Contents

1 Standard Tables 2

2 Spacing 2
  2.1 Row Spacing ........................................... 2
    2.1.1 Arraystretch ................................... 2
    2.1.2 Extrapageheight ................................. 3
    2.1.3 Bigstruts ......................................... 3
    2.1.4 Comparison of Methods ......................... 4
  2.2 Column Spacing ..................................... 4

3 Vertical Alignment and Text Wrapping 5

4 Ragged Right Alignment 5
  4.1 Simple Command ...................................... 5
  4.2 Column Types ........................................ 6
  4.3 Tabular .............................................. 6
  4.4 Comparison of Methods ................................ 7

5 Multiple Rows 7
  5.1 Multiple Rows with Text Wrapping .................. 8
  5.2 Over Sized Spanning Rows ......................... 9

6 Specifying Table Width 10
  6.1 Tabularx ............................................ 10
  6.2 Tabulary ............................................. 11
  6.3 Comparison of Methods ............................. 12

7 Larger Tables 12
  7.1 Sideways ............................................ 12
  7.2 Longtable ........................................... 12

8 Footnotes in Tables 13

9 Professional Layout 13
1 Standard Tables

Tables are made in \LaTeX\ using the \texttt{tabular} environment like this

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

A Table

1,1 1,2 1,3
2,1 2,2 2,3
3,1 3,2 3,3

For more information on basic tables consult a good text like \textit{The Not So Short Introduction to \LaTeX\ 2\epsilon} [9].

2 Spacing

2.1 Row Spacing

The standard row height is too small when \texttt{\hline} is used. There are three ways of correcting this: modifying the \texttt{\arraystretch} command, changing the \texttt{\extrarowheight} length, or using the \texttt{\bigstrut} command.

2.1.1 Arraystretch

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

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

The \texttt{\renewcommand} and the table are enclosed in braces to limit the scope of the redefinition. Its effect is to turn

\begin{tabular}{|c|l|}
\hline
a & Row 1 \\
b & Row 2 \\
c & Row 3 \\
d & Row 4 \\
\hline
\end{tabular}

into

\begin{tabular}{|c|l|}
\hline
a & Row 1 \\
b & Row 2 \\
c & Row 3 \\
d & Row 4 \\
\hline
\end{tabular}

and with a bigger value

\begin{tabular}{|c|l|}
\hline
a & Row 1 \\
b & Row 2 \\
c & Row 3 \\
d & Row 4 \\
\hline
\end{tabular}
2.1.2 Extrarowheight

A more sophisticated approach is to use the \texttt{array} package [8] and change the length \texttt{extrarowheight} as follows:

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

This adds space only above the rows’ text, which with the correct value compensates for the \texttt{\hline} commands. Its effect is to turn

\begin{verbatim}
\begin{tabular}{|l|l|}
\hline
a & Row 1 \bigstrut \bigstrut[t] \hline
b & Row 2 \bigstrut \bigstrut[b] \hline
c & Row 3 \bigstrut \bigstrut[t] \hline
d & Row 4 \bigstrut \bigstrut[b] \hline
\end{tabular}
\end{verbatim}

2.1.3 Bigstruts

The above methods apply the compensation to all rows, even if they do not have \texttt{\hline} commands. A subtler alternative is to use the \texttt{bigstrut} package [6] like this:

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

which changes the basic table

\begin{verbatim}
\begin{tabular}{|l|l|}
\hline
a & Row 1 \bigstrut[b] \hline
b & Row 2 \bigstrut[b] \hline
c & Row 3 \bigstrut\bigstrut[t] \hline
d & Row 4 \bigstrut[\bigstrut] \hline
\end{tabular}
\end{verbatim}
The \texttt{bigstrut} command is used when there is an \texttt{hline} above and below: \texttt{bigstrut[t]} when there is an \texttt{hline} only above; and \texttt{bigstrut[b]} when this is only one below.

The \texttt{bigstrut} package only works well with tables that have single line cells. It does not work with the text wrap column specifiers \texttt{p{}}, \texttt{m{}} and \texttt{b{}}. (For more information on text wrap see §3).

2.1.4 Comparison of Methods

The effect of the three methods is as follows:

\begin{tabular}{|l|l|l|}
\hline
arraystretch{1.2} & extrarowheight{1.5pt} & bigstrut \\
\hline
a & Row 1 & a & Row 1 \\
b & Row 2 & b & Row 2 \\
c & Row 3 & c & Row 3 & c & Row 2 \\
d & Row 4 & d & Row 4 & d & Row 4 \\
\hline
\end{tabular}

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

2.2 Column Spacing

Column width can be modified by changing \texttt{\tabcolsep} like this:

\setlength{\tabcolsep}{10pt}
\setlength{\extrarowheight}{1.5pt}
\begin{tabular}{|l|l|}
\hline
a & Row 1 \\
b & Row 2 \\
c & Row 3 \\
\hline
\end{tabular}

which changes the default

\begin{tabular}{|l|l|}
\hline
a & Row 1 \\
b & Row 2 \\
c & Row 3 \\
\hline
\end{tabular}

There is a standard column specifier \texttt{@{cmnd}}, which suppresses inter-column space and inserts \texttt{cmnd} instead. This can be used to insert or remove space into a particular column. For example:
Vertical alignment can be controlled with the \texttt{array} package [8], which has additional text wrap formatting commands:

\begin{itemize}
\item \texttt{p\{width\}}: Top align, the same as usual.
\item \texttt{m\{width\}}: Middle align
\item \texttt{b\{width\}}: Bottom align
\end{itemize}

These produce the following layouts:

\begin{tabular}{|@{\hspace{1cm}}l|@{}l|}
\hline
Column Format & Column Format & Column Format \\
p\{11\} & p\{22\} & 1 \\
11 & 22 & 33 \\
11 & 22 & 33 \\
11 & 22 & 33 \\
\hline
\end{tabular}

Notice how the \texttt{m\{\}} or \texttt{b\{\}} alignment affects the whole table. In addition, \texttt{p\{\}} m\{\} and \texttt{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 \texttt{\raggedright} (see §4).

\section{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 \texttt{tabular} 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.

\subsection{Simple Command}

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

\begin{tabular}{|@{\hspace{1cm}}l|@{}l|}
\hline
Abcd & Abcd \\
Abcd & Abcd \\
\hline
\end{tabular}
Using this method 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.

### 4.2 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:

```latex
\usepackage{array}

\newcolumntype{x}[1]{>{\raggedright}p{#1}}
\newcommand{\tn}{\tabularnewline}

\begin{tabular}{|c|x{5cm}|}
\hline
1,1 & ... text ... \tn \hline
2,1 & ... text ... \hline
\end{tabular}
```

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. The formatting could have been embedded in the tabular heading. Again `\tabularnewline` is needed when `\raggedright` is used.

### 4.3 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. See §6.2
\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.

4.4 Comparison of Methods

Simple commands are useful for small tables or when the formatting does not apply to the whole column. The \texttt{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.

5 Multiple Rows

The easiest way to have tables with spanning rows is to use the \texttt{multirow} package \cite{multirow}. In its simplest form it can be used like this:

\usepackage{multirow}
...
\begin{tabular}{|c|c|}
\hline
a & Row 1 \\
\hline
b & \texttt{\multirow{2}{*}{Spanning rows}} \\
\hline
c & \\
\hline
d & Row 4 \\
\hline
\end{tabular}
giving the following:

\begin{tabular}{|c|}
\hline
a & Row 1 \\
\hline
b & Spanning rows \\
\hline
c & \\
\hline
d & Row 4 \\
\hline
\end{tabular}

The \texttt{\multirow} command declares the location of the spanning rows. Its first argument is the number of rows to span. The second states, in this case, 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 \texttt{bigstrut} package is used, the number of struts in the spanned rows should be stated as the \texttt{bigstruts} parameter. Count 2 for every \texttt{\bigstrut} and 1 for a \texttt{\bigstrut[t]} or \texttt{\bigstrut[b]}.

The text width can be set with the \texttt{width} parameter, in which case the text will be wrapped and left justified. Line breaks can be forced with a \texttt{\\}.
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 \multicolumn 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 \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 \bigstrut \cline{1-1}
3,1 & & & \bigstrut \bigstrut \cline{1-2}
4,1 & 4,2 & & \bigstrut \bigstrut \hline
5,1 & \multirow{2}[2]{*}{Two bigstruts}
& & 5,3 & 5,4 \bigstrut[t] \bigstrut \hline
6,1 & 6,3 & 6,4 & \bigstrut[b] \hline
7,1 & \multicolumn{2}{l|}{\multirow{2}[4]{*}{Four bigstruts}} & 7,4 & \bigstrut \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}

5.1 Multiple Rows with Text Wrapping

Multiple row and text wrap column specifiers (see §3) 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:

\begin{tabular}{|c|p{2cm}|}
\hline
1,1 & ** text ** \\
\hline
2,1 & \multicolumn{2}{c}{** text **} \\
\cline{1-1}
3,1 & \\
\cline{1-1}
4,1 & \\
\cline{1-1}
5,1 & \\
\cline{1-1}
6,1 & ** text ** \\
\hline
\end{tabular}

The easiest way to resolve this is to make everything flush left by defining a ragged right column type as described in §4:

\begin{tabular}{|c|P{2cm}|} % could be |c|M{2cm}|
\hline
1,1 & \muchlessText \\
\hline
2,1 & \multicolumn{2}{c}{\-1.5pt\\lessText } \\
\cline{1-1}
3,1 & \\
\cline{1-1}
4,1 & \\
\cline{1-1}
5,1 & \\
\cline{1-1}
6,1 & \muchlessText \\
\hline
\end{tabular}

which creates the following layouts:

5.2 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
}\end{tabular}{|c|l|c|}
\hline
1,1 & 1,2 & 1,3 \bigstrut \hline
2,1 & \multirow{2}{5cm}{... lots of text ... } & 2,3 \strutA \hline
3,1 & & 3,3 \strutA \hline
4,1 & 4,2 & 4,3 \bigstrut \hline
\end{tabular}

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.

6 Specifying Table Width


6.1 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 \texttt{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}

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

Note the use of \texttt{0.75\textwidth} to specify the width as a proportion of page width.

6.2 Tabulary

The \texttt{tabulary} environment expands specific columns to meet the table’s width requirement and allows alignment to be specified for these columns as follows:

- \texttt{L} \texttt{\raggedright}
- \texttt{C} \texttt{\centering}
- \texttt{R} \texttt{\raggedleft}
- \texttt{J} normal justification

The maximum width for the table is given as a parameter. However, unlike \texttt{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 . . . not full width . . .

<table>
<thead>
<tr>
<th>1,1</th>
<th>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</th>
<th>1.1</th>
<th>Lorem ipsum.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,1</td>
<td>Curabitur id nisl nunc, non adipiscing arcu.</td>
<td>2.1</td>
<td>Curabitur id.</td>
</tr>
</tbody>
</table>
6.3 Comparison of Methods

The \texttt{tabularx} package is useful for absolute table widths, but it has limited alignment options.

The \texttt{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 \texttt{0.75\textwidth}, and let the package select appropriate column widths.

7 Larger Tables

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

7.1 Sideways

The easiest way the turn a table sideways is to use the \texttt{rotate} package \cite{rotate}. For example:

\begin{verbatim}
\usepackage{rotate}
\begin{sideways}
\begin{tabular}{|l|l|}
\hline
1,1 & 1,2 \\
\hline
2,1 & 2,2 \\
\hline
\end{tabular}
\end{sideways}
\end{verbatim}

7.2 Longtable

The \texttt{longtable} package is designed to make tables that span page breaks. It is rather complicated to use, and the primary documentation \cite{longtable} should be consulted for all of its features. It maintains column widths across page breaks, and centers the table. It is reported to be incompatible 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:

\begin{verbatim}
\usepackage{longtable}
\usepackage{array} \% for extrarowheight

\setlength{\extrarowheight}{1.5pt}
\begin{longtable}{|l|l|}
\hline
\endhead
Heading 1 & Heading 2 \hline
\end{longtable}
\end{verbatim}
8 Footnotes in Tables

There are problems with tables and footnotes. They work with \texttt{longtable} or \texttt{tabularx} environments; but they do not work with \texttt{tabular} and \texttt{tabulary} environments.

Footnotes in a \texttt{longtable} or \texttt{tabulary} table are put at the end of the table’s page with any normal footnotes. When a \texttt{longtable} breaks over a page, footnotes are placed on the correct page.

Footnotes can be kept next to their originating table by using a \texttt{minipage}. However this does not work for \texttt{tabulary}. This is a \texttt{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}

\ \hline
1,2 & 1,2\footnote{This is a footnote.} \\
\hline
2,1 & 2,2 \\
\hline
3,1 & 3,2 \\
\hline

9 Professional Layout

The \texttt{booktabs} package \cite{booktabs} 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 form the package documentation:

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

References


[8] Frank Mittelbach and David Carlisle, *A new implementation of B\LaTeX\’s tabular and array environment*, Comprehensive \TeX\ Archive Network (CTAN), 2006. ([array.pdf](http://www.ctan.org))