Some R notes

Kevin Donnelly

1 Setting up data for the chi-squared test

Getting data into R can be done manually in several ways.

You can bind the rows:

```
> ext = rbind(c(2430, 16776), c(2084, 27532))
> ext
```

```
     [,1]  [,2]
[1,] 2430 16776
[2,] 2084 27532
```

or bind the columns:

```
> ext = cbind(c(2430, 2084), c(16776, 27532))
> ext
```

```
     [,1]  [,2]
[1,] 2430 16776
[2,] 2084 27532
```

or read in all the data in one go in column order and segment at the end of every row:

```
> ext <- matrix(c(2430, 2084, 16776, 27532), nrow = 2)
> ext
```

```
     [,1]  [,2]
[1,] 2430 16776
[2,] 2084 27532
```

You can add headings manually:

```
> dimnames(ext) = list(Cognate = c("Trigger", "No trigger"), Codeswitch = c("Switch",
+     "No switch"))
> ext
```

```
            Codeswitch
Cognate      Switch No switch
  Trigger      2430     16776
  No trigger   2084     27532
```

and also sum rows and columns:

```
> addmargins(ext)

            Codeswitch
Cognate      Switch No switch   Sum
  Trigger      2430     16776 19206
  No trigger   2084     27532 29616
  Sum          4514     44308 48822
```

You can get proportions (convert to percent by multiplying by 100):

```
> prop.table(ext, 1)

            Codeswitch
Cognate          Switch No switch
  Trigger    0.12652296 0.8734770
  No trigger 0.07036737 0.9296326
```

A more complex way of getting a labelled table is as follows:

```
> Trigger = c("yes", "no", "yes", "no")
> Switch = c("yes", "yes", "no", "no")
> N = c(2430, 2084, 16776, 27532)
> ext_lab = data.frame(Switch, Trigger, N)
> ext_lab

  Switch Trigger     N
1    yes     yes  2430
2    yes      no  2084
3     no     yes 16776
4     no      no 27532

> my_ext_lab = xtabs(N ~ Switch + Trigger, data = ext_lab)
> my_ext_lab

      Trigger
Switch    no   yes
   no  27532 16776
   yes  2084  2430

> my_ext_lab = xtabs(N ~ Trigger + Switch, data = ext_lab)
> my_ext_lab
```

```
       Switch
Trigger    no   yes
    no  27532  2084
    yes 16776  2430
```

Run chi-squared on these tables (they will, of course, give identical results):

```
> chisq.test(ext)

        Pearson's Chi-squared test with Yates' continuity correction

data:  ext
X-squared = 437.1768, df = 1, p-value < 2.2e-16

> chisq.test(my_ext_lab)

        Pearson's Chi-squared test with Yates' continuity correction

data:  my_ext_lab
X-squared = 437.1768, df = 1, p-value < 2.2e-16
```

The chi-squared statistic is 437.18, with one degree of freedom. The p-value is the probability – if it less than 0.05, it is likely that the result is not due to chance. Here it is 22 with 15 zeros in front of it, ie 0.00000000000000022, which is really tiny.

On 2x2 tables, Yates correction is applied. If you don't want that, use:

```
> chisq.test(ext, correct = F)

        Pearson's Chi-squared test

data:  ext
X-squared = 437.8458, df = 1, p-value < 2.2e-16
```

To get the expected distributions, use:

```
> chisq.test(ext)$expected

            Codeswitch
Cognate        Switch No switch
  Trigger    1775.754  17430.25
  No trigger 2738.246  26877.75
```

You can also get the same thing by putting the results of the chi-squared into a variable. The benefit here is that you don't need to run the test each time you want to access information from it.

```
> results <- chisq.test(ext)
> results$expected
```

```
            Codeswitch
Cognate        Switch No switch
  Trigger    1775.754  17430.25
  No trigger 2738.246  26877.75
```

The test produces more data than is displayed – *expected* is just one of the items available:

```
> names(results)

[1] "statistic" "parameter" "p.value"   "method"    "data.name" "observed"
[7] "expected"  "residuals" "stdres"
```

All the data can be accessed by using the following items:

statistic : the value of the chi-squared test statistic.

parameter : the degrees of freedom of the approximate chi-squared distribution of the test statistic, 'NA' if the p-value is computed by Monte Carlo simulation.

p.value : the p-value for the test.

method : a character string indicating the type of test performed, and whether Monte Carlo simulation or continuity correction was used.

data.name : a character string giving the name(s) of the data.

observed : the observed counts.

expected : the expected counts under the null hypothesis.

residuals : the Pearson residuals, *(observed - expected) / sqrt(expected)*. These allow you to pick out the most important associations (and their direction).

(Thanks to: *http://courses.statistics.com/software/R/Rchisq.htm*, *http://www.gardenersown.co.uk/Education/Lectures/R/basics.htm*)

You can use short versions of the above (eg obs, exp, res) provided such names are not already being used for soemthing else.

If you wish to extract a single value from one of these tables you can do that by appending row and column headings:

```
> results$expected["Trigger", "Switch"]

[1] 1775.754
```

If you get a warning message: `Chi-squared approximation may be incorrect in: chisq.test(ext)` it probably means that the expected count in one cell is very small.

To import a *.csv* file directly, and attach the names to

```
> raw <- read.csv("data/raw.csv", row.names = 1)
```

You can apply basic operations to the columns:

```
> sum(raw$ext_st)
```

```
[1] 2430
```

Using *attach()* means that you can access columns in your dataset by name instead of using *dataset$column*:

```
> attach(raw)
> sum(ext_st)
```

```
[1] 2430
```

However, this can lead to problems. If you attach the same dataset twice in succession, you will get a warning: `The following object(s) are masked from 'dataset (position n)'`. This is because you now have two datasets with the same column names, and the columns from the most recently loaded dataset will block usage of the identically-named ones from the previously-loaded copy of the dataset (you can check loading order via *search()*). This is merely a nuisance, but if you have two completely different datasets which happen to have one or more identical column names, you can end up using a completely different set of data from the one you thought you were using. Using *attach()* is therefore not recommended unless there are strong reasons of convenience for using it.

You can detach a dataset by using:

```
> detach(raw)
```

2 System information

Check what data-structures are in memory:

```
> ls()
```

```
 [1] "breaks"        "ext"           "ext_lab"       "f17"
 [5] "my_ext_lab"    "N"             "raw"           "results"
 [9] "roberts2_ext"  "s6"            "s8"            "st_range"
[13] "st_range.cut"  "st_range.freq" "Switch"        "Trigger"
```

Check the searchpath:

```
> search()
```

```
[1] ".GlobalEnv"        "package:stats"     "package:graphics"
[4] "package:grDevices" "package:utils"     "package:datasets"
[7] "package:methods"   "Autoloads"         "package:base"
```

or (giving directory paths):

```
> searchpaths()
```

```
[1] ".GlobalEnv"                    "/usr/lib/R/library/stats"
[3] "/usr/lib/R/library/graphics"   "/usr/lib/R/library/grDevices"
[5] "/usr/lib/R/library/utils"      "/usr/lib/R/library/datasets"
[7] "/usr/lib/R/library/methods"    "Autoloads"
[9] "/usr/lib/R/library/base"
```

3 Using Sweave

Sweave inserts the output from R into a LaTeX document.

Set up a basic LaTeX document, and save it with an *.Rnw* extension instead of a *.tex* extension.

In the .Rnw document, enclose the R code with ≪≫= and @:

≪≫=
some R code
@

and ensure that data etc is accessible by giving the correct filepaths in the R commands..

In R, from the directory where the *test.Rnw* file is located, run:

```
Sweave("test.Rnw")
```

This will generate a *test.tex* file, which can be opened and compiled in Kile (Alt+6), or by running:

```
pdflatex text.tex
```

The most difficult part of this is remembering to edit the *.Rnw* file instead of the *.tex* one. If the file contains nicely-formatted tables between
begin{Soutput}
and
end{Soutput}
you are in the *.tex* file.

You can insert switches into the ≪≫= – inserting *echo=FALSE* will show the results of the R code, but not the code itself:

```
            Codeswitch
Cognate      Switch No switch
  Trigger      2430     16776
  No trigger   2084     27532
```

Inserting *eval=FALSE* will show the R code, but not the results:

```
> ext
```

4 Frequency distribution

Set up the bins for the frequencies:

```
> breaks = seq(0, 120, by = 10)
> breaks
```

```
 [1]   0  10  20  30  40  50  60  70  80  90 100 110 120
```

Then select and process the relevant column (splice the column data with the breaks(*cut*), tabulate the result (*table*), and make it show as a column instead of the default row (*cbind*):

```
> cbind(table(cut(raw$ext_snt, breaks, right = FALSE)))
```

```
          [,1]
[0,10)       8
[10,20)     18
[20,30)     11
[30,40)     11
[40,50)     12
[50,60)      2
[60,70)      4
[70,80)      1
[80,90)      1
[90,100)     1
[100,110)    0
[110,120)    0
```