7.1: Tabulating and Cross-tabulating Data

A very common task when analysing data is the construction of frequency tables, or cross-tabulation of one variable against another. There are several functions that you can use in R for that purpose. In this section I'll illustrate the use of three functions – `table()`, `xtabs()` and `tabulate()` – though there are other options (e.g., `ftable()`) available.

7.1.1 Creating tables from vectors

Let's start with a simple example. As the father of a small child, I naturally spend a lot of time watching TV shows like *In the Night Garden*. In the `nightgarden.Rdata` file, I've transcribed a short section of the dialogue. The file contains two variables, `speaker` and `utterance`, and when we take a look at the data, it becomes very clear what happened to my sanity.

```
library(lsr)
load("./rbook-master/data/nightgarden.Rdata" )
who()
```

```
##    -- Name --    -- Class --    -- Size --
##    speaker       character      10
##    utterance     character      10
```

```
print( speaker )
```

```
##  [1] "upsy-daisy"  "upsy-daisy"  "upsy-daisy"  "upsy-daisy"  "tombliboo"
##  [6] "tombliboo"   "makka-pakka" "makka-pakka" "makka-pakka" "makka-pakka"
```

```
print( utterance )
```

```
##  [1] "pip" "pip" "onk" "onk" "ee"  "oo"  "pip" "pip" "onk" "onk"
```

With these as my data, one task I might find myself needing to do is construct a frequency count of the number of words each character speaks during the show. The `table()` function provides a simple way do to this. The basic usage of the `table()` function is as follows:

```
table(speaker)
```

```
## speaker
## makka-pakka   tombliboo  upsy-daisy
##           4           2           4
```

The output here tells us on the first line that what we're looking at is a tabulation of the `speaker` variable. On the second line it lists all the different speakers that exist in the data, and on the third line it tells you how many times that speaker appears in the data. In other words, it's a frequency table[104] Notice that in the command above I didn't name the argument, since `table()` is another function that makes use of unnamed arguments. You just type in a list of the variables that you want R to tabulate, and it tabulates them. For instance, if I type in the name of two variables, what I get as the output is a cross-tabulation:

```
table(speaker, utterance)
```

```
##             utterance
## speaker       ee onk oo pip
##   makka-pakka  0   2  0   2
##   tombliboo    1   0  1   0
##   upsy-daisy   0   2  0   2
```

When interpreting this table, remember that these are counts: so the fact that the first row and second column corresponds to a value of 2 indicates that Makka-Pakka (row 1) says "onk" (column 2) twice in this data set. As you'd expect, you can produce three way or higher order cross tabulations just by adding more objects to the list of inputs. However, I won't discuss that in this section.

7.1.2 Creating tables from data frames

Most of the time your data are stored in a data frame, not kept as separate variables in the workspace. Let's create one:

```
itng <- data.frame( speaker, utterance )
itng
```

```
##        speaker utterance
## 1   upsy-daisy       pip
## 2   upsy-daisy       pip
## 3   upsy-daisy       onk
## 4   upsy-daisy       onk
## 5    tombliboo        ee
## 6    tombliboo        oo
## 7  makka-pakka       pip
## 8  makka-pakka       pip
## 9  makka-pakka       onk
## 10 makka-pakka       onk
```

There's a couple of options under these circumstances. Firstly, if you just want to cross-tabulate all of the variables in the data frame, then it's really easy:

```
table(itng)
```

```
##             utterance
## speaker       ee onk oo pip
##   makka-pakka  0   2  0   2
##   tombliboo    1   0  1   0
##   upsy-daisy   0   2  0   2
```

However, it's often the case that you want to select particular variables from the data frame to tabulate. This is where the `xtabs()` function is useful. In this function, you input a one sided `formula` in order to list all the variables you want to cross-tabulate, and the name of the `data` frame that stores the data:

```
xtabs( formula = ~ speaker + utterance, data = itng )
```

```
##             utterance
## speaker       ee onk oo pip
##   makka-pakka  0   2  0   2
##   tombliboo    1   0  1   0
##   upsy-daisy   0   2  0   2
```

Clearly, this is a totally unnecessary command in the context of the `itng` data frame, but in most situations when you're analysing real data this is actually extremely useful, since your data set will almost certainly contain lots of variables and you'll only want to tabulate a few of them at a time.

7.1.3 Converting a table of counts to a table of proportions

The tabulation commands discussed so far all construct a table of raw frequencies: that is, a count of the total number of cases that satisfy certain conditions. However, often you want your data to be organised in terms of proportions rather than counts. This is where the `prop.table()` function comes in handy. It has two arguments:

- `x` . The frequency table that you want to convert.
- `margin` . Which "dimension" do you want to calculate proportions for. By default, R assumes you want the proportion to be expressed as a fraction of all possible events. See examples for details.

To see how this works:

```
itng.table <- table(itng)  # create the table, and assign it to a variable
itng.table                 # display the table again, as a reminder
```

```
##             utterance
## speaker       ee onk oo pip
##   makka-pakka  0   2  0   2
##   tombliboo    1   0  1   0
##   upsy-daisy   0   2  0   2
```

```
prop.table( x = itng.table ) # express as proportion:
```

```
##             utterance
## speaker       ee onk  oo pip
##   makka-pakka 0.0 0.2 0.0 0.2
##   tombliboo   0.1 0.0 0.1 0.0
##   upsy-daisy  0.0 0.2 0.0 0.2
```

Notice that there were 10 observations in our original data set, so all that R has done here is divide all our raw frequencies by 10. That's a sensible default, but more often you actually want to calculate the proportions separately by row (`margin = 1`) or by column (`margin = 2`). Again, this is most clearly seen by looking at examples:

```
prop.table( x = itng.table, margin = 1)
```

```
##             utterance
## speaker       ee onk  oo pip
##   makka-pakka 0.0 0.5 0.0 0.5
##   tombliboo   0.5 0.0 0.5 0.0
##   upsy-daisy  0.0 0.5 0.0 0.5
```

Notice that each row now sums to 1, but that's not true for each column. What we're looking at here is the proportions of utterances made by each character. In other words, 50% of Makka-Pakka's utterances are "pip", and the other 50% are "onk". Let's contrast this with the following command:

```
prop.table( x = itng.table, margin = 2)
```

```
##             utterance
## speaker        ee onk  oo pip
##   makka-pakka 0.0 0.5 0.0 0.5
##   tombliboo   1.0 0.0 1.0 0.0
##   upsy-daisy  0.0 0.5 0.0 0.5
```

Now the columns all sum to 1 but the rows don't. In this version, what we're seeing is the proportion of characters associated with each utterance. For instance, whenever the utterance "ee" is made (in this data set), 100% of the time it's a Tombliboo saying it.

7.1.4 level tabulation

One final function I want to mention is the `tabulate()` function, since this is actually the low-level function that does most of the hard work. It takes a numeric vector as input, and outputs frequencies as outputs:

```
some.data <- c(1,2,3,1,1,3,1,1,2,8,3,1,2,4,2,3,5,2)
tabulate(some.data)
```

```
## [1] 6 5 4 1 1 0 0 1
```