Defining Pathway Signatures and Scores

Keith A. Baggerly, E. Shannon Neeley and Kevin R. Coombes

October 19, 2007

1 Introduction

In addition to using a signature to separate responders from nonresponders, Dressman et al computed pathway scores from all of the ovarian tumor array profiles. These were then examined to see if levels of pathway expression correlated with overall survival in recognizable subgroups (in particular, the nonresponders).

Here, we attempt to compute these pathway scores as well.

In order to define signatures of pathway disturbance, Bild et al worked with a set of cell lines. These cell lines were either undisturbed (GFP), or had been tweaked to break the key element of one specific pathway. The five pathways that they considered were Src, E2F3, Myc, Ras, and β-catenin. For each of these, they gathered approximately 10 U133+ arrays (there were 7 for Src, 9 for Bcat, and 9 for E2F3).

The CEL files for these arrays are available from the Duke web site for the Bild et al paper, `http://data.genome.duke.edu/oncogene.php`, as "PathwayPredictors.zip". This zip file also contains the CEL files for 17 tumor cell lines that they investigated, looking for "high" or "low" levels of alteration.

To identify the genes involved, they used two-sample t-tests to contrast the GFP group with the various subsets of CEL files. The lists of genes that they found were supplied as supplementary info on the Nature web site, and are also available on the Duke web site for the paper. The quantifications of these arrays (using MAS5.0) are available from GEO as GSE3151.

Since we are working with ovarian samples that were quantified using RMA, we decided to obtain the CEL files (just the 55 involving GFP or one pathway) and to quantify them using justRMA.

Once these are quantified, we use two-sample t-tests to define a signature involving the same number of genes that they report in each case. We also check the amount of overlap between the genes they reported and the ones that we find.

We then restrict our attention to the subsets of these gene lists that involve probesets that are present on the U133A platform. When the lists are thus restricted, we then center and scale all of the rows (genes) and compute the first singular vector, which tells us how the expression values should be weighted. Given these weights, we then compute "pathway scores" for each pathway/tumor combination.

2 Options and Libraries

```
> options(width = 80)
```

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

```
> load(file.path("RDataObjects", "ovcaRMAFromCEL.Rda"))
> load(file.path("RDataObjects", "ovcaRMAFromXLS.Rda"))
> load(file.path("RDataObjects", "ovcaRMAFromCELResids.Rda"))
> load(file.path("RDataObjects", "ovcaRMAFromXLSResids.Rda"))
> load(file.path("RDataObjects", "celRunDate.Rda"))
> load(file.path("RDataObjects", "clinicalInfo.Rda"))
```

Sort data.

```
> ovcaRMAFromCEL <- ovcaRMAFromCEL[, rownames(clinicalInfo)]
> ovcaRMAFromXLS <- ovcaRMAFromXLS[, rownames(clinicalInfo)]
> ovcaRMAFromCELResids <- ovcaRMAFromCELResids[, rownames(clinicalInfo)]
> ovcaRMAFromXLSResids <- ovcaRMAFromXLSResids[, rownames(clinicalInfo)]
> celRunDate <- celRunDate[rownames(clinicalInfo)]
```

3 Load Pathway Array Data and Gene Lists

Here, we compute the RMA expression values for the 55 CEL files involving pathway expression levels.

```
> pathwayCELDir <- file.path("OtherData", "BildNature06", "Andrea")
> pathwayCELs <- dir(pathwayCELDir)
> rda <- "pathwayRMAEset"
> rdaFile <- paste("RDataObjects", paste(rda, "Rda", sep = "."),
+     sep = .Platform$file.sep)
> if (file.exists(rdaFile)) {
+     cat(paste("loading", rda, "from cache\n"))
+     load(rdaFile)
+ } else {
+     pathwayRMAEset <- justRMA(filenames = paste(pathwayCELDir,
+         pathwayCELs, sep = .Platform$file.sep))
+     save(pathwayRMAEset, file = rdaFile)
+ }

loading pathwayRMAEset from cache
```

Now, given the Eset, we extract the Exprs and come up with abbreviated names.

```
> pathwayRMA <- exprs(pathwayRMAEset)
> tempNames <- pathwayCELs
> tempNames[1:3]

[1] "0159_6229_h133+_GFP-1_cel.txt" "0159_6230_h133+_GFP-2_cel.txt"
[3] "0159_6231_h133+_GFP-3_cel.txt"

> tempNames2 <- strsplit(tempNames, "\\+_")
> tempNames2 <- unlist(lapply(tempNames2, function(x) {
+     x[2]
+ }))
> tempNames2[1:3]

[1] "GFP-1_cel.txt" "GFP-2_cel.txt" "GFP-3_cel.txt"
```

```
> tempNames3 <- unlist(strsplit(tempNames2, "_cel.txt"))
> tempNames3[1:3]

[1] "GFP-1" "GFP-2" "GFP-3"

> pathwayShortNames <- tempNames3
> names(pathwayCELs) <- pathwayShortNames
> colnames(pathwayRMA) <- pathwayShortNames
> rm(tempNames, tempNames2, tempNames3)
```

Now that we have the expression values, we also want to have the lists of genes that they identified on hand. We have saved each of these in individual tab-delimited text files for easier loading. (Our files contain just the probesets and gene symbols, not the descriptions. Further, we don't need the symbols for our purposes here.)

```
> pathways <- c("Bcat", "E2F3", "Myc", "Ras", "Src")
> pathwayGenesBild <- vector("list", length(pathways))
> names(pathwayGenesBild) <- pathways
> for (i1 in 1:length(pathways)) {
+     pathwayGenesBild[[i1]] <- read.table(file.path("OtherData",
+         "BildNature06", paste(pathways[i1], "txt", sep = ".")),
+         colClasses = "character")[, 1]
+ }
> lapply(pathwayGenesBild, length)

$Bcat
[1] 98

$E2F3
[1] 298

$Myc
[1] 248

$Ras
[1] 348

$Src
[1] 73

> pathwayGenesBild$Bcat[1:3]

[1] "225098_at"   "218150_at"   "222667_s_at"
```

4 Compute Gene Lists

Given that we used a different quantification method (RMA) than Bild et al (MAS5.0), we'd like to produce our own gene lists and also to check that the ones initially proposed still seem reasonable. We will follow Bild et al in using two-sample t-tests to select the genes, and in the number of genes used for each list.

```
> pathwayGenes <- vector("list", length(pathways))
> names(pathwayGenes) <- pathways
> tempGFP <- pathwayRMA[, grep("GFP", colnames(pathwayRMA), ignore.case = TRUE)]
> tempPvalues <- matrix(0, dim(pathwayRMA)[1], length(pathways))
> rownames(tempPvalues) <- rownames(pathwayRMA)
> colnames(tempPvalues) <- pathways
> for (i1 in 1:length(pathways)) {
+     tempPathway <- pathwayRMA[, grep(pathways[i1], colnames(pathwayRMA),
+         ignore.case = TRUE)]
+     tempFactor <- rep(c("Baseline", "Experiment"), times = c(dim(tempGFP)[2],
+         dim(tempPathway)[2]))
+     tempFactor <- as.factor(tempFactor)
+     tempT <- MultiTtest(cbind(tempGFP, tempPathway), tempFactor)
+     tempPvalues[, i1] <- tempT@p.values
+     tempCutoff <- sort(tempT@p.values)[length(pathwayGenesBild[[pathways[i1]]])]
+     pathwayGenes[[pathways[i1]]] <- rownames(pathwayRMA)[tempT@p.values <=
+         tempCutoff]
+ }
```

At this point, we've assembled our lists. We'd also like to check how much overlap there is between our lists and the ones initially supplied.

```
> tempOverlap <- matrix(0, length(pathways), 4)
> rownames(tempOverlap) <- pathways
> colnames(tempOverlap) <- c("nGenes", "nOverlap", "minPvalue",
+     "maxPvalue")
> for (i1 in 1:length(pathways)) {
+     tempOverlap[i1, "nGenes"] <- length(pathwayGenes[[i1]])
+     tempOverlap[i1, "nOverlap"] <- length(intersect(pathwayGenes[[i1]],
+         pathwayGenesBild[[i1]]))
+     tempOverlap[i1, 3:4] <- range(tempPvalues[pathwayGenesBild[[i1]],
+         pathways[i1]])
+ }
> tempOverlap
```

```
     nGenes nOverlap    minPvalue    maxPvalue
Bcat     98       68 8.837375e-14 2.631748e-05
E2F3    298      233 0.000000e+00 4.852572e-06
Myc     248      186 0.000000e+00 2.156607e-04
Ras     348      307 0.000000e+00 5.729639e-08
Src      73       47 0.000000e+00 4.785899e-06
```

In general, somewhere between 2/3 and 3/4 of the genes in corresponding lists overlap. However, even though the overlap is not 100%, the largest p-values we see for genes in the lists initially supplied are all still pretty small. This suggests that they are still reasonable candidates, but that they just didn't make it to the top of the list. Given that different quantification methods were used, this strikes us as good agreement.

5 Compute Gene Weights and Cell Line Scores

There are a few steps to computing gene weights. First, we reduce our gene lists by restricting attention to just those probesets present on the U133A platform. Next, we assemble a matrix giving the expression values of these genes in just the samples from GFP and from the corresponding pathway. In order to make these results less dependent on absolute expression levels, we center and scale the results for each gene. We then compute a singular value decomposition (SVD); the weights will correspond to the elements of the first singular vector. We check signs for consistency; positive weights should correspond to the gene expression values being higher in the treated samples. Eventually, we're going to compute scores for both ovcaRMAFromCEL and ovcaRMAFromXLS, so we'd also like to know if the 100 probesets missing from ovcaRMAFromXLS (those with numerical parts in the 200000-200099 range) are used enough to be problematic. Let's check this first.

```
> unlist(lapply(pathwayGenes, function(x) {
+     grep("^2000", x)
+ }))

Myc
 16

> unlist(lapply(pathwayGenesBild, function(x) {
+     grep("^2000", x)
+ }))

Myc
141

> pathwayGenes$Myc[16]

[1] "200063_s_at"

> pathwayGenesBild$Myc[141]

[1] "200063_s_at"
```

Out of all of the gene lists, only the one for Myc uses one of the missing probesets. Since the total number of genes involved in the Myc signature is 248, we do not expect this to have a dramatic effect on the results.

Now let's shift to computing the weights. We begin with reducing to the U133A probesets.

```
> pathwayGenesU133A <- pathwayGenes
> for (i1 in 1:length(pathwayGenesU133A)) {
+     pathwayGenesU133A[[i1]] <- intersect(pathwayGenes[[i1]],
+         rownames(ovcaRMAFromCEL))
+ }
> pathwayGenesBildU133A <- pathwayGenesBild
> for (i1 in 1:length(pathwayGenesBildU133A)) {
+     pathwayGenesBildU133A[[i1]] <- intersect(pathwayGenesBild[[i1]],
+         rownames(ovcaRMAFromCEL))
+ }
> unlist(lapply(pathwayGenes, length))
```

```
Bcat E2F3  Myc  Ras  Src
  98  298  248  348   73
```

```
> unlist(lapply(pathwayGenesU133A, length))
```

```
Bcat E2F3  Myc  Ras  Src
  49  168  157  228   42
```

```
> unlist(lapply(pathwayGenesBildU133A, length))
```

```
Bcat E2F3  Myc  Ras  Src
  54  173  164  228   46
```

As we might have predicted, the number of genes involved drops by a bit less than half for most of the pathways. This should still be enough for us to get some redundancy.

Next, we extract the relevant submatrices, center and scale, and record the first singular vector. Given these weights, we also assign scores to the cell lines used to generate the signature.

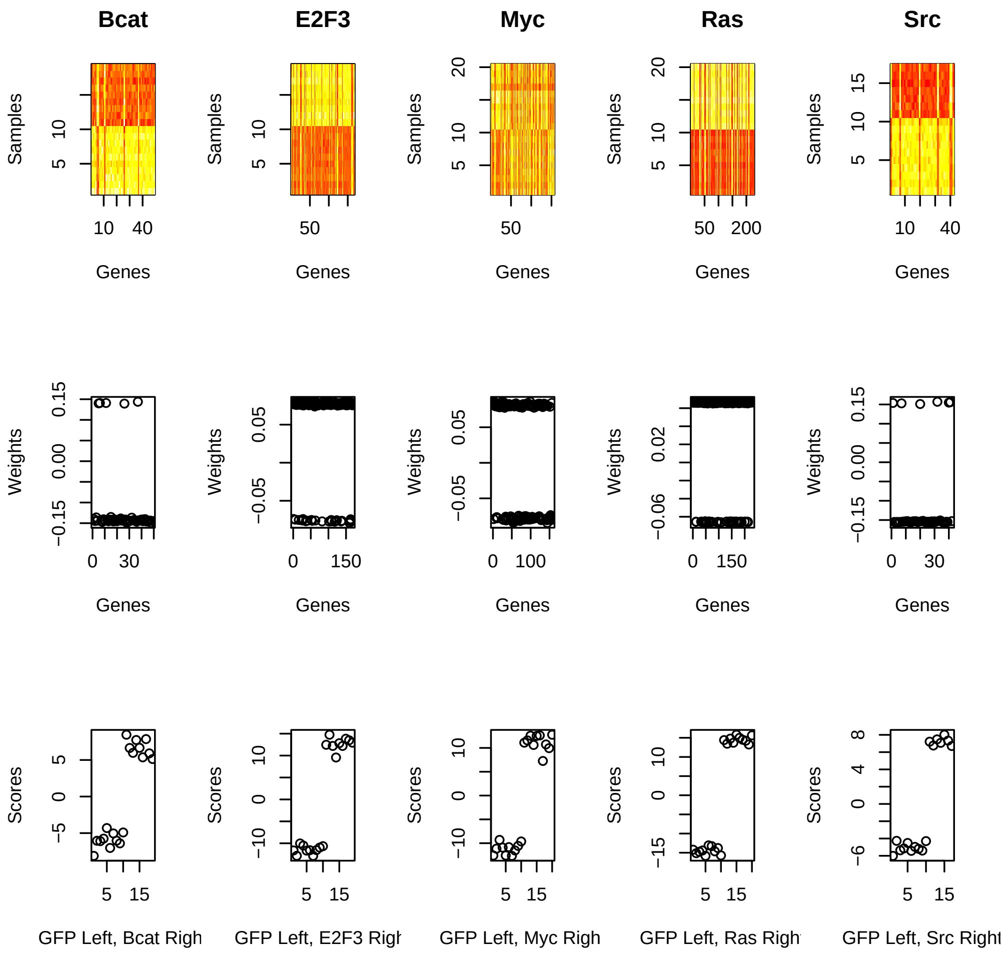
```
> pathwayWeightsU133A <- vector("list", length(pathways))
> names(pathwayWeightsU133A) <- pathways
> pathwayDataU133A <- vector("list", length(pathways))
> names(pathwayDataU133A) <- pathways
> tempGFP <- pathwayRMA[, grep("GFP", colnames(pathwayRMA), ignore.case = TRUE)]
> for (i1 in 1:length(pathways)) {
+     tempPathway <- pathwayRMA[, grep(pathways[i1], colnames(pathwayRMA),
+         ignore.case = TRUE)]
+     tempData <- cbind(tempGFP, tempPathway)
+     tempData <- tempData[pathwayGenesU133A[[i1]], ]
+     tempData <- t(scale(t(tempData)))
+     pathwayDataU133A[[i1]] <- tempData
+     tempWeights <- svd(tempData)$u[, 1]
+     names(tempWeights) <- pathwayGenesU133A[[i1]]
+     pathwayWeightsU133A[[i1]] <- tempWeights
+ }
```

Now, we want the sign of the weights to be such that the scores for the GFP samples will be negative and the scores for the treated samples will be positive. Let's compute the signs and scores.

```
> pathwayCellScoresU133A <- vector("list", length(pathways))
> names(pathwayCellScoresU133A) <- pathways
> for (i1 in 1:length(pathways)) {
+     tempScores <- pathwayWeightsU133A[[i1]] %*% pathwayDataU133A[[i1]]
+     if (median(tempScores[grep("GFP", colnames(tempScores))]) >
+         0) {
+         pathwayWeightsU133A[[i1]] <- -pathwayWeightsU133A[[i1]]
+         tempScores <- -tempScores
+     }
+     pathwayCellScoresU133A[[i1]] <- tempScores[1, ]
+ }
```

Before we go too much further, let's take a look at what the data, weights, and scores look like.

```
> par(mfrow = c(3, length(pathways)))
> for (i1 in 1:length(pathways)) {
+     image(1:dim(pathwayDataU133A[[i1]])[1], 1:dim(pathwayDataU133A[[i1]])[2],
+         pathwayDataU133A[[i1]], xlab = "Genes", ylab = "Samples",
+         main = pathways[i1])
+ }
> for (i1 in 1:length(pathways)) {
+     plot(pathwayWeightsU133A[[i1]], xlab = "Genes", ylab = "Weights")
+ }
> for (i1 in 1:length(pathways)) {
+     plot(pathwayCellScoresU133A[[i1]], xlab = paste("GFP Left,",
+         pathways[i1], "Right"), ylab = "Scores")
+ }
> par(mfrow = c(1, 1))
```



While this image is undeniably busy, we only want to make a few qualitative observations. First, the genes selected do not have ambiguous profiles; they are either on or off. Consequently, the weights given to these genes after the data has been standardized are all about the same magnitude. The overall score is, in effect, a signed sum of the standardized expression values. The signs of the weights appear to be correct, in that the GFP sample scores in the bottom row (always at left) are uniformly negative.

Now we repeat the above computations using the genes from the initially reported lists.

```
> pathwayWeightsBildU133A <- vector("list", length(pathways))
> names(pathwayWeightsBildU133A) <- pathways
> pathwayDataBildU133A <- vector("list", length(pathways))
> names(pathwayDataBildU133A) <- pathways
> tempGFP <- pathwayRMA[, grep("GFP", colnames(pathwayRMA), ignore.case = TRUE)]
> for (i1 in 1:length(pathways)) {
+     tempPathway <- pathwayRMA[, grep(pathways[i1], colnames(pathwayRMA),
+         ignore.case = TRUE)]
+     tempData <- cbind(tempGFP, tempPathway)
+     tempData <- tempData[pathwayGenesBildU133A[[i1]], ]
+     tempData <- t(scale(t(tempData)))
+     pathwayDataBildU133A[[i1]] <- tempData
+     tempWeights <- svd(tempData)$u[, 1]
+     names(tempWeights) <- pathwayGenesBildU133A[[i1]]
+     pathwayWeightsBildU133A[[i1]] <- tempWeights
+ }
> pathwayCellScoresBildU133A <- vector("list", length(pathways))
> names(pathwayCellScoresBildU133A) <- pathways
> for (i1 in 1:length(pathways)) {
+     tempScores <- pathwayWeightsBildU133A[[i1]] %*% pathwayDataBildU133A[[i1]]
+     if (median(tempScores[grep("GFP", colnames(tempScores))]) >
+         0) {
+         pathwayWeightsBildU133A[[i1]] <- -pathwayWeightsBildU133A[[i1]]
+         tempScores <- -tempScores
+     }
+     pathwayCellScoresBildU133A[[i1]] <- tempScores[1, ]
+ }
```

So, how well do these scores agree with the ones we got using the gene lists we computed?

```
> par(mfrow = c(2, 3))
> for (i1 in 1:length(pathways)) {
+     plot(pathwayCellScoresU133A[[i1]], pathwayCellScoresBildU133A[[i1]],
+         xlab = "Scores", ylab = "Scores (Bild)", main = pathways[i1])
+ }
> par(mfrow = c(1, 1))
```

In every case, the agreement is quite good. When the differences are this stark, and the lists are long enough to average out minor individual fluctuations, the fact that the gene lists do not precisely agree is not that important.

6 Compute Pathway Scores For Tumors

Now that we have the gene lists and weights, we wrap up by computing pathway scores for the tumor data.

Unfortunately, due to the combinatorics associated with various choices, we have 8 sets of scores to compute for each pathway. These are combinations of (a) CEL or XLS numbers, (b) computed or reported gene lists, and (c) with or without corrections for run batch.

6.1 pathwayScoresCEL

```
> pathwayScoresCEL <- vector("list", length(pathways))
> names(pathwayScoresCEL) <- pathways
> pathwayDataCEL <- vector("list", length(pathways))
> names(pathwayDataCEL) <- pathways
> for (i1 in 1:length(pathways)) {
+     tempData <- ovcaRMAFromCEL[pathwayGenesU133A[[i1]], rownames(clinicalInfo)]
+     tempData <- t(scale(t(tempData)))
+     pathwayDataCEL[[i1]] <- tempData
+     tempScores <- pathwayWeightsU133A[[i1]] %*% pathwayDataCEL[[i1]]
+     pathwayScoresCEL[[i1]] <- tempScores[1, ]
+ }
```

6.2 pathwayScoresXLS

```
> pathwayScoresXLS <- vector("list", length(pathways))
> names(pathwayScoresXLS) <- pathways
> pathwayDataXLS <- vector("list", length(pathways))
> names(pathwayDataXLS) <- pathways
> for (i1 in 1:length(pathways)) {
+     genesInXLS <- intersect(pathwayGenesU133A[[i1]], rownames(ovcaRMAFromXLS))
+     tempData <- ovcaRMAFromXLS[genesInXLS, rownames(clinicalInfo)]
+     tempData <- t(scale(t(tempData)))
+     pathwayDataXLS[[i1]] <- tempData
+     tempScores <- pathwayWeightsU133A[[i1]][genesInXLS] %*% pathwayDataXLS[[i1]]
+     pathwayScoresXLS[[i1]] <- tempScores[1, ]
+ }
```

6.3 pathwayScoresCELBild

```
> pathwayScoresCELBild <- vector("list", length(pathways))
> names(pathwayScoresCELBild) <- pathways
> pathwayDataCELBild <- vector("list", length(pathways))
> names(pathwayDataCELBild) <- pathways
> for (i1 in 1:length(pathways)) {
+     tempData <- ovcaRMAFromCEL[pathwayGenesBildU133A[[i1]], rownames(clinicalInfo)]
+     tempData <- t(scale(t(tempData)))
+     pathwayDataCELBild[[i1]] <- tempData
+     tempScores <- pathwayWeightsBildU133A[[i1]] %*% pathwayDataCELBild[[i1]]
+     pathwayScoresCELBild[[i1]] <- tempScores[1, ]
+ }
```

6.4 pathwayScoresXLSBild

```
> pathwayScoresXLSBild <- vector("list", length(pathways))
> names(pathwayScoresXLSBild) <- pathways
> pathwayDataXLSBild <- vector("list", length(pathways))
> names(pathwayDataXLSBild) <- pathways
> for (i1 in 1:length(pathways)) {
+     genesInXLS <- intersect(pathwayGenesBildU133A[[i1]], rownames(ovcaRMAFromXLS))
+     tempData <- ovcaRMAFromXLS[genesInXLS, rownames(clinicalInfo)]
+     tempData <- t(scale(t(tempData)))
+     pathwayDataXLSBild[[i1]] <- tempData
+     tempScores <- pathwayWeightsBildU133A[[i1]][genesInXLS] %*%
+         pathwayDataXLSBild[[i1]]
+     pathwayScoresXLSBild[[i1]] <- tempScores[1, ]
+ }
```

6.5 pathwayScoresCELBatch

```
> pathwayScoresCELBatch <- vector("list", length(pathways))
> names(pathwayScoresCELBatch) <- pathways
> pathwayDataCELBatch <- vector("list", length(pathways))
> names(pathwayDataCELBatch) <- pathways
> for (i1 in 1:length(pathways)) {
+     tempData <- ovcaRMAFromCELResids[pathwayGenesU133A[[i1]],
+         rownames(clinicalInfo)]
+     tempData <- t(scale(t(tempData)))
+     pathwayDataCELBatch[[i1]] <- tempData
+     tempScores <- pathwayWeightsU133A[[i1]] %*% pathwayDataCELBatch[[i1]]
+     pathwayScoresCELBatch[[i1]] <- tempScores[1, ]
+ }
```

6.6 pathwayScoresXLSBatch

```
> pathwayScoresXLSBatch <- vector("list", length(pathways))
> names(pathwayScoresXLSBatch) <- pathways
> pathwayDataXLSBatch <- vector("list", length(pathways))
> names(pathwayDataXLSBatch) <- pathways
> for (i1 in 1:length(pathways)) {
+     genesInXLS <- intersect(pathwayGenesU133A[[i1]], rownames(ovcaRMAFromXLSResids))
+     tempData <- ovcaRMAFromXLSResids[genesInXLS, rownames(clinicalInfo)]
+     tempData <- t(scale(t(tempData)))
+     pathwayDataXLSBatch[[i1]] <- tempData
+     tempScores <- pathwayWeightsU133A[[i1]][genesInXLS] %*% pathwayDataXLSBatch[[i1]]
+     pathwayScoresXLSBatch[[i1]] <- tempScores[1, ]
+ }
```

6.7 pathwayScoresCELBildBatch

```
> pathwayScoresCELBildBatch <- vector("list", length(pathways))
> names(pathwayScoresCELBildBatch) <- pathways
> pathwayDataCELBildBatch <- vector("list", length(pathways))
> names(pathwayDataCELBildBatch) <- pathways
> for (i1 in 1:length(pathways)) {
+     tempData <- ovcaRMAFromCELResids[pathwayGenesBildU133A[[i1]],
+         rownames(clinicalInfo)]
+     tempData <- t(scale(t(tempData)))
+     pathwayDataCELBildBatch[[i1]] <- tempData
+     tempScores <- pathwayWeightsBildU133A[[i1]] %*% pathwayDataCELBildBatch[[i1]]
+     pathwayScoresCELBildBatch[[i1]] <- tempScores[1, ]
+ }
```

6.8 pathwayScoresXLSBildBatch

```
> pathwayScoresXLSBildBatch <- vector("list", length(pathways))
> names(pathwayScoresXLSBildBatch) <- pathways
> pathwayDataXLSBildBatch <- vector("list", length(pathways))
> names(pathwayDataXLSBildBatch) <- pathways
> for (i1 in 1:length(pathways)) {
+     genesInXLS <- intersect(pathwayGenesBildU133A[[i1]], rownames(ovcaRMAFromXLSResids))
+     tempData <- ovcaRMAFromXLSResids[genesInXLS, rownames(clinicalInfo)]
+     tempData <- t(scale(t(tempData)))
+     pathwayDataXLSBildBatch[[i1]] <- tempData
+     tempScores <- pathwayWeightsBildU133A[[i1]][genesInXLS] %*%
+         pathwayDataXLSBildBatch[[i1]]
+     pathwayScoresXLSBildBatch[[i1]] <- tempScores[1, ]
+ }
```

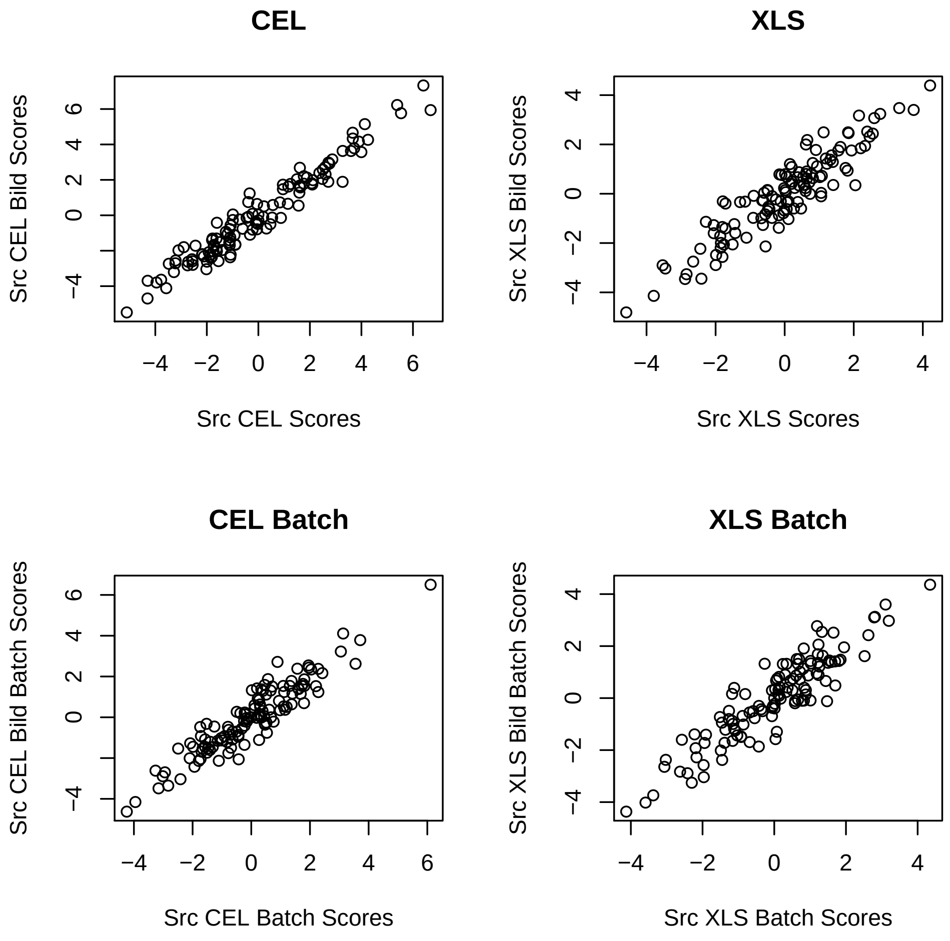
7 Comparing Scores

Given the various scores, we'd like to get a quick idea for how they relate to each other. We'll explore some of these contrasts using Src as our primary example. Given the 2^3 factorial nature of the different types of scores, we will focus on the main effects.

8 Choice of Gene List

```
> par(mfrow = c(2, 2))
> plot(pathwayScoresCEL$Src, pathwayScoresCELBild$Src, xlab = "Src CEL Scores",
+     ylab = "Src CEL Bild Scores", main = "CEL")
> plot(pathwayScoresXLS$Src, pathwayScoresXLSBild$Src, xlab = "Src XLS Scores",
+     ylab = "Src XLS Bild Scores", main = "XLS")
> plot(pathwayScoresCELBatch$Src, pathwayScoresCELBildBatch$Src,
+     xlab = "Src CEL Batch Scores", ylab = "Src CEL Bild Batch Scores",
+     main = "CEL Batch")
```

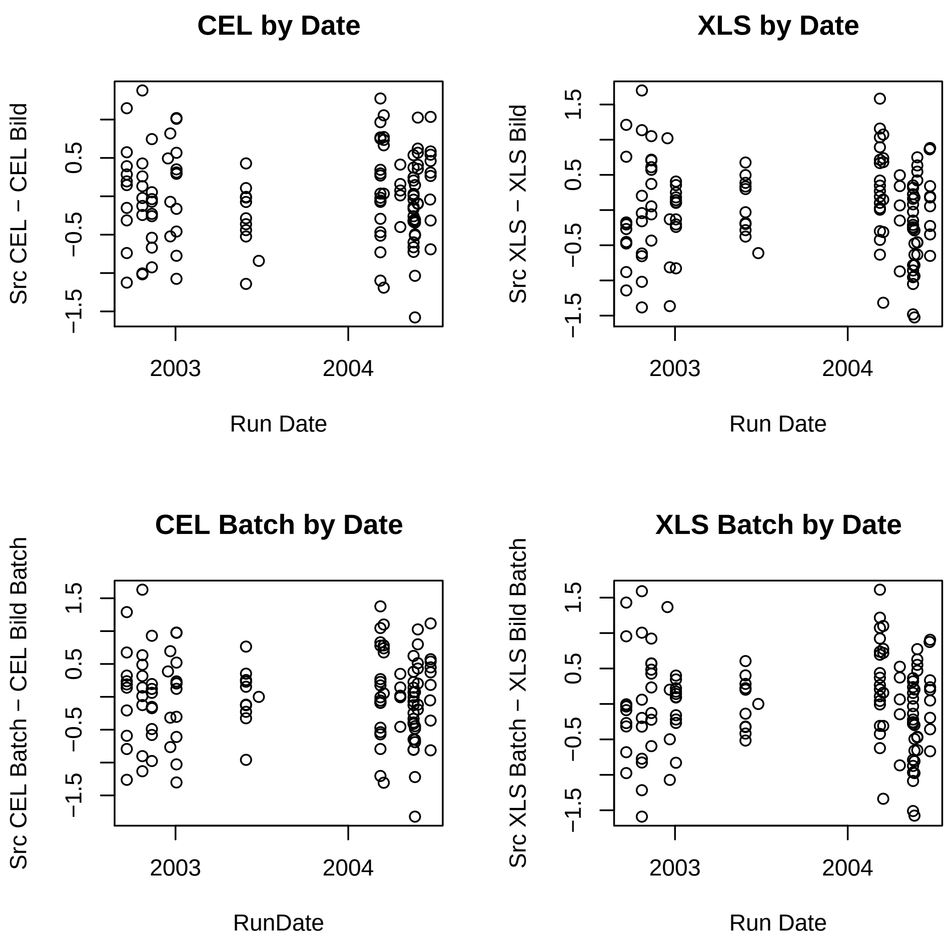
```
> plot(pathwayScoresXLSBatch$Src, pathwayScoresXLSBildBatch$Src,
+     xlab = "Src XLS Batch Scores", ylab = "Src XLS Bild Batch Scores",
+     main = "XLS Batch")
> par(mfrow = c(1, 1))
```



The scores are pretty tightly correlated. Shifting from one gene list to another here probably won't drastically affect the biological story.

```
> par(mfrow = c(2, 2))
> plot(celRunDate, pathwayScoresCEL$Src - pathwayScoresCELBild$Src,
+     xlab = "Run Date", ylab = "Src CEL - CEL Bild", main = "CEL by Date")
> plot(celRunDate, pathwayScoresXLS$Src - pathwayScoresXLSBild$Src,
+     xlab = "Run Date", ylab = "Src XLS - XLS Bild", main = "XLS by Date")
> plot(celRunDate, pathwayScoresCELBatch$Src - pathwayScoresCELBildBatch$Src,
+     xlab = "RunDate", ylab = "Src CEL Batch - CEL Bild Batch",
+     main = "CEL Batch by Date")
```

```
> plot(celRunDate, pathwayScoresXLSBatch$Src - pathwayScoresXLSBildBatch$Src,
+     xlab = "Run Date", ylab = "Src XLS Batch - XLS Bild Batch",
+     main = "XLS Batch by Date")
> par(mfrow = c(1, 1))
```

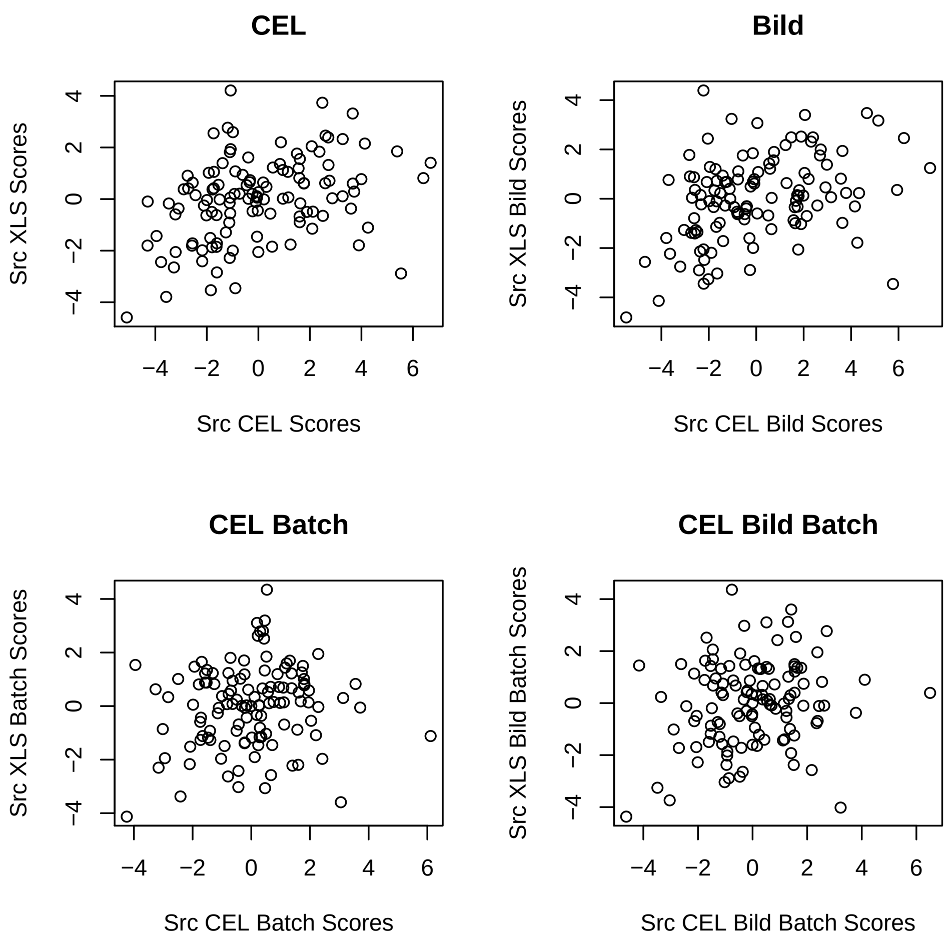


There's no real association between the differences in scores by Date (or Batch).

8.1 CEL vs XLS

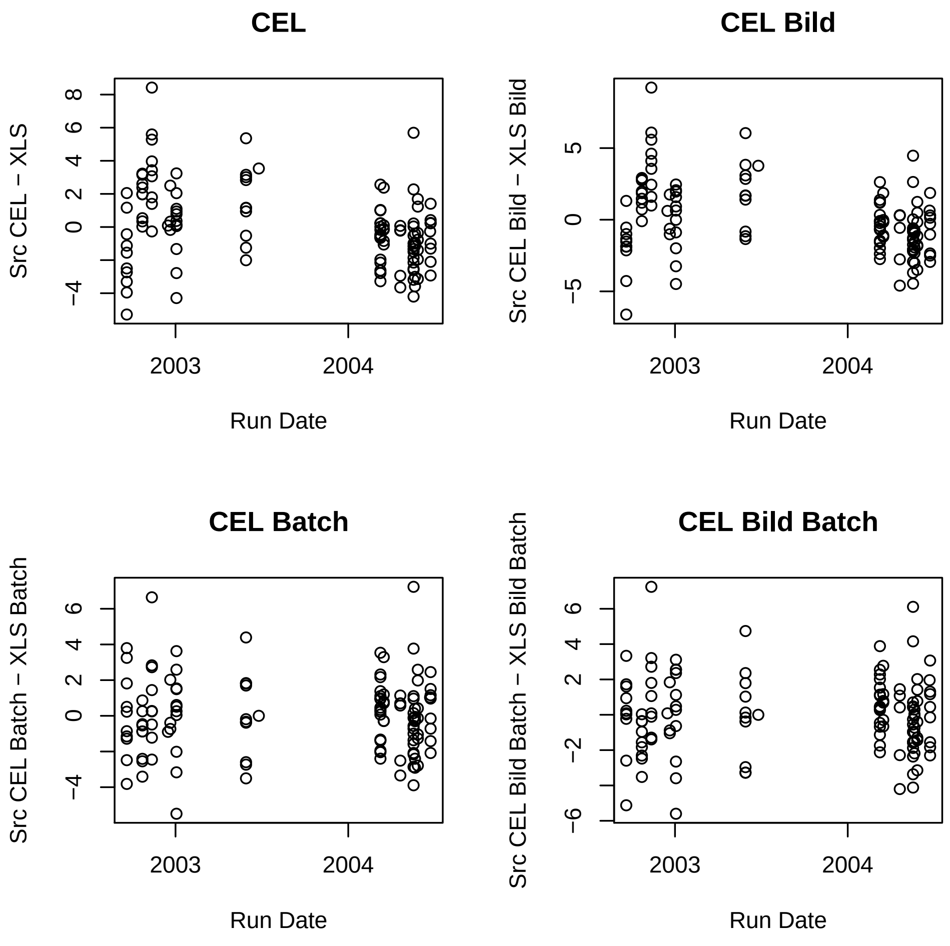
```
> par(mfrow = c(2, 2))
> plot(pathwayScoresCEL$Src, pathwayScoresXLS$Src, xlab = "Src CEL Scores",
+     ylab = "Src XLS Scores", main = "CEL")
> plot(pathwayScoresCELBild$Src, pathwayScoresXLSBild$Src, xlab = "Src CEL Bild Scores",
+     ylab = "Src XLS Bild Scores", main = "Bild")
> plot(pathwayScoresCELBatch$Src, pathwayScoresXLSBatch$Src, xlab = "Src CEL Batch Scores",
+     ylab = "Src XLS Batch Scores", main = "CEL Batch")
```

```
> plot(pathwayScoresCELBildBatch$Src, pathwayScoresXLSBildBatch$Src,
+     xlab = "Src CEL Bild Batch Scores", ylab = "Src XLS Bild Batch Scores",
+     main = "CEL Bild Batch")
> par(mfrow = c(1, 1))
```



Unsurprisingly, the scores are pretty pervasively scrambled. Very different samples will be identified as having high or low Src scores.

```
> par(mfrow = c(2, 2))
> plot(celRunDate, pathwayScoresCEL$Src - pathwayScoresXLS$Src,
+     xlab = "Run Date", ylab = "Src CEL - XLS", main = "CEL")
> plot(celRunDate, pathwayScoresCELBild$Src - pathwayScoresXLSBild$Src,
+     xlab = "Run Date", ylab = "Src CEL Bild - XLS Bild", main = "CEL Bild")
> plot(celRunDate, pathwayScoresCELBatch$Src - pathwayScoresXLSBatch$Src,
+     xlab = "Run Date", ylab = "Src CEL Batch - XLS Batch", main = "CEL Batch")
> plot(celRunDate, pathwayScoresCELBildBatch$Src - pathwayScoresXLSBildBatch$Src,
+     xlab = "Run Date", ylab = "Src CEL Bild Batch - XLS Bild Batch",
+     main = "CEL Bild Batch")
> par(mfrow = c(1, 1))
```

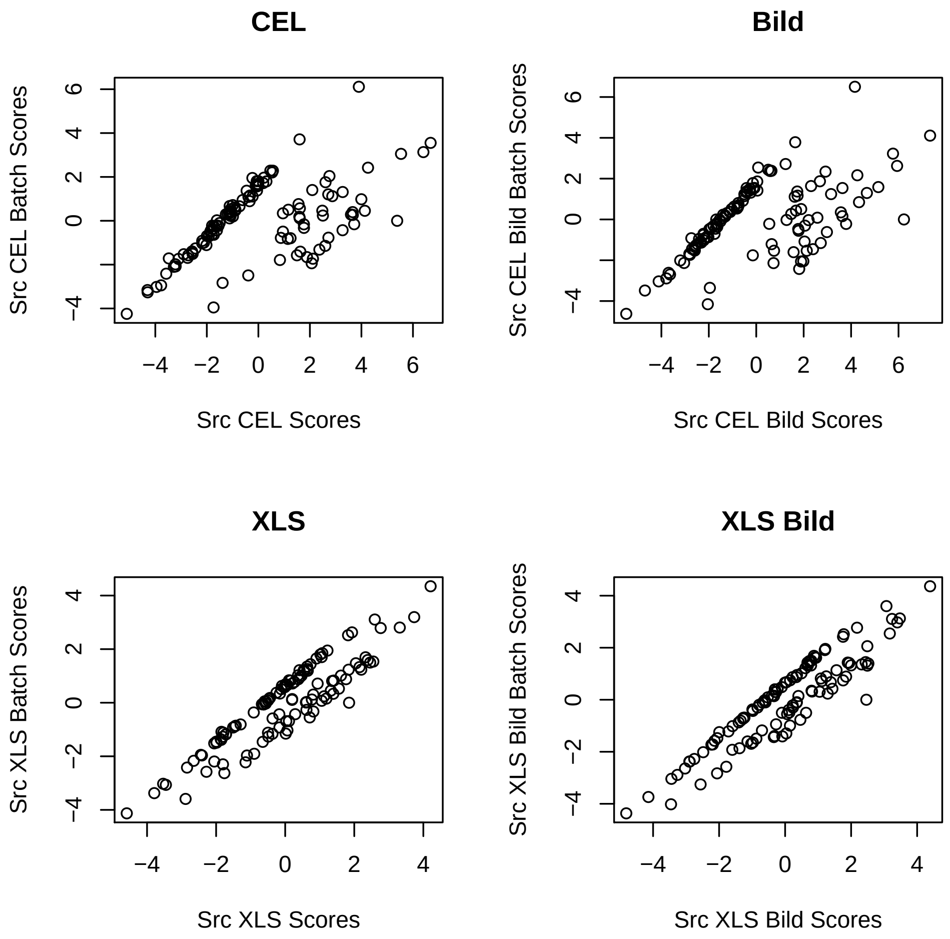


There are some systematic differences between CEL and XLS scores that are associated with Batch. This is possible because, as noted, the scrambling of names is not wholly random. In most cases in the early runs, the misnaming introduces an offset of a few spaces in run order.

8.2 Batch Correction

```
> par(mfrow = c(2, 2))
> plot(pathwayScoresCEL$Src, pathwayScoresCELBatch$Src, xlab = "Src CEL Scores",
+     ylab = "Src CEL Batch Scores", main = "CEL")
> plot(pathwayScoresCELBild$Src, pathwayScoresCELBildBatch$Src,
+     xlab = "Src CEL Bild Scores", ylab = "Src CEL Bild Batch Scores",
+     main = "Bild")
```

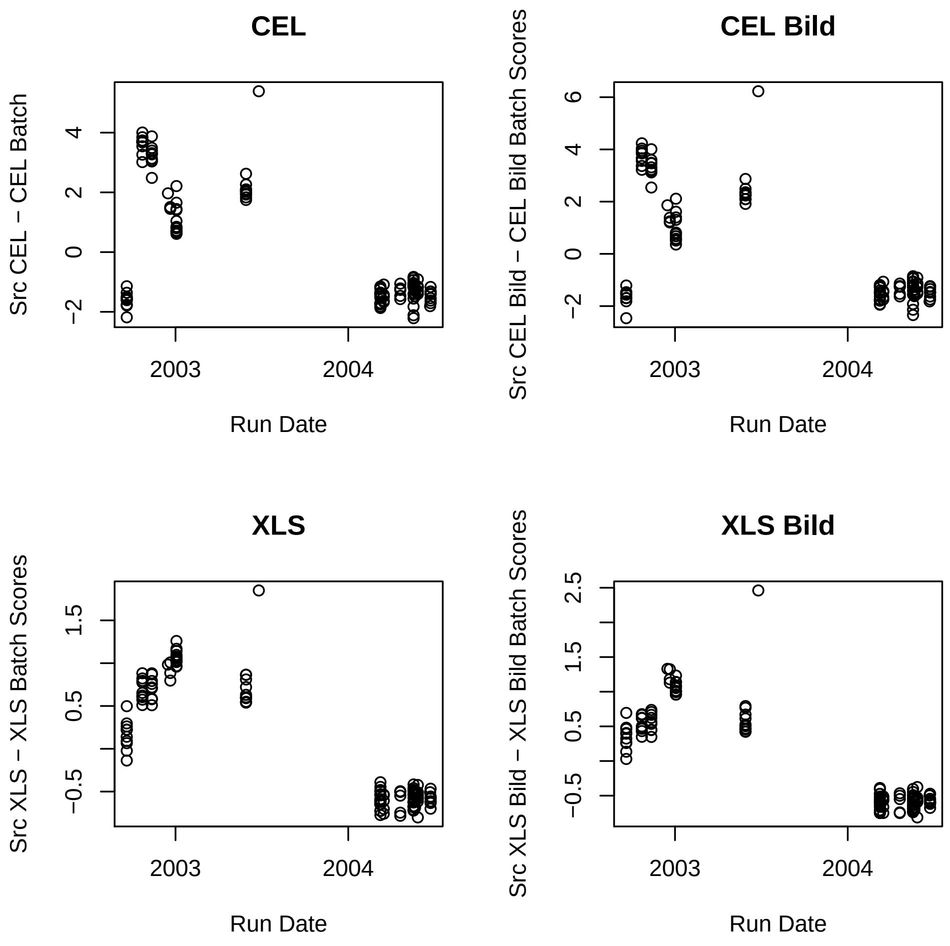
```
> plot(pathwayScoresXLS$Src, pathwayScoresXLSBatch$Src, xlab = "Src XLS Scores",
+     ylab = "Src XLS Batch Scores", main = "XLS")
> plot(pathwayScoresXLSBild$Src, pathwayScoresXLSBildBatch$Src,
+     xlab = "Src XLS Bild Scores", ylab = "Src XLS Bild Batch Scores",
+     main = "XLS Bild")
> par(mfrow = c(1, 1))
```



There are some very notable shifts with batch. In shifting from the CEL to the XLS scores, the batch effects are still visible, but they are more diffuse.

```
> par(mfrow = c(2, 2))
> plot(celRunDate, pathwayScoresCEL$Src - pathwayScoresCELBatch$Src,
+     xlab = "Run Date", ylab = "Src CEL - CEL Batch", main = "CEL")
> plot(celRunDate, pathwayScoresCELBild$Src - pathwayScoresCELBildBatch$Src,
+     xlab = "Run Date", ylab = "Src CEL Bild - CEL Bild Batch Scores",
+     main = "CEL Bild")
```

```
> plot(celRunDate, pathwayScoresXLS$Src - pathwayScoresXLSBatch$Src,
+     xlab = "Run Date", ylab = "Src XLS - XLS Batch Scores", main = "XLS")
> plot(celRunDate, pathwayScoresXLSBild$Src - pathwayScoresXLSBildBatch$Src,
+     xlab = "Run Date", ylab = "Src XLS Bild - XLS Bild Batch Scores",
+     main = "XLS Bild")
> par(mfrow = c(1, 1))
```



The batch effect on Src pathway scores is quite pronounced, and it is largest between batches run in the earlier (Berchuck et al) part of the experiment. The very large differences between the first two batches are the most problematic, as these are also very confounded with survival. The effect of using the XLS ordering is to smooth out the batch effects. Note that the vertical scales in the top and bottom rows are quite different.

9 Summary

1. When we compute gene lists from two-sample t-tests to find pathway signatures, we get between 2/3 and 3/4 of the genes reported by Bild et al. However, the largest p-values in the reported genes are still quite small (less than 10^{-4}), which is compatible with using different processing methods to quantify the data.
2. The selected genes are unambiguously on or off in the untreated and treated samples, and the resulting weights in the first principal component are roughly constant except for sign.
3. The scores are relatively robust to the precise gene list selected.
4. Differences in score by gene list are not confounded with run batch.
5. The scores themselves do appear to be confounded with run batch.
6. The scores based on the CEL data are uncorrelated with the scores based on the Excel data.
7. After correcting for the previously observed batch effects on gene expression, the scores change substantially.

10 Appendix

10.1 Saves

```
> save(pathwayCELDir, pathwayCELs, pathwayShortNames, pathways,
+     file = paste("RDataObjects", "pathwayNames.Rda", sep = .Platform$file.sep))
> save(pathwayRMA, file = paste("RDataObjects", "pathwayRMA.Rda",
+     sep = .Platform$file.sep))
> save(pathwayGenes, pathwayGenesBild, pathwayGenesU133A, pathwayGenesBildU133A,
+     file = paste("RDataObjects", "pathwayGenes.Rda", sep = .Platform$file.sep))
> save(pathwayWeightsU133A, pathwayWeightsBildU133A, file = paste("RDataObjects",
+     "pathwayWeights.Rda", sep = .Platform$file.sep))
> save(pathwayDataU133A, pathwayDataBildU133A, pathwayDataCEL,
+     pathwayDataXLS, pathwayDataCELBild, pathwayDataXLSBild, pathwayDataCELBatch,
+     pathwayDataXLSBatch, pathwayDataCELBildBatch, pathwayDataXLSBildBatch,
+     file = paste("RDataObjects", "pathwayData.Rda", sep = .Platform$file.sep))
> save(pathwayCellScoresU133A, pathwayCellScoresBildU133A, pathwayScoresCEL,
+     pathwayScoresXLS, pathwayScoresCELBild, pathwayScoresXLSBild,
+     pathwayScoresCELBatch, pathwayScoresXLSBatch, pathwayScoresCELBildBatch,
+     pathwayScoresXLSBildBatch, file = paste("RDataObjects", "pathwayScores.Rda",
+         sep = .Platform$file.sep))
```

10.2 SessionInfo

```
> sessionInfo()
```

```
R version 2.5.1 (2007-06-27)
i386-pc-mingw32
```

```
locale:
LC_COLLATE=English_United States.1252;LC_CTYPE=English_United States.1252;LC_MONETARY=English_United Sta

attached base packages:
[1] "splines"   "tools"     "stats"     "graphics"  "grDevices" "utils"
[7] "datasets"  "methods"   "base"

other attached packages:
     colorspace  ClassDiscovery         cluster ClassComparison      PreProcess
         "0.95"         "2.5.0"        "1.11.7"         "2.5.0"         "2.5.0"
      oompaBase            affy          affyio         Biobase
        "2.5.0"        "1.14.2"         "1.4.1"        "1.14.1"
```