The idea is to create a new dataframe with a separate row for each case. That is to say, we want 12 copies of the first row (for healthy young males), seven copies of the second row (for healthy old males), and so on. The trick is to use `lapply` to apply the repeat function `rep` to each variable in `count.table` such that each row is repeated by the number of times specified in the vector called count:

```
lapply(count.table, function(x) rep(x, count.table$count))
```

Then we convert this object from a list to a dataframe using `as.data.frame` like this:

```
dbtable <- as.data.frame(lapply(count.table, function(x) rep(x, count.table$count)))
```

To tidy up, we probably want to remove the redundant vector of counts:

```
dbtable <- dbtable[-1]
```

```
dbtable

sex   age condition
1 male young healthy
2 male young healthy
3 male young healthy
4 male young healthy
5 male young healthy
6 male young healthy
7 male young healthy
8 male young healthy
9 male young healthy
10 male young healthy
11 male young healthy
12 male young healthy
13 male old healthy
14 male old healthy
15 male old healthy
16 male old healthy
... 60 female old parasitized
61 female old parasitized
62 female old parasitized
```

Now we can use the contents of `dbtable` as explanatory variables in modelling other responses of each of the 62 cases (e.g. the animals’ body weights).

### Converting from a Dataframe to a Table

The reverse procedure of creating a table from a dataframe is much more straightforward, and involves nothing more than the `table` function:

```
table(dbtable)

, , condition = healthy

   age
sex  old young
female 8 9
   male 7 12
```
The R Book

condition = parasitized
age
sex   old  young
female 5    8
male   7    6

You might want this tabulated object itself to be another dataframe, in which case use

as.data.frame(table(dbtable))

<table>
<thead>
<tr>
<th>sex</th>
<th>age</th>
<th>condition</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>female</td>
<td>old</td>
<td>healthy</td>
<td>8</td>
</tr>
<tr>
<td>male</td>
<td>old</td>
<td>healthy</td>
<td>7</td>
</tr>
<tr>
<td>female</td>
<td>young</td>
<td>healthy</td>
<td>9</td>
</tr>
<tr>
<td>male</td>
<td>young</td>
<td>healthy</td>
<td>12</td>
</tr>
<tr>
<td>female</td>
<td>old</td>
<td>parasitized</td>
<td>5</td>
</tr>
<tr>
<td>male</td>
<td>old</td>
<td>parasitized</td>
<td>7</td>
</tr>
<tr>
<td>female</td>
<td>young</td>
<td>parasitized</td>
<td>8</td>
</tr>
<tr>
<td>male</td>
<td>young</td>
<td>parasitized</td>
<td>6</td>
</tr>
</tbody>
</table>

You will see that R has invented the variable name `Freq` for the counts of the various contingencies. To change this to ‘count’ use `names` with the appropriate subscript [4]:

frame<-as.data.frame(table(dbtable))
names(frame)[4]<-"count"
frame

<table>
<thead>
<tr>
<th>sex</th>
<th>age</th>
<th>condition</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>female</td>
<td>old</td>
<td>healthy</td>
<td>8</td>
</tr>
<tr>
<td>male</td>
<td>old</td>
<td>healthy</td>
<td>7</td>
</tr>
<tr>
<td>female</td>
<td>young</td>
<td>healthy</td>
<td>9</td>
</tr>
<tr>
<td>male</td>
<td>young</td>
<td>healthy</td>
<td>12</td>
</tr>
<tr>
<td>female</td>
<td>old</td>
<td>parasitized</td>
<td>5</td>
</tr>
<tr>
<td>male</td>
<td>old</td>
<td>parasitized</td>
<td>7</td>
</tr>
<tr>
<td>female</td>
<td>young</td>
<td>parasitized</td>
<td>8</td>
</tr>
<tr>
<td>male</td>
<td>young</td>
<td>parasitized</td>
<td>6</td>
</tr>
</tbody>
</table>

Calculating tables of proportions

The `margins` of a table (the row totals or the column totals) are often useful for calculating proportions instead of counts. Here is a data matrix called `counts`:

counts<-matrix(c(2,2,4,3,1,4,2,0,1,5,3,3),nrow=4)
counts

[,1] [ ,2] [ ,3]
[1, ] 2 1 1
[2, ] 2 4 5
[3, ] 4 2 3
[4, ] 3 0 3

The proportions will be different when they are expressed as a fraction of the row totals or as a fraction of the column totals. You need to remember that the row subscripts come first, which is why margin number 1 refers to the row totals: