Microarray Analysis: Self-Study Exercises

Martin Morgan, Chao-Jen Wong
Fred Hutchinson Cancer Research Center
9-10 December, 2010

(Adapted from F. Hahne and R. Gentleman, ‘The ALL Dataset’, in Bioconductor Case Studies [2])

1 Structures for genomic data: ExpressionSet

Genomic data can be very complex, usually consisting of a number of different bits and pieces. In Bioconductor we have taken the approach that these pieces should be stored in a single structure to easily manage the data. The package Biobase contains standardized data structures to represent genomic data. The ExpressionSet class is designed to combine several different sources of information into a single convenient structure. An ExpressionSet can be manipulated (e.g., subsetted, copied), and is the input to or output of many Bioconductor functions.

The data in an ExpressionSet consist of

- **assayData**: Expression data from microarray experiments (assayData is used to hint at the methods used to access different data components, as we show below).

- **metadata**: A description of the samples in the experiment (phenoData), metadata about the features on the chip or technology used for the experiment (featureData), and further annotations for the features, for example gene annotations from biomedical databases (annotation).

- **experimentData**: A flexible structure to describe the experiment.

The ExpressionSet class coordinates all of these data, so that you do not usually have to worry about the details. However, an ExpressionSet needs to be created in the first place, because it will be the starting point for many of the analyses using Bioconductor software.

ExpressionSet instances are created in one of two ways. Often, an ExpressionSet is the output of an R function. For instance, justRMA in the affy Bioconductor package reads in manufacturer CEL files and outputs an ExpressionSet.
Alternatively, an ExpressionSet can be assembled from its constituent parts; this is illustrated in the vignette called “An introduction to Biobase and ExpressionSets”. For future reference, this vignette can be viewed in your web browser with the command

```r
> browseVignettes("Biobase")
```

or by visiting the Biobase page on the Bioconductor web site.

Here we use an ExpressionSet derived from a study of Acute Lymphoblastic Leukemia (ALL). The data set is available in the data package ALL. The main object in the ALL package is ALL, an instance of ExpressionSet. It consist of microarray data (chip series HG-U95Av2) from 128 individuals – 95 sample with B-cell ALL and 33 with T-cell ALL. The expression measures have been preprocessed using the robust multichip average (RMA) method, implemented in the package affy.

Let’s first load the ALL data, and view the ALL object.

```r
> library(ALL) # attach the ALL package to the search path
> data(ALL)    # load the ALL data into the global work space
> ALL          # view the ALL instance -- our first ExpressionSet!
```

ExpressionSet (storageMode: lockedEnvironment)

assayData: 12625 features, 128 samples

- element names: exprs
- protocolData: none
- phenoData
- sampleNames: 01005 01010 ... LAL4 (128 total)
- varLabels: cod diagnosis ... date last seen (21 total)
- varMetadata: labelDescription
- featureData: none
- experimentData: use 'experimentData(object)'
- pubMedIds: 14684422 16243790
- Annotation: hgu95av2

This prints a short overview of the ExpressionSet. The ExpressionSet is an example of an S4 class. The authors of the class have provided methods for accessing the data in the class. For instance,

- `exprs(ALL)` returns the matrix of expression values (probe sets as rows, samples as columns). In the ALL data, the expression values are pre-processed and log-transformed.

- `pData(ALL)` extracts a data frame describing the sample phenotype data.

- `annotation(ALL)` reports the type of microarray chip used in this experiment.

There are additional methods defined to make common operations easy. For instance, `s` allows one to access columns of phenotypic data. This
retrieves the vector of values in the BT column of pData(ALL).

2 Subsetting

Another important method available with the ExpressionSet class is subsetting. The underlying expression values in an experiment are represented as a matrix, and the ExpressionSet reflects this data – it can be subset by rows (features) and / or columns (samples). For instance, to select the samples that have mol.biol phenotype NEG, we might extract the mol.biol phenotype data using $, and then compare this to NEG to produce a vector of logical values that are TRUE whenever the value of mol.biol is NEG; to illustrate, we summarize this vector by tabulating the TRUE and FALSE occurrences

```r
> table(ALL$mol.biol == "NEG")

FALSE  TRUE
54    74
```

We can use this to create a subset of the original ExpressionSet with just these samples

```r
> idx <- ALL$mol.biol == "NEG"
> ALL[,idx]
```

ExpressionSet (storageMode: lockedEnvironment)

assayData: 12625 features, 74 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: 01010 04007 ... LAL4 (74 total)
  varLabels: cod diagnosis ... date last seen (21 total)
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
  pubMedIds: 14684422 16243790
Annotation: hgu95av2
Note that there are only 74 samples in this ExpressionSet, and that the phenotype and expression data have been subset in a coordinated fashion.

Additional methods operating on ExpressionSet can be found on the help page accessible with

> class?ExpressionSet

**Exercise 1**

The objective of this exercise is to create a subset of ALL containing only those samples with B-cell ALL tumors with either BCR/ABL or NEG abnormalities.

- Which samples originate from B-cell tumors? The BT covariate in the phenotypic data encodes the tissue type; use grep to select any value of BT that begins with the letter B.

- Which samples are associated with either the molecular subtype BCR/ABL or NEG? The molecular subtypes are labeled in the mol.biol phenotypic covariate. To answer this question, convert ALL$mol.biol from a factor to a character vector using as.character. Then use the `%in%` function (see ?%in%) to match either element of the character vector c("NEG", "BCR/ABL").

- Use the intersect function to find the intersection between the set of samples from B-cell tumors, and the set of samples with either NEG or BCR/ABL molecular biology.

- Finally, create a subset of the ALL object that contains B-cell tumors with NEG or BCR/ABL cytotype.

**Solution:**

```r
> bcell <- grep("B", as.character(ALL$BT))
> types <- c("NEG", "BCR/ABL")
> moltyp <- which(as.character(ALL$mol.biol) %in% types)
> idx <- intersect(bcell, moltyp)
> ALL_bcrneg <- ALL[, idx]
> ALL_bcrneg
```

ExpressionSet (storageMode: lockedEnvironment)
assayData: 12625 features, 79 samples
element names: exprs
protocolData: none
phenoData
  sampleNames: 01005 01010 ... 84004 (79 total)
  varLabels: cod diagnosis ... date last seen (21 total)
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'

4
**3 Non-specific filtering**

A common next step is to use a non-specific filter to remove probe sets that we know, *a priori* will not be informative in our analysis. For instance, probe sets that are not annotated to a particular ENTREZ gene id will not be informative in a study relies on ENTREZ ids to understand biology. Here we use the `nsFilter` function from the `genefilter` package to filter on a number of different criteria.

```r
> library(genefilter)
> filt_bcrneg <- nsFilter(ALL_bcrneg, require.entrez=TRUE,
+ remove.dupEntrez=TRUE,
+ feature.exclude="^AFFX")
> filt_bcrneg$filter.log
```

```r
$numLowVar
[1] 4400

$numDupsRemoved
[1] 2907

$feature.exclude
[1] 19

$numRemoved.ENTREZID
[1] 900
```

This removes probes that do not have ENTREZ gene identifiers, groups of probes that map to the same ENTREZ gene id, and probes that are annotated as Affymetrix control probes. An entrance into the literature on non-specific filtering is [1].

**Exercise 2**

Consult the manual for the function `nsFilter` (use `?nsFilter` after loading the `genefilter` library) for the different options for filtering features from an `ExpressionSet` and the return value of `filt_bcrneg$eset`

```r
> ALLfilt_bcrneg <- filt_bcrneg$eset
```
4 Assessing Quality

Finally, assess the quality of the ALL subset using the `arrayQualityMetrics` package. To do this, attach the library, evaluate the `arrayQualityMetrics` function, and browse the result. This process can be time- and memory-intensive, so for the purposes of this lab it might make sense to perform array quality metrics on a small subset of the total sample.

```r
> library(arrayQualityMetrics)
> outdir <- tempfile()
> subset <- 1:6
> aqm <- arrayQualityMetrics(ALL[,subset], outdir)
```

Review the report in your browser by viewing the `outdir` directory and selecting the `QMreport.html` file.

```r
> browseURL(outdir)
```

Interpret the report with your neighbors.

5 Directions

The text and exercises introduce you to S4 objects, accessors, and subsetting. You gained practice using the R help system. The exercises illustrated key data manipulation steps in a microarray work flows. There are many opportunities for learning more; The Bioconductor Case Studies book [2] and limma Bioconductor package vignette (for two-color array pre-processing) are two excellent next steps.

- R version 2.12.1 beta (2010-12-07 r53813), i386-apple-darwin9.8.0
- Locale: C
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: ALL 1.4.7, AnnotationDbi 1.12.0, Biobase 2.10.0, DBI 0.2-5, GO.db 2.4.5, RSQLite 0.9-4, SeattleIntro2010 0.0.33, biomaRt 2.6.0, genefilter 1.32.0, hgu95av2.db 2.4.5, org.Hs.eg.db 2.4.6
- Loaded via a namespace (and not attached): RCurl 1.4-3, XML 3.2-0, annotate 1.28.0, splines 2.12.1, survival 2.36-2, tools 2.12.1, xtable 1.5-6

References
