Title – template for orgmode latex production

Hans Fangohr, University of Southampton

February 2, 2016

Abstract

This is an abstract abstract, in the sense of only providing a virtual abstract, also known as the interface. Somebody will have to provide an inherited class that provides the real abstract.

We have placed the abstract in the paper.tex file, so that all the content in the orgmode file `content.org` is organised into sections, and they can be unfolded, re-arranged, etc (the abstract doesn't go well into a section because it appears even before the first section starts). If you prefer, you can move the abstract latex definition as is into the `content.org` file, and all will work as before.

1 Introduction

Some introduction to the topic. Orgmode is cool, see `http://orgmode.org` fore details. URLs are converted into hyperrefs automatically (i.e. no need to use the `\href{}{}` command).

2 Method

2.1 This document

The overall workflow is this: the main part of the paper is in the file `content.org`.

This is translated by a (emacs) script into a full LaTeX document with name texttt{content.tex~.

An extra (python) script strips off the LaTeX preamble and the last few lines at the end of the document, and saves the remaining middle part of the document as `\_content.tex`.

The main latex file is `paper.tex`, which contains the latex preamble, all the style the requirements we or the journal may have, the abstract, the bibliographystyle and bibliography uses the `\input{_content.tex}` file to include the autogenerated latex into the main document.

There is a Makefile that automises all of these steps. It may be useful to run a pdf viewer that watches and automatically updates `paper.pdf` on any

change, and to have a loop (or the `watch` command that repeatedly calls `make`, so that the pdf is re-created when the modified `content.org` file is saved).

2.2 Subsection 1

Our document may well contain subsections. In fact, part of the joy of orgmode is that we can re-arrange sections and change their depth quite easily.

2.3 Details

We can use equations as if the orgmode source was LaTeX, for example $f(x) = x^2$ or

$$\int f(x)\mathrm{d}x = C \tag{1}$$

3 Results

3.1 Tables

Tables are converted into tables:

name	#
Apple	4
Pears	5
Tomatoes	16

3.2 Figures

Figures can be included like this

Or we use the latex command directly:

We can also provide guidance to orgmode how we would like the figure included (see `http://orgmode.org/manual/LaTeX-specific-attributes.html` for details):

If you prefer, you can inculde the full float figure environment as raw latex into the orgmode file. Here is an example for such a figure, and then we can refer to it through its figure number 1.

Figure 1: The skyline

3.3 Figure and Table environments

The `+CAPTION` directive instructions orgmode to create a caption, and wrap up the following graphic or table into a Table or Figure environment.

Table 1: This is the caption for the next table (or link)

Apples	1
Bananas	3
Fruit	4

Figure 2: This is the caption for the next Figure

We have just created figure 2.

3.4 Code

We can also include code. Again, it can be done directly from orgmode (and the option to also execute the code from within orgmode and include the output is exciting, but goes beyond the purpose of this template for document writing in orgmode. If you want to explore this further, look here: `http://orgmode.org/manual/Working-with-source-code.html#Working-with-source-code`

Figure 3 shows some code.

3.5 Include literal LaTeX

If necessary, we can use the `#+LATEX:` directive, to send a string directly to LaTeX, i.e. unmodified by orgmode.

We can also create a literal LaTeX block like this one.[1].

3.6 More results

Of course we can cite work [1].

[1] See http://orgmode.org/manual/Quoting-LaTeX-code.html for more details

```
def adder(n):
    def add_n(x):
        return x + n
    return add_n

f = adder(42)

print("What is the answer?")
print(f(0))
```

Figure 3: An example script in Python.

4 Discussion

Coming back to the method outlined in section 2.1, it may well be possible to achieve a similar setup without the Python script extracting the main part of the autogenerated document etc, and include all the required latex setup, extra packages, into special #+ commands in the orgmode file. However, I have found it efficient to be able to use journal latex templates directly, and thus came up with this arrangement. Not perfect, but a functional start.

5 Summary

I like using orgmode to author documents as the orgmode mark up is less intrusive (and overall fewer characters to type!) than the LaTeX mark up. Rearranging sections, and changing the depth (i.e. move sections to subsections etc) are trivial in orgmode. Overall, orgmode allows me to focus more on the content of the document and its structure.

6 Acknowledgements

Thanks to Sam Sinayoko for introducing me to his way of creating beamer latex slides from orgmode, and who wrote the original elisp script that executes the conversion of orgmode files to LaTeX. Thanks also to Maximilian Albert, who helped tidying up the Makefile.

7 TODO

Sometimes, a section with things to do is useful; with the understanding that this is completed and removed before the document is finished. (Or changed into a COMMENT section, so that it doesn't export to latex.)

An orgmode todo list (which can be nested) looks like this

- TODO [2/3]
  - ☒ create github repository
  - ☒ write up the setup for this document
  - ☐ Save planet [0/3]
    - ☐ understand challenge
    - ☐ find solution
    - ☐ implement it

References

[1] S. Body. About orgmode. *elisp reviews*, 2016.