Writing Word documents in RStudio

For collaborating in research it is often necessary to produce Word documents (.docx format) for sharing with co-authors for comments and changes. To produce research outputs that are reproducible, and to avoid spending any more time than is necessary in Word (plus having a system that doesn't 'corrupt') we can use RStudio, BibDesk, and a few setup steps to write documents in markdown and output in Word. (I'm assuming that you have a Mac).

If that previous sentence made no sense at all, don't worry. Each of these things is simple. But before explaining what each is, it's good to step back and address The Big difference between producing documents with Word and in this alternative, more complicated *sounding*, way.

The best analogy that I can come up with is the difference between cooking and baking. When we cook, we follow a recipe up to a point, but as we're stirring the pot we might add in a few extra ingredients, change the seasoning, cook for a bit longer, freeze the mixture and use it in something else, etc. When we're cooking, there's a 'thing' (the food) that we're creating over time, it's difficult to undo most or any of the steps, and if it goes wrong then we have to just start again and there's no way to recover anything that we put in. When the food is good, we can't always say how it was made, because there were so many little changes from the recipe that only a good cook would know to make.

When we're baking, we don't have any real-time control. Instead, we follow a recipe closely, and prepare all of the ingredients carefully in advance. Each ingredient is prepared according to its own needs: flour is sieved, vanilla pods ground, orange peel grated, etc. Bringing the ingredients together to bake is usually a one- or two-step process, and if we've read the recipe correctly, and prepared the ingredients properly, we'll be alright.

Working in Word is like cooking. In Word what-you-see-is-what-you-get; you're gradually building up the document with little changes here and there. Unless you have a separate version-control system (or save multiple versions of the same document) then there it is difficult to pull the ingredients apart and start again.

The alternative approach that I'm explaining here is like baking. You spend the vast majority of your time preparing different files, some big and some small and each with its own particular requirements. Then, when you're ready, you run a programme that 'bakes' (actually referred to as 'knits') the ingredients together to produce the output. If you don't like the output you can to change the ingredients and run the programme again.

So, lets assume that you're interested in this baking approach to writing documents and would like to hear how it is done (before considering in more detail the advantages and disadvantaged relative to just sticking with Word). Let's go back to that sentence in the opening paragraph: here's what you need to prep the ingredients (knives, graters, and sieves):

- **RStudio**. RStudio is a free and open-source software for using R, the statistics package. In theory, you don't need something as powerful as RStudio but it doesn't hurt. Using RStudio will be useful if you ever plan to combine R with your documents. So, step one, is download RStudio, and install it (and R) on your laptop.
- **BibDesk**. A key aspect of academic writing is referencing. Again, this is not the only option available, but it's free and open source and works pretty well. BibDesk

organises your references as a BibTex file, which means that unlike other programmes (Mendeley for example), your reference library is a long text file that's as easy for you to read as what you're reading now. That makes the format much less liable to becoming corrupted. BibTex files are available for import and export from all the major reference managers so it's easy to move your references into BibDesk. So, step 2, is download BibDesk and move over your library of references.

- **Markdown**. The final tool is a bit different, it's not a programme but a language for formatting writing. That sounds scary for a lot of people, but it's really simple. In markdown, you just write normally, with line-breaks to mark the separation between paragraphs, etc. When you want to write a title, you put a # sign at the start of the line. When you want a sub-title, you put ## (two pound signs) at the start of the line, ### for a sub-sub-title, and so on. If you want to add bullets, you start a new paragraph with a dash (-). Numbered lists are numbered lists: just start a new paragraph with '1.' and go from there. To italicise text, you just add one * on either side of the word or phrase, to bold text you add two (**). For better of for worse, it's basically that simple. The better is that it's easy to remember, fast, and doesn't clutter up your page with lots of stuff other than the words you're trying to write. The worse is that it's a bit restrictive. There's a few more options than I've covered above, but for 95% of your writing the commands above are all you'll need. So, to start using markdown you don't need to do anything for now, RStudio works with markdown so it's just the language that we'll use to format our documents.

At this point you should have RStudio (and R) installed, and BibDesk installed and loaded with your references. These are the tools that you will be using. So what are the ingredients?

The ingredients that will make up your document are:

1. Your BibTex library, managed in BibDesk
2. A Word file with formats
3. A reference format
4. An RStudio file called an ‘Markdown’ file

By visiting this link, you can download an example setup to work from. Just unzip the file and put it where you want to practice.

Lets set up each of these in order. Hopefully you already have the BibTex library going. The key thing is to know where it is saved, it can be anywhere really, but just make a note of where it is. If you’re working on a big project, like a PhD, you might want to create a specific library for the project, in a folder called ‘references’ – that’s what I’ve done in the example.

The second thing you need is a Word document. What? But I thought we were avoiding Word! We are, as much as possible, but since the output we are producing is a Word document we need to add just a pinch of .docx to get the output formatted properly. Although we can use markdown to tell the programme that some text is a title, how is the programme to know what the title should look like? That’s where this .docx file comes in; by saving a more-or-less empty file (it’s fine to have some text in there so you can see how things look) with the styles defined as we want them, the programme that ‘bakes’ the ingredients can look in this document and know how to make everything look.

If you’re using the example setup, open the Word file and observe that there are formatted headings and sub-headings. You can modify these formats using the styles pane. If you’re happy with them as they are, for now, then you can just close this file and move to the next step.

The third thing that we need is a file that defines how the references are going to be formatted. These can be found using Google; Zotero seems to have a nice collection of them online, or you can often get them from journals too. If you are submitting to a journal, it’s good to get the right formatting file before you start the paper so that you’re not worrying about it, or forgetting about it, at the last minute.

As you can see in the example, it’s fine to have more than one file in here. In the next and final set-up step we’ll see how we can tell the programme which one we want it to use.

CREATING YOUR RMARKDOWN FILE.

This is a big step, and warrants it’s own sub-heading. The Rmarkdown file is the closest thing to a Word document: this is where your text goes, and also where all the other ingredients are linked together.

First, the linking together. At the top of an Rmarkdown file there is a series of commands in what is called the ‘YAML front matter’ by some, but let’s just call it the linking commands. The linking commands do a bunch of different things. The title of the document is defined at the top, and the authors – add your own title and name here. Then there are a bunch of formatting based commands. The most important of these are the ones that make the links with the other ingredients. So check that the link to the .docx file is correct, same for the reference formatting file, and the BibTex library. (Note that these links can be ‘relative’ to the Rmarkdown file, which is great if you move your folder from one place to another.) Get these links right, and we’re pretty much there. You might notice a command ‘TOC: false’, this will determine whether or not a table of contents should be produced during the ‘baking’; if you want one, just change ‘false’ to ‘true’.

After the linking command section is the body of the file, this is where the writing happens! Finally! Here is where you can flex your markdown skills and enjoy writing in a simple, clean, and distraction-free editor, safe in the knowledge that all the formatting will be handled separately (that’s how I think of it anyway).

As you will see in the example, we can create titles, headings, and format text as described above. There’s also a table for illustration – tables are a bit more advanced, and not necessary, but you can probably work out what’s going on from this example (that’s part of the idea with markdown, that it looks fairly self explanatory).

You will also see in the text that ‘[@Name:1234]’ has been written in various places (with an actual name and an actual date). This is how you put in references. The stuff after the @ sign is called a ‘cite key’ and you can get the cite keys from BibDesk. Open BibDesk and highlight all of your library (command/ctrl - a), and then click on Publication > Generate Cite Key (or just command/ctrl - k), and click ‘okay’ if there is a warning. This will assign cite-keys to each of the references in your library, of the form ‘Name:’ and then the date published (if the author has more than one paper from the same year, BibDesk will add a letter, usually ‘a’, to the end to distinguish them). Double-clicking on a reference will open a window with all the info, with the cite-key highlighted. I find that to get the cite key quickly I can open this window, command/ctrl-c to copy the cite-key, and get back to RStudio pretty quickly (N.B. there are quicker ways, but they involve a bit more set up and this way works pretty well). Back in RStudio, to cite a reference you just type '[@' (RStudio will usually add in the closing bracket for you) and then paste the cite-key, and you're done!

BAKING (KNITTING) YOUR DOCUMENT

Once you've finished working on your document, including adding all your references, you're ready to put everything together. If everything has gone well so far, this last step is embarrassingly simple. You have two choices: you can click on the button at the top of the document that says 'knit', or, if you can't even be bothered to reach for your mouse, you can hit command/ctrl-shift-k.

RStudio will run some things, actually more than one programme but you don't need to worry about it, and very soon there will be a Word document created in the same folder as the Rmarkdown file. Open that up and take a look. If it looks nice, you're done! If something isn't quite right (and there usually is something) then you will have to check you've done your markdown property, or that the cite-keys are all correct, or that the formatting is how you want it. That all sounds tedious, but it doesn't take long and the good thing is that each improvement you make will be useful for all the projects that you're working on, either directly (in the case of the formatting files) or by learning more about how to create beautiful reproducible research documents.