Using Microsoft Word with git

Martin Fenner, Gobbledygook

August 25, 2014

One of the major challenges of writing a journal article is to keep track of versions - both the different versions you create as the document progresses, and to merge in the changes made by your collaborators. For most academics Microsoft Word is the default writing tool, and it is both very good and very bad in this. Very good because the *track changes* feature makes it easy to see what has changed since the last version and who made the changes. Very bad because this feature is built around keeping everything in a single Word document, so that only one person can work on on a manuscript at a time. This usually means sending manuscripts around by email, and being very careful about not confusing different versions of the document, which requires creativity.

Approaches to overcome these challenges are to a) integrate the Word documents into collaboration tools such as Sharepoint and Office 365, or document sharing services such as Dropbox and Google Docs (if you use it just for that), or b) use a different authoring tool altogether. If neither of these approaches works for you, you have a third option: use the version control system **git**.

Git is software that helps with tracking changes to files so that you can recall specific versions later. Git is typically used to track changes of software source code (and was originally developed by Linus Torvalds for Linux kernel development in 2005), but in fact git can be used for any file where we need to keep track of versions over time. Git is open source software that runs locally on your computer, so please go ahead and start tracking changes to your manuscripts (or other complex documents) with git. Any time you want to store a version, do a `git commit` with a little description and an optional tag.

This approach is not ideal, as git was written with source code in text format in mind and for example doesn't understand what has changed between two revisions of a Word document. Some people will tell you to never store binary files in a version control system, but don't listen to them. Instead give git a tool to convert Word documents into plain text, and git will then happily tell you what has changed between revisions. Several tools can do this, but since earlier this month Pandoc can read Word documents in `docx` format. Do the following to have Pandoc convert Word documents into markdown, and to compare the revisions by word and not by line (which makes more sense):

```
# .gitattributes file in root folder of your git project
*.docx diff=pandoc

# .gitconfig file in your home folder
[diff "pandoc"]
  textconv=pandoc --to=markdown
  prompt = false
[alias]
  wdiff = diff --word-diff=color --unified=1
```

You can then use `git wdiff important_file.docx` to see the changes (with deletions in red and insertions in green), or `git log -p --word-diff=color important_file.docx` to see all changes over time.

While you can now track revisions of a Word document and see the changes, you also want to be able to merge different versions of a Word document together so that you and your collaborators can work on the manuscript in parallel. Git can't merge binary files together, so you need to first convert the Word document into a format that git understands. Just as in the previous example we can use Pandoc for that, with markdown as the textual format. This would also work with HTML or LaTeX, but the simplicity of markdown makes it better suited for version control which doesn't know about the markup of these formats.

One of the reasons that git became so popular with software developers is that it is a **distributed version control system** instead of a centralized system such as Subversion. This means that you can track all revisions locally on your computer, but can still synchronize your revisions with another user. **Github** is a popular service that facilitates this synchronization and adds some nice features on top. One way to collaborate with your co-authors is therefore to set up a Github repository (public or private) for your manuscript, and store the master version of the manuscript in markdown format. Instead of working on the master version directly, you would use Pandoc to convert back and forth between this master version in markdown format and your Word document, and would continue to use Word as authoring tool. Rakali is a Pandoc tool that I released last week that can help automate this document conversion. Github has a a number of features to facilitate collaboration that can be used here, e.g. Github issues for discussion and task management.

There are still a few rough edges in the workflow described above (e.g. only partial support of Word track changes), but it is an interesting approach to collaborate using Microsoft Word and git. And this workflow can of course be enhanced to also include authors that write in LaTeX or one of the other formats that Pandoc supports. One nice side effect of using markdown is that Github will automatically render a webpage for the document (which it will not do for HTML without extra effort).