Introduction to pre-processing – lab

Martin Morgan

28 January 2010

This lab introduces pre-processing and quality assessment, focusing on Affymetrix single channel data. It is based on Chapter 3 of *Bioconductor Case Studies*, by Hahne, Huber, Gentleman, and Falcon.

1. Briefly read through the chapter 3 handout.
2. Ensure that you have the CLL package installed. To do this, start an R session and type the command

   ```
   > library(CLL)
   ```

   If R responds with `Error in library(CLL) : there is no package called 'CLL'` then use the ethernet connection to install the package:

   ```
   > source("http://wilson2.fhcrc.org/installScripts/biocLite.R")
   > biocLite(CLL)
   ```

   (use `source("http://bioconductor.org/biocLite.R")` for access after the course is complete.)
3. (Optional) Copy the CEL files from the thumb drive to your disk.

Exercise 1
Attach the CLL package, and use the `data` *function to load* `CLLbatch` *into your work space. The* `CLLbatch` *object contains raw probe values; they have not been summarized or normalized in any way. What is the class of* `CLLbatch`*? Can you find the help page describing this class? What about a help page describing the* `CLLbatch` *itself? From looking at what is printed when you type* `CLLbatch` *at the command line, what basic information (e.g., number of genes, number of samples, the chip on which the expression data were collected) can you obtain about* `CLLbatch`*?*

Some key command for exploring the `CLLbatch` *object include:*

```
> library(CLL)
> data(CLLbatch)
> CLLbatch
```

```
AffyBatch object
size of arrays=640x640 features (99224 kb)
cdf=HG_U95Av2 (12625 affyids)
number of samples=24
number of genes=12625
annotation=hgu95av2
notes=
```

```
> class(CLLbatch)
```

```
[1] "AffyBatch"
attr(,"package")
[1] "affy"
```

Find help with

```
> help(class(CLLbatch))
```

Exercise 2

As an exercise, we'll add some phenotypic data `CLLbatch`*. Do this by using* `data(disease)`*. Take a minute to explore this object. What class is it, how many rows does it have, what are its column names?*

Update the row names of the phenotype data.frame to reflect the SampleID, and remove the `.CEL` *file extension from the sampleNames of the AffyBatch.*

Create a vector `mt` *that tells us how the row names of the phenotype data.frame align with the column names of the AffyBatch.*

Create the 'metadata' describing the phenotype data, then an AnnotatedDataFrame combing the phenotype data and metadata, and finally add this to the `CLLbatch` *AffyBatch. Use* `mt` *to make sure the order of the phenotype and expression data is consistent.*

One array is missing disease status. Drop this from further analysis.

Load the data set

```
> data("disease")
> class(disease)
```

```
[1] "data.frame"
```

```
> head(disease)
```

```
  SampleID  Disease
1    CLL10     <NA>
2    CLL11 progres.
3    CLL12   stable
4    CLL13 progres.
5    CLL14 progres.
6    CLL15 progres.
```

Update row and sample names

```
> rownames(disease) <- disease$SampleID
> sampleNames(CLLbatch) <- sub("\\.CEL$", "",
+     sampleNames(CLLbatch))
```

Create a vector that describes how the row names of `disease` *align with the sample names of* `CLLbatch`*.*

```
> mt <- match(rownames(disease), sampleNames(CLLbatch))
```

Create the metadata and AnnotatedDataFrame for the phenotype data, and add this to the AffyBatch.

```
> vmd = data.frame(labelDescription = c("Sample ID",
+     "Disease status: progressive or stable disease"))
> phenoData(CLLbatch) = new("AnnotatedDataFrame",
+     data = disease[mt, ], varMetadata = vmd)
```

Drop the array that is missing disease status

```
> CLLbatch = CLLbatch[, !is.na(CLLbatch$Disease)]
```

Exercise 3

There is an array quality metrics report for `CLLbatch`*; we'll review it in class. The report suggests that array CLL1 is suspect, so remove it from further analysis.*

The command to generate an array quality metrics report is straight-forward, but the computation is time consuming. Here's the command; the report will be reviewed in class.

```
> reportDir <- "c:/path/to/empty/directory"
> arrayQualityMetrics(CLLbatch, reportDir)
```

To remove the bad array, we might

```
> badArray = match("CLL1", sampleNames(CLLbatch))
> CLLB = CLLbatch[, -badArray]
```

Exercise 4

Pre-process the AffyBatch using `rma`*. What is the class of the resulting object? How many probesets are there in this object? Can you extract the summarized expression values (using the* `exprs` *function? What's happened to the phenotypic data that we added to* `CLLbatch`*?*

Often `just.rma` *is used to go directly from CEL file to ExpressionSet in a fast and memory efficient way.*

```
> CLLrma <- rma(CLLB)

Background correcting
Normalizing
Calculating Expression

> class(CLLrma)

[1] "ExpressionSet"
attr(,"package")
[1] "Biobase"

> dim(CLLrma)

Features  Samples
   12625       22

> e <- exprs(CLLrma)
> phenoData(CLLrma)

An object of class "AnnotatedDataFrame"
  sampleNames: CLL11, CLL12, ..., CLL9  (22 total)
  varLabels and varMetadata description:
    SampleID: Sample ID
    Disease: Disease status: progressive or stable dise
  ase
```

Exercise 5

Functions can often help to reduce the need to repeatedly perform routine tasks, and in the process make work flow less error prone. The NUSE boxplot in the chapter represents two steps: processing an AffyBatch object using `fitPLM`*, and then plotting the results. Capture these steps in a function, and add it to your package.*

Some steps might include:

1. *Decide on what your function will do: for an AffyBatch object, fit a probe-wise linear model. Then create a NUSE plot.*
2. *Decide what the inputs and outputs of your function must be. The input will be an AffyBatch object. The 'output' is really a side effect, the creation of a plot.*
3. *Create a simple function taking the appropriate inputs and outputs, but doing nothing else. Choose a function name and arguments with an eye toward making their purpose clear. Create the function in a file in the* `R` *directory of your simple package.*
4. *Fill in the function with the details.*

5. *Test the function.*

Here is a solution

```
> plotNUSE <- function(abatch)
+ {
+     ## The package Depends: on affyPLM; otherwise, we might
+     ## library("affyPLM")
+     plmData <- fitPLM(abatch)
+     ## now draw the plot
+     boxplot(plmData, main="NUSE", outline=FALSE, col="lightblue",
+             las=3, whisklt=0, staplelty=0)
+ }
```