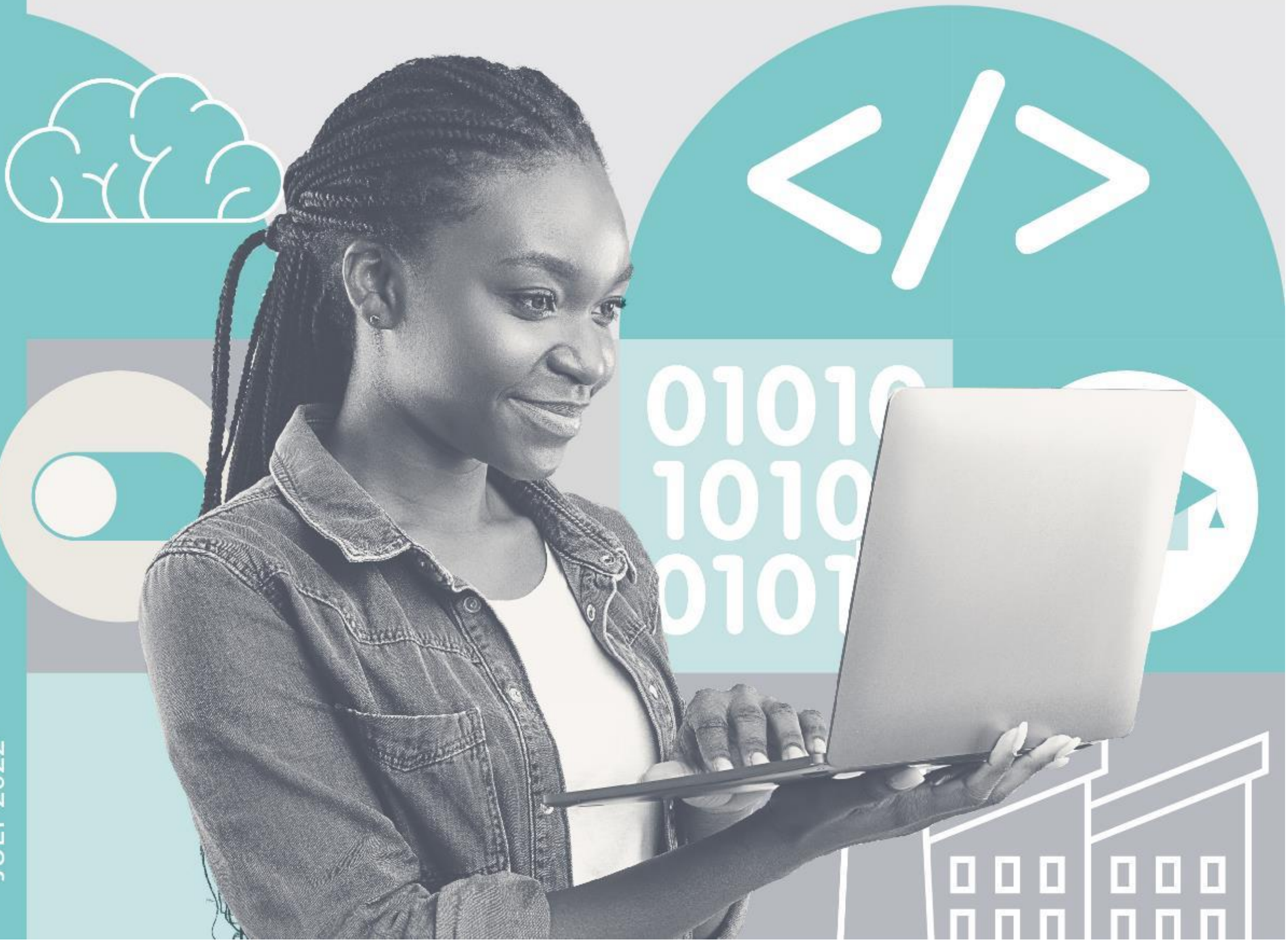
CODE DEVELOPMENT FOR ACADEMIC ENTERING INDUSTRIES

Data Science: from Academia to industry - supplementary report

Stephen Haben
Digital And Data Consultant (ESC)

Samuel Hinton
Senior Data Scientist (ARENKO)



JULY 2022

CONTENTS

1. INTRODUCTION

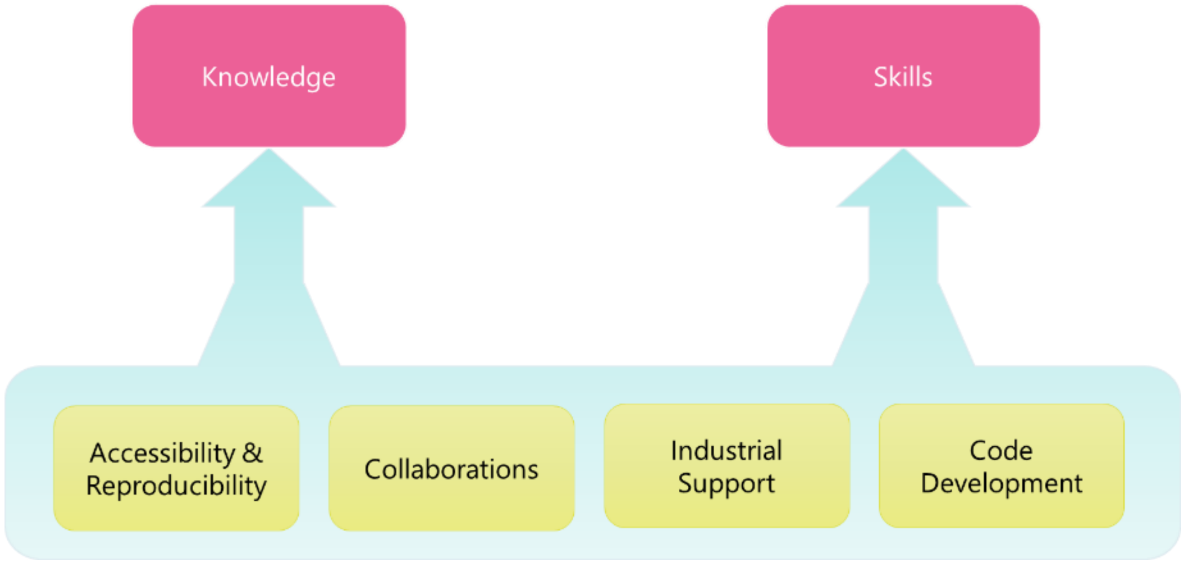


Figure 1. Illustration of the structure and the dependencies for our reports on academic driven impact within industry. The two main methods of impact are knowledge and skills (in pink) and these are supported by four supportive mechanisms (yellow). Each of the supportive mechanisms is explored in more detail in supplementary reports. This report focuses on Code Development for Academics.

The research produced in academia can have positive and wide ranging impacts industry and is essential for supporting the innovative data science products and services if the energy sector is going to achieve Net Zero. The main report discuss the two main impacts that academia can have in industry:

- Technical knowledge, and
- Skills

This report is one of four supplementary texts which delve into supporting mechanisms for these types of impact. This report considers **Code Development for Academics.** In data science academic papers can have limitations in fully describing the nuances that are required in developing their code. As discussed in the accessibility and reproducibility report, sharing code can support the methods and analysis described in the accompanying paper and can help an industry organisation utilising the code to fully reproduce or replicate within their organisation. Coding skills are a vital component of operational data science but are often a neglected part of the skills which are developed at universities where the focus is fundamentally on the technical models and their accuracy. Within industry writing good quality, reliable code is often more important than the complexity and there is a difficulty in recruiting data scientists who have all round domain knowledge, modelling knowledge and programming skills.

This short supplementary report will cover some of the learnings from the interviews we conducted and provides some guidelines on tools and resources which could be useful for developing necessary coding skills for an academic leaving industry.

2. CODE DEVELOPMENT FOR ACADEMICS

The technology stack is very difficult for academics to keep up to date. As one interviewee mentioned, software development is often 10 years or more ahead of the general academic research community. Every year there are more technologies which are used extensively within data driven organisations and the timeline is perhaps too quick for the slower moving academic community who often teach similar material from year to year. Further, academics may not have the experience in these new areas to be able to appropriately teach it in a way which be suitable for industry. Leaving university with these additional skills is perhaps more pertinent for smaller organisations compared to bigger ones.

Further, although researchers and PhDs are quite good at developing code for complicated models, for testing hypothesis etc, the code itself is not often of a suitable quality which would be required by an industry organisation. A professional company needs well documented code which is resilient to the underlying operating system or environment and should be easily extended by other individuals without the need of the original coder. In contrast, academic code can be personal to the individual and is often never reviewed, and while they may be sense checked throughout development, there may not be any rigorous [unit] testing.

For a good example of this, consider the review of the recent COVID modelling code from Imperial. John Carmack was one of the code reviewers and noted "Before the GitHub team started working on the code it was a single 15k line C file that had been worked on for a decade, and some of the functions looked like they were machine translated from Fortran. There are some tropes about academic code that have grains of truth, but it turned out that it fared a lot better going through the gauntlet of code analysis tools I hit it with than a lot of more modern code."[1]

This report details some of the coding best practice skills that the data industry desires from its employees and thus highlights some training that universities could provide to their students.

In many scientific domains it is becoming increasingly common for code products to be treated as first-class outputs of scientific research. Publications like the Journal of Open Source Software (JOSS) focus exclusively on providing a rigorous framework to publish code products, and integrations between established journals and preprint servers like arXiv and code repositories are becoming more common.

Despite this, publications in the forecasting and energy domain have been slow to adopt the focus on code growing in other domains. From an academic perspective, the effort required to turn *ad hoc* analysis code into something digestible by other people has little payoff. Although publishing code may potentially increase citations, researchers mentioned significant risk in exposing analysis code to wider scrutiny. Bugs might be found in the analysis code, researchers trying to run the code might raise support requests, and there are now two sources of truth for a publication: the paper and the code. However, these same arguments suggest that academic integrity could be improved by encouraging code to be shared since mistakes can lead to more robust trust in the science and experiments.

Beyond waiting for the cultural shift and change from the editorial level of journals, there are significant reasons why researchers, especially those still studying, should prioritize code development and release.

[1] https://twitter.com/id_aa_carmack/status/1254872368763277313

2.1. INDUSTRY SKILLS GAP

As alluded to in the main report (Haben, &, Hinton, 2022a), data science, analytics, and software engineering roles make up a significant portion of the career pathways for Masters and PhD level students that chose not to forge an academic career. According to the survey results from (UK Government, 2021), they showed that the largest gap found when hiring for data positions is that of programming (Table 11 withing (UK Government, 2021)). Out of the thirteen different data skills surveyed, less than half the students surveyed responded that they believed they will develop good programming skills in the future–the lowest of any data skill. For industry employers, to which coding skills are essential, this represents a challenge when finding strong candidates. Additionally, when students were surveyed about their current skills (as opposed to future skills they would develop), programming was also ranked as the lowest out of a longer list of 27 data-applicable skills.

For students aiming to pursue careers in data-focused areas such as data analytics, data science, or data engineering, spending time to develop coding skills would represent a valuable use of time.

2.2. CODE TRAINING WITHIN UNIVERSITIES

It is important to note that these skills gaps are not a failing of the students themselves. They are limited by what is provided within their educational institutions, who do not focus the resources in creating up-to-date coursework that accurately reflects the skills required of industry work. This gap is endemic to institutions not just in the UK, but around the world. Focusing our attention, however, to well-known universities in the UK, it is easy to understand how students graduating from data focused master programs lack these critical and much needed coding skills.

Students may graduate with conceptual knowledge of many advanced models and algorithms, but with little ability to practically implement those models in a rigorous context.

A non-exhaustive review of masters programs based on their course topics, it is apparent that students have little opportunity when guided by the provided coursework to gain proper proficiency in these critical skills. Of ten data science focused master’s programs from major universities in the UK[2] we found only two that had dedicated introductory course to programming and none of them had any dedicated intermediate or advanced courses.

The above list is not to say that students enrolled in those masters’ programs will have no exposure to programming, however there is a large difference between exposure to code within a course, and a dedicated syllabus designed to teach students the correct fundamentals of programming, design patterns, code structure, and how to use modern tools (or even not-so-modern tools like version control).

The message from these programs is clear: if students wish to address the skills gap that has been highlighted numerous times over the past decade, and have not done so in their undergraduate degree already, they will have to do it themselves.

[2] We considered Data Science and Machine learning masters courses from UCL, Imperial College, University of Edinburgh, King’s College London, University of Manchester, University of Sheffield, University of Bristol, University of Reading and University of Southampton

2.3. DESIRABLE CODING KNOWLEDGE AND SKILLS FOR STUDENTS ENTERING INDUSTRY

As part of our interviews we discussed several different desirable skills which industrial employers may want with respect to programming and coding which we detail below. In this section, we assume the coder is writing in Python for simplicity. Many, but not all, of these recommendations would be just as applicable to R or other programming languages. Additionally, these points are listed in rough order of benefit weighted against the time it takes to implement them.

So, for a student wishing to delve further into the software side of research to increase their prospects of landing an industry job:

1. **Make your code open source** on a version control platform like GitHub, GitLab, BitBucket, etc. For those wanting continued engagement with the code base, it would be useful to include a readme file with installation instructions, running instructions, and contribution guidelines if desired. A requirements file should detail the packages needed for running the provided code and adding a license file (such as the MIT license) will remove ambiguity to others about how they can use the code[3]. Additionally, these version control platforms often have useful features that can be enabled. For example, GitHub issues can be used to track feature requests and/or bugs in code, and GitHub discussions provide a more transparent way for users to discuss or request help with the code base outside of emailing contributors. The number of forks, stars, discussions, or issues might provide useful metrics for external engagement.
2. **Run a linter or formatter over your code**. Packages like Black will do this automatically and will not modify the execution of your code. This increases the readability of your code with minimal effort, just like having proper punctuation and grammar increases the readability of textual works.
3. **Document and comment your code.** In-code documentation conventions exist, such as googledoc, numpydoc, etc, and can be used as a basis for providing useful description of function signatures and complex pieces of code. Additionally, adding docstring to functions also encourages type hinting, which is highly valuable to add to code bases and is rapidly becoming an industry requirement for Python, and is simply part of the language for statically typed languages. These docstrings can be turned into online documentation and then hosted for free.
4. **Refactor code into functions and files.** There are many courses available at universities and online that detail code refactoring and the engineering principles that motivate them. From simply ensuring that a massive script is broken into smaller chunks, to performing intensive reorganisation of the code base. Refactoring with a goal to making the entry point to the code base easier to understand will greatly aid people wanting to utilise or further your own work. In conjunction with the prior point on code documentation, open-source libraries often provide great examples as to how to structure and document your code. While larger libraries like scipy and numpy may be overwhelming to students, smaller libraries may provide useful examples.
5. **Add code examples** in how to use your package, or how to call specific functions. Consider promoting these examples into unit tests if possible, using established testing frameworks

[3] There are some very useful tools such as http://ufal.github.io/public-license-selector/ which can help understand which licenses may be most suitable for your software or data.

like pytest. Code testing is a common area completely lacking in the skill set of academic students, and learning how it works is a great way to stand out.

Those that wish to delve even deeper into the software side of their data skills can also look at setting up continuous integration (via something like GitHub Actions), publishing their code (to PyPI for Python, CRAN for R), and publishing online documentation (for example with "Read the Docs").

Many of the points above are well known gaps with introductory programmers, and there are existing resources which are designed to address them. We encourage students to visit *The Turing Way* (The Turing Way, 2022), which provides guides and resources on reproducible research, code collaboration, project design, and more.

2.4. TOOLS TO IMPROVE CODING SKILLS AND AID IN REPRODUCIBLE RESEARCH

Many students may wish to remain in academic focused career tracks, but will still find utility in improving coding skills and adopting modern tooling around their projects. In this section, we therefore lay out a (non-exhaustive) selection of tools that are regularly utilised in other scientific domains, but have yet to flow into the sphere of energy research.

2.4.1. GUIDES AND REPRODUCIBLE RESEARCH PIPELINES

- **The Turing Way**: Mentioned above, this is a great one-stop-shop for setting up projects with a focus on standards and reproducibility. Its covers a range of topics including:
    - Code testing, quality, documentation and standards
    - Data archiving, versioning
    - Reproducing research using tools like binder
    - Continuous integration
- **Datalad**: For those wanting a pipeline to keep track of data, code execution, and paper compilation, DataLad may be a solution. It requires significant input to get set up, but for those with a strong focus on research reproducibility will appreciate it.
- **Cookiecutter Data Science**: A template repository focused on data analytics. For those starting a new data-centric project, this would be a good starting point. As it is only a template, it doesn't require the active integration that Datalad would need.

2.4.2. REPRODUCIBILITY AND ACCESSIBILITY

There has been an ever-increasing focus on making data and code more accessible. This not only help to increase the models which are out there, but it also strengthens the science as now models can be tested properly and reproducible. For individual tools which may be useful in increasing reproducibility:

- **Github/Gitlab/Bitbucket:** A cornerstone of collaborative software development, Git is a software for tracking changes in documents and is used extensively by industry and academics.
- **Binder/Google Colab**: The next step in collaborative coding, unlike Git in which code must be "pulled" to a local computer, there is emergence of more browser-based coding environments. Binder also supports the easy integration of Git pages, and allows researchers to have a one-click link which reproduces any or all aspects of their analysis.
- **Zenodo**: Archiving service, common place to upload large data.
- **SoftwareHeritage**: Another code archiving service
- **Licensing Tools**: such as http://ufal.github.io/public-license-selector/ which may help identify which license may be suitable for releasing data or software

2.4.3. OPEN-SOURCE PROJECT TOOLS

There is an increasing number of open-source tools being made available. Any academics hoping to share code with their papers should make it clear by including within the manuscript (preferably on the first page!). Other resources for open-source tools were also identified in our interviews:

- **ProtonTypes**: A community for the acceleration of open projects, business models, technologies and platforms.
- **Open Sustainable Technology**: A list of curated open technology projects in the domain of sustainability.

3. SUMMARY & RECOMMENDATIONS

Although university graduates and researchers are excellent at many technical skills, a large skill gap exists in the area of programming, software development, and coding practises. Additionally, the tooling, frameworks and technology which surround this area have been highlighted as areas requiring improvement for those wanting to contribute or transition into industry roles. Some training and practice would be extremely beneficial and make students and academics much more desirable to future employees. However, often these skills are not strongly developed within universities and hence it may not be possible to teach purely without support from industry. Some of the key recommendations from our research include:

- **Collaborative learning:** Close collaborations are beneficial for key coding skills. They can be a way for both sides to keep up to date with the latest knowledge, skills, and needs. In particular, industry partners will be able to inform academia about the latest technologies and software practices they are using.
- **Industry driven coding training:** Industry should provide or at the very least support key training in the area of coding practice and technologies. Since software development and technology stacks rapidly change, outsourcing this teaching may be a much more efficient way for universities to keep up to date with the latest training. At the very least universities should utilise industry feedback to help develop such courses. Guest lectures by industry professionals are another option. Some of these key skills are listed above in Section 2.3.
- **Code publishing refocus within academia:** An increased emphases from academics on producing higher-quality code products as first-class outputs from research would provide students with portfolio pieces, increase their coding skills, drive better engagement with industry partners, and provide metrics that could be used to demonstrate research impact. This may require a culture change in academia where publishing of code is properly acknowledged as a key academic output and increased in value relative to the traditional manuscript publishing.

4. BIBLIOGRAPHY

The Turing Way, (2022). Retrieved from The Turing Way: https://the-turing-way.netlify.app/welcome.html

DuBois, J. (2020). *The Data Scientist Shortage in 2020*. Retrieved February 8, 2022, from https://quanthub.com/data-scientist-shortage-2020/

Haben, S., & Hinton, S. (2022a). Data Science: From Academia to Industry - Making Impact in the Energy Sector. Energy Systems Catapult.

Haben, S., & Hinton S. (2022b). Academic and Industrial Collaborations, Data Science: From Academia to Industry – Supplementary Report. Energy Systems Catapult

Haben, S., & Hinton S. (2022c). Accessible and Reproducible Research, Data Science: From Academia to Industry – Supplementary Report. Energy Systems Catapult

Haben, S., & Hinton S. (2022d). Industrial Support for Academics, Data Science: From Academia to Industry – Supplementary Report. Energy Systems Catapult

Sandys, L., Dobson, R., Verma, J., Johnston, G., Roberts, D., Leland, B., Haben, S., Ainsworth, E., Guinta, F., & Pearson, S. (2022). *Delivering a Digitalised Energy System: Energy Digitalisation Taskforce Report*. Energy Systems Catapult.

UK Government. (2021). *Quantifying the UK Data Skills Gap - Full report*. Department for Digital, Culture, Media & Sport. Retrieved from https://www.gov.uk/government/publications/quantifying-the-uk-data-skills-gap/quantifying-the-uk-data-skills-gap-full-report

OUR MISSION

TO UNLEASH INNOVATION AND OPEN NEW MARKETS TO CAPTURE THE CLEAN GROWTH OPPORTUNITY.

ENERGY SYSTEMS CATAPULT
7TH FLOOR, CANNON HOUSE,
18 PRIORY QUEENSWAY,
BIRMINGHAM, B4 6BS.

ES.CATAPULT.ORG.UK
@ENERGYSYSCAT