# The Selection of the Top N Genes Varies With the Array Set

Kevin R. Coombes, Jing Wang, and Keith A. Baggerly

13 March 2007

## 1 Problem Definition

As noted elsewhere (`predCellLines.Rnw`), Potti and colleagues used the Novartis A arrays to find their signature genes. However, Novartis ran the arrays in triplicate, giving us a choice of three different individual data sets or one combined data set to select features. In this note, we explore how sensitive the top of the gene list is to changing the set of replicate array experiments.

## 2 Get the existing data objects

The first code chunk loads the Novartis U95A data for the NCI60 cell lines; note that arrays were only run on 59 cell lines. It also loads the data on the dose response to ten drugs for those cell lines. Basically, this reduces to loading the file `novartis.Rda` if it has already been created; otherwise, it creates it.

```
> prep <- file.path("Tangled", "prepareData.R")
> Stangle(file.path("RNowebSource", "prepareData.Rnw"),
+     output = prep)

Writing to file Tangled/prepareData.R

> source(prep)
> rm(prep)
```

The second code chunk loads the information we need in order to figure out which cell lines were called sensitive and resistant by Potti and colleagues.

```
> pred <- file.path("Tangled", "predCellLines.R")
> chem <- file.path("RDataObjects", "chemoPredictors.Rda")
> if (file.exists(chem)) {
+     load(chem)
+ } else {
+     Stangle(file.path("RNowebSource", "predCellLines.Rnw"),
```

```
+         output = pred)
+     source(pred)
+ }
> rm(pred, chem)
```

The third code chunk loads the specific information for the docetaxel response of the NCI60 cell lines.

```
> previous <- file.path("Tangled", "gi50ValuesOverlap.R")
> docegi50 <- file.path("RDataObjects", "doceGI50.Rda")
> if (file.exists(docegi50)) {
+     load(docegi50)
+ } else {
+     Stangle(file.path("RNowebSource", "gi50ValuesOverlap.Rnw"),
+         output = previous)
+     source(previous)
+ }
> rm(previous, docegi50)
```

## 3 Preprocess the Affymetrix data from Novartis

Load the libraries we need for the analysis:

```
> library(affy)
> library(ClassComparison)
> library(ClassDiscovery)
```

Use the `affy` library to perform quantile normalization of the full set of 180 arrays run by Novartis. After normalizing we transform the data by computing the base-two logarithm

```
> normed <- normalize.quantiles(as.matrix(novartis))
> dimnames(normed) <- dimnames(novartis)
> novartisNL <- log(normed, 2)
> rm(normed)
```

```
> L <- levels(novartisInfo$cellname)
> temp <- sapply(L, function(x) {
+     v <- novartisInfo$cellname == x
+     1/sum(v) * v
+ })
> novartisNLAvg <- novartisNL %*% temp
> rm(L, temp)
```

# 4 Pick out Sensitive and Resistant Subsets

There are two different sets of cell lines we want to work with:

1. The cell lines Potti et al. selected as sensitive or resistant to docetaxel
2. The cell lines we selected as sensitive or resistant to docetaxel

The next function produces named factors indicating which cell lines are resistant and which are sensitive. Note that the list of resistant cell line names must be the first argument.

```
> makeRSFactor <- function(rn, sn) {
+     temp <- c(rn, sn)
+     resp <- rep(c("Resistant", "Sensitive"), times = c(length(rn),
+         length(sn)))
+     names(resp) <- c(rn, sn)
+     factor(resp)
+ }
> ourCells <- makeRSFactor(ourCellnameDoceResi, ourCellnameDoceSens)
> pottiCells <- makeRSFactor(pottiCellnameDoceResi, pottiCellnameDoceSens)
```

Given the names and the response, we want to (i) pick the relevant cell line data out of the full Novartis data set and (2) produce a corresponding information data frame that includes the response. Here we do this for both our chosen cell lines and the cell lines selected by Potti and colleagues.

```
> makeSubset <- function(cellresp) {
+     anc <- as.character(novartisInfo[, "cellname"])
+     useful <- anc %in% names(cellresp)
+     subnova <- novartisNL[, useful]
+     subinfo <- novartisInfo[useful, ]
+     subinfo[, "Response"] <- cellresp[as.character(subinfo$cellname)]
+     dangd <- which(subinfo$ID == "D")
+     subinfo <- subinfo[-dangd, ]
+     subnova <- subnova[, -dangd]
+     subinfo$ID <- factor(subinfo$ID)
+     subinfo$cellname <- factor(subinfo$cellname)
+     list(NL = subnova, Info = subinfo)
+ }
> ourSubset <- makeSubset(ourCells)
> pottiSubset <- makeSubset(pottiCells)
```

## 5 Five analyses of the genes that are different using our cell lines

We are going to make use of the fact that Sweave allows us to reuse data chunks. Parts of the analysis are encoded in reusable functions. For example, we want to get the probe set IDs associated with the "top $n$" genes based on a vector $x$ of p-values from a statistical analysis performed on some data set.

```
> top <- function(x, n, data) {
+     target <- sort(x)[n]
+     sort(rownames(data)[x <= target])
+ }
```

Now, we set some parameters:

```
> subinfo <- ourSubset$Info
> subnova <- ourSubset$NL
> theCells <- ourCells
```

Next, we perform a set of five analyses on these data. First, we work with each of the A, B, and C Novartis data sets separately and use equal-variance two-sample t-tests to rank the genes.

```
> mtt.a <- MultiTtest(subnova[, subinfo$ID == "A"], subinfo[subinfo$ID ==
+     "A", "Response"])
> bum.a <- Bum(mtt.a@p.values)
> mtt.b <- MultiTtest(subnova[, subinfo$ID == "B"], subinfo[subinfo$ID ==
+     "B", "Response"])
> bum.b <- Bum(mtt.b@p.values)
> mtt.c <- MultiTtest(subnova[, subinfo$ID == "C"], subinfo[subinfo$ID ==
+     "C", "Response"])
> bum.c <- Bum(mtt.c@p.values)
```

Then we also perform two joint analyses using the combined data. One of these pretends that the cell lines are independent, giving three times as meany degrees of freedom to perform a t-test.

```
> mtt.joint <- MultiTtest(subnova, subinfo[, "Response"])
> bum.joint <- Bum(mtt.joint@p.values)
```

The final analysis builds a model that averages over cell lines.

```
> mtt.average <- MultiTtest(novartisNLAvg[, names(theCells)],
+     theCells)
> bum.average <- Bum(mtt.average@p.values)
```

The next step is to extract the top $N$ genes from each of the five analyses. For docetaxel, following Potti and colleagues, we take $N = 50$.

```
> N <- 50
> average50 <- top(mtt.average@p.values, N, subnova)
> joint50 <- top(mtt.joint@p.values, N, subnova)
> a50 <- top(mtt.a@p.values, N, subnova)
> b50 <- top(mtt.b@p.values, N, subnova)
> c50 <- top(mtt.c@p.values, N, subnova)
> alltogether <- sort(unique(c(average50, joint50, a50,
+     b50, c50)))
> getall <- which(rownames(subnova) %in% alltogether)
> fiveP <- data.frame(Average = mtt.average@p.values, Joint = mtt.joint@p.values,
+     A = mtt.a@p.values, B = mtt.b@p.values, C = mtt.c@p.values)[getall,
+     ]
> library(hgu95av2)
> sym <- unlist(mget(alltogether, envir = hgu95av2SYMBOL))
> fiveP <- data.frame(GeneSymbol = sym, fiveP)
```

### 5.1 How well do these gene lists agree?

In this section, we count the number of genes in common for each pair of gene lists.

```
> sew <- function(thread) {
+     a <- rep(FALSE, length(alltogether))
+     names(a) <- alltogether
+     a[thread] <- TRUE
+     a
+ }
> temp <- 1 * data.frame(Average = sew(average50), Joint = sew(joint50),
+     A = sew(a50), B = sew(b50), C = sew(c50))
> temp <- as.matrix(temp)
> overlap <- t(temp) %*% temp
> rm(temp)
```

We display the results in Table 1.

| | Average | Joint | A | B | C |
|---|---|---|---|---|---|
| Average | 50 | 21 | 12 | 17 | 10 |
| Joint | 21 | 50 | 12 | 15 | 21 |
| A | 12 | 12 | 50 | 7 | 4 |
| B | 17 | 15 | 7 | 50 | 7 |
| C | 10 | 21 | 4 | 7 | 50 |

Table 1: Overlap in the top 50 genes using different sets of replicate arrays.

### 5.2 Which list seems to be the most reliable?

We can use the BUM analyses performed above to estimate the false discovery rate (FDR) associated with each gene list.

```
> baz <- function(x, b) abs(50 - countSignificant(b, x))
> fdr.a <- optimize(baz, c(0, 1), b = bum.a)$minimum
> fdr.b <- optimize(baz, c(0, 1), b = bum.b)$minimum
> fdr.c <- optimize(baz, c(0, 1), b = bum.c)$minimum
> fdr.joint <- optimize(baz, c(0, 1), b = bum.joint)$minimum
> fdr.average <- optimize(baz, c(0, 1), b = bum.average)$minimum
> rm(baz)

> fdr.a

[1] 0.03954799

> fdr.b

[1] 0.1299731

> fdr.c

[1] 0.2841383

> fdr.joint

[1] 6.62732e-05

> fdr.average

[1] 0.07919994
```

We note that Series C has the worst FDR rate, A is the best, and the average data is not quite as good as A. Note that the FDR estimate for the "joint" analysis is unrealistic, since it treats the replicates as though they were statistically independent.

### 5.3 Getting ready to save stuff

Now we bundle things up in such a way that we can use them later.

```
> write.table(fiveP, file = file.path("Results", "ourCells-pValues.tsv"),
+     sep = "\t", quote = FALSE, col.names = NA, na = "")
> ourStuff <- list(Features = list(Average = average50,
+     Joint = joint50, A = a50, B = b50, C = c50), FDR = list(Average = fdr.average,
+     Joint = fdr.joint, A = fdr.a, B = fdr.b, C = fdr.c),
+     FiveP = fiveP, Overlap = overlap)
```

## 6 Repeating the analysis for cell lines from Potti

We simply reset the parameters to use the cell lines from Potti and colleagues instead of the cell lines we selected, and then we rerun the same code.

```
> subinfo <- pottiSubset$Info
> subnova <- pottiSubset$NL
> theCells <- pottiCells

> mtt.a <- MultiTtest(subnova[, subinfo$ID == "A"], subinfo[subinfo$ID ==
+     "A", "Response"])
> bum.a <- Bum(mtt.a@p.values)
> mtt.b <- MultiTtest(subnova[, subinfo$ID == "B"], subinfo[subinfo$ID ==
+     "B", "Response"])
> bum.b <- Bum(mtt.b@p.values)
> mtt.c <- MultiTtest(subnova[, subinfo$ID == "C"], subinfo[subinfo$ID ==
+     "C", "Response"])
> bum.c <- Bum(mtt.c@p.values)
> mtt.joint <- MultiTtest(subnova, subinfo[, "Response"])
> bum.joint <- Bum(mtt.joint@p.values)
> mtt.average <- MultiTtest(novartisNLAvg[, names(theCells)],
+     theCells)
> bum.average <- Bum(mtt.average@p.values)
> N <- 50
> average50 <- top(mtt.average@p.values, N, subnova)
> joint50 <- top(mtt.joint@p.values, N, subnova)
> a50 <- top(mtt.a@p.values, N, subnova)
> b50 <- top(mtt.b@p.values, N, subnova)
> c50 <- top(mtt.c@p.values, N, subnova)
> alltogether <- sort(unique(c(average50, joint50, a50,
+     b50, c50)))
> getall <- which(rownames(subnova) %in% alltogether)
> fiveP <- data.frame(Average = mtt.average@p.values, Joint = mtt.joint@p.values,
+     A = mtt.a@p.values, B = mtt.b@p.values, C = mtt.c@p.values)[getall,
+     ]
> library(hgu95av2)
> sym <- unlist(mget(alltogether, envir = hgu95av2SYMBOL))
> fiveP <- data.frame(GeneSymbol = sym, fiveP)
> sew <- function(thread) {
+     a <- rep(FALSE, length(alltogether))
+     names(a) <- alltogether
+     a[thread] <- TRUE
+     a
```

```
+ }
> temp <- 1 * data.frame(Average = sew(average50), Joint = sew(joint50),
+     A = sew(a50), B = sew(b50), C = sew(c50))
> temp <- as.matrix(temp)
> overlap <- t(temp) %*% temp
> rm(temp)
```

### 6.1 How well do these gene lists agree?

In this section, we display the results in Table 2.

| | Average | Joint | A | B | C |
|---|---|---|---|---|---|
| Average | 50 | 21 | 13 | 9 | 6 |
| Joint | 21 | 50 | 11 | 9 | 9 |
| A | 13 | 11 | 50 | 2 | 0 |
| B | 9 | 9 | 2 | 50 | 3 |
| C | 6 | 9 | 0 | 3 | 50 |

Table 2: Overlap in the top 50 genes using different sets of replicate arrays.

### 6.2 Which list seems to be the most reliable?

We can use the BUM analyses performed above to estimate the false dicovery rate (FDR) associated with each gene list.

```
> baz <- function(x, b) abs(50 - countSignificant(b, x))
> fdr.a <- optimize(baz, c(0, 1), b = bum.a)$minimum
> fdr.b <- optimize(baz, c(0, 1), b = bum.b)$minimum
> fdr.c <- optimize(baz, c(0, 1), b = bum.c)$minimum
> fdr.joint <- optimize(baz, c(0, 1), b = bum.joint)$minimum
> fdr.average <- optimize(baz, c(0, 1), b = bum.average)$minimum
> rm(baz)
> fdr.a

[1] 0.0402898

> fdr.b

[1] 0.206228

> fdr.c

[1] 0.4985472
```

```
> fdr.joint

[1] 0.001778902

> fdr.average

[1] 0.2580193
```

Here only the A arrays have a decent FDR.

### 6.3 Getting ready to save stuff

Now we bundle things up in such a way that we can use them later.

```
> write.table(fiveP, file = file.path("Results", "pottiCells-pValues.tsv"),
+     sep = "\t", quote = FALSE, col.names = NA, na = "")
> pottiStuff <- list(Features = list(Average = average50,
+     Joint = joint50, A = a50, B = b50, C = c50), FDR = list(Average = fdr.average,
+     Joint = fdr.joint, A = fdr.a, B = fdr.b, C = fdr.c),
+     FiveP = fiveP, Overlap = overlap)
```

## 7 The features reported by Potti and colleagues

In the supplementary material on the Nature Medicine web site, Potti and colleagues include a PDF table that contains a list of the features (genes) that they used in their predictive models foir different drugs. Using cut-and-paste, we copied the entire table and saved it as a text file (Table1.txt). In this section, we read that information into R in order to compare their lists of features to ours.

```
> geneProc <- function(filename = "../Original-info/Extracts/Table1.txt") {
+     featureTables <- read.table(file = filename, sep = "\n",
+         quote = "")
+     predictorEntries <- as.character(featureTables[,
+         1])
+     predictorBoundaries <- grep(pattern = "PREDICTOR",
+         predictorEntries, ignore.case = TRUE)
+     predictorBoundaries <- predictorBoundaries[2:9]
+     probesetRows <- grep(pattern = "_at$", predictorEntries)
+     drugSignatures <- list()
+     for (i1 in 1:7) {
+         drugSignatures[[i1]] <- predictorEntries[probesetRows[(probesetRows >
+             predictorBoundaries[i1]) & (probesetRows <
+             predictorBoundaries[i1 + 1])]]
```

```
+     }
+     names(drugSignatures) <- predictorEntries[predictorBoundaries[1:7]]
+     return(drugSignatures)
+ }
```

Using this function, we load all the features, and then extract the list of features for docetaxel.

```
> reportedFeatures <- geneProc(file.path("..", "PublicData",
+     "NatureMedicine", "Table1.txt"))
> pottiReportedDoce <- reportedFeatures$DOCE
```

Now we count the overlap with our list of genes.

```
[1] 2
```

```
[1] 1
```

With only 1 each, there is nothing to compare. However, our analysis of the other drugs suggested that there was likely to be an off-by-one error.

```
> pottiOBOE <- rownames(novartis)[1 + which(rownames(novartis) %in%
+     pottiReportedDoce)]
> sum(pottiOBOE %in% rownames(ourStuff$FiveP))
```

```
[1] 2
```

```
> sum(pottiOBOE %in% rownames(pottiStuff$FiveP))
```

```
[1] 29
```

```
> sum(pottiOBOE %in% pottiStuff$Features$A)
```

```
[1] 29
```

```
> sum(pottiOBOE %in% pottiStuff$Features$B)
```

```
[1] 2
```

```
> sum(pottiOBOE %in% pottiStuff$Features$C)
```

```
[1] 0
```

```
> sum(pottiOBOE %in% pottiStuff$Features$Joint)
```

```
[1] 8
```

```
> sum(pottiOBOE %in% pottiStuff$Features$Batch)
```

```
[1] 0
```

# 8 Wrapup

Clean up the stuff we no longer need.

```
> rm(fiveP, getall, alltogether, sym, N)
> rm(mtt.a, mtt.b, mtt.c, mtt.joint, mtt.average)
> rm(makeSubset, makeRSFactor, top, sew)
> rm(subinfo, subnova, overlap)
> rm(bum.a, bum.b, bum.c, bum.average, bum.joint)
> rm(fdr.a, fdr.b, fdr.c, fdr.average, fdr.joint)
> rm(a50, b50, c50, average50, joint50)
```

Save the stuff we will want to use again.

```
> save(ourStuff, ourCells, ourSubset, pottiStuff, pottiCells,
+     pottiSubset, novartisNL, novartisNLAvg, reportedFeatures,
+     pottiReportedDoce, pottiOBOE, file = file.path("RDataObjects",
+         "features.Rda"))
```