Imperial College
London

A template document

Author 1
Department of text editors, Vim University
author.1@vim.co.uk

Author 2
Department of text editors, The Emacs Institute
author.2@emacs.co.uk

This is the abstract of a template document which can be edited. The content of this document can be modified and added to as necessary. It provides examples of how to handle maths, tables, data analysis and images when making a document using knitr.

1 Introduction

The text in this document can be replaced. This document's source code (`paper.Rmd`) is available at https://github.com/tomravalde/workflow, along with the scripts and dependencies required to build the PDF. The GitHub repository provides the requirements and instructions in order to be able to produce this document and others like it. This workflow is based on that described in Healy (2014) and uses Markdown, LaTeX, R, the knitr R-package (Xie, 2014) and pandoc (MacFarlane, 2014).

2 Maths

You can mix and match Markdown with LaTeXsyntax, for example for mathematical typesetting as in Equation (1):

$$a^2 = b^2 + c^2. \tag{1}$$

3 Tables

When typesetting tables (such as Table 1), I like to use the `xtable` function (from the `xtable` R package). This allows you to create your table in a spreadsheet which you then save as a CSV file (see `text-editors.csv`).

4 Processing and presenting data with knitr

By the magic of knitr, data can be processed and presented using the same source code as the rest of the document's content. An example is given in Figure 1.

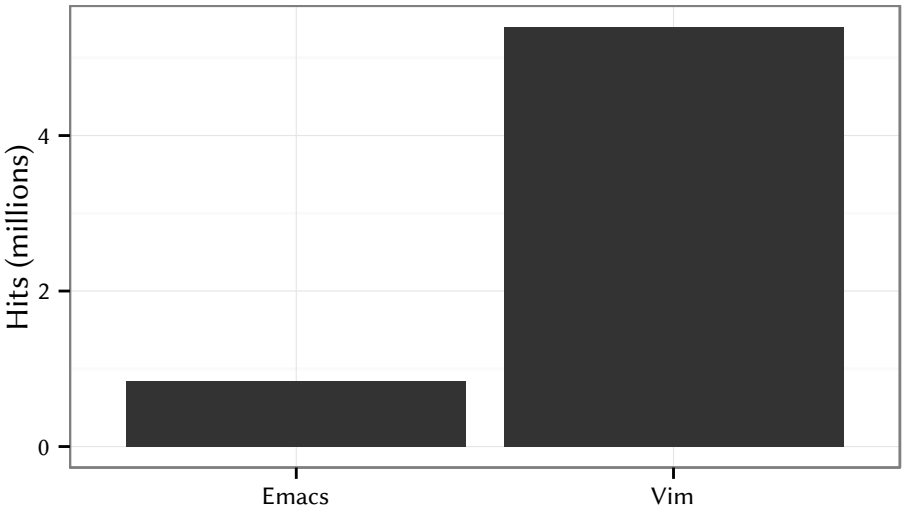

Figure 1: Google fights results.

5 Images

Unfortunately, using Markdown to specify a figure size in a PDF is near-impossible. So for figures, I use standard LaTeXcode, as for Figure 2.

References

Kieran Healy. Plain Text, Papers, Pandoc, 2014. URL http://kieranhealy.org/blog/archives/2014/01/23/plain-text/.

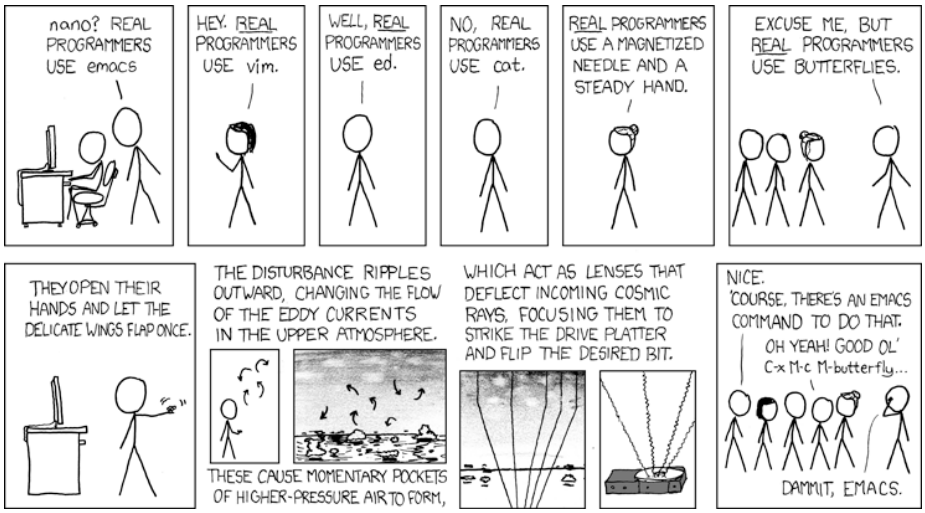

Figure 2: Text editors according to XKCD (http://xkcd.com/378/).

John MacFarlane. Pandoc, 2014. URL http://pandoc.org/index.html.

Yihui Xie. knitr: A General-Purpose Package for Dynamic Report Generation in R, 2014. URL http://yihui.name/knitr/.

Function	vi	Emacs
Keystroke execution	vi editing retains each permutation of typed keys . This creates a path in the decision tree which unambiguously identifies any command.	Emacs commands are key combinations for which modifier keys are held down while other keys are pressed; a command gets executed once completely typed. This still forms a decision tree of commands, but not one of individual keystrokes .
Memory usage and customizability	Historically, vi is a smaller and faster program, but with less capacity for customization. The vim version of vi has evolved to provide significantly more functionality and customization than vi, making it comparable to Emacs [notes 1] . vi start-up time is near instantaneous for small text files, while vim is almost as fast.	Emacs takes longer to start up (even compared to vim) and requires more memory . However, it is highly customizable and includes a large number of features, as it is essentially an execution environment for a Lisp program designed for text-editing.
User environment	vi was originally exclusively used inside of a text-mode console, offering no graphical user interface (GUI). Most modern vi derivatives, e.g. MacVim and gVim, include GUIs. However, support for proportionally spaced fonts remains absent. Also lacking is support for different sized fonts in the same document.	Emacs, while also initially designed for use on a console, grew a GUI fairly early on due to its Lisp machine heritage. Current Emacs GUIs include full support for proportionate spacing and font-size variation.
Function/navigation interface	vi uses distinct editing modes .	Emacs uses metakey chords. [notes 2]
Keyboard	vi uses no <Alt> key and seldom uses the <Ctrl> key. So vi uses only most of the "usual" keys.	Emacs uses <Alt> and <Ctrl> keys, and observes the <Shift> variety regularly . Emacs uses all the usual keys regularly, which tends to encourage ambidextrous usage of their symmetry.
Language and script support	vi has rudimentary support for languages other than English. Vim is partially multilingual, with support for European, Arabic, Hebrew, and Far East Asian language support only. Notably, Indic language and script support is absent.	Emacs has full support for all Unicode-compatible writing systems.

Table 1: The differences between vi and Emacs, from Wikipedia (http://en.wikipedia.org/wiki/Editor_war).