# Preparing Public Data for Analysis

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

13 March 2007

## 1 Reformatting the Raw Source

We first make sure we have extracted our form of the data from the raw sources. Relative to the current directory, the NCI60 Affymetrix data and drug response data are both contained in the directory:

```
> nci60Dir <- file.path("..", "PublicData", "NCI60Processed")
> nci60Dir

[1] "../PublicData/NCI60Processed"
```

There are R scripts in that directory that invoke corresponding perl scripts to extract and format the data. The next two blocks of code serve as wrappers that only invoke those (time-consuming) steps if necessary.

The first block makes sure we have converted the Novartis U95A data into tabular format.

```
> if (!file.exists(file.path(nci60Dir, "Novartis.tsv"))) {
+     home <- setwd(nci60Dir)
+     source("parseNovartis.R")
+     source("get36460.R")
+     setwd(home)
+     rm(home)
+ }
```

The second block makes sure we have extracted the drug response data for 10 drugs from the huge files with many thousands of drugs.

```
> if (!file.exists(file.path(nci60Dir, "cancer60tgi.lis.csv"))) {
+     home <- setwd(nci60Dir)
+     source("extractDrugs.R")
+     setwd(home)
+     rm(home)
+ }
```

## 2 Producing an R Object Containing the Novartis Data

In order to speed things up later, we produce binary R data objects for various entities. In this section, we create (or load, if it already exists), the binary file `novartis.Rda`. We start, however, by explaining how the code works.

We start by loading the raw Novartis data. Because one of the rows was accidentally duplicated in the source file, we remove it. We also use the `Probe.Set` column to provide unique row names for the object.

```
> novtemp <- read.table(file.path(nci60Dir, "Novartis.tsv"),
+     sep = "\t", header = TRUE, as.is = TRUE)
> duprow <- which(novtemp[, "Probe.Set"] == "100_g_at")
> novartis <- novtemp[-duprow[2], 1:181]
> rownames(novartis) <- novartis[, "Probe.Set"]
> novartis <- novartis[2:181]
> rm(novtemp, duprow)
```

Because R is picky about column names, it has mangled some of the characters in the names of the cell lines when reading the data. We have to undo this process in order to be able to match the cell line names in this file with the same names in other files.

```
> cnov <- sort(unique(substring(colnames(novartis), 11)))
> temp <- sub(".ATCC", "/ATCC", cnov)
> temp <- sub(".ADR", "/ADR", temp)
> temp <- sub(".TB.", "(TB)", temp)
> temp <- gsub("\\.", "-", temp)
> temp <- sub("COLO.205", "COLO 205", temp)
> temp <- sub("RXF.393", "RXF 393", temp)
> temp <- sub("LOX.IMVI", "LOX IMVI", temp)
> temp <- sub("HS.578T", "HS 578T", temp)
> names(temp) <- cnov
> novinfo <- paste(substring(colnames(novartis), 9, 10),
+     temp[substring(colnames(novartis), 11)], sep = "")
> colnames(novartis) <- novinfo
> rm(temp, cnov, novinfo)
```

Now the original data source also contains some useful information about the cell lines, but it repeats that information thousands of times. Our perl scripts removed the information from the full `Novartis.tsv` file, but left a copy of it in a file that only contains data on one probe set. The next step is to read in that information and make sure it is compatible with the Affymetrix data.

```
> nov <- read.table(file.path(nci60Dir, "Novartis36460_at.csv"),
+     sep = ",", header = TRUE)
> nov$cellname <- sub("RXF.393", "RXF 393", nov$cellname)
```

```
> novname <- paste(substr(nov$ID, 9, 9), nov$cellname,
+     sep = ".")
> rownames(nov) <- novname
> nov <- nov[colnames(novartis), ]
> novartisInfo <- nov[, c("ID", "cellname", "pname", "panelnbr",
+     "cellnbr")]
> novartisInfo$ID <- factor(substring(novartisInfo$ID,
+     9, 10))
> novartisInfo$cellname <- factor(as.character(novartisInfo$cellname))
> rm(novname)
> if (any(rownames(novartisInfo) != colnames(novartis))) {
+     stop("bad data matching!")
+ }
```

The heart of the algorithm, then is to load the `novartis.Rda` file if it exists, and otherwise to create it.

```
> novfile <- file.path("RDataObjects", "novartis.Rda")
> if (file.exists(novfile)) {
+     load(novfile)
+ } else {
+     novtemp <- read.table(file.path(nci60Dir, "Novartis.tsv"),
+         sep = "\t", header = TRUE, as.is = TRUE)
+     duprow <- which(novtemp[, "Probe.Set"] == "100_g_at")
+     novartis <- novtemp[-duprow[2], 1:181]
+     rownames(novartis) <- novartis[, "Probe.Set"]
+     novartis <- novartis[2:181]
+     rm(novtemp, duprow)
+     cnov <- sort(unique(substring(colnames(novartis),
+         11)))
+     temp <- sub(".ATCC", "/ATCC", cnov)
+     temp <- sub(".ADR", "/ADR", temp)
+     temp <- sub(".TB.", "(TB)", temp)
+     temp <- gsub("\\.", "-", temp)
+     temp <- sub("COLO.205", "COLO 205", temp)
+     temp <- sub("RXF.393", "RXF 393", temp)
+     temp <- sub("LOX.IMVI", "LOX IMVI", temp)
+     temp <- sub("HS.578T", "HS 578T", temp)
+     names(temp) <- cnov
+     novinfo <- paste(substring(colnames(novartis), 9,
+         10), temp[substring(colnames(novartis), 11)],
+         sep = "")
+     colnames(novartis) <- novinfo
```

```
+     rm(temp, cnov, novinfo)
+     nov <- read.table(file.path(nci60Dir, "Novartis36460_at.csv"),
+         sep = ",", header = TRUE)
+     nov$cellname <- sub("RXF.393", "RXF 393", nov$cellname)
+     novname <- paste(substr(nov$ID, 9, 9), nov$cellname,
+         sep = ".")
+     rownames(nov) <- novname
+     nov <- nov[colnames(novartis), ]
+     novartisInfo <- nov[, c("ID", "cellname", "pname",
+         "panelnbr", "cellnbr")]
+     novartisInfo$ID <- factor(substring(novartisInfo$ID,
+         9, 10))
+     novartisInfo$cellname <- factor(as.character(novartisInfo$cellname))
+     rm(novname)
+     if (any(rownames(novartisInfo) != colnames(novartis))) {
+         stop("bad data matching!")
+     }
+     save(novartis, novartisInfo, file = novfile)
+ }
> rm(novfile)
```

## 3 Producing an R Object With the NCI60 Drug Response Data

Above, we ran a perl script to extract the data on ten drugs from the GI50, LC50, and TGI data on more than 44,000 drugs for which the DTP has collected response data for the NCI60 cell lines. We now want to assemble that data into a format we can reuse, again storing the items as a binary R data object.

First, we have to map the drug names to their NSC numbers by hand; similar code appears in the "strip7.pl" perl script.

```
> nsc <- c(628503, 123127, 26271, 141540, 125973, 19893,
+     609699, 749, 740, 241240, 119875, 83142)
> abbrev <- c("Doce", "Adria", "Cytox", "Etopo", "Taxol",
+     "5-FU", "Topo", "azaG", "Metho", "Carbo", "Cisp",
+     "Dauno")
> names(nsc) <- abbrev
> names(abbrev) <- as.character(nsc)
```

Next, we define a function that can read the data from the comma-separated-values formatted source files.

```
> getNSC <- function(filename, target, path = NULL) {
+     fixup <- function(x) {
```

```
+         x <- as.character(x)
+         x <- sub(" $", "", x)
+         factor(x)
+     }
+     if (!is.null(path)) {
+         filename <- file.path(path, filename)
+     }
+     rt <- read.table(filename, sep = ",", header = TRUE,
+         row.names = NULL, quote = "", comment.char = "")
+     rt$NSC <- factor(rt$NSC)
+     rt$LCONC <- factor(rt$LCONC)
+     temp <- as.character(rt$STDDEV)
+     temp <- gsub(" ", "", temp)
+     temp[temp == "."] <- NA
+     rt$STDDEV <- as.numeric(temp)
+     rt$PANEL <- fixup(rt$PANEL)
+     rt$CELL <- fixup(rt$CELL)
+     rt[, "Key"] <- factor(paste(as.character(rt$NSC),
+         as.character(rt$LCONC), sep = "L"))
+     len <- table(rt$CELL, rt$Key)
+     form <- dummy ~ 1 | NSC/LCONC/CELL
+     form[[2]] <- as.name(target)
+     require(nlme)
+     gd <- groupedData(form, data = rt)
+     simple <- gsummary(gd, mean, na.rm = TRUE)
+     result <- tapply(simple[, target], list(simple$CELL,
+         simple$Key), mean, na.rm = TRUE)
+     result
+ }
```

Now we can go ahead and load the raw data.

```
> gi50 <- getNSC("cancer60gi50.lis.csv", "NLOGGI50", nci60Dir)
> lc50 <- getNSC("cancer60lc50.lis.csv", "NLOGLC50", nci60Dir)
> tgi <- getNSC("cancer60tgi.lis.csv", "NLOGTGI", nci60Dir)
```

Finally, we pick out the drug response data for the cell lines that are actually present in the Novartis Affymetrix study.

```
> novi <- (rownames(gi50) %in% as.character(novartisInfo[,
+     "cellname"]))
> GI50 <- gi50[novi, ]
> novi <- (rownames(lc50) %in% as.character(novartisInfo[,
```

```
+     "cellname"]))
> LC50 <- lc50[novi, ]
> novi <- (rownames(tgi) %in% as.character(novartisInfo[,
+     "cellname"]))
> TGI <- tgi[novi, ]
> rm(novi, gi50, lc50, tgi)
```

Again, the heart of the algorithm, then is to load the `nscData.Rda` file if it exists, and otherwise to create it.

```
> nscfile <- file.path("RDataObjects", "nscdata.Rda")
> if (file.exists(nscfile)) {
+     load(nscfile)
+ } else {
+     nsc <- c(628503, 123127, 26271, 141540, 125973, 19893,
+         609699, 749, 740, 241240, 119875, 83142)
+     abbrev <- c("Doce", "Adria", "Cytox", "Etopo", "Taxol",
+         "5-FU", "Topo", "azaG", "Metho", "Carbo", "Cisp",
+         "Dauno")
+     names(nsc) <- abbrev
+     names(abbrev) <- as.character(nsc)
+     getNSC <- function(filename, target, path = NULL) {
+         fixup <- function(x) {
+             x <- as.character(x)
+             x <- sub(" $", "", x)
+             factor(x)
+         }
+         if (!is.null(path)) {
+             filename <- file.path(path, filename)
+         }
+         rt <- read.table(filename, sep = ",", header = TRUE,
+             row.names = NULL, quote = "", comment.char = "")
+         rt$NSC <- factor(rt$NSC)
+         rt$LCONC <- factor(rt$LCONC)
+         temp <- as.character(rt$STDDEV)
+         temp <- gsub(" ", "", temp)
+         temp[temp == "."] <- NA
+         rt$STDDEV <- as.numeric(temp)
+         rt$PANEL <- fixup(rt$PANEL)
+         rt$CELL <- fixup(rt$CELL)
+         rt[, "Key"] <- factor(paste(as.character(rt$NSC),
+             as.character(rt$LCONC), sep = "L"))
+         len <- table(rt$CELL, rt$Key)
```

```
+         form <- dummy ~ 1 | NSC/LCONC/CELL
+         form[[2]] <- as.name(target)
+         require(nlme)
+         gd <- groupedData(form, data = rt)
+         simple <- gsummary(gd, mean, na.rm = TRUE)
+         result <- tapply(simple[, target], list(simple$CELL,
+             simple$Key), mean, na.rm = TRUE)
+         result
+     }
+     gi50 <- getNSC("cancer60gi50.lis.csv", "NLOGGI50",
+         nci60Dir)
+     lc50 <- getNSC("cancer60lc50.lis.csv", "NLOGLC50",
+         nci60Dir)
+     tgi <- getNSC("cancer60tgi.lis.csv", "NLOGTGI", nci60Dir)
+     novi <- (rownames(gi50) %in% as.character(novartisInfo[,
+         "cellname"]))
+     GI50 <- gi50[novi, ]
+     novi <- (rownames(lc50) %in% as.character(novartisInfo[,
+         "cellname"]))
+     LC50 <- lc50[novi, ]
+     novi <- (rownames(tgi) %in% as.character(novartisInfo[,
+         "cellname"]))
+     TGI <- tgi[novi, ]
+     rm(novi, gi50, lc50, tgi)
+     save(nsc, abbrev, GI50, LC50, TGI, file = nscfile)
+ }
> rm(nscfile)
```