Katie's database quality checks

This is a quick run-down of the two datasets I checked to make sure I was reading in my data properly. I chose one processed dataset, which means I was basically just uploading a text file, and a raw dataset, which means I downloaded the files and ran an automated script to upload it all to the database. For any type of expression study, my code:
-updates gene symbols
-puts one gene symbol/line (makes it easier down the road to match up gene symbols in studies)
-creates outlier reports (optional as this is computationally intensive)

I wanted to make sure all these processes were giving me similar results to those found in publications using this GEO data.

GSE1379
GSE1379 is a cool GEO study because it also includes a separate laser micro-dissected dataset from the same patients (I also have this uploaded, just trying to decide how/when to use it.) These patients all had breas cancer, and had surgery to remove their tumors. They then had tamoxifen (anti-estrogen therapy) for 5 years. Biopsies were taken before any treamtent was given (standard).

The study just took the top 25% of genes by variance and then ran a paired t-test on every single gene, using a final p-value cutoff of .001 (they were left with 19 probes.)

I first pull the data down using RmySQL and database-specific functions I've created (and need to document in R markdown!)

```
require(DBI)
```

```
## Loading required package: DBI
```

```
require(RMySQL)
```

```
## Loading required package: RMySQL
```

```
# code to pull out expression, outcomes data
source("~/Box Documents/Atul BC biomarkers/breastcancer/queryAndPackageExpression.R")

# errr...don't use this info!
username = "ywrfc09"
password = "aveelyau05"
host = "buttelab-db1"
dbname = "user_ywrfc09"


# get GSMIDs for this cohort
GSE_ID <- 1379

# select the correct patient IDs (GMIDs) from this specific GEO study
query <- paste("SELECT GEO_GSMID FROM breastCancer_humans_perPatientData WHERE GEO_GSE_reference_series = ",
    GSE_ID, " ORDER BY GEO_GSE_reference_series", sep = "")
m <- dbDriver("MySQL")
con = dbConnect(m, username = username, password = password, host = host, dbname = dbname)
res <- dbSendQuery(con, query)
GSMIDs <- fetch(res, n = -1)
dbDisconnect(con)
```

```
## [1] TRUE
```

```


# pass in these IDs to get the corresponding clinical data. You'll note
# that these patients has Disease Free Survival (DFS) reported (this
# function also always spits out the corresponding class variable), and I
# chose to not create an outlier/quality report.
GSMIDs <- t(GSMIDs)
GSMIDs <- as.vector(GSMIDs)
data <- queryAndPackageOutcomesAndExpression(GSMIDs = GSMIDs, query_name = GSE_ID,
    outcomes_var = "DFS", arrayQualityReport = FALSE, qualityCheckFilePath = "~/Box Documents/Atul BC biomar
```

```
## Loading required package: arrayQualityMetrics
```

```
## Creating a generic function for 'boxplot' from package 'graphics' in
## package 'affyPLM'
```

```
## Creating a generic function for 'hist' from package 'graphics' in package
## 'affyPLM'
```

```
## Loading required package: limma
## Warning: package 'limma' was built under R version 2.15.1
```

```
## [1] "SELECT GEO_GSMID, GSM_GSMID, GEO_GSE_reference_series, site_prefix, GEO_platform_ID ,DFS, \nmicroarr
## [1] "total number of samples is:"
## [1] 60
## [1] "current starting index is:"
## [1] 1
## GSE ID is:  1379
## platform is:  GPL1223
## site prefix is  NA
## treat_days are:  0[1] "probe table is:"
## [1] "probes_GSE1379_GPL1223"
## successfully downloaded expression values for this sub-cohort 1379_GPL1223_NA[1] 1379
## [1] "current class is:"
##  [1] 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 1 0 1 1 0 1 1 0 1 0 1 1 1 0 0 1 0 0
## [36] 1 1 0 1 1 1 1 0 1 1 1 0 0 1 1 0 0 1 1 1 1 1 1 1 1
## attr(,"levels")
## [1] "0" "1"
## [1] "length of keys is:"
## [1] 22582
## [1] "dim of expr is:"
## [1] 22582    60
## all dataset names are:  GSE1379_GPL1223
```

```

data <- data$allDataSets[[1]]
```

You'll from my ramblings that pretty quickly I'd be getting at least some different results from the publication because they claim top 25% = 5475 genes. I could NOT figure out how they got this number.

```
# NOTE: they claim 25% of the genes = 5475. Without removing all the NAs
# at the top, I got 5645.6 non-NAs start at row 217, which is 'NUDT2'.
# this leaves us with 25% = 5592.... careful because you also expanded the
# dataset to parse out /// & update symbols. 22575 vs. 22582 - that's not
# a TON though however, even when I remove the NAs and look at the
# original top gene list, that's only 4201! where the HELL did they get
# 5475 from???

# did they remove ALL NAs first? but then get lower than 5475.

# keys = gene symbols (I used Purvesh's notation)
expr <- data$expr[which(!is.na(data$keys)), ]
keys <- data$keys[which(!is.na(data$keys))]
probes <- data$probes[which(!is.na(data$keys))]
class <- data$class
```

As an extra double-check, I downloaded the original processed data file from GEO again to compare.

```
# double-check original text file note: NAs are '' here in keys (no NAs in
# expr)
orig_data <- read.delim("~/Box Documents/Atul BC biomarkers/documentation/arrayQualityCheck/1379GSE1379_GPL1
    header = TRUE, na.strings = "")
expr_orig <- orig_data[, 3:dim(orig_data)[2]]
keys_orig <- orig_data[, 1]
# remove NAs.finicky...-is.na vs !is.na sometimes zeros out all rows??
keys_orig <- keys_orig[which(!is.na(keys_orig))]
expr_orig <- expr_orig[which(!is.na(keys_orig)), ]

# re-align classes for original data matrix! match up here already though
# :)
which(colnames(expr) == colnames(expr_orig))
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## [24] 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
## [47] 47 48 49 50 51 52 53 54 55 56 57 58 59 60
```

Now that I have my data prepped and ready to go, I calculated the variance (I realize I could have done this using sapply()…or heck way faster in MATLAB with matrices…but I at least like seeing what I'm doing in R with loops.)

```
var <- array(data = NA, dim = length(keys))
var_orig <- array(data = NA, dim = length(keys_orig))

expr <- as.matrix(expr)

for (v in 1:length(keys)) {

    var[v] <- var(expr[v, ])

}

# need it in matrix format to work with var() for some reason
expr_orig <- as.matrix(expr_orig)
for (t in 1:length(keys_orig)) {

    var_orig[t] <- var(expr_orig[t, ])

}
```

I then rank the variances and look for the top 25%.

```
sortVar <- sort.int(decreasing = TRUE, var, index.return = TRUE)
sortVar_orig <- sort.int(decreasing = TRUE, var, index.return = TRUE)

# take top 75th percentile of genes by variance.
top25_var_values <- sortVar$x[1:round(0.25 * (length(keys)))]
top25_var_indices <- sortVar$ix[1:round(0.25 * (length(keys)))]


top25_expr <- expr[top25_var_indices, ]
top25_keys <- keys[top25_var_indices]
top25_probes <- probes[top25_var_indices]

# do the same for original list
top25_var_valuesO <- sortVar_orig$x[1:round(0.25 * (length(keys_orig)))]
top25_var_indicesO <- sortVar_orig$ix[1:round(0.25 * (length(keys_orig)))]


top25_exprO <- expr[top25_var_indicesO, ]
top25_keysO <- keys[top25_var_indicesO]
top25_probesO <- probes[top25_var_indicesO]
```

Then a quick gut-check. HOXB13 was a highlighted gene in the paper…is it at least in the top 25?

```
# good...at least getting HOXB13 to show up...oddly enough grep didn't
# work here! due to data format?
which(top25_keys == "HOXB13")
```

```
## [1]  76 468
```

```
which(top25_keysO == "HOXB13")
```

```
## [1]  76 468
```

Great! On to the paired t-statistic.

```
t_stat <- array(data = NA, dim = length(top25_keys))
t_pValue <- array(data = NA, dim = length(top25_keys))
t_parameter <- array(data = NA, dim = length(top25_keys))
top25_expr <- as.matrix(top25_expr)

classM <- as.numeric(matrix(class))
for (e in 1:length(top25_keys)) {

    # may not have equal case vs. control grps. run sample t-test via formual
    # then, not just t.test(x,y) rbind changes class to 1,2
    singleGene <- as.numeric(matrix(top25_expr[e, ]))
    dataMatrix <- matrix(data = NA, nrow = length(classM), ncol = 2)
    dataMatrix[, 1] <- singleGene
    dataMatrix[, 2] <- classM
    cols <- c("gene", "class")
    colnames(dataMatrix) <- cols

    test <- t.test(gene ~ class, data = dataMatrix)
    t_stat[e] <- test$statistic
    t_parameter[e] <- test$parameter
    t_pValue[e] <- test$p.value

}

sort_pValues <- sort.int(decreasing = FALSE, t_pValue, index.return = TRUE)
# only 19 genes indeed made .001 rounded cutoff
top19pValues <- sort_pValues$x[1:19]
top19GeneIndices <- sort_pValues$ix[1:19]
top19Genes <- top25_keys[top19GeneIndices]

# now do for original data
t_statO <- array(data = NA, dim = length(top25_keysO))
t_pValueO <- array(data = NA, dim = length(top25_keysO))
t_parameterO <- array(data = NA, dim = length(top25_keysO))
top25_exprO <- as.matrix(top25_exprO)

classM <- as.numeric(matrix(class))
for (e in 1:length(top25_keysO)) {

    # may not have equal case vs. control grps. run sample t-test via formual
    # then, not just t.test(x,y) rbind changes class to 1,2
    singleGene <- as.numeric(matrix(top25_exprO[e, ]))
    dataMatrix <- matrix(data = NA, nrow = length(classM), ncol = 2)
    dataMatrix[, 1] <- singleGene
    dataMatrix[, 2] <- classM
    cols <- c("gene", "class")
    colnames(dataMatrix) <- cols

    test <- t.test(gene ~ class, data = dataMatrix)
    t_statO[e] <- test$statistic
    t_parameterO[e] <- test$parameter
    t_pValueO[e] <- test$p.value

}

sort_pValuesO <- sort.int(decreasing = FALSE, t_pValueO, index.return = TRUE)
# only 19 genes indeed made .001 rounded cutoff
top19pValuesO <- sort_pValuesO$x[1:19]
top19GeneIndicesO <- sort_pValuesO$ix[1:19]
top19GenesO <- top25_keysO[top19GeneIndicesO]
```

Look at the top genes and p values. Looks like at least my database data matches up with the original data matrix in GEO!

```
print(top19pValues)
```

```
##  [1] 2.142e-05 9.351e-05 1.153e-04 1.889e-04 2.077e-04 2.459e-04 3.389e-04
##  [8] 5.334e-04 6.647e-04 6.703e-04 7.154e-04 8.213e-04 1.013e-03 1.034e-03
## [15] 1.110e-03 1.146e-03 1.254e-03 1.276e-03 1.316e-03
```

```
print(top19Genes)
```

```
##  [1] "CCL4"      "IL1R2"     "IL17RB"    "DOK2"      "SH2B2"
##  [6] "CHDH"      "ABCC11"    "PTGER3"    "ANO3"      "LYPD6"
## [11] "CCL3L3"    "GUCY2D"    "HOXB13"    "HOXB13"    "PLA2G7"
## [16] "RHD"       "TNFAIP8L2" "LILRA5"    "SLAMF8"
```

```

print(top19pValuesO)
```

```
##  [1] 2.142e-05 9.351e-05 1.153e-04 1.889e-04 2.077e-04 2.459e-04 3.389e-04
##  [8] 5.334e-04 6.647e-04 6.703e-04 7.154e-04 8.213e-04 1.013e-03 1.034e-03
## [15] 1.110e-03 1.146e-03 1.254e-03 1.276e-03 1.316e-03
```

```
print(top19GenesO)
```

```
##  [1] "CCL4"       "IL1R2"      "IL17RB"     "DOK2"       "SH2B2"
##  [6] "CHDH"       "ABCC11"     "PTGER3"     "ANO3"       "LYPD6"
## [11] "CCL3L3"     "GUCY2D"     "HOXB13"     "HOXB13"     "PLA2G7"
## [16] "RHD"        "TNFAIP8L2" "LILRA5"     "SLAMF8"
```

Looking further at the excel sheet errorCheck_GSE1379 on Central Desktop, you can see that I was able to translate the paper's old gene symbols to mine (i.e. the R gene symbol update package HGNChelper I used is working), and that the p-values are very similar (some ranks are different.) It turns out only 14 of the paper's top 19 genes are actually gene symbols, the rest were probes….that still weren't in their dataset in GEO?

Given the widely accepted notion that sadly, it's quite hard to replicate gene signature studies, I'd say this exercise at least proves I'm reading in my processed data correctly. On to raw data!

GSE19615
I actually had to go add a new dataset to find a raw dataset with clear enough methods I could *somewhat* reproduce. The normalization scheme and background correction wasnt' explicitly stated, but I do know that the paper used PAMR on the procesesed data, and gave their top gene list.

Load up the data like before (same functions). Becuase this data is taken from my database, it was already background corrected and normalized via ReadAffy() and gcrma().

```
# get GSMIDs for this cohort
GSE_ID <- 19615


query <- paste("SELECT GEO_GSMID FROM breastCancer_humans_perPatientData WHERE GEO_GSE_reference_series = ",
    GSE_ID, " ORDER BY GEO_GSE_reference_series", sep = "")
m <- dbDriver("MySQL")
con = dbConnect(m, username = username, password = password, host = host, dbname = dbname)
res <- dbSendQuery(con, query)
GSMIDs <- fetch(res, n = -1)
dbDisconnect(con)
```

```
## [1] TRUE
```

```


GSMIDs <- t(GSMIDs)
GSMIDs <- as.vector(GSMIDs)

# outcomes variable in paper wasn't RFS/relapse, but distant (bone)
# metastasis
data <- queryAndPackageOutcomesAndExpression(GSMIDs = GSMIDs, query_name = GSE_ID,
    outcomes_var = "RFS", arrayQualityReport = FALSE, qualityCheckFilePath = "~/Box Documents/Atul BC biomar
```

```
## [1] "SELECT GEO_GSMID, GSM_GSMID, GEO_GSE_reference_series, site_prefix, GEO_platform_ID ,RFS, \nmicroarr
## [1] "total number of samples is:"
## [1] 115
## [1] "current starting index is:"
## [1] 1
## GSE ID is:  19615
## platform is:  GPL570
## site prefix is  NA
## treat_days are:  0[1] "probe table is:"
## [1] "probes_GSE19615_GPL570_allSites"
## successfully downloaded expression values for this sub-cohort 19615_GPL570_NA[1] 19615
## [1] "current class is:"
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [71] 1 1 0 0 1 1 1 0 1 0 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [106] 1 1 1 1 0 1 1 1 1 1
## attr(,"levels")
## [1] "0" "1"
## [1] "length of keys is:"
## [1] 54696
## [1] "dim of expr is:"
## [1] 54696   115
## all dataset names are:  GSE19615_GPL570
```

```


data <- data$allDataSets[[1]]

expr <- data$expr
keys <- data$keys
probes <- data$probes
class <- data$class
```

Now this time, I just need to run the PAMR package. PAMR creates a class prediction model based on nearest shrunken centroids. The model output is weights for both 0 and 1 for each gene. You can run adaptthresh() to find the appropriate shrunken threshold; the paper says it used 2 (although when I ran adaptthresh out of curiosity, the lowest error was around 3!)

```
library("pamr")
```

```
## Loading required package: cluster
```

```
## Warning: package 'cluster' was built under R version 2.15.1
```

```
## Loading required package: survival
```

```
## Loading required package: splines
```

```


# nice package bc also lets you do knnimpute on missing data. set up list
# of expression and class labels for pamr package nice...looks like we can
# use >2 groups?

# expr rownames already probes. want this, as publication's PAM score list
# is by probe, not gene symbol. need to specify this with geneids or else
# it yells at you! had to find in a forum...
data <- list(x = expr, y = class, geneids = rownames(expr))


# publication let threshold/'shrinkage parameter delta' = 2
train <- pamr.train(data = data, ngroup.survival = 2, threshold = 2)
```

```
## 1
```

```
train_noThresh <- pamr.train(data = data, ngroup.survival = 2)
```

```
## 123456789101112131415161718192021222324252627282930
```

```
# try with, and without, adaptthresh
adaptthresh <- pamr.adaptthresh(object = train_noThresh, ntries = 10)
```

```
## Initial errors:  6.233 14.267 Roc 188.7
## Update 1
## 123456789101112131415161718192021222324252627282930
## Errors  6.867 13.400 Roc 190.5
## Update 2
## 123456789101112131415161718192021222324252627282930
## Errors  7.267 12.500 Roc 188.7
## Update 3
## 123456789101112131415161718192021222324252627282930
## Errors  7.667 12.033 Roc 189.7
## Update 4
## 123456789101112131415161718192021222324252627282930
## Errors  7.867 11.700 Roc 190.6
## Update 5
## 123456789101112131415161718192021222324252627282930
## Errors  7.967 11.733 Roc 192.8
## Update 6
## 123456789101112131415161718192021222324252627282930
## Errors  8.033 11.867 Roc 196.3
## Update 7
## 123456789101112131415161718192021222324252627282930
## Errors  7.933 12.167 Roc 200.7
## Update 8
## 123456789101112131415161718192021222324252627282930
## Errors  7.933 12.567 Roc 206
## Update 9
## 123456789101112131415161718192021222324252627282930
## Errors  7.933 12.933 Roc 211.7
## Update 10
## 123456789101112131415161718192021222324252627282930
## Errors  7.90 13.23 Roc 215.7
```

```

train2 <- pamr.train(data = data, ngroup.survival = 2, threshold.scale = adaptthresh)
```

```
## 123456789101112131415161718192021222324252627282930
```

```
# paper says used CV of 10.
results <- pamr.cv(fit = train, data = data, nfold = 10)
```

```
## 12Fold 1 :1
## Fold 2 :1
## Fold 3 :1
## Fold 4 :1
## Fold 5 :1
## Fold 6 :1
## Fold 7 :1
## Fold 8 :1
## Fold 9 :1
## Fold 10 :1
```

```
# I WOULD use results2 in my final analysis with the optimal threshold if
# I wasn't trying to match the paper.
results2 <- pamr.cv(fit = train2, data = data, nfold = 10)
```

```
## 12Fold 1 :123456789101112131415161718192021222324252627282930
## Fold 2 :123456789101112131415161718192021222324252627282930
## Fold 3 :123456789101112131415161718192021222324252627282930
## Fold 4 :123456789101112131415161718192021222324252627282930
## Fold 5 :123456789101112131415161718192021222324252627282930
## Fold 6 :123456789101112131415161718192021222324252627282930
## Fold 7 :123456789101112131415161718192021222324252627282930
## Fold 8 :123456789101112131415161718192021222324252627282930
## Fold 9 :123456789101112131415161718192021222324252627282930
## Fold 10 :123456789101112131415161718192021222324252627282930
```

```
# not directly related, but I tried to used their FDR function and it kept
# claiming some data wasn missing. FDR_thresh <- pamr.fdr(trained.obj =
# train, data = data, nperms=100) FDR_thresh2 <- pamr.fdr(trained.obj =
# train2, data = data, nperms=100) got an error: try w/o threshold = 2?
# Error in dimnames(results) <- list(NULL, c('Threshold', 'Number of
# significant genes', : 'dimnames' applied to non-array


# get genes that survive thresholding keep the row names (gene symbol
# names) If fitcv is provided, the function also reports the average rank
# of the gene in the cross-validation folds
gene_list <- pamr.listgenes(fit = train, data = data, threshold = 2, fitcv = results,
    genenames = TRUE)
```

```
##        id           0-score 1-score av-rank-in-CV prop-selected-in-CV
##   [1,] 206166_s_at  0.3918  -0.0543 3.5           1
##   [2,] 235599_at    0.348   -0.0482 17.4          1
##   [3,] 243200_at    0.298   -0.0413 160.9         0.9
##   [4,] 1554712_a_at 0.2923  -0.0405 10.9          1
##   [5,] 225459_at    0.2617  -0.0363 19.9          1
##   [6,] 1554906_a_at 0.2589  -0.0359 61.4          0.9
##   [7,] 217528_at    0.2554  -0.0354 28.9          1
##   [8,] 206165_s_at  0.2413  -0.0335 39.8          1
##   [9,] 203059_s_at  0.2325  -0.0322 22.2          1
##  [10,] 227811_at    -0.232  0.0322  20            1
##  [11,] 242342_at    0.2277  -0.0316 63.8          0.9
##  [12,] 206164_at    0.2262  -0.0314 54.7          0.9
##  [13,] 237496_at    0.2174  -0.0301 38.9          1
##  [14,] 224180_x_at  0.2067  -0.0287 34.9          1
##  [15,] 1552378_s_at 0.1877  -0.026  36.7          1
##  [16,] 211148_s_at  0.1876  -0.026  34.8          1
##  [17,] 211814_s_at  0.1853  -0.0257 39.9          1
##  [18,] 218571_s_at  -0.1851 0.0257  35.7          1
##  [19,] 233025_at    0.1844  -0.0256 98.7          0.9
##  [20,] 211273_s_at  0.1803  -0.025  37.7          1
##  [21,] 210571_s_at  0.1784  -0.0247 99.1          0.9
##  [22,] 212777_at    0.177   -0.0245 30            1
##  [23,] 206510_at    0.1649  -0.0229 68.8          1
##  [24,] 209591_s_at  0.1635  -0.0227 44.4          1
##  [25,] 228523_at    0.1629  -0.0226 55.2          1
##  [26,] 219505_at    -0.1623 0.0225  45.7          1
##  [27,] 239006_at    0.158   -0.0219 115.4         0.8
##  [28,] 1554640_at   0.1485  -0.0206 56.4          1
##  [29,] 226789_at    -0.1417 0.0196  64.6          1
##  [30,] 238045_at    0.1384  -0.0192 60            1
##  [31,] 228610_at    0.1365  -0.0189 180.8         0.9
##  [32,] 218572_at    -0.1331 0.0185  66.9          1
##  [33,] 218541_s_at  -0.1312 0.0182  60.5          1
##  [34,] 204914_s_at  0.1299  -0.018  77.5          1
##  [35,] 201942_s_at  0.1287  -0.0178 100.7         0.9
##  [36,] 224311_s_at  0.1265  -0.0175 71.7          1
##  [37,] 205034_at    0.1247  -0.0173 70            1
##  [38,] 206582_s_at  0.1242  -0.0172 98.8          1
##  [39,] 222758_s_at  0.1135  -0.0157 101           1
##  [40,] 227467_at    0.1117  -0.0155 85.7          0.9
##  [41,] 225811_at    -0.1101 0.0153  121.2         0.9
##  [42,] 230316_at    -0.109  0.0151  75.9          1
##  [43,] 204913_s_at  0.1022  -0.0142 120           0.9
##  [44,] 213539_at    -0.0999 0.0139  131.4         0.9
##  [45,] 202357_s_at  -0.0985 0.0137  139.8         0.8
##  [46,] 213217_at    0.0979  -0.0136 143.6         0.9
##  [47,] 201287_s_at  0.0972  -0.0135 86.5          1
##  [48,] 223484_at    -0.0949 0.0132  168.9         0.9
##  [49,] 219270_at    0.0945  -0.0131 279.6         0.9
##  [50,] 223125_s_at  -0.0932 0.0129  136.3         1
##  [51,] 205376_at    -0.0925 0.0128  110.1         1
##  [52,] 34210_at     -0.0924 0.0128  156.8         0.8
##  [53,] 200814_at    -0.0914 0.0127  115.2         0.9
##  [54,] 221087_s_at  -0.0913 0.0127  164.6         0.8
##  [55,] 1553436_at   0.0908  -0.0126 388.6         0.7
##  [56,] 201943_s_at  0.0899  -0.0125 149           0.9
##  [57,] 204856_at    0.0859  -0.0119 270           0.8
##  [58,] 233825_s_at  -0.0835 0.0116  147.7         0.9
##  [59,] 214370_at    0.0812  -0.0113 250.9         0.7
##  [60,] 206511_s_at  0.0804  -0.0111 180.7         0.9
##  [61,] 220953_s_at  0.0797  -0.0111 207           0.8
##  [62,] 204915_s_at  0.0782  -0.0108 159.6         0.9
##  [63,] 214583_at    0.0781  -0.0108 177.5         0.9
##  [64,] 244644_at    0.0778  -0.0108 274.3         0.8
##  [65,] 231484_at    0.0777  -0.0108 335.3         0.7
##  [66,] 204949_at    -0.0771 0.0107  144.8         0.9
##  [67,] 220253_s_at  0.0761  -0.0105 191.8         0.8
##  [68,] 206533_at    0.0756  -0.0105 157.2         0.7
##  [69,] 235548_at    0.0754  -0.0105 224.5         0.8
##  [70,] 225945_at    -0.0738 0.0102  186.2         0.8
##  [71,] 227758_at    -0.0734 0.0102  159           0.9
##  [72,] 218553_s_at  0.0727  -0.0101 130.9         1
##  [73,] 206837_at    0.0722  -0.01   240.5         0.7
##  [74,] 219359_at    -0.0704 0.0098  133.9         0.9
##  [75,] 219121_s_at  0.07    -0.0097 150.7         0.8
##  [76,] 232573_at    0.0687  -0.0095 205           0.8
##  [77,] 203904_x_at  0.0683  -0.0095 157.8         0.9
##  [78,] 209590_at    0.0682  -0.0094 216.4         0.9
##  [79,] 205590_at    -0.0679 0.0094  223.9         0.7
##  [80,] 203220_s_at  0.0672  -0.0093 358.5         0.9
##  [81,] 214697_s_at  0.0656  -0.0091 154.6         0.8
##  [82,] 236351_at    0.065   -0.009  304.7         0.7
##  [83,] 213926_s_at  0.0649  -0.009  156.4         0.9
##  [84,] 221029_s_at  0.0647  -0.009  402.1         0.8
##  [85,] 227884_at    0.0629  -0.0087 198           0.9
##  [86,] 201940_at    0.0621  -0.0086 222.7         0.9
##  [87,] 238480_at    -0.0595 0.0082  164.4         0.7
##  [88,] 204541_at    -0.0595 0.0082  145.8         1
##  [89,] 229085_at    0.0578  -0.008  463.7         0.8
##  [90,] 219631_at    0.0567  -0.0079 188.5         0.7
##  [91,] 204735_at    -0.0566 0.0078  170.3         1
##  [92,] 218905_at    0.0563  -0.0078 208.4         0.7
##  [93,] 211478_s_at  0.0547  -0.0076 319.1         0.8
##  [94,] 218921_at    -0.0531 0.0074  203.3         0.8
##  [95,] 211194_s_at  0.0516  -0.0071 524.6         0.9
##  [96,] 227582_at    -0.0506 0.007   336.8         0.7
##  [97,] 231513_at    0.05    -0.0069 218           0.8
##  [98,] 211078_s_at  0.0492  -0.0068 324.6         0.8
##  [99,] 226810_at    -0.049  0.0068  263           0.8
## [100,] 241342_at    0.0483  -0.0067 224.8         0.7
## [101,] 227940_at    0.0481  -0.0067 223.5         0.7
## [102,] 235391_at    0.0478  -0.0066 187.6         0.9
## [103,] 243918_at    0.0459  -0.0064 272.9         0.8
## [104,] 225407_at    -0.0451 0.0063  220           0.8
## [105,] 213035_at    -0.0443 0.0061  211.4         1
## [106,] 213959_s_at  0.0442  -0.0061 281.3         0.7
## [107,] 220622_at    0.0426  -0.0059 1061.3        0.7
## [108,] 211981_at    0.042   -0.0058 259.9         0.8
## [109,] 209394_at    -0.0415 0.0058  253.8         0.7
## [110,] 226446_at    0.04    -0.0055 330.6         0.7
## [111,] 228763_at    -0.0396 0.0055  195.1         0.9
## [112,] 235205_at    0.0396  -0.0055 335.2         0.8
## [113,] 204818_at    0.0386  -0.0053 728.3         0.7
## [114,] 230323_s_at  0.0384  -0.0053 609           0.7
## [115,] 214811_at    0.0366  -0.0051 831.8         0.8
## [116,] 212611_at    0.0352  -0.0049 423.9         0.6
## [117,] 210117_at    0.0348  -0.0048 448.5         0.5
## [118,] 202316_x_at  0.0346  -0.0048 288.5         0.8
## [119,] 1557239_at   0.034   -0.0047 312.9         0.7
## [120,] 218747_s_at  -0.0333 0.0046  251.8         0.7
## [121,] 238467_at    0.033   -0.0046 271.8         0.9
## [122,] 207223_s_at  0.0329  -0.0046 280.3         0.7
## [123,] 202531_at    -0.0328 0.0045  310.1         0.6
## [124,] 221666_s_at  -0.0327 0.0045  290.8         0.5
## [125,] 220128_s_at  0.0327  -0.0045 451.1          0.7
## [126,] 223126_s_at  -0.0317 0.0044  271.2          0.8
## [127,] 229689_s_at  0.0303  -0.0042 461.3          0.8
## [128,] 227265_at    -0.0303 0.0042  351.1          0.6
## [129,] 201522_x_at  -0.0296 0.0041  398.6          0.7
## [130,] 220225_at    0.0292  -0.0041 600.4          0.8
## [131,] 1555964_at   0.0292  -0.004  393.6          0.7
## [132,] 1554245_x_at 0.0285  -0.004  496.7          0.7
## [133,] 1555716_a_at 0.0276  -0.0038 440.8          0.7
## [134,] 225534_at    -0.0275 0.0038  288.7          0.8
## [135,] 209948_at    0.0274  -0.0038 368.9          0.6
## [136,] 206307_s_at  0.0264  -0.0037 554.6          0.7
## [137,] 230793_at    0.0263  -0.0036 335.3          0.7
## [138,] 209772_s_at  0.0258  -0.0036 245.1          0.7
## [139,] 232279_at    -0.0255 0.0035  258.3          0.6
## [140,] 210915_x_at  -0.0253 0.0035  340.9          0.6
## [141,] 216294_s_at  0.0248  -0.0034 401.1          0.6
## [142,] 203571_s_at  -0.0246 0.0034  307.6          0.6
## [143,] 233713_at    0.0244  -0.0034 255.6          0.7
## [144,] 226226_at    0.0243  -0.0034 1005.6         0.7
## [145,] 241763_s_at  0.0241  -0.0033 390.2          0.6
## [146,] 237301_at    -0.0223 0.0031  260.4          0.9
## [147,] 206079_at    0.0222  -0.0031 422.3          0.6
## [148,] 239586_at    0.0215  -0.003  1258.4         0.8
## [149,] 205503_at    0.0213  -0.0029 333.8          0.7
## [150,] 226473_at    0.0205  -0.0028 250.8          0.7
## [151,] 208650_s_at  0.0199  -0.0028 268            0.6
## [152,] 224451_x_at  -0.0198 0.0028  312.2          0.6
## [153,] 235247_at    0.0188  -0.0026 1412.7         0.7
## [154,] 203100_s_at  0.0179  -0.0025 263.4          0.7
## [155,] 216557_x_at  -0.0178 0.0025  301.9          0.6
## [156,] 233446_at    0.0176  -0.0024 400            0.6
## [157,] 205236_x_at  0.0171  -0.0024 443.8          0.5
## [158,] 203779_s_at  0.0164  -0.0023 615.3          0.6
## [159,] 202874_s_at  0.0161  -0.0022 382.6          0.6
## [160,] 222379_at    -0.0161 0.0022  285.5          0.6
## [161,] 203616_at    -0.016  0.0022  326.3          0.6
## [162,] 207933_at    0.0153  -0.0021 835.2          0.8
## [163,] 229656_s_at  0.0153  -0.0021 367.5          0.7
## [164,] 244272_s_at  0.0152  -0.0021 522            0.7
## [165,] 226191_at    0.015   -0.0021 305.7          0.7
## [166,] 216375_s_at  0.0149  -0.0021 662.8          0.8
## [167,] 211796_s_at  -0.0143 0.002   399.6          0.6
## [168,] 36553_at     -0.0142 0.002   333.6          0.5
## [169,] 214581_x_at  0.0142  -0.002  325.2          0.5
## [170,] 204537_s_at  0.0141  -0.002  452.5          0.6
## [171,] 205307_s_at  0.0137  -0.0019 1123.1         0.7
## [172,] 208153_s_at  0.0129  -0.0018 678.1          0.7
## [173,] 213562_s_at  0.0124  -0.0017 371            0.7
## [174,] 225801_at    0.0122  -0.0017 579.3          0.8
## [175,] 217077_s_at  0.0121  -0.0017 525.4          0.7
## [176,] 212998_x_at  -0.0119 0.0017  362.4          0.6
## [177,] 212531_at    0.0109  -0.0015 554.8          0.5
## [178,] 209341_s_at  -0.0109 0.0015  428.7          0.6
## [179,] 226568_at    -0.0102 0.0014  347.9          0.6
## [180,] 205258_at    0.0095  -0.0013 586.1          0.8
## [181,] 238710_at    0.0092  -0.0013 829.7          0.8
## [182,] 227742_at    -0.0091 0.0013  325.4          0.5
## [183,] 213540_at    -0.0091 0.0013  376.1          0.6
## [184,] 222699_s_at  0.0086  -0.0012 412.3          0.4
## [185,] 212070_at    0.008   -0.0011 356.4          0.6
## [186,] 203222_s_at  0.0079  -0.0011 433.7          0.6
## [187,] 218092_s_at  0.0075  -0.001  334.4          0.7
## [188,] 220254_at    0.007   -0.001  415.2          0.6
## [189,] 242447_at    0.0063  -9e-04  486.5          0.5
## [190,] 227232_at    -0.0063 9e-04   339.3          0.6
## [191,] 202768_at    -0.0062 9e-04   324.4          0.5
## [192,] 236398_s_at  0.006   -8e-04  2895.2         0.9
## [193,] 204269_at    -0.006  8e-04   344            0.6
## [194,] 236203_at    -0.0058 8e-04   324.8          0.5
## [195,] 65591_at     0.0056  -8e-04  373.4          0.7
## [196,] 204638_at    -0.0056 8e-04   568.4          0.7
## [197,] 208884_s_at  0.0056  -8e-04  374.2          0.7
## [198,] 230391_at    -0.0053 7e-04   466.2          0.5
## [199,] 211812_s_at  0.0051  -7e-04  396            0.4
## [200,] 219355_at    0.005   -7e-04  490.6          0.7
## [201,] 222399_s_at  0.0049  -7e-04  337.8          0.6
## [202,] 217208_s_at  0.0046  -6e-04  379.4          0.7
## [203,] 205572_at    0.0045  -6e-04  352.6          0.5
## [204,] 229231_at    0.0031  -4e-04  341.6          0.6
## [205,] 205777_at    0.0027  -4e-04  463.3          0.6
## [206,] 234650_at    0.002   -3e-04  650.5          0.7
## [207,] 211430_s_at  -0.0019 3e-04   426            0.7
## [208,] 205787_x_at  0.0016  -2e-04  401.2          0.6
## [209,] 231070_at    0.0015  -2e-04  1422.5         0.8
## [210,] 205868_s_at  9e-04   -1e-04  363.2          0.5
## [211,] 213143_at    6e-04   -1e-04  856.3          0.7
```

I then read in the final published gene symbol list to see which ones matched (most of the published symbols seemed updated).

```
# compare with truth
truthData <- read.delim(header = TRUE, "~/Box Documents/Atul BC biomarkers/breastcancer/GSE19615_genelist_tr
truthProbes <- truthData[, 1]
# do all the truth probes match those in our DB?
length(which(!is.na(match(truthProbes, probes)))) == length(truthProbes)
```

```
## [1] TRUE
```

```

testProbes <- gene_list[, 1]
truthScore <- truthData[, 7]
# second column is zero-score or recurrence
testScore <- gene_list[, 2]

# good - all the outputted probes are in my database careful with MATCH - may need to remove NAs if there ar
testGeneSymbols <- keys[match(testProbes, probes)]
# if lengths equal, didn't lost any probes because weren't recognized in my DB
length(testProbes) == length(testGeneSymbols)
```

```
## [1] TRUE
```

```

# get the full list of updated gene symbols from my DB corresponding to publication's list use probes to lin
truthGeneSymbols <- keys[match(truthProbes, probes)]

# make sure lengths match - i.e. all our probes are in the DB.
length(truthGeneSymbols) == length(truthProbes)
```

```
## [1] TRUE
```

```

# we have 211 gene from my DB, 114 genes from the publication,
length(testGeneSymbols)
```

```
## [1] 211
```

```
length(truthGeneSymbols)
```

```
## [1] 114
```

```

# a few probes in the publication don't link to an identifiable gene symbol (haha now I know why they only p
which(is.na(truthGeneSymbols))
```

```
## [1]  5  9 20 52 85
```

```
which(is.na(testGeneSymbols))
```

```
##  [1]   2   3  11  13  41  65  76  82  97 101 103 112 121 127 143 146 153 165 174 192 198 206
```

I only got 58 probes matcheda against the published dataset. Some are still NAs - meaning the probes matched up, but there's no concordant gene symbol.

```
# NOTE: there are extra gene symbols matching up probably due to the NAs, and perhaps a few duplicated probe
length(which(!is.na(match(truthGeneSymbols, testGeneSymbols))))
```

```
## [1] 68
```

```
length(which(!is.na(match(truthProbes, testProbes))))
```

```
## [1] 58
```

```
matchingIndices <- match(truthProbes, testProbes)
matchingIndices <- matchingIndices[-which(is.na(matchingIndices))]
matchingIndicesTruth <- match(testProbes, truthProbes)
matchingIndicesTruth <- matchingIndicesTruth[-which(is.na(matchingIndicesTruth))]

matchingGenes <- testGeneSymbols[matchingIndices]

cat("our matching 58 genes are: ", "\n", matchingGenes, "\n")
```

```
## our matching 58 genes are:
##  CLCA2 CLCA2 CLCA2 CLCA2 S100A8 C1orf21 NA SOX11 SOX11 SDC1 SOS1 CAB39 B3GALNT1 IRX4 CMAHP TNFRSF21 CD24
```

PAMR uses a random seed, and I also probably normalized slightly differently…BUT it turns out there are a ton of duplicated probes in both datasets- 49 in mine, 28 in the published/truth set. But there are only 11 matched genes in the final list, so the 58 exact matches start to sound reasonable. We're reallyonly trying to match 114-28=86 unique genes in the original datbase.

```
length(which(duplicated(testGeneSymbols) == TRUE))
```

```
## [1] 49
```

```
length(which(duplicated(truthGeneSymbols) == TRUE))
```

```
## [1] 28
```

```
length(which(duplicated(matchingGenes) == TRUE))
```

```
## [1] 11
```

I then pulled out the corresponding scores and put it into an excel file (also on Central Desktop- GSE19615_errorCheckOutput.xls)

```
# get the corresponding scores from test to put alongside the truth scores
testScoreMatches <- testScore[matchingIndices]
testProbeMatches <- testProbes[matchingIndices]

truthScoreMatches <- truthScore[matchingIndicesTruth]
# fudging a bit - using the matching gene symbol indices because the duplicated probes are causing an issue
truthProbeMatches <- truthProbes[matchingIndicesTruth]
```

58/114 isn't perfect, but I'm also looking to see if I followed PAMR exactly. The paper claims “genes were selected at a false discovery threshold of that minimized a 10-fold cross-validation and test errors near the shrinkage parameter delta =2.” So theoretically, doing the 10-fold CV with a threshold of 2 should get me at *least* their gene list of about 120 genes, even if I didn't do the FDR threshold…this was not the case. But many gene symbols were also repeated in the list, so it's a little difficult to tell if digging further with get me much more. At least the results are in general concordant and not completely off the wall, which would indicate my automated database processing had a major bug in it somewhere.

Next step: whenever I run an analysis, I just need to make sure my gene symbols look intuitive - have most of them appeared in publications before?