7. R Data Structures

7.1 Vectors
Recall that vectors may have mode logical, numeric or character.

7.1.1 Subsets of Vectors
Recall (section 2.6.2) two common ways to extract subsets of vectors:

Specify the numbers of the elements that are to be extracted. One can use negative numbers to omit elements.

Specify a vector of logical values. The elements that are extracted are those for which the logical value is T.

Thus suppose we want to extract values of x that are greater than 10.

The following demonstrates a third possibility, for vectors that have named elements:

> c(Andreas=178, John=185, Jeff=183)[c("John","Jeff")]
> John
> 185
> Jeff
> 183

A vector of names has been used to extract the elements.

7.1.2 Patterned Data
Use 5:15 to generate the numbers 5, 6, ..., 15. Entering 15:5 will generate the sequence in the reverse order.

To repeat the sequence (2, 3, 5) four times over, enter rep(c(2,3,5), 4) thus:

> rep(c(2,3,5), 4)
> [1] 2 3 5 2 3 5 2 3 5 2 3 5

If instead one wants four 2s, then four 3s, then four 5s, enter rep(c(2,3,5), c(4,4,4)).

Note further that, in place of c(4,4,4) we could write rep(4,3). So a further possibility is that in place of rep(2,3,5), c(4,4,4) we could enter rep(c(2,3,5), rep(4,3)).

In addition to the above, note that the function rep() has an argument length.out, meaning “keep on repeating the sequence until the length is length.out.”

7.2 Missing Values
In R, the missing value symbol is NA. Any arithmetic operation or relation that involves NA generates an NA.

This applies also to the relations <, <=, >, >=, ==, !=. The first four compare magnitudes, == tests for equality, and != tests for inequality. Users who do not carefully consider implications for expressions that include NAs may be puzzled by the results. Specifically, note that x=NA generates NA.

Be sure to use is.na(x) to test which values of x are NA. As x=NA gives a vector of NAs, you get no information at all about x. For example:

> x <- c(1,2,3,NA)
> is.na(x) # TRUE for when NA appears, and otherwise FALSE
> [1] FALSE FALSE FALSE TRUE
> x==NA # All elements are set to NA
> [1] NA NA NA NA
> NA==NA
[1] NA

WARNING: This is chiefly for those who may move between R and S-PLUS. In important respects, R’s behaviour with missing values is more intuitive than that of S-PLUS. Thus in R

y[x>2] <- x[x>2]
gives the result that the naive user might expect, i.e. replace elements of y with corresponding elements of x wherever x>2. Wherever x≤2 gives the result NA, no action is taken. In R, any NA in x[x>2] yields a value of NA for y[x>2] on the left of the equation, and a value of NA for y[x>2] on the right of the equation.

In S-PLUS, the result on the right is the same, i.e. an NA. However, on the left, elements that have a subscript NA drop out. The vector on the left to which values will be assigned has, as a result, fewer elements than the vector on the right.

Thus the following has the effect in R that the naive user might expect, but not in S-PLUS:

x <- c(1,6,2,NA,10)
y <- c(1,4,2,3,0)
y[x>2] <- x[x>2]

In S-PLUS it is essential to specify, in the example just considered:

y[s>2] <- x[s>2]

Here is a further example of R’s behaviour:

> x <- c(1,6,2,NA,10)
> x>2
> [1] FALSE TRUE FALSE NA TRUE
> x[x>2] # Now, explain the result that follows
> [1] 1 2 3 2 NA 21

The safe way, in both S-PLUS and R, is to use !is.na(x) to limit the selection, on one or both sides as necessary, to those elements of x that are not NAs. We will have more to say on missing values in the section on data frames that now follows.

7.3 Data frames
The concept of a data frame is fundamental to the use of most of the R modelling and graphics functions. A data frame is a generalisation of a matrix, in which different columns may have different modes. All elements of any column must however have the same mode, i.e. all numeric or all factor, or all character.

Data frames where all columns hold numeric data have some, but not all, of the properties of matrices. There are important differences that arise because data frames are implemented as lists. To turn a data frame of numeric data into a matrix of numeric data, use as.matrix().

Lists are discussed below, in section 7.6.

7.3.1 Extraction of Component Parts of Data frames
Consider the data frame barley that accompanies the lattice package:

> names(barley)
> [1] "yield" "variety" "year" "site"
> levels(barley$site)
> [1] "Grand Rapids" "Duluth" "University Farm" "Morris"
> [5] "Crookston" "Waseca"

We will extract the data for 1932, at the Duluth site.

> barley[barley$year==1932 & barley$site=="Duluth",]
> [1] variety yield
> 66 Manchuria 22.56667
> 72 Glabron 25.86667
> 78 Swansota 22.23333
> 84 Velvet 22.46667
> 90 Trebi 30.60000

> barley$site...
The function `read.table()` offers a ready means to read a rectangular array into an R data frame. Suppose that the file `primates.dat` contains:

```
Species Bodywt Brainwt
Potar monkey 10 115
Gorilla 207.0 406
Human 62.0 1320
Rhesus monkey 6.8 179
Chimp 52.2 440
```

Then

```
primates <- read.table("a:/primates.dat")
```

will create the data frame `primates`. From a file on the `a` drive. The text strings in the first column will become the first column in the data frame.

Suppose that `primates` is a data frame with three columns – species name, body weight, and brain weight. You can give the column names by typing in:

```
names(primates) <- c("Species","Bodywt","Brainwt")
```

Here then are the contents of the data frame.

```
Species   Bodywt Brainwt
Potar monkey 10.0 115
Gorilla 207.0 406
Human 62.0 1320
Rhesus monkey 6.8 179
Chimp 52.2 440
```

Specify `header=TRUE` if there is an initial header of question. If the number of headers is one less than the number of columns of data, then the first column will be used, providing entries are unique, for row labels.

7.4.1 Idiosyncrasies

The function `read.table()` is straightforward for reading in rectangular arrays of data that are entirely numeric. When, as in the above example, one of the columns contains text strings, the column is by default stored as a factor with as many different levels as there are unique text strings.\(^15\)

Problems may arise when small mistakes in the data cause R to interpret a column of supposedly numeric data as character strings, which are automatically turned into factors. For example there may be an O (oh) somewhere where there should be a 0 (zero), or an el (l) where there should be a 1 (I). If you use any missing value symbols other than the default (NA) you need to make this explicit see section 7.3.2 below. Otherwise any appearance of such symbols as *, period(.) and blank (in a case where the separator is something other than a space) will cause to whole column to be treated as character data.

Users who find this default behaviour of `read.table()` confusing may wish to use the parameter setting `as.is = TRUE`.\(^17\) If the column is later required for use as a factor in a model or graphics formula, it may be necessary to make it into a factor at that time. Some functions do this conversion automatically.

7.4.2 Missing values when using `read.table()`

The function `read.table()` expects missing values to be coded as `NA`, unless you set `na.strings` to recognise other characters as missing value indicators. If you have a text file that has been output from SAS, you will probably want to set `na.strings=c("", ",", ",", ",")`. The `""` will ensure that empty cells are entered as `NA`.

7.4.3 Separators when using `read.table()`

With data from spreadsheets\(^11\), it is sometimes necessary to use `tab("\t")` or `comma` as the separator. The default separator is white space. To set tab as the separator, specify `sep="\t"`.

7.5 Factors and Ordered Factors

Factors are vectors which have mode numeric and class `factor`. They have an attribute levels that holds the level names.

Factors are vectors which have mode numeric and class `factor`. They have an attribute levels that holds the level names.

---

\(^{15}\) Storage of columns of character strings as factors is efficient when a small number of distinct strings that are of modest length are each repeated a large number of times.

\(^{16}\) Specifying `as.is = T` prevents columns of (intended or unintended) character strings from being converted into factors.

\(^{17}\) One way to get mixed text and numeric data across from Excel is to save the worksheet in a .csv file with comma as the separator. If for example file name is `myfile.csv` and is on drive a: use `read.table("a:/myfile.csv", sep="",`) to read the data into R. This exposes with any spaces which may appear in text strings. [But watch that none of the cell entries include commas.]
7.7 Lists

Lists make it possible to collect an arbitrary set of R objects together under a single name. You might for example collect together vectors of several different modes and lengths, scalars, matrices or more general arrays, functions, etc. Lists can be, and often are, a rag-tag of different objects. We will use for illustration the list object that R creates as output from an lm calculation.

For example, consider the linear model (lm) object elastic.lm (c.f. sections 1.1.4 and 2.1.4) created thus:

```r
elastic.lm <- lm(distance~stretch, data=elasticband)
```

It is readily verified that `elastic.lm` consists of a variety of different kinds of objects, stored as a list. You can get the names of these objects by typing in:

```r
names(elastic.lm)
```

The first list element is:

```r
elastic.lm$coefficients
```

The second list element is a vector of length 7 which consists of a variety of different kinds of objects, stored as a list. You can get the names of these objects by typing in:

```r
names(elastic.lm)$coefficients
```

The third list element is:

```r
elastic.lm$call
```

Alternative ways to extract this first list element are:

```r
elastic.lm$["coefficients"]
```

We can alternatively ask for the sublist whose only element is the vector `elastic.lm$coefficients`. For this, specify `elastic.lm$["coefficients"]` or `elastic.lm[[1]]`. There is a subtle difference in the result that is printed out. The information is preceded by `Scoefficients`, meaning “list element with name coefficients".

```r
elastic.lm[[1]]
```

The second list element is a vector of length 7:

```r
elastic.lm$call
```

The third list element documents the function call:

```r
elastic.lm$call
```

7.8 Matrices and Arrays

All elements of a matrix have the same mode, i.e. all numeric, or all character. Thus a matrix is a more restricted structure than a data frame. One reason for numeric matrices is that they allow a variety of mathematical operations that are not available for data frames. Matrices are likely to be important for those users who wish to implement new regression and multivariate methods. The matrix construct generalises to array, which may have more than two dimensions.

Note that matrices are stored columnwise. Thus consider:

```r
xx <- matrix(1:6, ncol=3) # Equivalently, enter matrix(1:6, nrow=2)
xx
```

```r
[,1] [,2] [,3]
[1,]  1  3  5
[2,]  2  4  6
```
If \( xx \) is any matrix, the assignment
\[
x \leftarrow \text{as.vector}(xx)
\]
places columns of \( xx \), in order, into the vector \( x \). In the example above, we get back the elements 1, 2, …, 6.

Matrices have the attribute “dimension”. Thus
\[
> \text{dim}(xx)
\]
\[
[1] 2 3
\]
In fact a matrix is a vector (numeric or character) whose dimension attribute has length 2.

Now set
\[
> x34 \leftarrow \text{matrix}(1:12, \text{ncol}=4)
\]
\[
[1,] 1 4 7 10 
[2,] 2 5 8 11 
[3,] 3 6 9 12
\]
Here are examples of the extraction of columns or rows or submatrices
\[
x34[2:3,c(1,4)] \# Extract rows 2 & 3 & columns 1 & 4 
\]
\[
x34[,2] \# Extract the second row
\]
\[
x34[-2,] \# Extract all rows except the second
\]
\[
x34[-2,-3] \# Extract the matrix obtained by omitting row 2 & column 3
\]
The \texttt{dimnames()} function assigns and/or extracts matrix row and column names. The \texttt{dimnames()} function gives a list, in which the first list element is the vector of row names, and the second list element is the vector of column names. This generalises in the obvious way for use with arrays, which we now discuss.

### 7.8.1 Arrays

The generalisation from a matrix (2 dimensions) to allow > 2 dimensions gives an array. A matrix is a 2-dimensional array.

Consider a numeric vector of length 24. So that we can easily keep track of the elements, we will make them 1, 2, …, 24. Thus
\[
x \leftarrow 1:24
\]
Then
\[
> \text{dim}(x) \leftarrow c(2,12)
\]
turns this into a 2 x 12 matrix.
\[
[1,] 1 4 7 10 13 16 19 22 25 28 31 34 
[2,] 2 5 8 11 14 17 20 23 26 29 32 35
\]
Now try
\[
> \text{dim}(x) \leftarrow c(3,4,2)
\]
\[
[1,] 1 4 7 10 
[2,] 2 5 8 11 
[3,] 3 6 9 12
\]
\[
[1,] 13 16 19 22 
[2,] 14 17 20 23
\]

### 7.8.2 Conversion of Numeric Data frames into Matrices

There are various manipulations that are available for matrices, but not for data frames. Use \texttt{as.matrix()} to handle any conversion that may be necessary.

### 7.9 Exercises

Generate the numbers 101, 102, …, 112, and store the result in the vector \( x \).
Generate four repeats of the sequence of numbers (4, 6, 3).
Generate the sequence consisting of eight 4s, then seven 6s, and finally nine 3s. Store the numbers obtained, in order, in the columns of a 6 by 4 matrix.
Create a vector consisting of one 1, then two 2’s, three 3’s, etc., and ending with nine 9’s.
For each of the following calculations, what you would expect? Check to see if you were right!
\[
a) \quad \text{answer} \leftarrow c(2, 7, 1, 5, 12, 3, 4)
\]
\[
> \text{for (j in 2:length(answer))} \{ \text{answer}\[j\] \leftarrow \max(\text{answer}\[j\], \text{answer}\[j-1\]) \}
\]
\[
b) \quad \text{answer} \leftarrow c(2, 7, 1, 5, 12, 3, 4)
\]
\[
> \text{for (j in 2:length(answer))} \{ \text{answer}\[j\] \leftarrow \text{sum(\text{answer}\[j\], \text{answer}\[j-1\])} \}
\]
In the built-in data frame \texttt{airquality} (datasets package): (a) Determine, for each of the columns of the data frame \texttt{airquality} (datasets package), the median, mean, upper and lower quartiles, and range; (b) Extract the row or rows for which \texttt{Ozone} has its maximum value; (c) Extract the vector of values of \texttt{Wind} for values of \texttt{Ozone} that are above the upper quartile.
For the Eurasian snow data that is given in Exercise 1.6. Find the mean of the snow cover (a) for the odd-numbered years and (b) for the even-numbered years.
Determine which columns of the data frame \texttt{Cars93} (MASS package) are factors. For each of these factor columns, print out the levels vector. Which of these are ordered factors?
Use \texttt{summary()} to get information about data in the data frames \texttt{attitude} (both in the datasets package), and \texttt{cpus} (MASS package). Write brief notes, for each of these data sets, on what this reveals.
From the data frame \texttt{mtcars} (MASS package) extract a data frame \texttt{mtcars6} that holds only the information for cars with 6 cylinders.
From the data frame \texttt{Cars93} (MASS package), extract a data frame which holds only information for small and sporty cars.