Introduction to Random Forests for High-Dimensional Data

Utah State University – Fall 2019

Statistical Bioinformatics (Biomedical Big Data)

Notes 8

References

- Breiman, Machine Learning (2001) 45(1): 5-32.
- Diaz-Uriarte and Alvarez de Andres, BMC Bioinformatics (2006) 7:3.
- Cutler, Cutler, and Stevens (2012) Random Forests. In Zhang and Ma, editors, *Ensemble Machine Learning: Methods and Applications*, pp. 157-175.

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

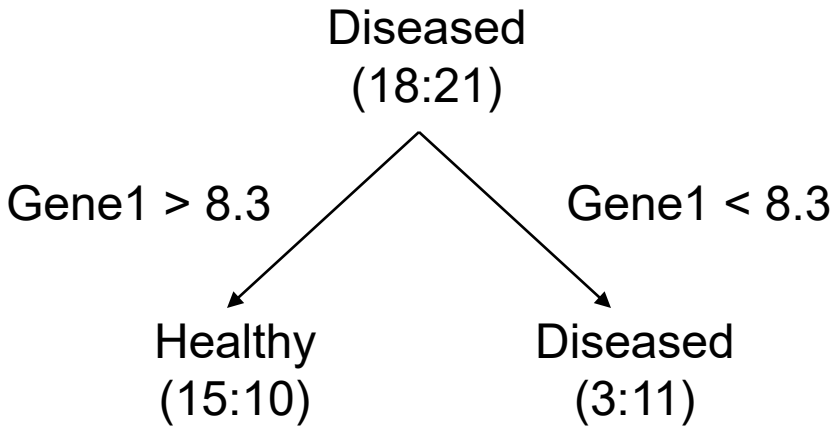
- Computational & statistical inference processes:
  observed data → reusable algorithms for prediction
- Why "machine"?
  want minimal: human involvement
- Why "learning"?
  develop ability to predict
- Here, supervised learning:
  use knowledge of condition type

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: passing test cases down tree

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:
  % misclassification

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 Naples RNA-Seq Example

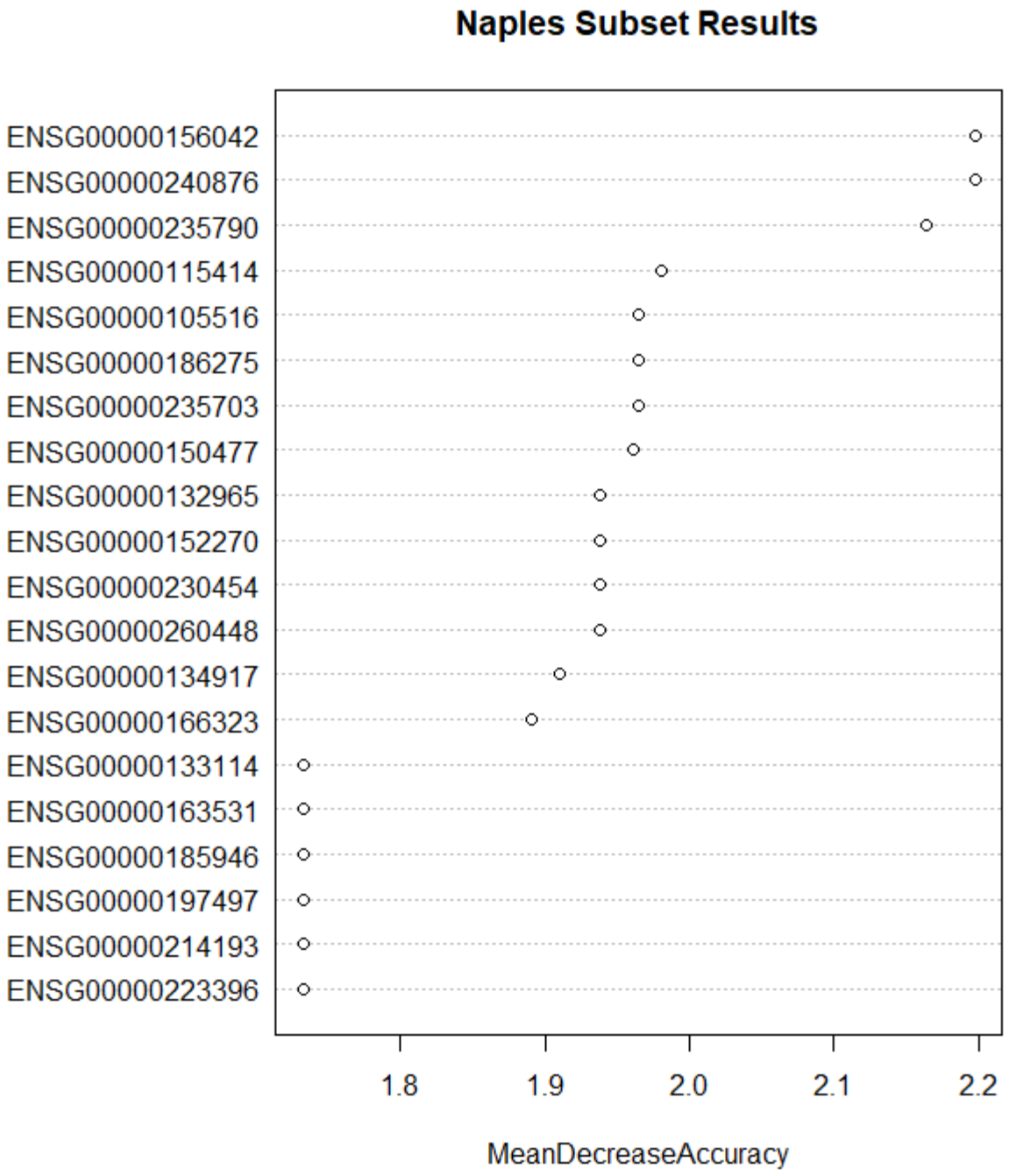
- 8 heart tissue samples, 56,620 genes
  - 4 control (no heart disease)
  - 4 cardiomyopathy (heart disease)
    - 2 restrictive (contracts okay, relaxes abnormally)
    - 2 dilated (enlarged left ventricle)
- These "Naples" data made public Nov 2015 by Institute of Genetics and Biophysics (Naples, Italy)

	Ctrl_3	RCM_3	Ctrl_4	DCM_4	Ctrl_5	RCM_5	Ctrl_6	DCM_6
ENSG00000000003	308	498	362	554	351	353	220	309
ENSG00000000005	3	164	2	43	13	83	22	16
ENSG00000000419	1187	1249	1096	1303	970	863	637	684
ENSG00000000457	163	239	168	195	153	194	44	117
ENSG00000000460	63	108	83	109	87	43	54	51
ENSG00000000938	369	328	272	669	1216	193	861	292
...								

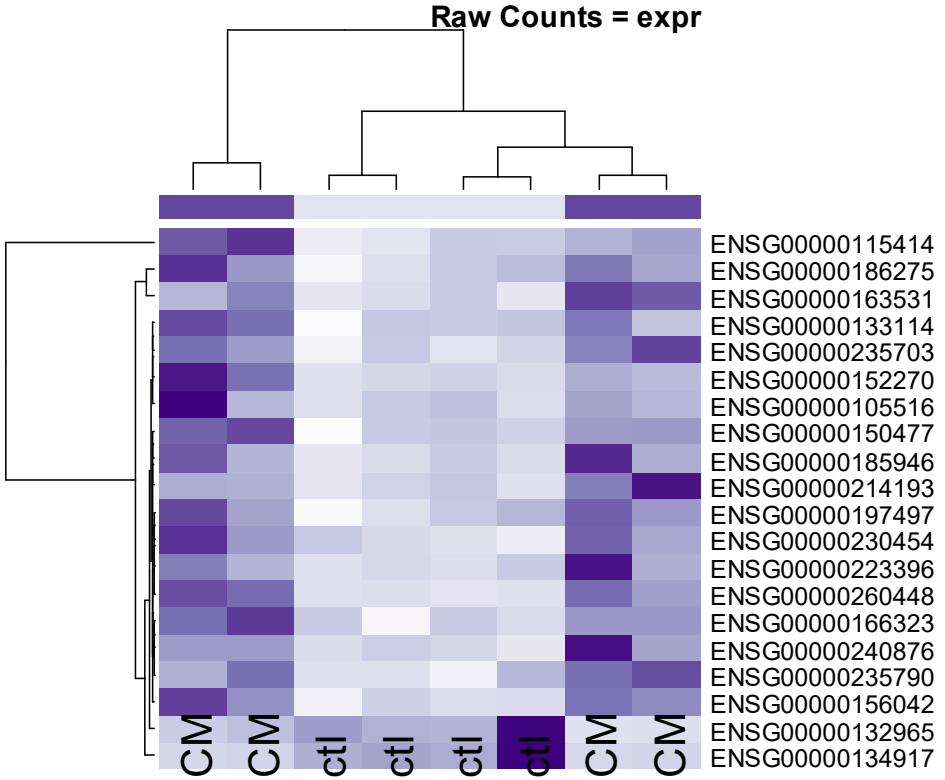
http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE71613

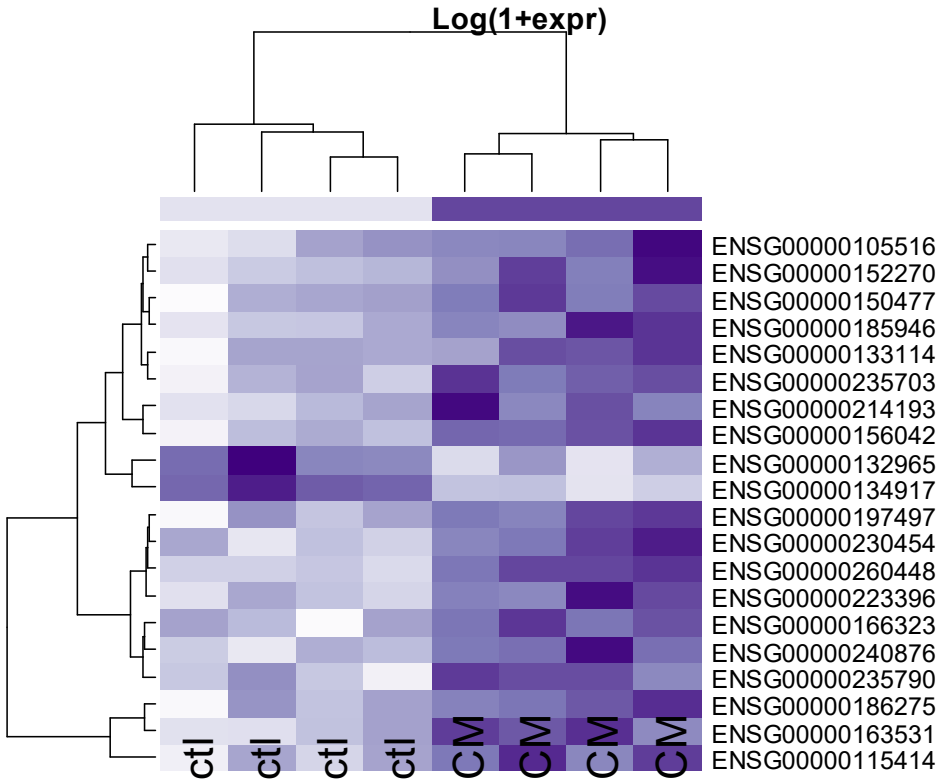
Naples subset:
13,878 genes,
8 patients
(4 ctl, 4 CM)

Look at top 20 most important genes from Random Forest.



Expression color scale:
dark (low) to light (high)





```
### First prepare objects for RF

# obtain expression estimates on the UN-LOGGED scale
url <- "http://www.stat.usu.edu/jrstevens/bioinf/naples.csv"
naples <- read.csv(url, row.names=1)
gn <- rownames(naples)
emat <- as.matrix(naples)

# filter -- (just as a working example) keep genes
# with count at least 10 on at least 2 samples
# and CV between .25 and 2.5
library(genefilter)
ffun <- filterfun(kOverA(2,10), cv(.25,2.5))
t.fil <- genefilter(emat,ffun)
# apply filter
eset.fil <- emat[t.fil,]
dim(emat)     # 56621  8
dim(eset.fil) # 13878  8

# define groups: control and cardiomyopathy (RCM or DCM)
group <- c('ctl','CM','ctl','CM','ctl','CM','ctl','CM')
```

```
# One RF
library(randomForest)
set.seed(1234)
print(date())
rf <- randomForest(x=t(eset.fil),y=as.factor(group),
      ntree=10000, importance=TRUE)
print(date()) # about 9 seconds

# Make variable importance plot
varImpPlot(rf, n.var=20, type=1,
 main='Naples Subset Results')

# Get names of most important genes
imp.temp <- importance(rf,type=1)
t <- order(imp.temp,decreasing=TRUE)
sort.imp <- imp.temp[t,]
   # these are in order most...least important
gn.20 <- names(sort.imp[1:20])
```

```
# Get expression values for 20 most important genes
t <- is.element(gn,gn.20)
small.eset <- emat[t,]
  # matrix of expression values,
  # not necessarily in order of importance

# Make a heatmap, with group differences obvious on plot
library(RColorBrewer)
hmcol <- colorRampPalette(brewer.pal(9,"Purples"))(256)
colnames(small.eset) <- group
  # This will label the heatmap columns
csc <- rep(hmcol[50],ncol(small.eset))
csc[group=='CM'] <- hmcol[200]
  # column side color will be dark for CM and light for ctl
heatmap(small.eset,scale="row", col=hmcol,ColSideColors=csc,
   cexCol=2.5,cexRow=1.5, main='Raw Counts = expr')
heatmap(log(1+small.eset),scale="row", col=hmcol,
        ColSideColors=csc,
        cexCol=2.5,cexRow=1.5, main='Log(1+expr)')
```

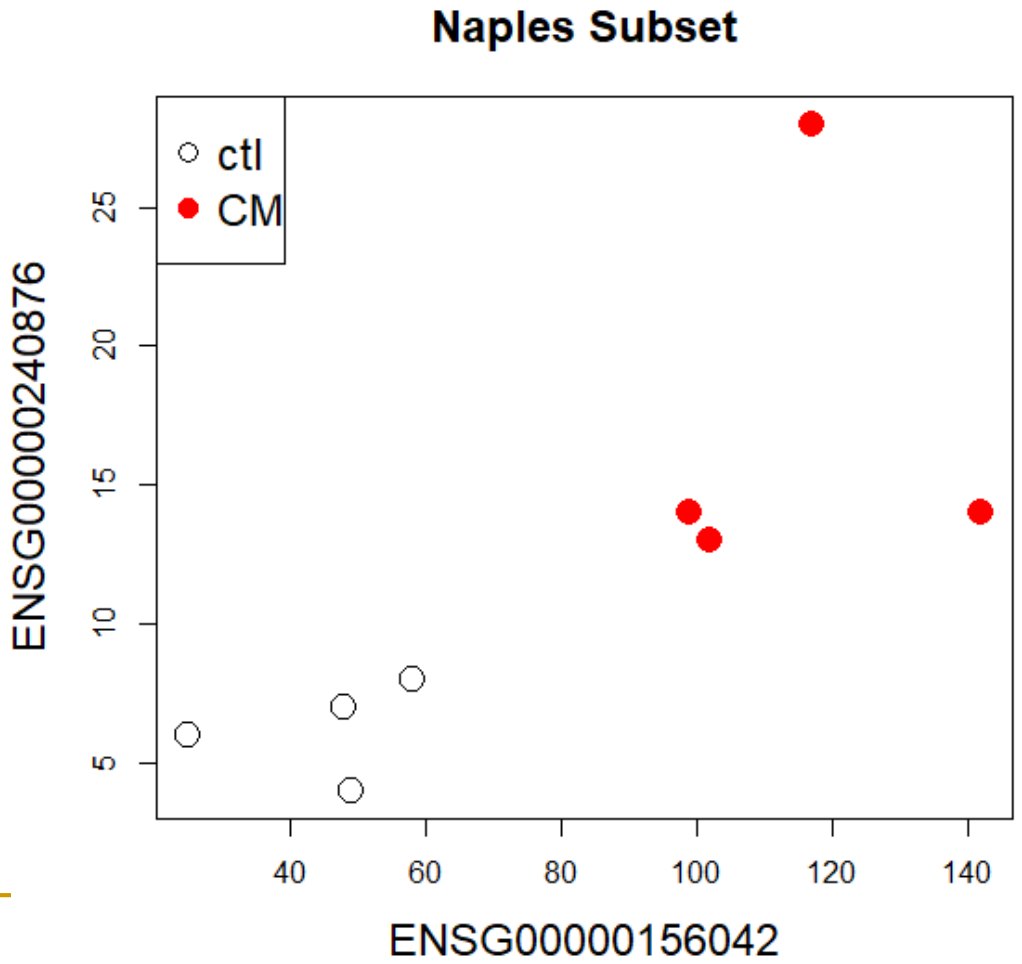
Can focus on "variable selection"

- 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 Naples filtered subset

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.

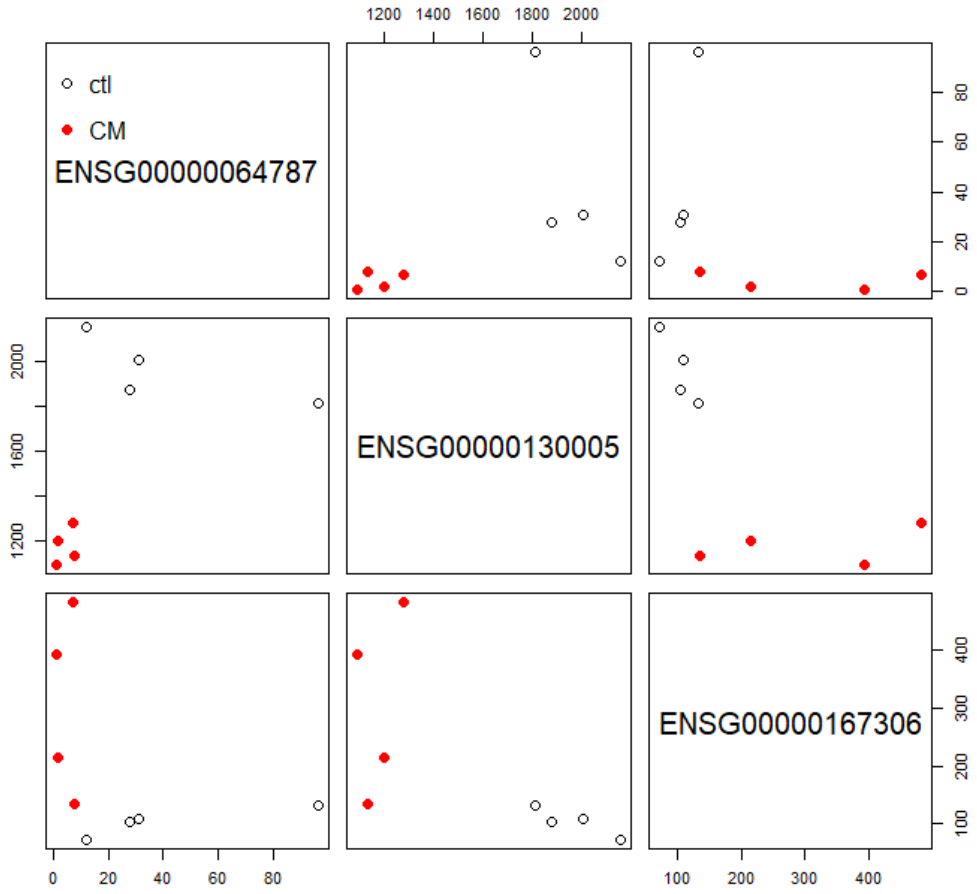


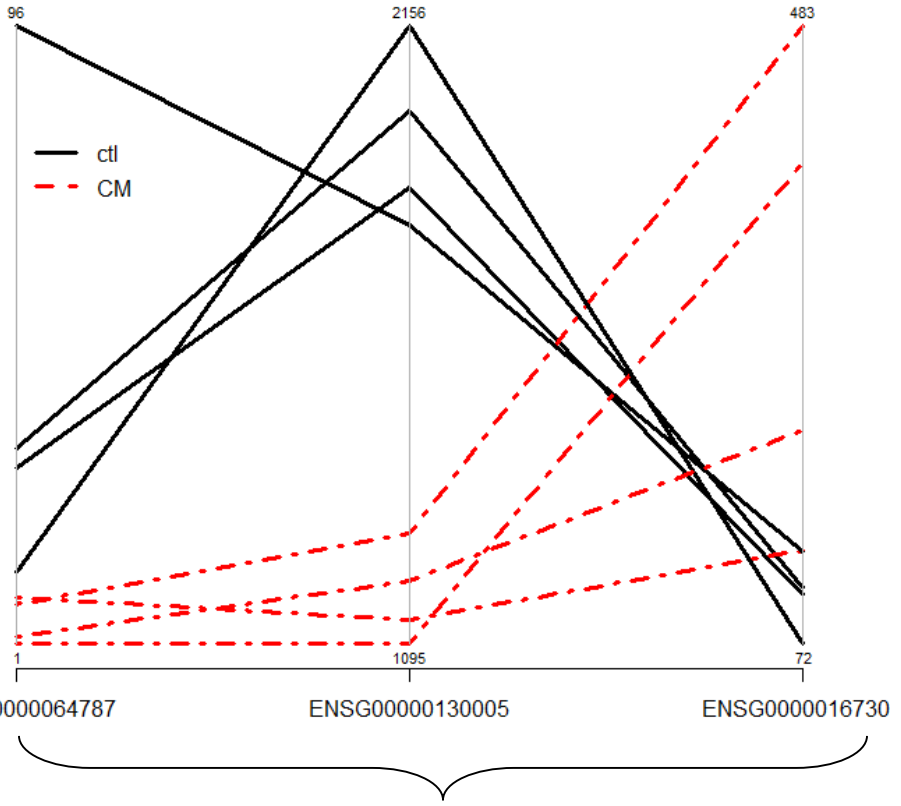
```
# Look at variable selection
library(varSelRF)
set.seed(1234)
print(date())
rfsel <- varSelRF(t(eset.fil),as.factor(group),
 ntree=10000, ntreeIterat=2000, vars.drop.frac=0.2)
print(date()) # 25 seconds
# rfsel$firstForest is the same as the slide 15 rf object
rf.sig.gn <- rfsel$selected.vars
   # "ENSG00000156042" "ENSG00000240876"



# Visualize these two genes
exp.gn.1 <- emat[gn==rf.sig.gn[1],]
exp.gn.2 <- emat[gn==rf.sig.gn[2],]
use.pch <- c(1,16,1,16,1,16,1,16) # Define plotting chars.
use.col <- c(1,2,1,2,1,2,1,2) # Define plotting colors
plot(exp.gn.1,exp.gn.2,col=use.col,main='Naples Subset',
  cex.main=1.5, cex.lab=1.5, xlab=rf.sig.gn[1],
  ylab=rf.sig.gn[2], pch=use.pch,cex=2)
legend('topleft',
       c('ctl','CM'),pch=c(1,16),col=c(1,2),cex=1.5)
```

RF Variable Selection on Naples (full)





Note: data here are on raw count scale; for some data, log(1+expr) may show better.

full Naples data: all 56,620 genes

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

```
# RF variable selection with full data set

# set seed and define initial objects
set.seed(134)

print(date())
rf.big <- varSelRF(t(emat),as.factor(group),
  ntree=5000, ntreeIterat=2000, vars.drop.frac=0.2)
print(date())  # about 50 seconds

rf.gn <- rf.big$selected.vars
  #    "ENSG00000064787" "ENSG00000130005" "ENSG00000167306"
```

```
# make scatterplot matrix, with points colored by cell type
t.rf <- is.element(gn,rf.gn)
rf.eset <- t(emat[t.rf,])
 # this rf.eset has rows for obs and columns for 3 genes
use.pch <- c(1,16,1,16,1,16,1,16) # Define plotting chars.
use.col <- c(1,2,1,2,1,2,1,2) # Define plotting colors
pairs(rf.eset,col=use.col,pch=use.pch,cex=1.5)
 # pairs function makes scatterplot matrix of rf.eset cols.
par(xpd=TRUE)
legend(.05,.95,c('ctl','CM'),pch=c(1,16),
   col=c(1,2), bty='n')
par(xpd=FALSE)


# Now - make a profile plot (parallel coordinates plot)
library(MASS)
parcoord(rf.eset, col=use.col, lty=use.pch,
  lwd=3, var.label=TRUE)
legend(1,.85,c('ctl','CM'),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
  - so scaling genes or log wouldn’t change anything
  - what about “normalizing” sample (total read counts)?
- Free! (Fortran code by Breiman and Cutler)
- Returns the importance of predictors (gene)
- Little need to tune parameters

Summary: RF, beyond simple case

- Doesn't automatically "allow" multiple design factors (treated as response variables, outputs, or targets here)
  - Bovine oviduct:
    Lactation (3 levels)
    Location (4 levels)
  - But see:
    - Multivariate random forests (2011 Segal & Xiao)
    - IntegratedMRF (2017 Rahman et al.)
    - R package MultivariateRandomForest (2017 Rahman)
- Still, no statistical inference*

* yet…