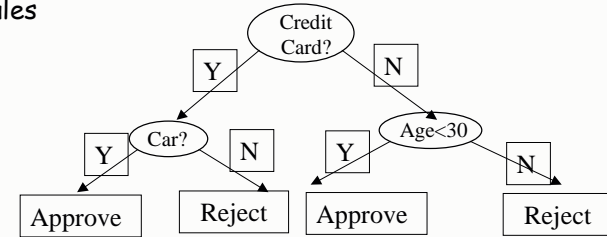


## Classification and Regression trees:

- CART
- BOOSTING AND BAGGING
- RANDOM FOREST
- Data Mining Trees: ARF

### Tree methods: Dependent variable is categorical

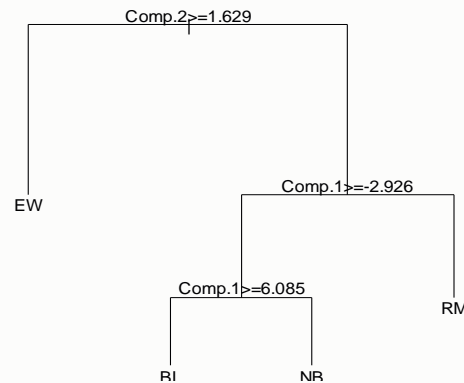
- Classification trees (e.g., CART, C5, Firm, Tree)
- Decision Trees
- Decision Rules



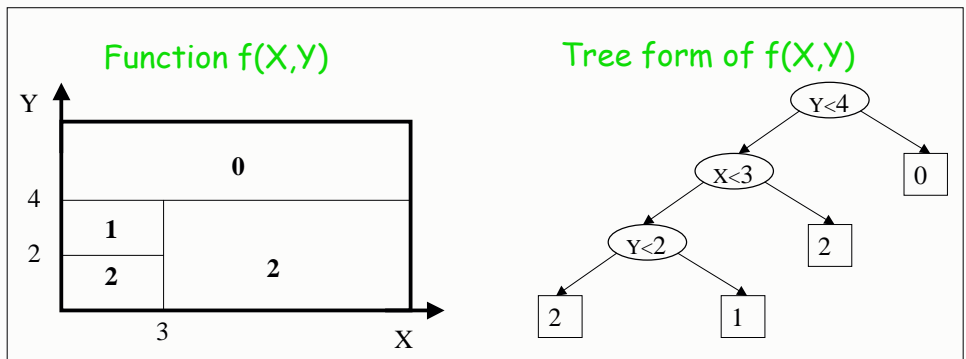
### Tree methods: Dependent variable is numeric

- Regression Trees

Classification tree for the cancer groups using 10 principal components of the top 100 cancer genes. The classification rule produces zero mistakes in the training set and five mistakes in the testing set.

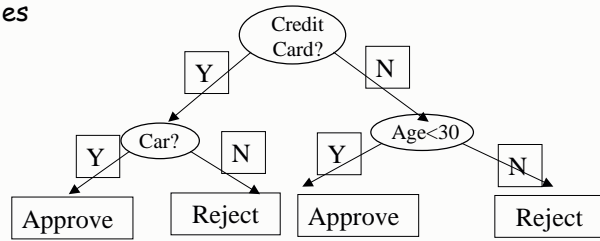


## Classification trees



Tree methods: Dependent variable is categorical

- Classification trees (e.g., CART, C5, Firm, Tree)
- Decision Trees
- Decision Rules

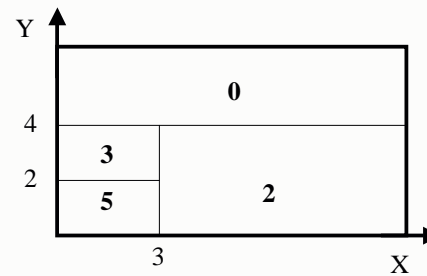


Tree methods: Dependent variable is numeric

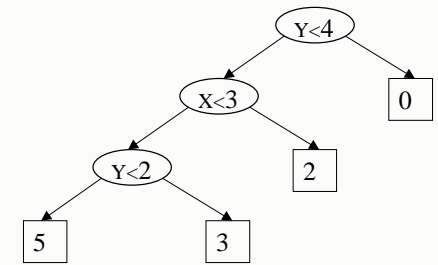
- Regression Trees

**Trees**

Function  $f(X,Y)$



Tree form of  $f(X,Y)$



**Classification & Regression Trees**

- Fit a tree model to data.
- Recursive Partitioning Algorithm.
- At each node we perform a split: we chose a variable  $X$  and a value  $t$  that minimizes a criteria.
- The split:  $L = \{X < t\}$  ;  $R = \{X \geq t\}$

- For regression trees two criteria functions are:

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

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

- For classification trees: criteria functions

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

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

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

**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 codes 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.table("project2/hospital.txt", sep=",")
colnames(hospital) <-
  c("ZIP", "HID", "CITY", "STATE", "BEDS", "RBEDS", "OUTV", "ADM", "SIR",
    "SALESY", "SALES12", "HIP95", "KNEE95", "TH", "TRAUMA", "REHAB", "HIP96",
    "KNEE96", "FEMUR96")

hosp = hospital[,-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)

```

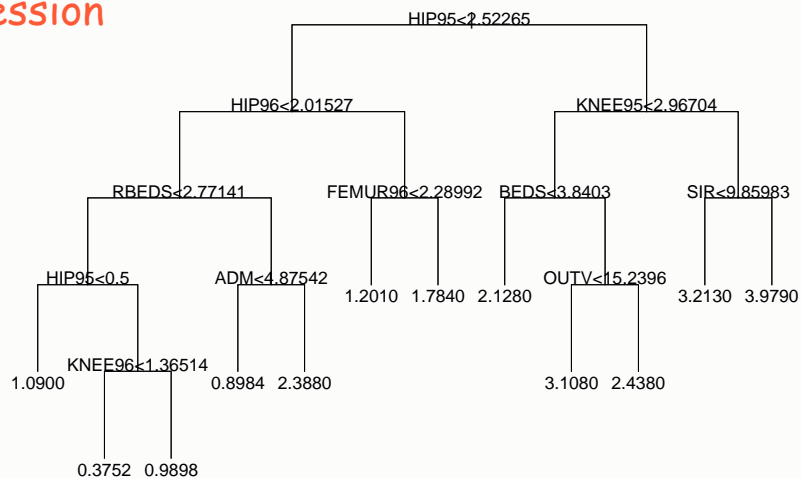
## Regression Tree for log(1+Sales)

```

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

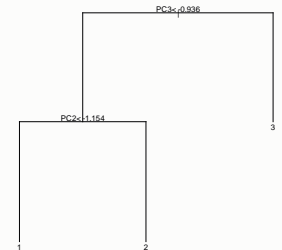
```

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)

1. Select  $n_{tree}$ , the number of trees to grow, and  $m_{try}$ , a number no larger than number of variables.
2. For  $i = 1$  to  $n_{tree}$ :
3. Draw a bootstrap sample from the data. Call those not in the bootstrap sample the "out-of-bag" data.
4. Grow a "random" tree, where at each node, the best split is chosen among  $m_{try}$  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`

## Boosting (Ada boosting)

Input:

Data  $(x_i, y_i) \ i=1, \dots, n$  ;  $w_i = 1/n$

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  $i=1, \dots, n$  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((hosp[-c(1:4,10:11),]))
ylearn = 1*( hosp$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)
```

## Paradigm for data mining: Selection of interesting subsets

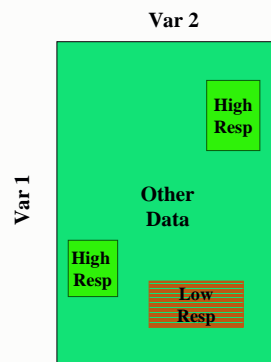
### Recursive Partition:

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

Data Mining Trees

### Bump Hunting:

- Find subsets that optimize some criterion.
- Subsets are more "robust"
- Not all interesting subsets are found



## Data Mining Trees: ARF

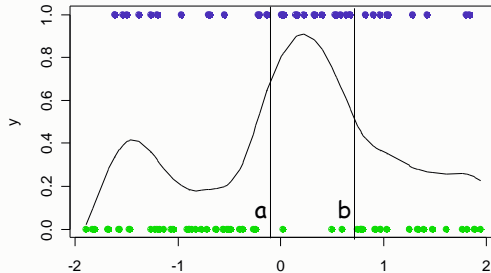
SPLIT FOR CONTINUOUS DESCRIPTORS

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

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

is much larger than

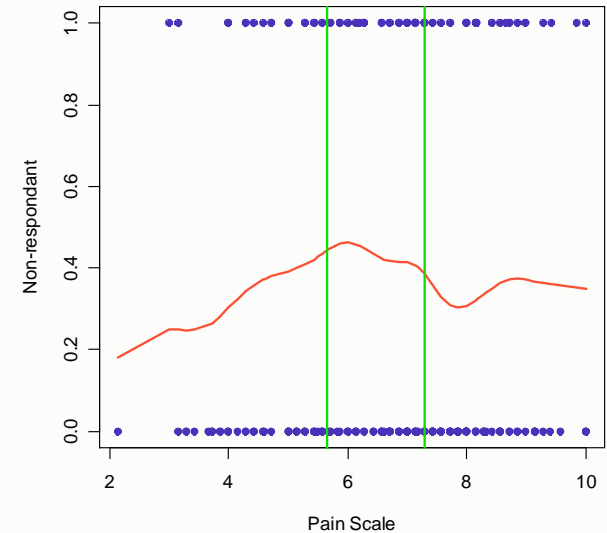
$$\pi = \text{Prob}[Z=1]$$



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

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

## Example from a pain study



## Data mining tree(ARF)

Method: Select the variable and subset that maximizes

$$T = (p - \pi) / \sigma_p \quad (+ \lambda \min \{ \log(h), \log(fN) \} / \log(fN))$$

where  $\lambda$  and  $f$  are a prespecified constants and  $h$  is the number of observations within the interval.

Iteration: Iterate the process (like growing a classification tree) a few times until no significant intervals are found.

Minimum bucket size: 15-20 cases  
(5 is recommended by CART but it is too small)

Continuous response: subsets with high mean or high median

Categorical Predictors: Split into groups

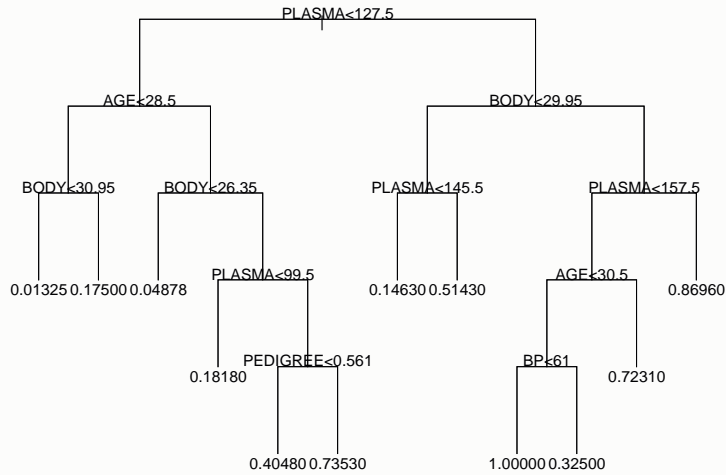
## Case Study: Pima Indians Diabetes

- 768 Pima Indian females, 21+ years old
- 268 tested positive to diabetes

Variables:

- PRG: Number of times pregnant
- PLASMA: Plasma glucose concentration in saliva
- BP: Diastolic Blood Pressure
- THICK: Triceps skin fold thickness
- INSULIN: Two hours serum insulin
- BODY: Body mass index (Weight/Height)
- PEDIGREE: Diabetes pedigree function
- AGE: In years
- RESPONSE: 1: Diabetes, 0: Not

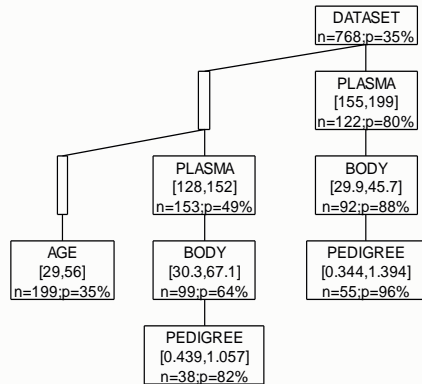
# CART Tree



# CART Tree

- 1) root 768 174.500 0.34900
- 2) PLASMA < 127.5 485 75.780 0.19380
- 4) AGE < 28.5 271 21.050 0.08487
- 8) BODY < 30.95 151 1.974 0.01325 \*
- 9) BODY >= 30.95 120 17.320 0.17500 \*
- 5) AGE >= 28.5 214 47.440 0.33180
- 10) BODY < 26.35 41 1.902 0.04878 \*
- 11) BODY >= 26.35 173 41.480 0.39880
- 22) PLASMA < 99.5 55 8.182 0.18180 \*
- 23) PLASMA >= 99.5 118 29.500 0.50000
- 46) PEDIGREE < 0.561 84 20.240 0.40480 \*
- 47) PEDIGREE >= 0.561 34 6.618 0.73530 \*
- 3) PLASMA > 127.5 283 67.020 0.61480
- 6) BODY < 29.95 76 16.420 0.31580
- 12) PLASMA < 145.5 41 5.122 0.14630 \*
- 13) PLASMA > 145.5 35 8.743 0.51430 \*
- 7) BODY >= 29.95 207 41.300 0.72460
- 14) PLASMA < 157.5 115 27.390 0.60870
- 28) AGE < 30.5 50 12.420 0.46000
- 56) BP < 61 10 0.000 1.00000 \*
- 57) BP >= 61 40 8.775 0.32500 \*
- 29) AGE >= 30.5 65 13.020 0.72310 \*
- 15) PLASMA > 157.5 92 10.430 0.86960 \*

# Data mining tree



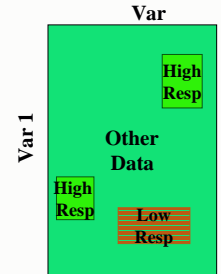
	Subset	%Success	n
1	PLASMA in [155,199] & BODY in [29.9,45.7] & PEDIGREE in [0.344,1.394]	96.364	55
2	PLASMA in [128,152] & BODY in [30.3,67.1] & PEDIGREE in [0.439,1.057]	81.579	38
3	PLASMA in [0,127] & AGE in [29,56]	35.176	199

# 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 several categories. The categorical response is converted into several Binary responses.

## 3. Continuous Responses: High mean or low mean response

## 4. Categorical Predictors: Two groups.

## 5. Data Visualization:

## 6. PDF report:

## Report

**Simple Tree:** Only statistically significant nodes.

**Full Tree:** All nodes.

**Table of Numerical Outputs:** Detailed statistics of each node

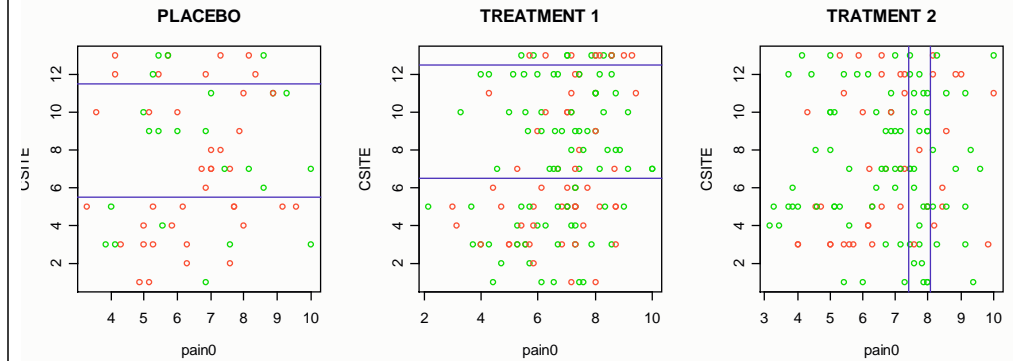
**List of Interesting Subsets:** List of significant subsets

**Conditional Scatter Plot (optional):** Data Visualization.

[See file](#)

## Data Visualization

**Conditional Plot:** Condition on one or two variables.



## How to Use it

1. Import the data into R:

```
library(foreign)
w <- read.xport("C:/crf155.xpt")
```

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

```
sapply(w, is.factor)
w$x = factor(w$x)
```

3. Recode factors: (sometimes),

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

4. Run ARF

```
mod = f.arf(RSP30 ~ pain0+CSITE+RXGP, data=w,
            highresp=c("0", "1"))
f.report(mod, file="c:/report.pdf")
```

## How to Use it

4. More options

```
mod1 = f.arf(RSP30 ~ pain0+Wk1chnng+RXGP, data=w,
            highresp="1",
            varlist= c("RXGP", "Wk1chnng", "RXGP"))
f.report(mod1, file="c:/replrstmeasure.pdf")
```

[See file](#)

5. More options

```
mod2 = f.arf(RESPONSE ~ SEX + BBPRS + ANERGIA + SMOKEYN +
            BCGIS + MNBARNES + MNAIMS + dose, data = all,
            highresp = c("YES", "NON", "ICR"))
f.report(mod2, file="c:/rep2Bprs.pdf")
```

[See file](#)