LaTeX Table Hints and Tips

Adrian P. Robson
adrian.robson@nepsweb.co.uk

1st June, 2013

Contents

1 Introduction

This is a collection of various methods for laying out and formatting LaTeX tables. All of these examples work with pdfLaTeX [13], which is the author's preferred tool.

A more general discussion of LaTeX methods can be found in *LaTeX Hints and Tips* [12].

2 Standard Tables

Tables are made in LaTeX using the `tabular` environment like this

```
\begin{tabular}{|l|l|l|}
\hline
\multicolumn{3}{|c|}{A Table}\\\hline
\hline
1,1 & 1,2 & 1,3 \\\hline
2,1 & 2,2 & 2,3 \\\cline{1-2}
3,1 & 3,2 &     \\\hline
\end{tabular}
```

A Table		
1,1	1,2	1,3
2,1	2,2	2,3
3,1	3,2	

For more information on basic tables consult a good text like *The Not So Short Introduction to LaTeX 2ε* [10].

3 Spacing

3.1 Row Spacing

The standard row height is too small when `\hline` is used. There are three explicit ways of correcting this: modifying the `arraystretch` command, changing the `extrarowheight` length, or using the `bigstrut` command. Alternatively, the `booktabs` or `ctable` packages can be used as discuses in §12.

3.1.1 Arraystretch

The crudest way to increase row spacing, which works without any additional packages, is to increase the `arraystretch` ratio. This injects space above and below all of the rows' text. It is done like this:

```
{
\renewcommand{\arraystretch}{1.2}
\begin{tabular}{|c|l|}
\hline
a  & Row 1 \\ \hline
b  & Row 2 \\ \hline
c  & Row 2 \\
d  & Row 4 \\ \hline
\end{tabular}
}
```

The `renewcommand` and the table are enclosed in braces to limit the scope of the redefinition. Its effect is to turn

a	Row 1
b	Row 2
c	Row 3
d	Row 4

into

a	Row 1
b	Row 2
c	Row 3
d	Row 4

and with a bigger value

a	Row 1
b	Row 2
c	Row 3
d	Row 4

3.1.2 Extrarowheight

A more sophisticated approach is to used the `array` package [8] and change the length `\extrarowheight` as follows:

```
\usepackage{array}
...
{
\setlength{\extrarowheight}{1.5pt}
\begin{tabular}{|l|l|}
  \hline
  a  & Row 1 \\ \hline
  b  & Row 2 \\ \hline
  c  & Row 3 \\
  d  & Row 4 \\ \hline
\end{tabular}
}
```

This adds space only above the text. With the correct value, it compensates for the `\hline` command spacing. Its effect is to turn

a	Row 1
b	Row 2
c	Row 3
d	Row 4

into

a	Row 1
b	Row 2
c	Row 3
d	Row 4

and with a bigger value

a	Row 1
b	Row 2
c	Row 3
d	Row 4

3.1.3 Bigstruts

The above methods apply the compensation to all rows, even if they do not have `\hline` commands. A subtler alternative is to use the `bigstrut` package [11] like this:

```
\usepackage{bigstrut}
...
\begin{tabular}{|l|l|}
\hline
a  & Row 1 \bigstrut \\ \hline
b  & Row 2 \bigstrut \\ \hline
c  & Row 2 \bigstrut[t] \\
d  & Row 4 \bigstrut[b] \\ \hline
\end{tabular}
```

which changes the basic table

a	Row 1
b	Row 2
c d	Row 3 Row 4

into

a	Row 1
b	Row 2
c d	Row 2 Row 4

The `\bigstrut` command is used when there is are `\hline` above and below; `\bigstrut[t]` when there is an `\hline` only above; and `\bigstrut[b]` when there is only one below.

The `bigstrut` package only works well with tables that have single line cells. It does not work with the text wrap column specifiers `p{}, m{}` and `b{}`. (For more information on text wrap see §5).

3.1.4 Comparison of Methods

The affect of the three methods is as follows:

`arraystretch{1.2}`

a	Row 1
b	Row 2
c	Row 3
d	Row 4

`extrarowheight{1.5pt}`

a	Row 1
b	Row 2
c	Row 3
d	Row 4

`bigstrut`

a	Row 1
b	Row 2
c d	Row 2 Row 4

For tables without text wrapping, the `bigskip` approach is definitely the best if the table has some rows without horizontal lines. Otherwise, `extrarowheight` is less verbose and gives a more compact layout. Use `arraystretch` if the table needs a large row height. For tables with text wrapping, `extrarowheight` is probably best.

3.2 Column Spacing

Column width can be modified by changing `\tabcolsep` like this:

```
\setlength{\tabcolsep}{10pt}
\begin{tabular}{|l|l|}
  \hline
  a  & Row 1 \\ \hline
  b  & Row 2 \\ \hline
  c  & Row 3 \\ \hline
\end{tabular}
```

which changes the default

a	Row 1
b	Row 2
c	Row 3

into

a	Row 1
b	Row 2
c	Row 3

There is a standard column specifier `@{cmnd}`, which suppresses inter-column space and inserts `cmnd` instead. This can be used to insert or remove space into a particular column. For example:

```
\setlength{\extrarowheight}{1.5pt}
\begin{tabular}{|@{\hspace{1cm}}l|@{}l|}
\hline
Abcd & Abcd \\ \hline
Abcd & Abcd \\
\hline
\end{tabular}
```

Abcd	Abcd
Abcd	Abcd

4 Decimal Point Alignment

The traditional method of aligning decimal points uses the `@` command. Two columns are used for a decimal number, and a `multicolumn` command is used in the heading, as the following shows:

```
\begin{tabular}{|l|r@{.}l|l|}
a    & \multicolumn{2}{c|}{b} & c \\
test & 2 & 8  & test \\
test & 1 & 45 & test \\
test & 0 & 5  & test
\end{tabular}
```

a	b	c
test	2.8	test
test	1.45	test
test	0.5	test

A disadvantage of this method is that the numbers are expressed in the code without a decimal point.

As an alternative, the `dcolumn` package [1] allows decimal numbers to be expressed naturally in a single column. The package defines a column specifier

`D{`<sepcode>`}{`<sepout>`}{`<decimal places>`}`

The first argument is the decimal separator character used in the code, the second is the character in the output, and the last is the number of decimal places. Typically, a `newcolumntype` command is used in the preamble to make table declaration a little simpler.

Even though there is just one column, A `multicolumn` command still has to be used to get the correct alignment and font style in the header, as the following example shows:

```
\newcolumntype{d}[1]{D{.}{.}{#1}}
...
\begin{tabular}{|l|d{2}|l|}
a    & \multicolumn{1}{c|}{b} & c \\
test & 2.8  & test \\
test & 1.45 & test \\
test & 0.5  & test
\end{tabular}
```

a	b	c
test	2.8	test
test	1.45	test
test	0.5	test

As the example shows, using an appropriate positive value for number of decimal places produces the same table layout as the traditional method. This aligns the decimal places, and minimises the column width.

Alternatively, a negative value for decimal places will centre the decimal place in the column, but will not minimise the width. For number lists with reasonably balanced whole and decimal parts this looks good. However, for lists

with disproportionate whole and decimal parts, the columns can have too much empty space, as the following shows:

a	b	c
test	208.6	test
test	1.45	test
test	0.532	test

a	b	c
test	2000008.6	test
test	1.45	test
test	0.5	test

5 Vertical Alignment and Text Wrapping

Vertical alignment can be controlled with the `array` package [8], which has additional text wrap formatting commands:

`p{width}`	Top align, the same as usual.
`m{width}`	Middle align
`b{width}`	Bottom align

These produce the following layouts:

Column Format		
`p{}`	`p{}`	`l`
1 1 1 1 1 1	2 2 2 2	3 3

Column Format		
`m{}`	`m{}`	`l`
1 1 1 1 1 1	2 2 2 2	3 3

Column Format		
`b{}`	`b{}`	`l`
1 1 1 1 1 1	2 2 2 2	3 3

Notice how the `m{}` or `b{}` alignment affects the whole table. In addition, `p{}` `m{}` and `b{}` formats cannot be successfully mixed in the same table.

Occasionally the text wrap formats cause 'bad box' warnings, which can often be resolved by with `\raggedright` (see §6).

6 Ragged Right Alignment

For narrow wrapped text blocks left justification often looks best, and can get rid of 'bad box' warnings. There are three ways to achieve this: a simple command, column types, and the `tabulary` package. All of these methods support or can be adapted to provide alignments other than left justified. The first two are very versatile and can be used to inject general formatting commands.

6.1 Simple Command

The most verbose but flexible way to achieve left justification is just to use `\raggedright` in the cell:

```
\newcommand{\rr}{\raggedright}
\newcommand{\tn}{\tabularnewline}
...
{ \renewcommand{\arraystretch}{1.2}
  \begin{tabular}{|c|p{5cm}|}
  \hline
  1,1 & \rr ... text ... \tn \hline
  2,1 &     ... text ... \\  \hline
  \end{tabular} }
```

1,1	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
2,1	Nullam rhoncus, sem luctus ultrices.

Using this method for left justification can be applied to individual cells. Note that `\tabularnewline` replaces `\\` in the row containing the `\raggedright` command. The definitions for `\rr` and `\tn` are just to make things a little more compact.

6.2 Column Types

Another way of managing ragged right formatting is to define a new column type using the `array` package [8]. Its `>{decl}` option inserts `decl` directly before the entry for the column; and `<{decl}` directly after. The following shows how it can be used:

```
\usepackage{array}
...
\newcolumntype{x}[1]
           {>{\raggedright}p{#1}}
\newcommand{\tn}{\tabularnewline}
...
{ \renewcommand{\arraystretch}{1.2}
  \begin{tabular}{|c|x{5cm}|}
  \hline
  1,1 & ... text ... \tn \hline
  2,1 & ... text ... \tn \hline
  \end{tabular} }
```

1,1	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
2,1	Nullam rhoncus, sem luctus ultrices.

With this method ragged right formatting is applied to a whole column. The `newcolumntype` command is used to define a column type that can be reused. However, the formatting could have been embedded in the tabular heading. Again `\tabularnewline` is needed when `\raggedright` is used.

6.3 Tabulary

Another alternative is to use the `tabulary` package [4]. With this method ragged right columns are simply declared with the `L` command. However, the total table width must be defined as a parameter, as shown below. See §9.2 for more on this package.

```
\usepackage{tabulary}
...
\renewcommand{\arraystretch}{1.2}
\begin{tabulary}{6.5cm}{|c|L|}
\hline
1,1 & ... text ... \\ \hline
2,1 & ... text ... \\ \hline
\end{tabulary}
```

1,1	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
2,1	Nullam rhoncus, sem luctus ultrices.

6.4 Comparison of Methods

Simple commands are useful for small tables or when the formatting does not apply to the whole column. The `tabulary` package is much simpler than column types, but requires the table width to be specified. Column types can be used for any appropriate formatting or space requirements.

7 Multiple Rows

The easiest way to have tables with spanning rows is to use the `multirow` package [11]. In its simplest form, it can be used like this:

```
\usepackage{multirow}
...
\begin{tabular}{|c|l|}
\hline
a  & Row 1                              \\ \hline
b  & \multirow{2}{*}{Spanning rows} \\ \cline{1-1}
c  &                                    \\ \hline
d  & Row 4                              \\ \hline
\end{tabular}
```

giving the following:

a	Row 1
b	Spanning rows
c	
d	Row 4

The `\multirow` command declares the location of the spanning rows. Its first argument is the number of rows to span. The second, in this case, states that the text argument's natural width should be used. The relevant columns in lower rows must be left blank. The full `multirow` command is more complicated:

```
\multirow{nrows}[bigstruts]{width}[fixup]{text}
```

If the `bigstrut` package is used, the number of struts in the spanned rows should be stated as the `bigstruts` parameter. Count 2 for every `\bigstrut` and 1 for a `\bigstrut[t]` or `\bigstrut[b]`.

The text width can be set with the `width` parameter, in which case the text will be wrapped and left justified. Line breaks can be forced with a `\\` command. However, the text must have no more lines than the number of rows spanned. Using an `*` for the width, as in the example above, makes the column's cells single line and as wide as necessary.

If the vertical position of the text needs fine tuning, it can be moved up or down with the `fixup` optional parameter.

To span rows and columns together, a `\multirow` should be nested in a `\multicolumn`. Matching but empty `\multicolmn` commands are needed for all of the lower spanned rows.

All of these options are shown in the following example:

```
\begin{tabular}{|c|c|l|l|}
\hline
1,1  & 1,2  & 1,3  & 1,4      \bigstrut \\ \hline
2,1  &
  \multirow{2}[4]{1.5cm}{Four bigstruts} &
    \multirow{3}[6]{*}{Six bigstruts} &
      \multirow{3}[6]{*}[1ex]{Six bigstruts and fixup}
                               \bigstrut \\ \cline{1-1}
3,1  &      &      &          \bigstrut \\ \cline{1-2}
4,1  & 4,2  &      &          \bigstrut \\ \hline
5,1  &
```

```
  \multirow{2}[2]{*}{Two bigstruts}
          & 5,3
                & 5,4       \bigstrut[t] \\
6,1  &    & 6,3  & 6,4       \bigstrut[b] \\ \hline
7,1  &
  \multicolumn{2}{l|}{\multirow{2}[4]{*}{Four bigstruts}}
                & 7,4       \bigstrut \\ \cline{1-1}\cline{4-4}
8,1  &
  \multicolumn{2}{l|}{}
                & 8,4       \bigstrut \\ \hline
9,1  & 9,2  & 9,3  & 9,4       \bigstrut \\ \hline
\end{tabular}
```

1,1	1,2	1,3	1,4
2,1	Four bigstruts	Six bigstruts	Six bigstruts and fixup
3,1			
4,1	4,2		
5,1 6,1	Two bigstruts	5,3 6,3	5,4 6,4
7,1	Four bigstruts		7,4
8,1			8,4
9,1	9,2	9,3	9,4

7.1 Multiple Rows with Text Wrapping

Multiple row and text wrap column specifiers (see §5) do not mix well because `multirow` is left justified, and the the `p{}`, `m{}` and `b{}` formats are normally justified. This is shown in the following example, which generates two 'bad box' warnings:

```
\setlength{\extrarowheight}{1.5pt}
\begin{tabular}{|c|p{2cm}|}
\hline
1,1 & ** text **    \\\hline
2,1 & \multirow{2}{2cm}{** text **}
                    \\\cline{1-1}
3,1 &               \\\cline{1-1}
4,1 &               \\\cline{1-1}
5,1 &               \\\hline
6,1 & ** text **    \\\hline
\end{tabular}
```

1,1	Lorem ipsum dolor sit amet.
2,1	Lorem ipsum dolor sit amet consectetur.
3,1	
4,1	
5,1	
6,1	Lorem ipsum dolor sit amet.

The easiest way to resolve this is to make everything flush left by defining a ragged right column type as described in §6:

```
\newcolumntype{P}[1]{>{\raggedright}p{#1}}
\newcolumntype{M}[1]{>{\raggedright}m{#1}}

\setlength{\extrarowheight}{1.5pt}
\begin{tabular}{|c|P{2cm}|} % could be |c|M{2cm}|
```

```
\hline
1,1 & \muchlessText \tn\hline
2,1 & \multirow{2}{2cm}[-1.5pt]{\lessText }
                    \tn\cline{1-1}
3,1 &               \tn\cline{1-1}
4,1 &               \tn\cline{1-1}
5,1 &               \tn\hline
6,1 & \muchlessText \tn\hline
\end{tabular}
```

which creates the following layouts:

With `P{}`

1,1	Lorem ipsum dolor sit amet.
2,1	Lorem ipsum dolor sit amet consectetur.
3,1	
4,1	
5,1	
6,1	Lorem ipsum dolor sit amet.

With `M{}`

1,1	Lorem ipsum dolor sit amet.
2,1	Lorem ipsum dolor sit amet consectetur.
3,1	
4,1	
5,1	
6,1	Lorem ipsum dolor sit amet.

7.2 Over Sized Spanning Rows

Having a `\multirow` that has more lines than the rows it spans is not so easy to layout well. Expanding the spanned rows with struts appears to be the only solution, but it requires a lot of trial-and-error adjustments. The following table shows the method:

```
\newlength{\rowA}
\setlength{\rowA}{8ex} % modify as needed
\newcommand{\strutA}{% no space before strut
\rule[-0.45\rowA]{0pt}{\rowA}%  put text approx mid strut
}
...
\begin{tabular}{|c|l|c|}
\hline
1,1  & 1,2  & 1,3   \bigstrut \\ \hline
2,1  & \multirow{2}{5cm}[1ex]{ ... lots of text ... }
            & 2,3\strutA
                                \\ \cline{1-1}\cline{3-3}
3,1  &      & 3,3\strutA      \\ \hline
4,1  & 4,2  & 4,3   \bigstrut \\ \hline
\end{tabular}
```

1,1	1,2	1,3
2,1	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur id nisl nunc, non adipiscing arcu. Morbi nec leo sit amet.	2,3
3,1		3,3
4,1	4,2	4,3

A `\rule` of zero width is used for the strut. A negative raise height positions the text in the rows vertical centre. In the example, the strut is put in a centre aligned column. In this context it must be put next to the column's text with no spaces, or the alignment will be disturbed. This is not normally an issue for other alignments.

For convenience the strut is defined as a command called `\strutA`, which is used in the spanned rows 2 and 3. The height of the strut is given by the length `rowA`. This is more complicated to write, but makes the trial-and-error layout process easier. Adjust `rowA` until there is room for the spanning row's text. The `fixup` parameter in the `\multirow` command is used to correct the vertical position of its text as necessary.

8 Table Headings

8.1 Heading Alignment

It is sometimes nice to have a column heading with a different alignment to its column. The standard `multicolumn` command can be used to do this:

```
\begin{tabular}{|l|l|l|}
\hline
\multicolumn{1}{|c}{C1}&
   \multicolumn{1}{|c}{C2}&
      \multicolumn{1}{|c|}{C2}\\\hline
aa & b & cccc \\\hline
aaa & bb & cccc \\\hline
aaaa & bbb & cccc \\\hline
aaaaa & bbbb & cccc \\\hline
aaaaaa & bbbb & cccc \\\hline
\end{tabular}
```

C1	C2	C2
aa	b	c
aaa	bb	cc
aaaa	bbb	ccc
aaaaa	bbbb	cccc
aaaaaa	bbbb	cccc

8.2 Sideways Headings

Horizontal space can be saved by printing long column headings sideways. The `rotating` package [6] can be used to achieve this with its `sideways` environment:

```
\usepackage{rotating}
...
\begin{tabular}{|l|l|l|}
\hline
\multicolumn{1}{|c|}{
   \begin{sideways}
      Column 1 \,
   \end{sideways}}&
      \multicolumn{1}{c|}{
         \begin{sideways}
            Column 22 \,
         \end{sideways}}&
            \multicolumn{1}{c|}{
               \begin{sideways}
                  Column 333 \,
               \end{sideways}} \\\hline
aa & bbbb & cccc \\\hline
aaa & bbbb & cccc \\\hline
aaaa & bbbb & cccc \\\hline
aaaaa & bbbb & cccc \\\hline
aaaaaa & bbbb & cccc \\\hline
\end{tabular}
```

Column 1	Column 22	Column 333
aa	bbbb	cccc
aaa	bbbb	cccc
aaaa	bbbb	cccc
aaaaa	bbbb	cccc
aaaaaa	bbbb	cccc

The `multicolumn` method given in §8.1 is used to centre the headings; and `\,` is used to add a little space at the top of the heading.

9 Specifying Table Width

The `tabularx` [3] and `tabulary` [4] packages are much better than the standard `tabular*` for specifying table width.

9.1 Tabularx

The `tabularx` environment expands specific columns to meet the table's width requirement. The width of the table is given as a parameter, and the columns that can be expanded are denoted with the `X` alignment command, as the following shows:

```
\begin{center}
\setlength{\extrarowheight}{1.5pt}
\begin{tabularx}{0.75\textwidth}{|l|X|}
\hline
1,1 & ** some text ** \\ \hline
2,1 & ** some text ** \\ \hline
\end{tabularx}
\end{center}
```

1,1	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur id nisl nunc, non adipiscing arcu. Morbi nec leo sit amet.
2,1	Nullam rhoncus, sem luctus ultrices accumsan, urna enim fermentum.

There must be at least one `X` column. If there is more than one `X` column the necessary space is equally distributed. The columns are always padded to give the the table its specified width. Text is wrapped and justified if it does not fit into the column.

Note the use of `0.75\textwidth` to specify the width as a proportion of page width.

9.2 Tabulary

The `tabulary` environment expands specific columns to meet the table's width requirement and allows alignment to be specified for these columns as follows:

`L`	`\raggedright`
`C`	`\centering`
`R`	`\raggedleft`
`J`	normal justification

The maximum width for the table is given as a parameter. However, unlike `tabularx`, columns are not padded if they are too narrow.

```
\setlength{\extrarowheight}{1.5pt}
\begin{tabulary}{4cm}{|l|L|}
\hline
1,1 & ** some text ** \\ \hline
2,1 & ** some text ** \\ \hline
\end{tabulary}
```

full width ...

1,1	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
2,1	Curabitur id nisl nunc, non adipiscing arcu.

not full width ...

1,1	Lorem ipsum.
2,1	Curabitur id.

9.3 Comparison of Methods

The `tabularx` package is useful for absolute table widths, but it has limited alignment options.

The `tabulary` package provides better alignment options; and its adaptive width behaviour does not normally cause problems. It is convenient to set its width at a suitable maximum, say `0.75\textwidth`, and let the package select appropriate column widths.

10 Larger Tables

Larger tables can sometimes be handled by turning them sideways, or by letting them span pages.

10.1 Sideways

The most versatile way to turn a table sideways is to use the `sidways` environment from the `rotate` package [6]. For example:

```
\usepackage{rotating}
...
\begin{sideways}
\begin{tabular}{|l|l|}
\hline
1,1 & 1,2\\ \hline
2,1 & 2,2\\ \hline
\end{tabular}
\end{sideways}
```

1,1	1,2
2,1	2,2

This method can be used in a floating table environment or just embedded in the text.

10.2 Longtable

The `longtable` package is designed to make tables that span page breaks. It is rather complicated to use, and its documentation [2] should be consulted for all of its features. It maintains column widths across page breaks, and centres the table. There are reports of incompatibility with many other packages, but for simple use it appears to be okay. Multiple compilation passes are normally needed to get the layout correct. Here is a very simple example:

```
\usepackage{longtable}
\usepackage{array} % for extrarowheight

\setlength{\extrarowheight}{1.5pt}
\begin{longtable}{|l|l|}
% header ------------------------
\hline
Heading 1 & Heading 2 \\\hline\hline
\endhead
% header ------------------------

Lorem ipsum ...  &  Consectetur ... \\\hline
Lorem ipsum ...  &  Consectetur ... \\\hline
 .
 .
\end{longtable}
```

Heading 1	Heading 2
Lorem ipsum dolor sit amet.	Consectetur adipiscing elit.
Lorem ipsum dolor sit amet.	Consectetur adipiscing elit.
Lorem ipsum dolor sit amet.	Consectetur adipiscing elit.

```
-------- Page break --------
```

Heading 1	Heading 2
Lorem ipsum dolor sit amet.	Consectetur adipiscing elit.
Lorem ipsum dolor sit amet.	Consectetur adipiscing elit.

In the example above, `\extrarowheight` is used to improve the table's vertical spacing. Alternatively `arraystretch` could be use if space is needed below the lines. However, the normally useful `bigstrut` package does not work with `longtable`. See §3.1 for more about these methods.

11 Footnotes in Tables

A footnote in a non-floating `longtable` or `tabulary` table is put with any normal page footnotes at the end of the page containing the table. When a `longtable` breaks over a page, footnotes are placed on the correct page. Table and page footnotes have the same counter, and are numbered from the start of the document. Floating tables cannot have normal footnotes.

Footnotes can be kept next to their table by using a `minipage` to hold the table. Each `minipage` has its own disjoint footnote counter, and uses alphabetic lower case footnote marks. This method does not work for `tabulary`, but it can be used with floating tables. The following is a `tabular` example:

```
\begin{minipage}{6cm}
\begin{tabular}{|l|l|}
\hline
1,2 & 1,2\footnote{This is a footnote.}
                          \\\hline
2,1 & 2,2                 \\\hline
3,1 & 3,2                 \\\hline
\end{tabular}
\end{minipage}
```

1,2	1,2[a]
2,1	2,2
3,1	3,2

[a] This is a footnote.

12 Professional Layout

The `booktabs` package [7] provides support for 'formal tables,' which the package's author promotes as a better way of presenting data. In particular, he derides the use of vertical rules and double rules. Here is an example taken from the package documentation:

```
\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}
```

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

As this shows, the package provides better vertical spacing around horizontal rules.

There is also the derived `ctable` package [5], which focuses on table and figure floats, and provides facilities for making table footnotes. Lapo Mori's article *Tables in LaTeX 2ε: Packages and Methods* [9] has a detailed discussion of good table layout and the `ctable` and `booktab` packages.

References

[1] David Carlisle, *The dcolumn package*. Download from www.ctan.org/tex-archive/macros/latex/required/tools/dcolumn.pdf

[2] David Carlisle, *The longtable package*. Download from www.ctan.org/tex-archive/macros/latex/required/tools/longtable.pdf

[3] David Carlisle, *The tabularx package*. Download from www.ctan.org/tex-archive/macros/latex/required/tools/tabularx.pdf

[4] David Carlisle, *The tabulary package*. Download from www.ctan.org/tex-archive/macros/latex/contrib/tabulary/tabulary.pdf

[5] Wybo Dekker, *The ctable package*. Downloadable from www.ctan.org/tex-archive/macros/latex/contrib/ctable/ctable.pdf

[6] Robin Fairbairns, Sebastian Rahtz and Leonor Barroca, *A package for rotated objects in LaTeX*. Downloadable from www.ctan.org/tex-archive/macros/latex/contrib/rotating/rotating.pdf

[7] Simon Fear, *Publication quality tables in LaTeX*. Download from www.ctan.org/tex-archive/macros/latex/contrib/booktabs/booktabs.pdf

[8] Frank Mittelbach and David Carlisle, *A new implementation of LaTeX's tabular and array environment*. Downloadable from www.ctan.org/tex-archive/macros/latex/required/tools/array.pdf

[9] Lapo Filippo Mori, *Tables in LaTeX 2ε: Packages and Methods*, The PracTeX Journal, 2007, No. 1. Downloadable from tug.org/pracjourn/2007-1/mori/mori.pdf

[10] Tobias Oetiker, *The Not So Short Introduction to LaTeX 2ε*. Downloadable from www.ctan.org/tex-archive/info/lshort/english/lshort.pdf

[11] Piet van Oostrum, Øystein Bache and Jerry Leichter, *The multirow, bigstrut and bigdelim packages*. Downloadable from www.ctan.org/tex-archive/macros/latex/contrib/multirow/doc/multirow.pdf

[12] Adrian Robson, *LaTeX Hints and Tips*. Download from nepsweb.co.uk/docs/latexTricks.pdf

[13] The TeX Users Group (TUG), *The pdfTeX project page*. Available at www.tug.org/applications/pdftex