Typesetting tables with LaTeX

Klaus Höppner
Haardtring 230 a
64295 Darmstadt
Germany
klaus.hoeppner (at) gmx dot de

Abstract

From a LaTeXoligist's point of view, LaTeX is a perfect tool to typeset nearly everything in a beautiful manner. Without any doubt, LaTeX can typeset tables, but it is easy to produce bad tables with ugly lines and text touching the lines. This talk is intended to introduce how to typeset tables with LaTeX on a beginners' level, mentioning some typographic aspects, showing some packages that help the author in formatting tables and concluding with how to typeset tables with page breaks.

1 Basic tables

LaTeX already has built-in support to typeset tables. For beginners it may be a bit confusing, since LaTeX provides two environments: `tabular` and `table`. To typeset material in rows and columns, `tabular` is needed, while the `table` environment is a container for floating material similar to `figure`, into which a `tabular` environment may be included.

So, let's have a look how to typeset a simple table:

```
\begin{tabular}{lcr}
a   & b   & c\\
aa  & ab  & ac\\
aaa & aab & aac
\end{tabular}
```

will result in

a	b	c
aa	ab	ac
aaa	aab	aac

The rows of the table are divided by LaTeX's usual `\\` command (in some cases, it may be needed to use `\tabularnewline` instead, as we will see later in this article). Columns are separated by `&`, the ampersand character.

The required argument of `\begin{tabular}` defines the basic layout of the table, especially the alignment of the columns:

l left aligned column

c centered column

r right aligned column

p{⟨*width*⟩**}** paragraph-like column of a predefined width (with the baseline of the paragraph's first line aligned relative to the other cells in the table row)

The normal space between columns, which is also added before the first and after the last column, may be overridden by `@{`⟨*sep*⟩`}`, where ⟨*sep*⟩ is any LaTeX code, inserted as the separator. For illustration, let's typeset some flight data:

flight no.	route
LH 402	Frankfurt–Newark
KL 3171	Amsterdam–Cork
US 1152	San Diego–Philadelphia

Here, the `@` command is used twice: The space that normally would have been inserted left of the first column is replaced by nothing, thus the table is left aligned with the surrounding text (compare it with the first tabular example in this article, you will see the difference). Additionally, the inter-column space between the points of departure and destination is replaced by a dash. So the code used to produce this table looks as follows (silently introducing `\multicolumn` to combine cells):

```
\begin{tabular}{@{}lr@{--}l}
flight no. & \multicolumn{2}{c}{route}\\
LH\,402 & Frankfurt & Newark\\
KL\,3171 & Amsterdam & Cork\\
US\,1152 & San Diego & Philadelphia
\end{tabular}
```

2 Extra packages for typesetting tables

Beyond LaTeX's built-in ability to typeset tables, several extra packages exist. Some of them add new effects in typography and layout, others simplify the task of writing the document's source code. The packages that I will introduce in this article (and more that I won't) are covered in detail in the *LaTeX Companion* [8].

Here are some important packages for authors who want to typeset tables:

array adds paragraph-like columns `m{`⟨*width*⟩`}` and `b{`⟨*width*⟩`}` similar to the `p`-column, but vertically aligned to the center or bottom. Additionally, the package allows defining command sequences to be executed before or after the contents of a column.

tabularx typesets a table with fixed widths, introducing the column type `X` that works like a `p`-column with automatically calculated width.

booktabs provides fancy commands for horizontal lines with appropriate spacing above and below.

ctable we won't discuss this one, but have a look on CTAN, it's a modern table package with many nice features.

2.1 Using array

From my point of view, the most important feature of `array` [7] is the new possibility of defining commands that are added automatically before or after a column's cell. It saves a lot of typing, making the source code more concise and more flexible. `array` is one of the required LaTeX packages, so it must be part of any LaTeX installation.

For a simple example, have a look at the following table:

Command	Symbol
`\alpha`	α
`\beta`	β
`\gamma`	γ

The left column displays LaTeX commands, beginning with a backslash and using a typewriter font, while the right columns displays the corresponding math symbol. Using the `array` package, the source code is pretty straightforward:

```
\begin{tabular}%
    {>{\ttfamily\char`\\}c>{$}c<{$}}
\multicolumn{1}{c}{Command} &
    \multicolumn{1}{c}{Symbol}\\
\hline
alpha & \alpha\\
beta & \beta\\
gamma & \gamma
\end{tabular}
```

As shown in this code, we can now define a command sequence inside the `>{...}` preceding the column type definition. These commands are executed before each cell of that column. Similarly, using `<{...}` after the column type defines which commands to be executed after typesetting the column's cells.

In the example above, the first row is different from the others, since it contains the column titles that obviously should not be typeset in typewriter font or math mode. This is handled by 'abusing' the `\multicolumn` command, which prevents the `>` and `<` command hooks from being applied for these cells.

Another use of these command hooks is typesetting paragraphs in narrow columns. LaTeX typesets these paragraphs left and right justified by default, but in narrow columns it is often more appropriate to typeset them using `\raggedright`. So we might think of trying the following code:

```
\begin{tabular}{l>{\raggedright}p{3in}}
```

Unfortunately this fails when ending the table rows with the `\\` command, with rather weird error messages about misplaced aligns. The problem is that `\raggedright` redefines `\\`, so it can't be recognized as the end of table rows. There are three solutions for this problem:

1. Use `\tabularnewline` instead of `\\`. In fact, it does no harm to always use this, even when you don't have problems with `\raggedright`.
2. Restore the original definition of `\\` by using the command `\arraybackslash`, as follows:
   ```
   \begin{tabular}%
       {l>{\raggedright\arraybackslash}p{3in}}
   ```
3. Use the `ragged2e` [9] package. It redefines the command `\raggedright` to prevent it from redefining `\\`, so the problem disappears without any further change to the original code. Additionally, `ragged2e` provides the new command `\RaggedRight` that typesets the paragraph left aligned, but doesn't disable hyphenation.

2.2 Using tabularx

Besides the normal `tabular` environment, a rarely used environment `tabular*` exists. In addition to the column definition, it takes a table width as argument. The resulting table is typeset to this width, but often — surprisingly — it expands the space between columns.

A more convenient implementation is done by the `tabularx` [5] package (another required LaTeX package present in every LaTeX installation). This introduces a column type `X` for paragraph-like columns whose width is automatically calculated in order to achieve a table of a desired total width. For example, let's look at the following:

```
\begin{tabularx}{\linewidth}{lX}
Label & Text\\
\hline
One & This is some text without meaning,
  just using up some space. It is not
  intended for reading.\\
...
\end{tabularx}
```

This produces a table across the full line width, where the right column just uses the space remaining after typesetting the left column:

Label	Text
One	This is some text without meaning, just using up some space. It is not intended for reading.
Two	This is another text without meaning, just using up some space. It's not intended for reading either.
Three	This is yet another text without meaning. Guess what? It's not intended for reading. It is just there.
Four	How often did I mention that you should not read this text?

It is possible to use more than one `X`-column. By default, all of them are typeset to the same width, but it is possible to manually adjust how the available space is divided. Here's our next example:

Label	Text	More text
One	This is some text without meaning.	This is another text without meaning, just using up some space. It is not meant for reading either.

This table was produced with the following code:

```
\begin{tabularx}{\linewidth}%
 {l>{\setlength\hsize{0.6\hsize}\raggedright}X%
   >{\setlength\hsize{1.4\hsize}\raggedright}X}
Label & Text & More text\tabularnewline
\hline
...
\end{tabularx}
```

When balancing the column widths manually, it is important that the `\hsize` fractions add up to the number of `X`-columns, as in the example above, where $0.6 + 1.4 = 2$. To achieve *automatic* balancing of columns, take a look at the `tabulary` package.

Be aware that the way `tabularx` parses the contents of a table limits the possibility of defining new environments based on the `tabularx`. If you consider doing this, first look at the documentation.

3 Using lines in tables

LaTeX provides the possibility of using lines in tables: vertical lines are added by placing a `|` at the appropriate position in the definition of the column layout, and horizontal lines are added by using the command `\hline`.

While using lines in tables can help the reader in understanding the contents of a table, it is quite easy to produce really ugly tables like the following:

Label	Text	More text
One	This is some text without meaning.	This is another text without meaning, just using up some space.

Though nobody would typeset this particular table in real life, it illustrates a general and common problem — the column titles and the word "another" in the rightmost column touch the horizontal lines above them.

As a first step to improve the spacing between the table rows and the horizontal lines in such cases, set `\extrarowheight` to a non-zero length, e. g. to 4 pt. If this isn't enough, additional adjustment may be done by adding invisible rules. Here is revised source code for the above example illustrating both these points:

```
\setlength{\extrarowheight}{4pt}
\begin{tabularx}{\linewidth}%
    {|l|>{\setlength\hsize{0.67\hsize}}X%
       |>{\setlength\hsize{1.33\hsize}}X|}
\hline
\Large Label & \Large Text
    & \Large More text\tabularnewline
\hline
\hline
One & This is some text without meaning.
    & \rule{0pt}{18pt}%
      This is {\huge another} text without meaning,
      just using up some space.\\
\hline
\end{tabularx}
```

we get a somewhat better result:

Label	Text	More text
One	This is some text without meaning.	This is another text without meaning, just using up some space.

Please notice that the `\rule` used as an additional spacer was typeset with a horizontal width of 0.4 pt instead of 0 pt (as shown in the code) in order to make its effect and location visible.

Even after this, the layout of the table still looks quite poor, e. g. the broken vertical lines between the double horizonal line. This might be solved with the package `hhline` [2], but for typesetting tables with pretty lines, have a look at the `booktabs` [6] package. It starts by giving users a basic piece of advice, namely to avoid vertical lines, and introduces commands to typeset horizontal lines with appropriate thickness and spacing. Using `booktabs`, the source code for our weird example now looks as follows:

```
\begin{tabularx}{\linewidth}%
    {l>{\setlength\hsize{0.67\hsize}}X%
      >{\setlength\hsize{1.33\hsize}}X}
\toprule
\Large Label & \Large Text
    & \Large More text\tabularnewline
\midrule
One & This is some text without meaning.
    & This is {\huge another} text without meaning,
      just using up some space.\\
\bottomrule
\end{tabularx}
```

Using this, the result becomes:

Label	Text	More text
One	This is some text without meaning.	This is another text without meaning, just using up some space.

At last, we've improved the layout of the table quite a bit. The content with arbitrary changes of font size still looks weird, but that's something for which the author and not LaTeX must be blamed.

For a more realistic example of using rules, here I present an example from the booktabs manual:

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

4 Typesetting tables across multiple pages

The usual `tabular(x)` environment is restricted to single-page tables, i. e. no page breaks are allowed within tables.

However, two extension packages provide support for typesetting tables across multiple pages, namely `longtable` [3] and `supertabular` [1]. The main difference between them is that `longtable` keeps the column widths the same throughout the entire table, while `supertabular` recalculates the column widths on each page. According to the documentation, `longtable` doesn't work in two- or multicolumn mode, and I didn't try `supertabular` on this case. The syntax of the two packages is different, so one has to decide which one to use.

Let's have a look at the general structure of a `longtable`:

```
\begin{longtable}{ll}
Label (cont.) & Text (cont.)\\
\endhead
\multicolumn{2}{l}{This is the first head}\\
Label & Text\\
\endfirsthead
\multicolumn{2}{l}{to be cont'd on next page}
\endfoot
\multicolumn{2}{l}{this is the end (finally!)}
\endlastfoot
One & Some content\\
Two & Another content\\
Three & Yet another content\\
[...]
\end{longtable}
```

As shown in this source code, the `longtable` environment may contain definitions for headers and footers before the normal content of the table:

`\endfirsthead` defines what to typeset at the very first head of the table,

`\endhead` defines a table head that is used on continuation pages,

`\endfoot` defines what to typeset on the foot of the table if a continuation page follows, and

`\endlastfoot` defines the foot at the very end of the table.

I personally prefer `longtable` simply because there exists yet another package `ltxtable` [4], which combines `longtable` and `tabularx`. It provides the command `\LTXtable{`⟨*width*⟩`}{`⟨*file*⟩`}`, which reads a given file containing a `longtable` environment using `X`-columns and typesets it to the requested width.

References

[1] Johannes Braams and Theo Jurriens. The `supertabular` package. `CTAN:macros/latex/contrib/supertabular/`.

[2] David Carlisle. The `hhline` package.

[3] David Carlisle. The `longtable` package.

[4] David Carlisle. The `ltxtable` package. `CTAN:macros/latex/contrib/carlisle/`.

[5] David Carlisle. The `tabularx` package.

[6] Simon Fear and Danie Els. The `booktabs` package. `CTAN:macros/latex/contrib/booktabs/`.

[7] Frank Mittelbach and David Carlisle. The `array` package.

[8] Frank Mittelbach, Michel Goossens, et al. *The LaTeX Companion*. Addison-Wesley, 2nd edition, 2004.

[9] Martin Schröder. The `ragged2e` package. `CTAN:macros/latex/contrib/ms/`.

Packages [2, 3, 5] are part of the required LaTeX tools, available from `CTAN:macros/latex/required/tools`.