The PracTEX Journal, 2006, No. 3
Article revision 2006/08/15

Productivity with macros and packages

Will Robertson

Email `wspr81@gmail.com`
Address School of Mechanical Engineering, University of Adelaide, Australia

Abstract LATEX's advantages in productivity, for me, are due to its ability to be customised. The first half of this article discusses small macros written to ease document production, with some examples of how I use macros to save time and effort. Then, I briefly cover a selection of packages that provide a whole heap of functionality that other people have kindly implemented.

1 Introduction

One of LATEX's advantages lies in its separation of content and formatting. This concept, called logical markup, enables authors to write without the distraction of worrying about typesetting. Of course, authors aren't *forced* to work this way, but it's highly recommended.

Implicit in the idea of logical markup is the ability to define your own logical elements as the text requires. These are known in LATEX as macros or commands, and can be as simple as `\newcommand\strong[1]{\textbf{#1}}` to define a command for **strongly** emphasising words, as so:

```
...for \strong{strongly} emphasising...
```

A whole world opens up when you can write your own macros; in my own documents, whenever there's an issue of formatting that will re-occur, it gets a macro in case I change my mind later. But when things start to get more complex, there are better ways to do things — it's much better to use other people's solutions; *i.e.*, other people's packages.

This article is split into two halves. The first half, section 2, discusses small macros written to ease document production, with some examples of how I (personally) use macros to save time and effort. In the second half, section 3 on page 9, I briefly cover a bunch of packages that provide a whole heap of functionality that other people have kindly implemented — no need to do it yourself!

Formatting in this article Verbatim in this document is represented by typewriter text in `bright blue.` Packages & classes are typeset in a sans serif font, 'hyperlinked' to their CTAN documentation (where available; a couple won't work at time of publication but should in the future), and coloured dark red like the other external links in the document. Internal links within the document are coloured green.

2 Writing your own macros

Writing things like `Figure~\ref{fig:myfigure}` every single time a figure is referred to results in errors creeping in that are hard to detect; typical of such errors might include misspelling 'Figure', the absence of the non-breaking space (the tilde `~`, something even experienced LaTeX users often forget to use), and so on. The command `\newcommand` becomes a new friend to relieve these problems. Suddenly, it's easy to define commands such as `\figref` that will expand out to the figure string and the reference:

```
\newcommand\figref[1]{Figure~\ref{fig:#1}}
```

After adding this command to the preamble, I can now write `\figref{myfigure}` to refer to my figures. Two advantages are gained with this method: error-checking (the output will now always be consistent and correct or throw an error) and flexibility. If we wish to change our string to look like 'Fig. 7', it's a matter of changing one line in the preamble:

```
\newcommand\figref[1]{Fig.\,\ref{fig:#1}}
```

(Note the thin space `\,` which is appropriate in this case after the period and which is also often omitted — to the detriment of the output quality.)

While you can use the above macro in your documents, there is a macro package refstyle (see page 12) that does the job a lot more completely. It includes macros for all sorts of references (tables, sections, footnotes, *etc.*) and even allows you to reference more than one figure at a time, such as

```
\figref{myfig1,myfig2,myfig3}
```
Figures 1, 2 and 3

2.1 How *not* to use macros

I have now briefly introduced macros as a way to be productive. In the section after this one (page 5), I'll continue along these lines. But it is said that a little bit of knowledge can be dangerous. It's worth considering how macros should not be used. The following is a good template for inserting figures:

```
\begin{figure}[htbp]
  \centering
  \includegraphics{*}
  \caption{*}
  \label{fig:*}
\end{figure}
```

In fact, in my editor (TeXShop for Mac OS X), I've set it up so that dragging an image into a source document inserts this snippet automatically with the filename and label filled in with the path and name of the graphic respectively. All I have to write is the caption itself.

This isn't an ideal situation, however. Using such a template 'freezes' the formatting decisions at the instant it's inserted; if changes are desired on a large scale, repetitive search-and-replace is required. It is better to start to use macros that define the formatting from a single place. Here is an example of one of my first efforts. Coincidentally, it's quite similar to an example given by Dave Walden [3]:

```
\newcommand\insertfig[2]{%
  \begin{figure}[htbp]
    \centering
    \includegraphics{#1}
    \caption{#2}
    \label{fig:#1}
  \end{figure}}
```

At first, the attraction is clear. Isn't the following much easier to write?

```
\insertfig{figname}{This is an example figure caption.}
```

This version has much to offer, in that the formatting may now be adjusted from a single place in the document — a feature always to be aspired to. But we quickly run into trouble.

Firstly, semantics have been lost. The arguments to the command are unnamed and arbitrary. Consider a smart editor which parses the document for `\label` commands and builds a list from which to prompt for `\ref` instances. How will it know that `fig:figname` is a label to use?

Secondly, we begin to want to add more functionality. Say we want a short caption (for headers/footers and the Table of Contents) and a long caption (for the actual text). Well, easy. Just change the macro to accept an optional argument, just like the `\caption` command does. Okay, now let's say we want to stick in some graphics options, such as scaling. More work, more arguments. And don't forget the option to change the figure placement from `[htbp]`, which should always be considered in the fine-tuning of the typesetting of the completed document.

Now our previous calls with this command, throughout the entire document, need to be amended to take the other argument into account. This might not be that much work with a fancy search 'n' replace, but this is supposed to be 'LaTeX for *productivity*', right?

That was a long winded way of saying that the approach taken above is too coarse-grained for our application. As a rule of thumb, restrict new commands to performing a single action, to avoid such problems. In the above example, the only formatting is to centre the figure on the page. The following redefinition of the `figure` environment provides this:

```
\renewenvironment{figure}[1][\fps@figure]
  {\@float{figure}[#1]\centering}
  {\end@float}
```

Because we're overwriting the existing definition of the `figure` environment, it's necessary to use `\renewenvironment` instead of `\newenvironment`. `\fps@figure` is the default float placement (usually `tbp`), and the `\@float`...`\end@float` pseudo-environment is the internal LaTeX generalised method for creating floats.[1]

1. I should warn some of you that if this code is pasted into a regular document (not your own package), it must be preceded by `\makeatlatter` and ended with `\makeatother`. More details can be found in Robin Fairbairns' TeX FAQ answer '`\@` and `@` in macro names'.

2.2 Examples of my macros

Two sections ago, I discussed how macros should be defined for error-checking, flexibility, and consistency. In the previous section, I showed that macros should be as specific as possible; in general they shouldn't be used as a method for simplifying the input as an end to itself. Below, I'll give some examples of macros (and the types of macros) I often use.

2.2.1 Foot or margin (or end) notes

The first is a macro for adding notes to text. Generally, this refers to footnotes[2] but sometimes one might wish to annotate their texts in a slightly more interesting manner (→). Or even using end notes, as discussed by Dave Walden [4]. Semantically, these could be exactly the same thing, so the markup for denoting this should be the same. Therefore,

As a more interesting example.

```
\newcommand\note[1]{\unskip\footnote{#1}}
```

Note the `\unskip`. This is included so that the note needn't be placed with no whitespace preceding it in the source. For example, `some text \note{A note.}` will appear as 'some text[3]'. Note the omission of the space before the superscript. Now, to adjust this definition to send the note out to the margin instead, it's simply a matter of a new definition:

```
\newcommand\note[1]{%
  \unskip~\marginpar{\hspace{0pt}\raggedright\small\itshape #1}%
  ($\rightarrow$)}
```

. . . or however. The `\unskip~` is a nice trick to ensure that there's never a line break before the note call-out in the text;[4] and the `\hspace{0pt}` is to ensure hyphenation in case of a long first word in the margin note.

In this case, the `\note` command adds flexibility to the document (it's easy to later change the definition to adjust the typesetting of the notes), and to enforce consistency in the output by controlling how the space around the note callout (be it superscript or other) behaves.

2. Such as this one. 3. A note.
4. I also use this in a redefinition of natbib's `\cite` command when using numerical references.

2.2.2 Abbreviations

The use of Latin abbreviations (*e.g.*, *i.e.*, *cf.*, *etc.*) in formal text isn't always encouraged. Nonetheless, they can be handy, and it's important to remember how they should be punctuated. Macros can address both of these issues by easily being able to switch out the abbreviations if necessary or ensure that the punctuation is always correct. Let's begin with a simple macro to mark up words in another language; *e.g.*, `\foreign{a priori}`.

```
\newcommand\foreign[1]{\emph{#1}}
```

Since 'e.g.' and 'i.e.' will generally always be followed by a comma, it's possible to define a macro to ensure this comma isn't omitted:[5]

```
\newcommand\ensurecomma{%
  \@ifnextchar,{}{\@latex@error{Don't forget the comma!}{}}}
```

Now, it's simply a matter of defining our abbreviations:

```
\newcommand\eg{\foreign{e.g.}\ensurecomma}
\newcommand\ie{\foreign{i.e.}\ensurecomma}
\newcommand\cf{\foreign{cf.\@}}
```

The `\@` ensures that no extra space is added after the period as it would if the period ended a sentence. These macros in use:

`I concur; \ie, I agree`	I concur; *i.e.*, I agree
`Add lots of sour; \eg, five lemons`	Add lots of sour; *e.g.*, five lemons
`Use white sugar; \cf\ brown sugar`	Use white sugar; *cf.* brown sugar

I usually also define the uppercase variants `\Eg`, *etc.*

Now here's some more definitions along these lines. Abbreviations such as 'etc.' and 'et al.' may occur at the end of sentences, so it would be unfortunate to insert the period incorrectly in these cases. Here we go:

```
\newcommand\ensuresingleperiod{\@ifnextchar.{}{.\@}}
\newcommand\etc{\foreign{etc}\ensuresingleperiod}
\newcommand\etal{\foreign{et al}\ensuresingleperiod}
```

5. Inside a package, `\PackageError` would be better to use than `\@latex@error`.

These are used similarly:

`Sentence ending, \etc.`	Sentence ending, *etc.*
`riverrun, \etc, livvy.`	riverrun, *etc.*, livvy.
`As discussed by Robertson \etal.`	As discussed by Robertson *et al.*

Of course, many people will argue against emphasising these abbreviations with italics, while others will disagree with using the abbreviations at all. The poet E. E. Cummings might have recommended omitting all of the punctuation. The definitions above can be adjusted appropriately to suit such requirements and make the appropriate changes throughout the entire document.

In this application, using macros for the abbreviations ensures both consistency and flexibility with the formatting, not to mention that using the macros precludes spelling errors in the output without an error in the compilation — provided they are used exclusively!

2.2.3 Punctuation

Many people, now, have written macros for inserting smart dashes into their documents. I copied the TUGboat macro [1] in this regard when writing the template for this very journal.[6] But I'm going to add a twist at the end, so keep reading.

There is more than meets the eye when using dashes in text — like this one. According to the tradition followed, differing amounts of space are used around dashes. From no space—like this—to a full-width space — such as here. In all cases, it's desirable to avoid line breaks before them, although a break *after* — like here — is fine. Others will like to use an en-dash instead – depending on the taste of the typographer and the font being used.

These points aren't what you want to think about while writing, and it's easy to get things wrong. An ideal case for a macro, then:

```
\DeclareRobustCommand\dash{%
  \unskip\nobreak\thinspace\textemdash\thinspace\ignorespaces}
```

This is the definition straight from the pracjourn class, and incorporates all of the details described above. The amount of surrounding space is customisable, and

6. The pracjourn class can be found at: `http://tug.org/pracjourn/styles/latex/`

the `\unskip`/`\ignorespaces` arrangement ensures consistency no matter how the macro is used in the text.

Here's a couple of small details to make things even better. First, we don't want to use the `\dash` definition above when writing things like PDF bookmarks, where plain text is the order of the day. So if the hyperref package is being used, the following line substitutes an ASCII dash in such cases:

```
\pdfstringdefDisableCommands{\renewcommand{\dash}{ - }}
```

Secondly, it can be a little distracting having to literally write '`\dash`' all the time in the source. Those of us lucky enough to be using unicode-aware editors might prefer to use a literal em-dash in our source to denote a text dash, naturally enough. After `\usepackage[utf8]{inputenc}` in the preamble, it is then possible to bind the meaning of `\dash` above to a UTF-8 em-dash in the source with the following incantation:

```
\DeclareUnicodeCharacter{2014}{\dash}
```

A snippet of the source document could then *look* like 'this is a dash — in the source', but it would be typeset according to all the rules given above. I consider this a great advantage for readable source!

2.3 Where to keep your macros

I generally use macros on a per-document basis. Others may prefer to keep their macros all together in one place. Both have their advantages and disadvantages. Since each document requires different logical markup, I write and adapt my macros as I go, copy and pasting from previous documents.

Alternatively, it can be very convenient to write a private package incorporating all of your own macros. This can be as simple as starting a new file `mymacros.sty` with the line

```
\ProvidesPackage{mymacros}[2006/08/15 v0.1 My custom macros]
```

Placing this file in the 'local' `texmf` tree of your distribution[7] then provides a com-

7. *E.g.*, `~/Library/texmf/tex/latex` on Mac OS X, `C:\localtexmf\tex\latex` on Windows, or `/usr/local/texmf/tex/latex` on Linux.

mon location to maintain your macros, accessible with `\usepackage{mymacros}` in your documents.

I don't use this method personally because I'm scared of backwards compatibility problems, although with enough foresight this shouldn't be a problem in general. Furthermore, this method assumes you want the same output from your macros in each and every document, and I don't always find this to be the case.

Further information on LaTeX class and package writing can be found in the documentation file clsguide, which can be found in `$TEXMF/doc/latex/base/` or by using the command line: `texdoc clsguide`.

2.4 Mini summary on macros

Bear in mind I've only touched the surface of how macros can be used to help you write documents more efficiently. I haven't even mentioned the xspace package, which simplifies how LaTeX commands deal with following space; `\TeX the Book` and `\TeX\ the Book` would produce identical output — no need to worry about how spaces are gobbled:

```
\let\oldTeX\TeX
\renewcommand\TeX{\oldTeX\xspace}

\TeX the Book; the \TeX{}book          TeX the Book; the TeXbook
```

When writing your own macros, be as fine-grained as possible; a macro should only do a single thing, ideally, and if more is required then write two macros. This was exemplified when I discussed wrapping up a figure environment into a single macro — in my opinion, more trouble than it's worth!

The macros shown in this section are supposed to simply be indicative of how *I* use macros. The sky's the limit when writing your own documents. Be creative! Just remember, when writing your own documents, if you find yourself writing repetitive formatting commands, a macro would make it more convenient.

3 Using others' macros in packages

At some stage in the development of learning LaTeX, we've learned how to write our own shortcuts and definitions well enough that we're no longer intimidated by delving into other people's packages and classes to see what's going on. I don't know about you, but I was pretty pretty happy with myself that to customise captions, say, I understood that all I had to do was edit the following from the article class:[8]

```
\long\def\@makecaption#1#2{%
  \vskip\abovecaptionskip
  \sbox\@tempboxa{#1: #2}%
  \ifdim \wd\@tempboxa >\hsize
    #1: #2\par
  \else
    \global \@minipagefalse
    \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}%
  \fi
  \vskip\belowcaptionskip}
```

So if I needed, say, an en-dash instead of a colon after 'Figure' or 'Table' in my captions, it was simply a matter of copying the above and changing the appropriate parts to `#1~--~#2`.

I used to do things this way, until I came to the realisation that I was wasting my time. Why? Other people had already solved to problems I was having. What happens if we want to start using the hyperref package? Our macro redefinition spoils everything, because now we can't link to the figure caption. Similar problems arise adding hyperlink support to the `\figref` command from the introduction. The point is that many packages are designed to fit in around each other, and doing this is not always a straightforward task when you're trying to hack your own support.

Things work fine doing simple things, and for one-off solutions sometimes it can be quicker to hack your way to output that looks right. This is an oft-repeated criticism of LaTeX in general: it's so much easier in Plain TeX to do this-or-that with a quick macro (re)definition. What is wrong with these sorts of ideas it that LaTeX isn't complicated because its authors wanted to obfuscate their work; rather, various functionality has worked its way into its facets that cover edge cases you're not even considering when trying to hack your own way.

8. To make such changes, you'd look through the class and copy/paste the snippets you were interested in modifying to the preamble of your document (see footnote (1) on page 4) or your own class file. The article class can be found at `$TEXMF/tex/latex/base/article.cls`, with documentation at `CTAN:macros/latex/base/classes.dtx`.

3.1 Choosing classes and packages

In the previous sections, I've shown two tiny sets of macros that make my writing more productive. Perhaps they've inspired you along similar lines. But in many cases, it's not the best thing to do, to build up your own macro packages from scratch, because other people will have done it before. Don't waste time replicating the work of smarter and/or more experienced people!

There are a huge number of packages on CTAN which are conveniently organised by category in the TEX catalogue: http://texcatalogue.sarovar.org/bytopic.html. There's a very high chance that something you wish to do is contained somewhere within. But where to start looking?

First off: choose your class. For beginners, I highly recommend an 'all-in-one' class such as memoir (well summarised in this issue of The PracTEX Journal [5]) or one of the KOMA classes (also featuring in this issue [2]). These have the huge advantage of a single reference. If the formatting requires adjustment, simply search through the manual. There's not much of an easier way to get started on your own with LATEX.

These integrated classes are more resistant, however, to being amended with packages. And in some cases, single-purpose packages provide more features. These days, I use memoir for large documents and article with packages for smaller things.

To follow is an incomplete, subjective list of packages that shouldn't be overlooked. I've left out the ones that everyone knows, such as geometry and hyperref. In the spirit of the article, you shouldn't take my word for it, but look into any other options available and see if there are alternatives that are better (for example, I've frequently heard the typearea package is easier in many cases than geometry, but I must admit to have failed to investigate it).

Note that there are a few packages for formatting section titles and contents tables (titlesec, sectsty, titletoc, tocloft) and I'm not qualified in all of them enough to give a firm recommendation on their use. Personally, I've enjoyed the simplicity of the sectsty packages, but it is very limited.

booktabs	For great tables. The manual is worth reading for its advice alone.
caption	Customise the formatting of captions used for figures, tables, and anything else.
csquotes	Error-checking and flexible quotation markup. *E.g.*, `"quoted text"`

	can be typeset with quotation symbols chosen in the preamble, such as ‘ ’ or “ ” or « ». This is useful not just for multilingual documents — it also helps with ensuring quotation consistency.
bigfoot	Does absolutely everything related to footnotes.
enumitem	Provides almost everything one could want for formatting lists (such as the one you’re reading).
fixltx2e	Fixes and adds many little details. Better to be safe than sorry!
mathpazo	Don’t miss out on the `[sc,osf]` options to activate better kerning & real small caps, and old-style (lowercase) numbers. Also see the FPL Neu font, which provides further improvements with maths support soon to come.
microtype	Enables pdfTeX’s microtypographical features: margin kerning (active for this document) and font expansion (gives better output but slower processing and greater file sizes — best for print work).
natbib	Formatting citations in either author/year or numerical format. The hypernat package is essential when using numerical citations to allow the `sort&compress` feature to work in conjunction with hyperref.
pdfcolmk	Fixes colour in pdfTeX, which can sometimes break (half a page of coloured text for no reason, for example).
refstyle	The most convenient and flexible method for cross referencing, as discussed in section 2. Also uses the varioref package for smart ‘on the following page’–type functionality.
SIstyle	The most convenient way to typeset numbers with units. *E.g.*, instead of `1.2\times 10^{-3}\,N{\cdot}m`, just write `\SI{1.2e-3}{N.m}`.
subfig	For creating sub-tables and figures.
textpos	For putting things anywhere on the page, even using absolute positioning (*e.g.*, 3 cm from the top of the page, 4 cm left of the margin).
xcolor	Very flexible colour functionality.
zref	When it’s released, looks to be amazingly comprehensive: reference anything.

3.2 Finding packages

There's little advice I can offer on how to find out about packages that have the functionality that you're looking for. Every time I look, there's more on CTAN[9] and the TEX Catalogue[10] that I've never seen before. Various esoteric things, such as sorting index entries from within TEX itself (see Kees van der Laan's BLUe format), have been implemented ten years ago! (I hope to look into this format in more detail in the future.)

Reading `comp.text.tex`, I am regularly surprised by the breadth of packages that already exist. When looking specifically, I can only recommend exploration of CTAN, and asking questions of people who have done similar explorations themselves.

4 Summary

Formatting shouldn't be hard-coded into a document; macros to ensure flexibility and consistency are preferable. There are many packages available that provide such functionality, along with a plethora of other convenient customisations. These should be experimented with!

References

[1] Robin Fairbairns. The new (LATEX 2_ε) TUGboat macros. *TUGboat*, 17(3):282–288, September 1996. ISSN 0896-3207. URL `http://www.tug.org/TUGboat/Articles/tb17-3/tb52guid.pdf`.

[2] Yuri Robbers, Markus Kohm, and Rasmus Pank Roulund. Replacing LATEX 2_ε standard classes with koma-script. *The PracTEX Journal*, 3, 2006. URL `http://tug.org/pracjourn/2006-3/robbers`.

[3] David Walden. Travels in TEX land: LATEX for productivity in book writing. *The PracTEX Journal*, 2, 2006. URL `http://tug.org/pracjourn/2006-2/walden`.

[4] David Walden. Travels in TEX land: Final layout of a book. *The PracTEX Journal*, 3, 2006. URL `http://tug.org/pracjourn/2006-3/walden`.

[5] Peter Wilson. The memoir class. *The PracTEX Journal*, 3, 2006. URL `http://tug.org/pracjourn/2006-3/wilson`.

9. `http://tug.ctan.org/` 10. `http://texcatalogue.sarovar.org/`