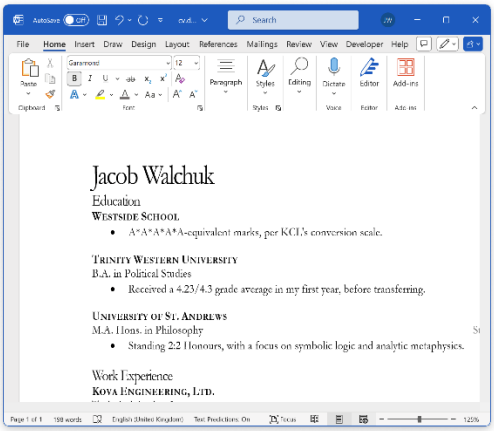
Programming Project Sample



In this write-up, I describe my most recent programming project, a CV GENERATOR that converts YAML-formatted data into a well-formatted Word document. I elaborate on how small projects like this lead to practical solutions in data analysis, and how a "full-stack" understanding, brought on by a breadth of experience in computing, can lead to more perceptive evaluation of tech equities.

Comparatively, however, the project discussed here is small. More complex – and ostensibly, more impressive – projects are available on my GitHub:

- BOOKTRACKER, whose refactor is in progress – the main link leads to an older version. This node.js tool tracks my reading and publishes the notes. Creating this involved reverse-engineering the SQL schema used by org-roam, as well as the SQL schema used by Zotero.
- MC-MAILER, a RESTful web service written in Rust which automated email verification for a Minecraft server using a salted hashing system. All with no database.
- A URL SHORTENER AND PASTEBIN written in Rust and actix-web, which used MySQL.

It is my hope that these projects show how I can not only add value to MCM in discerning the tech space, but also in *doing better analysis quicker.*

Introduction

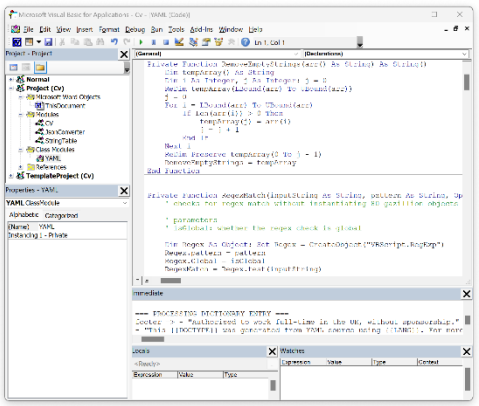
Having a vocational technology background – *i.e.*, a background in using technology to *get things done* – provides a more-than-sufficient grounding for data analysis, and, therefore, financial analysis.

I'm no stranger to problem-solving odd jobs. On one occasion, I needed to obtain a list of all IT consulting firms with less than 50 employees in the *EC*, *WC*, and *E14* London postcode areas. I was able to process a multi-gigabyte file from Companies House using the *awk* command, and import the resulting smaller file into Excel for processing. On another occasion, I used LISP – a functional programming language[1] – to hook into Anthropic's *claude-3.5-sonnet* LLM, and turn a database of point-form notes into summaries I could gain insights from.

All of these have something in common: they're *solo projects*, and don't have to work within the constraints of an organization. *I* chose the platform. Moreover, I did not have to present the data to an audience in the form of a presentation or report: as a result, I lacked fluency in the corporate world's chief tools – Word, Excel, and Powerpoint. It seemed as though I was due to write a project that used Office's automation tools to accomplish a complex task.

What did I *do*, exactly?

I wrote this CV generator as part of a larger pipeline that would allow me to edit my CV in a data serialization format called YAML,[2] and then generate well-formated *.docx*, LaTeX, and HTML files. As part of the *.docx* pipeline, I did the following:



1: Microsoft's VBA editor is antiquated, and by default, there is no way to separate the code from the workbook or document.

- Write a parser for YAML code in pure VBA; this can be re-used, for example, to configure a macro-enabled Excel spreadsheet with an external file.
- Write business logic which iterates through the data and sends it to functions that type out the CV.
- Devise a build system that allows me to use my normal text editor, *emacs*, which allows much more elegant keybinding-based editing than Microsoft's own UI. This also allows me to use normal version control software, such as *git*, and, in future, seamlessly use GenAI as a coding aid.

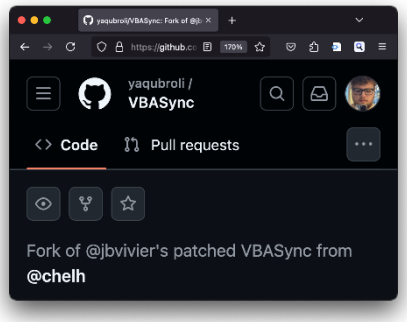
A full-stack understanding

A CV generator is a very low-stakes piece of software. In something more important – for example, an Excel workbook containing a maze of pivots and cash-flow-projection formulae – understanding each moving part is important. When *automating* this workbook, it is useful to be acquainted with the Component Object Model that

[1] Functional languages handle computation symbolically, rather than procedurally: in practice, this means that a few lines of code in LISP, or Jane Street's beloved OCaml, can elegantly perform what might take a complex network of loops in a more procedural language.

[2] Acronym for *YAML Ain't a Markup Language.*

Excel uses to expose its features to programmers: if one is trying to automate the addition of a table column to a slicer, for instance, and receives a cryptic error, they can decode it.



2: The CV generator project allowed me to fork and modernize a tool which may be indispensable to automating future analysis.

This "full-stack" understanding benefitted me when I was trying to access my VBA code from a normal text editor, and could not find a tool to accomplish this task that worked with non-Excel files, save for an older program called *VBASync*. In order to use the program, I had to compile it, and received an error. I:

- Quickly educated myself on how Windows (specifically, C# .NET) applications were compiled and packaged, to diagnose the specific error.
- With no specific knowledge of C#, but enough context clues from previous experience, I edited the code to create a version that works on modern platforms, and published it to GitHub as a fork; in the week it has been live, it has already received 2 stars from interested users.

Understanding each moving part of a piece of software also, of course, aids in actual valuation of equities tied to software. For instance, understanding the architecture of a WeChat Mini Program[3] would have placed someone evaluating an early-stage PinDuoDuo at a massive advantage. They would have recognized the sheer breadth of the total addressable market, as well as the low overhead for implementation, as Mini Programs are written in HTML, CSS, and Javascript. Consider, as well, a possible IPO of the Motion calendar app, which uses a similar solution – called Electron – to utilize the HTML/CSS/Javascript stack: in the case of a calendar app, this technology may hinder interoperability between Motion and other software to the point that it cannot provide value to consumers, and thus the equity is devalued.

Final thoughts

It is, again, my hope that this write-up provided some insight into how a hobbyist technical background can add value to MCM. While the work done at, *e.g.,* internships can be quantified on a CV, the technical fluency and element of communication that served as the *x*-factor in those roles is more effectively demonstrated longform.

As I mention in my 300-word application statement, MCM will allow me to not only exercise this technical fluency, but allow me to work with complex systems more broadly – complex systems where knowledge of everything from tech to languages to geopolitics can aid in ascertaining value.

[3] A uniquely Chinese way of creating a platform-agnostic mobile app – PinDuoDuo, and its exclusively-agricultural predecessor, PinHaoHuo, were originally only available as WeChat Mini Programs.