LaTeX/Tables

Tables are a common feature in academic writing, often used to summarise research results. Mastering the art of table construction in LaTeX is therefore necessary to produce quality papers and with sufficient practice one can print beautiful tables of any kind.

Keeping in mind that LaTeX is not a spreadsheet, it makes sense to use a dedicated tool to build tables and then to export these tables into the document. Basic tables are not too taxing, but anything more advanced can take a fair bit of construction; in these cases, more advanced packages can be very useful. However, first it is important to know the basics. Once you are comfortable with basic LaTeX tables, you might have a look at more advanced packages or the export options of your favorite spreadsheet. Thanks to the modular nature of LaTeX, the whole process can be automated in a fairly comfortable way.

For a long time, LaTeX tables were quite a chaotic topic, with dozens of packages doing similar things, while not always being compatible with one another. Sometimes you had to make trade-offs. The situation changed recently (2010) with the release of the `tabu` package which combines the power of `longtable`, `tabularx` and much more. The `tabu` environment is far less fragile and restricted than the older alternatives. Nonetheless, before attempting to use this package for the first time it will be beneficial to understand how the classic environment works, since `tabu` works the same way. Note however that the author of `tabu` will not fix bugs to the current version, and that the next version introduces new syntax that will likely break existing documents.[1]

Contents

The *tabular* environment

The `tabular` environment can be used to typeset tables with optional horizontal and vertical lines. LaTeX determines the width of the columns automatically.

The first line of the environment has the form:

```
\begin{tabular}[pos]{table spec}
```

The `table spec` argument tells LaTeX the alignment to be used in each column and the vertical lines to insert.

The number of columns does not need to be specified as it is inferred by looking at the number of arguments provided. It is also possible to add vertical lines between the columns here. The following symbols are available to describe the table columns (some of them require that the package `array` has been loaded):

`l`	left-justified column
`c`	centered column
`r`	right-justified column
`p{'width'}`	paragraph column with text vertically aligned at the top
`m{'width'}`	paragraph column with text vertically aligned in the middle (requires array package)
`b{'width'}`	paragraph column with text vertically aligned at the bottom (requires array package)
`|`	vertical line
`||`	double vertical line

By default, if the text in a column is too wide for the page, LaTeX won't automatically wrap it. Using `p{'width'}` you can define a special type of column which will wrap-around the text as in a normal paragraph. You can pass the width using any unit supported by LaTeX, such as 'pt' and 'cm', or *command lengths*, such as `\textwidth`. You can find a list in chapter Lengths.

The optional parameter `pos` can be used to specify the vertical position of the table relative to the baseline of the surrounding text. In most cases, you will not need this option. It becomes relevant only if your table is not in a paragraph of its own. You can use the following letters:

`b`	bottom
`c`	center (default)
`t`	top

To specify a font format (such as bold, italic, etc.) for an entire column, you can add `>{\format}` before you declare the alignment. For example `\begin{tabular}{ >{\bfseries}l c >{\itshape}r }` will indicate a three column table with the first one aligned to the left and in bold font, the second one aligned in the center and with normal font, and the third aligned to the right and in italic. The "array" package needs to be activated in the preamble for this to work.

In the first line you have pointed out how many columns you want, their alignment and the vertical lines to separate them. Once in the environment, you have to introduce the text you want, separating between cells and introducing new lines. The commands you have to use are the following:

`&`	column separator
`\\`	start new row (additional space may be specified after `\\` using square brackets, such as `\\[6pt]`)
`\hline`	horizontal line
`\newline`	start a new line within a cell (in a paragraph column)
`\cline{i-j}`	partial horizontal line beginning in column *i* and ending in column *j*

Note, any white space inserted between these commands is purely down to ones' preferences. I personally add spaces between to make it easier to read.

Basic examples

This example shows how to create a simple table in LaTeX. It is a three-by-three table, but without any lines.

```
\begin{tabular}{ l c r }
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9 \\
\end{tabular}
```

1	2	3
4	5	6
7	8	9

Expanding upon that by including some vertical lines:

```
\begin{tabular}{ l | c || r }
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9 \\
\end{tabular}
```

1	2	3
4	5	6
7	8	9

To add horizontal lines to the very top and bottom edges of the table:

```
\begin{tabular}{ l | c || r }
  \hline
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9 \\
  \hline
\end{tabular}
```

1	2	3
4	5	6
7	8	9

And finally, to add lines between all rows, as well as centering (notice the use of the center environment - of course, the result of this is not obvious from the preview on this web page):

```
\begin{center}
  \begin{tabular}{ l | c || r }
    \hline
    1 & 2 & 3 \\ \hline
    4 & 5 & 6 \\ \hline
    7 & 8 & 9 \\
    \hline
  \end{tabular}
\end{center}
```

1	2	3
4	5	6
7	8	9

```
\begin{tabular}{|r|l|}
  \hline
  7C0 & hexadecimal \\
  3700 & octal \\ \cline{2-2}
  11111000000 & binary \\
  \hline \hline
  1984 & decimal \\
  \hline
\end{tabular}
```

7C0	hexadecimal
3700	octal
11111000000	binary
1984	decimal

Text wrapping in tables

LaTeX's algorithms for formatting tables have a few shortcomings. One is that it will not automatically wrap text in cells, even if it overruns the width of the page. For columns that will contain text whose length exceeds the column's width, it is recommended that you use the `p` attribute and specify the desired width of the column (although it may take some trial-and-error to get the result you want). For a more convenient method, have a look at The tabularx package, or The tabulary package.

Instead of `p`, use the `m` attribute to have the lines aligned toward the middle of the box or the `b` attribute to align along the bottom of the box.

Here is a simple example. The following code creates two tables with the same code; the only difference is that the last column of the second one has a defined width of 5 centimeters, while in the first one we didn't specify any width. Compiling this code:

```
\documentclass{article}
\usepackage[english]{babel}

\begin{document}

Without specifying width for last column:
\begin{center}
    \begin{tabular}{| l | l | l | l |}
    \hline
    Day & Min Temp & Max Temp & Summary \\ \hline
    Monday & 11C & 22C & A clear day with lots of sunshine.
    However, the strong breeze will bring down the temperatures. \\ \hline
    Tuesday & 9C & 19C & Cloudy with rain, across many northern regions. Clear spells
    across most of Scotland and Northern Ireland,
    but rain reaching the far northwest. \\ \hline
    Wednesday & 10C & 21C & Rain will still linger for the morning.
    Conditions will improve by early afternoon and continue
    throughout the evening. \\
    \hline
    \end{tabular}
\end{center}

With width specified:
\begin{center}
```

```
        \begin{tabular}{ | l | l | l | p{5cm} |}
        \hline
        Day & Min Temp & Max Temp & Summary \\ \hline
        Monday & 11C & 22C & A clear day with lots of sunshine.
        However, the strong breeze will bring down the temperatures. \\ \hline
        Tuesday & 9C & 19C & Cloudy with rain, across many northern regions. Clear spells
        across most of Scotland and Northern Ireland,
        but rain reaching the far northwest. \\ \hline
        Wednesday & 10C & 21C & Rain will still linger for the morning.
        Conditions will improve by early afternoon and continue
        throughout the evening. \\
        \hline
        \end{tabular}
    \end{center}

    \end{document}
```

You get the following output:

Without specifying width for last column:

Day	Min Temp	Max Temp	Summary
Monday	11C	22C	A clear day with lots of sunshine. However, the strong breeze w
Tuesday	9C	19C	Cloudy with rain, across many northern regions. Clear spells ac
Wednesday	10C	21C	Rain will still linger for the morning. Conditions will improve by

With width specified:

Day	Min Temp	Max Temp	Summary
Monday	11C	22C	A clear day with lots of sunshine. However, the strong breeze will bring down the temperatures.
Tuesday	9C	19C	Cloudy with rain, across many northern regions. Clear spells across most of Scotland and Northern Ireland, but rain reaching the far northwest.
Wednesday	10C	21C	Rain will still linger for the morning. Conditions will improve by early afternoon and continue throughout the evening.

Note that the first table has been cropped, since the output is wider than the page width.

Manually broken paragraphs in table cells

Sometimes it is necessary to not rely on the breaking algorithm when using the `p` specifier, but rather specify the line breaks by hand. In this case it is easiest to use a `\parbox`:

```
    \begin{tabular}{cc}
      boring cell content & \parbox[t]{5cm}{rather long par\\new par}
    \end{tabular}
```

Space between columns

To tweak the space between columns (LaTeX will by default choose very tight columns), one can alter the column separation: `\setlength{\tabcolsep}{5pt}`. The default value is 6pt.

Space between rows

Re-define the `\arraystretch` command to set the space between rows:

```
    \renewcommand{\arraystretch}{1.5}
```

Default value is 1.0.

An alternative way to adjust the rule spacing is to add `\noalign{\smallskip}` before or after the `\hline` and `\cline{i-j}` commands:

```
\begin{tabular}{ | l | l | r | }
  \hline\noalign{\smallskip}
  \multicolumn{2}{c}{Item} \\
  \cline{1-2}\noalign{\smallskip}
  Animal & Description & Price (\$) \\
  \noalign{\smallskip}\hline\noalign{\smallskip}
  Gnat  & per gram & 13.65 \\
        & each     &  0.01 \\
  Gnu   & stuffed  & 92.50 \\
  Emu   & stuffed  & 33.33 \\
  Armadillo & frozen & 8.99 \\
  \noalign{\smallskip}\hline
\end{tabular}
```

You may also specify the skip after a line explicitly using glue after the line terminator

```
\begin{tabular}{ll}
\hline
Mineral & Color \\[1cm]
Ruby & red \\
Sapphire & blue \\
\hline
\end{tabular}
```

Other environments inside tables

If you use some LaTeX environments inside table cells, like `verbatim` or `enumerate`:

```
\begin{tabular}{c c}
        \hline
        \begin{verbatim}
        code
        \end{verbatim}
        & description
        \\ \hline
\end{tabular}
```

you might encounter errors similar to

```
! LaTeX Error: Something's wrong--perhaps a missing \item.
```

To solve this problem, change column specifier to "paragraph" (`p`, `m` or `b`).

```
\begin{tabular}{m{5cm} c}
```

Defining multiple columns

It is possible to define many identical columns at once using the `*{''num''}{''str''}` syntax. This is particularly useful when your table has many columns.

Here is a table with six centered columns flanked by a single column on each side:

```
\begin{tabular}{l*{6}{c}r}
Team              & P & W & D & L & F  & A & Pts \\
\hline
Manchester United & 6 & 4 & 0 & 2 & 10 & 5 & 12  \\
Celtic            & 6 & 3 & 0 & 3 &  8 & 9 &  9  \\
Benfica           & 6 & 2 & 1 & 3 &  7 & 8 &  7  \\
FC Copenhagen     & 6 & 2 & 1 & 2 &  5 & 8 &  7  \\
\end{tabular}
```

Team	P	W	D	L	F	A	Pts
Manchester United	6	4	0	2	10	5	12
Celtic	6	3	0	3	8	9	9
Benfica	6	2	1	3	7	8	7
FC Copenhagen	6	2	1	2	5	8	7

Column specification using *>{\cmd}* and *<{\cmd}*

The column specification can be altered using the array package. This is done in the argument of the tabular environment using >{\command} for commands executed right *before* each column element and <{\command} for commands to be executed right *after* each column element. As an example: to get a column in math mode enter: \begin{tabular}{>{$}c<{$}}. Another example is changing the font: \begin{tabular}{>{\small}c} to print the column in a small font.

The argument of the > and < specifications must be correctly balanced when it comes to { and } characters. This means that >{\bfseries} is valid, while >{\textbf} will not work and >{\textbf{} is not valid. If there is the need to use the text of the table as an argument (for instance, using the \textbf to produce bold text), one should use the \bgroup and \egroup commands: >{\textbf\bgroup}c<{\egroup} produces the intended effect. This works only for some basic LaTeX commands. For other commands, such as \underline to underline text, it is necessary to temporarily store the column text in a box using lrbox. First, you must define such a box with \newsavebox{\boxname} and then you can define:

```
>{\begin{lrbox}{\boxname} }%
l%
<{\end{lrbox}%
  \underline{\unhbox\boxname} }%
 }
```

This stores the text in a box and afterwards, takes the text out of the box with \unhbox (this destroys the box, if the box is needed again one should use \unhcopy instead) and passing it to \underline. (For LaTeX2e, you may want to use \usebox{\boxname} instead of \unhbox\boxname.)

This same trick done with \raisebox instead of \underline can force all lines in a table to have equal height, instead of the natural varying height that can occur when e.g. math terms or superscripts occur in the text.

Here is an example showing the use of both p{...} and >{\centering} :

```
\begin{tabular}{>{\centering}p{3.5cm}<{\centering}p{3.5cm} }
Geometry  & Algebra
\tabularnewline
\hline
 Points & Addition
\tabularnewline
 Spheres & Multiplication
\end{tabular}
```

Note the use of \tabularnewline instead of \\ to avoid a Misplaced \noalign error.

@-expressions

The column separator can be specified with the @{...} construct.

It typically takes some text as its argument, and when appended to a column, it will automatically insert that text into each cell in that column before the actual data for that cell. This command kills the inter-column space and replaces it with whatever is between the curly braces. To add space, use @{\hspace{''width''}}.

Admittedly, this is not that clear, and so will require a few examples to clarify. Sometimes, it is desirable in scientific tables to have the numbers aligned on the decimal point. This can be achieved by doing the following:

```
\begin{tabular}{r@{.}l}
  3   & 14159 \\
  16  & 2     \\
  123 & 456   \\
\end{tabular}
```

3.14159
16.2
123.456

The space-suppressing qualities of the @-expression actually make it quite useful for manipulating the horizontal spacing between columns. Given a basic table, and varying the column descriptions:

```
\begin{tabular}{ |l|l| }
  \hline
  stuff & stuff \\ \hline
  stuff & stuff \\
  \hline
\end{tabular}
```

{|l|l|}

stuff	stuff
stuff	stuff

```
{|@{}l|l@{}|}
```

stuff	stuff
stuff	stuff

```
{|@{}l@{}|l@{}|}
```

stuff	stuff
stuff	stuff

```
{|@{}l@{}|@{}l@{}|}
```

stuff	stuff
stuff	stuff

Aligning columns at decimal points using dcolumn

Instead of using @-expressions to build columns of decimals aligned to the decimal point (or equivalent symbol), it is possible to center a column on the decimal separator using the `dcolumn` package, which provides a new column specifier for floating point data. See the dcolumn package documentation (http://anorien.csc.warwick.ac.uk/mirrors/CTAN/macros/latex/required/tools/dcolumn.pdf) for more information, but a simple way to use `dcolumn` is as follows.

```
\usepackage{dcolumn}
\ldots
\newcolumntype{d}[1]{D{.}{\cdot}{#1} }
%the argument for d specifies the maximum number of decimal places
\begin{tabular}{l r c d{1} }
Left&Right&Center&\mathrm{Decimal}\\
1&2&3&4\\
11&22&33&44\\
1.1&2.2&3.3&4.4\\
\end{tabular}
```

Left	Right	Center	Decimal
1	2	3	4
11	22	33	44
1.1	2.2	3.3	4·4

A negative argument provided for the number of decimal places in the new column type allows unlimited decimal places, but may result in rather wide columns. Rounding is not applied, so the data to be tabulated should be adjusted to the number of decimal places specified. Note that a decimal aligned column is typeset in math mode, hence the use of \mathrm for the column heading in the example above. Also, text in a decimal aligned column (for example the header) will be right-aligned before the decimal separator (assuming there's no decimal separator in the text). While this may be fine for very short text, or numeric column headings, it looks cumbersome in the example above. A solution to this is to use the \multicolumn command described below, specifying a single column and its alignment. For example to center the header *Decimal* over its column in the above example, the first line of the table itself would be `Left&Right&Center&\multicolumn{1}{c}{Decimal}\\`

Bold text and dcolumn

To draw attention to particular entries in a table, it may be nice to use bold text. Ordinarily this is easy, but as dcolumn needs to *see* the decimal point it is rather harder to do. In addition, the usual bold characters are wider than their normal counterparts, meaning that although the decimals may align nicely, the figures (for more than 2--3 digits on one side of the decimal point) will be visibly misaligned. It is however possible to use normal width bold characters and define a new bold column type, as shown below.[2]

```
\usepackage{dcolumn}
%here we're setting up a version of the math fonts with normal x-width
\DeclareMathVersion{nxbold}
\SetSymbolFont{operators}{nxbold}{OT1}{cmr} {b}{n}
\SetSymbolFont{letters}  {nxbold}{OML}{cmm} {b}{it}
\SetSymbolFont{symbols}  {nxbold}{OMS}{cmsy}{b}{n}

\begin{document}
\makeatletter
\newcolumntype{d}{D{.}{.}{-1} } %decimal column as before
%wide bold decimal column
\newcolumntype{B}[3]{>{\boldmath\DC@{#1}{#2}{#3} }c<{\DC@end} }
%normal width bold decimal column
\newcolumntype{Z}[3]{>{\mathversion{nxbold}\DC@{#1}{#2}{#3} }c<{\DC@end} }
\makeatother
\begin{tabular}{l l d}
    Type &M & \multicolumn{1}{c}{N} \\
    Normal & 1 & 22222.222 \\
    Bold (standard)&10 & \multicolumn{1}{B{.}{.}{-1} }{22222.222}\\
    Bold (nxbold)&100 & \multicolumn{1}{Z{.}{.}{-1} }{22222.222}\\
\end{tabular}
\end{document}
```

Type	M	N
Normal	1	22222.222
Bold (standard)	10	**22222.222**
Bold (nxbold)	100	**22222.222**

Row specification

It might be convenient to apply the same command over every cell of a row, just as for column. Unfortunately the `tabular` environment cannot do that by default. We will need `tabu` instead, which provides the `\rowfont` option.

```
\begin{tabu}{XX}
\rowfont{\bfseries\itshape\large} Header1 & Header2 \\
\hline
Cell2 & Cell2
\end{tabu}
```

Spanning

To complete this tutorial, we take a quick look at how to generate slightly more complex tables. Unsurprisingly, the commands necessary have to be embedded within the table data itself.

Rows spanning multiple columns

The command for this looks like this: `\multicolumn{'num_cols'}{'alignment'}{'contents'}`. `num_cols` is the number of subsequent columns to merge; `alignment` is either `l`, `c`, `r`, or to have text wrapping specify a width `p{5.0cm}` . And `contents` is simply the actual data you want to be contained within that cell. A simple example:

```
\begin{tabular}{ |l|l| }
  \hline
  \multicolumn{2}{|c|}{Team sheet} \\
  \hline
  GK & Paul Robinson \\
  LB & Lucus Radebe \\
  DC & Michael Duberry \\
  DC & Dominic Matteo \\
  RB & Dider Domi \\
  MC & David Batty \\
  MC & Eirik Bakke \\
  MC & Jody Morris \\
  FW & Jamie McMaster \\
  ST & Alan Smith \\
  ST & Mark Viduka \\
  \hline
\end{tabular}
```

Team sheet	
GK	Paul Robinson
LB	Lucus Radebe
DC	Michael Duberry
DC	Dominic Matteo
RB	Dider Domi
MC	David Batty
MC	Eirik Bakke
MC	Jody Morris
FW	Jamie McMaster
ST	Alan Smith
ST	Mark Viduka

Columns spanning multiple rows

The first thing you need to do is add `\usepackage{multirow}` to the preamble[3]. This then provides the command needed for spanning rows: `\multirow{''num_rows''}{''width''}{''contents''}`. The arguments are pretty simple to deduce (* for the *width* means the content's natural width).

```
...
\usepackage{multirow}
...

\begin{tabular}{ |l|l|l| }
\hline
\multicolumn{3}{ |c| }{Team sheet} \\
\hline
Goalkeeper & GK & Paul Robinson \\ \hline
\multirow{4}{*}{Defenders} & LB & Lucus Radebe \\
 & DC & Michael Duburry \\
 & DC & Dominic Matteo \\
 & RB & Didier Domi \\ \hline
\multirow{3}{*}{Midfielders} & MC & David Batty \\
 & MC & Eirik Bakke \\
 & MC & Jody Morris \\ \hline
Forward & FW & Jamie McMaster \\ \hline
\multirow{2}{*}{Strikers} & ST & Alan Smith \\
 & ST & Mark Viduka \\
\hline
\end{tabular}
```

Team sheet		
Goalkeeper	GK	Paul Robinson
Defenders	LB	Lucus Radebe
	DC	Michael Duberry
	DC	Dominic Matteo
	RB	Didier Domi
Midfielders	MC	David Batty
	MC	Eirik Bakke
	MC	Jody Morris
Forward	FW	Jamie McMaster
Strikers	ST	Alan Smith
	ST	Mark Viduka

The main thing to note when using \multirow is that a blank entry must be inserted for each appropriate cell in each subsequent row to be spanned.

If there is no data for a cell, just don't type anything, but you still need the "&" separating it from the next column's data. The astute reader will already have deduced that for a table of n columns, there must always be $n-1$ ampersands in each row (unless \multicolumn is also used).

Spanning in both directions simultaneously

Here is a nontrivial example of how to use spanning in both directions simultaneously and have the borders of the cells drawn correctly:

```
\usepackage{multirow}

\begin{tabular}{cc|c|c|c|l}
\cline{3-6}
& & \multicolumn{4}{ c| }{Primes} \\ \cline{3-6}
& & 2 & 3 & 5 & 7 \\ \cline{1-6}
\multicolumn{1}{ |c  }{\multirow{2}{*}{Powers} } &
\multicolumn{1}{ |c| }{504} & 3 & 2 & 0 & 1 &     \\ \cline{2-6}
\multicolumn{1}{ |c  }{}                          &
\multicolumn{1}{ |c| }{540} & 2 & 3 & 1 & 0 &     \\ \cline{1-6}
\multicolumn{1}{ |c  }{\multirow{2}{*}{Powers} } &
\multicolumn{1}{ |c| }{gcd} & 2 & 2 & 0 & 0 & min \\ \cline{2-6}
\multicolumn{1}{ |c  }{}                          &
\multicolumn{1}{ |c| }{lcm} & 3 & 3 & 1 & 1 & max \\ \cline{1-6}
\end{tabular}
```

		Primes				
		2	3	5	7	
Powers	504	3	2	0	1	
	540	2	3	1	0	
Powers	gcd	2	2	0	0	min
	lcm	3	3	1	1	max

The command \multicolumn{1}{ is just used to draw vertical borders both on the left and on the right of the cell. Even when combined with \multirow{2}{*}{...}, it still draws vertical borders that only span the first row. To compensate for that, we add \multicolumn{1}{ in the following rows spanned by the multirow. Note that we cannot just use \hline to draw horizontal lines, since we do not want the line to be drawn over the text that spans several rows. Instead we use the command \cline{2-6} and opt out the first column that contains the text "Powers".

Here is another example exploiting the same ideas to make the familiar and popular "2x2" or double dichotomy:

```
\begin{tabular}{ r|c|c| }
\multicolumn{1}{r}{}
 &  \multicolumn{1}{c}{noninteractive}
 & \multicolumn{1}{c}{interactive} \\
\cline{2-3}
massively multiple & Library & University \\
\cline{2-3}
one-to-one & Book & Tutor \\
\cline{2-3}
\end{tabular}
```

	noninteractive	interactive
massively multiple	Library	University
one-to-one	Book	Tutor

Controlling table size

Resize tables

The graphicx packages features the command \resizebox{width}{height}{object} which can be used with tabular to specify the height and width of a table. The following example shows how to resize a table to 8cm width while maintaining the original width/height ratio.

```
\usepackage{graphicx}
% ...

\resizebox{8cm}{!} {
  \begin{tabular}...
  \end{tabular}
}
```

Resizing table including the caption

```
\begin{table}[h]
\resizebox{1.4\textwidth}{!}{\begin{minipage}{\textwidth}
\begin{tabular}{r|c|c|}
```

```
& \multicolumn{1}{c}{noninteractive}
& \multicolumn{1}{c}{interactive} \\
\cline{2-3}
massively multiple & Library & University \\
\cline{2-3}
one-to-one & Book & Tutor \\
\cline{2-3}
\end{tabular}
\caption[Table caption text]{Table taken from \cite[p.10]{refid} }
\label{table:name}
\end{minipage} }
\end{table}
```

Alternatively you can use \scalebox{ratio}{object} in the same way but with ratios rather than fixed sizes:

```
\usepackage{graphicx}
% ...

\scalebox{0.7}{
  \begin{tabular}...
  \end{tabular}
}
```

Changing font size

A table can be globally switched to a different font size by simply adding the desired size command (here: \footnotesize) in the table scope, which may be after the \begin{table} statement if you use floats, otherwise you need to add a group delimiter.

```
{\footnotesize
  \begin{tabular}{| r | r || c | c | c |}
      % ...
  \end{tabular}
}
```

```
\begin{table}[h]\footnotesize
  \caption{Performance at peak F-measure}
  \begin{tabular}{| r | r || c | c | c |}
      % ...
  \end{tabular}
\end{table}
```

Alternatively, you can change the default font for all the tables in your document by placing the following code in the preamble:

```
\let\oldtabular\tabular
\renewcommand{\tabular}{\footnotesize\oldtabular}
```

See Fonts for named font sizes. The table caption font size is not affected. To control the caption font size, see Caption Styles.

Colors

Alternate row colors in tables

The `xcolor` package provides the necessary commands to produce tables with alternate row colors, when loaded with the `table` option. The command \rowcolors{<''starting row''>}{<''odd color''>}{<''even color''>} has to be specified right before the `tabular` environment starts.

```
\documentclass{article}

\usepackage[table]{xcolor}

\begin{document}

\begin{center}
```

```
\rowcolors{1}{green}{pink}

\begin{tabular}{lll}
odd     & odd   & odd \\
even    & even  & even\\
odd     & odd   & odd \\
even    & even  & even\\
\end{tabular}
\end{center}

\end{document}
```

odd	odd	odd
even	even	even
odd	odd	odd
even	even	even

The command \hiderowcolors is available to deactivate highlighting from a specified row until the end of the table. Highlighting can be reactivated within the table via the \showrowcolors command. If while using these commands you experience "misplaced \noalign errors" then use the commands at the very beginning or end of a row in your tabular.

```
\hiderowcolors odd & odd & odd \\
```

or

```
odd & odd & odd \\ \showrowcolors
```

Colors of individual Cells

As above this uses the xcolor package.

```
% Include this somewhere in your document
\usepackage[table]{xcolor}

% Enter this in the cell you wish to color a light grey.
% NB: the word 'gray' here denotes the grayscale color scheme, not the color grey. '0.9' denotes how dark the grey is.
\cellcolor[gray]{0.9}
% The following will color the cell red.
\cellcolor{red}
```

Width and stretching

We keep providing documentation for tabular* and tabularx although they are completely eclipsed by the much more powerful and flexible tabu environment. Actually tabu is greatly inspired by those environments, so it may be worth it to have an idea how they work, particularly for tabularx.

The *tabular** environment

This is basically a slight extension on the original tabular version, although it requires an extra argument (before the column descriptions) to specify the preferred width of the table.

```
\begin{tabular*}{0.75\textwidth}{ | c | c | c | r | }
  \hline
  label 1 & label 2 & label 3 & label 4 \\
  \hline
  item 1  & item 2  & item 3  & item 4  \\
  \hline
\end{tabular*}
```

label 1	label 2	label 3	label 4	
item 1	item 2	item 3	item 4	

However, that may not look quite as intended. The columns are still at their natural width (just wide enough to fit their contents) while the rows are as wide as the table width specified. If you do not like this default, you must also explicitly insert extra column space. LaTeX has *rubber lengths*, which, unlike others, are not fixed. LaTeX can dynamically decide how long the lengths should be. So, an example of this is the following.

```
\begin{tabular*}{0.75\textwidth}{@{\extracolsep{\fill} } | c | c | c | r | }
  \hline
  label 1 & label 2 & label 3 & label 4 \\
  \hline
```

label 1	label 2	label 3	label 4
item 1	item 2	item 3	item 4

```
  item 1  & item 2  & item 3  & item 4  \\
  \hline
\end{tabular*}
```

You will notice the @{...} construct added at the beginning of the column description. Within it is the \extracolsep command, which requires a width. A fixed width could have been used. However, by using a rubber length, such as \fill, the columns are automatically spaced evenly.

The *tabularx* package

This package provides a table environment called `tabularx` which is similar to the `tabular*` environment, except that it has a new column specifier `X` (in uppercase). The column(s) specified with this specifier will be stretched to make the table as wide as specified, greatly simplifying the creation of tables.

```
\usepackage{tabularx}
% ...

\begin{tabularx}{\textwidth}{ |X|X|X|X| }
  \hline
  label 1 & label 2 & label 3 & label 4 \\
  \hline
  item 1  & item 2  & item 3  & item 4  \\
  \hline
\end{tabularx}
```

label 1	label 2	label 3	label 4
item 1	item 2	item 3	item 4

The content provided for the boxes is treated as for a `p` column, except that the width is calculated automatically. If you use the package `array`, you may also apply any >{\cmd} or <{\cmd} command to achieve specific behavior (like \centering, or \raggedright\arraybackslash) as described previously.

Another option is the use of \newcolumntype in order to get selected columns formatted in a different way. It defines a new column specifier, e.g. `R` (in uppercase). In this example, the second and fourth column is adjusted in a different way (\raggedleft):

```
\usepackage{tabularx}
% ...

\newcolumntype{R}{>{\raggedleft\arraybackslash}X}%
\begin{tabularx}{\textwidth}{ |l|R|l|R| }
  \hline
  label 1 & label 2 & label 3 & label 4 \\
  \hline
  item 1  & item 2  & item 3  & item 4  \\
  \hline
\end{tabularx}
```

label 1	label 2	label 3	label 4
item 1	item 2	item 3	item 4

Tabularx with rows spanning multiple columns using \multicolumn. The two central columns are posing as one by using the `X@{}` option. Note that the \multicolumn width (which in this example is 2) should equal the (in this example 1+1) width of the spanned columns:

```
\usepackage{tabularx}
% ...

\begin{tabularx}{1\textwidth}{ |>{\setlength\hsize{1\hsize}\centering}X|>
{\setlength\hsize{1\hsize}\raggedleft}X@{} >
{\setlength\hsize{1\hsize}\raggedright}X|>{\setlength\hsize{1\hsize}\centering}X| }
  \hline
Label 1 & \multicolumn{2}{>{\centering\setlength\hsize{2\hsize} }X|}{Label 2} &
Label 3\tabularnewline
\hline
  123  & 123  & 456  & 123  \tabularnewline
  \hline
  123  & 123  & 456  & 123  \tabularnewline
  \hline
\end{tabularx}
```

Label 1	Label 2	
123	123456	
123	123456	

In a way analogous to how new commands with arguments can be created with \newcommand, new column types with arguments can be created with \newcolumntype as follows:

```
\usepackage{tabularx}
\usepackage[table]{xcolor} %Used to color the last column
% ...

\newcolumntype{L}[1]{>{\hsize=#1\hsize\raggedright\arraybackslash}X}%
\newcolumntype{R}[1]{>{\hsize=#1\hsize\raggedleft\arraybackslash}X}%
\newcolumntype{C}[2]{>{\hsize=#1\hsize\columncolor{#2}\centering\arraybackslash}X}%

\begin{tabularx}{\textwidth}{ | L{1} | R{0.5} | R{0.5} | C{2}{gray} | }
  \hline
  label 1 & label 2 & label 3 & label 4 \\
  \hline
  item 1  & item 2  & item 3  & item 4  \\
  \hline
\end{tabularx}
```

where since there are 4 columns, the sum of the \hsize's (1 + 0.5 + 0.5 + 2) must be equal to 4. The default value used by tabularx for \hsize is 1.

The *tabulary* package

tabulary (http://www.ctan.org/pkg/tabulary) is a modified tabular* allowing width of columns set for equal heights. tabulary allows easy and convenient writing of well balanced tables.

The problem with tabularx is that it leaves much blank if your cells are almost empty. Besides, it is not easy to have different column sizes.

tabulary tries to balance the column widths so that each column has at least its natural width, without exceeding the maximum length.

```
\usepackage{tabulary}
...

\begin{center}
  \begin{tabulary}{0.7\textwidth}{LCL}
    Short sentences      & \#  & Long sentences                                                  \\
    \hline
    This is short.       & 173 & This is much looooooooonger, because there are many more words.  \\
    This is not shorter. & 317 & This is still looooooooonger, because there are many more words. \\
  \end{tabulary}
\end{center}
```

The first parameter is the maximum width. tabulary will try not to exceed it, but it will not stretch to it if there is not enough content, contrary to tabularx.

The second parameter is the column disposition. Possible values are those from the tabular environment, plus

L	left-justified balanced column
C	centered balanced column
R	right-justified balanced column
J	left-right-justified balanced column

These are all capitals.

The *tabu* environment

It works pretty much like tabularx.

```
\begin{tabu} to \linewidth {llX[2]lllXl}
% ...
\end{tabu}
```

'to \linewidth' specifies the target width. The X parameter can have an optional span factor.

Table across several pages

Long tables are natively supported by LaTeX thanks to the longtable environment. Unfortunately this environment does not support stretching (X columns).

The tabu packages provides the longtabu environment. It has most of the features of tabu, with the additional capability to span multiple pages.

LaTeX can do well with long tables: you can specify a header that will repeat on every page, a header for the first page only, and the same for the footer.

```
\begin{longtabu} to \linewidth {lX[2]lXl}

\rowfont\bfseries H1 & H2 & H3 & H4 & H5 \\ \hline
\endhead

\\ \hline
\multicolumn{5}{r}{There is more to come} \\
\endfoot

\\ \hline
\endlastfoot

% Content ...
```

It uses syntax similar to `longtable`, so you should have a look at its documentation if you want to know more.

Alternatively you can try one of the following packages `supertabular` (http://www.ctan.org/pkg/supertabular) or `xtab` (http://www.ctan.org/pkg/xtab), an extended and somewhat improved version of `supertabular`.

Partial Vertical Lines

Adding a partial vertical line to an individual cell:

```
\begin{tabular}{ l c r }
  \hline
  1 & 2 & 3 \\ \hline
  4 & 5 & \multicolumn{1}{r|}{6}  \\ \hline
  7 & 8 & 9 \\ \hline
\end{tabular}
```

1	2	3
4	5	6
7	8	9

Removing part of a vertical line in a particular cell:

```
\begin{tabular}{ | l | c | r | }
  \hline
  1 & 2 & 3 \\ \hline
  4 & 5 & \multicolumn{1}{r}{6} \\ \hline
  7 & 8 & 9 \\ \hline
\end{tabular}
```

1	2	3
4	5	6
7	8	9

Vertically centered images

Inserting images into a table row will align it at the top of the cell. By using the `array` package this problem can be solved. Defining a new columntype will keep the image vertically centered.

```
\newcolumntype{V}{>{\centering\arraybackslash} m{.4\linewidth} }
```

Or use a parbox to center the image.

```
\parbox[c]{1em}{\includegraphics{image.png} }
```

A raisebox works as well, also allowing to manually fine-tune the alignment with its first parameter.

```
\raisebox{-.5\height}{\includegraphics{image.png} }
```

Footnotes in tables

The `tabular` environment does not handle footnotes properly. The `longtabular` fixes that.

Instead of using `longtabular` we recommend `tabu` which handles footnotes properly, both in normal and long tables.

Professional tables

Many professionally typeset books and journals feature simple tables, which have appropriate spacing above and below lines, and almost *never* use vertical rules. Many examples of LaTeX tables (including this Wikibook) showcase the use of vertical rules (using "|"), and double-rules (using \hline\hline or "||"), which are regarded as unnecessary and distracting in a professionally published form. The booktabs (http://www.ctan.org/tex-archive/macros/latex/contrib/booktabs/) package is useful for easily providing this professionalism in LaTeX tables, and the documentation (http://mirrors.ctan.org/macros/latex/contrib/booktabs/booktabs.pdf) also provides guidelines on what constitutes a "good" table.

In brief, the package uses \toprule for the uppermost rule (or line), \midrule for the rules appearing in the middle of the table (such as under the header), and \bottomrule for the lowermost rule. This ensures that the rule weight and spacing are acceptable. In addition, \cmidrule can be used for mid-rules that span specified columns. The following example contrasts the use of booktabs and two equivalent normal LaTeX implementations (the second example requires \usepackage{array} or \usepackage{dcolumn}, and the third example requires \usepackage{booktabs} in the preamble).

Normal LaTeX

```
\begin{tabular}{llr}
\hline
\multicolumn{2}{c}{Item} \\
\cline{1-2}
Animal    & Description & Price (\$) \\
\hline
Gnat      & per gram    & 13.65      \\
          & each        & 0.01       \\
Gnu       & stuffed     & 92.50      \\
Emu       & stuffed     & 33.33      \\
Armadillo & frozen      & 8.99       \\
\hline
\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

Using array

```
\usepackage{array}
%or \usepackage{dcolumn}
...
\begin{tabular}{llr}
\firsthline
\multicolumn{2}{c}{Item} \\
\cline{1-2}
Animal    & Description & Price (\$) \\
\hline
Gnat      & per gram    & 13.65      \\
          & each        & 0.01       \\
Gnu       & stuffed     & 92.50      \\
Emu       & stuffed     & 33.33      \\
Armadillo & frozen      & 8.99       \\
\lasthline
\end{tabular}
```

Using booktabs

```
\usepackage{booktabs}

...
\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

Usually the need arises for footnotes under a table (and not at the bottom of the page), with a caption properly spaced above the table. These are addressed by the ctable (http://www.ctan.org/tex-archive/macros/latex/contrib/ctable/) package. It provides the option of a short caption given to be inserted in the list of tables, instead of the actual caption (which may be quite long and inappropriate for the list of tables). The `ctable` uses the `booktabs` package.

Sideways tables

Tables can also be put on their side within a document using the `rotating` or the `rotfloat` package. See the Rotations chapter.

Table with legend

To add a legend to a table the caption (http://www.ctan.org/tex-archive/macros/latex/contrib/caption/) package can be used. With the caption package a `\caption*{...}` statement can be added besides the normal `\caption{...}`. Example:

```
\begin{table}
  \begin{tabular}{| r | r || c | c | c |}

      ...

  \end{tabular}
  \caption{A normal caption}
  \caption*{
    A legend, even a table can be used
    \begin{tabular}{l l}
      item 1 & explanation 1 \\
    \end{tabular}
  }
\end{table}
```

The normal caption is needed for labels and references.

The *eqparbox* package

On rare occasions, it might be necessary to stretch every row in a table to the natural width of its longest line, for instance when one has the same text in two languages and wishes to present these next to each other with lines synching up. A tabular environment helps control where lines should break, but cannot justify the text, which leads to ragged right edges. The `eqparbox` package provides the command `\eqmakebox` which is like `\makebox` but instead of a `width` argument, it takes a tag. During compilation it bookkeeps which `\eqmakebox` with a certain tag contains the widest text and can stretch all `\eqmakebox`es with the same tag to that width. Combined with the `array` package, one can define a column specifier that justifies the text in all lines:

```
\newsavebox{\tstretchbox}
\newcolumntype{S}[1]{%
  >{\begin{lrbox}{\tstretchbox} }%
  l%
  <{\end{lrbox}%
  \eqmakebox[#1][s]{\unhcopy\tstretchbox} }%
}
```

See the documentation of the `eqparbox` package for more details.

Floating with *table*

In WYSIWYG document processors, it is common to put tables in the middle of the text. This is what we have been doing until now. Professional documents, however, often make it a point to print tables on a dedicated page so that they do not disrupt the flow. From the point of view of the source code, one has no idea on which page the current text is going to lie, so it is hardly possible to guess which page may be appropriate for our table. LaTeX can automate this task by abstracting objects such as tables, pictures, etc., and deciding for us where they might fit best. This abstraction is called a *float*. Generally, an object that is floated will appear in the vicinity of its introduction in the source file, but one can choose to control its position also.

To tell LaTeX we want to use our table as a float, we need to place a `tabular` environment in a `table` environment, which is able to float and add a label and caption.

The `table` environment initiates a type of float just as the environment `figure`. In fact, the two bear a lot of similarities (positioning, captions, etc.). More information about floating environments, captions etc. can be found in Floats, Figures and Captions.

The environment names may now seem quite confusing. Let's sum it up:

- `tabular` is for the content itself (columns, lines, etc.).
- `table` is for the location of the table on the document, plus caption and label support.

```
\begin{table}[position specifier]
  \centering
  \begin{tabular}{|l|}
    ... your table ...
  \end{tabular}
  \caption{This table shows some data}
  \label{tab:myfirsttable}
\end{table}
```

In the table, we used a label, so now we can refer to it just like any other reference:

```
\ref{tab:myfirsttable}
```

The `table` environment is also useful when you want to have a list of tables at the beginning or end of your document with the command

```
\listoftables
```

The captions show now up in the list of tables, if displayed.

You can set the optional parameter `position specifier` to define the position of the table, where it should be placed. The following characters are all possible placements. Using sequences of it define your "wishlist" to LaTeX.

`h`	where the table is declared (**h**ere)
`t`	at the **t**op of the page
`b`	at the **b**ottom of the page
`p`	on a dedicated **p**age of floats
`!`	override the default float restrictions. E.g., the maximum size allowed of a `b` float is normally quite small; if you want a large one, you need this `!` parameter as well.

Default is *tbp*, which means that it is by default placed on the top of the page. If that's not possible, it's placed at the bottom if possible, or finally with other floating environments on an extra page.

You can force LaTeX to use one given position. E.g. *[!h]* forces LaTeX to place it exactly where you place it (Except when it's really impossible, e.g you place a table *here* and this place would be the last line on a page). Again, understand it correctly: it urges LaTeX to put the table at a specific place, but it will not be placed there if LaTeX thinks it will not look great. If you really want to place your table manually, do not use the `table` environment.

Centering the table horizontally works like everything else, using the `\centering` command just after opening the `table` environment, or by enclosing it with a `center` environment.

Using spreadsheets

For complex or dynamic tables, you may want to use a spreadsheet. You might save lots of time by building tables using specialized software and exporting them in LaTeX format. The following plugins and libraries are available for some popular software:

- calc2latex (http://calc2latex.sourceforge.net/): for OpenOffice.org Calc spreadsheets,
- excel2latex (http://www.ctan.org/tex-archive/support/excel2latex/): for Microsoft Office Excel,
- matrix2latex (http://www.mathworks.com/matlabcentral/fileexchange/4894-matrix2latex): for MATLAB,
- matrix2latex (https://code.google.com/p/matrix2latex/): for Python and MATLAB,
- latex-tools (http://rubygems.org/gems/latex-tools): a Ruby library,
- xtable (http://cran.r-project.org/web/packages/xtable/index.html): a library for R,
- org-mode (http://orgmode.org/): for Emacs users, org-mode tables can be used inline in LaTeX documents, see [1] (https://www.gnu.org/software/emacs/manual/html_node/org/A-LaTeX-example.html) for a tutorial.
- Emacs align commands (http://emacswiki.org/emacs/AlignCommands): the align commands can clean up a messy LaTeX table.
- Online Table generator for LATeX (http://truben.no/latex/table/): An online tool for creating simple tables within the browser. LaTeX format is directly generated as you type.

However, copying the generated source code to your document is not convenient at all. For maximum flexibility, generate the source code to a separate file which you can input from your main document file with the \input command. If your speadsheet supports command-line, you can generate your complete document (table included) in one command, using a Makefile for example.

See Modular Documents for more details.

Need more complicated features?

Have a look at one of the following packages:

- hhline (http://www.ctan.org/pkg/hhline): do whatever you want with horizontal lines
- array (http://www.ctan.org/pkg/array): gives you more freedom on how to define columns
- colortbl (http://www.ctan.org/pkg/colortbl): make your table more colorful
- threeparttable (http://ctan.org/tex-archive/macros/latex/contrib/threeparttable) makes it possible to put footnotes both within the table and its caption
- arydshln (http://www.ctan.org/pkg/arydshln): creates dashed horizontal and vertical lines
- ctable (http://www.ctan.org/pkg/ctable): allows for footnotes under table and properly spaced caption above (incorporates booktabs package)
- slashbox (http://www.ctan.org/pkg/slashbox): create 2D tables with the first cell containing a description for both axes. Not available in Tex Live 2011 or later.
- diagbox (http://mirror.jmu.edu/pub/CTAN/macros/latex/contrib/diagbox/): compatible to slashbox, come with Tex Live 2011 or later
- dcolumn (http://www.ctan.org/pkg/dcolumn): decimal point alignment of numeric cells
- rccol (http://www.ctan.org/pkg/rccol): advanced decimal point alignment of numeric cells with rounding
- numprint (http://www.ctan.org/pkg/numprint): print numbers, in the current mode (text or math) in order to use the correct font, with separators, exponent and/or rounded to a given number of digits. tabular(*), array, tabularx, and longtable environments is supported using all features of numprint
- spreadtab (http://www.ctan.org/pkg/spreadtab): spread sheets allowing the use of formulae
- siunitx (http://ctan.org/tex-archive/macros/latex/contrib/siunitx): alignment of tabular entries
- pgfplotstable (http://www.ctan.org/tex-archive/graphics/pgf/contrib/pgfplots): Loads, rounds, formats and postprocesses numerical tables.

References

1. ↑ http://tex.stackexchange.com/questions/121841/is-the-tabu-package-obsolete
2. ↑ D Carlisle. "Decimals in table don't align with dcolumn when bolded". Stackexchange. http://tex.stackexchange.com/questions/118458/decimals-in-table-dont-align-with-dcolumn-when-bolded.
3. ↑ Package multirow on CTAN (http://www.ctan.org/tex-archive/macros/latex/contrib/multirow/)