Load and Check File Quantifications

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

October 9, 2007

1 Introduction

Here, we load the raw quantification data from Dressman et al, and compare these with the values that we obtain directly from the CEL files.

2 Options and Libraries

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

```
> library(affy)
> library(geneplotter)
```

3 Describing The Raw Data

The paper makes reference to 119 ovarian tumors and 12 ovarian cancer cell lines. The tumors were acquired from either Duke or the Moffitt cancer center. The tumor samples were all from patients treated with primary platinum-based therapy. Patients either exhibited a complete response (CR = 1) or an incomplete response (NR = 0). To build a predictive model, the tumor samples were randomly split into a training set of 83 samples and a test set of 36 samples. Most of the patients showed a complete response: 59/83 in the training set and 26/36 in the testing set. If we ignore the training/test divide, there 85 CR patients and 34 NR patients. We do not know which samples were in the training set and which samples were in the test set.

All of the tumor samples were run on Affy U133A chips, and the samples were quantified using the RMA methods in Bioconductor. The paper mentions that the cell lines were run on Affy U133+2's, but we didn't see any indication as to where the quantifications for the cell lines could be found.

The tumor samples were quantified using MAS5.0 in an earlier study (Nature 2006); those quantificiations were posted to GEO as GSE3149. The GEO posting actually lists values for 153 arrays, not 119. For 8 samples (1024, 1877, 2063, 2424, 2479, 2505, 2673, and 2739) there are two listings in GEO (e.g., 1024_a and 1024_b), suggesting that these may have been rerun. For these 8, we don't know which of these we're working with.

Further information is given at the Duke website, `http://data.cgt.duke.edu/platinum.php`. There are 4 files as of September 30, 2007.

- **PlatinumJCO.zip**. This contains the 119 initial CEL files from which the tumor sample quantifications were derived. The files are in version 3 (human readable) format.

- **correctedplatinum_RMA.xls**. This contains the RMA quantifications of the 119 tumor samples. We're not quite sure what the "corrected" in the name refers to. There are 22115 rows of data, as opposed to the 22283 probesets on the array; 168 probes have been excluded. There are 68 AFFX control probes on this chip, which are likely missing, but we don't know which others were omitted yet. The first row gives the sample id.
- **OVCclinicalinfo.xls**. This gives, for each of the 119 tumor samples, an ID to identify the array, post-treatment survival in months, grade, stage, debulking status (optimal or suboptimal), CA 125 post-treatment level, and NR/CR status (NR = 0, CR = 1).
- **Parameters for SSS software.txt**. This gives the parameter values they used when running their software. Unfortunately, several entries refer to files that were not posted as such (e.g., "data.txt", "response.txt", "weight2.txt", and "lung_censor.91.txt"), so these are useful only in terms of suggesting the input format. It also lists NVARIABLES = 6088, which may refer to a subset of the probesets available.

There is also a link to the software package used, SSS (for "shotgun stochastic search"), at `http://xpress.isds.duke.edu:8080/sss/`.

4 Loading the Raw Data

We begin by loading the quantifications provided; we saved the xls file in csv format to make this easier.

```
> rda <- "ovcaRMAFromXLS"
> 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 {
+     ovcaRMAFromXLS <- read.table(file.path("DukeWebSite", "correctedPlatinum_RMA.csv"),
+         header = TRUE, sep = ",", row.names = 1, check.names = FALSE)
+     ovcaRMAFromXLS <- as.matrix(ovcaRMAFromXLS)
+     save(ovcaRMAFromXLS, file = rdaFile)
+ }
```

```
loading ovcaRMAFromXLS from cache
```

Let's take a look at the first few values just to make sure everything looks ok.

```
> dim(ovcaRMAFromXLS)
```

```
[1] 22115   119
```

```
> ovcaRMAFromXLS[1:3, 1:5]
```

```
               0.08      1024      1447      1451      1504
1007_s_at 11.348198 10.326897 10.981040 10.751732 10.792526
1053_at    5.808186  6.420700  6.060294  6.123403  6.834273
117_at     7.062677  6.985818  7.089425  6.682984  6.974176
```

Looks ok.

We'd also like to get this information from the CEL files. We begin with a bit of parsing, in order to line the CEL file names up with the short identifiers used in ovcaRMAFromXLS.

```
> celFiles <- dir(file.path("DukeWebSite", "PlatinumJCO"))
> length(celFiles)

[1] 119

> celFiles[1:3]

[1] "0074_1772_h133a_872.cel"  "0074_1773_h133a_922.cel"
[3] "0074_1774_h133a_1451.cel"
```

Looking at the cel file names, there is a common structure. We want the short string that is prefixed by "h133a_" and suffixed by ".cel".

```
> temp1 <- unlist(strsplit(celFiles, ".cel"))
> temp1[1:3]

[1] "0074_1772_h133a_872"  "0074_1773_h133a_922"  "0074_1774_h133a_1451"

> temp2 <- unlist(lapply(strsplit(temp1, "h133a_"), function(x) {
+     x[2]
+ }))
> temp2[1:3]

[1] "872"  "922"  "1451"

> celShortNames <- temp2
> rm("temp1", "temp2")
> names(celShortNames) <- celFiles
> names(celFiles) <- celShortNames
> celFiles[1:3]

                       872                        922
 "0074_1772_h133a_872.cel"  "0074_1773_h133a_922.cel"
                      1451
"0074_1774_h133a_1451.cel"

> celShortNames[1:3]

 0074_1772_h133a_872.cel  0074_1773_h133a_922.cel 0074_1774_h133a_1451.cel
                   "872"                    "922"                   "1451"
```

The names shown here appear to make sense. Before we use them, let's make sure that they agree with what we get from the quantification table.

```
> sum(!is.na(match(celShortNames, colnames(ovcaRMAFromXLS))))

[1] 117
```

```
> celShortNames[is.na(match(celShortNames, colnames(ovcaRMAFromXLS)))]

 0074_1827_h133a_.08.cel 0074_2484_h133a_3250.cel
                   ".08"                   "3250"

> colnames(ovcaRMAFromXLS)[is.na(match(colnames(ovcaRMAFromXLS),
+     celShortNames))]

[1] "0.08" "3249"
```

All but two of the names match. In the case of “0.08”, I suspect that the leading 0 was added by Excel at some point. In the case of 3249 vs 3250, we will assume for now that these refer to the same sample and there was simply a typo. In both of these cases, we fix things by changing the values of celShortNames to match those from ovcaRMAFromXLS.

```
> celShortNames["0074_1827_h133a_.08.cel"] <- "0.08"
> celShortNames["0074_2484_h133a_3250.cel"] <- "3249"
> sum(!is.na(match(celShortNames, colnames(ovcaRMAFromXLS))))

[1] 119

> names(celFiles)[celFiles == "0074_1827_h133a_.08.cel"] <- "0.08"
> names(celFiles)[celFiles == "0074_2484_h133a_3250.cel"] <- "3249"
> sum(!is.na(match(names(celFiles), colnames(ovcaRMAFromXLS))))

[1] 119
```

Now the names line up.

Next, we the CEL files supplied using RMA.

```
> rda <- "ovcaRMAFromCELEset"
> 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 {
+     ovcaRMAFromCELEset <- justRMA(celfile.path = file.path("DukeWebSite",
+         "PlatinumJCO"))
+     save(ovcaRMAFromCELEset, file = rdaFile)
+ }

loading ovcaRMAFromCELEset from cache
```

For this analysis, we're willing to work with the matrix of expression values rather than the full ExpressionSet to allow for greater parallelism with ovcaRMAFromXLS. Let's extract this, and adjust the names.

```
> ovcaRMAFromCEL <- exprs(ovcaRMAFromCELEset)
> ovcaRMAFromCEL[1:3, 1:3]
```

```
          0074_1772_h133a_872.cel 0074_1773_h133a_922.cel
1007_s_at               10.637310               10.730829
1053_at                  6.614612                6.452423
117_at                   6.774868                7.264226
          0074_1774_h133a_1451.cel
1007_s_at                10.869773
1053_at                   6.323276
117_at                    7.157966
```

```
> colnames(ovcaRMAFromCEL) <- celShortNames[colnames(ovcaRMAFromCEL)]
> ovcaRMAFromCEL[1:3, 1:3]
```

```
                872       922      1451
1007_s_at 10.637310 10.730829 10.869773
1053_at    6.614612  6.452423  6.323276
117_at     6.774868  7.264226  7.157966
```

5 Comparing Quantifications

Our first question here has to do with identifying the probesets that are “missing” from ovcaRMAFromXLS.

```
> omittedProbesets <- setdiff(rownames(ovcaRMAFromCEL), rownames(ovcaRMAFromXLS))
> length(omittedProbesets)
```

```
[1] 168
```

```
> affyControls <- grep("^AFFX", omittedProbesets)
> length(affyControls)
```

```
[1] 68
```

```
> omittedProbesets[-affyControls]
```

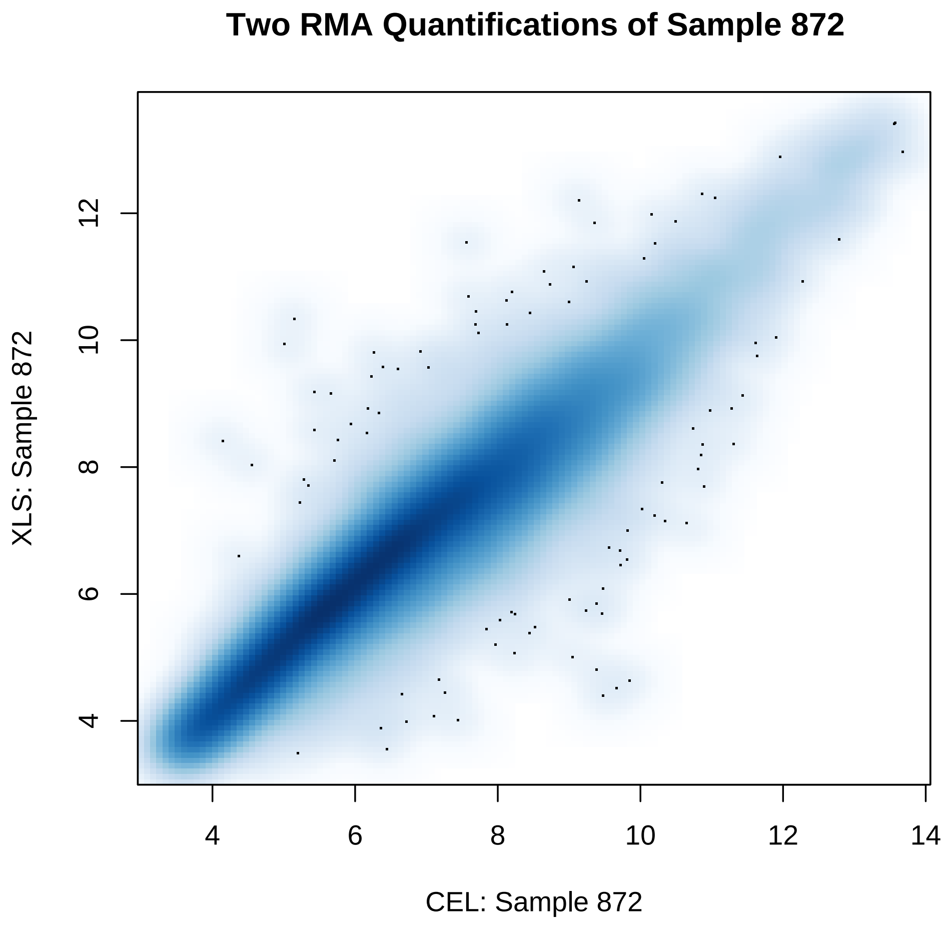
```
  [1] "200000_s_at" "200001_at"   "200002_at"   "200003_s_at" "200004_at"
  [6] "200005_at"   "200006_at"   "200007_at"   "200008_s_at" "200009_at"
 [11] "200010_at"   "200011_s_at" "200012_x_at" "200013_at"   "200014_s_at"
 [16] "200015_s_at" "200016_x_at" "200017_at"   "200018_at"   "200019_s_at"
 [21] "200020_at"   "200021_at"   "200022_at"   "200023_s_at" "200024_at"
 [26] "200025_s_at" "200026_at"   "200027_at"   "200028_s_at" "200029_at"
 [31] "200030_s_at" "200031_s_at" "200032_s_at" "200033_at"   "200034_s_at"
 [36] "200035_at"   "200036_s_at" "200037_s_at" "200038_s_at" "200039_s_at"
 [41] "200040_at"   "200041_s_at" "200042_at"   "200043_at"   "200044_at"
 [46] "200045_at"   "200046_at"   "200047_s_at" "200048_s_at" "200049_at"
 [51] "200050_at"   "200051_at"   "200052_s_at" "200053_at"   "200054_at"
 [56] "200055_at"   "200056_s_at" "200057_s_at" "200058_s_at" "200059_s_at"
 [61] "200060_s_at" "200061_s_at" "200062_s_at" "200063_s_at" "200064_at"
 [66] "200065_s_at" "200066_at"   "200067_x_at" "200068_s_at" "200069_at"
 [71] "200070_at"   "200071_at"   "200072_s_at" "200073_s_at" "200074_s_at"
 [76] "200075_s_at" "200076_s_at" "200077_s_at" "200078_s_at" "200079_s_at"
```

```
[81] "200080_s_at" "200081_s_at" "200082_s_at" "200083_at"   "200084_at"
[86] "200085_s_at" "200086_s_at" "200087_s_at" "200088_x_at" "200089_s_at"
[91] "200090_at"   "200091_s_at" "200092_s_at" "200093_s_at" "200094_s_at"
[96] "200095_x_at" "200096_s_at" "200097_s_at" "200098_s_at" "200099_s_at"
```

As expected, the 68 Affymetrix controls are among those dropped. The numerical prefixes for the other 100 run sequentially from 200000 to 200099, so they do form a contiguous block. We have no idea why these were omitted.

Our next question has to do with how well the two sets of numerical values agree for the probesets that remain. Let's take a look at this agreement the first sample in ovcaRMAFromCEL, sample 872.

```
> smoothScatter(ovcaRMAFromCEL[rownames(ovcaRMAFromXLS), "872"],
+     ovcaRMAFromXLS[, "872"], xlab = "CEL: Sample 872", ylab = "XLS: Sample 872",
+     main = "Two RMA Quantifications of Sample 872")
```

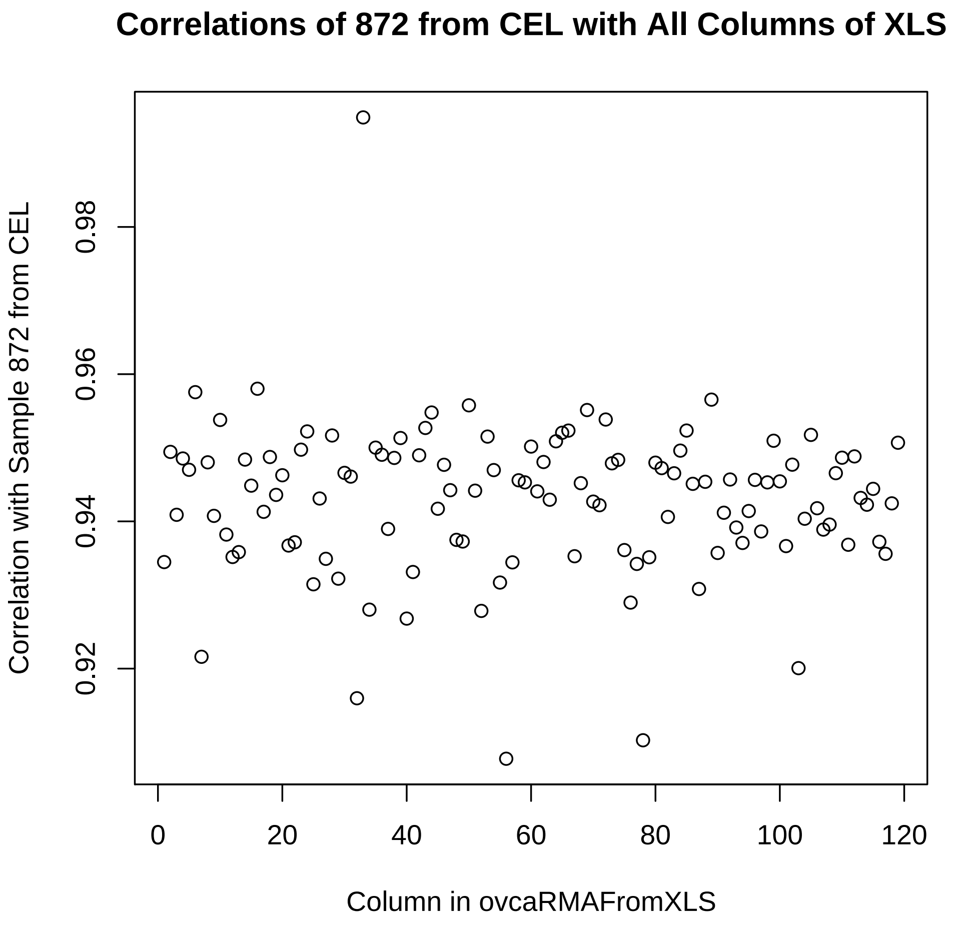


Actually, the agreement is not as good as we would have expected *a priori*. We do not necessarily expect the values to coincide perfectly (including other samples in the RMA modeling might tweak the normalization). But we wouldn't expect expression levels for a given gene to change by a factor or 20 or more either, and there presence of points with x-values near 10 and y-values near 5 implies precisely this given the log2 nature of RMA values.

Let's take a look at the correlations between the results for sample 872 from ovcaRMAFromCEL and all of the samples in ovcaRMAFromXLS, to see if there is simply poor correlation throughout.

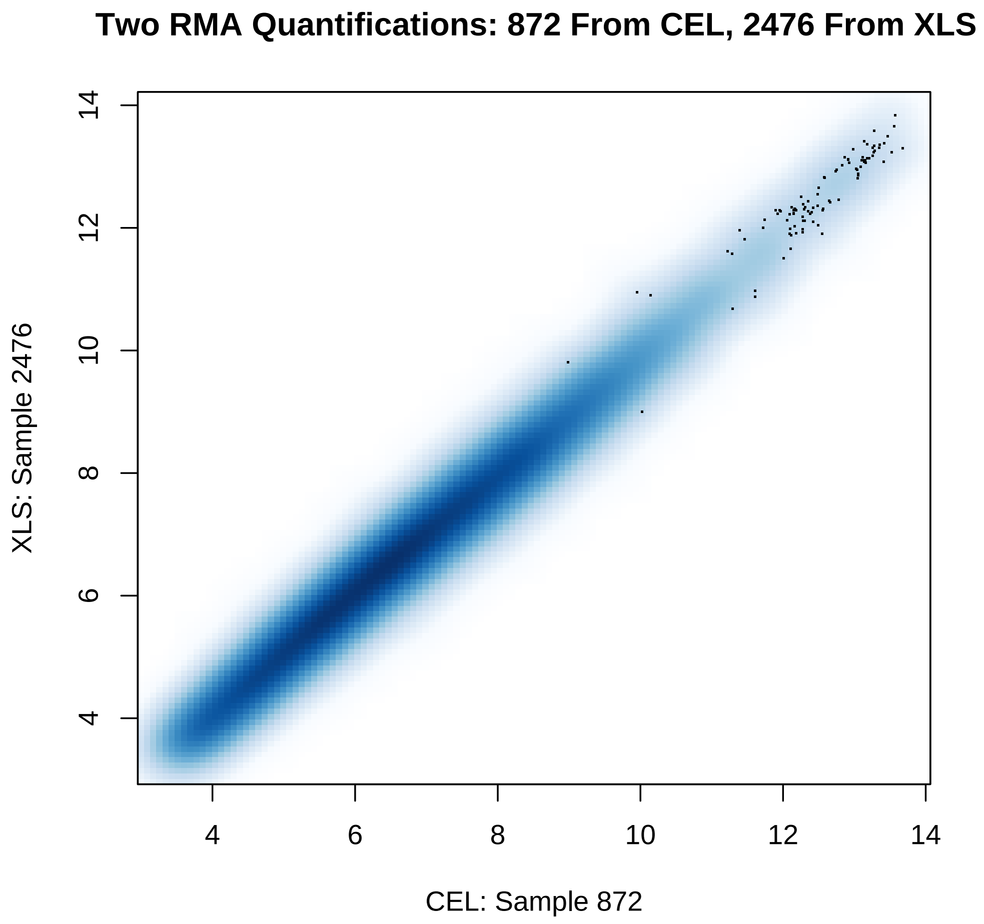
```
> corWith872 <- cor(ovcaRMAFromCEL[rownames(ovcaRMAFromXLS), "872"],
+     ovcaRMAFromXLS)
> plot(t(corWith872), xlab = "Column in ovcaRMAFromXLS", ylab = "Correlation with Sample 872 from CEL",
+     main = "Correlations of 872 from CEL with All Columns of XLS")
> colnames(corWith872)[which.max(corWith872)]

[1] "2476"
```



Actually, there is one sample that is clearly the best match. However, in ovcaRMAFromXLS, this column is identified as coming from sample 2476. Let's plot these two quantifications against each other.

```
> smoothScatter(ovcaRMAFromCEL[rownames(ovcaRMAFromXLS), "872"],
+     ovcaRMAFromXLS[, "2476"], xlab = "CEL: Sample 872", ylab = "XLS: Sample 2476",
+     main = "Two RMA Quantifications: 872 From CEL, 2476 From XLS")
```

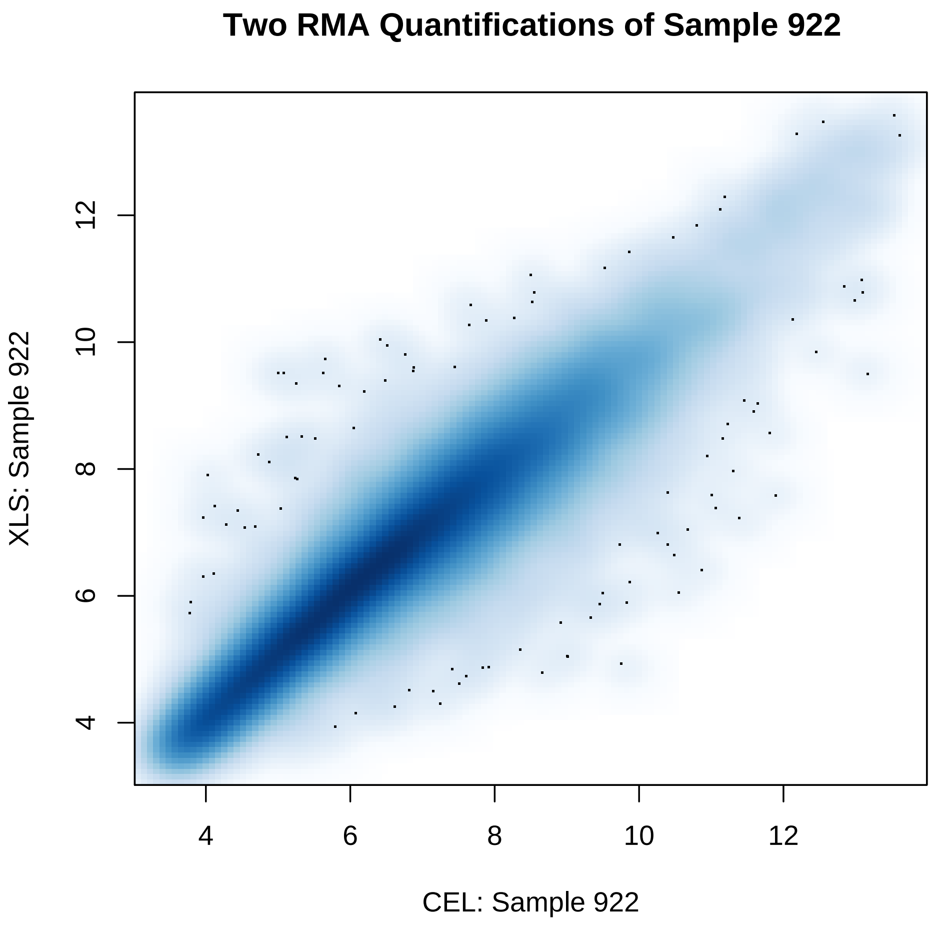


This is the type of agreement that we would expect to see between two quantifications of the same file with minor modifications in processing.

The mismatch that we see here suggests that the the results for sample 872 are mislabeled in ovcaRMAFromXLS. If this is the case, and these were indeed the quantifications used to derive clinical conclusions, those conclusions may be mistaken.

Let's look at the next sample (sample 922) as a quick check to see whether this mixup is a fluke.

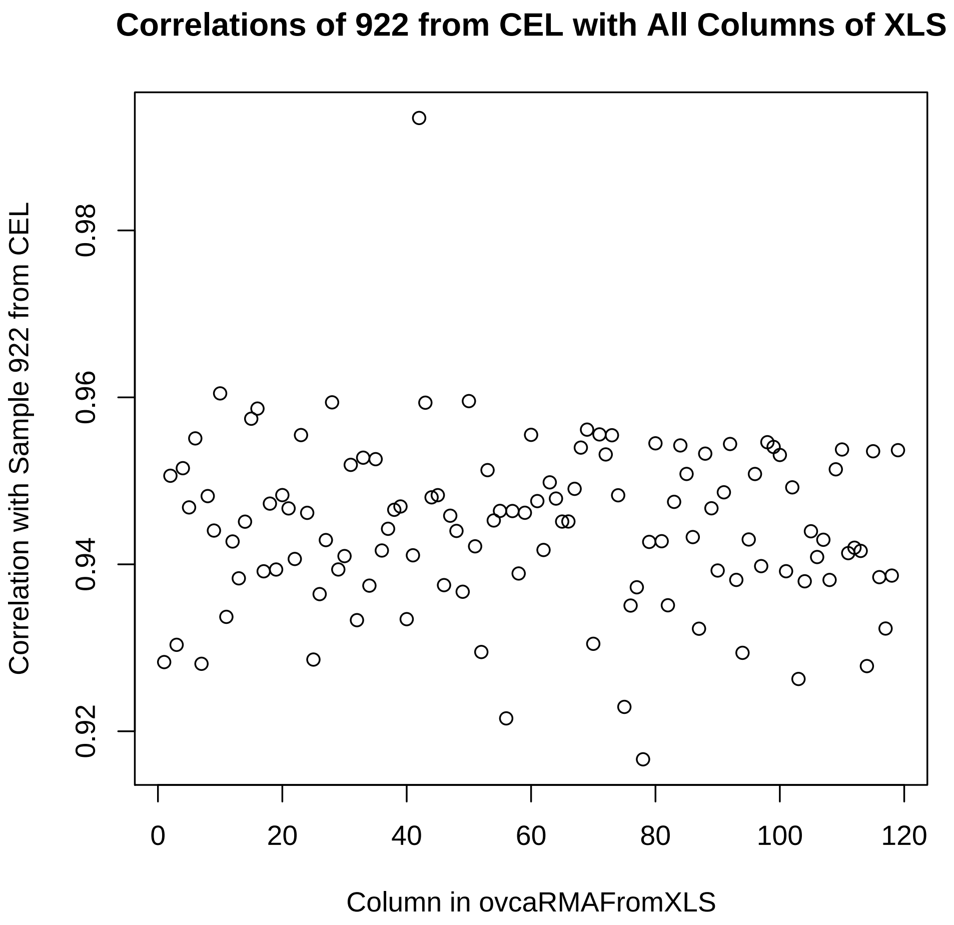
```
> smoothScatter(ovcaRMAFromCEL[rownames(ovcaRMAFromXLS), "922"],
+     ovcaRMAFromXLS[, "922"], xlab = "CEL: Sample 922", ylab = "XLS: Sample 922",
+     main = "Two RMA Quantifications of Sample 922")
```



Again, the fit is poor when the names match.

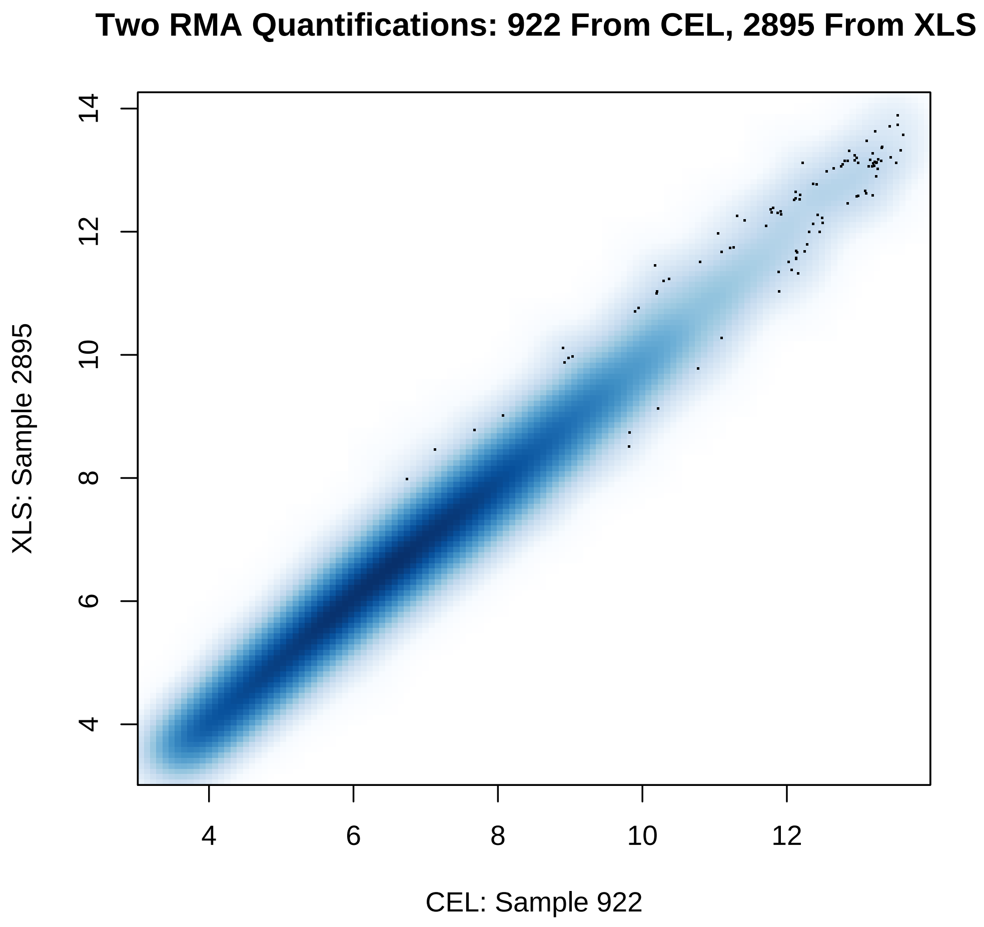
```
> corWith922 <- cor(ovcaRMAFromCEL[rownames(ovcaRMAFromXLS), "922"],
+     ovcaRMAFromXLS)
> plot(t(corWith922), xlab = "Column in ovcaRMAFromXLS", ylab = "Correlation with Sample 922 from CEL",
+     main = "Correlations of 922 from CEL with All Columns of XLS")
> colnames(corWith922)[which.max(corWith922)]

[1] "2895"
```



Again, there is a clear best match; in this case it is with 2895.

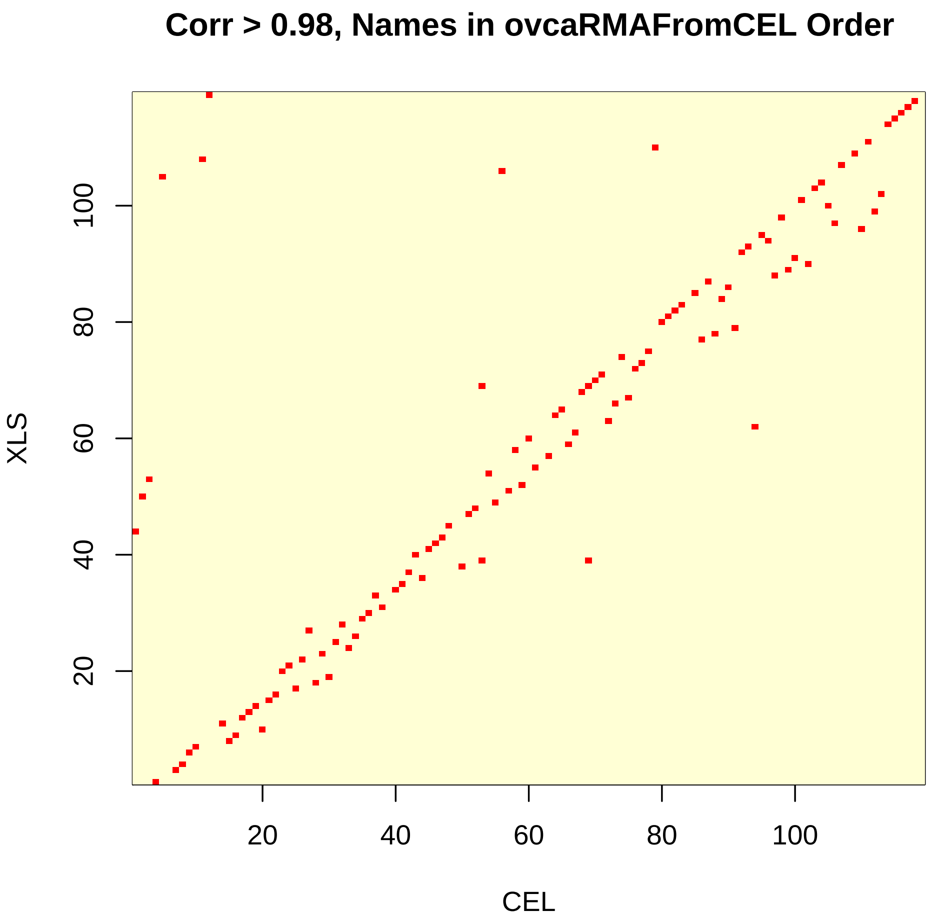
```
> smoothScatter(ovcaRMAFromCEL[rownames(ovcaRMAFromXLS), "922"],
+     ovcaRMAFromXLS[, "2895"], xlab = "CEL: Sample 922", ylab = "XLS: Sample 2895",
+     main = "Two RMA Quantifications: 922 From CEL, 2895 From XLS")
```



Again, the fit with the best match looks just like what we might expect from two quantifications of the same file. The mislabeling does not appear to have been a fluke.

At this point, we need to know just how extensive the problem is. Let's take a look at all of the correlations.

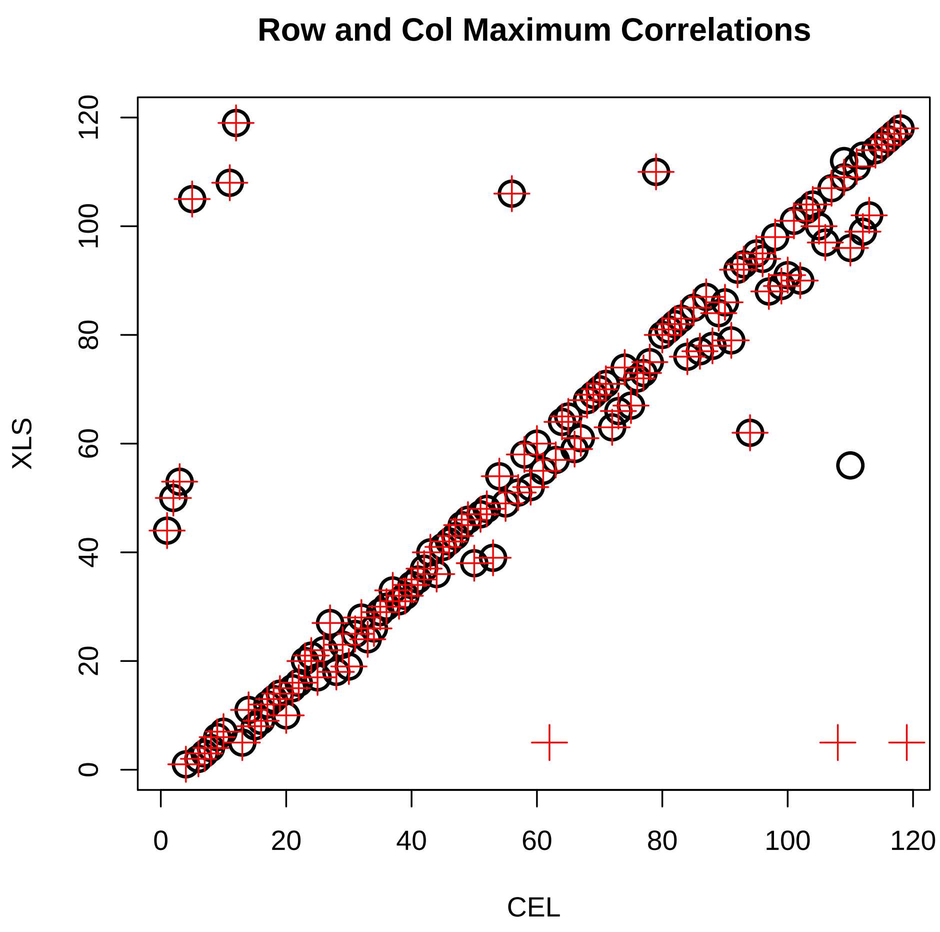
```
> corCELWithXLS <- cor(ovcaRMAFromCEL[rownames(ovcaRMAFromXLS),
+     ], ovcaRMAFromXLS[, colnames(ovcaRMAFromCEL)])
> image(1:119, 1:119, corCELWithXLS < 0.98, xlab = "CEL", ylab = "XLS",
+     main = "Corr > 0.98, Names in ovcaRMAFromCEL Order")
```



Looking at where the high correlations are, and ordering the sample names along each axis to match that for ovcaRMAFromCEL (alphabetic ordering of the CEL file names), we see that while there are only 32 cases where we have matches, most of the high correlations are very close to the main diagonal. Further, those not on the diagonal are consistently slightly below it. This suggests some type of indexing offset, though we do not have an explanation for it.

Now we want to specify our best guess as to what the mapping should be. We do this by first searching the correlation matrix for values that are the biggest to be found in their respective rows and columns. When row and column maxima coincide, we've found a good match. Rows and columns that remain ambiguous will then be dealt with.

```
> bestXLSFitsToGivenCEL <- max.col(corCELWithXLS)
> bestCELFitsToGivenXLS <- max.col(t(corCELWithXLS))
> plot(bestCELFitsToGivenXLS, 1:119, cex = 2, lwd = 2, xlab = "CEL",
+     ylab = "XLS", main = "Row and Col Maximum Correlations")
> points(1:119, bestXLSFitsToGivenCEL, pch = 3, cex = 2, col = "red")
```



Looking at the plot, we see that the maxima coincide for 116 of the 119 samples. We first identify the CEL files which did not find their match on the first try.

```
> which(duplicated(bestXLSFitsToGivenCEL))

[1]  62 108 119

> bestXLSFitsToGivenCEL[c(62, 108, 119)]

[1] 5 5 5

> which(bestXLSFitsToGivenCEL == 5)

[1]  13  62 108 119

> bestCELFitsToGivenXLS[5]

[1] 13
```

```
> rownames(corCELWithXLS)[c(62, 108, 119)]

[1] "D2358" "M3484" "M810"
```

Next, we look for the XLS entries that don't find their best match immediately.

```
> which(duplicated(bestCELFitsToGivenXLS))

[1]  96 112 113

> bestCELFitsToGivenXLS[c(96, 112, 113)]

[1] 110 109 112

> which(bestCELFitsToGivenXLS == 110)

[1] 56 96

> bestXLSFitsToGivenCEL[110]

[1] 96

> which(bestCELFitsToGivenXLS == 109)

[1] 109 112

> bestXLSFitsToGivenCEL[109]

[1] 109

> which(bestCELFitsToGivenXLS == 112)

[1]  99 113

> bestXLSFitsToGivenCEL[112]

[1] 99

> colnames(corCELWithXLS)[c(56, 112, 113)]

[1] "D1837" "M4161" "M444"
```

Now we know which ones to be on the alert for. The XLS quantifications for D1837, M4161, and M444 do not have very good matches the set of CEL file quantifications.

6 Expanding the Mapping

In addition to the 119 ovarian CEL files supplied on the website for Dressman et al, there are 146 ovarian CEL files supplied on the website for Bild et al (Nature 2006); the latter are a superset of the former. We can also quantify this larger set.

```
> rda <- "ovcaRMAFromBildEset"
> 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 {
+     ovcaRMAFromBildEset <- justRMA(celfile.path = file.path("OtherData",
+         "BildNature06", "OvarianTumorData"))
+     save(ovcaRMAFromBildEset, file = rdaFile)
+ }
```

```
loading ovcaRMAFromBildEset from cache
```

```
> ovcaRMAFromBild <- exprs(ovcaRMAFromBildEset)
```

Given the larger set, let's check the correlations again, and tabulate the XLS file name, the best matching CEL file name, the best matching Bild file name, and the top three Bild correlation values for each XLS file. First, put together the structure.

```
> corBildWithXLS <- cor(ovcaRMAFromBild[rownames(ovcaRMAFromXLS),
+     ], ovcaRMAFromXLS)
> bildCheck <- cbind(xlsName = colnames(corCELWithXLS), celName = colnames(corCELWithXLS),
+     bildName = colnames(corCELWithXLS), bildCor1 = rep(0, 119),
+     bildCor2 = rep(0, 119), bildCor3 = rep(0, 119))
> for (i1 in 1:length(colnames(corCELWithXLS))) {
+     bildCheck[i1, "celName"] <- rownames(corCELWithXLS)[which.max(corCELWithXLS[,
+         i1])]
+     bildCheck[i1, "bildName"] <- rownames(corBildWithXLS)[which.max(corBildWithXLS[,
+         colnames(corCELWithXLS)[i1]])]
+     bildCheck[i1, 4:6] <- sort(corBildWithXLS[, colnames(corCELWithXLS)[i1]])[c(146,
+         145, 144)]
+ }
> bildCheck <- as.data.frame(bildCheck)
> bildCheck["xlsName"] <- as.character(bildCheck[, "xlsName"])
> bildCheck["celName"] <- as.character(bildCheck[, "celName"])
> bildCheck["bildName"] <- as.character(bildCheck[, "bildName"])
> bildCheck[, "bildCor1"] <- as.numeric(as.character(bildCheck[,
+     "bildCor1"]))
> bildCheck[, "bildCor2"] <- as.numeric(as.character(bildCheck[,
+     "bildCor2"]))
> bildCheck[, "bildCor3"] <- as.numeric(as.character(bildCheck[,
+     "bildCor3"]))
> bildCheck <- bildCheck[order(bildCheck[, "bildName"]), ]
> rownames(bildCheck) <- 1:119
```

Next, take a look at the mappings.

```
> bildCheck
```

```
   xlsName celName                   bildName  bildCor1  bildCor2  bildCor3
1    M2807    1784  0074_01776_h133a_1784.cel 0.9926820 0.9566297 0.9557930
2    M3484    0.08   0074_01827_h133a_.08.cel 0.9881092 0.9473432 0.9470600
3     M810     860   0074_01828_h133a_860.cel 0.9914995 0.9688161 0.9670850
4     1784    1615  0074_01829_h133a_1615.cel 0.9748212 0.9710153 0.9683388
5     0.08    1665  0074_01830_h133a_1665.cel 0.9941898 0.9467627 0.9463406
6      860    2465  0074_01834_h133a_2465.cel 0.9845509 0.9440225 0.9440143
7     1615    2999  0074_01835_h133a_2999.cel 0.9964939 0.9700833 0.9692109
8     1665    3142  0074_01836_h133a_3142.cel 0.9808840 0.9485003 0.9483186
9     2465    1774  0074_01906_h133a_1774.cel 0.9854140 0.9453324 0.9431743
10    2064    2064  0074_01908_h133a_2064.cel 0.9856633 0.9661539 0.9608221
11    2999    2967  0074_01909_h133a_2967.cel 0.9920582 0.9605766 0.9577776
12    3142    2573  0074_02003_h133a_2573.cel 0.9874445 0.9712619 0.9697764
13    1774    2849  0074_02004_h133a_2849.cel 0.9984232 0.9697679 0.9682319
14    2967    3102  0074_02005_h133a_3102.cel 0.9950447 0.9718080 0.9697266
15    2573    2802  0074_02026_h133a_2802.cel 0.9945379 0.9604171 0.9591595
16    2849    2424  0074_02028_h133a_2424.cel 0.9955498 0.9546375 0.9537399
17    3102    2063  0074_02029_h133a_2063.cel 0.9724740 0.9554820 0.9542840
18    2802    2476  0074_02394_h133a_2476.cel 0.9944180 0.9508070 0.9506363
19    2424    2895  0074_02400_h133a_2895.cel 0.9900188 0.9626413 0.9593705
20    2063    2981  0074_02403_h133a_2981.cel 0.9957414 0.9824611 0.9637875
21    3249    3249  0074_02484_h133a_3250.cel 0.9877743 0.9562246 0.9538962
22    2476     872    0074_1772_h133a_872.cel 0.9943223 0.9636400 0.9618316
23    2895     922    0074_1773_h133a_922.cel 0.9928437 0.9604397 0.9597264
24    2981    1451   0074_1774_h133a_1451.cel 0.9937170 0.9632513 0.9623755
25     872    1526   0074_1775_h133a_1526.cel 0.9816869 0.9597752 0.9585888
26     922    1834   0074_1777_h133a_1834.cel 0.9695620 0.9544054 0.9536421
27    1451    1846   0074_1778_h133a_1846.cel 0.9911025 0.9606119 0.9592577
28    1526    2075   0074_1779_h133a_2075.cel 0.9966748 0.9640004 0.9612208
29    1834    2204   0074_1780_h133a_2204.cel 0.9912310 0.9568317 0.9556369
30    1846    2419   0074_1781_h133a_2419.cel 0.9927894 0.9589094 0.9565815
31    2075    1675   0074_1831_h133a_1675.cel 0.9951890 0.9503800 0.9499588
32    2204    2422   0074_1833_h133a_2422.cel 0.9970365 0.9596536 0.9573983
33    2419    1504   0074_1900_h133a_1504.cel 0.9938924 0.9506857 0.9492229
34    1675    1590   0074_1901_h133a_1590.cel 0.9882833 0.9565666 0.9562508
35    2422    1623   0074_1902_h133a_1623.cel 0.9862560 0.9636948 0.9576494
36    1504    2324   0074_1904_h133a_2324.cel 0.9876288 0.9573655 0.9571443
37    1590    1674   0074_1905_h133a_1674.cel 0.9930011 0.9558601 0.9557393
38    1623    1929   0074_1907_h133a_1929.cel 0.9817851 0.9521843 0.9487794
39    2324    2198   0074_1989_h133a_2198.cel 0.9932159 0.9702529 0.9698943
40    1674    1877   0074_2019_h133a_1877.cel 0.9946044 0.9523580 0.9464071
41    1929    2046   0074_2020_h133a_2046.cel 0.9954422 0.9627558 0.9581968
42    2198    2479   0074_2021_h133a_2479.cel 0.9971700 0.9516141 0.9511909
43    1877    2542   0074_2027_h133a_2542.cel 0.9910925 0.9514289 0.9497953
44    2046    1024   0074_2030_h133a_1024.cel 0.9866241 0.9503525 0.9498897
45    2479    2739   0074_2031_h133a_2739.cel 0.9947668 0.9546257 0.9543300
46    2542    2673   0074_2032_h133a_2673.cel 0.9961120 0.9561790 0.9543924
47    1024    2505   0074_2033_h133a_2505.cel 0.9927848 0.9602439 0.9590956
```

```
48     2739    1447   0074_2395_h133a_1447.cel 0.9928757 0.9640413 0.9618055
49     2673    1913   0074_2396_h133a_1913.cel 0.9909242 0.9587711 0.9585161
50     2505    1552   0074_2397_h133a_1552.cel 0.9941829 0.9647011 0.9634440
51     1447    1578   0074_2398_h133a_1578.cel 0.9935861 0.9578255 0.9564654
52     1913    3107   0074_2399_h133a_3107.cel 0.9791127 0.9537286 0.9535498
53     1552    3018   0074_2401_h133a_3018.cel 0.9947323 0.9535522 0.9533976
54     1578    3090   0074_2402_h133a_3090.cel 0.9909406 0.9585605 0.9561681
55     3107   D1805 0193_00000_h133a_D1805.cel 0.9872051 0.9504583 0.9493932
56     3018   D1859 0193_00000_h133a_D1859.cel 0.9912899 0.9543120 0.9512162
57    D2098   D2098 0193_00000_h133a_D2098.cel 0.9947249 0.9609262 0.9602408
58     3090   D2208 0193_00000_h133a_D2208.cel 0.9950948 0.9550372 0.9542026
59    D1805   D2342 0193_00000_h133a_D2342.cel 0.9961207 0.9733474 0.9685885
60    D1859   D2421 0193_00000_h133a_D2421.cel 0.9968884 0.9675697 0.9667681
61    D2208   D2480 0193_00000_h133a_D2480.cel 0.9875381 0.9593251 0.9579144
62    D2342   D2557 0193_00000_h133a_D2557.cel 0.9942855 0.9668900 0.9650679
63    D2421   D2576 0193_00000_h133a_D2576.cel 0.9947073 0.9600370 0.9584904
64    D2480   D2581 0193_00000_h133a_D2581.cel 0.9896129 0.9634672 0.9588543
65    D2557   D2611 0193_00000_h133a_D2611.cel 0.9949943 0.9576065 0.9545058
66    D2576   D2629 0193_00000_h133a_D2629.cel 0.9950710 0.9685574 0.9662622
67    D2581   D2640 0193_00000_h133a_D2640.cel 0.9966334 0.9677558 0.9663792
68    D2611   D2648 0193_00000_h133a_D2648.cel 0.9949376 0.9550945 0.9498305
69    D2629   D2727 0193_00000_h133a_D2727.cel 0.9700504 0.9493768 0.9464584
70    D2640   D2738 0193_00000_h133a_D2738.cel 0.9926036 0.9479970 0.9471094
71    D2648   D2776 0193_00000_h133a_D2776.cel 0.9913903 0.9350622 0.9348196
72    D2727   D2792 0193_00000_h133a_D2792.cel 0.9943549 0.9610750 0.9608598
73    D2738   M1054 0193_00000_h133a_M1054.cel 0.9924206 0.9597049 0.9542825
74    D2358   M1390 0193_00000_h133a_M1390.cel 0.9967037 0.9613274 0.9609212
75    M1390   M1572 0193_00000_h133a_M1572.cel 0.9956644 0.9582878 0.9567982
76    D2776     M17   0193_00000_h133a_M17.cel 0.9921536 0.9613312 0.9593830
77    D2792   M2070 0193_00000_h133a_M2070.cel 0.9968531 0.9605534 0.9588873
78    M1054   M2437 0193_00000_h133a_M2437.cel 0.9953592 0.9529479 0.9525583
79      M17   M3142 0193_00000_h133a_M3142.cel 0.9967593 0.9620009 0.9569580
80    M1572    M359  0193_00000_h133a_M359.cel 0.9893443 0.9609522 0.9608295
81    M2070   M4161 0193_00000_h133a_M4161.cel 0.9972001 0.9671245 0.9645470
82    M2437    M444  0193_00000_h133a_M444.cel 0.9968686 0.9686836 0.9627136
83    M3142   D1837 0193_10000_h133a_D1837.cel 0.9956409 0.9596538 0.9586865
84    M4161   M3514 0193_10000_h133a_D2159.cel 0.9940312 0.9592796 0.9568668
85     M444   M4161 0193_10000_h133a_D2171.cel 0.9933361 0.9610775 0.9577707
86    D1837    M359 0193_10000_h133a_D2247.cel 0.9886348 0.9316258 0.9291977
87    D2332   D2332 0193_10000_h133a_D2332.cel 0.9959141 0.9666855 0.9659541
88    D2432   D2432 0193_10000_h133a_D2432.cel 0.9963716 0.9651230 0.9650317
89    D2433   D2433 0193_10000_h133a_D2433.cel 0.9946305 0.9614134 0.9579562
90    D2559   D2559 0193_10000_h133a_D2559.cel 0.9937811 0.9663077 0.9627052
91    D2560   D2560 0193_10000_h133a_D2560.cel 0.9964361 0.9813157 0.9607497
92    D2572   D2572 0193_10000_h133a_D2572.cel 0.9963744 0.9590666 0.9502720
93    D2575   D2575 0193_10000_h133a_D2575.cel 0.9915036 0.9744510 0.9682305
94    D2603   D2603 0193_10000_h133a_D2603.cel 0.9860988 0.9634429 0.9630732
95     M359   D2668 0193_10000_h133a_D2668.cel 0.9961116 0.9667837 0.9667216
```

```
96    D2689   D2689 0193_10000_h133a_D2689.cel 0.9954061 0.9684339 0.9682928
97    D2691   D2691 0193_10000_h133a_D2691.cel 0.9956504 0.9493081 0.9482940
98    D2700   D2700 0193_10000_h133a_D2700.cel 0.9895614 0.9479288 0.9479090
99    D2726   D2726 0193_10000_h133a_D2726.cel 0.9925030 0.9603756 0.9578855
100   D2733   D2733 0193_10000_h133a_D2733.cel 0.9966983 0.9704670 0.9690580
101   D2749   D2749 0193_10000_h133a_D2749.cel 0.9956807 0.9436386 0.9430509
102   D2668   M1055 0193_10000_h133a_M1055.cel 0.9941089 0.9646771 0.9621903
103    M120    M120  0193_10000_h133a_M120.cel 0.9952542 0.9647887 0.9633458
104   M1241   M1241 0193_10000_h133a_M1241.cel 0.9969275 0.9579454 0.9572306
105   M1503   M1503 0193_10000_h133a_M1503.cel 0.9943912 0.9601343 0.9594766
106   M1891   M1891 0193_10000_h133a_M1891.cel 0.9911004 0.9678596 0.9637527
107   M1055   M2097 0193_10000_h133a_M2097.cel 0.9958288 0.9642369 0.9634825
108   M2184   M2184 0193_10000_h133a_M2184.cel 0.9945751 0.9579148 0.9565246
109   M2515   M2515 0193_10000_h133a_M2515.cel 0.9947837 0.9513034 0.9436120
110   M2729   M2729 0193_10000_h133a_M2729.cel 0.9904464 0.9563503 0.9558432
111   M2097   M2807 0193_10000_h133a_M2807.cel 0.9958400 0.9649520 0.9638534
112    M337    M337  0193_10000_h133a_M337.cel 0.9897121 0.9559832 0.9525690
113   M3514   M3514 0193_10000_h133a_M3514.cel 0.9965436 0.9686661 0.9665613
114   M3627   M3627 0193_10000_h133a_M3627.cel 0.9967617 0.9568071 0.9550400
115    M485    M485  0193_10000_h133a_M485.cel 0.9940410 0.9580871 0.9533050
116    M503    M503  0193_10000_h133a_M503.cel 0.9925675 0.9605806 0.9595132
117   M5668   M5668 0193_10000_h133a_M5668.cel 0.9939380 0.9604345 0.9579929
118   M5775   M5775 0193_10000_h133a_M5775.cel 0.9959831 0.9537094 0.9528543
119   M6199   M6199 0193_10000_h133a_M6199.cel 0.9945113 0.9598195 0.9564696
```

Looking at the mappings above, the results are mostly consistent with what we found before, in that the CEL mappings match the Bild mappings. However, there are three discrepancies:

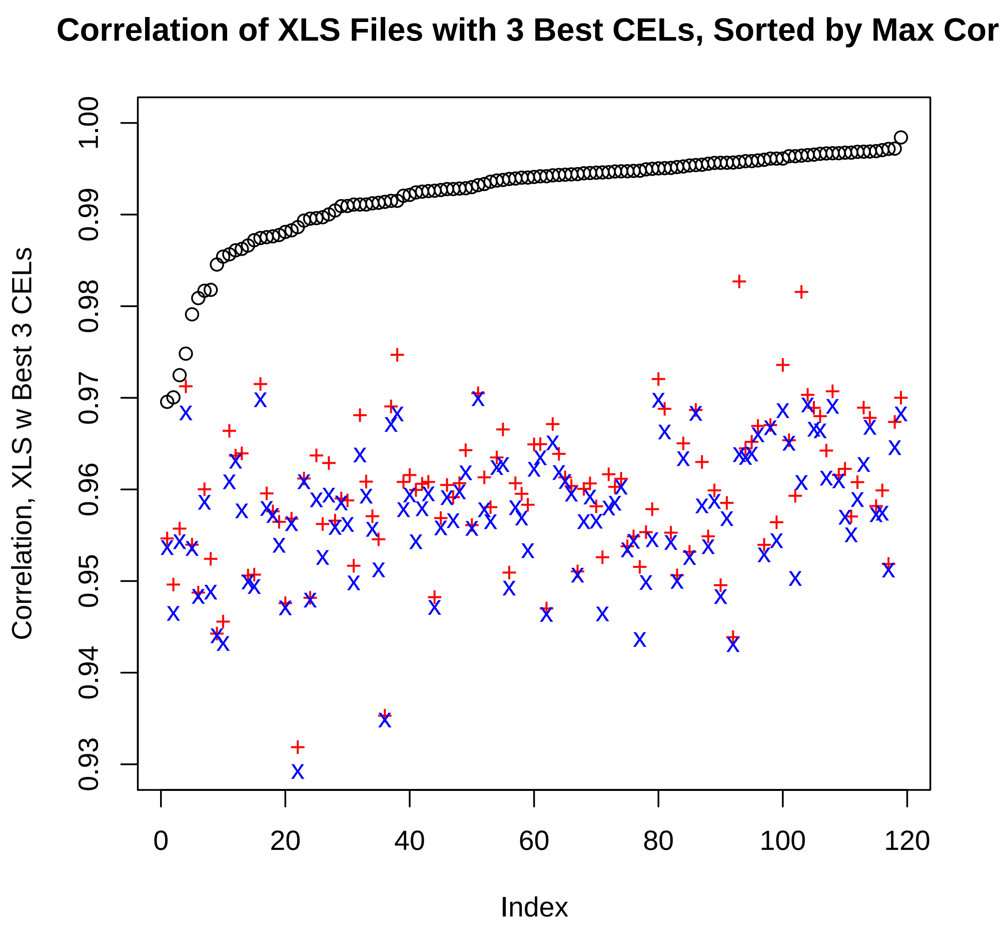
- Row 84, XLS M4161, CEL M3514, Bild D2159
- Row 85, XLS M444, CEL M4161, Bild D2171
- Row 86, XLS D1837, CEL M359, Bild D2247

These are the three ambiguous cases noted in the section above. Looking at these three rows, it is clear that the new best fit is much better in each instance, as the correlations are now 0.994, 0.993, and 0.989, respectively, with the next best being 0.961 or less. For these three entries in the XLS quantifications, the source files are not in the PlatinumJCO files. Conversely, the three samples D2358, M3484, and M810 are named in the XLS file, but their quantifications are not present.

As noted above, the type of mismatch seen using the names of the CEL files to order things suggests a systematic offset, in most cases of 3 rows.

Let's check how good the fits are at this point.

```
> plot(bildCheck[order(bildCheck[, 4]), 4], ylim = c(0.93, 1),
+     xlab = "Index", ylab = "Correlation, XLS w Best 3 CELs",
+     main = "Correlation of XLS Files with 3 Best CELs, Sorted by Max Corr")
> points(bildCheck[order(bildCheck[, 4]), 5], ylim = c(0.93, 1),
+     col = "red", pch = "+")
> points(bildCheck[order(bildCheck[, 4]), 6], ylim = c(0.93, 1),
+     col = "blue", pch = "x")
```



Here, we've plotted the three best correlations for each XLS quantification, sorted so that the maximum correlations are monotonically increasing. What we see is that there is a very large gap between the best correlations and the others – there is a very "clear winner" for almost all of the samples. The least clear case corresponds to XLS 1784, CEL 1615, which has the third lowest maximum correlation overall. Here, however, we have some additional consistency in that the match chosen fits with the systematic name offset noted above.

7 Summary

1. One hundred probe sets (with consecutive probe set IDs starting with 200000_s_at) were omitted from the reported Excel RMA quantifications.

2. Two of the CEL file names do not match names in the Excel spreadsheet.

3. More importantly, based on the correlation coefficients, the sample names appear to have been scrambled between the CEL files and the Excel spreadsheet. Only 32 out of 119 samples appear to have the

correct names in the Excel spreadsheet; most of the problems appear to arise from an undetermined indexing error.

4. For 116 out of 119 samples, we can fairly reliably reconstruct the correct mapping of names. The other three samples do not have obvious matches.

5. The 119 CEL files that are part of the study by Dressman form a subset of the 146 CEL that are part of the study by Bild. The three anomolous columns on the Excel spreadsheet give better matches to three of the files that are in the Bild set but not tyhe Dressman set.

8 Appendix

8.1 Saves

There are a few objects we have constructed here that we would like to keep around.

```
> save(celFiles, file = paste("RDataObjects", "celFiles.Rda", sep = .Platform$file.sep))
> save(celShortNames, file = paste("RDataObjects", "celShortNames.Rda",
+     sep = .Platform$file.sep))
> save(ovcaRMAFromCEL, file = paste("RDataObjects", "ovcaRMAFromCEL.Rda",
+     sep = .Platform$file.sep))
> save(corCELWithXLS, file = paste("RDataObjects", "corCELWithXLS.Rda",
+     sep = .Platform$file.sep))
> save(ovcaRMAFromBild, file = paste("RDataObjects", "ovcaRMAFromBild.Rda",
+     sep = .Platform$file.sep))
> save(corBildWithXLS, file = paste("RDataObjects", "corBildWithXLS.Rda",
+     sep = .Platform$file.sep))
> save(bildCheck, file = paste("RDataObjects", "bildCheck.Rda",
+     sep = .Platform$file.sep))
```

8.2 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:
       survival  ClassDiscovery         cluster ClassComparison      PreProcess
         "2.32"         "2.5.0"        "1.11.7"         "2.5.0"         "2.5.0"
      oompaBase     geneplotter         lattice        annotate            affy
        "2.5.0"        "1.14.0"       "0.15-11"        "1.14.1"        "1.14.2"
```

```
   affyio      Biobase
  "1.4.1"     "1.14.1"
```