Publication quality tables in LaTeX*

Simon Fear
University of Liverpool

Printed July 13, 1997

Abstract

This article describes some additional commands to enhance the quality of tables in LaTeX. Guidelines are given as to what constitutes a good table in this context.

1 Introduction

The routines described below are to enable the easy production of tables such as (should) appear in published scientific books and journals. What distinguishes these from the tables plain LaTeX is able to produce is additional space above and below rules and rules of varying 'thickness'. What further distinguishes them from the tables many people *do* produce using LaTeX is the absence of vertical rules and double rules.

I must draw a clear distinction between what I mean by a *formal table*, which is a set of values/labels in columns, as distinct from what I will call a *tableau*, which is the kind of thing illustrated in the LaTeX manual, and which is increasingly common as the output of of many database/management systems, that is with icons in abundance, and probably colour into the bargain. The layout of a *tableau* is determined (hopefully) as a one-off, given a jumble of material the designer is trying to combine into a meaningful configuration. But the layout of a *table* has been established over literally centuries of experience and should only be altered in truly extraordinary circumstances.

By way of illustration, consider this tableau from the LaTeX manual (p. 64 old edition):

gnats	gram	$13.65
	each	.01
gnu	stuffed	92.50
emu		33.33
armadillo	frozen	8.99

This is a hotch-potch of information that is probably reasonably clearly presented as is (but is the emu stuffed or not?). However, as a published table, this should certainly instead appear along the lines suggested further down the page in the manual:

*This file has version number v1.00, last revised 6 November 1995.

Item		
Animal	Description	Price ($)
Gnat	per gram	13.65
	each	0.01
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	8.99

It may be noted that it in fact takes much less work to lay this out as a formal table; we don't have to work out a new layout for everything we do. Moreover, we can be almost certain that the data cannot be misread (because the reader does not have to learn how to read each of the infinite number of different possible tableaux).

The above table cannot be produced in pure LaTeX, unfortunately. It can be laid out as it should be, but despite your best efforts, using plain `\hline` commands produces

Item		
Animal	Description	Price ($)
Gnat	per gram	13.65
	each	0.01
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	8.99

Note (if it is not already obvious) that there is not enough space between the top line and the capital I of 'Item', and so on for all the lines: contrast with the previous version. Also, the top and bottom rules (ie lines) in the first version are heavier than the middle rule, which is turn heavier than the subrule underneath 'Item'. Yes, I *know* you can redefine `\doublerulesep` and then use `\hline\hline` to get something near the same effect, and you can use struts to improve the spacing. But that is a fiddle and you should not have to think of such things. The `booktabs` style defines its commands so that such things are taken care of automatically.

In general, I would say that this package is of no interest to those looking for an alternative to PicTeX to conjure up fancy tableaux. Rather, it is a style guide for authors of scientific papers and books as regards table layout. It is not going too far to say that if you cannot create a table using the commands in this package, you have designed it wrong.

1.1 A note on terminology

In British typesetting, a 'line' is always called a 'rule'. Perhaps confusingly (for historic reasons in fact), the 'thickness' of rule is often referred to as is its 'width' (whereas just about everyone else would call this 'depth' or 'height', if they were thinking of a horizontal rule). A 'thick black line' is called a 'heavy rule'. I have used this terminology in most of the new commands below. If nothing else it avoid confusion with `\hline`.

2 The layout of formal tables

You will not go far wrong if you remember two simple commandments at all times:

1. Never, ever use vertical rules.
2. Never use double rules.

These commandments may seem extreme but in years of experience I have never found a good argument in favour of breaking them. For example, if you feel that the information in the left half of a table is so different from that on the right that it needs to be separated by a vertical line, then you should use two tables instead. The second commandment is very, very occasionally violated: I have worked for a publisher who insisted on a double light rule above a row of totals. But this would not have been my choice.

There are three further guidelines I might mention here as they are so poorly known outside the circles of professional typesetters and subeditors:

1. Put the units in the column heading (not in the body of the table).
2. Always precede a decimal point by a digit; thus 0.1 *not* just .1.
3. Do not use 'ditto' signs or any other such convention to repeat a previous value. In many circumstances a blank will serve just as well. If it won't, then repeat the value.

Is this just me being pedantic? These last guidelines are violated with increasing frequency in published work. To me, this just indicates that the typesetting is amatuer.

Anyway, whether or not you wish to follow the minor niceties, if you use only the following commands in your formal tables your reader will be grateful. (I repeat that the guidelines are not just to keep the pedantic happy. Enforced structure of presentation enforces structured thought in the first instance.)

3 Use of the new commands

`\toprule` `\midrule` `\bottomrule`

In the simplest of cases a table begins with a `\toprule`, has a single row of column headings, then a dividing rule called here a `\midrule`; after the columns of data we finish off with a `\bottomrule`. Most book publishers set the `\toprule` and `\bottomrule` heavier (ie thicker, or darker; see notation) than the intermediate `\midrule`. However, when tables appear in very small typesizes it is sometimes impossible to make this distinction, and moreover quite a few journals use all rules of the same heaviness. The rule commands here all take a default which may be reset within the document (preferably, but not necessarily, in the preamble). For the top and bottom rules this is `\heavyrulewidth` and for midrules it is `\lightrulewidth` (fully described below). In very rare cases where you need to do something special, you may use the optional arguments to the rule commands which have formal syntax as follows:

`\toprule[`⟨*wd*⟩`]`
`\midrule[`⟨*wd*⟩`]`
`\bottomrule[`⟨*wd*⟩`]`

where ⟨*wd*⟩ is a TEXdimension (for example 1pt, .5em, etc.).

All the rule commands here go immediately after the closing `\\` of the preceding row (except of course `\toprule`, which comes right after the `\tabular{}` command); in other words, exactly where plain LATEX allows `\hline` or `\cline`.

`\cmidrule` Of course more often than not we need a rule to extend over certain of the columns, for which we need a `\cmidrule` (the analogue of LATEX's `\cline` command). Generally, this rule should not come to the full width of the end columns, and this is especially the case when we need to begin a `\cmidrule` straight after the end of another one (LATEX's `\cline`s crash into each other here if you are not extra careful with `\extracolsep`). Thus, you will generally want to use the optional 'trimming' commands, which are `(r)`, `(l)` and `(rl)` or `(lr)`, indicated whether the right and/or left ends of the rule should be trimmed. Note the exceptional use of parentheses instead of braces or brackets for this command, the full syntax of which is

`\cmidrule[`⟨*wd*⟩`](`⟨*trim*⟩`){`*a*–*b*`}`

where again ⟨*wd*⟩ is an optional rule width command (the default here is `\cmidrulewidth`) and the last argument, which is not optional, gives the column numbers to be spanned.

An example of the commands in use is given by the code used to produce the example table above:

```
\begin{tabular}{@{}llr@{}} \toprule
\multicolumn{2}{c}{Item} \\ \cmidrule(r){1-2}
Animal & Description & Price (\$)\\ \midrule
Gnat  & per gram  & 13.65 \\
      & each      & 0.01 \\
Gnu   & stuffed   & 92.50 \\
Emu   & stuffed   & 33.33 \\
Armadillo & frozen & 8.99 \\ \bottomrule
\end{tabular}
```

`\addlinespace` Occasionally we want to put an extra space between certain rows of a table; for example, before the last row, if this is a total (space is better than another `\midrule` here, in my opinion). This is simply a matter of inserting

`\addlinespace[`⟨*wd*⟩`]`

after the `\\` alignment marker. It is not a bad idea to think of `\addlinespace` as being a white rule of width ⟨*wd*⟩. The default space is `\defaultaddspace` which gives rather less than a whole line space (such as you could create by using `\\ \\` at the end of the line; this is really too much space in most cases).

4 Abuse of the new commands

Let's face it, nobody can leave well alone, so here are some guidelines and extra commands for TEXperts and meddlers.

The new rule commands are not guaranteed to work with `\hline` or `\cline`, although these remain available and unchanged. I cannot foresee any reason to want to mix them.

More importantly the rules generated by the new commands are in no way guaranteed to connect with verticals generated by `{|}` characters in the preamble. This is a feature (see above). You should not use vertical rules in tables, period.

`\morecmidrules` If you just cannot stop yourself from using a double rule, even a construction as bizarre as `\toprule\bottomrule\midrule` will work without generating an error message (just as you can double `\hline`). These rules will be separated by the ordinary LaTeX separator `\doublerulesep`. However if your perversion is to want double `\cmidrule`s you will need the extra command `\morecmidrules` to do so properly, because normally two `\cmidrule`s in a row is a completely sane construction calling for two rules on the same 'rule row'. Thus in

```
\cmidrule{1-2}\cmidrule{1-2}
```

the second command writes a rule that just overwrites the first one; I suppose you wanted

```
\cmidrule{1-2}\morecmidrules\cmidrule{1-2}
```

which gives you a double rule between columns one and two, separated by `\cmidrulesep` (note: since a `\cmidrule` is generally very light, the ordinary `\doublerulesep` is probably too much space). Finish off a whole row of rules before giving the `\morecmidrules` command. Note that `\morecmidrules` has no effect whatsoever if it does not immediately follow a `\cmidrule` (ie it is not a general space-generating command).

`\specialrule` Talking of which brings us to the abuse of `\addlinespace` to generate weird extra space between rules. Don't do it. (It's not actually illegal though.) Instead use

`\specialrule{`⟨*wd*⟩`}{`⟨*abovespace*⟩`}{`⟨*belowspace*⟩`}`

where it should be noted all three arguments are mandatory (I couldn't be bothered to program in defaults). If you use this frequently, you have misunderstood the purpose and content of the guidelines given above. Technical note: no space is added after a preceding rule, but a following rule will generate a `\doublerulesep` above itself. Why are you reading this section?

5 Technical summary of commands

The new rule commands are valid inside the `tabular` (and in fact also `array`) environment, in all versions of LaTeX (you must remove the

```
\ProvidesPackage{booktabs}
```

line from `booktabs.sty` to run in LaTeX2.09 and earlier), and fully compatible with the extended array environment of the LaTeX2ϵ tools package.

The commands follow the standard placement syntax of `\hline`. If rule commands are doubled, it is safest to ensure there is no space between the commands. (In many circumstances violations of this safety rule will give the weird message "`misplaced noalign{`".) Don't use double rules!

For the present purposes a 'rule' is any of `\toprule`, `\midrule`, `\bottomrule`, `\cmidrule`, `\specialrule` or `\addlinespace`. Note especially the inclusion of `\addlinespace` in this list; and the exclusion of `\hline` and `\cline`, which will give unpredicatble results if mixed with these.

`\toprule[`⟨*wd*⟩`]`

A rule of width ⟨*wd*⟩ (default `\heavyrulewidth`) with `\belowrulesep` extra vertical space inserted below it (unless followed by another rule command, in which case `\doublerulesep` vertical space follows).

`\midrule[`⟨*wd*⟩`]`

A ⟨*wd*⟩ (default `\lightrulewidth`) rule with `\aboverulesep` space above it (unless preceded by another rule, from which it will be separated by `\doublerulesep`) and with `\belowrulesep` space below it (unless another rule follows).

`\bottomrule[`⟨*wd*⟩`]`

A ⟨*wd*⟩ (default `\heavyrulewidth`) rule with `\aboverulesep` space above it (unless preceded by another rule, from which it will be separated by `\doublerulesep`) and with `\belowrulesep` space below it (unless another rule follows). The extra space below is to make space for table footnotes.

`\cmidrule[`⟨*wd*⟩`](`⟨*trim*⟩`){`*a*–*b*`}`

A ⟨*wd*⟩ (default `\cmidrulewidth`) rule with `\aboverulesep` space above it (unless following another `\cmidrule`, in which case it is on the same vertical alignment; or if following any other rule, separated by `\doublerulesep`; or if following `\morecmidrules`, separated by `\cmidrulesep`), and with `\belowrulesep` below it (unless followed by another `\cmidrule`, in which case the following rule is on the same vertical alignment; or if followed by `\morecmdirules`, with `\cmidrulesep` below it).

The rule spans columns *a* to *b*. The optional argument ⟨*trim*⟩ which note goes in parentheses if at all, can be either `r` for right trimming, `l` for left trimming, or both.

`\addlinespace[`⟨*wd*⟩`]`

Actually classed as a zero-width rule with no extra space above and with ⟨*wd*⟩ (default `\defaultaddspace`) space below (if another rule follows, this will be separated by a further `\doublerulesep`). In practice use this only to add space between rows in the body of the table.

`\specialrule{`⟨*wd*⟩`}{`⟨*abovespace*⟩`}{`⟨*belowspace*⟩`}`

A ⟨*wd*⟩ rule (note mandatory argument) with ⟨*abovespace*⟩ above it and ⟨*belowspace*⟩ below it (unless another rule follows, in which case this will be separated by a further `\doublerulesep`).

`\morecmidrules`

Instructs LaTeX to begin a new row of `\cmidrules`, separated from the last by `\cmidrulesep`. Has no effect outside this environment.

The default dimensions are

```
\heavyrulewidth=.08em
\lightrulewidth=0.5em
\cmidrulewidth=0.3em
\belowrulesep=.65ex
\aboverulesep=.4ex
\defaultaddspace=.5em
\cmidrulekern=.25em
```

The last of these, `\cmidrulekern`, is the amount by which a `\cmidrule` is trimmed at each end indicated in the `()` options. In the construction

```
\cmidrule(r){1-2}\cmidrule(l){3-4}
```

there is a total of .5 em separating the two rules. Currently the only way to get special effects is to reset `\cmidrulekern` as appropriate; the amount of trimming is not available as an argument in the current implementation of `\cmidrule`.

The user can change these defaults on the fly by simply inserting a command in exactly the same format as above; the redefinition will stay in effect for the rest of the document or until redefined again.

6 Support for `firsthline` and `lasthline`

Yes in a way, but mostly no. Just not applicable. The commands here are not about doing neat things with boxes; those are tableaux not tables. You do not ever, ever want a formal table in the middle of a line of text.

However, if you use the `array.sty` package, the commands are not altered by the present code (just as `\hline` and `\cline` remain active).

7 Acknowledgments

Hugely indebted of course to DEK and Lamport; the optional argument and `\cmidrule` stuff especially is filched and adapted from `latex.sty`. The documentation driver stuff is stolen wholesale from the tools package description `dcolumn.dtx` by David Carlisle.

For beta testing and encouragement ...

8 The code

The current version is defined at the top of the file looking something like this

```
⟨*package⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{booktabs}
        [\filedate\space version\fileversion]
```

First we set up the new dimensions described above:

```
\newdimen\heavyrulewidth
\newdimen\lightrulewidth
\newdimen\cmidrulewidth
\newdimen\belowrulesep
\newdimen\aboverulesep
\newdimen\cmidrulesep
\newdimen\cmidrulekern
\newdimen\defaultaddspace
\heavyrulewidth=.08em
\lightrulewidth=.05em
\cmidrulewidth=.03em
\belowrulesep=.65ex
\aboverulesep=.4ex
```

```
\cmidrulesep=\doublerulesep
\cmidrulekern=.25em
\defaultaddspace=.5em
```

And some internal counters of no interest to the end user:

```
\newcount\rulesflag
\newdimen\@cmidrulewidth
\newcount\@cmidla
\newcount\@cmidlb
\rulesflag=0
```

which will be described as needed below.

8.1 Full width rules

We put the full width rules into a `\noalign{}` group, using a dirty trick (`\ifnum=0‘}`) to fool the parser that the bracket count is OK. The bracket really gets closed after all the skipping at the end of the `\@endrule` macro.

`\toprule`

```
\def\toprule{\noalign{\ifnum0=‘}\fi
    \@ifnextchar[{\@toprule}{\@toprule[\heavyrulewidth]}}
```

This allows for `\toprule`'s optional argument; if it has one, it is passed to `\@toprule`, otherwise we call this with the default `\heavyrulewidth`.

In the following, if `\rulesflag` has been set (to one) we have just done a previous rule that has been exceptionally modified to not have its normal space below, so we need to put `\doublerulesep` before this `\toprule`; then we clear the flag. Note: we can't just always add `\belowrulesep` below a `\toprule`, because there should be `\doublerulesep` between succesive rules. But we could just ban double rules!

```
\def\@toprule[#1]{\ifnum\rulesflag=1\vskip
    \doublerulesep\global\rulesflag=0\fi
    \hrule \@height#1\futurelet\@tempa\@endrule}
```

In the third line above we have put in the rule, and we call the closing routine `\@endrules` with `\@tempa` set to the token following the rule command in the document.

```
\def\@endrule{\ifx\@tempa\toprule\global\rulesflag=1%
    \else\ifx\@tempa\midrule\global\rulesflag=1%
    \else\ifx\@tempa\bottomrule\global\rulesflag=1%
    \else\ifx\@tempa\cmidrule\global\rulesflag=1%
    \else\ifx\@tempa\specialrule\global\rulesflag=1%
    \else\ifx\@tempa\addlinespace\global\rulesflag=1%
    \else\vskip \belowrulesep\fi\fi\fi\fi\fi\fi\ifnum0=‘{\fi}}
```

Here if the next command is another rule or linespace (shame on user!) we have set the `\rulesflag` to 1 (and added no space); otherwise we have added the proper space below.

`\midrule` `\bottomrule`

The code is almost the same as for a `\toprule` except for the addition of space above.

Note that as far as programming is concerned, a `\bottomrule` is just a heavy `\midrule` (but the end user should not think of it in this way).

```
\def\midrule{\noalign{\ifnum0=‘}\fi
```

```
    \@ifnextchar[{\@midrule}{\@midrule[\lightrulewidth]}}
\def\@midrule[#1]{\ifnum\rulesflag=1\vskip
    \doublerulesep\global\rulesflag=0
    \else\vskip \aboverulesep\fi
    \hrule \@height#1\futurelet\@tempa\@endrule}
\def\bottomrule{\noalign{\ifnum0=`}\fi
    \@ifnextchar[{\@midrule}{\@midrule[\heavyrulewidth]}}
```

`\addlinespace` An `\addlinespace` is handled like a zero-width rule with no space above and argument (or default) space below. Note that a following rule will be added after an *additional* `\doublerulesep` space. (Users are not encouraged to add space before/after rules with `\addlinespace`. If this is needed, they should use a `\specialrule`.)

```
\def\addlinespace{\noalign{\ifnum0=`}\fi
    \@ifnextchar[{\@addspace}{\@addspace[\defaultaddspace]}}
\def\@addspace[#1]{\ifnum\rulesflag=1\global\rulesflag=0\fi
    \vskip #1\futurelet\@tempa\@endrule}
```

`\specialrule` This is included with some misgivings as it allows the user to do silly things. But a style designer might need this (or a modification of it) in times of need.

Note that a following `\toprule` will behave as expected (no extra space above), but a `\midrule` or `\bottomrule` will add `\aboverulespace`, whereas a following `\cmidrule` will add `\doublerulesep`. Why would you ever want to do this anyway?

```
\def\specialrule#1#2#3{\noalign{
    \ifnum\rulesflag=1\global\rulesflag=0
    \else\vskip #2\fi\hrule \@height#1\vskip #3}}
```

8.2 Special subrules

`\cmidrule` The `\cmidrule` uses `\rulesflag` in a slightly different way. This is (left) set to one if you are in the middle of a row of `\cmidrules`, or starting a new one (with `\morecmidrules`). Otherwise, when `\rulesflag` is zero, we precede the rule with `\aboverulesep`.

```
\def\cmidrule{\noalign{\ifnum0=`}\fi
    \@ifnextchar[{\@cmidrule}{\@cmidrule[\cmidrulewidth]}}
\def\@cmidrule[#1]{\@ifnextchar({\@@cmidrule[#1]}{\@@cmidrule[#1]()}}
\def\@@cmidrule[#1](#2)#3{\@@@cmidrule[#3]{#1}{#2}}
```

The above is fiddling around to set defaults for missing optional arguments. We also pass to `\@@@cmidrule` in a different order, namely `[a-b]{width required}{kerning commands}` (can't remember why I did this):

```
\def\@@@cmidrule[#1-#2]#3#4{\global\@cmidla#1\relax
    \global\advance\@cmidla\m@ne
    \ifnum\@cmidla>0\global\let\@gtempa\@cmidrulea\else
    \global\let\@gtempa\@cmidruleb\fi
    \global\@cmidlb#2\relax
    \global\advance\@cmidlb-\@cmidla
```

This has set up a switch to call the relevant routine, `\@cmidrulea` or `\@cmidruleb`, depending on whether we start from column one or not (what a faff).

```
    \global\@cmidrulewidth=#3
```

That is, set per default or given argument.

Now we parse the trimming arguments (if there are any):

```
    \global\let\cmlkern@l\z@ \global\let\cmlkern@r\z@
    \@tfor\@tempa :=#4\do{\global\expandafter\let
        \csname cmlkern@\@tempa\endcsname\cmidrulekern}%
```

Now insert space above if needed, close the `\noalign`, then switch to appropriate rule drawing routine as determined above (`\let` to `\@gtempa`):

```
\ifnum\rulesflag=0\vskip \aboverulesep\fi\ifnum0=`{\fi}\@gtempa
```

Now open another `\noalign`, and call the closing routine:

```
    \noalign{\ifnum0=`}\fi\futurelet\@tempa\@xcmidrule}
```

This code (called above) actually draws the rules:

```
\def\@cmidrulea{\multispan\@cmidla&\multispan\@cmidlb
    \unskip\hskip \cmlkern@l\leaders\hrule \@height\@cmidrulewidth\hfill
    \hskip \cmlkern@r\cr}
\def\@cmidruleb{\multispan\@cmidlb
    \unskip\hskip \cmlkern@l\leaders\hrule \@height\@cmidrulewidth\hfill
    \hskip \cmlkern@r\cr}
```

Finally, the closing routine. If another `\cmidrule` follows, backspace vertical so it will line up, and `\rulesflag=1` will suppress adding space above the next. If a `\morecimdrules` follows, we add (positive) `\cmidrulesep` (and again set `\rulesflag` to one). Otherwise this is the last rule of the current group and we can just add `\belowrulesep`.

Finally, we close the `\noalign`.

```
\def\@xcmidrule{\ifx\@tempa\cmidrule\vskip-\@cmidrulewidth
    \global\rulesflag=1\else
    \ifx\@tempa\morecmidrules\vskip \cmidrulesep
    \global\rulesflag=1\else
    \vskip \belowrulesep\global\rulesflag=0\fi\fi
    \ifnum0=`{\fi}}
```

`\morecmidrules` This is really a dummy command; all the work is done above within the `\cmidrule` routine. We look one step ahead there to see if a `\morecmidrules` follows the current `\cmidrule`, and if so set the flag. Otherwise, `\morecmidrules` itself does nothing.

```
\def\morecmidrules{\noalign{\relax}}
```

⟨/package⟩