Effective Data Science

Zak Varty

2022-09-22

Contents

Preface

These notes are intended for students on the course **MATH70076: Data Science** in the academic year 2022/23.

As the course is schedled to take place over five weeks, the suggested schedule is

- 1st week: Chapters 1 and 2
- 2nd week: Chapter 3
- 3rd week: Chapter 4
- 4th week: Chapters 5
- 5th week: Chapter 6

A pdf version of these notes may be downloaded here.

Acknowledgements

These notes were created by Dr Zak Varty based on a lecture series at Imperial College London that was developed by Dr Purvasha Chakravarti and others.

Chapter 1

Introduction

1.1 Module Description

Model building and evaluation are are necessary but not sufficient skills for the effective practice of data science. In this module you will develop the technical and personal skills that are required to work successfully as a data scientist within an organisation.

During this module you will critically explore how to:

- effectively scope and manage a data science project;
- efficiently acquire, manipulate, and present data;
- interpret and explain your work for a variety of stakeholders;
- ensure that your work can be put into production;
- assess the ethical implications of your work as a data scientist.

This interdisciplinary course will draw from fields including statistics, computing, management science and data ethics. Each topic will be investigated through a selection of lecture videos, conference presentations and academic papers, hands-on lab exercises, and readings on industry best-practices from recognised professional bodies.

This course will be assessed entirely by coursework, reflecting the practical and pragmatic nature of the course material.

1.2 Allocation of Study Hours

Lectures: 10 Hours (2 hours per week)

Group Teaching: 5 Hours (1 hour per week)

Lab / Practical: 5 hours (1 hour per week)

Independent Study: 105 hours (17 hours per week + 20 hours coursework)

1.2.1 Learning outcomes

On successful completion of this module students should be able to:

1. Independently scope and manage a data science project;
2. Source data from the internet through web scraping and APIs;
3. Clean, explore and visualise data, justifying and documenting the decisions made;
4. Evaluate the need for (and implement) approaches that are explainable, reproducible and scalable;
5. Appraise the ethical implications of a data science projects, particularly the risks of compromising privacy or fairness and the potential to cause harm.

1.2.2 Module Content

This module will cover:

- effective management of a data science project;
- open and reproducible work flows;
- sourcing and preparing data for analysis;
- exploratory and expository data visualisation;
- minimal requirements for models to go into production;
- ethical implications of modern data science.

An unnumbered section

Chapters and sections are numbered by default. To un-number a heading, add a `{.unnumbered}` or the shorter `{-}` at the end of the heading, like in this section.

Chapter 2

Data Science Workflows

2.1 Introduction

As a data scientist you will never work alone. Within a single project a data scientist is likely that you will interact with a range of other people, including but not limited to: one or more project managers, stakeholders and subject matter experts. These experts might for example be trained as sociologists, chemists or civil servants, depending on the type of work that you are doing. To get your project put into use and working at scale you will likely have to collaborate with data engineers. You will also work closely with other data scientists. For smaller projects this might be to act as reviewers for one another's work, while for larger projects working collaboratively allows you to tackle larger challenges that would not be feasible alone due to constraints on time or the skill set of any one person.

Familiarity with the skills, processes and practices that make collaborative working is instrumental to being a successful as a data scientist.

Even if you work in a small organisation where you are the sole data scientist, adopting a way of working that welcomes collaboration will pay dividends over time. This is because when you inevitably return to the project that you are working on in several weeks, months or years you will have forgotten almost all of what you did. You will also have forgotten why you made the decisions that you did and what the other options are. This is exactly

like working with a current colleague who has poor working practices. Nobody wants to be that colleague to someone else, never mind to themselves. By treating your future self as a current collaborator that you want to get along well with, you can be kind colleague to your future self.

The aim of this chapter is to provide you with a structure on how you organise and perform your work, so that you can be a good collaborator to current colleges and your future self. This is going to require a bit more effort upfront, but the benefits will compound over time. You will get more done by wasting less time staring quizzically at messy folders of indecipherable code. You will also gain a reputation of someone who is good to work with. This promotes better professional relationships and greater levels of trust, which can in turn lead to working on more exciting and impactful projects.

The structures and workflows that I recommend here (and throughout the module) are focused around a workflow that predominantly uses R, markdown and LaTeX.

Similar techniques, code and software are available to achieve the same things when working with Python, C, Quarto and a range of other programming and mark-up languages. Once you understand what good habits are and build them in one language, transferring these skills is largely a matter of learning some new vocabulary or slightly different syntax.

2.2 Organising your work

2.2.1 One directory per project

The golden rule when organising your work is that all the work for one project should live in a single directory (folder) on your computer. This is the digital equivalent of having one ringbinder folder for each of your MSc modules.

Doing this sounds easy, but in practice it is not. The first issue is that it requires a predetermined scope of what is (and what is not) covered by a given project. The second issue is that the second law of thermodynamics might well have been written about project management. It takes continual, external effort to prevent the contents of that one folder

from becoming chaotic and disordered.

That being said, having a single directory has several benefits that justify this additional work. Having a projects be self-contained allows us to create projects that are:

- Portable
- Version controlled
- Reproducible
- Development environment friendly.

A project is **portable** if it can easily be moved without breaking. This might be a small move, relocating the directory on your own computer, a moderate shift to another machine (say if yours dies just before a big deadline), or a large shift to another user on a different operating system.

- All work for one project goes into a single directory
  - Portability (filepaths, backslashes, setwd())
  - Version control
  - IDEs play nicely (RStudio projects)
  - reproducibility
- Organising within that directory: Every project different, will develop over course of project. Want to give a sensible starting point but often a company will have a 'house style'. If so IGNORE ME (unless the house style is rubbish, in which case only ignore me while you lobby for that to be changed.)
  - README.md
  - data
    - raw (anything you do not make for yourself)
    - refined (everything that you make for yourself)
  - src (functions)
  - tests (checks for each of your functions)
  - analyses (scripts, models)
  - outputs (results of all your hard work)
    - analysis-1

            - data
            - tables
            - figures
        * analysis-2
            - data
            - tables
            - figures
    - reports (write-up)
        * analysis-1
        * analysis-2
    - *bonus:* Makefiles & meta-programming

2.3 File types and development environments

- What type of files do you use and where do you code?
    - plaintext vs proprietry (csv / xlsx, markdown / googledoc)
    - command line vs notebook vs scripting (no file, .Rmd, .R)
    - Open source languages vs closed source

2.4 Naming things

- Naming things
    - Jenny Bryan slide summary (https://speakerdeck.com/jennybc/how-to-name-files)
    - We would like file names to be:
        * Machine Readable
        * Human Readable
        * Order friendly
    - Machine readable:
        * regex and globbing friendly: avoid spaces, punctuation, accents, cases

        * easy to compute on: deliberate use of delimiters (spaces that aren't spaces)
            · hyphens separate words, underscores separate metadata
        * useful when: searching for files later, narrow file list based on names, extract information from file names, new to regex (or not a sadist)
    - Human Readable:
        * Name contains information on content. (untitled31.R, finalreportV8.docx, temp.txt)
        * connects to a concept of a slug from URLs
        * Which set of filenames do you want at 3am before a deadline?
    - Default order friendly
        * put something numeric first
        * use ISO 8601 standard for dates: YYYY-MM-DD
        * left pad with zeros to achieve chronological or logical order within each directory
    - Summary:
        * Machine readable, human readable, default order friendly
        * Brushing teeth analogy: tedious until you get in the habit. Huge long-term rewards

2.5 Code

- Code
    - If you do the same thing twice write a function
    - If your write a function, document it
    - If you write a function, test it
    - If might ever want to use your function again, add it to a package
    - naming things revisited:
        * functions=verbs,
        * objects=nouns,
        * readable code,

    - CamelCase snakecase pointless.points
    - tidyverse and google style guides for R
  - all filepaths relative to the root directory (the top level of your project)
    - advanced: here::here

2.6 Bigger Picture: Project management

- Project management
  - Defining outcomes
  - scoping projects, (rembmember how we said all code for a silg eproject should live in a single directory?)
  - continuous development, agile + jira ?
  - Linking to github (extension)

2.7 Tasks:

- go to github and find 3 different data science projects, explore how they organise their work.
- create your own projects for this course and for the assignments.
- *bonus:* put these on Github (make sure the assignments are private repos!)

2.8 Reading

- Good enough practices in scientific computing
- Bayesian workflows: Micheal Battencourt
- https://www.atlassian.com/agile/project-management
- R4DS project workflow
- here::here
- R packages

- happy git with R

2.9 Live Session Plan

Discussion point in live session:

- Did you make the assignment projects as subdirectories or as their stand alone projects? Why?
- What were some terms that you had not met before during the readings?

Activity in live session:

- making a minimal R package for this course

Chapter 3

Aquiring and Sharing Data

You can add parts to organize one or more book chapters together. Parts can be inserted at the top of an .Rmd file, before the first-level chapter heading in that same file.

Add a numbered part: `# (PART) Act one {-}` (followed by `# A chapter`)

Add an unnumbered part: `# (PART\*) Act one {-}` (followed by `# A chapter`)

Add an appendix as a special kind of un-numbered part: `# (APPENDIX) Other stuff {-}` (followed by `# A chapter`). Chapters in an appendix are pre-pended with letters instead of numbers.

Data can be difficult to acquire and gnarly when you get it. - Structure of a webpage. Web Scraping as a source of data - APIs as a source of data, data files beyond csv. - Data bases and SQL

- Data will not always be in a nicely formatted csv
- Beginner: reading from clipboard and googlesheets
- Intermediate: reading messy csvs and making your workflow robust to new versions
- Advanced:
  - Webscraping and APIs as data sources, data files beyond CSV

- data files beyond csv, benefits and drawbacks: JSON, xml, parquet, .Rdata, .pkl
- Data bases as data sources: Basic SQL verbs
- Learning SQL theory on a small scale: dplyr verbs

Chapter 4

Cleaning, Exploring and Visualising

4.1 Footnotes

Footnotes are put inside the square brackets after a caret `^[]`. Like this one [1].

4.2 Citations

Reference items in your bibliography file(s) using `@key`.

For example, we are using the **bookdown** package (Xie, 2022) (check out the last code chunk in index.Rmd to see how this citation key was added) in this sample book, which was built on top of R Markdown and **knitr** (Xie, 2015) (this citation was added manually in an external file book.bib). Note that the `.bib` files need to be listed in the index.Rmd with the YAML `bibliography` key.

The `bs4_book` theme makes footnotes appear inline when you click on them. In this example book, we added `csl: chicago-fullnote-bibliography.csl` to the `index.Rmd` YAML, and include the `.csl` file. To download a new style, we recommend: https://www.zotero.org/styles/

The RStudio Visual Markdown Editor can also make it easier to insert citations: https:

[1] This is a footnote.

//rstudio.github.io/visual-markdown-editing/#/citations

Chapter 5

Getting Your Work into Production

5.1 Equations

Here is an equation.

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k} \tag{5.1}$$

You may refer to using `\@ref(eq:binom)`, like see Equation (5.1).

5.2 Theorems and proofs

Labeled theorems can be referenced in text using `\@ref(thm:tri)`, for example, check out this smart theorem 5.1.

Theorem 5.1. *For a right triangle, if c denotes the* length *of the hypotenuse and a and b denote the lengths of the* ***other*** *two sides, we have*

$$a^2 + b^2 = c^2$$

Read more here https://bookdown.org/yihui/bookdown/markdown-extensions-by-

bookdown.html.

5.3 Callout blocks

The `bs4_book` theme also includes special callout blocks, like this `.rmdnote`.

You can use **markdown** inside a block.

```
head(beaver1, n = 5)
#>   day time  temp activ
#> 1 346  840 36.33     0
#> 2 346  850 36.34     0
#> 3 346  900 36.35     0
#> 4 346  910 36.42     0
#> 5 346  920 36.55     0
```

It is up to the user to define the appearance of these blocks for LaTeX output.

You may also use: `.rmdcaution`, `.rmdimportant`, `.rmdtip`, or `.rmdwarning` as the block name.

The R Markdown Cookbook provides more help on how to use custom blocks to design your own callouts: https://bookdown.org/yihui/rmarkdown-cookbook/custom-blocks.html

Chapter 6

Wider Context and Ethics

6.1 Publishing

HTML books can be published online, see: https://bookdown.org/yihui/bookdown/publishing.html

6.2 404 pages

By default, users will be directed to a 404 page if they try to access a webpage that cannot be found. If you'd like to customize your 404 page instead of using the default, you may add either a `_404.Rmd` or `_404.md` file to your project root and use code and/or Markdown syntax.

6.3 Metadata for sharing

Bookdown HTML books will provide HTML metadata for social sharing on platforms like Twitter, Facebook, and LinkedIn, using information you provide in the `index.Rmd` YAML. To setup, set the `url` for your book and the path to your `cover-image` file. Your book's `title` and `description` are also used.

This `bs4_book` provides enhanced metadata for social sharing, so that each chapter shared

will have a unique description, auto-generated based on the content.

Specify your book's source repository on GitHub as the `repo` in the `_output.yml` file, which allows users to view each chapter's source file or suggest an edit. Read more about the features of this output format here:

https://pkgs.rstudio.com/bookdown/reference/bs4_book.html

Or use:

```
?bookdown::bs4_book
```

Bibliography

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2022). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.26.