Data Technologies

Computational tools for working with data

Martin Frigaard

2023-06-07

Table of contents

Welcome!

This book describes the essential tools and techniques for working with data, especially those 'behind-the-scenes' tasks that will take considerable time and effort but don't receive much attention.

Code

This section introduces programming languages, the command-line interface, regular expressions, and computational notebooks.

Object-oriented programming

In this section covers the object-oriented programming (OOP) paradigm, common features, and why OOP languages are useful for working with data. Three languages (R, Python, and JavaScript) are introduced as examples to compare and contrast.

Functions

Data

Git

Preface

This book is focused on computational tools for working with data. Data science is a rapidly evolving field, so in many ways, this text is an attempt to capture whatever 'best practices' can be codified or generalized.

The topics include coding languages, computational documents, data storage and file formats, version control systems, and general computer science topics.

This text emphasizes using tools that require typing code on a keyboard (rather than using a mouse to point and click).

This book was written in Quarto (learn more about Quarto books).

Introduction

> *This is a book created from markdown and executable code. See Knuth (1984) for additional discussion of literate programming.*

I've left the boilerplate text above because everyone should know about Donald Knuth and his contributions to literate programming that lead to the book you're currently reading.

How the sausage is made

I've included the steps for publishing this book *inside* the book because it's serves as an excellent example of the combined topics contained in this text can be used. This book is written in Quarto which is an '*open-source scientific and technical publishing system.*' All of the tools in this text are open-source, which means the source code (i.e. files) to create it are '*made freely available for possible modification and redistribution.*'

The code files used to create this book are displayed in the folder tree below:

```
.
+-- _book
+-- _freeze
+-- _inst.R
+-- _quarto.yml
+-- code.qmd
+-- command-line.qmd
+-- cover.png
+-- data-tech.Rproj
+-- files.qmd
+-- fun-arguments.qmd
+-- fun-environments.qmd
+-- fun-names.qmd
+-- fun-structure.qmd
+-- index.html
+-- index.qmd
+-- intro.html
+-- intro.qmd
```

```
+-- intro.rmarkdown
+-- intro_files
+-- notebooks.qmd
+-- oop-basics.qmd
+-- oop-in-python.qmd
+-- oop-in-r.qmd
+-- preface.html
+-- preface.qmd
+-- references.bib
+-- references.qmd
+-- regex.qmd
+-- renv
+-- renv.lock
\-- site_libs
```

These files are stored in a GitHub repository. GitHub is a platform for hosting open source projects that use Git, the world's most popular distributed version control system.

Using Git

Basic knowledge of Git has become somewhat necessary when you decide to enter the data technology ecosystem (or other open-source projects). I won't be diving into the Git workflow here, but I *will* cover the commands I used to store and publish this book.

The following commands are entered at the command line (Terminal if you're using macOS).

1. I created a repository on GitHub.com like this one: mjfrigaard/data-tech
2. Add (`-A`) and commit (`commit -m`) the files in the book folder:

```
$ git add -A
$ git commit -m "first commit"
```

3. `push` local files to GitHub repo

```
$ git remote add origin git@github.com:mjfrigaard/data-tech.git
$ git branch -M main
$ git push -u origin main
```

4. Create (checkout) an empty `gh-pages` branch

```
$ git checkout --orphan gh-pages
Switched to a new branch 'gh-pages'
$ git reset --hard
$ git commit --allow-empty -m "Initialising gh-pages branch"
Initialising gh-pages branch
```

5. `push` the book files to the `gh-pages` branch

```
$ git push origin gh-pages
Enumerating objects: 2, done.
Counting objects: 100% (2/2), done.
Writing objects: 100% (2/2), 176 bytes | 176.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'gh-pages' on GitHub by visiting:
remote:      https://github.com/mjfrigaard/data-tech/pull/new/gh-pages
remote:
To github.com:mjfrigaard/data-tech.git
 * [new branch]      gh-pages -> gh-pages
```

5. Switch (`checkout`) back to `main` branch

```
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

6. `publish` to `gh-pages` branch

```
$ quarto publish gh-pages
? Update site at https://mjfrigaard.github.io/data-tech/? (Y/n)
```

```
? Update site at https://mjfrigaard.github.io/data-tech/? (Y/n) Yes
```

Part I

Code

Writing code

Language is a technology that gives humans the ability to express ideas with precision. A programming language–what is typically meant when someone refers to 'code'–is the arrangement of words, numbers, and/or symbols we use to communicate our intentions to a computer.

> *"The code that translates a web of conceptual relations in our heads into an early-to-late order in our mouths, or into a left-to-right order on the page, is called* ***syntax****…Different languages have different* ***grammars****, but they all convey conceptual relationships by modifying and arranging words."* - *The Sense of Style, Steven Pinker (2014)*

Natural languages are used for a wide range of human communication, from slang terms in casual conversation to complex expressions of abstract ideas. The pragmatic quality of natural languages requires them to have high levels of redundancy. Redundancy helps us understand each other even when the rules of syntax are violated (i.e., when the same word can have different meanings in different contexts or when a sentence can have multiple valid interpretations).

Programming languages require higher levels of precision than natural languages. Poorly written sentences can be painful to read or difficult to comprehend, but they don't ruin an entire book. Computers are uncompromising when it comes to syntax–all code must perfectly conform to the rules for syntax, grammar, and semantics before the code will work. Writing code is like writing a book where a single typo makes the entire text illegible.

> **i** Terms to know
>
> **Grammar**: the rules for how a language uses symbols (words, numbers, symbols, punctuation, etc.).
> **Syntax**: how to arrange words and phrases into sentences to create meaning.
> **Semantics**: the meaning of words and sentences.

Machine vs. human-readable code

Machine-readable code consists of binary instructions (a series of 1s and 0s) that can be processed by computers directly. Machine code is fast and efficient to execute but is difficult for humans to read and write.

Assembly languages are low-level languages that are a step above machine-readable code. These languages are *somewhat* human-readable but are still much more difficult for humans to read than high-level languages.

Languages like R, Python, JavaScript, and many others are high-level languages designed to be human-readable with syntax and grammar that imitate natural languages. All higher-level code must be translated into machine-readable code before the computer can run it.

Markup languages

A markup language is a text-encoding system for annotating a document with a set of symbols that are syntactically distinguishable from its text. Markup languages are not programming languages per se but refer to a set of characters or symbols you can insert at certain places in a text file to indicate how the style of the output document should look when it's printed or displayed.

HTML

`HTML`, or Hypertext Markup Language, is the standard language used to structure content on the web and is a fundamental technology used in all websites.

Syntax: `HTML` tags denote different types of content. Tags are enclosed in angle brackets (`</>`) and come in pairs to create `HTML` elements. The first tag in the pair is the start tag (`<`), and the second tag is the end tag, which is the same as the start tag, but also includes a forward slash (`/>`).

```
<!DOCTYPE html>
<html>
  <head>
    <title>Title for page</title>
  </head>
  <body>
    <h1>This is a level 1 heading</h1>
    <p>This is the first paragraph.</p>
    <p>This is the second paragraph.</p>
  </body>
</html>
```

The HTML document above has a basic structure that includes `<html>`, `<head>` and `<body>` tags:

- The document above begins with the `<!DOCTYPE html>`declaration to tells your browser application is an HTML5 document
- The `<html>`/`</html>` tags enclose the entire document
- Inside `<html>`/`</html>`, the `<head>`/`</head>` elements contain meta-information about the document (`<title>`/`</title>`), which is displayed in the browser's title bar or tab.
- The `<body>`/`</body>`element contains the actual content of the webpage (text, add images, create links, forms, embed audio, video, and other media, etc.)

> **i** Terms to know
>
> **HTML5** is refers to the '*fifth and final major HTML version that is a World Wide Web Consortium recommendation*.' Wikipedia

Markdown

Markdown is a 'lightweight' markup language for creating formatted text using a text editor. Markdown is primarily used to convert plain text into `HTML`, but it's also used to format `README` files, and for writing messages in online discussion forums like Stack Overflow.

Syntax:

- Headers from levels 1 to 6 can be created using a number of # symbols corresponding to the header level:
  - `# Header 1` for a first-level header
  - `## Header 2` for a second-level header, and so on.
- Asterisks (`*`) or underscores (`_`) are used for emphasis:
  - `*italic*` or `_italic_` for italic
  - `**bold**` or `__bold__`for bold
  - You can also combine them to create bold italic text like `***bold italic***` or `___bold italic___`
- Unordered lists are creatd using asterisks, plus signs, or hyphens interchangeably (`*`, `+`, `-`)
- Ordered lists are created using numbers followed by periods

0.0.0.0.1 * *Markdown syntax*

```
- Item A
- Item B
- Item C
```

0.0.0.0.2 * *Rendered HTML*

- Item A
- Item B
- Item C

Markdown syntax

```
1. Item 1
2. Item 2

3. Item 3
```

Rendered HTML

1. Item 1
2. Item 2
3. Item 3

YAML

`YAML` is a human-friendly, easily readable data serialization standard that can be used in conjunction with all programming languages.

R

Python

JavaScript

Code files

Open-source code is written in plain text files.

The command line

"*In the beginning, there was the command line…*" - essay by Neal Stephenson

Regular expressions

Computational notebooks

Jupyter Notebooks

R Markdown

Quarto

Part II

Object-oriented programming

Basics

Object-oriented programming is...

R

R is a ‘functional, object-oriented programming language.’

Python

Part III

Functions

Names

Naming your functions should...

Arguments

Function arguments can be

Structure

Function structure refers to...

Environments

Inside your function...

Knuth, Donald E. 1984. "Literate Programming." *Comput. J.* 27 (2): 97–111. https://doi.org/10.1093/comjnl/27.2.97.