Lab: Introduction to Bioconductor's ExpressionSet Class

Seth Falcon, Martin Morgan, and Robert Gentleman

6 October, 2006

1 Introduction

In this lab you will learn how to create and manipulate *ExpressionSet* objects. In the processes you will have an opportunity to practice some basic R skills.

2 Loading Packages

The definition of the *ExpressionSet* class along with many methods (OOP-speak for functions) for manipulating *ExpressionSet* objects are defined in the Biobase package. In general, you need to load class and method definitions before you use them. When using Bioconductor, this means loading R packages using `library` or `require`.

```
> library("Biobase")
```

Exercise 1
What happens when you try to load a package that is not installed?

3 Building an ExpressionSet From Scratch

The data from many high-throughput genomic experiments, such as microarray experiments, can be summarized by a matrix of expression. The matrix has F rows and S columns, where F is the number of features on the chip and S is the number of samples. In addition, one will have a data table that provides information on the samples (e.g., sex, age, and treatment status). The information describing the samples, or *phenotypes*, can be represented as an S by V table, where V is the number of covariates. In R, we use a *data.frame* to hold this "phenoData". Note that the columns of the expression matrix must align with the rows of the phenoData table. The *ExpressionSet* class provides a container for the expression matrix and phenoData and keep the two properly aligned.
In the exercises below, you will learn how to create a new *ExpressionSet* instance given a matrix of expression values and a *data.frame* containing the phenoData. Other labs will cover the creation of the expression matrix from raw CEL files for microarray data.

3.1 Loading the Expression Matrix

3.1.1 R Binary Files

You can save objects in your R session to a file using the `save` function. By default, this will create a file in R's internal binary format. The same binary file produced by a call to `save` can be used on Linux, OS X, and Windows.
You can load the objects saved in an rda file using the `load` function.

3.1.2 Loading `ALLmat`

Below, we will load a large *matrix* of expression values stored in the file `ALLmat.rda`. The example assumes that the file is in the current working directory. You can change the working directory using `setwd`.

```
> getwd()

[1] "/Users/seth/proj/COURSES/bioc_R_intro/Bioc_intro"

> ls()

character(0)

> load("ALLmat.rda")
> ls()

[1] "ALLmat"
```

The `ls` function lists the R objects in your current working environment. You should see a new object named `ALLmat` appear after the call to `load`.

Exercise 2
Open and read the help page for `load`.

Exercise 3
Determine the class and dimension of the matrix.

3.2 Loading the Phenotype Data

Covariates describing the samples in this experiment have been saved to a whitespace-delimited text file called `ALL-sample-info.txt`. Delimited text files are common and can be produced from Microsoft Excel by saving as a "csv" file (this stands for comma separated values, but the separator does not have to be a comma).
R's `read.table` function is a powerful tool for reading delimited text files. Below, you will use it to read in the phenoData.

```
> samples <- read.table("ALL-sample-info.txt", header = TRUE,
+     check.names = FALSE)
```

Exercise 4
What class does `read.table` *return?*

Exercise 5
Determine the column names of `samples`*. Hint:* `apropos("name")`*.*

Exercise 6
Use `sapply` *to determine the classes of each column of* `samples`*. Hint: read the help page for* `sapply`*.*

Exercise 7
Examine the sex and age of the 15th and 30th samples. Do the same for the sample with `cod` *matching 11005.*

To make the phenoData more self-documenting, we have a file `ALL-varMeta.txt` that gives a description for each column of `samples`. You can use `read.table` to read this file into an R object.

```
> varInfo <- read.table("ALL-varMeta.txt", header = TRUE,
+     colClasses = "character")
> varInfo[c("sex", "cod", "mol.biol"), , drop = FALSE]
```

```
               labelDescription
sex       Gender of the patient
cod                  Patient ID
mol.biol      molecular biology
```

Bioconductor's Biobase package provides a class called *AnnotatedDataFrame* that allows you to store the column descriptions with the data. Create an *AnnotatedDataFrame* instance for our phenoData by following the example below.

```
> pd <- new("AnnotatedDataFrame", data = samples, varMetadata = varInfo)
```

3.3 Creating an *ExpressionSet*, finally

Now that you have a *matrix* of expression values (`ALLmat`) and an *AnnotatedDataFrame* containing the phenotype information (`pd`), you are ready to put the pieces together and create an *ExpressionSet*.

```
> ALLSet <- new("ExpressionSet", exprs = ALLmat, phenoData = pd,
+     annotation = "hgu95av2")
```

The `annotation` argument is intended to hold the name of the R package that provides annotation data for the chip used in the experiment. In this case, the appropriate annotation package is hgu95av2.

3.4 *ExpressionSet* Basics

Now that you have an *ExpressionSet* instance, let's explore some of the basic operations. You can get an overview of the structure and available methods for *ExpressionSet* objects by reading the help page:

```
> help("ExpressionSet-class")
> "?"(class, ExpressionSet)
```

When you print an *ExpressionSet* object, a brief summary of the contents of the object is displayed. All of the data contained by the *ExpressionSet* is not shown. This would not be useful as it would fill your screen with data.

```
> ALLSet

Instance of ExpressionSet

assayData
  Storage mode: lockedEnvironment
  featureNames: 1000_at, 1001_at, 1002_f_at, ..., AFFX-YEL021w/URA3_at, AFFX-YEL024w/RIP
  Dimensions:
        exprs
Rows    12625
Samples   128

phenoData
  sampleNames: 01005, 01010, 03002, ..., 83001, LAL4 (128 total)
  varLabels:
    cod:  Patient ID
    diagnosis:  Date of diagnosis
    sex:  Gender of the patient
    age:  Age of the patient at entry
    BT:  does the patient have B-cell or T-cell ALL
    ...: ...
    relapse:  Relapse? Derived from f.u
    transplant:  did the patient receive a bone marrow transplant? Derived from f.u
    f.u:  follow up data available
    date last seen:  date patient was last seen
    (21 total)

Experiment data
  Experimenter name:
  Laboratory:
  Contact information:
```

```
  Title:
  URL:
  PMIDs:
  No abstract available.

Annotation [1] "hgu95av2"
```

3.4.1 Accessing Data Elements

A number of accessor functions are available to extract data from an *ExpressionSet* instance. You can access the columns of the phenotype data (an *AnnotatedDataFrame* instance) using `$`:

```
> ALLSet$sex[1:5] == "F"

[1] FALSE FALSE  TRUE FALSE FALSE

> ALLSet$"t(9;22)"[1:5]

[1]  TRUE FALSE    NA FALSE FALSE
```

You can retrieve the names of the features using `featureNames`. For many microarray datasets, the feature names are the probeset identifiers.

```
> featureNames(ALLSet)[1:5]

[1] "1000_at"   "1001_at"   "1002_f_at" "1003_s_at" "1004_at"
```

The unique identifiers of the samples in the data set are available via the `sampleNames` method. The `varLabels` method lists the column names of the phenotype data:

```
> sampleNames(ALLSet)[1:5]

[1] "01005" "01010" "03002" "04006" "04007"

> varLabels(ALLSet)

 [1] "cod"            "diagnosis"      "sex"
 [4] "age"            "BT"             "remission"
 [7] "CR"             "date.cr"        "t(4;11)"
[10] "t(9;22)"        "cyto.normal"    "citog"
[13] "mol.biol"       "fusion protein" "mdr"
[16] "kinet"          "ccr"            "relapse"
[19] "transplant"     "f.u"            "date last seen"
```

You can extract the expression *matrix* and the *AnnotatedDataFrame* of sample information using `exprs` and `phenoData`, respectively:

```
> mat <- exprs(ALLSet)
> adf <- phenoData(ALLSet)
```

3.4.2 Subsetting

Probably the most useful operation to perform on *ExpressionSet* objects is subsetting. Subsetting an *ExpressionSet* is very similar to subsetting the expression *matrix* that is contained within the *ExpressionSet*, the first argument subsets the features and the second argument subsets the samples. Here are some examples:
A new *ExpressionSet* consisting of the 5 features and the first 3 samples:

```
> vv <- ALLSet[1:5, 1:3]
> dim(vv)

   Rows Samples
      5       3

> featureNames(vv)

[1] "1000_at"   "1001_at"   "1002_f_at" "1003_s_at" "1004_at"

> sampleNames(vv)

[1] "01005" "01010" "03002"
```

A subset consisting of only the male samples:

```
> males <- ALLSet[, ALLSet$sex == "M"]
```

Samples that have B-cell type ALL:

```
> anyB <- grep("^B", ALLSet$BT)
> bcell <- ALLSet[, anyB]
```

4 What was used to create this document

The version number of R and the packages and their versions that were used to generate this document are listed below.

- Version 2.3.1 Patched (2006-06-08 r38315), `powerpc-apple-darwin8.6.0`
- Base packages: base, datasets, grDevices, graphics, methods, stats, tools, utils
- Other packages: Biobase 1.10.0