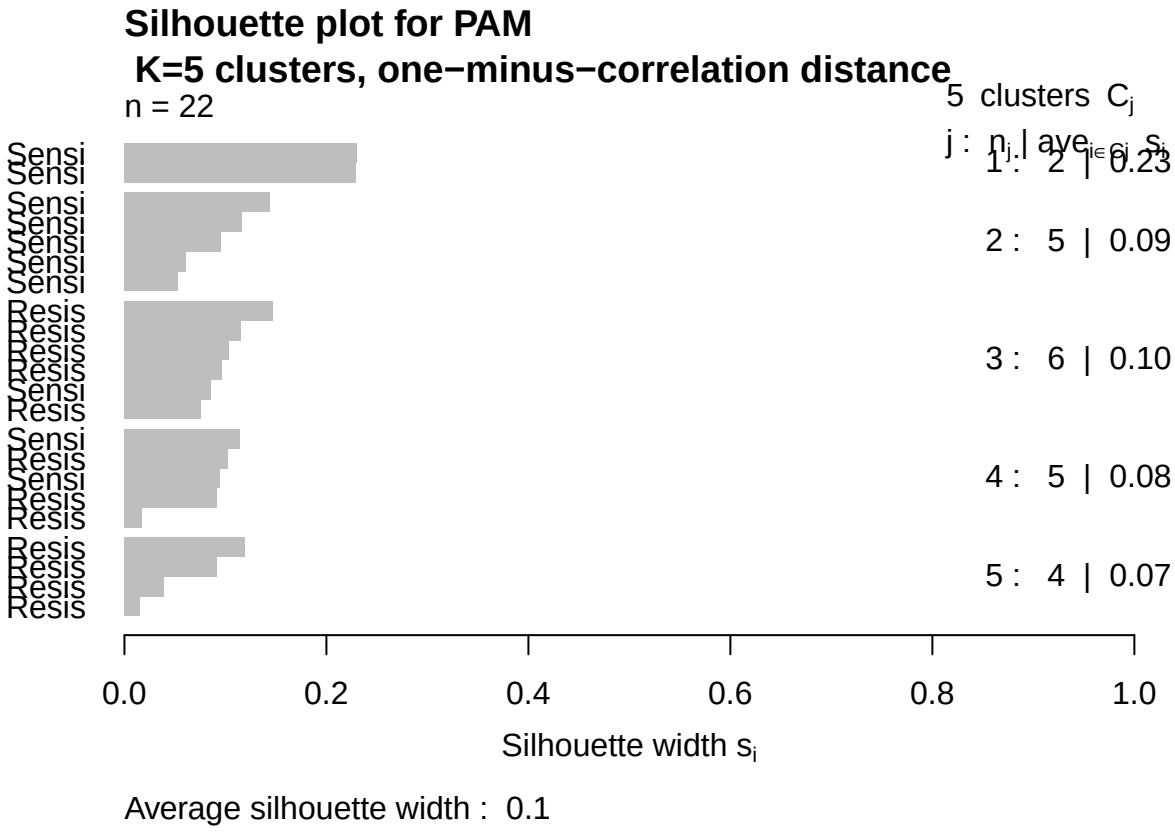

```r
plot(K, avgSil, pch=16, cex=2, xlab="Number of clusters, K",
     ylab="Average silhouette width")
```




```
## ----pamGraphSum-----------------------------------------------------------
# PAM, K=5
pam5 <- pam(as.dist(d), k=5, diss=TRUE)

# Graphical summaries
clusplot(d, pam5$clustering, diss=TRUE, labels=3, col.p=1, col.txt=rank(unique(Y))[factor(Y)]+1, main="
```




```
plot(pam5,which.plots=2,main="Silhouette plot for PAM \n K=5 clusters, one-minus-correlation distance")
```




```
## ----avgSil----------------------------------------------------------------
round(avgSil,3)
```

```
##     2     3     4     5     6     7     8     9    10
## 0.078 0.087 0.075 0.102 0.085 0.077 0.072 0.068 0.066
```

```
K[which.max(avgSil)]
```

```
## [1] 5
```

```
## ----pam5------------------------------------------------------------------
table(pam5$clustering, Y)
```

```
##    Y
##     Resistant Sensitive
##   1         0         2
##   2         0         5
##   3         5         1
##   4         3         2
##   5         4         0
```

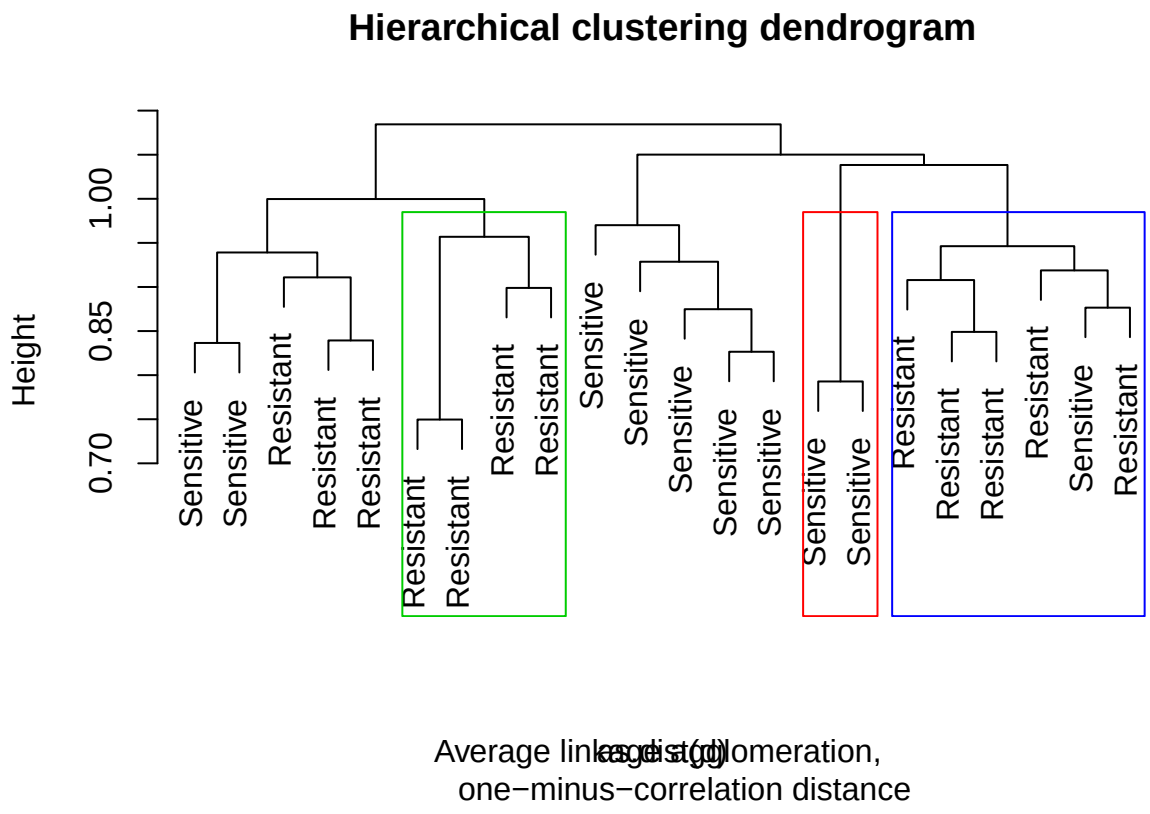
Clustering with PAM shows that the data does not fit well using the PAM clustering method. K = 5 has highest average silhouette width, but it's still not a good fit as the value is pretty close to zero.

Hierarchical Clustering

```
## ----hclust----------------------------------------------------------------
# One-minus-correlation distance matrix
r <- cor(exprSet.loess)
d <- 1-r
dimnames(d) <- list(as.vector(Y),as.vector(Y))

# Average linkage agglomerative hierarchical clustering
hc <- hclust(as.dist(d), method="average")
hc
```

```
##
## Call:
## hclust(d = as.dist(d), method = "average")
##
## Cluster method   : average
## Number of objects: 22
```

```
# Dendrogram
plot(hc, labels=Y, main="Hierarchical clustering dendrogram",
     sub="Average linkage agglomeration,
     one-minus-correlation distance")
rect.hclust(hc, k=5, which=c(2,4,5),border=c(3,2,4))
```



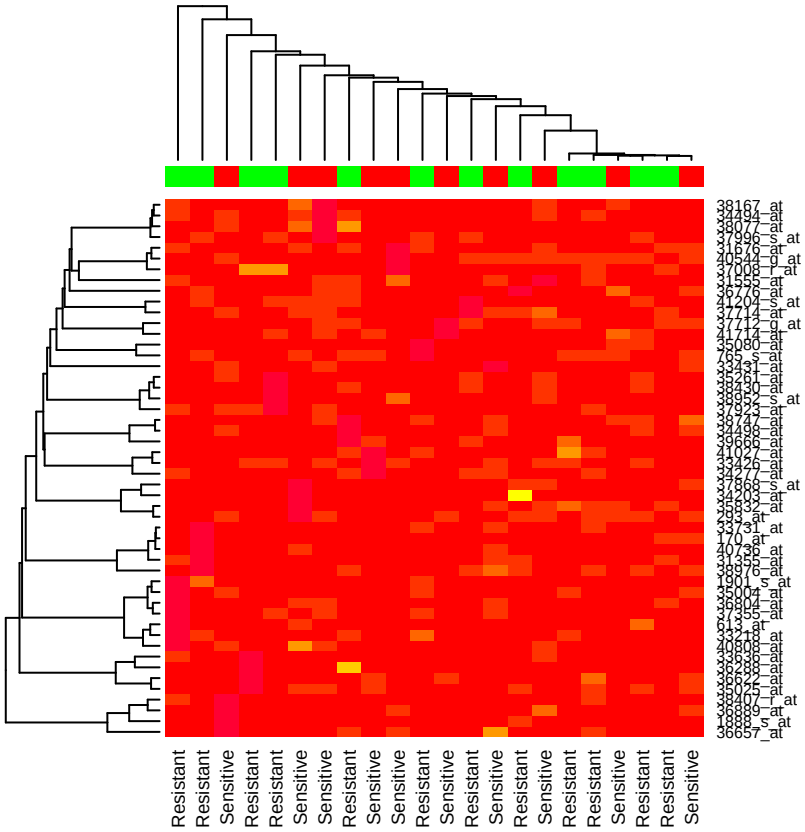
```
## ----cophenetic-----------------------------------------------------------
round(cor(cophenetic(hc),as.dist(d)),2)
```

```
## [1] 0.83
```

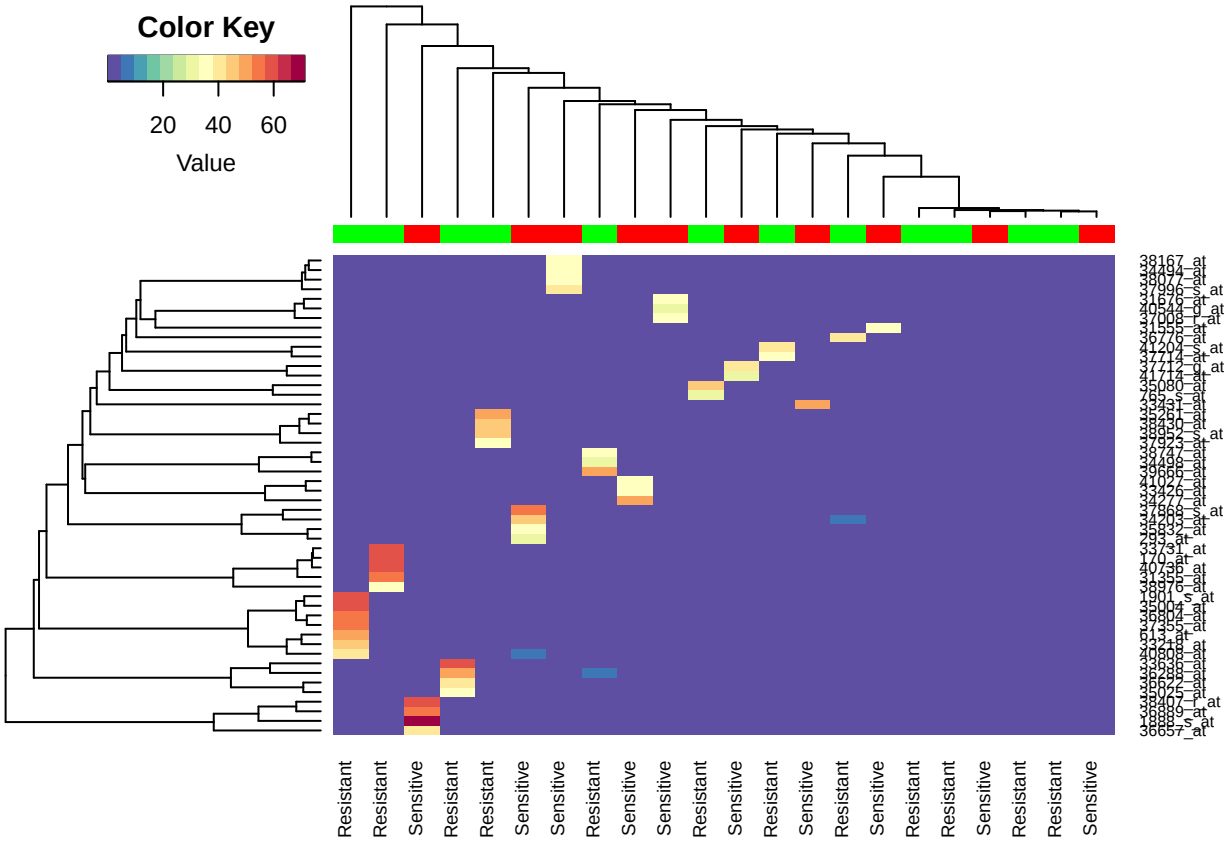
```
## ----cutree---------------------------------------------------------------
table(cutree(hc,5),Y)
```

```
##    Y
##     Resistant Sensitive
##   1         0         2
##   2         0         5
##   3         5         1
##   4         3         2
##   5         4         0
```

```
## ----hclustDendro---------------------------------------------------------
# Select the 50 probes with the largest absolute coefficients of variation
X <- exprSet.loess
cv <- apply(X,1, function(z) abs(sd(z)/mean(z)))
Xtop <- X[rev(order(cv))[1:50],]
dimnames(Xtop)[[2]] <- Y

# Heatmaps

myPalette <- colorRampPalette(rev(brewer.pal(11, "Spectral")))


heatmap(Xtop,col=rainbow(30),
        ColSideColors=c("red","green","blue")[rank(unique(Y))
                                              [factor(Y)]])
```




```
heatmap.2(Xtop,
          ColSideColors=c("red","green","blue")
          [rank(unique(Y))[factor(Y)]],
          labCol=colnames(Xtop),
          trace="none", key=TRUE, density.info="none",
          col=myPalette)
```




```
invisible(dev.off())
```

The agglomerative hierarchical cluster analysis reveals that the clustering is a good fit as revealed by the cophenetic correlation coefficient. The heatmaps reveal the different levels of gene expressions by the cells. The dendrograms were not able to classify the resistant and the sensitive cell lines into two clear grous. The gene expression patterns revealed by the heat maps do not suggest a clear distinction between the resistant cell lines and the sensitive cell lines.

c. Differential expression analysis

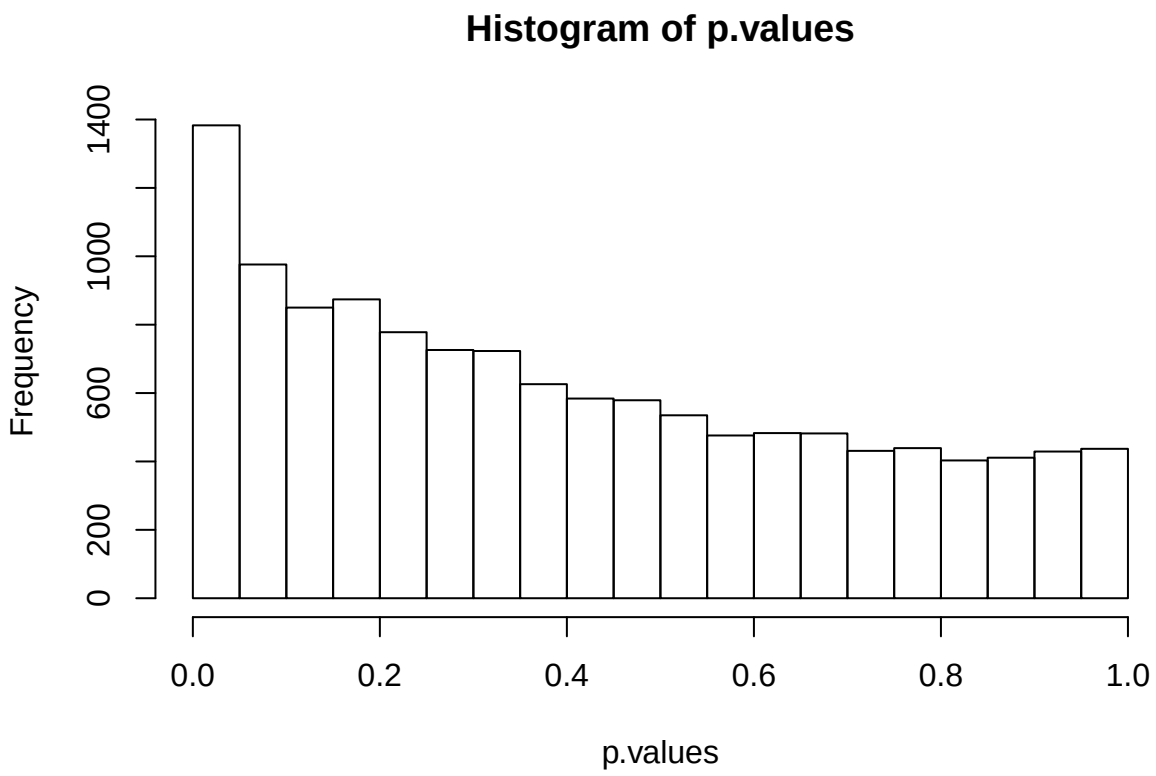
I use the two-sample t-test to see if the gene expressions between the resistant cell lines and the sensitive cell lines are different. The assumption I am making is that the gene expressions are indepent and normally distributed among the cell lines. Also, another assumption I am making is that the two cell groups, resistant and sensitive, are independent of each other. I am also assuming equal variance of gene expression levels between the two groups for each of the genes.

```
ttest <- matrix(data = NA,
                nrow = nrow(exprSet.quantile),
                ncol = 2)

for(i in 1:nrow(exprSet.quantile)){
  test <- t.test(exprSet.quantile[i, 1:10], exprSet.quantile[i, 11:22])
  ttest[i, ] <- c(test$statistic, test$p.value)
}
```

```r
dim(ttest)
```

```
## [1] 12625     2
```

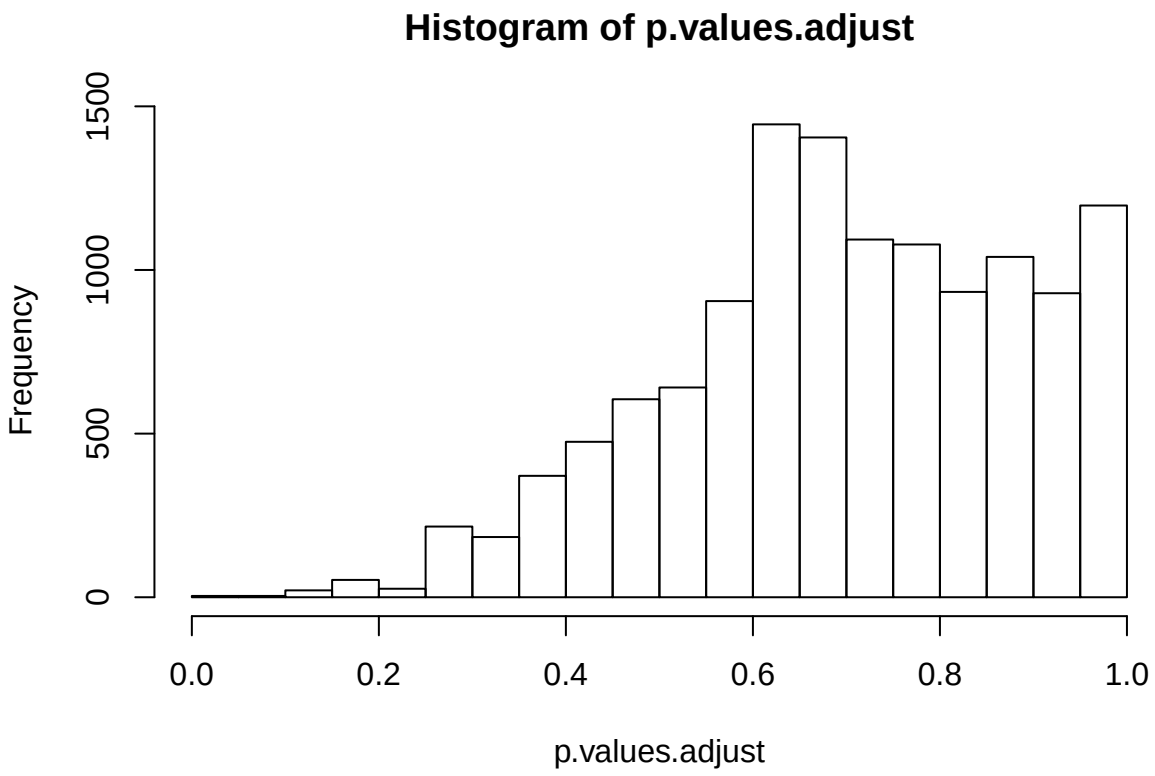
```r
colnames(ttest) <- c('T.stat', 'p.value')

#adjustments
ttest[is.na(ttest[,2]),2] <- 1 #replace all rows where p-value is NA is 1.

p.values <- ttest[,2]
hist(p.values)
```




```r
sum(p.values < 0.05)
```

```
## [1] 1383
```

```r
p.values.adjust <- p.adjust(p.values, method = 'fdr') #false decovery rate
hist(p.values.adjust)
```




```
sum(p.values.adjust < 0.05)
```

```
## [1] 4
```

```
rownames(exprSet.quantile[which(p.values.adjust < 0.05),])
```

```
## [1] "33824_at" "34213_at" "35766_at" "36133_at"
```

After adjusting for the false discovery rate, the four genes that are differentially expressed in the resistant and the sensitive cell lines are 33824_at, 34213_at, 35766_at, 36133_at.