Machine Learning

John Hamre III

August 11, 2017

Overview

In this presentation I show some of the ways computers can "learn" or be trained, and then predict classifications based off of training sets.

Bullet Points

- To begin with machine learning I start with knn prediction. It is a powerful and straight forward algorithm to use in R
- PCA clustering is demonstrated next, and is more complicated because it uses eigen values to project multi-dimensional data into two dimensions
- Next, we go into tree-based classification methods of machine learning by looking at rpart and random forest algorithms
- After an important gene or factor is found it can be used as a measure of prediction -Finally, I end with clustering as a means of prediction by using k-means and PAM clustering

k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression (wikipedia)

```
library(HiDimDA)
data("AlonDS")
library(class)

set.seed(1234)
ind <- sample(2, nrow(AlonDS), replace=TRUE, prob=c(.39, 0.61))
apo.training <- AlonDS[ind==1 , 2:2001]
apo.test <- AlonDS[ind==2 , 2:2001]
apo.trainlabels <- AlonDS[ind==1, 1]
apo.testlabels <- AlonDS[ind==2, 1]
apo_pred <- knn(apo.training, apo.test, apo.trainlabels, k=1)
table(apo_pred, apo.testlabels)
```

```
##          apo.testlabels
## apo_pred  colonc healthy
##   colonc      25       2
##   healthy      2      11
```

PCA dimension reduction

```
alon.data <- (AlonDS[1:dim(AlonDS)[1], 2:dim(AlonDS)[2]])
alon.names <- read.csv("data/alon.names.csv", header=F)
Alon.cl <- c(0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,1,1,0,0,1,1,0,0,0,1,0,1)
AlonFactor <- factor(Alon.cl,levels=0:1, labels= c("colonc","healthy"))
talon.data <- t(alon.data)
eigen(cor(talon.data))$values[1:5]
```

```
## [1] 46.266564  2.831612  2.264533  1.758461  1.151688
```

```
data <- talon.data; p <- ncol(data); n <- nrow(data) ; nboot<-1000
eigenvalues <- array(dim=c(nboot,p))
for (i in 1:nboot) {
    dat.star <- data[sample(1:n,replace=TRUE),]
    eigenvalues[i,] <- eigen(cor(dat.star))$values
}
for (j in 1:5) {
    print(quantile(eigenvalues[,j],c(0.025,0.975)))
}
```

```
##     2.5%    97.5%
## 44.36836 48.28612
##     2.5%    97.5%
## 2.450823 3.759073
##     2.5%    97.5%
## 1.785127 2.827118
##     2.5%    97.5%
## 1.382515 2.014506
##     2.5%    97.5%
## 1.011605 1.421319
```

PCA results

```
sum(eigen(cor(talon.data))$values[1:5])/62*100
```

```
## [1] 87.53687
```

```
#Thus, the first five components represent more than 87% of the variance in the data. The data can allow for a reduction in dimensions from
#sixty two data points to five, while still capturing over 87% of the total variance

# to verify that the correlations between the patients are positive
# we run;
-eigen(cor(talon.data))$vec[,1:5]
```

```
##             [,1]        [,2]        [,3]        [,4]        [,5]
##  [1,] 0.13334450  0.033986091  0.064299240  0.190182448  0.087596270
##  [2,] 0.13101243  0.055453409 -0.160080373  0.192870293  0.065330616
##  [3,] 0.08123138  0.381194248  0.254511766 -0.158840079  0.066887819
##  [4,] 0.12186663  0.187345101  0.024326909  0.076170309  0.227739581
##  [5,] 0.13492308 -0.025234418  0.064778140 -0.131490100  0.106656352
##  [6,] 0.12573668 -0.042508361 -0.162798987 -0.243283225  0.205817286
##  [7,] 0.13142976 -0.067702980  0.063955075  0.139911681  0.108777444
##  [8,] 0.11497311 -0.040887571 -0.147787136 -0.086017008  0.171674091
##  [9,] 0.12035762 -0.123376691  0.127601183  0.018987419  0.194064424
## [10,] 0.12759766  0.073671501 -0.135938797 -0.112838720  0.114489142
## [11,] 0.11994081 -0.165215734  0.028956883  0.037587940 -0.130940977
## [12,] 0.13114523  0.087751355 -0.077048073  0.079323831  0.095345885
## [13,] 0.13671340 -0.055573830  0.010252231 -0.021086622  0.114873382
## [14,] 0.11569421  0.038688741 -0.049658214 -0.029725807  0.199764625
## [15,] 0.13809274  0.001120874  0.094184776 -0.103703484  0.063806124
## [16,] 0.12442516 -0.025277073  0.134594415  0.034626258  0.109465939
## [17,] 0.12394763 -0.160422616  0.109688079 -0.003395834  0.039133282
## [18,] 0.12247192 -0.115504816 -0.118420118 -0.070117793  0.227967374
## [19,] 0.12982113 -0.162356072  0.079103129 -0.044542409  0.158668333
## [20,] 0.12278348 -0.095501004 -0.195402405 -0.210246203  0.170166072
## [21,] 0.12798672 -0.044834231  0.029938642  0.227943617  0.092220435
## [22,] 0.13003189  0.176046859 -0.094013733  0.070443325  0.068489667
## [23,] 0.11531068 -0.204686374  0.100432637 -0.204685846  0.019681785
## [24,] 0.12168741 -0.095188189 -0.082774295 -0.091211027  0.282243516
## [25,] 0.13290451  0.029251113  0.168042056 -0.160266523  0.014045905
## [26,] 0.13323856  0.019802248  0.106565887  0.165275129 -0.058442060
## [27,] 0.13511859 -0.020930960  0.000432636  0.230172036 -0.041733403
## [28,] 0.13085940 -0.086379885  0.094902516  0.174007957  0.054348256
## [29,] 0.13023929  0.024270947  0.054169492  0.263010813 -0.022021684
## [30,] 0.12650756 -0.107720674  0.015253178  0.206960704 -0.067399870
## [31,] 0.12657652 -0.013398653  0.006917491  0.150793132  0.038063135
## [32,] 0.12781952 -0.152868743 -0.082453566 -0.014880923 -0.164087648
## [33,] 0.13120837 -0.199350101  0.028402632 -0.036243480 -0.119104805
## [34,] 0.13165334  0.095107519  0.173911064 -0.079763756  0.024130942
## [35,] 0.13654398  0.006862681  0.033967618  0.105180091 -0.038677095
```

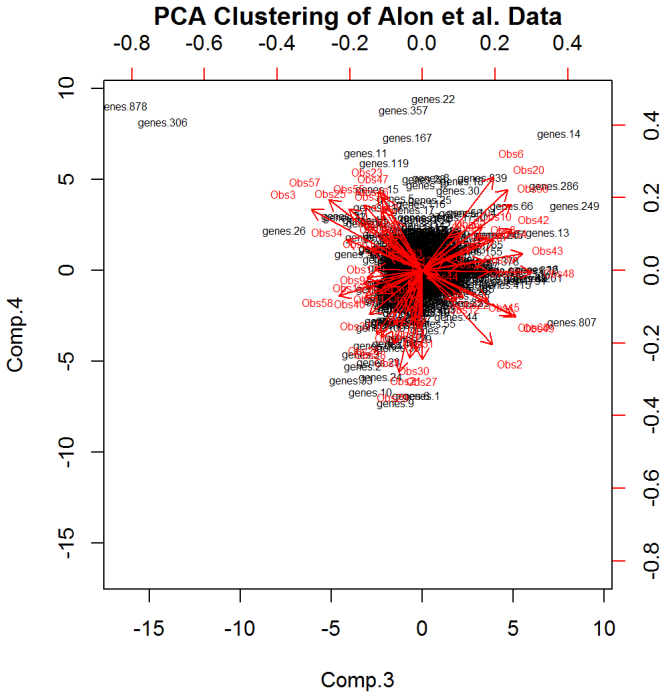
```
## [36,] 0.12753263 -0.176498716  0.094728221 -0.155394342 -0.106052817
## [37,] 0.12499657  0.091633181  0.121564126  0.113979885  0.154206875
## [38,] 0.12142376 -0.247474501  0.086513377 -0.086719621 -0.098649101
## [39,] 0.12972644  0.140905505 -0.141355244 -0.024807465 -0.007195856
## [40,] 0.13562229 -0.011701548  0.132153449  0.068448400 -0.122473596
## [41,] 0.12758597  0.138075514  0.097537111  0.059408314  0.058157902
## [42,] 0.12805261  0.038792278 -0.203808899 -0.105816551 -0.004528208
## [43,] 0.12695986  0.048085391 -0.229061725 -0.042818861 -0.186404835
## [44,] 0.12507601 -0.189008038 -0.037042016  0.011765187 -0.207960639
## [45,] 0.13269938  0.093324863 -0.149496926  0.075425565 -0.122914086
## [46,] 0.13018857 -0.076919772  0.090216978 -0.165045860 -0.215177634
## [47,] 0.12702667 -0.032740376  0.090144639 -0.191153217 -0.166982357
## [48,] 0.12247626  0.110721335 -0.250108174  0.004394967 -0.095453279
## [49,] 0.12597808  0.092051645 -0.214253387  0.119008694 -0.150459561
## [50,] 0.13084612  0.055682312 -0.152138267  0.003286583 -0.100791706
## [51,] 0.13174309 -0.052137569  0.020586878  0.130532967 -0.103712971
## [52,] 0.13318200 -0.110529496  0.075826283 -0.092541590 -0.090512358
## [53,] 0.13503694 -0.023589364  0.032351753  0.056959693  0.039407886
## [54,] 0.12620290  0.081782916 -0.163850480 -0.077543180  0.059139961
## [55,] 0.11426325  0.236062064  0.133687846 -0.170587949 -0.203857916
## [56,] 0.12847582  0.049489774 -0.202409188 -0.171449349 -0.101526188
## [57,] 0.09379796  0.361373049  0.214588704 -0.183748694 -0.034864277
## [58,] 0.12571744  0.153957449  0.191561791  0.065713023 -0.038236070
## [59,] 0.13366113 -0.079822957  0.081939721  0.039871818 -0.090216999
## [60,] 0.12649362  0.126979038 -0.204223030  0.116561202 -0.153176973
## [61,] 0.12977076  0.089854430  0.116352577 -0.056596296 -0.177553088
## [62,] 0.13421142  0.097037064 -0.088053764 -0.098567643 -0.132623999
```

PCA results in five dimensions

```
#From the previous slide we can match up the negative and positive signs with what we actually observed and express as;
Alon.cl
```

```
##  [1] 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0
## [36] 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 1 0 0 1 1 0 0 0 0 1 0 1
```

```
#By matching these we find that the 3rd and 4th dimensions matched the real data best. We can plot these with a biplot to show vectors
biplot(princomp(data,cor=TRUE),choices = c(3,4), pc.biplot=TRUE,cex=0.5,expand=0.8, main = "PCA Clustering of Alon et al. Data")
```




```
#There is a clear seperation from the left and right in the biplot
```

PCA results; finding the top genes

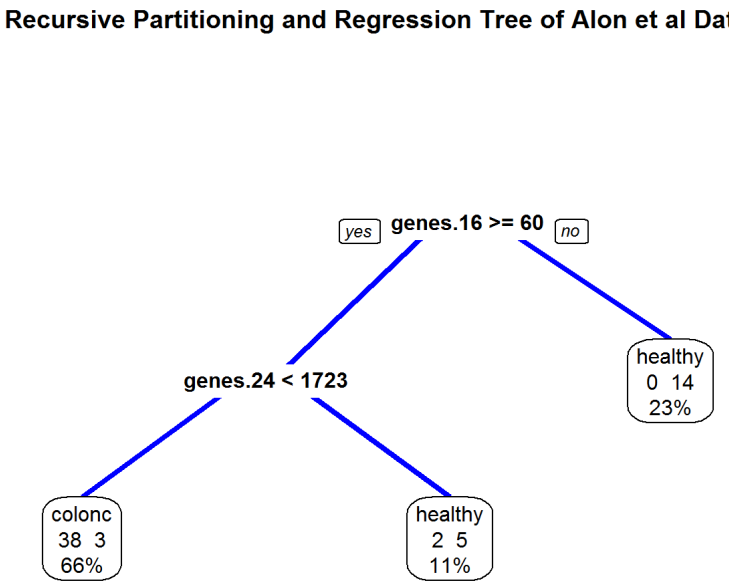
```
#No we can prints the first ten gene names with respect to the third principal comonent to find out what the main genes of differentiating diseased and healthy state
pca <- princomp(talon.data, center = TRUE, cor=TRUE, scores=TRUE)
o <- order(pca$scores[,3])
alon.names[o[1:10],6]
```

```
##  [1] IG GAMMA-1 CHAIN C REGION (HUMAN);.
##  [2] Human Ig gamma3 heavy chain disease OMM protein mRNA.
##  [3] 40S RIBOSOMAL PROTEIN S18 (Homo sapiens)
##  [4] LAMININ RECEPTOR (HUMAN);.
##  [5] 60S RIBOSOMAL PROTEIN L30E (Kluyveromyces lactis)
##  [6] 40S RIBOSOMAL PROTEIN S24 (HUMAN).
##  [7] P24050 40S RIBOSOMAL PROTEIN.
##  [8] EUKARYOTIC INITIATION FACTOR 4A (Oryctolagus cuniculus)
##  [9] EUKARYOTIC INITIATION FACTOR 4A (Oryctolagus cuniculus)
## [10] 40S RIBOSOMAL PROTEIN S6 (Nicotiana tabacum)
## 1824 Levels:  ...
```

```
pr.out =prcomp (alon.data , scale =TRUE)
```

rpart finds important genes with recursive partioning and regression to make predictions

```
library(rpart.plot)

row.names(talon.data)<- paste("gene", 1:2000, sep = "")

alonRpart <- rpart(AlonFactor~., data=alon.data, method="class", cp=0.001)
prp(alonRpart,  branch.lwd=4,  branch.col="blue",  extra=101, main="Recursive Partitioning and Regression Tree of Alon et al Data")
```




```
alon.names[ 16, 6]
```

```
## [1] Human tra1 mRNA for human homologue of murine tumor rejection antigen gp961
## 1824 Levels:  ...
```

```
alon.names[ 24, 6]
```

```
## [1] THYMOSIN BETA-4 (HUMAN);.
## 1824 Levels:  ...
```

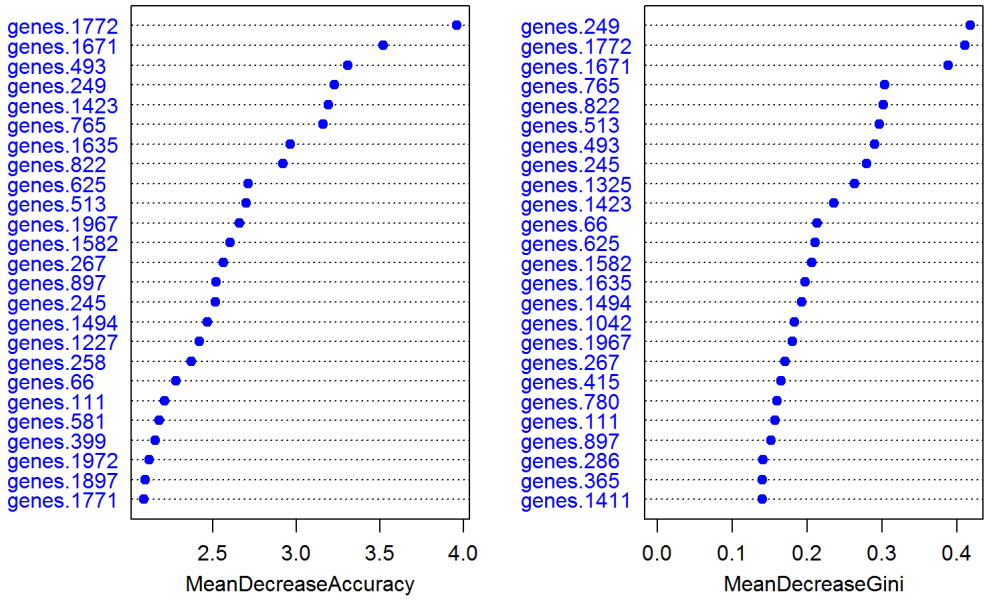
Random forests or random decision forests operate by constructing a multitude of decision trees at training time and outputting mean prediction (regression) of the individual trees. (wikipedia)

```
library(randomForest)
set.seed(1234)
alon.rf <- randomForest(grouping ~ ., data=AlonDS, importance=TRUE,
                    proximity=TRUE)

#We can predict the diseased state
 print(alon.rf)
```

```
##
## Call:
##  randomForest(formula = grouping ~ ., data = AlonDS, importance = TRUE,      proximity = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 44
##
##         OOB estimate of  error rate: 19.35%
## Confusion matrix:
##         colonc healthy class.error
## colonc      34       6   0.1500000
## healthy      6      16   0.2727273
```

Random forest accuracy measures by best gene

```
## Random forest plot of gene accuracy

 varImpPlot(alon.rf,
            n.var = 25,
            pch=19,
             main=NULL,
             col="blue",
             gcolor="blue",
             lcolor="black")
```




```
#Next we can find and print the genes that give the most accuracy
alon.names[249,6]
```

```
## [1] Human desmin gene, complete cds.
## 1824 Levels:  ...
```

```
alon.names[1772,6]
```

```
## [1] COLLAGEN ALPHA 2(XI) CHAIN (Homo sapiens)
## 1824 Levels:  ...
```

```
alon.names[1671,6]
```

```
## [1] Human monocyte-derived neutrophil-activating protein (MONAP) mRNA, complete cds.
## 1824 Levels:  ...
```

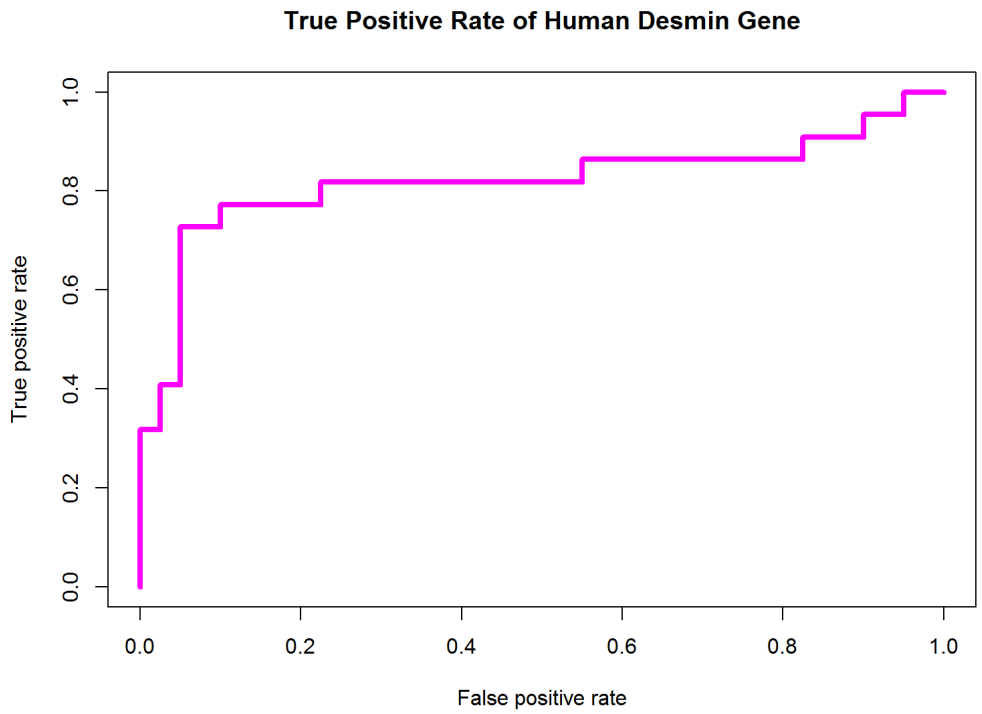
Desmin gene prediction of diagnosis by expression level

```
colon.data <- read.table("data/AlonGene.txt")
alonLabels<-factor(Alon.cl, levels = 0:1, labels=c("colonc","healthy"));desmin <- grep("Human desmin gene, complete cds",alon.names[,6], ignore.case = TRUE)
alonPredictor <- factor(colon.data[desmin,]> 1700,levels=c("FALSE","TRUE"), labels=c("colonc","healthy"))
table(alonPredictor,alonLabels)
```

```
##              alonLabels
## alonPredictor colonc healthy
##       colonc      38       6
##       healthy      2      16
```

We can also analyze a specific gene to measure its change of true positive or negative

```
library(ROCR)
alonLabels <- factor(Alon.cl,levels=0:1,labels= c("FALSE","TRUE"))
desmin <- grep("Human desmin gene, complete cds",alon.names[,6], ignore.case = TRUE)
  pred <- prediction(talon.data[desmin,], alonLabels)
  perf <- performance(pred, "tpr", "fpr" )
  plot(perf,main ="True Positive Rate of Human Desmin Gene",
         lwd=4,
        col="magenta")
```



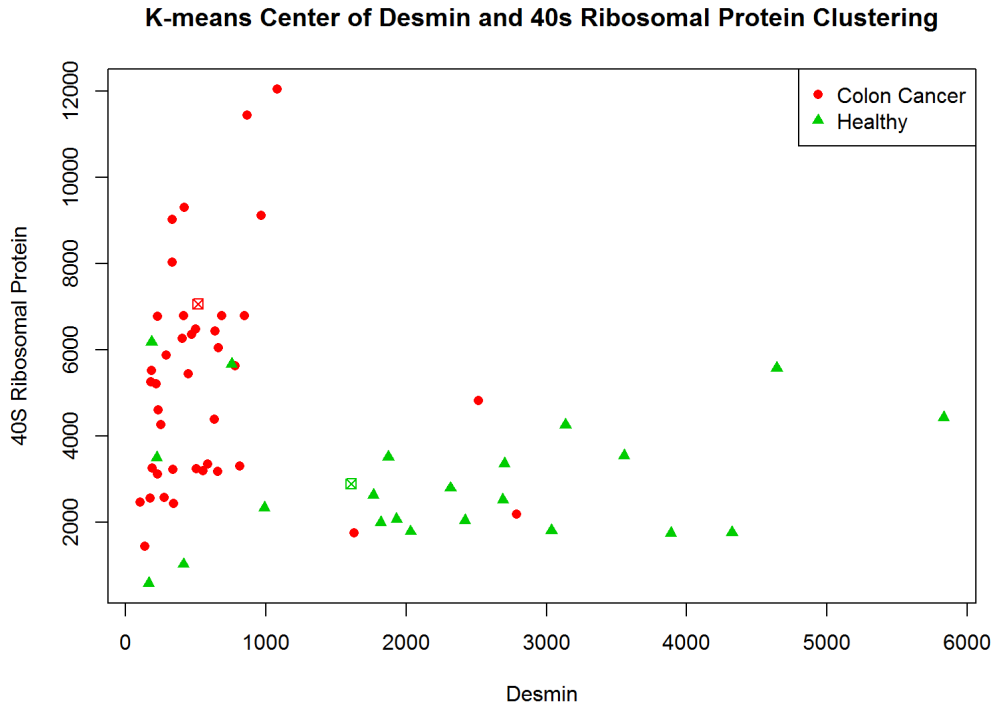

```
performance(pred,"auc")
```

```
## An object of class "performance"
## Slot "x.name":
## [1] "None"
```

```
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.8204545
##
##
## Slot "alpha.values":
## list()
```

K-means clustering is used after the top genes are found

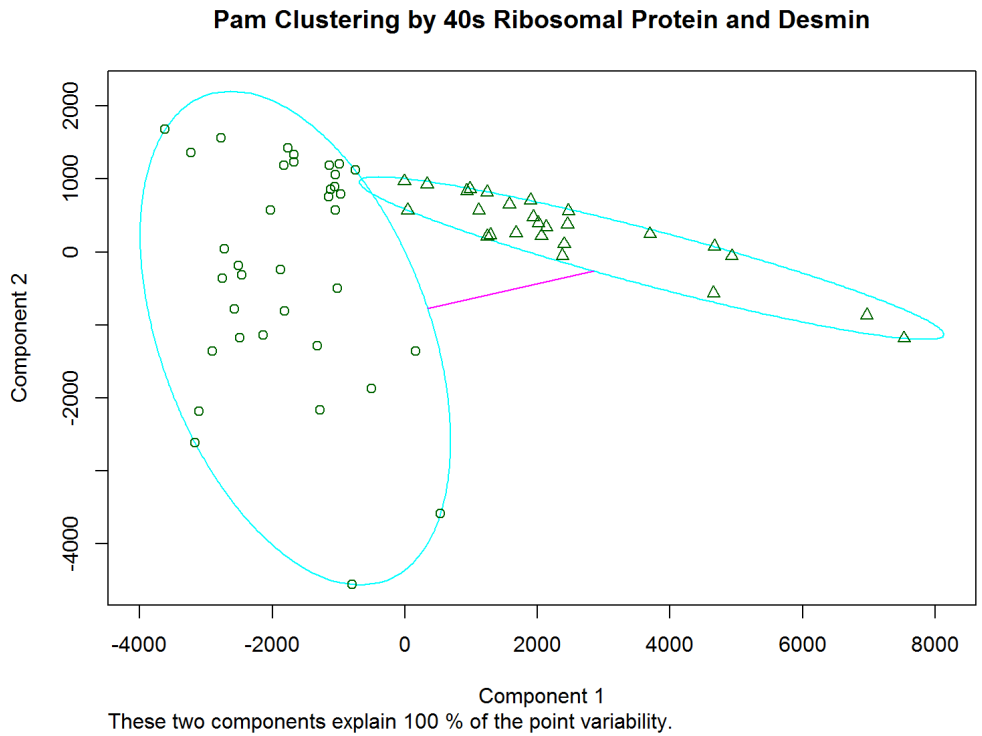
```
RibosomalProtein <- grep("40S RIBOSOMAL PROTEIN S18",alon.names[,6], ignore.case = TRUE)
Desmin <- grep("Human desmin gene, complete cds",alon.names[,6], ignore.case = TRUE)
clusdata <- data.frame(talon.data[Desmin,],talon.data[RibosomalProtein,])
colnames(clusdata)<- c("Desmin", "40S Ribosomal Protein")
cl <- kmeans(clusdata, 2, 10)
plot(clusdata,    pch=as.numeric(AlonFactor) + 15,   col=as.numeric(AlonFactor) + 1,main = "K-means Center of Desmin and 40s Ribosomal Protein Clustering")
legend("topright",
          legend=c("Colon Cancer","Healthy"),
          pch=16:17,
          col=c(2,3))
 points(cl$centers, col=c(3,2), pch = 7, lwd=1)
```



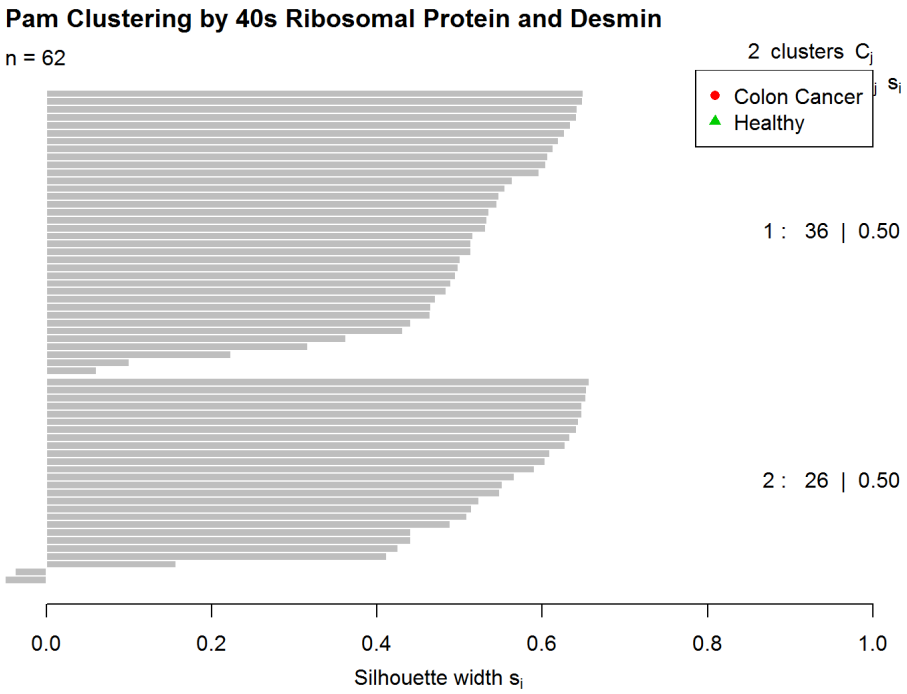
##Here I use PAM clustering with the genes 40s Ribosomal Protein and Desmin

```
##Desmin gene prediction of diagnosis by expression level
library(cluster)
```

```
pamx <- pam(clusdata, 2)
plot(pamx,main="Pam Clustering by 40s Ribosomal Protein and Desmin")
```




```
legend("topright", legend=c("Colon Cancer","Healthy"), pch=16:17, col=c(2,3))
```


Pam Clustering by 40s Ribosomal Protein and Desmin
n = 62
2 clusters C_j
s_i
Colon Cancer
Healthy
1 : 36 | 0.50
2 : 26 | 0.50
0.0
0.2
0.4
0.6
0.8
1.0
Silhouette width s_i
Average silhouette width : 0.5