The **tables** Package

Duncan Murdoch

November 28, 2011

Contents

1 Introduction

This is a short introduction to the **tables** package. Inspired by my 20 year old memories of SAS PROC TABULATE, I decided to write a simple utility to create nice looking tables in Sweave documents. For example, we might display summaries of some of Fisher's iris data using the code

```
> tabular( (Species + 1) ~ (n=1) + Format(digits=2)*
+          (Sepal.Length + Sepal.Width)*(mean + sd), data=iris )
```

```
                 Sepal.Length      Sepal.Width
 Species      n  mean         sd   mean         sd
 setosa      50  5.01         0.35 3.43         0.38
 versicolor  50  5.94         0.52 2.77         0.31
 virginica   50  6.59         0.64 2.97         0.32
 All        150  5.84         0.83 3.06         0.44
```

You can also pass the output through the `Hmisc::latex()` function to produce LaTeX output, which when processed by `pdflatex` will produce the following table:

		Sepal.Length		Sepal.Width	
Species	n	mean	sd	mean	sd
setosa	50	5.01	0.35	3.43	0.38
versicolor	50	5.94	0.52	2.77	0.31
virginica	50	6.59	0.64	2.97	0.32
All	150	5.84	0.83	3.06	0.44

The idea of a table in the **tables** package is a rectangular array of values, with each row and column labelled, and possibly with groups of rows and groups of columns also labelled. These arrays are specified by "table formulas".

Table formulas are R formula objects, with the rows of the table described before the tilde (`"~"`, and the columns after. Each of those is an expression containing `"*"`, `"+"`, `"="`, as well as functions, function calls and variables, and parentheses for grouping. There are also various directives included in the formula, entered as "pseudo-functions", i.e. expressions that look like function calls but which are interpreted by the `tabular()` function.

For example, in the formula

```
(Species + 1) ~ (n=1) + Format(digits=2)*
          (Sepal.Length + Sepal.Width)*(mean + sd)
```

the rows are given by `(Species + 1)`. The summation here is interpreted as concatenation, i.e. this says rows for `Species` should be followed by rows for `1`.

In the `iris` dataframe, `Species` is a factor, so the rows for it correspond to its levels.

The `1` is a place-holder, which in this context will mean "all groups".

The columns in the table are defined by

```
(n=1) + Format(digits=2)*(Sepal.Length + Sepal.Width)*(mean + sd)
```

Again, summation corresponds to concatenation, so the first column corresponds to `(n=1)`. This is another use of the placeholder, but this time it is labelled as `n`. Since we haven't specified any other statistic to use, the first column contains the counts of values in the dataframe in each category.

The second term in the column formula is a product of three factors. The first, `Format(digits=2)`, is a pseudo-function to set the format for all of the entries to come. (For more on formats, see section 2.4.1 below.) The second factor, `(Sepal.Length + Sepal.Width)`, is a concatenation of two variables. Both of these variables are numeric vectors in `iris`, and they each become the variable to be analyzed, in turn. The last factor, `(mean + sd)` names two R functions. These are assumed to be functions that operate on a vector and produce a single value, as `mean` and `sd` do. The values in the table will be the results of applying those functions to the two different variables and the subsets of the dataset.

2 Reference

For the examples below we use the following definitions:

```
> set.seed(100)
> X <- rnorm(10)
> X

 [1] -0.50219235  0.13153117 -0.07891709  0.88678481
 [5]  0.11697127  0.31863009 -0.58179068  0.71453271
 [9] -0.82525943 -0.35986213
```

```
> A <- sample(letters[1:2], 10, rep=TRUE)
> A

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

> F <- factor(A)
> F

 [1] b b b b a a b b b a
Levels: a b
```

2.1 Function syntax

2.1.1 `tabular()`

`tabular(table, data=parent.frame(), n, suppressLabels=0)`

The `tabular` function has 4 arguments, but usually only the first two are used.

table The `table` argument is the table formula, described in detail below.

data The `data` argument is a dataframe or environment in which to look for the data referenced by the table.

n The `tabular` function needs to know the length of vectors on which it operates, because some formulas (e.g. `1 ~ 1`) contain no data. Normally `n` is taken as the number of rows in `data`, or the length of the first referenced object in the formula, but sometimes the user will need to specify it. Once specified, it can't be modified: all data in the table should be the same length.

suppressLabels By default, `tabular` adds a row or column label for each term, but this does sometimes make the table messy. Setting `suppressLabels` to a positive integer will cause that many labels to be suppressed at the start of each term. The pseudo-function `Heading()` can achieve the same effect, one term at a time.

The value returned is a list-mode matrix corresponding to the entries in the table, with a number of attributes to help with formatting. See the `?tabular` help page.

2.1.2 `format(), print(), latex()`

```
format(x, digits=4, justification="n", ...)
print(x, ...)
latex(x, file="", justification="c", ...)
```

The `tables` package provides methods for the `format`, `print` and `latex()` generics. The arguments are:

x The tabular object returned from `tabular()`.

digits The default number of digits to use when formatting.

justification The default text justification to use when printing. For text display, the recognized values are `"n"`, `"l"`, `"c"`, `"r"`, standing for none, left, center and right justification respectively. For LaTeX any justification string will be accepted; it should be one that the LaTeX `\tabular` environment accepts.

file The default method for the `Hmisc::latex()` generic writes the LaTeX code to a file; `latex.tabular()` can optionally do the same, but it defaults to writing to screen, for use in Sweave documents like this one.

2.2 Operators

2.2.1 $e_1 + e_2$

Summing two expressions indicates that they should be displayed in sequence. For rows, this means e_1 will be displayed just above e_2; for columns, e_1 will be just to the left of e_2.

Example:

```
> latex( tabular(F + 1 ~ 1) )
```

F	All
a	3
b	7
All	10

2.2.2 $e_1 * e_2$

Multiplying two expressions means that each element of e_1 will be applied to each element of e_2. If e_1 is a factor, then e_2 will be displayed for each element of it. NB: $*$ has higher precedence than $+$, so the expression $(e_1+e_2)*(e_3+e_4)$ is equivalent to $e_1 * e_3 + e_1 * e_4 + e_2 * e_3 + e_2 * e_4$.

Example:

```
> latex( tabular( X*F*(mean + sd) ~ 1 ) )
```

	F		All
X	a	mean	0.02525
		sd	0.34842
	b	mean	−0.03647
		sd	0.65611

2.2.3 $e_1 \sim e_2$

The tilde separates row specifications from column specifications, but otherwise acts the same as $*$, i.e. each row value applies to each column.

Example:

```
> latex( tabular( X*F ~ mean + sd ) )
```

	F	mean	sd
X	a	0.02525	0.3484
	b	−0.03647	0.6561

2.2.4 $e_1 = e_2$

The operator $=$ is used to set the name of e_2 to a displayed version of e_1. It is an abbreviation for `Heading(`e_1`)*`e_2. NB: because $=$ has lower operator precedence than any other operator, we usually put parentheses around these expressions, i.e. $(e_1 = e_2)$.

Example: `F` is renamed to "Newname".

```
> latex( tabular( X*(Newname=F) ~ mean + sd ) )
```

	Newname	mean	sd
X	a	0.02525	0.3484
	b	−0.03647	0.6561

2.3 Terms in Formulas

R parses table formulas into sums, products, and bindings separated by the tilde formula operator. What comes between the operators are other expressions. Other than the pseudo-functions described in section 2.4, these are evaluated and the actions depend on the type of the resulting value.

2.3.1 Closures or other functions

If the expression is the name of a function, or it evaluates to a function, then that is assumed to be the summary statistic to be displayed. The summary statistic should take a vector of values as input, and return a single value (either numeric, character, or some other simple printable value). If no summary function is specified, the default is `length`, to count the length of the vector being passed.

Example: `mean` and `sd` are specified functions; `n` is the renamed default statistic.

```
> latex( tabular( (F+1) ~ (n=1) + X*(mean + sd) ) )
```

		X	
F	n	mean	sd
a	3	0.02525	0.3484
b	7	−0.03647	0.6561
All	10	−0.01796	0.5611

2.3.2 Factors

If the expression evaluates to a factor, the dataset is broken up into subgroups according to the levels of the factor. Most of the examples above have shown this for the factor `F`, but this can also be used to display complete datasets:

Example: creating a factor to show all data

```
> latex( tabular( (i = factor(seq_along(X)))  ~
+        Heading()*(function(x) x)*(X+A +
+               (F = as.character(F) ) ) ) )
```

i	X	A	F
1	−0.50219	b	b
2	0.13153	b	b
3	−0.07892	b	b
4	0.88678	b	b
5	0.11697	a	a
6	0.31863	a	a
7	−0.58179	b	b
8	0.71453	b	b
9	−0.82526	b	b
10	−0.35986	a	a

2.3.3 Logical vectors

If the expression evaluates to a logical vector, it is used to subset the data.
Example: creating subsets on the fly.

```
> latex( tabular( (X > 0) + (X < 0)  + 1
+     ~ ((n = 1) + X*(mean + sd)) ) )
```

		X	
	n	mean	sd
$X > 0$	5	0.43369	0.3496
$X < 0$	5	−0.46960	0.2761
All	10	−0.01796	0.5611

2.3.4 Language Expressions

If the expression evaluates to a language object, e.g. the result of `quote()` or `substitute()`, then it will be replaced in the table formula by its result. This allows complicated table formulas to be saved and re-used. For examples, see section 2.5.

2.3.5 Other vectors

If the expression evaluates to something other than the above, then it is assumed to be a vector of values to be summarized in the table. If you would like to summarize a factor or logical vector, wrap it in `I()` to prevent special handling.

Note that only one value vector can be specified in any term, and all value vectors must be the same length, or an error will be reported.

Example: treating a logical vector as values.

```
> latex( tabular( I(X > 0) + I(X < 0)
+      ~ ((n=1) + mean + sd) ) )
```

	n	mean	sd
I(X > 0)	10	0.5	0.527
I(X < 0)	10	0.5	0.527

2.4 "Pseudo-functions"

Several directives to **tables** may be embedded in the table formula. This is done using "pseudo-functions". Syntactically they look like function calls, but special names are used. In each case, their action applies to later factors in the term in which they appear. For example, `X*Justify(r)*(Y + Format(digits=2)*Z) + A` will apply the `Justify(r)` directive to both `Y` and `Z`, but the `Format(digits=2)` directive will only apply to `Z`, and neither will apply to `A`.

2.4.1 `Format()`

By default **tables** formats each column using the standard `format()` function, with arguments taken from the `format.tabular()` call (see section 2.1.2).

The `Format()` pseudo-function does two things: it changes the formatting, and it specifies that all values it applies to will be formatted together. The "call" to `Format` looks like a call to `format`, but without specifying the argument `x`. When `tabular()` formats the output it will construct `x` from the entries in the table governed by the `Format()` specification.

Example: The mean and standard deviation are both governed by the same format, so they are displayed with the same number of decimal places.

```
> latex( tabular( (F+1) ~ (n=1)
+             + Format(digits=2)*X*(mean + sd) ) )
```

		X	
F	n	mean	sd
a	3	0.025	0.348
b	7	−0.036	0.656
All	10	−0.018	0.561

For customized formatting, an alternate syntax is to pass a function call to `Format()`, rather than a list of arguments. The function should accept an argument named `x`, to contain the data. It should return a character vector of the same length as x.

Example: Use a custom function and `sprintf()` to display a standard error in parentheses.

```
> stderr <- function(x) sd(x)/sqrt(length(x))
> fmt <- function(x, digits, ...) {
+   s <- format(x, digits=digits, ...)
+   is_stderr <- (1:length(s)) > length(s) %/% 2
+   s[is_stderr] <- sprintf("$(%s)$", s[is_stderr])
+   s[!is_stderr] <- latexNumeric(s[!is_stderr])
+   s
+ }
> latex( tabular( Format(fmt(digits=1))*(F+1) ~ X*(mean + stderr) ) )
```

	X	
F	mean	stderr
a	0.03	(0.20)
b	−0.04	(0.25)
All	−0.02	(0.18)

2.4.2 `.Format()`

The pseudo-function `.Format()` is mainly intended for internal use. It takes a single integer argument, saying that data governed by this call uses the same formatting the format specification indicated by the integer. In this way entries can be commonly formatted even when they are not contiguous. The integers are assigned sequentially as the format specification is parsed; users will likely need trial and error to find the right value in a complicated table with multiple formats.

Example: Format two separated columns with the same format.

```
> latex( tabular( (F+1) ~ X*(Format(digits=2)*mean
+                      + (n=1) + .Format(1)*sd) ) )
```

	X		
F	mean	n	sd
a	0.025	3	0.348
b	−0.036	7	0.656
All	−0.018	10	0.561

2.4.3 `Heading()`

Normally `tabular()` generates row and column labels by deparsing the expression being tabulated. These can be changed by using the `Heading()` pseudo-function, which replaces the heading on the next object found. The heading can either be the name of a function or a string in quotes, which will be displayed as entered (so LaTeX codes can be used).

If no argument is passed, the next label is suppressed.

Example: Replace `F` with a Greek Φ, and suppress the label for `X`.

```
> latex( tabular( (Heading("$\\Phi$")*F+1) ~ (n=1)
+              + Format(digits=2)*Heading()*X*(mean + sd) ) )
```

Φ	n	mean	sd
a	3	0.025	0.348
b	7	−0.036	0.656
All	10	−0.018	0.561

2.4.4 `Justify()`

The `Justify()` pseudo-function is used to specify the text justification of the headers and data values in the table. If called with one argument, that value is used for both labels and data; if called with two arguments, the first is used for the labels, the second for the data. If no `Justify()` specification is given, the default passed to `format()`, `print()` or `latex()` will be used.

Example:

```
> latex( tabular( Justify(r)*(F+1) ~ Justify(c)*(n=1)
+     + Justify(c,r)*Format(digits=2)*X*(mean + sd) ) )
```

		X	
F	n	mean	sd
a	3	0.025	0.348
b	7	−0.036	0.656
All	10	−0.018	0.561

2.5 Formula Functions

Currently only two examples of formula functions are provided, and neither is particularly robust. Users can provide their own as well. Such functions should return a language object, which will be substituted into the formula in place of the Formula function call.

2.5.1 `Hline(columns)`

This function produces horizontal lines in the table. It only works for LaTeX output, and must be the first factor in a term in the table formula. It has syntax

```
Hline(columns)
```

The argument is

`columns` An optional specification of which columns should get the line.

Example:

```
> latex( tabular( Species + Hline()  + 1 ~ (Sepal.Length + Sepal.Width
+                          + Petal.Length + Petal.Width)*mean, data=iris) )
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Species	mean	mean	mean	mean
setosa	5.006	3.428	1.462	0.246
versicolor	5.936	2.770	4.260	1.326
virginica	6.588	2.974	5.552	2.026
All	5.843	3.057	3.758	1.199

2.5.2 `PlusMinus()`

This function produces table entries like $x \pm y$ with an optional header. It has syntax

```
PlusMinus(x, y, head, xhead, yhead, digits=2, ...)
```

The arguments are

`x, y` These are expressions which should each generate a single column in the table. The `x` value will be flush right, the `y` value will be flush left, with the ± symbol between.

`head` If not missing, this header will be put over the pair of columns.

`xhead, yhead` If not missing, these will be put over the individual columns.

`digits, ...` These arguments will be passed to the standard `format()` function.

Example: Display mean ± standard error.

```
> stderr <- function(x) sd(x)/sqrt(length(x))
> latex( tabular( Justify(l)*(Species+1) ~ (Sepal.Length + Sepal.Width)*
+           PlusMinus(mean, stderr, digits=1), data=iris ) )
```

Species	Sepal.Length	Sepal.Width
setosa	5.01 ± 0.05	3.43 ± 0.05
versicolor	5.94 ± 0.07	2.77 ± 0.04
virginica	6.59 ± 0.09	2.97 ± 0.05
All	5.84 ± 0.07	3.06 ± 0.04