1

The ALL Dataset

F. Hahne and R. Gentleman

Abstract

In this initial chapter we briefly describe the typical data preprocessing steps for a sample dataset that will be used in many of the following exercises.

1.1 Introduction

In the course of this book we frequently need a dataset that can be used to demonstrate the usage of Bioconductor software. For many of the methods we consider, the initial steps of the analyis procedure are more or less identical. Usually, they comprise subsetting of the data followed by a nonspecific filtering step to remove probe sets that are not likely to be informative. In this introductory chapter we briefly guide you through these first steps needed to obtain data suitable for the various analyses.

1.2 The ALL data

The `ALL` data consist of microarrays from 128 different individuals with acute lymphoblastic leukemia (ALL). There are 95 samples with B-cell ALL and 33 with T-cell ALL and because these are different tissues and quite different diseases we consider them separately, and typically focus on the B-cell ALL tumors. Two different analyses have been reported in (Chiaretti et al., 2004, 2005), which can be consulted for more detail. A number of covariates are stored along with data, describing more general properties of the patients such as age or sex but also a whole range of clinical parameters about type and stage of the disease. Details about these can be found on the manual page for this dataset. The data have been jointly normalized using `rma` and stored in the form of an *ExpressionSet* (see Chapter 2 for details on this class).

F. Hahne et al., *Bioconductor Case Studies*, DOI: 10.1007/978-0-387-77240-0_1,

1.3 Data subsetting

We first load the **ALL** package and attach the data to our work space.

```
> library("Biobase")
> library("ALL")
> library("genefilter")
> data("ALL")
```

An interesting subset, with two groups having approximately the same number of samples in each group, is the comparison of the B-cell tumors found to carry the BCR/ABL mutation to those B-cell tumors with no observed cytogenetic abnormalities. These samples are labeled BCR/ABL and NEG in the `mol.biol` covariate. The BCR/ABL mutation, also known as the *Philadelphia chromosome*, was the first cytogenetic aberration that could be associated with the development of cancer, leading the way to the current understanding of the disease. In tumors harboring the BCR/ABL translocation a short piece of chromosome 22 is exchanged with a segment of chromosome 9. As a consequence, a constitutively active fusion protein is transcribed which acts as a potent mitogene, leading to uncontrolled cell division.

Not all leukemia tumors carry the Philadelphia chromosome; there are other mutations that can be responsible for neoplastic alterations of blood cells, for instance a translocation between chromosomes 4 and 11 (ALL1/AF4), and in one of the exercises we also use data from a subset of these tumors.

First, we select those samples originating from B-cell tumors by searching the `BT` variable (which distinguishes the B-cell from the T-cell tumors) for entries beginning with the letter *B* using a regular expression.

```
> bcell = grep("^B", as.character(ALL$BT))
```

Next, we want to know which of the samples are of molecular types BCR/ABL or NEG.

```
> types = c("NEG", "BCR/ABL")
> moltyp = which(as.character(ALL$mol.biol) %in% types)
```

Combining these two selection criteria gives us a data set for carrying out comparisons between tumors harboring the BCR/ABL translocation and those that did not have any of the tested molecular abnormalities.

```
> ALL_bcrneg = ALL[, intersect(bcell, moltyp)]
```

One more step is required. Some of the sample annotation data is kept in variables of type *factor*. These are variables that can only take a number

of discrete categorical values. The set of possible values of a factor is called its levels (you may consult the manual page by typing "`? factor`"). Because we have reduced the set of samples, we only need fewer factor levels than were present in `ALL`, and the most succint way to reduce the set of levels of a factor to those that are actually present is by calling the constructor function `factor` on it again.

```
> ALL_bcrneg$mol.biol = factor(ALL_bcrneg$mol.biol)
> ALL_bcrneg$BT = factor(ALL_bcrneg$BT)
```

1.4 Nonspecific filtering

Some fraction of genes were not expressed at all in the cells that were assayed, at least not to a level that we could detect with the microarrays used here. For further genes, the data did not show enough variation to allow any reliable detection of differential expression. It is a good idea to remove probe sets for these genes before further analysis, and we can do that on the basis of variance. You should consult Chapter 6 and Chapter 7 for more details on nonspecific filtering. Here, we show how to proceed using the function `nsFilter` from the **genefilter** package to filter for a number of different criteria, all controlled by the function's various parameters. Setting `feature.exclude="^AFFX"` removes the control probes, which can be identified by the prefix `AFFX` in their name. As a measure of dispersion for the variance filtering step, we use the interquartile range (IQR), and we choose the 0.5 quantile of the IQR values as a cutoff. This appears to be a reasonable value for the biological setting and the microarray design used here, but it is likely that you may need to adjust this value to your experiment. In Chapter 6 we give an example how to derive a more data-driven cutoff value.

```
> varCut = 0.5
> filt_bcrneg = nsFilter(ALL_bcrneg, require.entrez=TRUE,
      require.GOBP=TRUE, remove.dupEntrez=TRUE,
      var.func=IQR, var.cutoff=varCut,
      feature.exclude="^AFFX")
> filt_bcrneg$filter.log
$numDupsRemoved
[1] 968

$numLowVar
[1] 5212

$feature.exclude
[1] 19
```

```
$numNoGO.BP
[1] 1779

$numRemoved.ENTREZID
[1] 404
> ALLfilt_bcrneg = filt_bcrneg$eset
```

1.5 BCR/ABL ALL1/AF4 subset

There are also other subsets of the data in which we might be interested. The following code produces a subset consisting of samples from BCR/ABL positive tumors harboring the t9;22 translocation and ALL1/AF4 positive tumors with t4;11 translocations.

```
> types = c("ALL1/AF4", "BCR/ABL")
> moltyp = which(ALL$mol.biol %in% types)
> ALL_af4bcr = ALL[, intersect(bcell, moltyp)]
> ALL_af4bcr$mol.biol = factor(ALL_af4bcr$mol.biol)
> ALL_af4bcr$BT = factor(ALL_af4bcr$BT)
> filt_af4bcr = nsFilter(ALL_af4bcr,require.entrez=TRUE,
      require.GOBP=TRUE, remove.dupEntrez=TRUE,
      var.func=IQR, var.cutoff=varCut)
> ALLfilt_af4bcr = filt_af4bcr$eset
```

Note that nsFilter's default choice of variance measure is not really appropriate for this case because the sizes of the sample groups differ quite a lot. See Chapter 8 for a more thourough discussion.

2

R and Bioconductor Introduction

R. Gentleman, F. Hahne, S. Falcon, and M. Morgan

Abstract

In this chapter we cover basic uses of R and begin working with Bioconductor datasets and tools. Topics covered include simple R programming, R graphics, and working with *environments* as hash tables. We introduce the *ExpressionSet* class as an example for a basic Bioconductor structure used for holding genomic data, in this case expression microarray data. And we explore some visualization techniques for gene expression data to get a feeling for R's extensive graphical capabilities.

2.1 Finding help in R

To get started with R and Bioconductor it is important to know where you can find help for the numerous functions, classes, and concepts you are about to come across. The ? operator is the most immediate source of information about R objects. Preceding the name of a function with ? quickly gets you to the manual page of this function. Possible arguments and return values should be introduced there, and you will find basic information about the purpose and application of the function. A special flavor of ? exists for classes. `class ? foo` will get you to the manual page of class `foo` where you will often also find information about available methods for this class.

Function apropos can be used to find objects in the search path partially matching the given character string. find also locates objects, yet in a more restrictive manner.

F. Hahne et al., *Bioconductor Case Studies*, DOI: 10.1007/978-0-387-77240-0_2,