Unsupervised Classification

(Cluster Analysis Group the observations into *k* distinct natural groups)

Hierarchical clustering:

We have a dataset with *n* observations and we want to group the observations into *k* distinct natural groups of similar observations.

We distinguish three stages of cluster analysis:

Input Stage
Algorithm stage
Output stage

Input Stage

1. Scaling:

a) Divide variables by the standard deviation.

b) Spherize the data: Invariance under afine transformations.

$Z = A Y$; $A = \text{Chol}(S)^{-1}$ or the symmetric square root $S^{-1/2}$;

c) Spherize the data with the within variance.

$T = W + B$

To obtain W use iteration.

Hierarchical clustering

2. Similarity and dissimilarity measures.

Clustering methods require the definition of a similarity or dissimilarity measure. For example an inter-point distance d(x1,x2) and an inter-cluster distance d*(C1,C2) are examples of dissimilarity.

The inter point distance is often taken to be the Euclidean distance or Mahalanobis distance. Some times we may use the Manhattan distance.

When the data is not metric we may define any distance or similarity measure from characteristics of the problem. For example for binary data given any two vector observations we construct the table

	1	0	Total
1	a	b	a+b
0	c	d	c+d
Total	A+c	b+d	P

Then we define distance as the square root of the χ^2 statistic.

Also d = 1 - (a+d)/p or d = 1- a/(a+b+c)

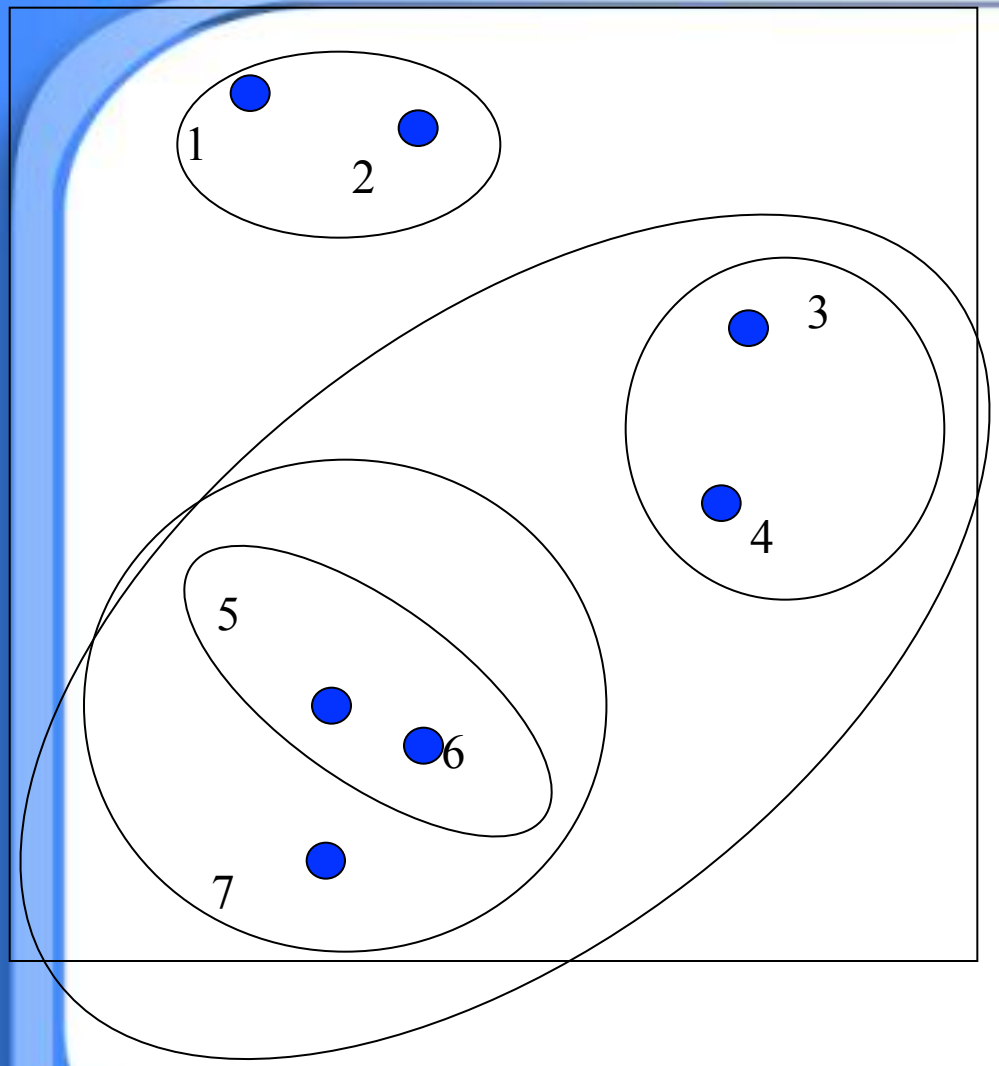
Hierarchical clustering

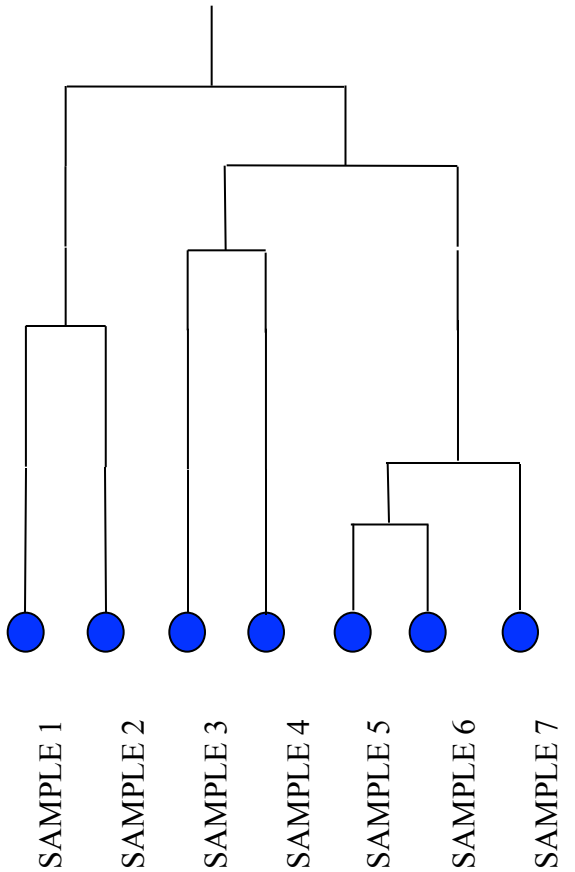
Build a hierarchical tree

- Inter point distance is normally the Euclidean distance
  (some times we may use Manhattan distance).

- Inter cluster distance:
  - Single Linkage: distance between the closes two points
  - Complete Linkage: distance between the furthest two points
  - Average Linkage: Average distance between every pair of points
  - Ward: R^2 change.
- Build a hierarchical tree:
  1. Start with a cluster at each sample point
  2. At each stage of building the tree the two closest clusters joint to form a new cluster.

Hierarchical Cluster Example





Hierarchical clustering: Ward's method

At any stage we construct the dissimilarity matrix (or distance matrix) reflecting all the inter-cluster distances between any pair of categories.

We build a hierarchical tree starting with a cluster at each sample point, and at each stage of the tree

Build dissimilarity matrix

The two closest clusters joint to form a new cluster.

Once we finish building the tree the question becomes: "how many clusters do we chose?"

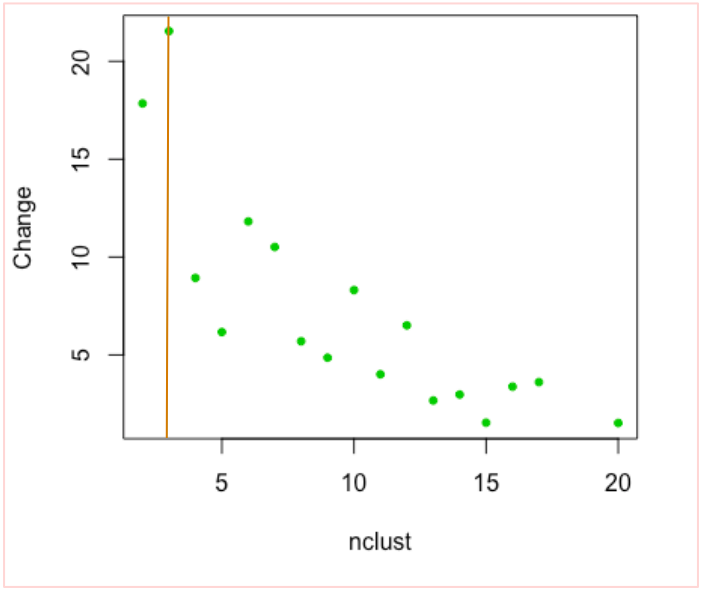
One way of making this determination is by inspecting the hierarchical tree and finding a reasonable point to break the clusters. We can also plot the criteria function for the different number of cluster and visually look for unusually large jumps. In the example below with *WARD's* clustering method we stop at the first place where the R^2 change (percent-wise) is large.

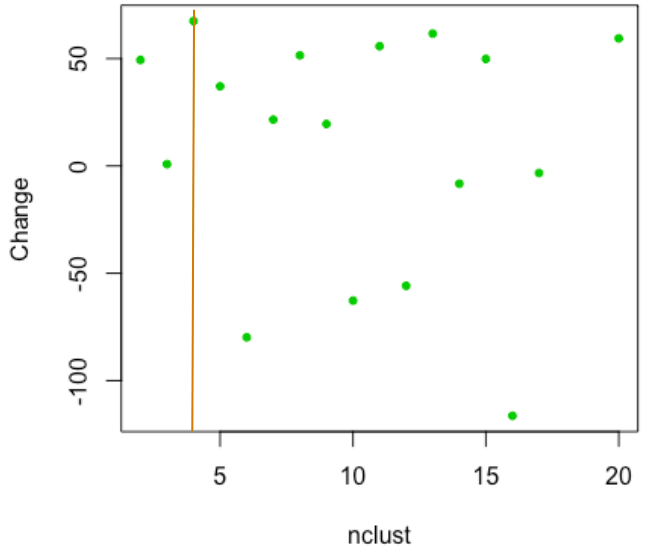
10	CL45	CL15	24	0.008555	0.824891
9	CL25	CL16	84	0.009749	0.815142
8	CL23	CL13	49	0.009836	0.805306
7	CL8	CL22	67	**0.009713**	0.795593
6	CL17	CL11	134	**0.037362**	0.758231
5	CL9	CL14	102	0.037383	0.720848

Determine the number of clusters

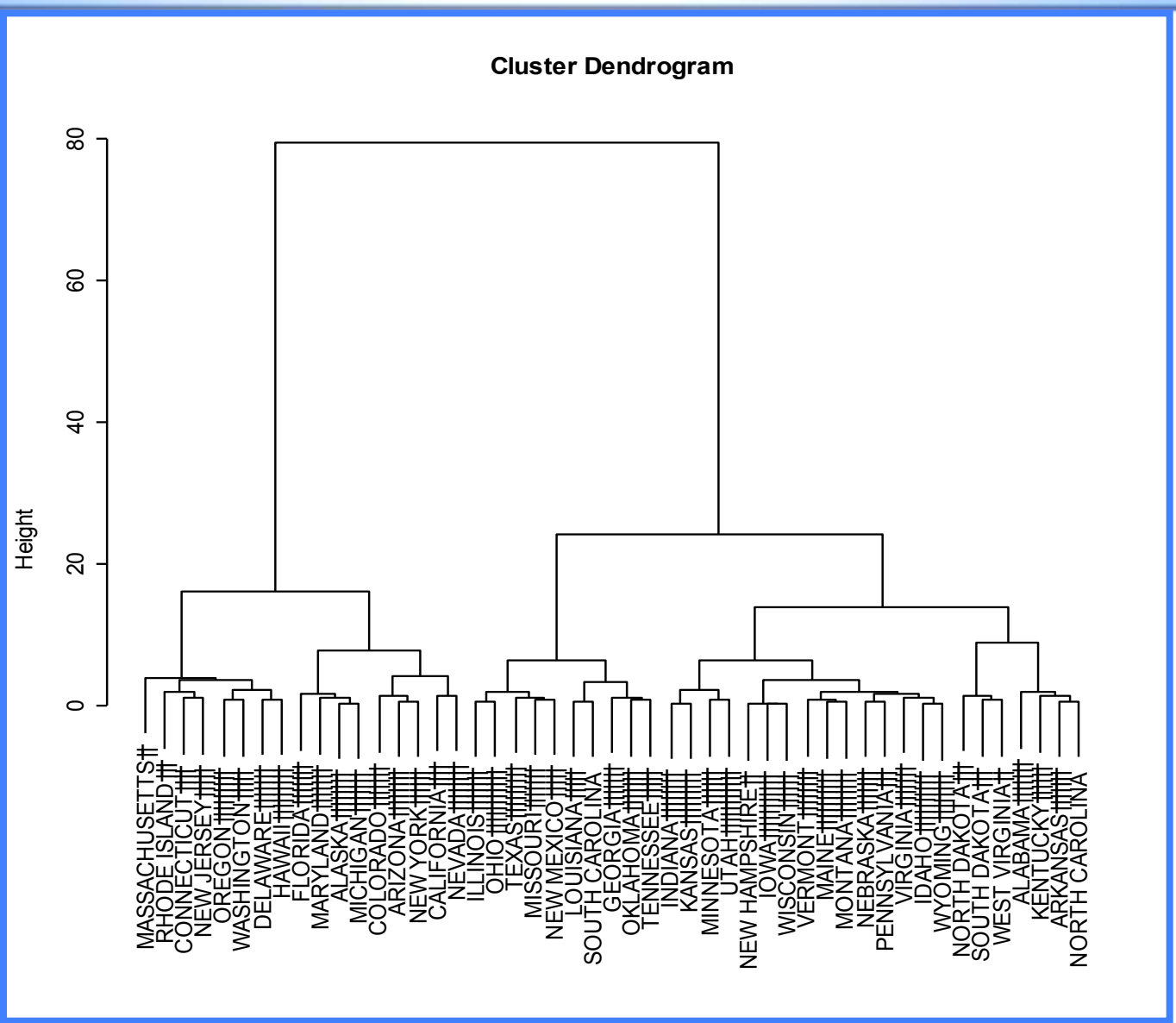
This function uses the change in percent in the second difference or the first difference

```
clust_sel = function(x,y,jrange=3:25,dd=2) {
  ## x is an array,              ##  jrange n of clusters to be checked
  ## y is an hclust object       ##  dd number of differences
wss4 = function(x,y,w = rep(1, length(y))) sum(lm(x~factor(y),weights = w)
$resid^2*w)
### wss4 calculates within cluster sum of squares
sm1 = NULL
for(i in jrange) sm1[i] = wss4(x,cutree(y,i))
sm1=sm1[jrange]
k = if(dd==1) sm1[-1] else -diff(sm1)
plot(jrange[-length(k)+1:0], -diff(k)/k[-length(k)]*100)
jrange [sort.list(diff(k)/k[-length(k)]*100)[1:4]]
}
```

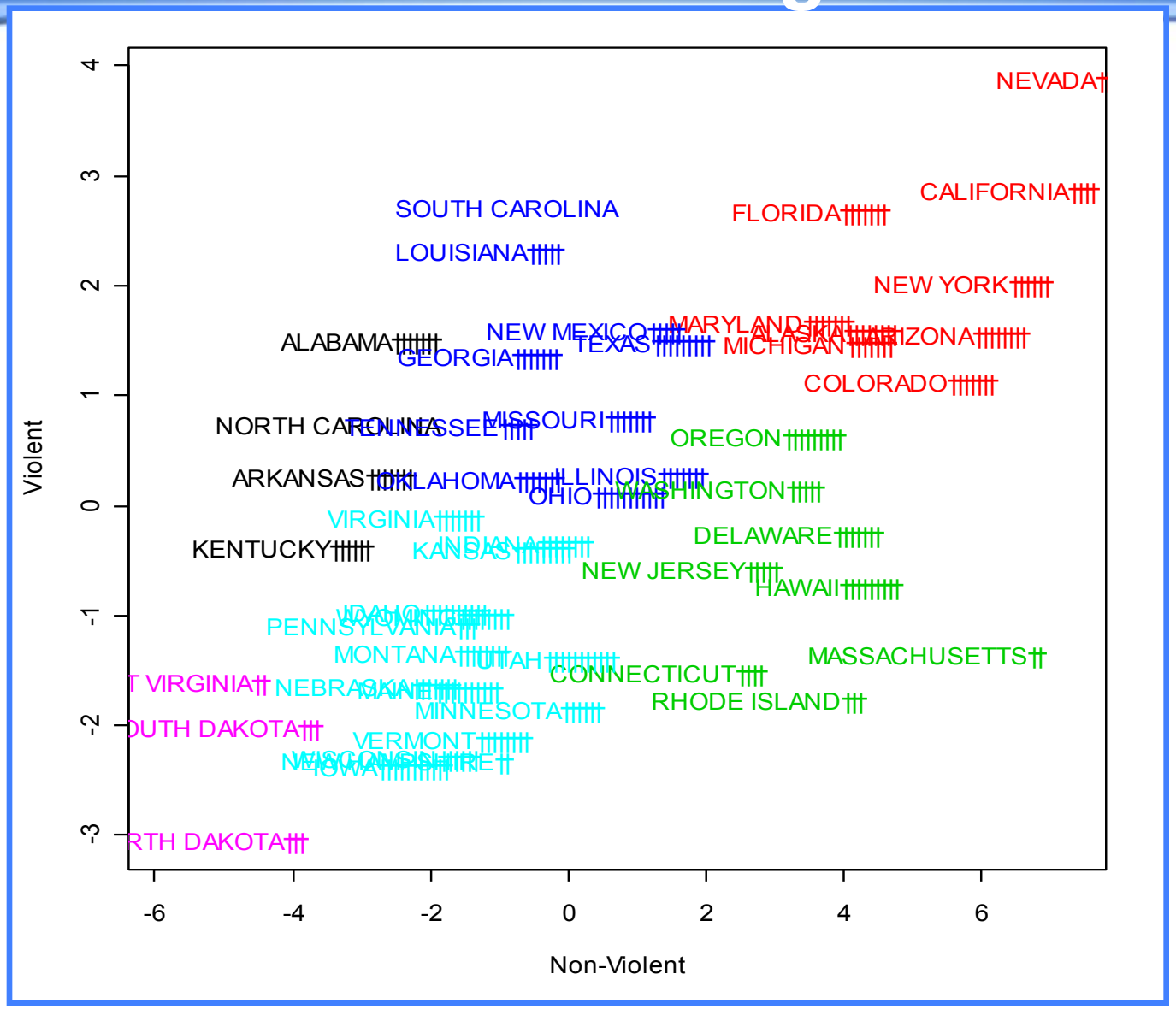




Cluster Analysis : Dendrogram using Ward‘s method



Cluster Analysis : 6 clusters selected using Ward's method



Non Hierarchical clustering: k-means

Centroid methods. *k-means* algorithm:

We start with a choice of k clusters and a choice of distance.

a. Determine the initial set of k clusters. k seed points are chosen and the data is distributed among k clusters.

b. Calculate the centroids of the k clusters and move each point to the cluster whose centroid is closest.

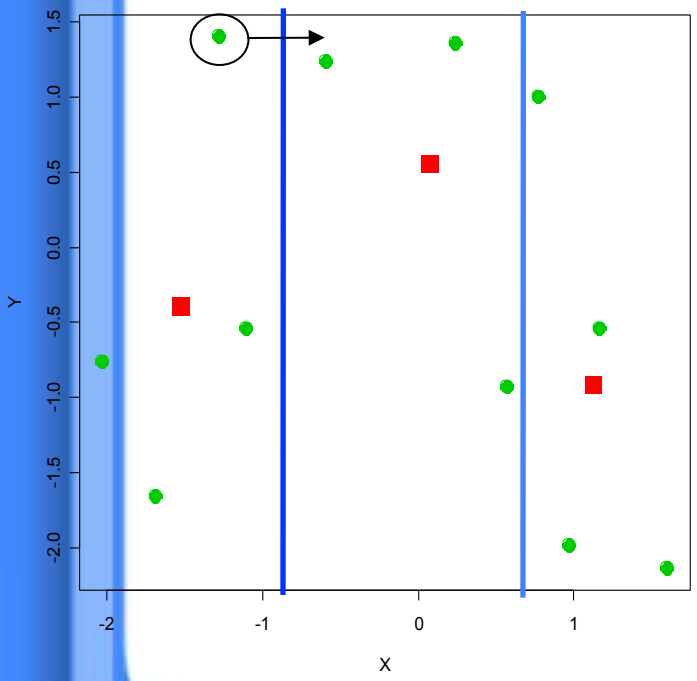
c. Repeat step b. until no change is observed.

This is the same as optimizing the R^2 criteria. At each stage of the algorithm one point is moved to the cluster that will optimize the criteria function. This is iterated until convergence occurs. The final configuration has some dependence on the initial configuration so it is important to take a good start.

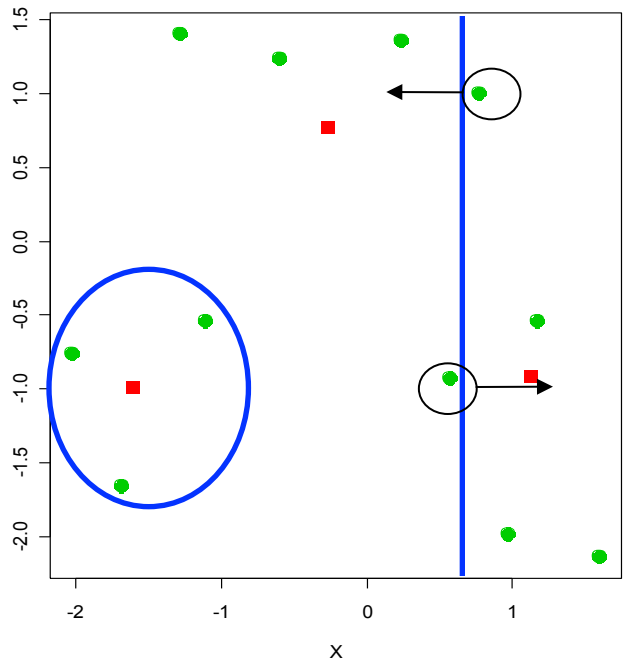
One possibility is to run *WARD*'s method and use the outcome as initial configuration for *k-means*.

Centroid methods: K-means algorithm. .

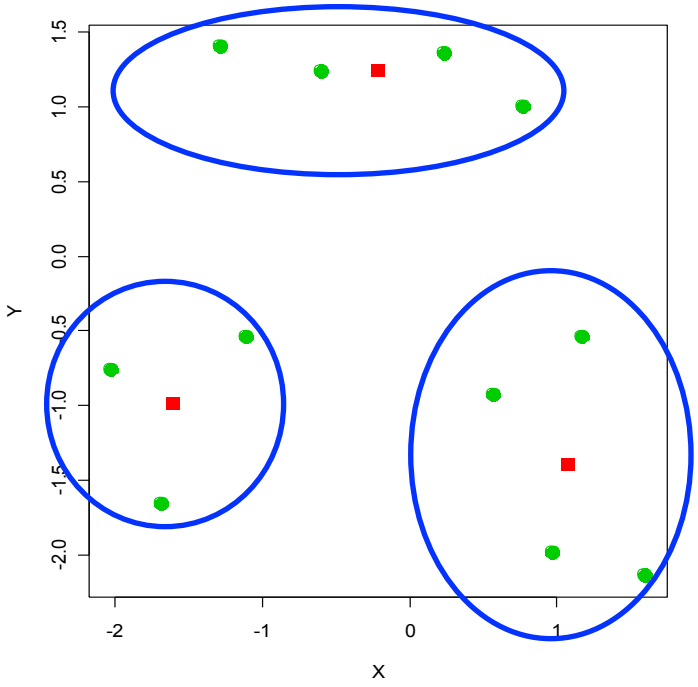
1. K seed points are chosen and the data is distributed among k clusters.
2. At each step we switch a point from one cluster to another if the R^2 is increased.
3. Then the clusters are slowly optimized by switching points until no improvement of the R^2 is possible



Step 1



Step 2



Step n

Non Hierarchical clustering: PAM

PAM

Pam is a robust version of *k-means*.

It used the medioids as the center and L_1 distance (Manhattan) and it is otherwise the same as K-means.

The cluster ***R*** package contains the ***pam*** function.

Model Based Hierarchical Clustering

Another approach to hierarchical clustering is model-based clustering, which is based on the assumption that the data are generated by a mixture of underlying probability distributions.

The ***mclust*** function fits model-based clustering models. It also fits models based on heuristic criteria similar to those used by ***pam***.

The R package ***mclust*** and the function of the same name are available from CRAN. The ***mclust*** function is separate from the cluster library, and has somewhat different semantics than the methods discussed previously.

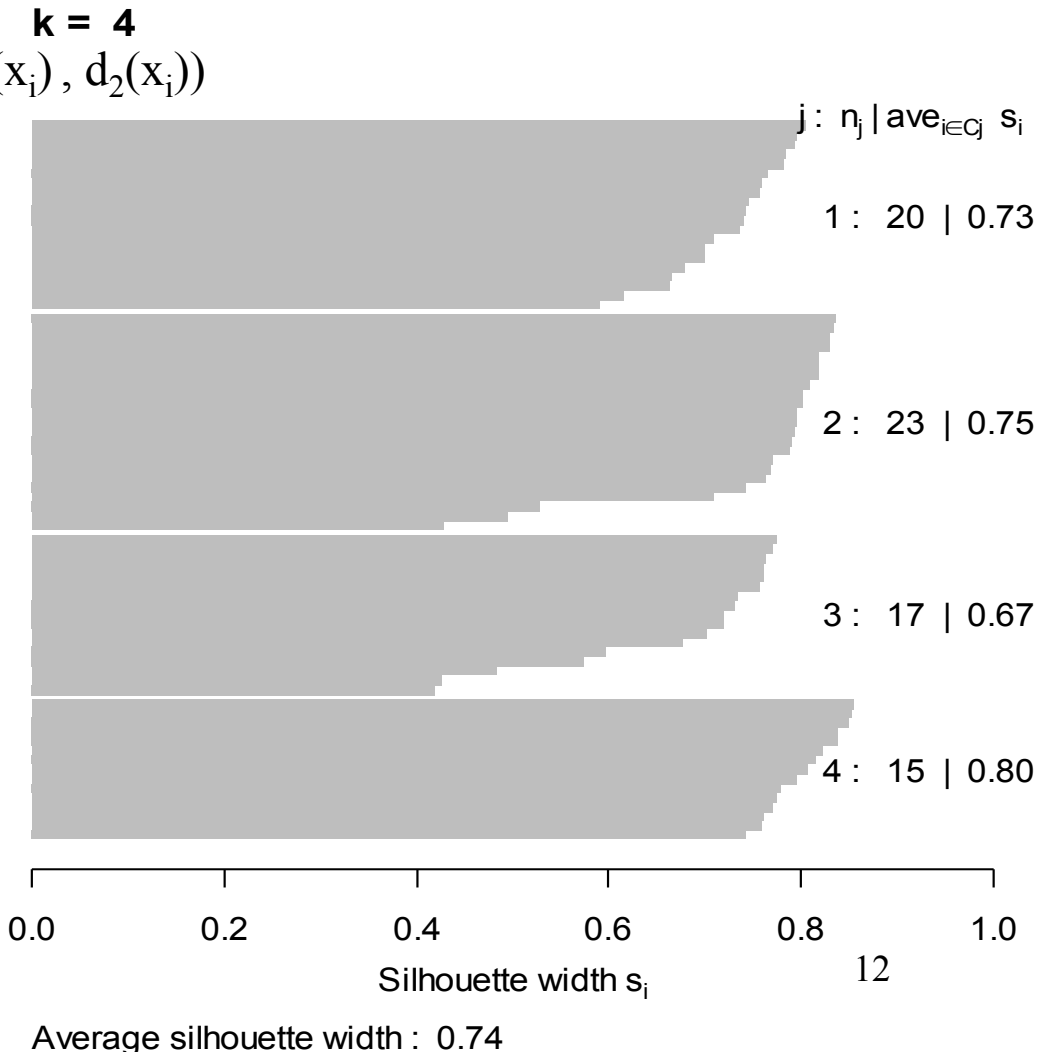
Detecting the number of clusters: silhouette graphs

library(cluster); data(ruspini); plot(silhouette(pam(ruspini, k=4)), main = paste("k = ",4), do.n.k=FALSE)

Silhouette value $s_i = (d_2(x_i) - d_1(x_i)) / \mathrm{Max}((d_1(x_i), d_2(x_i))$

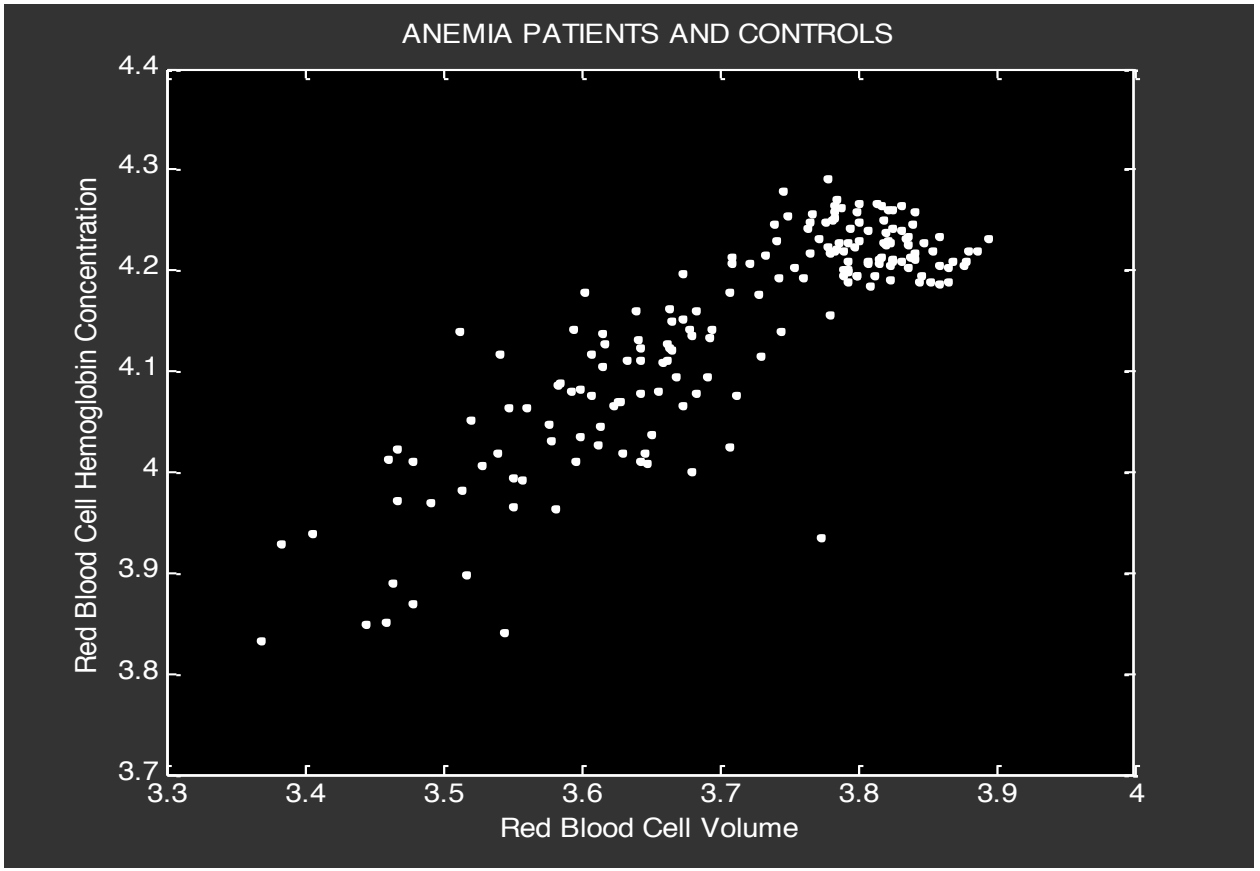
where $d_1(x_i)$ is the average distance of x_i to its cluster and
$d_2(x_i)$ is the average distance of x_i to its closest cluster.

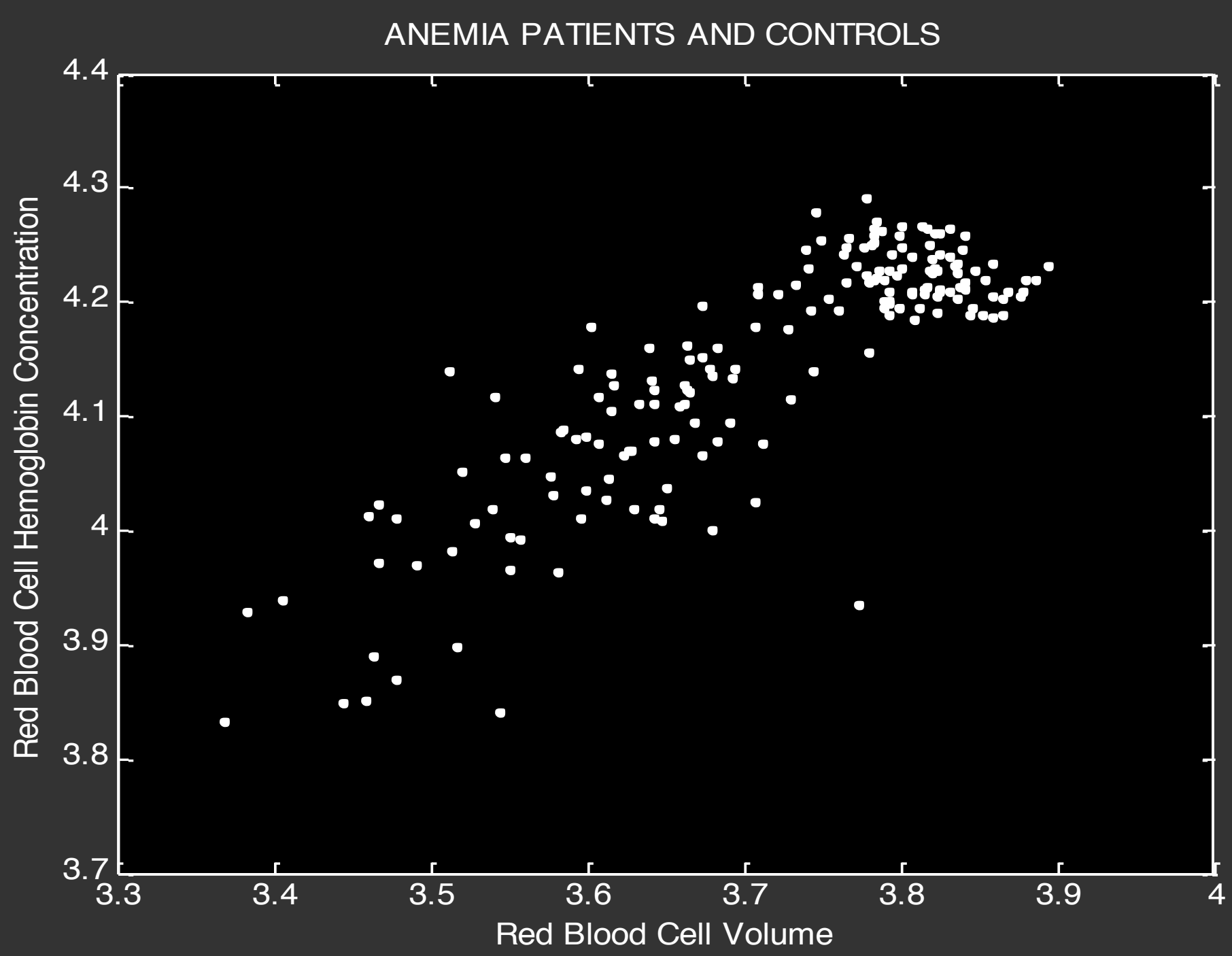
Ave Silhouette statistic = Ave(s_i)

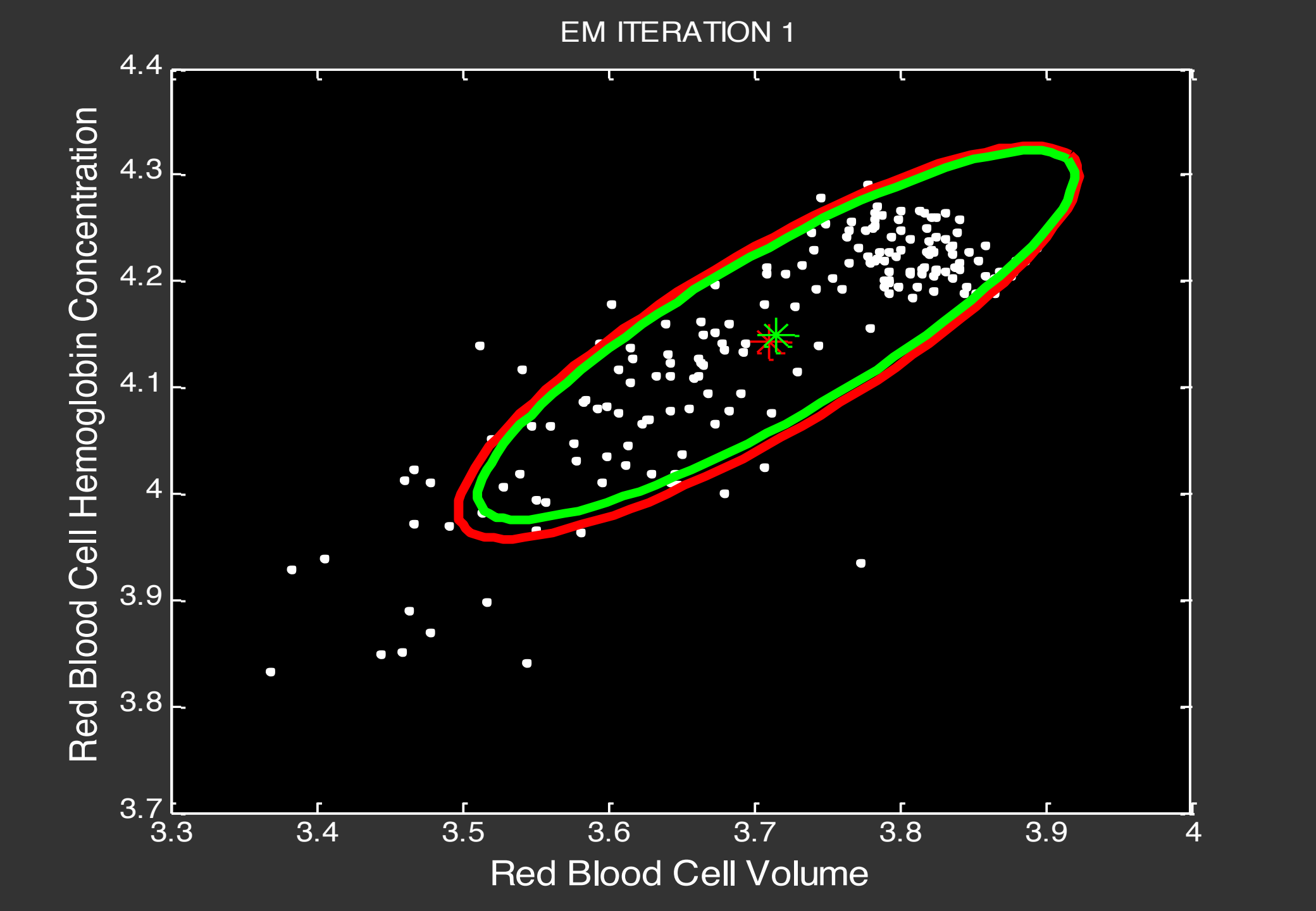


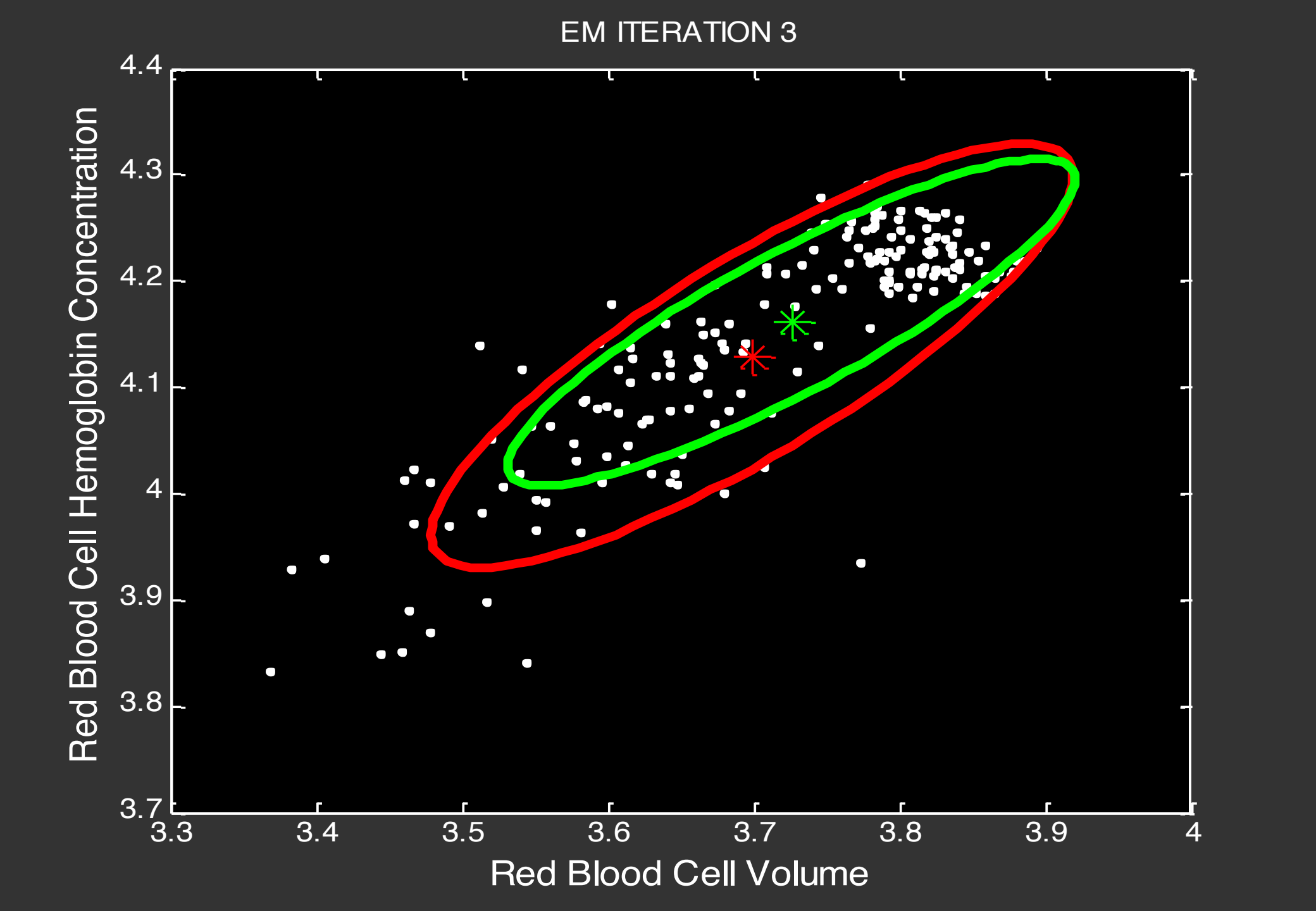
Model-based Clustering

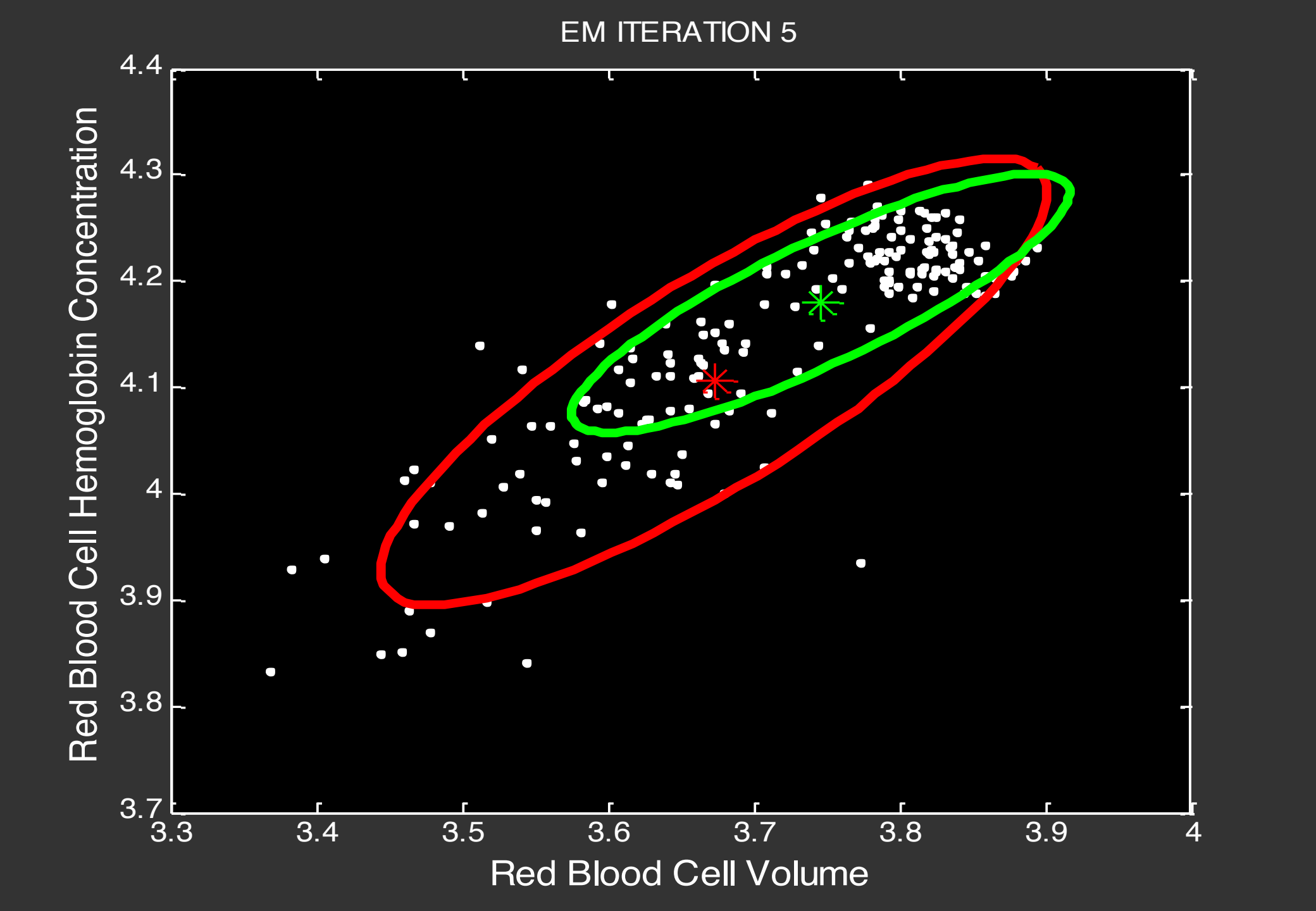
$$f(x) = \sum_{k=1}^{K} \pi_k f_k(x;\theta_k)$$

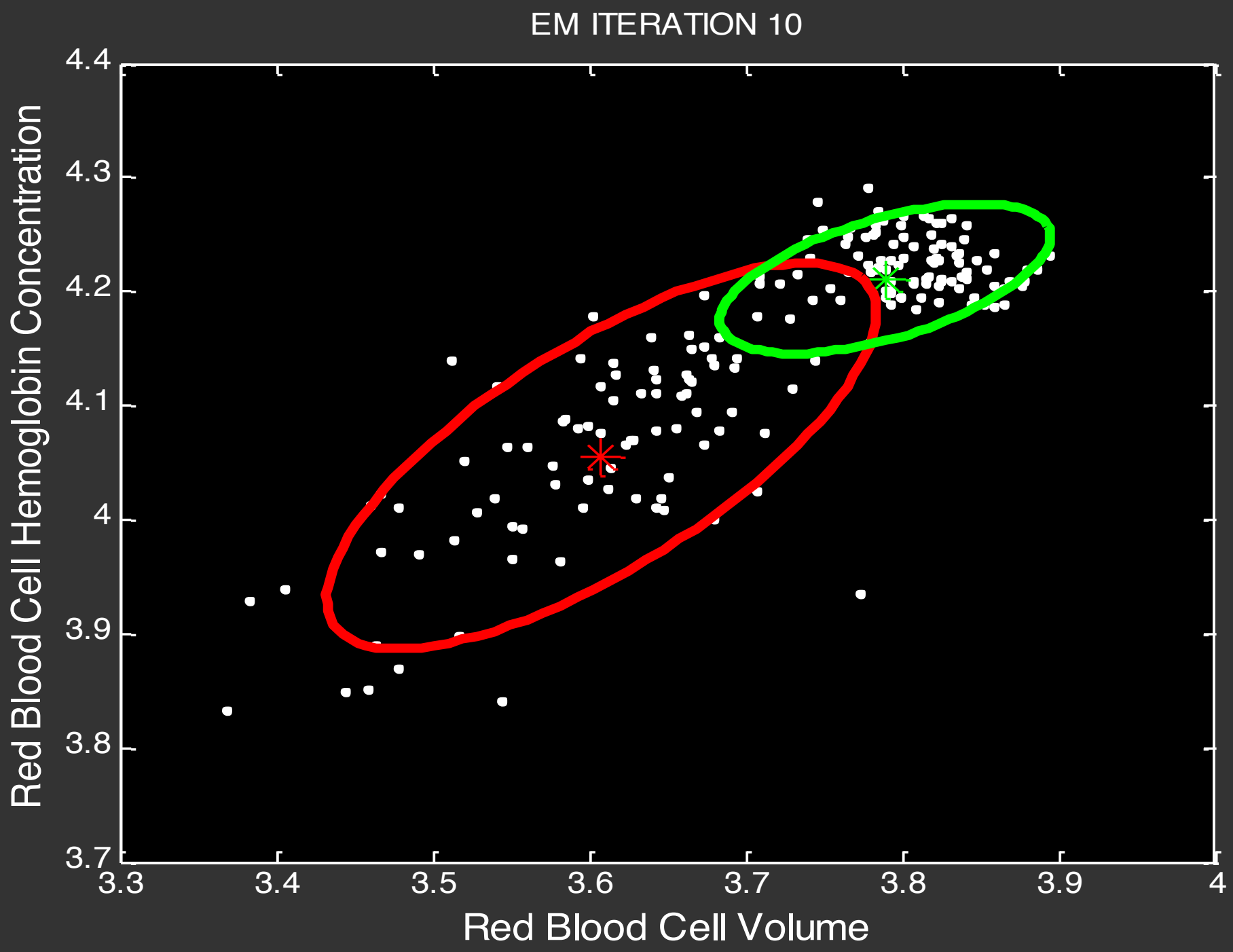


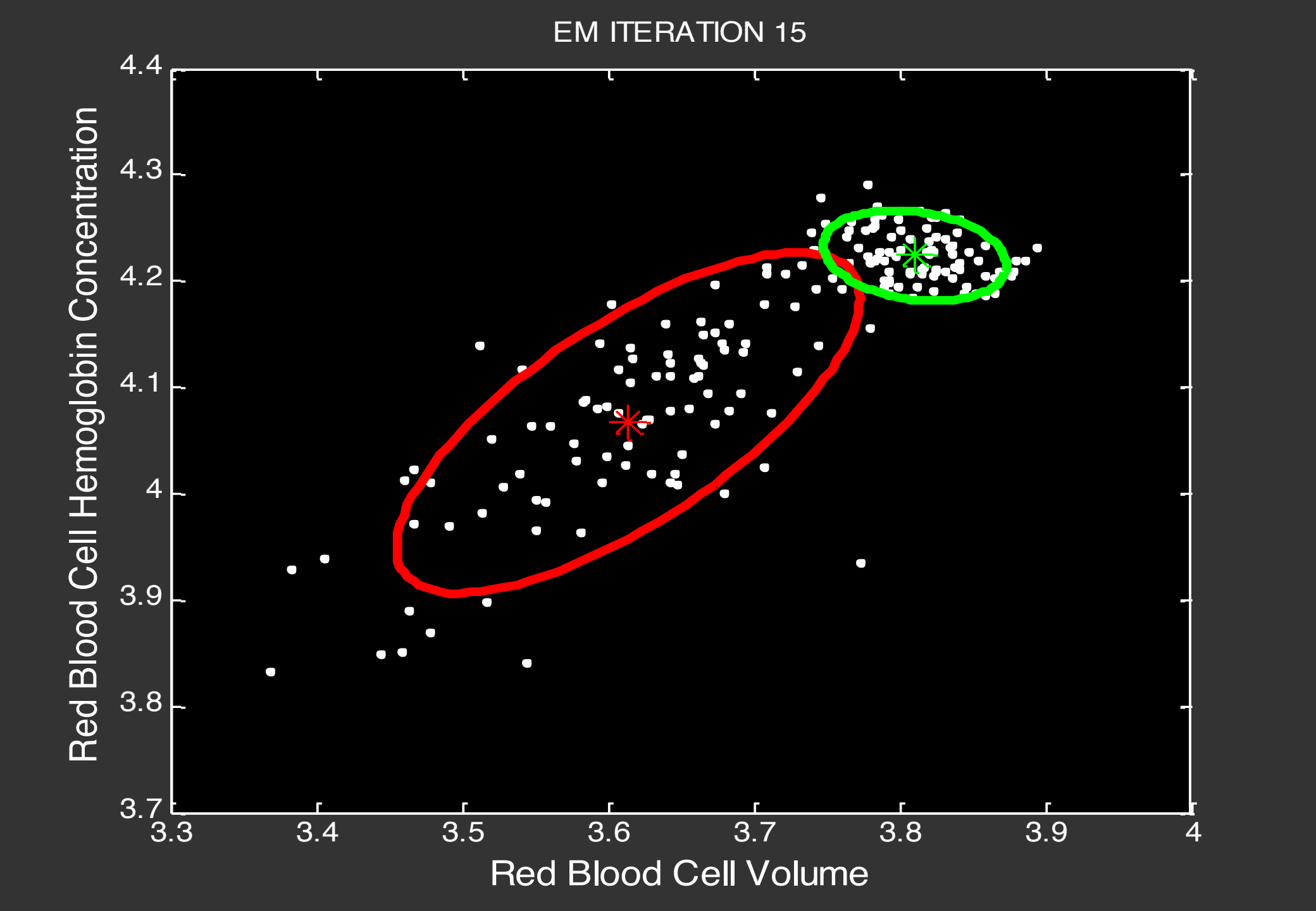

ANEMIA PATIENTS AND CONTROLS
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4

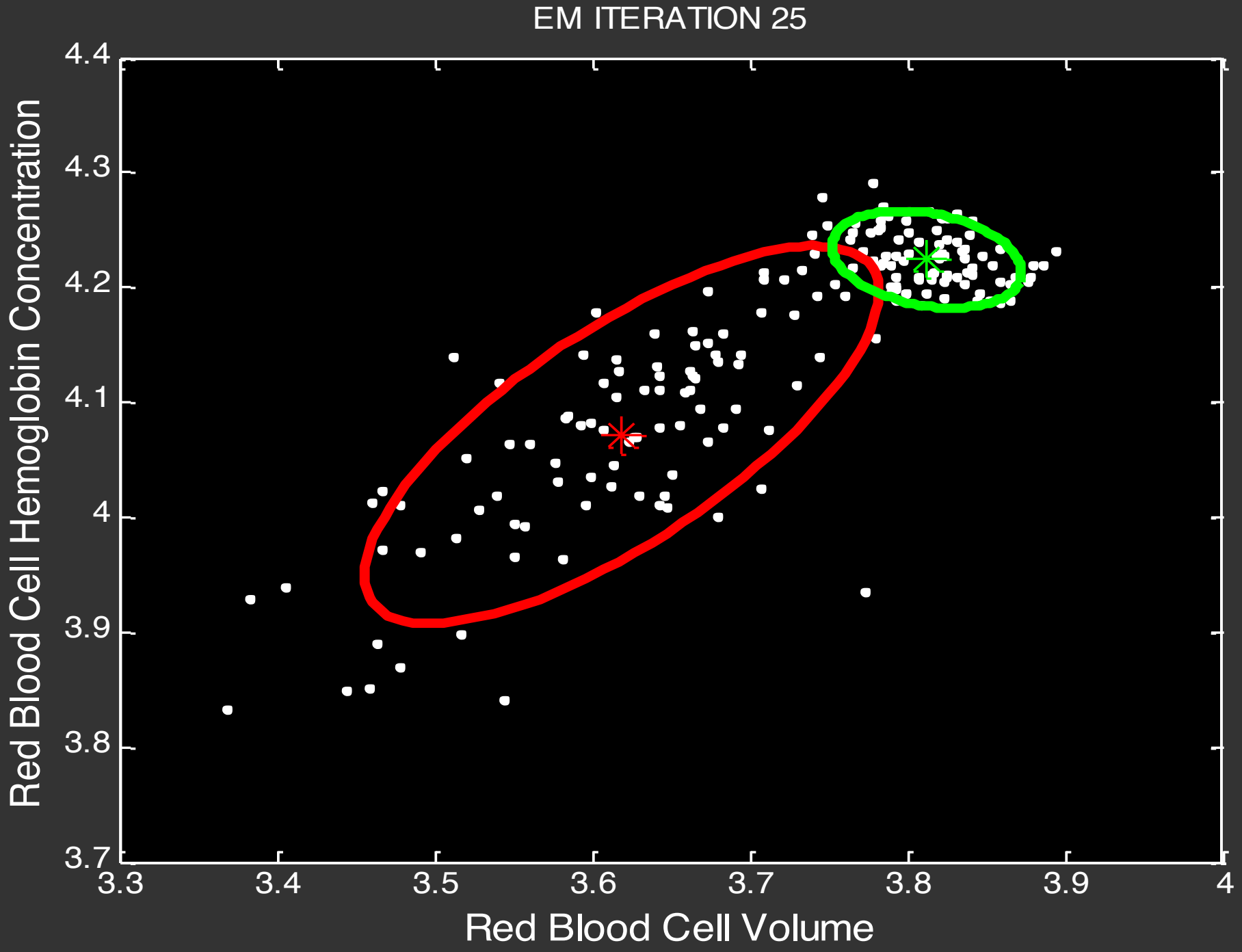

EM ITERATION 1
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4


EM ITERATION 3
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4


EM ITERATION 5
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4


EM ITERATION 10
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4


EM ITERATION 15
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4


EM ITERATION 25
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4

Mixtures of {Sequences, Curves, …}

$$p(D_i) = \sum_{k=1}^{K} p(D_i \mid c_k)\,\alpha_k$$

Generative Model

- select a component c_k for individual i

- generate data according to $p(D_i \mid c_k)$

- $p(D_i \mid c_k)$ can be very general

- e.g., sets of sequences, spatial patterns, etc

[Note: given $p(D_i \mid c_k)$, we can define an EM algorithm]

More general: Finite Mixture Models

Megavariate data: ABC clustering

1. *A Bootstrap approach called ABC*
   Refers to the Bagging of genes and samples from Microarray data. Genes are bagged using weights proportional to their variances.

2. *By creating new datasets out of subsets of columns and genes we are able to create estimates of the class response several hundred times.*

3. *These estimates are then used to obtain a dissimilarity (distance) measure between the samples of the original data.*

4. *This dissimilarity matrix is then adopted to cluster the data.*

Data

Gene	S1	S2	S3	S4	S5	S6
G8521	1003	1306	713	1628	1268	1629
G8522	890	705	566	975	883	1005
G8523	680	749	811	669	724	643
G8524	262	311	336	1677	1286	1486
G8525	254	383	258	1652	1799	1645
G8526	81	140	288	298	241	342
G8527	4077	2557	2600	3394	2926	2755
G8528	2571	1929	1406	2439	1613	5074
G8529	55	73	121	22	141	44
G8530	1640	1693	1517	1731	1861	1550
G8531	168	229	284	220	310	315
G8532	323	258	359	345	308	315
G8533	12131	11199	14859	11544	11352	11506
G8534	11544	11352	12131	11199	14859	12529
G8535	1929	1406	2439	254	383	258
G8536	191	140	288	298	241	342
G8537	4077	2557	2600	3394	2926	2755
G8538	2571	1613	5074	1652	1799	1645
G8539	55	73	121	22	91	24
G8540	1640	1693	1517	1731	1861	1750
G8541	168	229	284	220	312	335
G8542	323	258	359	345	298	325
G8543	2007	1878	1502	1758	2480	1731
G8544	2480	1731	2007	1878	1502	1758
G8545	1652	1799	1645	254	383	258
G8546	298	241	342	81	150	298
G8547	2607	3394	2926	2755	3077	2227
G8548	2571	1929	1406	2439	1613	5074
G8549	121	22	55	730	201	35

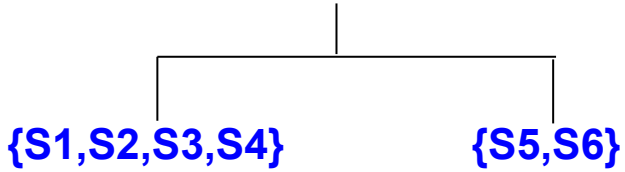
Select n samples and g genes

Gene	S1	S2	S4	S5	S6
G8523	680	749	669	724	643
G8524	262	311	1677	1286	1486
G8528	2571	1929	2439	1613	5074
G8530	1640	1693	1731	1861	1550
G8537	4077	2557	3394	2926	2755
G8545	1652	1799	254	383	258
G8547	2607	3394	2755	3077	2227

Compute similarity

Similarity	S1	S2	S3	S4	S5	S6
S1	0	6	7	7	0	0
S2	6	0	5	5	1	1
S3	7	5	0	8	0	0
S4	7	5	8	0	2	2
S5	0	2	0	2	0	10
S6	0	2	0	2	10	0

Final Clusters



Examples

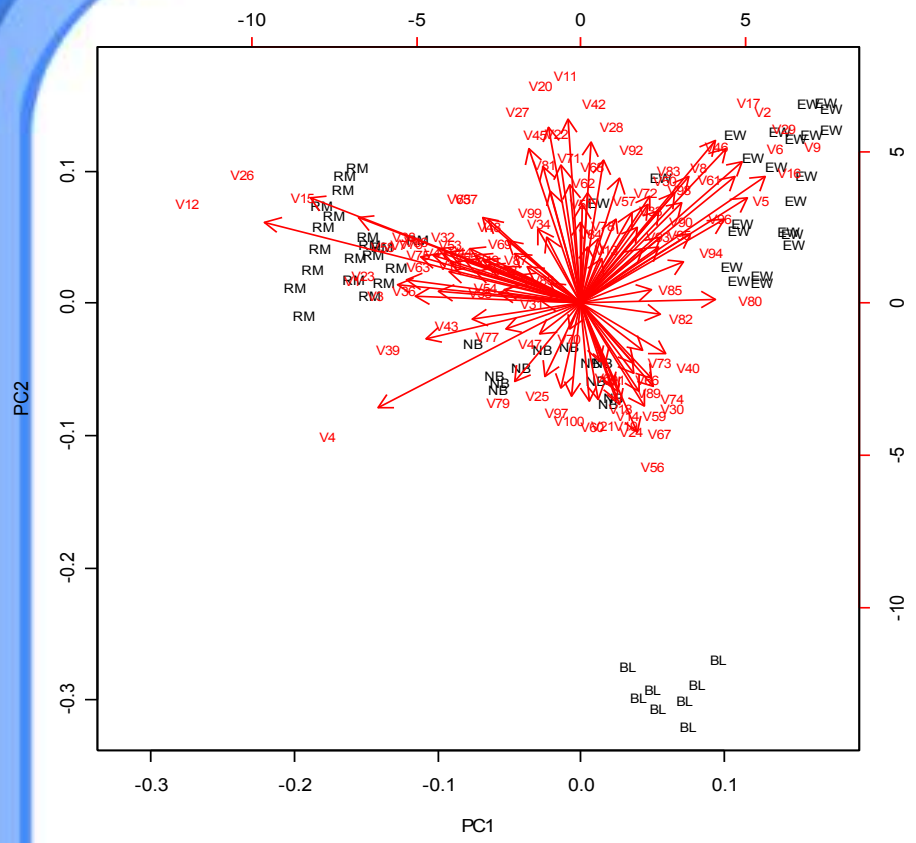
For each data set:

 # Genes Selected= √G, *# Simulations = 500*

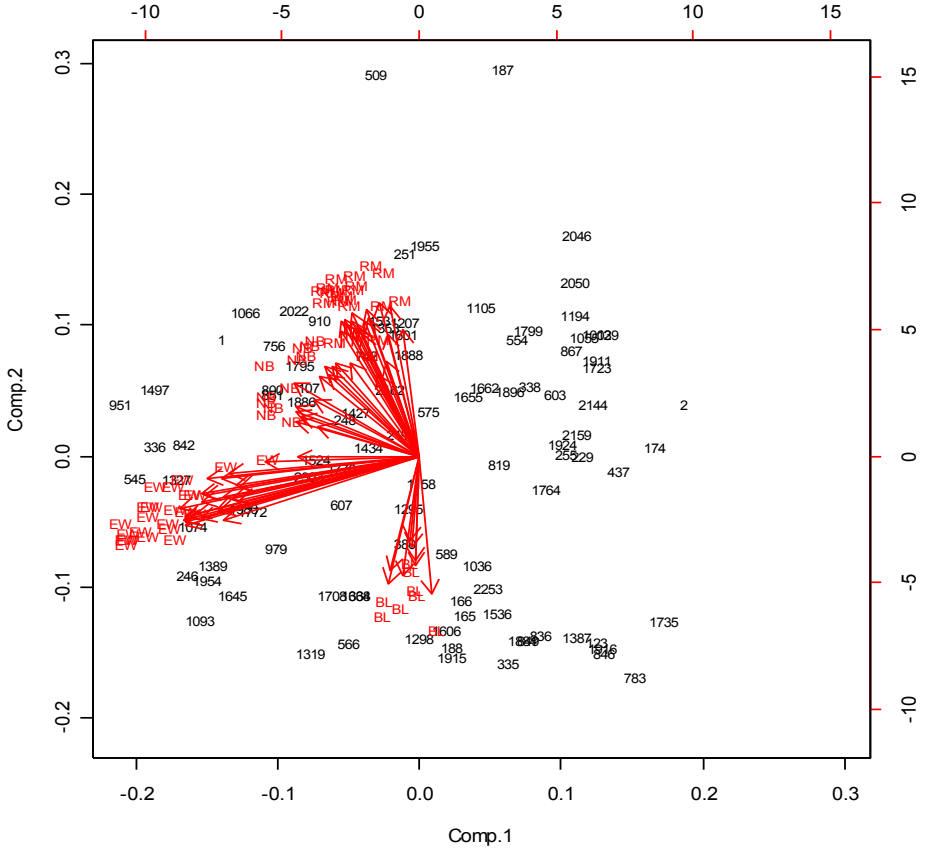
Genes Bagged By Variance

	Armstrong	Colon	Tao	Golub	Iris
BagWeight	**0.01**	**0.1**	**0.2**	**0.17**	**0.05**
BagEquiWeight	0.07	0.48	0.2	0.36	0.11
BagWholeData	0.08	0.48	0.3	0.4	**0.05**
NoBagWeight	**0.01**	**0.1**	**0.2**	**0.17**	0.08
NoBagEquiWeight	0.03	0.37	0.2	0.4	0.13
Ward	0.1	0.48	0.4	0.29	0.09
Kmeans	0.06	0.48	0.4	0.21	0.11

Biplot of the first two principal components.



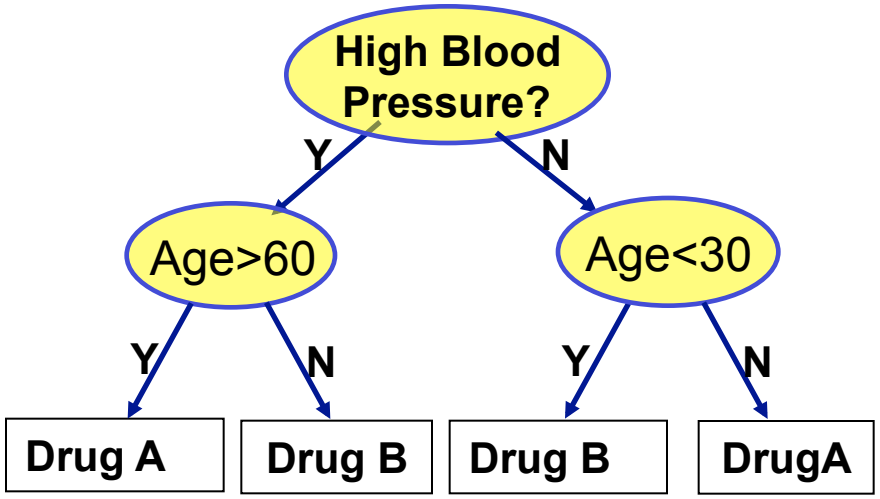
Biplot of the first two Principal components.



Supervised Classification: Recursive Partition (CART)

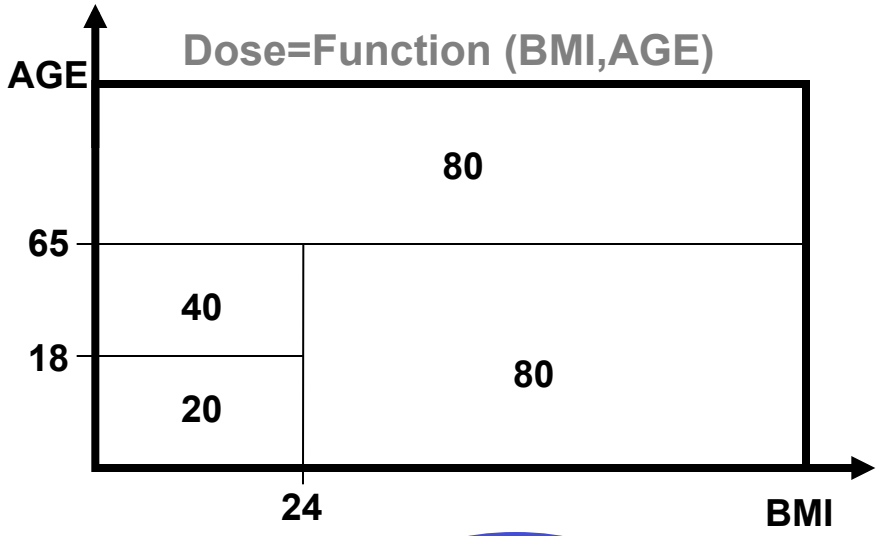
I. Dependent variable is categorical

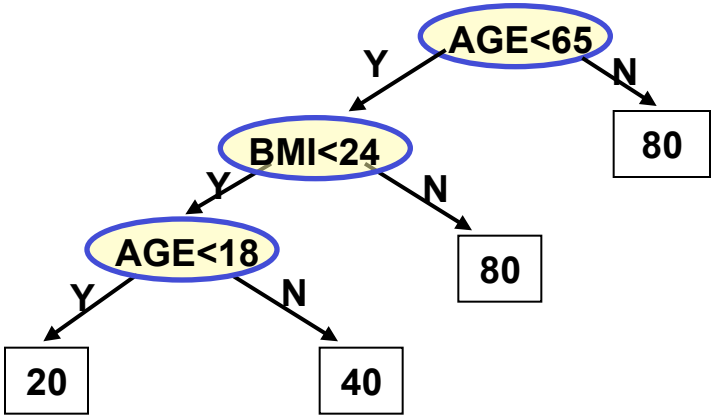
•**Classification Trees, Decision Trees**
Example: A doctor might have a rule for choosing which drug to prescribe to high cholesterol patients.



II. Dependent variable is numerical

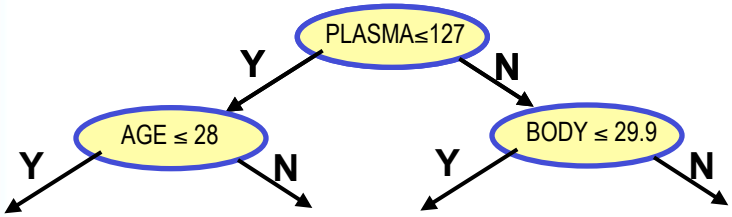
• Regression Tree



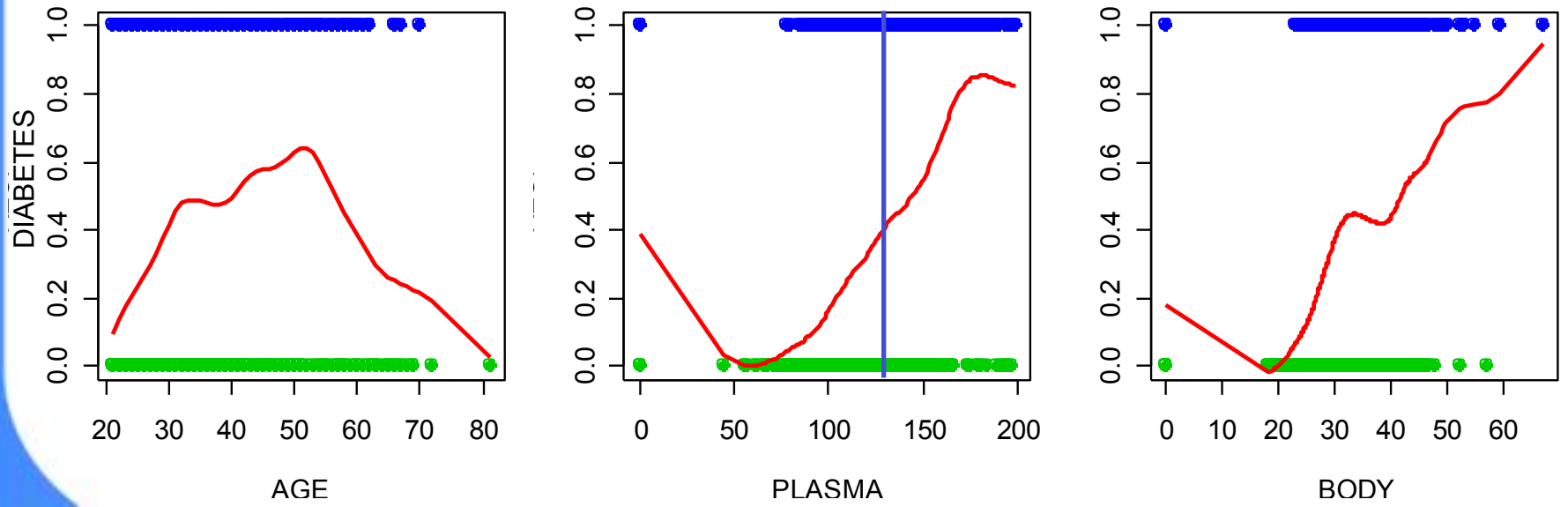


Classic Example of CART: Pima Indians Diabetes

- 768 Pima Indian females, 21+ years old ; 268 tested positive to diabetes
- 8 predictors: PRG, PLASMA, BP, THICK, INSULIN, BODY, PEDIGREE, AGE
- OBJECTIVE: PREDICT DIABETES



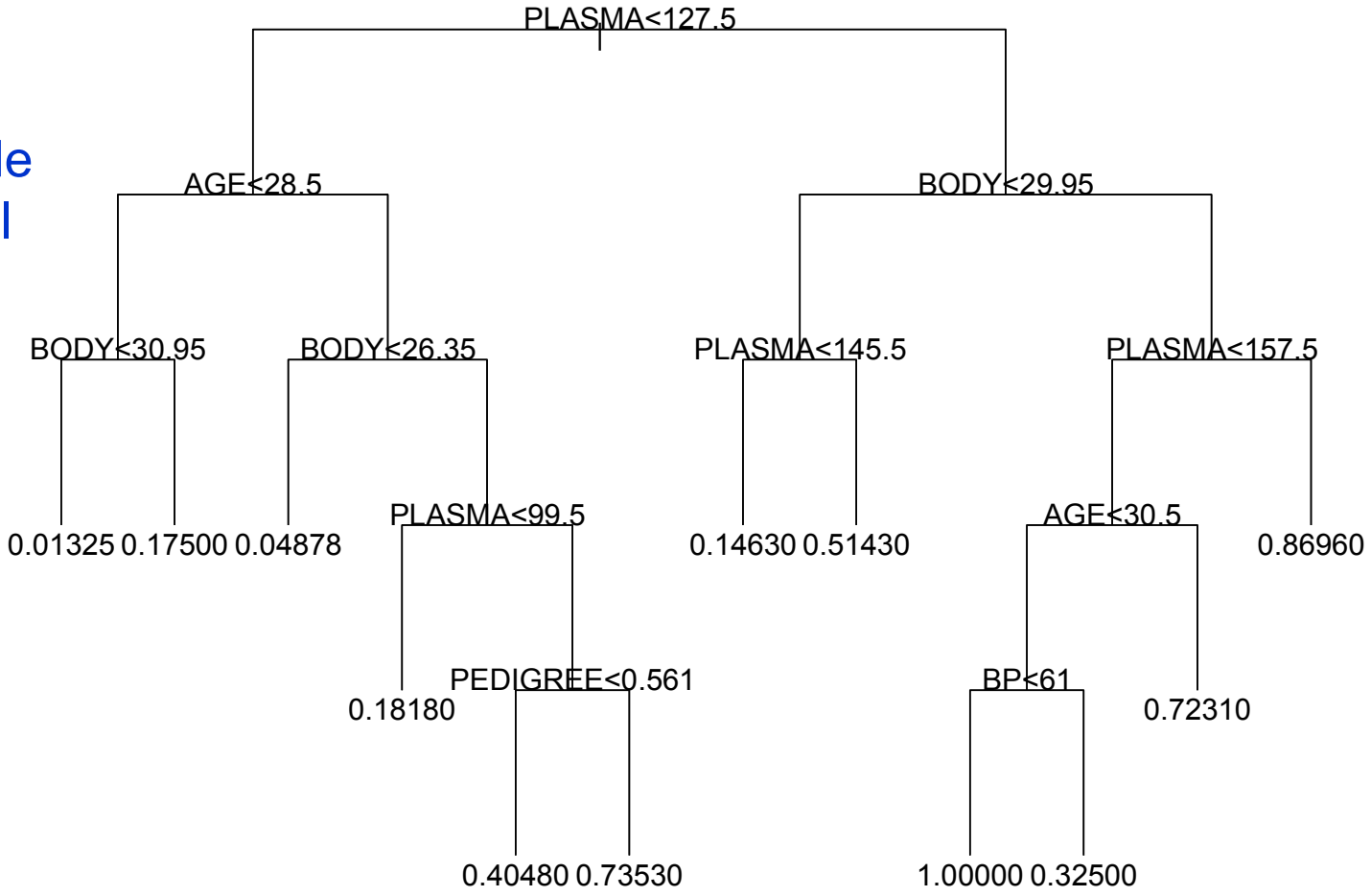
Node	CART	N	P(Diabetes)
Combined	993.5	768	35%
PLASMA<=127	854.3	485	19%
PLASMA>127		283	61%
AGE<=28	916.3	367	19%
AGE>28		401	49%
BODY<=27.8	913.7	222	12%
BODY>27.8		546	44%



Classic Example of CART: Pima Indians Diabetes

CART Algorithm

- Grow tree
- Stop when node sizes are small
- Prune tree



CART Algorithm & criteria functions

ALGORITHM

-Build the tree until the terminal buckets are small.

-At each node find the split that optimizes the criteria

-Prune the tree using Cross-Validation or Mallow's Cp

Criteria For regression trees :

$$\textbf{Equal variances}(CART)\textbf{: h} = \frac{N_L\hat{\sigma}_L^2 + N_R\hat{\sigma}_R^2}{N_L + N_R}$$

$$\textbf{Non equal variances : h} = \frac{N_L \log\hat{\sigma}_L^2 + N_R \log\hat{\sigma}_R^2}{N_L + N_R}$$

For classification trees: criteria functions

$$\mathbf{h} = p_L \min(p_L^0, p_L^1) + p_R \min(p_R^0, p_R^1)$$

$$\mathbf{h} = p_L(-p_L^0 \log p_L^0 - p_L^1 \log p_L^1) + p_R(-p_R^0 \log p_R^0 - p_R^1 \log p_R^1) \text{ (C5)}$$

$$\mathbf{h} = p_L p_L^0 p_L^1 + p_R p_R^0 p_R^1 \text{ (CART)}$$

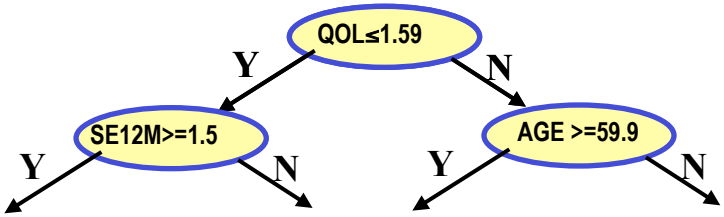
Case Study

A study compares azithromycin extended release (AZ-ER) versus moxicillin/ clavulanate potassium (A/C) in subjects with acute bacterial sinusitis (ABS)

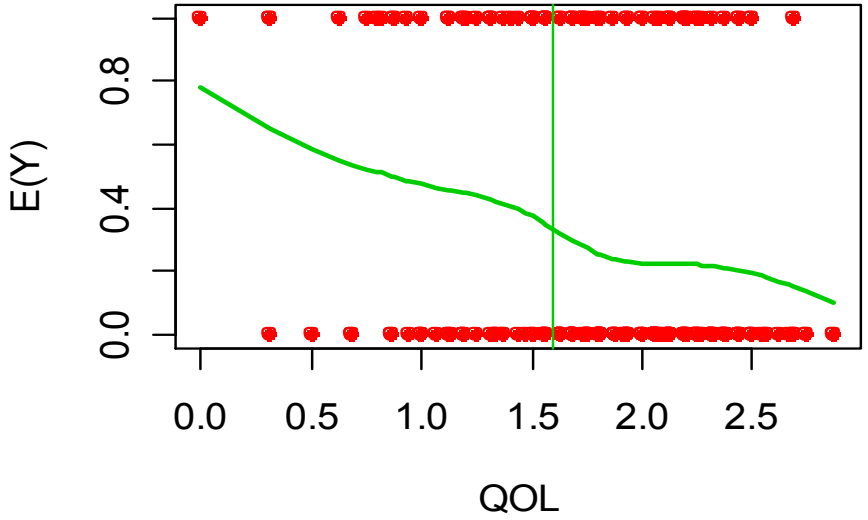
	AZ-ER(N=236)	A/C (N=238)
Age of Subject	46.5 (13.5)	46(12.4)
BMI	29.3(7.7)	29.2(7.4)
Gender	F: 161 M: 75	F:162 M:76
Smoking	Never:150 Ex:44 Current: 42	Never:156 Ex:43 Current: 39
Med in last 30 days	Nasal : 12 Decongestant: 70 Antistamine: 38	16 63 29
Race	A:3 B:8 H:11 O:214	A:5 B:5 H:7 O:221
History of Allergy	N: 145 Y:91	N:152 Y:66
Fever	N: 174 Y:62	N:152 Y:66
Symptoms Duration	Med =4W	Med=4W
QOL at Enrolment	Med =1.8	Med=1.78
Episodes in 12 months	Med =1	Med=1

Example of using CART:

- **236 AZ-ER PATIENTS.**
- **11 Predictors**
- **OBJECTIVE: PREDICT RESPONSE IN 5 DAYS**



Node	CART	N	P()
Combined	70	236	30%
QOL_ENRL>=1.59	32		
QOL_ENRL<1.59	33		
SE12M>=1.5	21		
SE12M <1.5	48		
AGE >= 65	15		
AGE <65	54		



Competing methods

Recursive Partition:

Find the partition that best approximates the response.

For moderate/large datasets partition tree may be too big

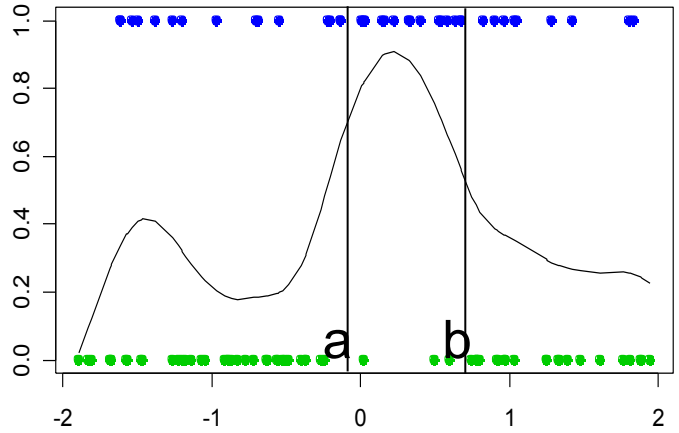
Data Mining Trees (ARF)

Bump Hunting:

Find bump that optimizes criteria

Subsets are more “robust”

Not all interesting subsets are found



DATA PREPROCESSING RECOMMENDATIONS FOR TREES

a. Make sure that all the factors are declared as factors

Some times factor variables are read into R as numeric or as character variables. Suppose that a variable RACE on a SAS dataset is coded as 1, 2, 3, 4 representing 4 race groups. We need to be sure that it was not read as a numeric variable, therefore we will first check the types of the variables. We may use the functions "class" and "is.factor" combined with "sapply" in the following way.

```
sapply(w,is.factor) or sapply(w,class)
```

Suppose that the variable "x" is numeric when it is supposed to be a factor. Then we convert it into factor:

```
w$x = factor(w$x)
```

b. Recode factors:

Sometimes the codes assigned to factor levels are very long phrases and when those code are inserted into the tree the resulting graph can be very messy. We prefer to use short words to represent the codes. To recode the factor levels you may use the function "f.recode":

```
> levels(w$Muscle)
  [1] ""                          "Mild Weakness"
  [3] "Moderate Weakness"         "Normal"
> musc =f.recode(w$Muscle,c("","Mild","Mod","Norm"))
> w$Musclenew = musc
```

Example Hospital data

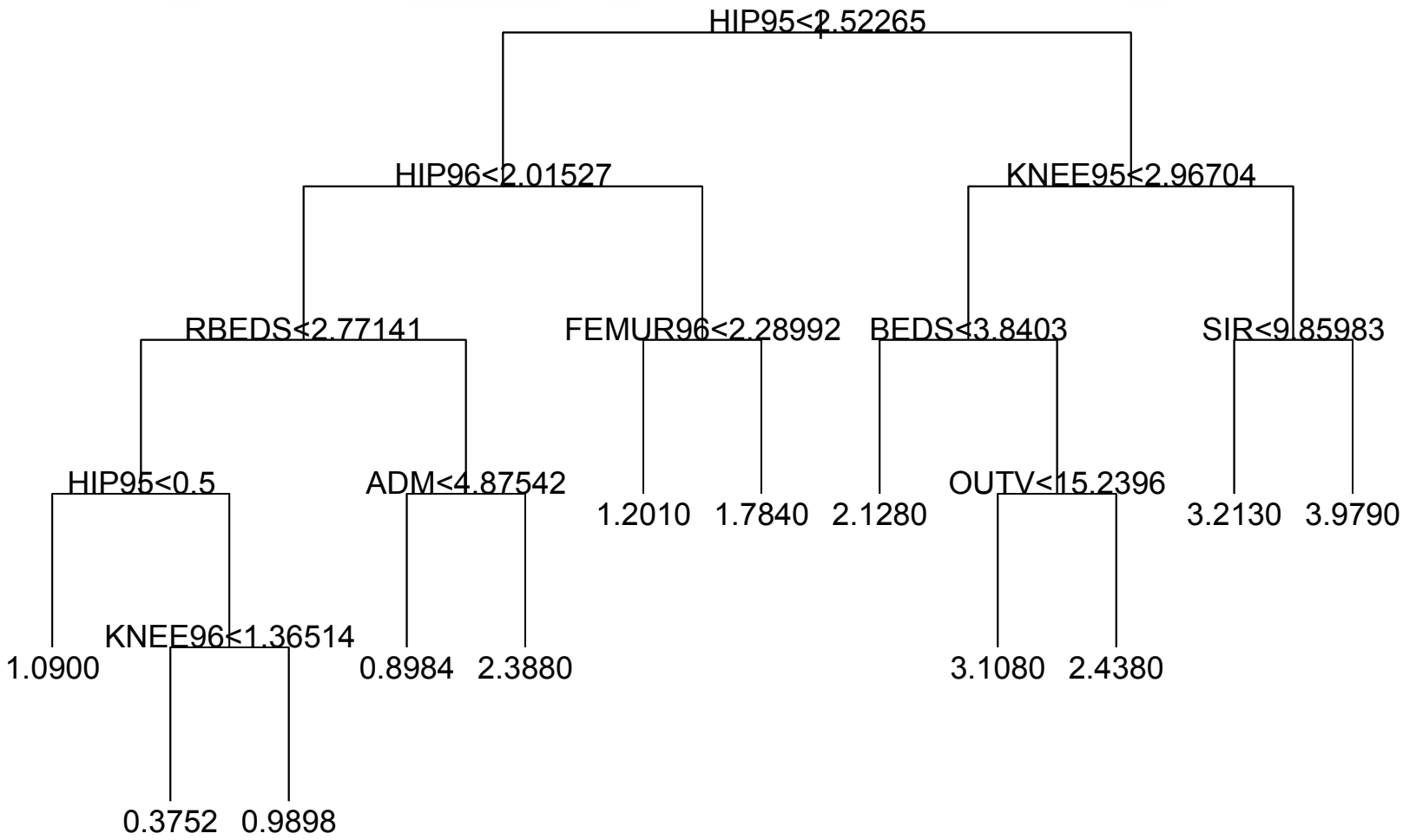
```
hospital = read.csv("hospital.csv")
library(rpart)
hosp = hospital[1:1000,-c(1:4,10)]
hosp$TH = factor(hosp$TH)
hosp$TRAUMA = factor(hosp$TRAUMA)
hosp$REHAB = factor(hosp$REHAB)

u=rpart(log(1+SALES12)~.,data=hosp,control=rpart.control(cp=.01))
plot(u);  text(u)
u=rpart(log(1+SALES12)~.,data=hosp,control=rpart.control(cp=.001))
plot(u,uniform=T) ; text(u)
```

Regression Tree for log(1+Sales)

```
HIP95  < 40.5 [Ave: 1.074, Effect: -0.76 ]
    HIP96  < 16.5 [Ave: 0.775, Effect: -0.298 ]
        RBEDS  < 59 [Ave: 0.659, Effect: -0.117 ]
            HIP95  < 0.5 [Ave: 1.09, Effect: +0.431 ] -> 1.09
            HIP95 >= 0.5 [Ave: 0.551, Effect: -0.108 ]
                KNEE96  < 3.5 [Ave: 0.375, Effect: -0.175 ] -> 0.375
                KNEE96 >= 3.5 [Ave: 0.99, Effect: +0.439 ] -> 0.99
        RBEDS >= 59 [Ave: 1.948, Effect: +1.173 ] -> 1.948
    HIP96 >= 16.5 [Ave: 1.569, Effect: +0.495 ]
        FEMUR96  < 27.5 [Ave: 1.201, Effect: -0.368 ] -> 1.201
        FEMUR96 >= 27.5 [Ave: 1.784, Effect: +0.215 ] -> 1.784
HIP95 >= 40.5 [Ave: 2.969, Effect: +1.136 ]
    KNEE95  < 77.5 [Ave: 2.493, Effect: -0.475 ]
        BEDS  < 217.5 [Ave: 2.128, Effect: -0.365 ] -> 2.128
        BEDS >= 217.5 [Ave: 2.841, Effect: +0.348 ]
            OUTV  < 53937.5 [Ave: 3.108, Effect: +0.267 ] -> 3.108
            OUTV >= 53937.5 [Ave: 2.438, Effect: -0.404 ] -> 2.438
    KNEE95 >= 77.5 [Ave: 3.625, Effect: +0.656 ]
        SIR  < 9451 [Ave: 3.213, Effect: -0.412 ] -> 3.213
        SIR >= 9451 [Ave: 3.979, Effect: +0.354 ] -> 3.979
```

Regression Tree



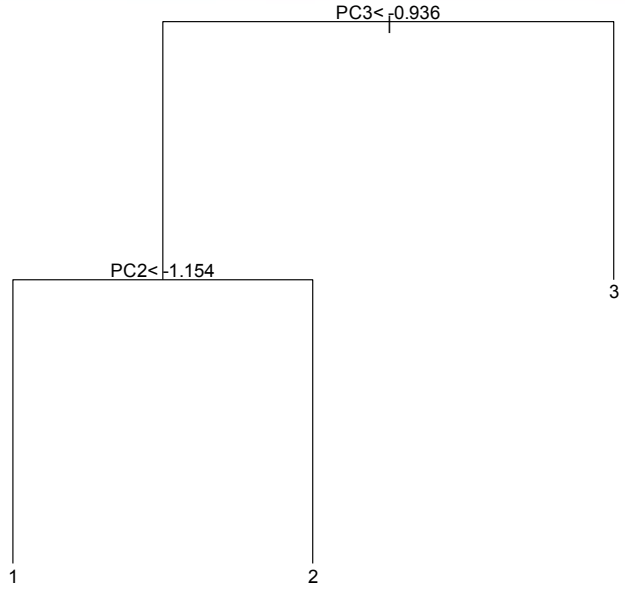
Classification tree:

- **data(tissue)**
- gr = rep(1:3, c(11,11,19))

```
> x <- f.pca(f.toarray(tissue))$scores[,1:4]
> x= data.frame(x,gr=gr)
> library(rpart)
> tr =rpart(factor(gr)~., data=x)
n= 41

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 41 22 3 (0.26829268 0.26829268 0.46341463)
  2) PC3< -0.9359889 23 12 1 (0.47826087 0.47826087 0.04347826)
    4) PC2< -1.154355 12  1 1 (0.91666667 0.00000000 0.08333333) *
    5) PC2>=-1.154355 11  0 2 (0.00000000 1.00000000 0.00000000) *
  3) PC3>=-0.9359889 18  0 3 (0.00000000 0.00000000 1.00000000) *
> plot(tr)
> text(tr)
>
```



Random forest Algorithm (A variant of bagging)

•Select ntree, the number of trees to grow, and mtry, a number no larger than number of variables.

•For i = 1 to ntree:

•Draw a bootstrap sample from the data. Call those not in the bootstrap sample the "out-of-bag" data.

•Grow a "random" tree, where at each node, the best split is chosen among mtry randomly selected variables. The tree is grown to maximum size and not pruned back.

5.Use the tree to predict out-of-bag data.

6.In the end, use the predictions on out-of-bag data to form majority votes.

7.Prediction of test data is done by majority votes from predictions from the ensemble of trees.

R-package: randomForest with function called also randomForest

Enriched methods for supervised/ unsupervised classification

Objectives: Response Prediction (class or continuous), Variable Selection, Clustering, …

Enriched Methods: Assigning weights to variables, not to observations

- Amaratunga, Cabrera, Kovtun (2007). Microarray learning with ABC, *Biostatistics*.
- Amaratunga, Cabrera, Li (2008) Enriched random forests, *Bioinformatics*
- Amaratunga, Cabrera, Cherkas, Li (2011) Enriched Ensembles, *AMS series.*
- Philippe Haldermans thesis and Yauheniya Cherkas thesis (2010).

Issues:

Modeling and Variable Selection are the obvious ideas but in practice are difficult because of the high level of spurious information.

Our idea is to apply weights to variables that correct for spurious information contained by each variable.

1. Construct FDR corrected weights
2. Model the data using these Weights : Directly or using Ensembles.

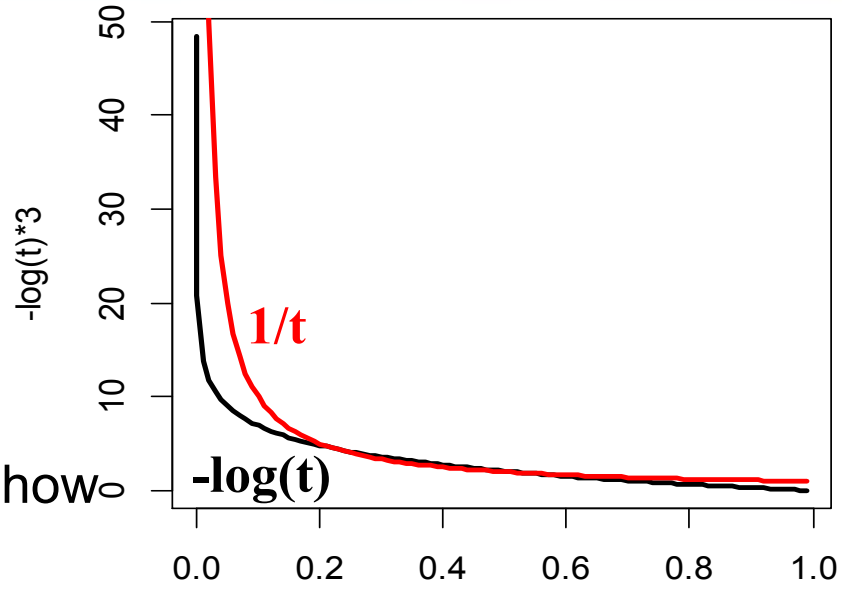
Enriched Methods

Assign a weight to each gene based on FDR

Assign weights to the individual predictor variables according to their relationship to the response.

Weight functions: $w(t) = -log(t)$

$w(t) = 1/t$



Suppose that t is the p-value of some gene wise model or some other measure of how predictive a gene is.

In unsupervised analysis, we have used gene variance as a measure of how predictive a gene is.

To avoid over-fitting we remove any signal that could be explained by chance. To do this we correct p-values for multiplicity p-value FDR corrections (sometimes called q-values).

Benjamini & Hosberg , Yekutieli & Benjamini, Storey& Tibshirani

Enriched Methods

Assign a weight to each gene based on FDR

Steps for obtaining FDR based Weights:

- Perform Statistical analysis of individual variables: Obtain p-values by testing statistically significant difference across the groups (t, F…)
- Assume a null distribution for p-values, for example uniform p-value distribution $p_i \sim Uniform[0,1]$, then $p_{(i)} \sim Beta[i, G\text{-}i]$ (order statistics)

FDR corrected p-values: $q_{(i)} = p_{(i)} / p_{(i),a}$ and make $q_{(i)}$ monotone on (i)

FDR corrected weights: $W_i = 1/q_i$ or $W_i = -log(q_i)$

- Suppose that a is a tuning constant and that $p_{(i)}$,a is the a-percentile of distribution of the ith order statistic.

We fix α =0.05 (Fisher) and calculate the weights.

Simple version: $q_{(i)} = p_{(i)} / (i/G)$ and $W_i = 1/q_i$ or $W_i = -log(q_i)$

Boosting (Ada boosting)

Input:
Data (x_i,y_i) $i=1,\ldots,n$; $w_i=1/n$

1. *Fit tree or any other learning method:* $h_1(x_i)$
2. *Calculate misclassification error* E_1
3. *If* $E_1 > 0.5$ *stop and abort loop*
4. $b_1 = E_1/(1-E_1)$
5. *for* $i=1,\ldots,n$ *if* $h_1(x_i) = y_i$ $w_i = w_i \; b_1$ *else* $w_i = w_i$
6. *Normalize the* w_i*'s to add up to 1.*
7. *Go back to 1. and repeat until no change in prediction error.*

R-package: *bagboost* with function called also *bagboost* and also *adaboost*

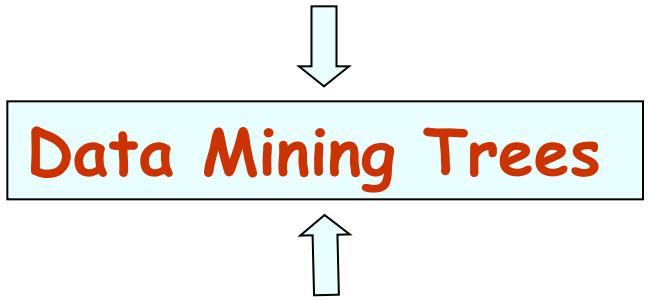
Boosting (Ada boosting)

```
i=sample(nrow(hosp),1000,rep=F)
xlearn = f.toarray((hospital[-c(1:4,10:11),]))
ylearn = 1*( hospital$SALES12 > 50)
xtest = xlearn[i,]
xlearn = xlearn[-i,]
ytest = ylearn[i]
ylearn = ylearn[-i]
## BOOSTING EXAMPLE
u = bagboost(xlearn[1:100,], ylearn[1:100],
   xtest,presel=0,mfinal=20)
summarize(u,ytest)
## RANDOM FOREST EXAMPLE
u = randomForest(xlearn[1:100,], ylearn[1:100],
   xtest,ytest)
round(importance(u),2)
```

Competing methods

Recursive Partition:

- Find the partition that best approximates the response.
- For moderate/large datasets partition tree may be too big

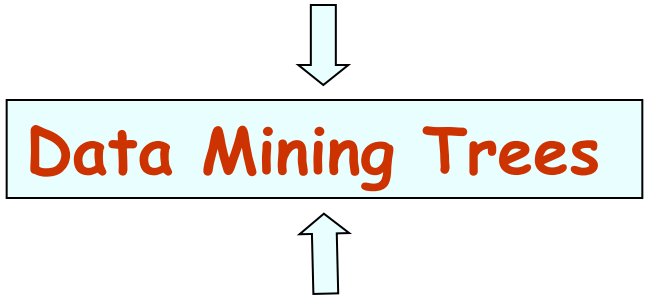


Bump Hunting:

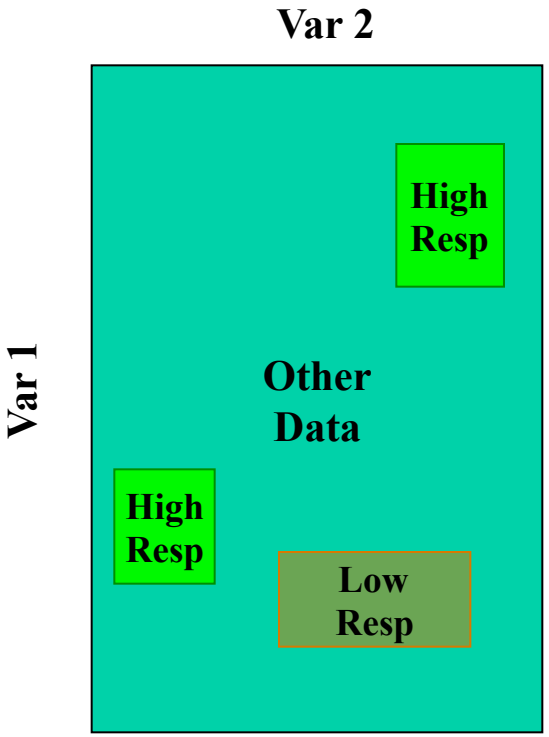
- Find subsets that optimize some criterion
- Subsets are more “robust”
- Not all interesting subsets are found

Paradigm for data mining: Selection of interesting subsets

Recursive Partition



Bump Hunting



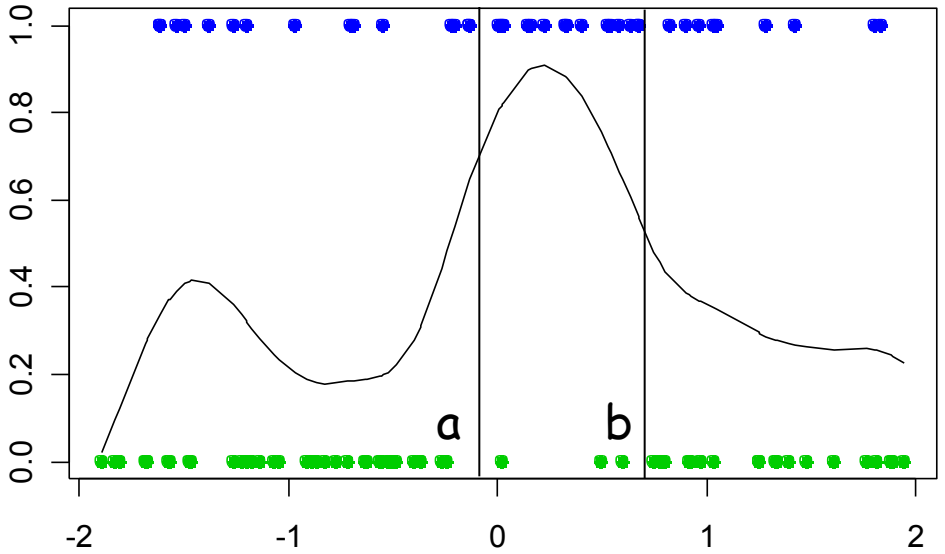
Data Mining Trees
ARF (Active Region Finder)

Naive thought: For the *j*th descriptor variable x_j, an "interesting" subset $\{a<x_{ji}<b\}$ is one such that

$$p = \text{Prob}[\, Z=1 \mid a<x_{ji}<b \,]$$

is much larger than

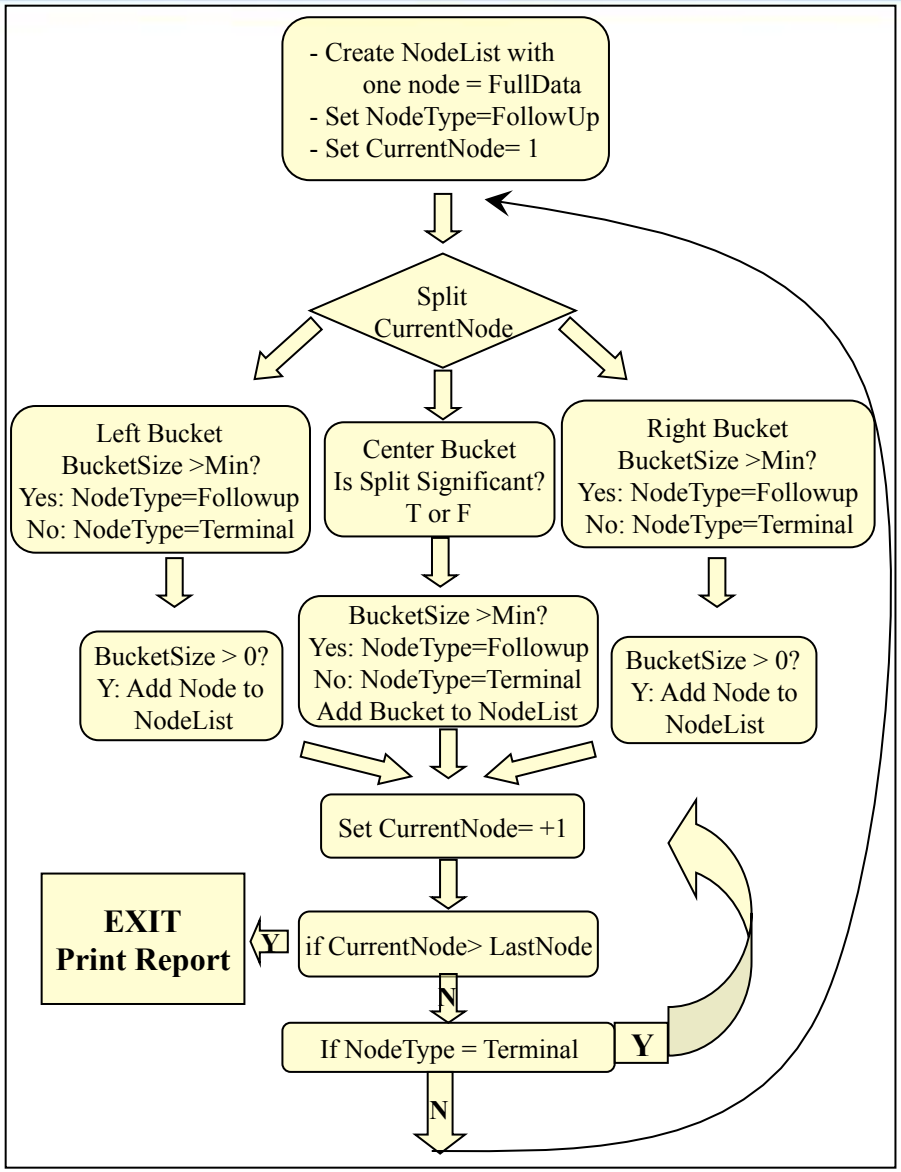
$$\pi = \text{Prob}[\, Z=1 \,]$$



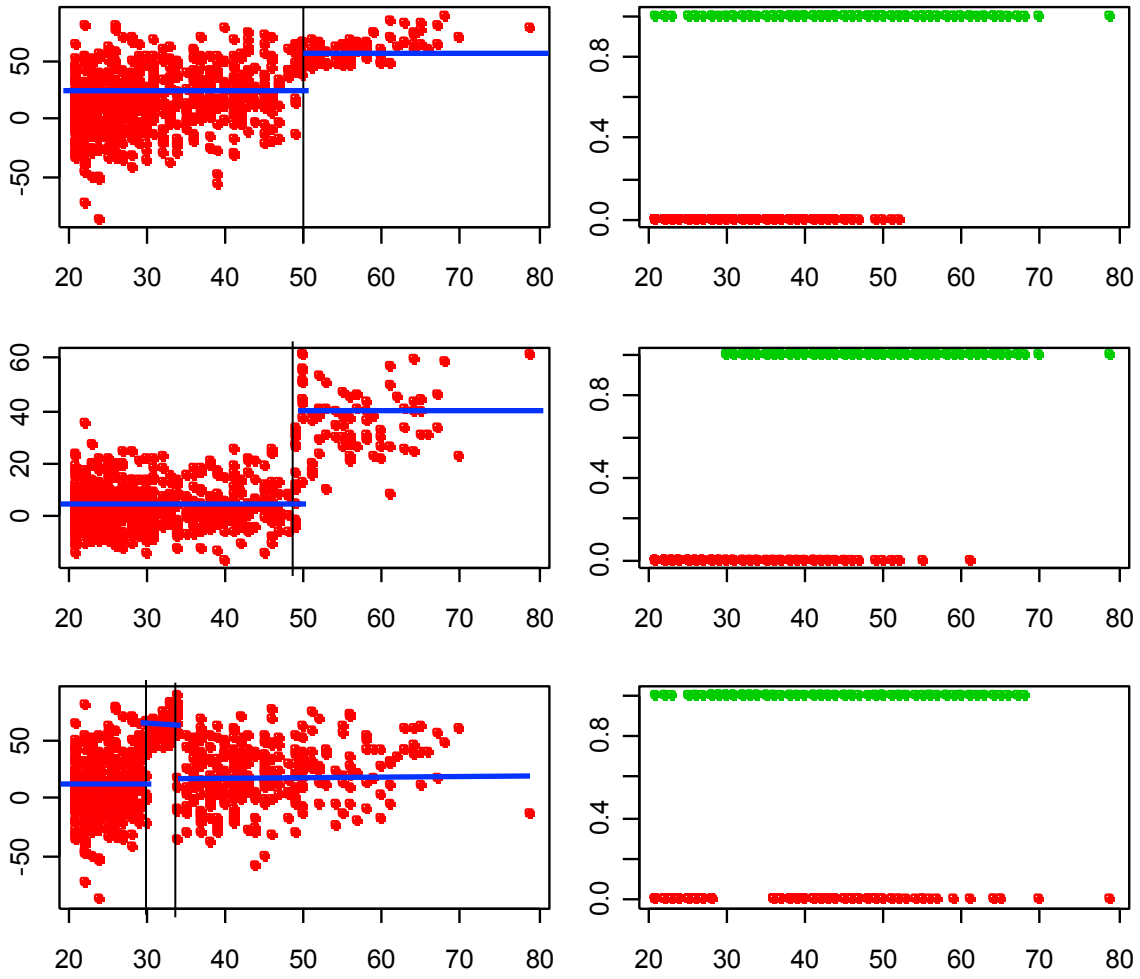
$T = (p-\pi)/\sigma_p$ measures how *interesting* a subset is.

Add a penalty term to prevent selection of subsets that are too small or too large.

ARF algorithm diagram



Comparing CART & ARF



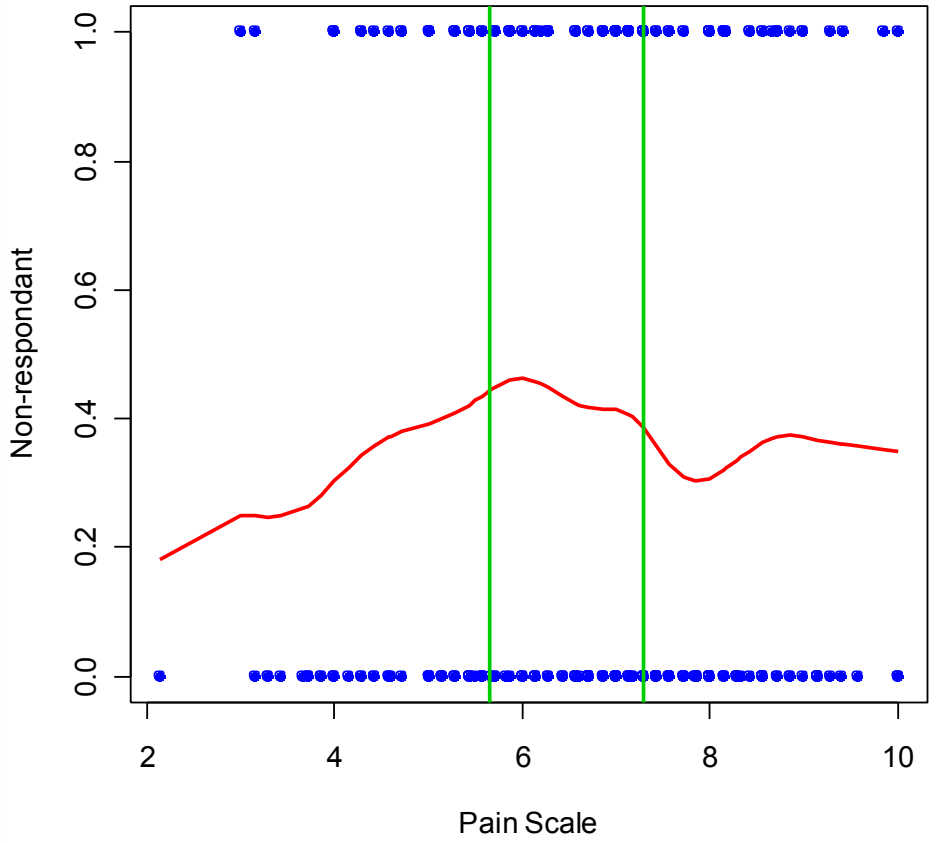
ARF: Captures subset with small variance (but not the rest).

CART Needs both subset with small variance relative to mean diff.

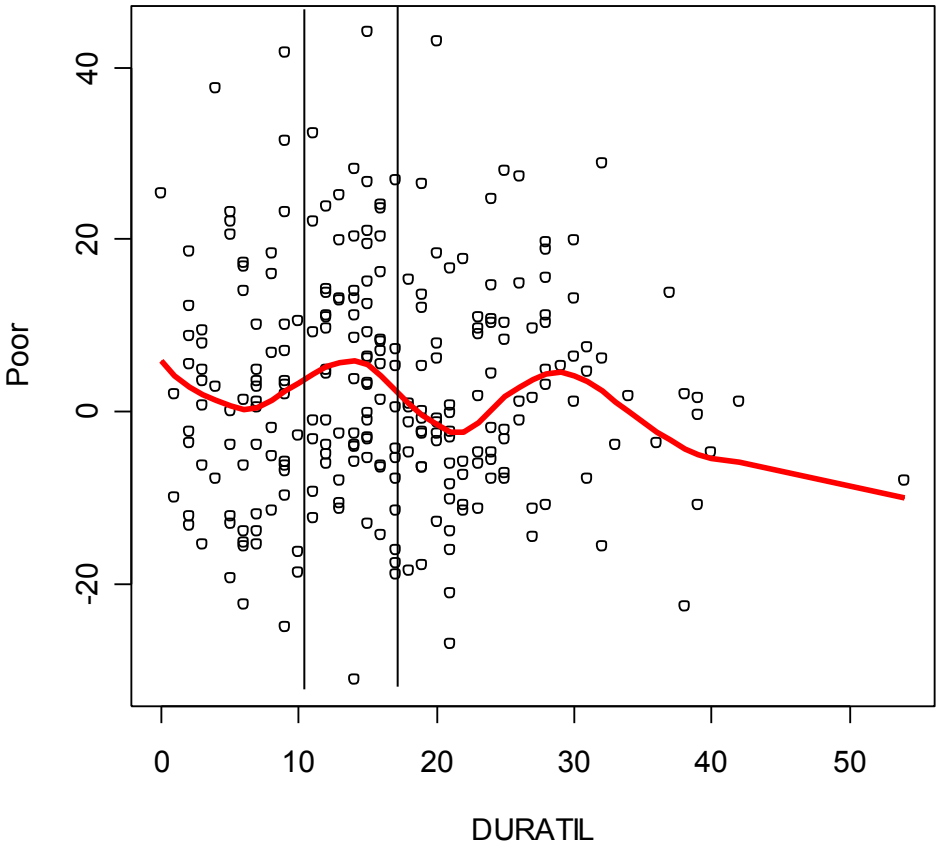
ARF: captures interior subsets.

Two Examples

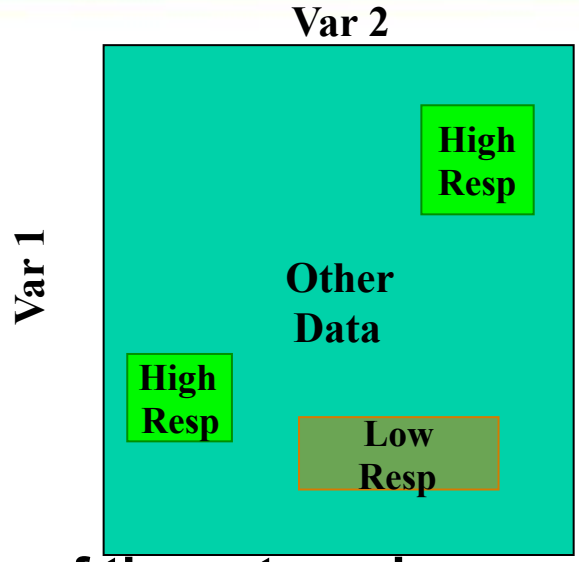
Subset that are hidden in the middle



Point density is important



Methodology



1. **Methodology Objective:**

   The Data Space is divided between
   - **High response subsets**
   - **Low Response subsets**
   - **Other**

2. **Categorical Responses:**

   Subsets that have high response on one of the categories.

   $$T = (p-\pi)/\sigma_p$$

3. **Continuous Responses: High mean response measured by**

   $$Z = (\bar{x} - \mu) / \sigma_{\bar{x}}$$

4. **Statistical significance should be based on the entire tree building process.**
5. **Categorical Predictors**
6. **Data Visualization**
7. **PDF report.**

Report

Simple Tree or Tree sketch : Only statistically significant nodes.

Full Tree: All nodes.

Table of Numerical Outputs: Detailed statistics of each node

List of Interesting Subsets: List of significant subsets

Conditional Scatter Plot (optional): Data Visualization.

How about outliers?

For Regression trees

- Popular belief: Trees are not affected by outlier (are robust)
- Outlier detection:

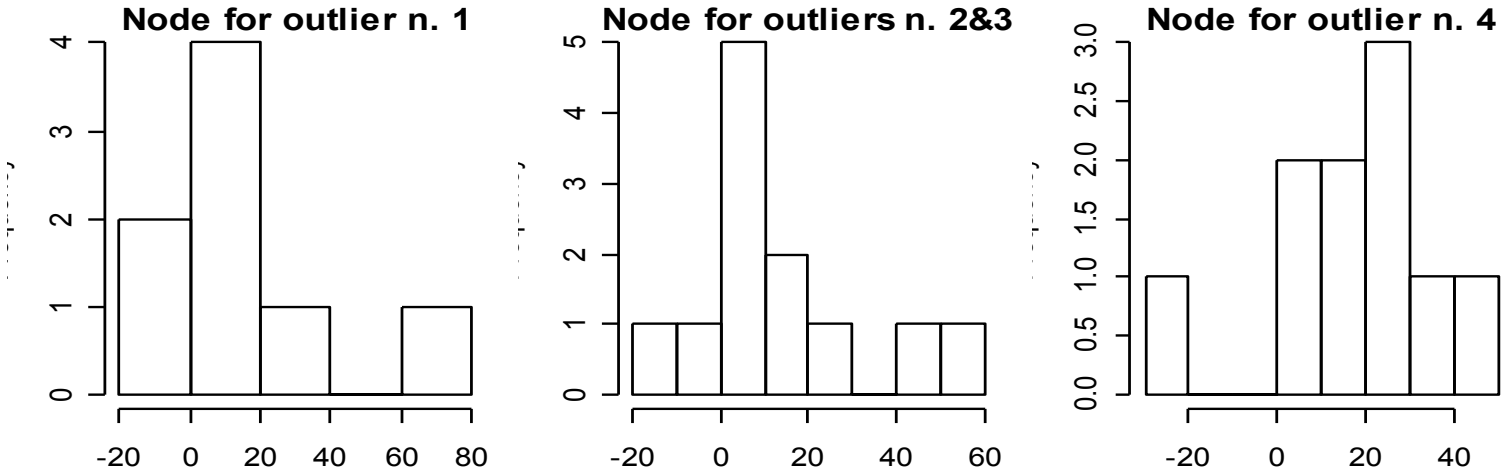
Run the data mining tree allowing for small buckets.

For observation X_i in terminal node j calculate the score

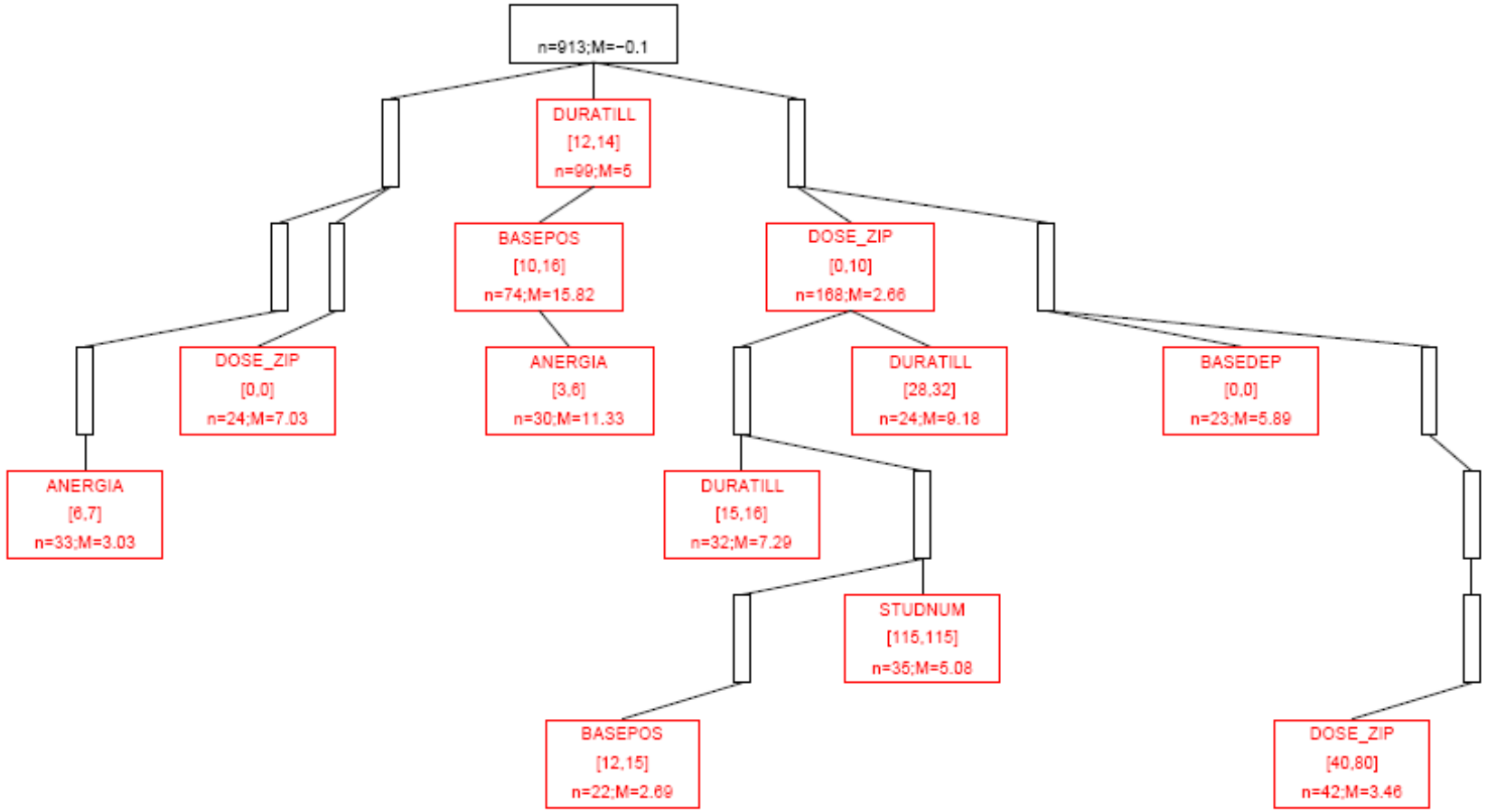
$$Z_i = \frac{|X_i - Median|}{MAD}$$

Z_i is the number of std dev away from the mean

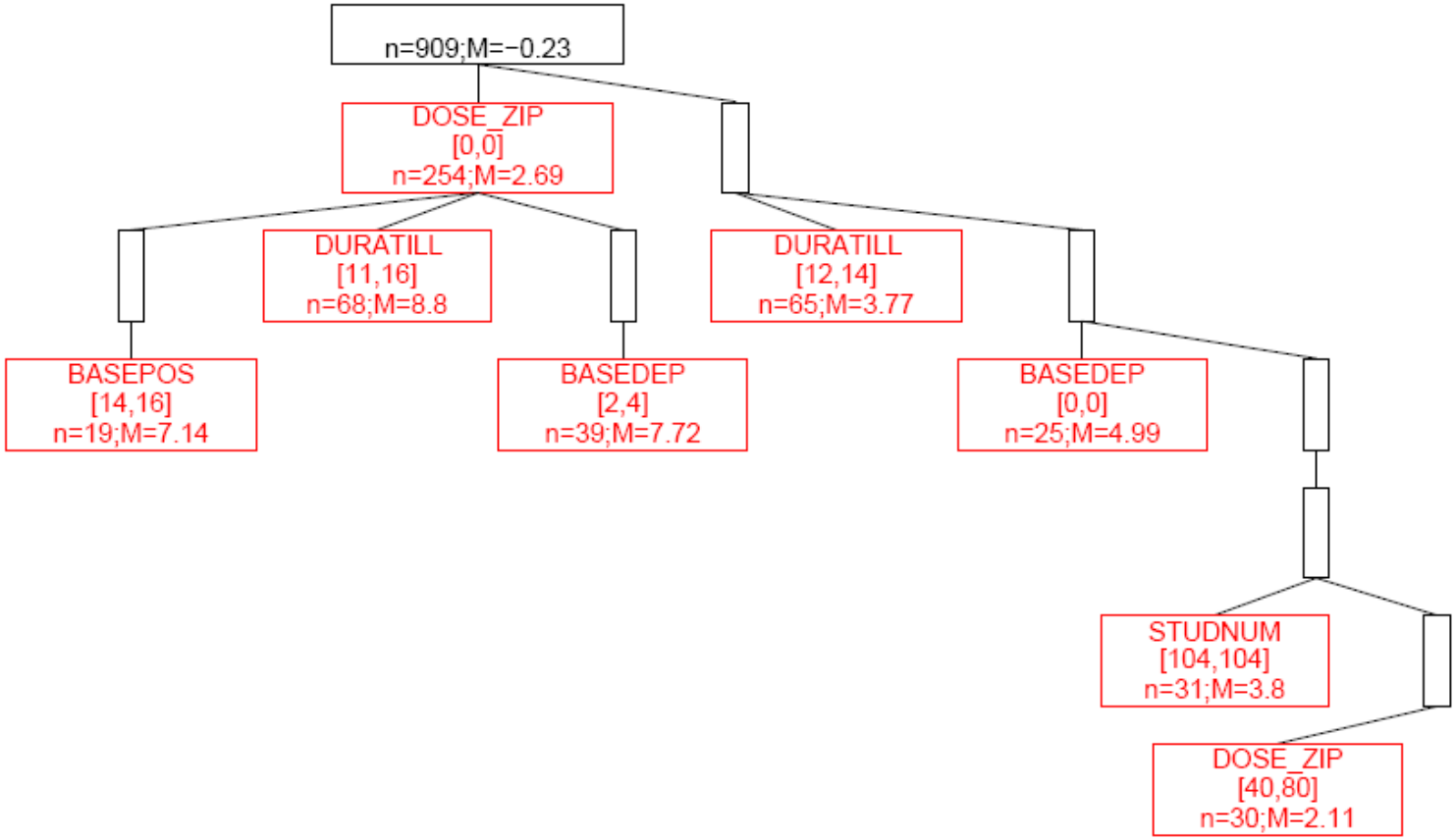
$Z_i > 3.5$ then X_i is noted as an outlier.



Tree with Outliers



After Outlier removal



Robustness issues

ISSUE

In regulatory environments outliers are rarely omitted.
Our method is easily adaptable to robust splits by calculating the robust version of the criterion by replacing the mean and std dev by suitable estimators of location and scale:

$$Z = (T - \mu_T^R) / \sigma_T^R$$

Binary/Categorical Response

- How do we think about Robustness of trees?
- One outlier might not make any difference.
- 5% , 10% or more outliers could make a difference.

Further work

Binary/Categorical Response

- How do we think about Robustness of trees?
- One outlier might not make any difference.
- 5% , 10% or more outliers could make a difference.

Alvir, Cabrera, Caridi and Nguyen (2006)
Mining Clinical Trial data.

R-package: www.rci.rutgers.edu/DM/ARF_1.0.zip