Lecture 6: More on R and Affymetrix Arrays

Kevin R. Coombes

18 September 2007

1 Beyond Matrices

We have gone from scalar to vector to matrix, attaching names as we go, with the goal of keeping associated information together. So far, we've done this with numbers, but we could use character strings instead:

```
> letters[1:3]

[1] "a" "b" "c"

> x <- letters[1]
> x <- letters[1:3]
> x <- matrix(letters[1:12], 3, 4)
```

1.1 Mixing Modes in Lists

In R, we cannot easily mix data of different modes in a vector or matrix:

```
> x <- c(1, "a")
> x

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

However, a list can have (named) components that are of different modes and even different sizes:

```
> x <- list(teacher = "Keith", n.students = 14,
+     grades = letters[c(1:4, 6)])
> x

$teacher
[1] "Keith"

$n.students
[1] 14

$grades
[1] "a" "b" "c" "d" "f"
```

Note that we named the components of the list at the same time that we created it. Many functions in R return answers as lists.

1.2 Extracting Items From Lists

If we want to access the first element of x, we might try using the index or the name in single brackets:

```
> x[1]

$teacher
[1] "Keith"

> x["teacher"]

$teacher
[1] "Keith"
```

These don't quite work. The single bracket extracts a component, but keeps the same mode; what we have here is a list of length 1 as opposed to a character string. Two brackets, on the other hand...

```
> x[[1]]

[1] "Keith"

> x[["teacher"]]

[1] "Keith"
```

The double bracket notation can be cumbersome, so there is a shorthand notation with the dollar sign. Using names keeps the goals clear.

```
> x$teacher

[1] "Keith"
```

1.3 Lists with Structure

Now, there are some very common types of structured arrays. The most common is simply a table, where the rows correspond to individuals and the columns correspond to various types of information (potentially of multiple modes). Because we want to allow for multiple modes, we can construct a table as a list, but this list has a constraint imposed on it – all of its components must be of the same length. This is similar in structure to the idea of a matrix that allows for multiple modes. This structure is built into R as a `data frame`.

This structure is important for data import. Before loking at that, however, we are going to revisit the notion of reproducibility of our analyses.

2 The Reproducibility Problem

1. Researcher contacts analyst: "I just read this interesting paper. Can you perform the same analysis on my data?"
2. Analyst reads paper. Finds algorithms described by biologists in English sentences that occupy minimal amount of space in the methods section.
3. Analyst gets public data from the paper. Takes wild guesses at actual algorithms and parameters. Is unable to reproduce reported results.
4. Analyst considers switching to career like bicycle repair, where reproducibility is less of an issue.

2.1 Alternate Forms of the Same Problem

1. Remember that microarray analysis you did six months ago? We ran a few more arrays. Can you add them to the project and repeat the same analysis?
2. The statistical analyst who looked at the data I generated previously is no longer available. Can you get someone else to analyze my new data set using the same methods (and thus producing a report I can expect to understand)?
3. Please write/edit the methods sections for the abstract/paper/grant proposal I am submitting based on the analysis you did several months ago.

2.2 The Code/Documentation Mismatch

Most of our analyses are performed using R. We can usually find an R workspace in a directory containing the raw data, the report, and one or more R scripts.

There is no guarantee that the objects in the R workspace were actually produced by those R scripts. Nor that the report matches the code. Nor the R objects.

Because R is interactive, unknown commands could have been typed at the command line, or the commands in the script could have been cut-n-pasted in a different order.

This problem is even worse if the software used for the analysis has a fancy modern GUI. It is impossible to document how you used the GUI in such a way that someone else could produce the exact same results—on the same data—six months later.

2.3 The Solution: Sweave

$$\text{Sweave} = \text{R} + \text{LaTeX}.$$

This talk was prepared using Sweave. So was this standard report.

If you already know both R and LaTeX, then the ten-second version of this talk takes only two slides:

1. Prepare a LaTeX document. Give it an "Rnw" extension instead of "tex". Say it is called "myfile.Rnw"
2. Insert an R code chunk starting with <<>>=
3. Terminate the R code chunk with an "at" sign (@) followed by a space.

2.4 Using Sweave

To produce the final document

1. In an R session, issue the command

   ```
   Sweave("myfile.Rnw")
   ```

   This executes the R code, inserts input commands and output computations and figures into a LaTeX file called "myfile.tex".

2. In the UNIX or DOS window (or using your favorite graphical interface), issue the command

   ```
   pdflatex myfile
   ```

   This produces a PDF file that you can use as you wish.

2.5 Viewing The Results

Here is a simple example, showing how the R input commands can generate output that is automatically included in the LaTeX output of Sweave.

```
> x <- rnorm(30)
> y <- rnorm(30)
> mean(x)

[1] -0.04602806

> cor(x, y)

[1] 0.09811465
```

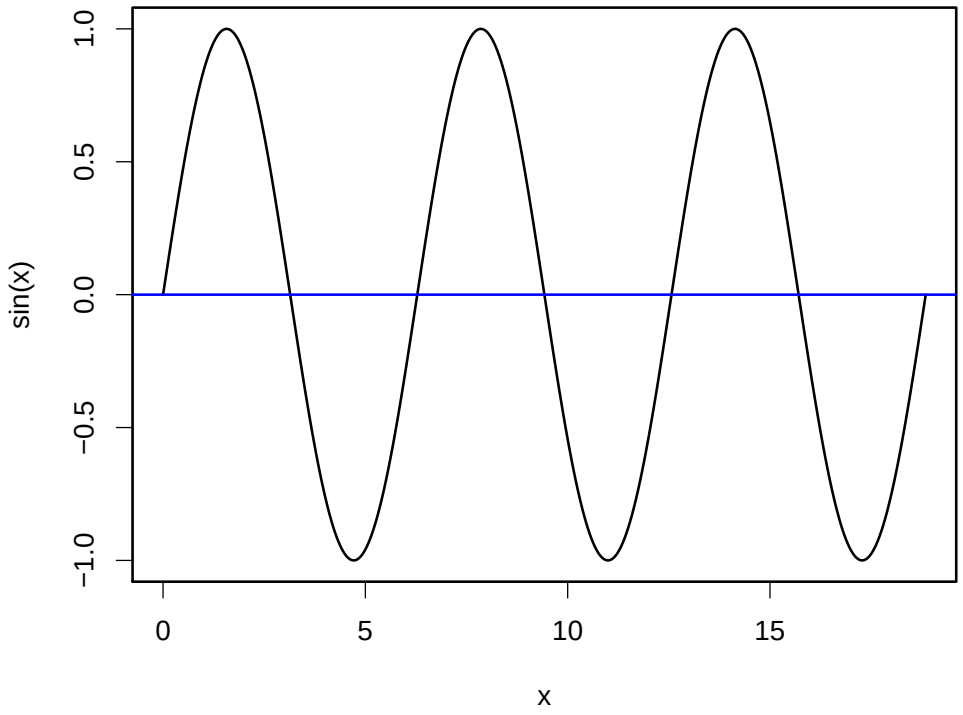
2.6 A Figure

Next, we are going to insert a figure. First, we can look at the R commands that are used to produce the figure.

```
> x <- seq(0, 6 * pi, length = 450)
> par(bg = "white", lwd = 2, cex = 1.3, mai = c(1.2,
+     1.2, 0.2, 0.2))
> plot(x, sin(x), type = "l")
> abline(h = 0, col = "blue")
```

On the next slide, we can look at the actual figure. (Part of the point of this example is to illustrate that you can separate the input from the output. You can even completely hide the input in the source file and just include the output in the report.)

2.7 Sine Curve



2.8 A Table

```
> library(xtable)
> x <- data.frame(matrix(rnorm(12), nrow = 3,
+     ncol = 4))
> dimnames(x) <- list(c("A", "B", "C"), c("C1",
+     "C2", "C3", "C4"))
> tab <- xtable(x, digits = c(0, 3, 3, 3, 3))
> tab
```

	C1	C2	C3	C4
A	1.366	−0.303	−0.128	0.637
B	0.188	0.036	−2.038	−1.363
C	−0.324	−0.520	−1.488	0.897

2.9 A Table, Repeated

Again, we want to point out that you can show the results—including tables—without showing the commands that generate them.

	C1	C2	C3	C4
A	1.366	−0.303	−0.128	0.637
B	0.188	0.036	−2.038	−1.363
C	−0.324	−0.520	−1.488	0.897