Research Software Engineering with Python

- Damien Irving
- Kate Hertweck
- Luke Johnston
- Joel Ostblom
- Charlotte Wickham
- Greg Wilson

1. Aims

Software is now as essential to research as telescopes, test tubes, and reference libraries, which means that researchers now need to know how to build, check, use, and share programs. However, most introductions to programming focus on developing commercial applications, not on exploring problems whose answers aren't yet known. Our goal is show readers how to do that, both on their own and as part of a team.

We believe every researcher should know how to write short programs that analyze data in a reproducible way, and how to use version control to keep track of what they have done. But just as some astronomers spend their careers designing telescopes, some researchers focus on building the software that makes research possible. People who do this are called *research software engineers*, and the aim of this book is to get you ready for this role, i.e., to help you go from writing code for yourself to creating tools to help your entire field advance.

2. Synopsis

This book introduces the core skills researchers need to develop robust software that others can use, to share their work with others, and to be productive as part of a research software team. We assume readers have some basic programming knowledge, and build on that to cover:

- using the Unix shell to manage work and make it repeatable
- effective use of version control
- building reusable software tools
- automating workflows
- configuring software
- testing and error handling
- creating productive, inclusive teams
- documenting, packaging, and releasing software for use by the wider research community

Our book is a ready-to-go university semester course. All of this material has been used and refined in workshops, some of it by multiple instructors over many years. Please see the end of this proposal for a detailed table of contents.

3. Target Market

At the beginning of the book, we introduce three personas who characterize our audience:

Amira Khan completed a master's in library science five years ago and has since worked for a small aid organization. She did some statistics during her degree, and has learned some R and Python by doing data science courses online, but has no formal training in programming. Amira would like to tidy up the scripts, data sets, and reports she has created in order to share them with her colleagues. These lessons will show her how to do this and what "done" looks like.

Jun Hsu completed a 4-month data science bootcamp last year after doing a PhD in Geology and now works for a company that does forensic audits. He uses a variety of machine learning and visualization packages, and would now like to turn some of his own work into an open source project. This book will show him how such a project should be organized and how to encourage people to contribute to it.

Sami Virtanen became a competent programmer during a bachelor's degree in applied math and was then hired by the university's research computing center. The kinds of applications they are being asked to support have shifted from fluid dynamics to data analysis; this guide will teach them how to build and run data pipelines so that they can pass those skills on to their users.

4. Competing Titles

The Turing Way (https://the-turing-way.netlify.app/), produced by the Turing Institute in the UK, is the only up-to-date resource we know of with similar breadth. Other books cover only one topic in more depth than our audience needs (e.g., Ray & Ray's *Unix and Linux: Visual QuickStart Guide*), are aimed at commercial software developers (any number of books on testing, coding style, and dev ops), or go into data science rather than telling readers how to build software (e.g., Zhang's upcoming *A Tour of Data Science*).

Our book has broader and deeper coverage, and includes exercises with solutions. Our focus on programming best practices (e.g. principles like writing modular, reusable, testable code). Others teach basic Python and R syntax and how to complete isolated tasks with particular libraries, but today, people can pick up that information from a Google search. Our content is the bigger picture you can't glean from online code snippets or Stack Overflow answers.

We go through the process of building a data science workflow/package from scratch. Most other texts simply present information (like a reference book): we tackle a real data science problem from beginning to end, weaving in relevant information as we go.

5. Format and Timeline

Our manuscript is approximately 470 pages long, with 40 diagrams and other graphics and 150 code listing.

6. Other Information

- Dr. Damien B. Irving: post-doctoral researcher in climate science at the University of New South Wales. https://damienirving.github.io/
- Dr. Kate L. Hertweck: bioinformatics training manager at the Fred Hutchinson Cancer Research Center. http://katehertweck.com/
- Dr. Luke Johnston: post-doctoral researcher in bioinformatics at Aarhus University. http://lukewjohnston.com/
- Joel Ostblom: graduate student in biomedical engineering at the University of Toronto. https://joelostblom.com/
- Dr. Charlotte Wickham: a data scientist and instructor at Oregon State University. https://www.cwick.co.nz/
- Dr. Gregory V. Wilson: co-founder of Software Carpentry, now part of the education group at RStudio. http://third-bit.com/

Table of Contents

The current table of contents is given below. Please note that we are currently planning to divide the material in Chapter 15 (Publishing) among the other chapters so that the book will build toward creating and distributing a reusable research software package. Please also note that we have started work on a companion volume that uses R instead of Python for examples. If this proposal is accepted, we expect to be able to complete the second book some time in 2021.

```
1) Introduction
1.1) The Big Picture
1.2) Audience
1.3) Syllabus
1.4) Acknowledgments
```

2) The Basics of the Unix Shell
2.1) Exploring Files and Directories
2.2) Moving Around
2.3) Creating New Files and Directories
2.4) Moving Files and Directories
2.5) Copy Files and Directories
2.6) Deleting Files and Directories
2.7) Wildcards
2.8) Reading the Manual
2.9) Combining Commands
2.10) How Pipes Work
2.11) Repeating Commands on Many Files
2.12) Variable Names
2.13) Redoing Things
2.14) Creating New Filenames Automatically
2.15) Summary
2.16) Exercises
2.17) Key Points
3) Going Further with the Unix Shell
3.1) Creating New Commands
3.2) Making Scripts More Versatile
3.3) Turning Interactive Work into a Script
3.4) Finding Things in Files
3.5) Finding Files
3.6) Configuring the Shell
3.7) Summary
3.8) Exercises
3.9) Key Points
4) Command Line Programs in Python
4.1) Programs and Modules
4.2) Handling Command-Line Options
4.3) Documentation
4.4) Counting Words
4.5) Pipelining
4.6) Positional and Optional Arguments
4.7) Collating Results
4.8) Writing Our Own Modules
4.9) Plotting
4.10) Summary
4.11) Exercises
4.12) Key Points
5) Git at the Command Line
5.1) Setting Up
5.2) Creating a New Repository
5.3) Adding Existing Work
5.4) Describing Commits

5.5) Saving and Tracking Changes
5.6) Synchronizing with Other Repositories
5.7) Exploring History
5.8) Restoring Old Versions of Files
5.9) Ignoring Files
5.10) Summary
5.11) Exercises
5.12) Key Points
6) Advanced Git
6.1) What's a Branch?
6.2) Creating a Branch
6.3) What Curve Should We Fit?
6.4) Verifying Zipf's Law
6.5) Merging
6.6) Handling Conflicts
6.7) A Branch-Based Workflow
6.8) Using Other People's Work
6.9) Pull Requests
6.10) Handling Conflicts in Pull Requests
6.11) Summary
6.12) Exercises
6.13) Key Points
7) Automating Analyses
7.1) Updating a Single File
7.2) Managing Multiple Files
7.3) Updating Files When Programs Change
7.4) Reducing Repetition in a Makefile
7.5) Automatic Variables
7.6) Generic Rules
7.7) Defining Sets of Files
7.8) Documenting a Makefile?
7.9) Automating Entire Analyses
7.10) Summary
7.11) Exercises
7.12) Key Points
8) Program Configuration
8.1) Configuration File Formats
8.2) Matplotlib Configuration
8.3) The Global Configuration File
8.4) The User Configuration File
8.5) Adding Command-Line Options
8.6) A Job Control File
8.7) Summary
8.8) Exercises
8.9) Key Points
9) Error Handling

9.1) Exceptions
9.2) Kinds of Errors
9.3) Writing Useful Error Messages
9.4) Reporting Errors
9.5) Summary
9.6) Exercises
9.7) Key Points
10) Working in Teams
10.1) Include Everyone
10.2) Establish a Code of Conduct
10.3) Include a License
10.4) Planning
10.5) Bug Reports
10.6) Labeling Issues
10.7) Prioritizing
10.8) Meetings
10.9) Making Decisions
10.10) Handling Conflict
10.11) Summary
10.12) Exercises
10.13) Key Points
11) Code Style, Review, and Refactoring
11.1) Python Style
11.2) Order
11.3) Checking Style
11.4) Refactoring
11.5) Code Reviews
11.6) Python Features
11.7) Summary
11.8) Exercises
11.9) Key Points
12) Project Structure
12.1) What is a Project?
12.2) Standard Information
12.3) Organizing Project Content
12.4) Using Compiled Programs
12.5) Documenting Software
12.6) What to Document
12.7) Creating a FAQ
12.8) Data versus Code
12.9) Managing External Data
12.10) Summary
12.11) Exercises
12.12) Key Points
13) Testing
13.1) Assertions

13.2) Unit Testing
13.3) Testing Frameworks
13.4) Testing Floating-Point Values
13.5) Testing Error Handling
13.6) Integration Testing
13.7) Regression Testing
13.8) Test Coverage
13.9) Continuous Integration
13.10) When to Write Tests
13.11) Summary
13.12) Exercises
13.13) Key Points
14) Python Packaging
14.1) Creating a Python Package
14.2) Virtual Environments
14.3) Installing a Development Package
14.4) What Installation Does
14.5) Distributing Packages
14.6) Documenting Packages
14.7) Hosting Documentation Online
14.8) Summary
14.9) Exercises
14.10) Key Points
15) Publishing
15.1) Identifying Reports and Authors
15.2) Where to Publish
15.3) Publishing Data
15.4) The FAIR Principles
15.5) Publishing Software
15.6) Summary
15.7) Exercises
15.8) Key Points
16) Finale
A) License
B) Code of Conduct
C) Contributing
D) Glossary
E) Setting Up
F) Learning Objectives
G) Key Points
H) Solutions
I) YAML
J) Working Remotely
References