Introduction to Random Forests for Gene Expression Data

Utah State University – Spring 2012

STAT 5570: Statistical Bioinformatics

Notes 3.4

References

- Breiman, Machine Learning (2001) 45(1): 5-32.
- Diaz-Uriarte and Alvarez de Andres, BMC Bioinformatics (2006) 7:3.
- Cutler, Cutler, and Stevens (2009). In Li and Xu, editors, *High-Dimensional Data Analysis in Cancer Research*, pages 83-101.

Gene Profiling / Selection

- “Observe” gene expression in different conditions – healthy vs. diseased, e.g.
- Use simultaneous expression “profiles” of thousands of genes (what are the genes doing across arrays)
- Look at which genes are “important” in “separating” the two conditions; i.e., what determines the conditions’ “signatures”

Machine Learning

- Computational & statistical inference processes:
  observed data → reusable algorithms for prediction
- Why "machine"?
  want minimal: human involvement
- Why "learning"?
  develop ability to predict
- Here, supervised learning:
  use knowledge of condition type

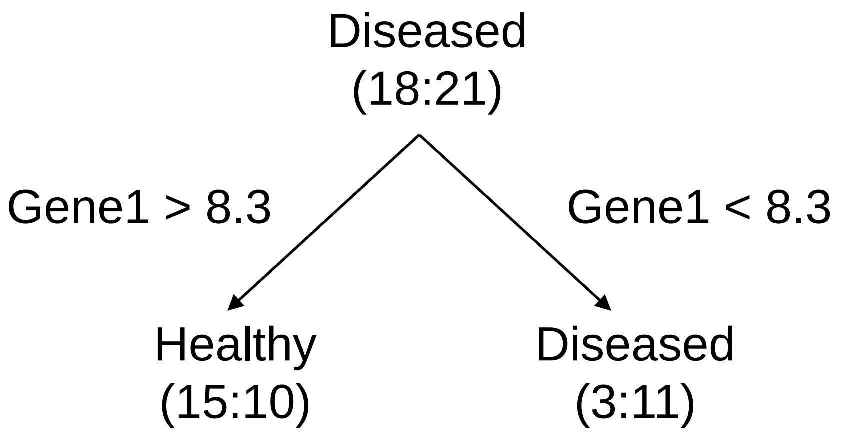
Machine Learning Methods

- Neural Network
- SVM (Support Vector Machine)
- RPART (Recursive PArtitioning and Regression Trees)
- CART (Classification and Regression Trees)
- Ensembling Learning (average of many trees)
  - Boosting (Shapire et al., 1998)
  - Bagging (Breiman, 1996)
  - RandomForests (Breiman, 2001; Cutler & Stevens 2006; Cutler, Cutler, & Stevens 2008)

CART: Classification and Regression Trees

- Each individual (array) has data on many predictors (genes) and one response (disease state)

- Think of a tree, with splits based on levels of specific predictors



- Choose predictors and split levels to maximize “purity” in new groups; the best split at each node

- Prediction made by:

CART generalized: Random Forests

- Rather than using all predictors and all individuals to make a single tree, make a forest of many (**`ntree`**) trees, each one based on a random selection of predictors and individuals

- Each tree is fit using a bootstrap sample of data
  (draw with replacement)
  and ‘grown’ until each node is ‘pure’

- Each node is split using the best among a subset (of size **`mtry`**) of predictors randomly chosen at that node (default is sqrt. of # of predictors)
  (special case using all predictors: bagging)

- Prediction made by aggregating across the forest
  (majority vote or average)

How to measure "goodness"?

- Each tree fit on a "training" set (bootstrap sample), or the "bag"
- The left-over cases ("out-of-bag") can be used as a "test" set for that tree

  (usually 1/3 of original data)
- The "out-of-bag" (OOB) error rate is the:

What does RF give us?

- Kind of a “black box”
  – but can look at “variable importance”
- For each tree, look at the OOB data:
  - Permute values of predictor j among all OOB cases
  - Pass OOB data down the tree, save the predictions
  - For case i of OOB and predictor j , get:
    OOB error rate with variable *j* permuted **–**
    OOB error rate before permutation
- Average across forest to get overall variable importance for each predictor j

Why “variable importance”?

- Recall: identifying conditions’ signatures
- Sort genes by some criterion
- Want smallest set of genes to achieve good diagnostic ability

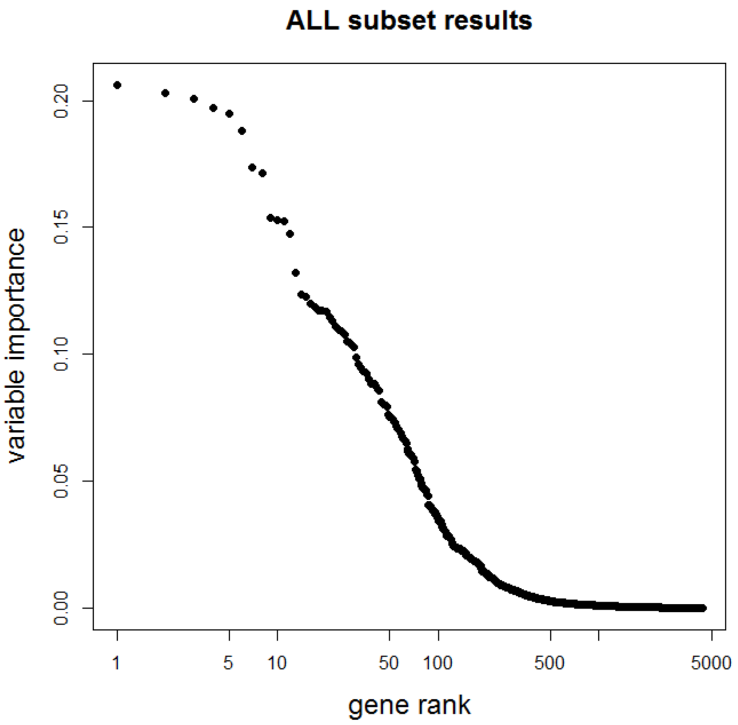
Recall – ALL Data

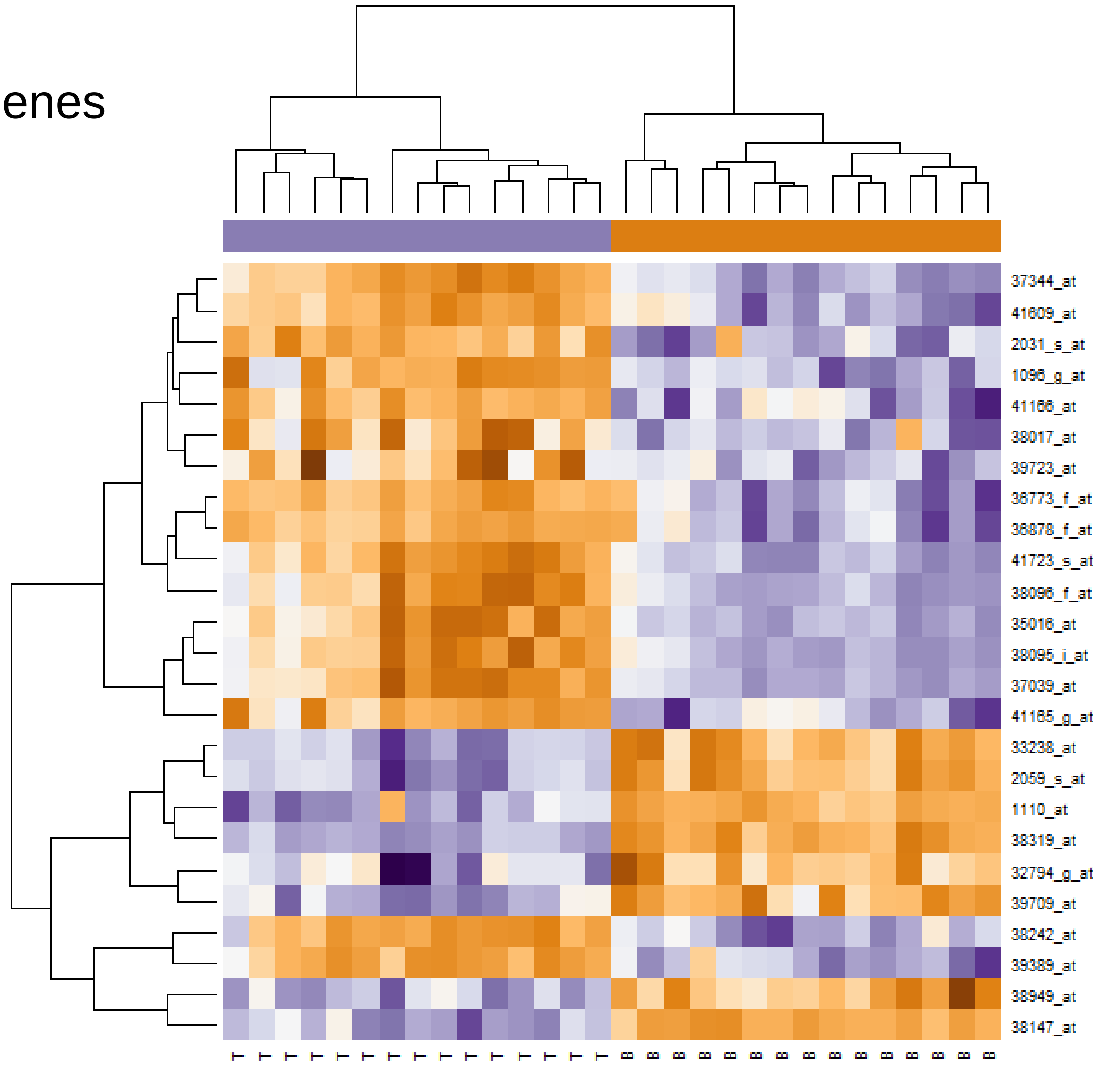
- RMA-preprocessed gene expression data
- Chiaretti et al., Blood (2004) 103(7)
- 12625 genes (hgu95av2 Affymetrix GeneChip)
- 128 samples (arrays)
- phenotypic data on all 128 patients, including:
  - 95 B-cell cancer
  - 33 T-cell cancer

ALL subset: 4,400 genes, 30 arrays (15 B, 15 T)

Look at top 25 most important genes from Random Forest.

Color scale:
purple (low) to orange (high)



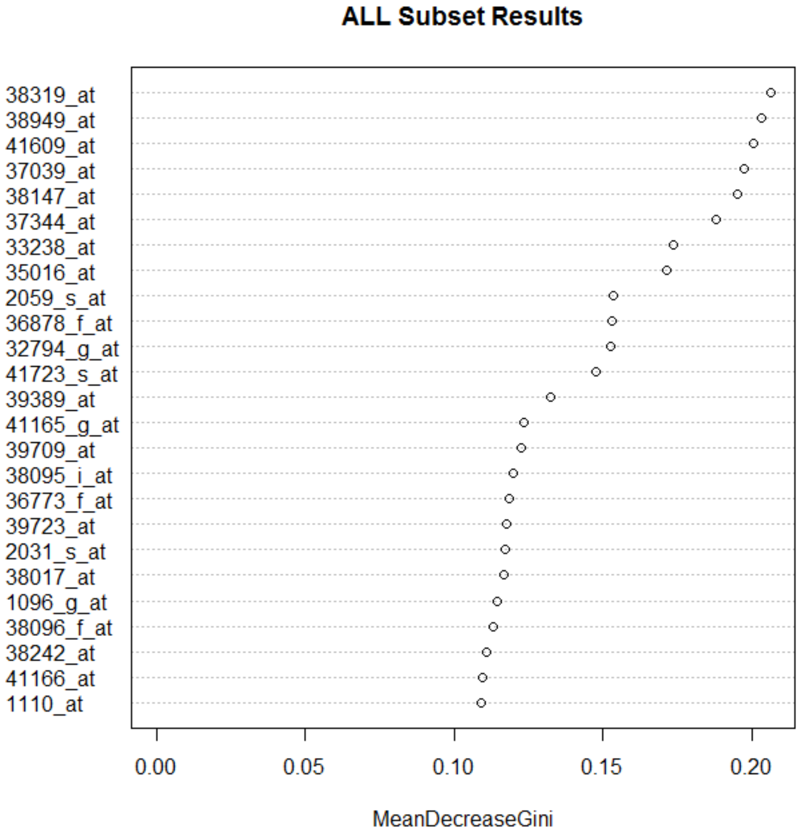


```
memory.limit(size=4000)

# load data
library(affy); library(ALL); data(ALL)

# obtain relevant subset of data; similar to slide 11 of Notes 3.2
# (here, filter genes on raw scale, then return to log scale)
# also, keep 30 arrays here JUST for computational convenience
#  (in-class)
library(genefilter); e.mat <- 2^(exprs(ALL)[,c(81:110)])
ffun <- filterfun(pOverA(0.20,100))
t.fil <- genefilter(e.mat,ffun)
small.eset <- log2(e.mat[t.fil,])
dim(small.eset) # 4400 genes, 30 arrays (15 B and 15 T)
group <- c(rep('B',15),rep('T',15)) # classification, in order

# One RF
library(randomForest)
set.seed(1234)
print(date())
rf <- randomForest(x=t(small.eset),y=as.factor(group),ntree=10000)
print(date()) # about 25 seconds
```

```
# Look at variable importance
imp.temp <- abs(rf$importance[,])
t <- order(imp.temp,decreasing=TRUE)
plot(c(1:nrow(small.eset)),imp.temp[t],log='x',cex.main=1.5,
   xlab='gene rank',ylab='variable importance',cex.lab=1.5,
   pch=16,main='ALL subset results')

# Get subset of expression values for 25 most 'important' genes
gn.imp <- names(imp.temp)[t]
gn.25 <- gn.imp[1:25] # vector of top 25 genes, in order
t <- is.element(rownames(small.eset),gn.25)
sig.eset <- small.eset[t,]
  # matrix of expression values, not necessarily in order

## Make a heatmap, with group differences obvious on plot
library(RColorBrewer)
hmcol <- colorRampPalette(brewer.pal(11,"PuOr"))(256)
colnames(sig.eset) <- group # This will label the heatmap columns
csc <- rep(hmcol[50],30)
csc[group=='T'] <- hmcol[200]
  # column side color will be purple for T and orange for B
heatmap(sig.eset,scale="row", col=hmcol,ColSideColors=csc)
```

A Better Variable Importance Plot

```
varImpPlot(rf, n.var=25, main='ALL Subset Results')

imp.temp <- importance(rf)
t <- order(imp.temp,decreasing=TRUE)
gn.imp <- names(imp.temp)[t]
```



Can focus on “variable selection”

- Iteratively fit many forests, each time discarding predictors with low importance from previous iterations

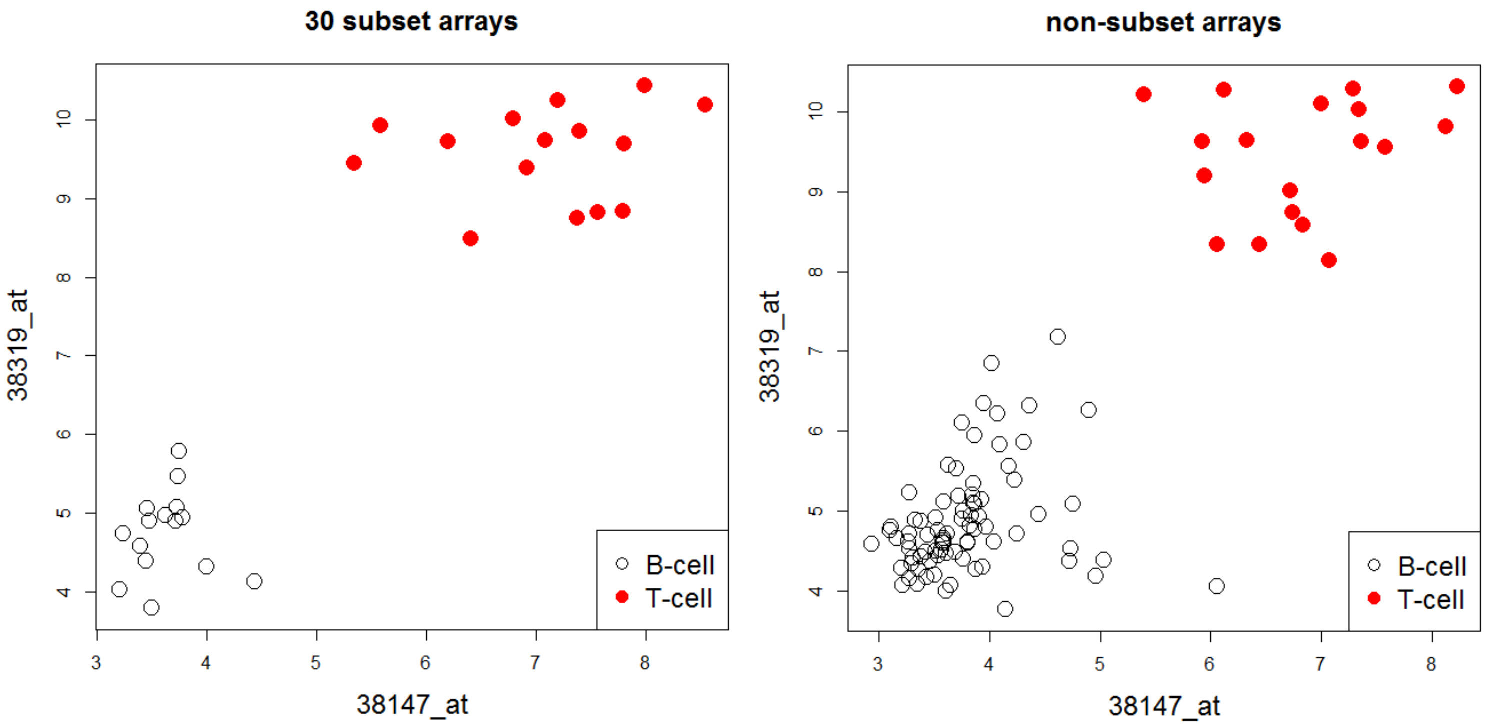
- Use bootstrap to assess standard error of error rates

- Choose the forest with the smallest number of genes whose error rate is within *u* standard errors of the minimum error rate of all forests (*u* = 0 or 1, typically)

- Reference: Diaz-Uriarte and Alvarez de Andres, BMC Bioinformatics (2006) 7:3.

- Online tool: http://genesrf.bioinfo.cnio.es

RF Variable Selection on ALL subset



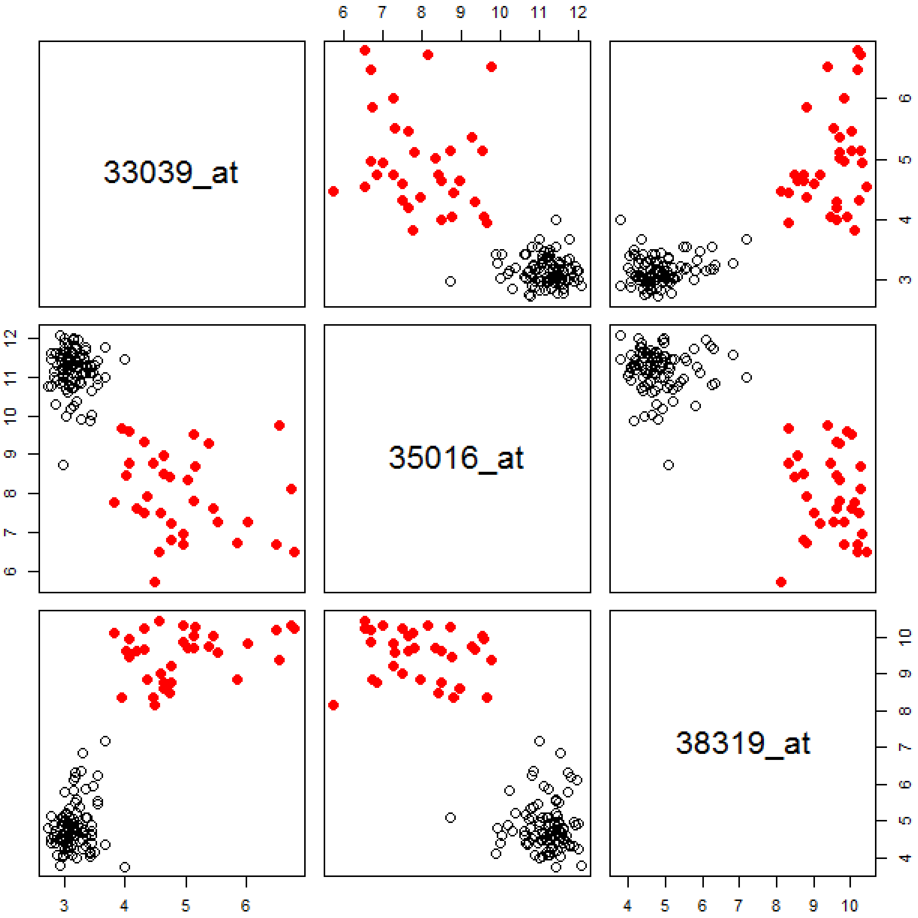
Subset: 30 arrays, 4400 genes

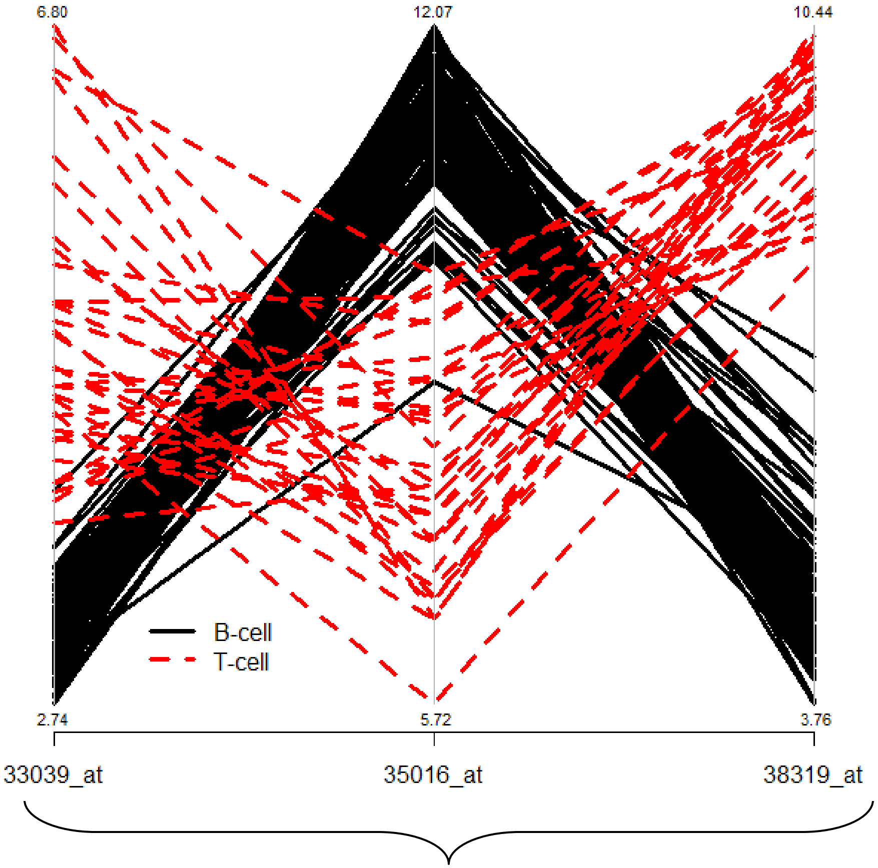
First forest: 10,000 trees; Subsequent forests: 2,000 trees

At each iteration, drop 20% of variables (genes) until a 2-variable model is built; return the best of the series of models considered.

```
# Look at variable selection
library(varSelRF)
set.seed(1234)
print(date())
rfsel <- varSelRF(t(small.eset),as.factor(group),
 ntree=10000, ntreeIterat=2000, vars.drop.frac=0.2)
 # 50 seconds
print(date())
rf.sig.gn <- rfsel$selected.vars # "38147_at" "38319_at"

# Visualize these two genes
exp.gn.1 <- small.eset[rownames(small.eset)==rf.sig.gn[1],]
exp.gn.2 <- small.eset[rownames(small.eset)==rf.sig.gn[2],]
use.pch <- c(rep(1,15),rep(16,15)) # Define plotting characters
use.col <- c(rep(1,15),rep(2,15)) # Define plotting colors
plot(exp.gn.1,exp.gn.2,col=use.col,main='30 subset arrays',
  cex.main=1.5, cex.lab=1.5, xlab=rf.sig.gn[1], ylab=rf.sig.gn[2],
  pch=use.pch,cex=2)
legend('bottomright',
       c('B-cell','T-cell'),pch=c(1,16),col=c(1,2),cex=1.5)

# Note that set.seed(123) leads to two other genes:
#  "2059_s_at" "38319_at"
# Also: rfsel$firstForest is the same as the slide 13 rf object
# Did this overfit these 30 arrays?
# Look at the other 98 (first 80 are B-cell, last 18 are T-cell)
eset.2 <- exprs(ALL)[,c(1:80,111:128)]
group.2 <- c(rep(0,80),rep(1,18))
exp.gn.1 <- eset.2[rownames(eset.2)==rf.sig.gn[1],]
exp.gn.2 <- eset.2[rownames(eset.2)==rf.sig.gn[2],]
use.pch.2 <- c(rep(1,80),rep(16,18))
use.col.2 <- c(rep(1,80),rep(2,18))
plot(exp.gn.1,exp.gn.2,col=use.col.2, main='non-subset arrays',
  cex.main=1.5,cex=2,cex.lab=1.5, xlab=rf.sig.gn[1],
  ylab=rf.sig.gn[2], pch=use.pch.2)
legend('bottomright',
       c('B-cell','T-cell'),pch=c(1,16),col=c(1,2),cex=1.5)
```

RF Variable Selection on ALL (full)





Each condition has a profile / signature (across these genes)

full ALL data: 12,625 genes, 128 arrays

```
# RF variable selection with full data set

# set seed and define initial objects
set.seed(123)
eset <- exprs(ALL) # 12625 genes, 128 arrays
cell <- c(rep(0,95),rep(1,33))
  # first 95 are B-cell; last 33 are T-cell

print(date())
rf.big <- varSelRF(t(eset),as.factor(cell),
  ntree=10000, ntreeIterat=2000, vars.drop.frac=0.2)
print(date())  # about 14 minutes

rf.gn <- rf.big$selected.vars
  #   "33039_at" "35016_at" "38319_at"
# make scatterplot matrix, with points colored by cell type
rf.eset <- t(eset[is.element(rownames(eset),rf.gn),])
use.pch <- c(rep(1,95),rep(16,33))
use.col <- cell+1
pairs(rf.eset,col=use.col,pch=use.pch,cex=1.5)


# Now - make a profile plot
t <- is.element(rownames(eset),rf.gn)
rf.eset <- eset[t,]
library(MASS)
parcoord(t(rf.eset), col=cell+1, lty=cell+1, lwd=3, var.label=TRUE)
legend(1.2,.15,c('B-cell','T-cell'),lty=c(1,2),
       lwd=3,col=c(1,2),bty='n')
```

Summary: RF for gene expression data

- Works well even with: many more variables than observations, many-valued categoricals, extensive missing values, badly unbalanced data
- Low error (comparable w/ boosting and SVM)
- Robustness (even with large “noise” in genes)
- Does not overfit
- Fast, and invariant to monotone transformations of the predictors
- Free! (Fortran code by Breiman and Cutler)
- Returns the importance of predictors (gene)
- Little need to tune parameters
- But, no: