Processing of the Chang data

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

16 March 2007

1 Description of the problem

We need to load the Chang data, but we have two choices. We can start with the CEL files and quantify them ourselves (in dChip), or we can use the already processed data in the GEO SOFT files. Of course, we choose to do both.

2 Load the data

We start with the individual data from Novartis and the data on 10 drugs from the DTP. We only really need the Novartis data here, since we use it to make certain that the gene-rows are in the same order in all the data sets.

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

Writing to file Tangled/prepareData.R

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

Use this function to load cached data, if it exists, and produce it from scratch otherwise.

```
> getCached <- function(rda, r) {
+     rfile <- file.path("Tangled", paste(r, "R", sep = "."))
+     rdafile <- file.path("RDataObjects", paste(rda, "Rda",
+         sep = "."))
+     if (file.exists(rdafile)) {
+         cat("loading from cache\n")
+         load(rdafile, .GlobalEnv)
+     }
+     else {
```

```
+         Stangle(file.path("RNowebSource", paste(r, "Rnw",
+             sep = ".")), output = rfile)
+         source(rfile)
+     }
+ }
```

We also need the normalized and log-transformed data (`novartisNL`), which is stored in the `features.Rda` binary file.

```
> getCached("features", "wobblingFeatures")

loading from cache
```

3 Loading SOFT files

Here, we make use of the processed Affymetrix data available from GEO:

- GSE349, 14 samples resistant to docetaxel, and
- GSE350, 10 samples sensitive to docetaxel.

From personal communications with Sue Hilsenbeck, we know that one of the samples was mislabeled: "Sample #377 is mislabeled in the GEO DB. It is listed there as resistant, but in reality it was sensitive." Checking the GEO annotation, sample 377 corresponds to GSM4913.

Array quantifications were supplied in GEO in addition to the CEL files (which were listed as supplementary). Grabbing the SOFT file for GSE349, we have quantifications for all of the arrays from resistant samples (non-responders). Based on the description in the Chang paper in the *Lancet*, we believe these data were produced using the $PM - MM$ model in dChip. Lines beginning with "Sample_description" indicate "Antibody amplified expression data, normalized and modeled".

The individual samples give rise to distinct sample quantifications located sequentially within the SOFT file. Each sample entry begins with a line starting with

```
^SAMPLE =
```

and includes descriptor info lines beginning with an exclamation point. The actual numerical quantifications are prefixed here with

```
!Sample_series_id = GSE349
!Sample_data_row_count = 12625
#ID_REF =
#VALUE =
!sample_table_begin
ID_REF  VALUE
AFFX-MurIL2_at  54.1184
```

and ending with

```
!sample_table_end
```

Invoking

```
grep -n SAMPLE GSE349_family.soft
grep -n sample_table_end GSE349_family.soft
```

on a UNIX machine, we find that the first sample record starts at line number 12752, then next at line number 25409, and so on. Each record here involves 12657 rows. (The corresponding operation on a Windows machine uses "find /N" instead of "grep -n".) For now, we want to extract the quantifications and the associated sample GEO accession numbers.

Grab the individual sample names (here we will use sample GEO accession numbers).

```
> softDir <- file.path("..", "PublicData", "Chang", "Soft")
> if (R.version$os == "mingw32") {
+     temp <- system(paste("find /N \"Sample_geo_accession\" ",
+         file.path(softDir, "GSE349_family.soft")), intern = TRUE)
+     temp <- strsplit(temp, " = ")
+     temp <- unlist(temp)
+     resistantNames <- temp[seq(from = 3, to = length(temp),
+         by = 2)]
+     temp <- system(paste("find /N \"sample_table_begin\" ",
+         file.path(softDir, "GSE349_family.soft")), intern = TRUE)
+     temp <- sub("\\[", "", temp)
+     temp <- strsplit(temp, "]")
+     temp <- unlist(temp)
+     startRows <- as.numeric(temp[seq(from = 2, to = length(temp),
+         by = 2)])
+     rm(temp)
+ } else {
+     temp <- system(paste("grep -n Sample_geo_accession ",
+         file.path(softDir, "GSE349_family.soft")), intern = TRUE)
+     temp <- strsplit(temp, " = ")
+     temp <- unlist(temp)
+     resistantNames <- temp[seq(from = 2, to = length(temp),
+         by = 2)]
+     temp <- system(paste("grep -n sample_table_begin ",
+         file.path(softDir, "GSE349_family.soft")), intern = TRUE)
+     temp <- strsplit(temp, ":")
+     temp <- unlist(temp)
+     startRows <- as.numeric(temp[seq(from = 1, to = length(temp),
+         by = 2)])
+     rm(temp)
+ }
```

Now that we know where we want to look, we go in and extract the quantifications one file at a time. It is not clear if those submitting the files are required to sort the probeset ids consistently, so we will not assume this.

```
> nProbesets <- 12625
> resistantData <- data.frame(matrix(0, nProbesets, length(resistantNames)))
> colnames(resistantData) <- resistantNames
> temp <- read.table(file.path(softDir, "GSE349_family.soft"),
+     nrows = nProbesets, skip = startRows[1], header = TRUE,
+     row.names = "ID_REF")
> rownames(resistantData) <- rownames(temp)
> resistantData[, 1] <- temp$VALUE
> for (i1 in 2:length(startRows)) {
+     temp <- read.table(file.path(softDir, "GSE349_family.soft"),
+         nrows = nProbesets, skip = startRows[i1], header = TRUE,
+         row.names = "ID_REF")
+     resistantData[, i1] <- temp$VALUE[match(rownames(resistantData),
+         rownames(temp))]
+ }
```

We repeat these steps essentially verbatim for the sensitive samples contained in GSE350.

```
> if (R.version$os == "mingw32") {
+     temp <- system(paste("find /N \"Sample_geo_accession\" ",
+         file.path(softDir, "GSE350_family.soft")), intern = TRUE)
+     temp <- strsplit(temp, " = ")
+     temp <- unlist(temp)
+     sensitiveNames <- temp[seq(from = 3, to = length(temp),
+         by = 2)]
+     temp <- system(paste("find /N \"sample_table_begin\" ",
+         file.path(softDir, "GSE350_family.soft")), intern = TRUE)
+     temp <- sub("\\[", "", temp)
+     temp <- strsplit(temp, "]")
+     temp <- unlist(temp)
+     startRows <- as.numeric(temp[seq(from = 2, to = length(temp),
+         by = 2)])
+     rm(temp)
+ } else {
+     temp <- system(paste("grep -n Sample_geo_accession ",
+         file.path(softDir, "GSE350_family.soft")), intern = TRUE)
+     temp <- strsplit(temp, " = ")
+     temp <- unlist(temp)
+     sensitiveNames <- temp[seq(from = 2, to = length(temp),
+         by = 2)]
+     temp <- system(paste("grep -n sample_table_begin ",
+         file.path(softDir, "GSE350_family.soft")), intern = TRUE)
+     temp <- strsplit(temp, ":")
+     temp <- unlist(temp)
+     startRows <- as.numeric(temp[seq(from = 1, to = length(temp),
+         by = 2)])
+     rm(temp)
+ }
```

```
> sensitiveData <- data.frame(matrix(0, nProbesets, length(sensitiveNames)))
> colnames(sensitiveData) <- sensitiveNames
> temp <- read.table(file.path(softDir, "GSE350_family.soft"),
+     nrows = nProbesets, skip = startRows[1], header = TRUE,
+     row.names = "ID_REF")
> rownames(sensitiveData) <- rownames(temp)
> sensitiveData[, 1] <- temp$VALUE
> for (i1 in 2:length(startRows)) {
+     temp <- read.table(file.path(softDir, "GSE350_family.soft"),
+         nrows = nProbesets, skip = startRows[i1], header = TRUE,
+         row.names = "ID_REF")
+     sensitiveData[, i1] <- temp$VALUE[match(rownames(sensitiveData),
+         rownames(temp))]
+ }
> rm(softDir)
```

Ok, we now have the Chang quantifications loaded in. We would like to pull this together into a single table, reordering things slightly to correct for the mislabeling.

```
> badCol <- which(names(resistantData) == "GSM4913")
> changSoft <- cbind(sensitiveData, resistantData[, badCol],
+     resistantData[, -badCol])
> names(changSoft)[11] <- "GSM4913"
```

Next, we reorder the gene rows in the Chang data to match the order in the Novartis data.

```
> changSoft <- changSoft[rownames(novartis), ]
```

Finally, we make an "info" file with the GEO names and the response status.

```
> changSoftStatus <- factor(rep(c("Resp", "NR"), times = c(11,
+     13)))
> changInfo <- data.frame(GEO.ID = colnames(changSoft),
+     Response = changSoftStatus)
```

4 Loading the dChip quantifications

We also downloaded the CEL files for the Chang data set from the GEO web site. The processed data in the SOFT files was produced using the DNA Chip Analyzer (dChip) using the $PM-MM$ algorithm developed by Li and Wong. [?] We processed the CEL files ourselves, using `dchip2006.exe` and the PM-only model. Here we simply load the results of that analysis.

```
> changDChip <- read.table(file.path("..", "PublicData",
+     "Chang", "Chang expression.txt"), sep = "\t", header = TRUE,
+     row.names = 1)
```

To ensure consistency, we order the rows (genes) to match the Novartis data set, and we order the columns (samples) to match the processed data from the SOFT files.

```
> changDChip <- changDChip[rownames(novartis), colnames(changSoft)]
```

5 Quantile normalization to match Novartis

We performed quantile normalization to map the distributions in the Chang data onto same quantiles used to normalize the NCI60 cell line data. To perform this normalization step, we must first get the quantiles. By using the Novartis quantiles on the log scale, this step simultaneously has the affect of log transforming the Chang data (base two).

```
> temp <- novartisNL
> for (i in 1:ncol(temp)) temp[, i] <- sort(temp[, i])
> nov.q <- sort(apply(temp, 1, median))
```

Now we can normalize and log transform the SOFT data.

```
> changSoftNL <- changSoft
> for (i in 1:ncol(changSoftNL)) changSoftNL[, i] <- nov.q[rank(changSoft[,
+     i])]
```

And we can do the same thing for the dChip data.

```
> changDChipNL <- changDChip
> for (i in 1:ncol(changDChipNL)) changDChipNL[, i] <- nov.q[rank(changDChip[,
+     i])]
```

6 Differences in the dChip models

The quantifications using the $PM - MM$ model differ slightly from the PM-only model. Here we pick one array and plot the normalized values from the two methods against one another (Figure 1).

Next, we plot the log-transformed but unnormalized data, just to point out that this is not caused by the normalization (Figure 2).

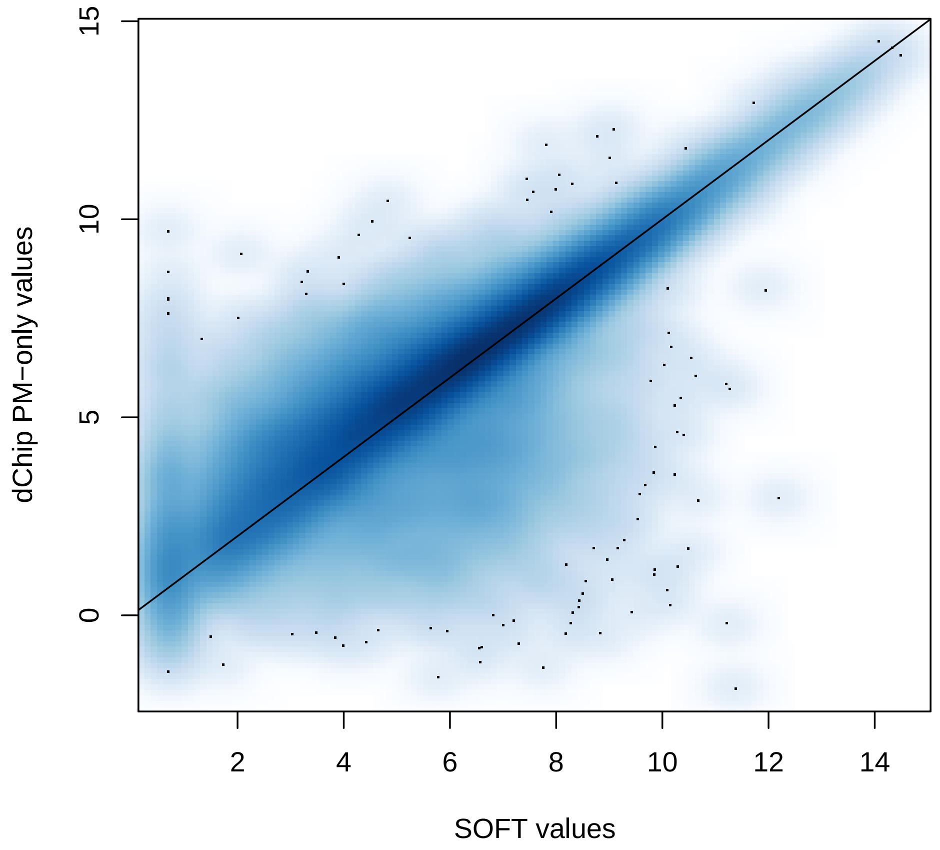

Figure 1: Smooth scatter plot of the PM-MM values from the SOFT files against the PM-only values, using normalized data.

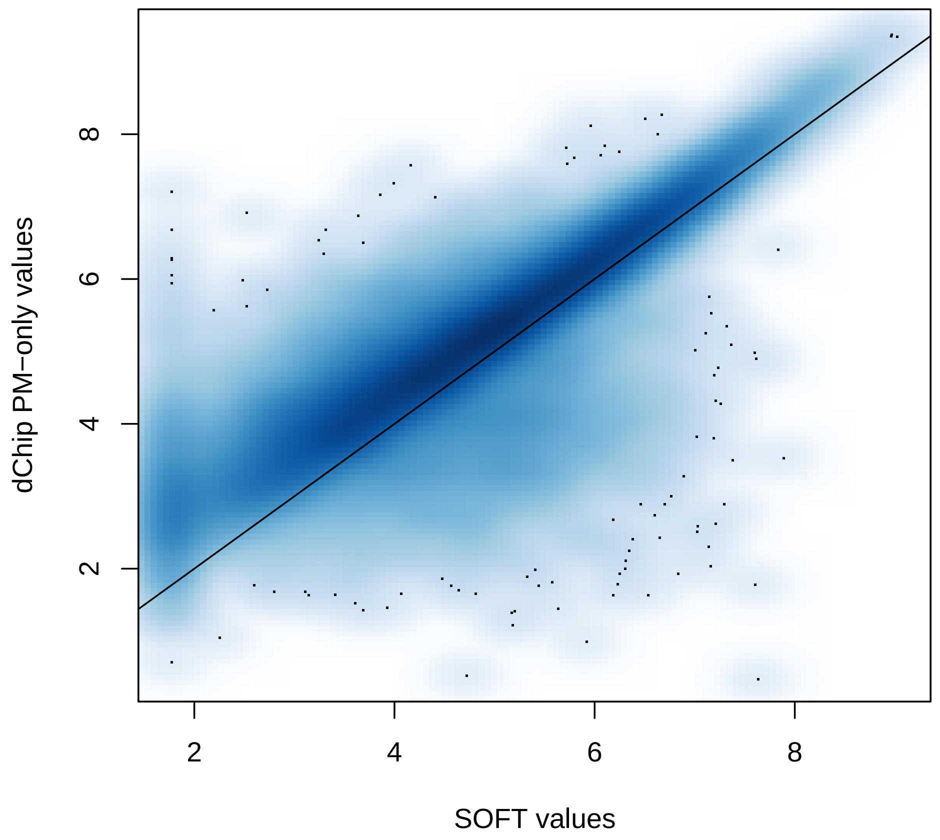

Figure 2: Smooth scatter plot of the PM-MM values from the SOFT files against the PM-only values, using unnormalized data.

7 Wrapup

```
> rm(temp, badCol, i, i1, startRows, changSoftStatus, nov.q,
+     nProbesets, resistantNames, resistantData, sensitiveNames,
+     sensitiveData)
```

```
> save(changInfo, changSoft, changSoftNL, changDChip, changDChipNL,
+     file = file.path("RDataObjects", "changData.Rda"))
```