Lab: Using R and Bioconductor

Robert Gentleman
Wolfgang Huber
Paul Murrell

June 7, 2004

Introduction

In this lab we will cover some basic uses of R and also begin working with some of the Bioconductor data sets and tools. Topics covered include basic use of R, R graphics, working with environments as hash tables.

Some Basic R

First load the *Biobase* package and then the data set `eset`.

```
> library("Biobase")

Welcome to Bioconductor
         Vignettes contain introductory material.  To view,
         simply type: openVignette()
         For details on reading vignettes, see
         the openVignette help page.

> data(eset)
> eset

Expression Set (exprSet) with
        500 genes
        26 samples
                 phenoData object with 3 variables and 26 cases
         varLabels
                cov1: Covariate 1; 2 levels
                cov2: Covariate 2; 2 levels
                cov3: Covariate 3; 3 levels
```

The expression set is an S4 class and `eset` is an instance of this class. You can get help (a description of the class) by using the `?` operator; `class?exprSet`.

```
> class(eset)

[1] "exprSet"

> slotNames(eset)

[1] "exprs"       "se.exprs"    "phenoData"   "description" "annotation"
[6] "notes"

> eset$cov1

 [1] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2

> eset[1, ]

Expression Set (exprSet) with
        1 genes
        26 samples
                 phenoData object with 3 variables and 26 cases
         varLabels
                cov1: Covariate 1; 2 levels
                cov2: Covariate 2; 2 levels
                cov3: Covariate 3; 3 levels

> eset[, 1]

Expression Set (exprSet) with
        500 genes
        1 samples
                 phenoData object with 3 variables and 1 cases
         varLabels
                cov1: Covariate 1; 2 levels
                cov2: Covariate 2; 2 levels
                cov3: Covariate 3; 3 levels
```

You can extract the values in the slots using the `@` operator, or in many cases accessor functions are available. The names of the slots can be obtained using *slotNames*, as shown above. Extract the values for some of the named slots.

Exercise 1
What happens when we subset `eset`*? What kind of an object do we get? What happened to the phenotypic data? What happened to the expression data? Subset* `eset` *by selecting all elements for which* `cov1` *has value* `1`*.*

Environments

In R an `environment` is a set of symbol-value pairs. These are very similar to lists, but there is no natural ordering of values and you cannot make use of numeric indices. Otherwise they behave the same way.
We first create an environment and then add, remove, list etc.

```
> e1 = new.env(hash = TRUE)
> e1$a = rnorm(10)
> e1$b = runif(20)
> ls(e1)

[1] "a" "b"

> xx = as.list(e1)
> names(xx)

[1] "a" "b"
```

Exercise 2

- *Create an environment and put a copy of* `eset` *into it.*
- *Fit a linear model to the data* `x=1:10`, `y=2*x`*rnorm(10, sd=0.25)+, and also place this into your environment.*
- *Write a function, myExtract, that takes an environment as an argument and returns a list, one element is the variable* `cov2` *from* `eset` *and the other is the vector of coefficients from the linear model.*

Something Harder

Later we will spend some time discussing machine learning (ML), but here we will just use one simple algorithm, k-nearest neighbors (knn) to make predictions. You should read the R manual page for a description of knn.

```
> library("class")
> apropos("knn")

[1] "knn"    "knn.cv" "knn1"
```

The knn algorithm predicts the class of a given observation (the test case) according to a majority vote of the k nearest neighbors in the training set. We will show how you can use this to predict the class of sample 1, given data on samples 2 through 26.

```
> exprsEset = exprs(eset)
> classEset = eset$cov2
> esub = eset[, -1]
> pred1 = knn(t(exprs(esub)), exprs(eset)[, 1], esub$cov2)
> classEset[1]
```

```
[1] 1
```

Exercise 3

- *Write a function, that takes an exprSet as its input and carries out a leave-one-out set of predictions.*
- *Your function should return the vector of predicted values for the given covariate.*
- *Modify your function to allow the user to specify some of the parameters for knn, such as k.*

The `apply` functions

In R a great deal of work is done by applying some function to all elements of a list, matrix or array. There are several functions available for you to use, *apply*, *lapply*, *sapply* are the most commonly used. From the next release of R onwards there will also be an *eapply* for use with environments.
To get some understanding of the apply functions we will attempt to extract some information from the Gene Ontology information that is supplied with each data package.
This next code chunk shows how to use apply to extract all the molecular function GO terms for each Affymetrix probe set.

```
> library("GO")
> library("hgu95av2")
> affyGO = as.list(hgu95av2GO)
> affyMF = lapply(affyGO, function(x) {
+     onts = sapply(x, function(z) z$Ontology)
+     if (is.null(onts) || is.na(onts))
+         NA
+     else unique(names(onts)[onts == "MF"])
+ })
```

Exercise 4

- *How are the GO terms stored? What information is available for each?*
- *What are the evidence codes and what do they mean?*
- *Turn this code into a function that would allow users to obtain either the MF, BP or CC data.*
- *Extend this function to allow the user to include only given evidence codes. (Or if you think it better - to exclude specific codes).*

Graphics

In this section you will work through some examples that allow you to create very general plots in R.
Given the following data produce a plot that looks like the one in Figure 1. The relevant features are the tick marks on the y-axis and the vertical positioning of the data symbols.

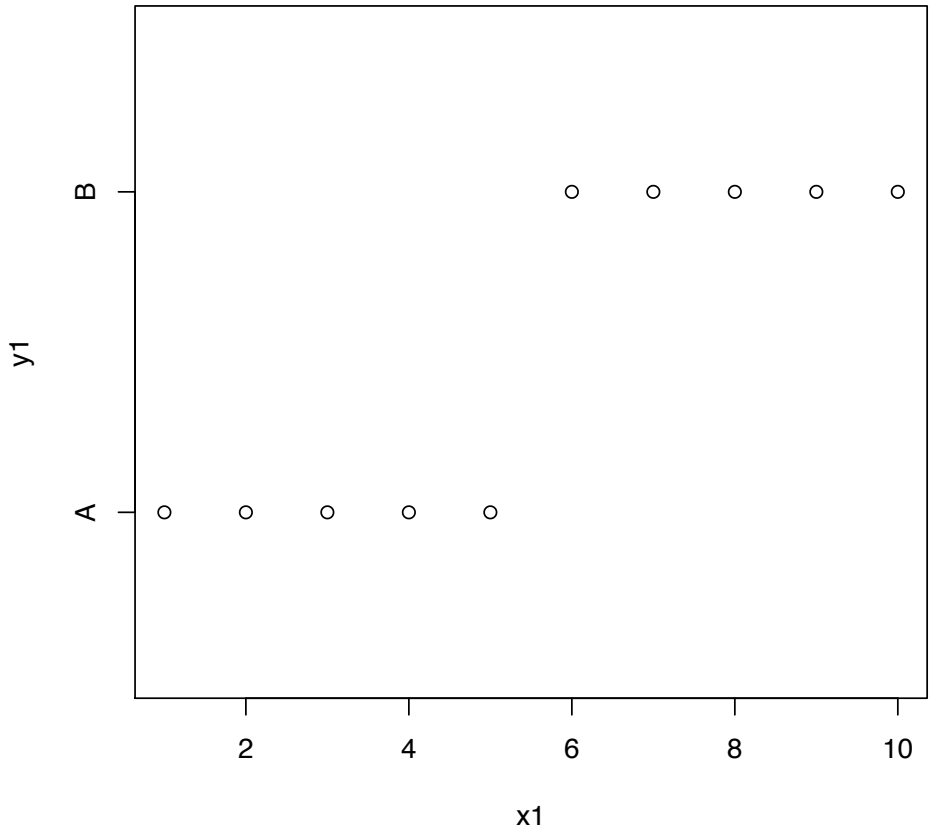

Figure 1: Figure for Graphics Question 1.

The following data represent a value recorded regularly over time (e.g., television viewing gures). The variable `v` contains the raw values, and `mean.8` contains "moving average" values (for week i, the moving average is average of weeks $i-7, i-6, \ldots, i$).

```
> v <- rnorm(20) + 4
> mean.8 <- rep(0, length(v) - 7)
> for (i in 1:length(mean.8)) mean.8[i] <- mean(v[i:(i + 7)])
```

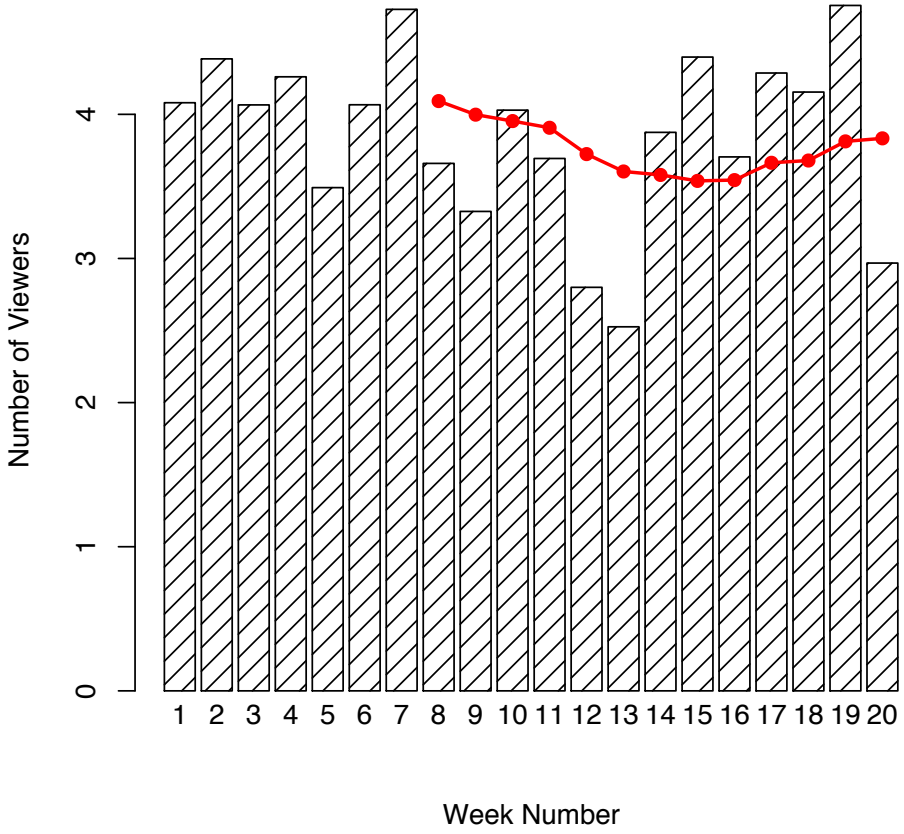

Figure 2: Figure for Graphics Question 2.

Now, let us try for something that is related to bioinformatics. We first find out which genes are located in which chromosomes.

```
> whCHR = unlist(mget(geneNames(eset), hgu95av2CHR))
> table(whCHR)

whCHR
 1 10 11 12 13 14 15 16 17 18 19  2 20 21 22  3  4  5  6  7  8  9  X  Y
34 14 18 32 12 14  7 13 18  4 22 27  4  4 13 15 16 10 33 21 22 12 20  7

> max(table(whCHR))

[1] 34
```

We can see how many genes from each chromosome are included in our data set. We want to plot these data, basically creating plots similar to those in *geneplotter*, such as *alongChrom*.

```
> library("geneplotter")
```

Exercise 5

- *Select a chromosome (any one) to produce your plot.*
- *Find out the length of this chromosome (in bases). [Hint: the necessary data is in hgu95av2.]*
- *Find the position for each gene, on your selected chromosome. [Hint:* `hgu95av2CHRLOC`*]*
- *Create a plot with a single horizontal line and add a tick mark for each gene (perpendicular to the horizontal line).*
- *Can you color the tick marks according to gene expression?*